

Amazon Simple Storage Service



Amazon Simple Storage Service: Developer Guide

Table of Contents

What is Amazon S3?	1
API call recommendations	2
Making direct REST API calls	2
Permissions	3
Developing with Amazon S3	4
Making requests	4
About access keys	5
Request endpoints	7
Making requests over IPv6	7
Making requests using the Amazon SDKs	17
Making requests using the REST API	56
Using the Amazon CLI	68
Learn more about the Amazon CLI	69
Developing with Amazon SDKs	69
SDK Programming interfaces	70
Specifying the Signature Version in request authentication	70
Get Amazon S3 request IDs for Amazon Web Services Support	79
Using the Amazon CLI to obtain request IDs	80
Using Windows PowerShell to obtain request IDs	80
Using Amazon CloudTrail data events to obtain request IDs	80
Using S3 server access logging to obtain request IDs	80
Using HTTP to obtain request IDs	80
Using a web browser to obtain request IDs	81
Using the Amazon SDKs to obtain request IDs	81
Supported S3 object-level API operations for S3 Tables	86
Code examples	92
Amazon S3	98
Basics	103
Scenarios	792
Serverless examples	1243
Amazon S3 Control	1255
Basics	1256
S3 Directory Buckets	1328
Basics	1330

Scenarios	1454
Authenticating Requests (Amazon Signature Version 4)	1458
Authentication Methods	1459
Introduction to Signing Requests	1460
Using an Authorization Header	1461
Overview	1462
Signature Calculation: Transfer Payload in a Single Chunk	1466
Signature Calculation: Transfer Payload in Multiple Chunks	1485
Signature Calculation: Including Trailing Headers	1498
Using Query Parameters	1504
Calculating a Signature	1508
An Example	1512
Example 2	1514
Examples: Signature Calculations	1514
Signature Calculation Examples Using Java	1515
Signature Calculation Examples Using C#	1516
Authenticating HTTP POST Requests	1517
Calculating a Signature	1520
Amazon S3 Signature Version 4 Authentication Specific Policy Keys	1521
Bucket policy examples using Signature Version 4 related condition keys	1524
Browser-Based Uploads Using POST	1528
POST Object	1529
Description	1529
Versioning	1530
Requests	1530
Examples	1548
Related Resources	1549
POST Object restore	1550
Description	1550
Querying Archives with Select Requests	1550
Restoring Archives	1552
Requests	1553
Responses	1568
Examples	1570
More Info	1572
Browser-Based Uploads Using HTTP POST	1572

Calculating a Signature	1574
Creating HTML Forms	1575
HTML Form Declaration	1576
HTML Form Fields	1577
POST Policy	1583
Expiration	1584
Condition Matching	1584
Conditions	1586
Character Escaping	1591
POST Upload Example	1592
Uploading a File to Amazon S3 Using HTTP POST	1592
Browser-based uploads using Amazon Amplify	1595
Using the Amazon Amplify JavaScript library to Upload Files to Amazon S3	1595
More Info	1596
Common request headers	1597
Common response headers	1601
Error responses	1606
REST error responses	1607
List of error codes	1608
List of SELECT Object Content Error Codes	1633
List of Replication-related error codes	1646
List of Tagging-related error codes	1649
List of Amazon S3 on Outposts error codes	1650
List of Amazon S3 Storage Lens error codes	1651
List of Amazon S3 Object Lambda error codes	1659
List of Amazon S3 asynchronous error codes	1663
List of Amazon S3 Access Grants error codes	1665
List of Amazon FSx-related error codes	1668
List of Amazon S3 Tables error codes	1669
Amazon S3 error best practices	1670
Retry InternalErrors	1670
Tune application for repeated SlowDown errors	1670
Isolate errors	1671
Amazon Glossary	1672
Resources	1673
Appendix	1675

Appendix: SelectObjectContent Response	1676
Description	1676
Responses	1676
Related Resources	1686
Appendix: OPTIONS object	1688
Description	1688
Requests	1688
Responses	1689
Examples	1691
Related Resources	1691
Appendix: SOAP API	1692
Operations on the Service (SOAP API)	1692
Operations on Buckets (SOAP API)	1694
Operations on Objects (SOAP API)	1707
Authenticating SOAP requests	1732
Setting access policy with SOAP	1733
Common elements	1735
SOAP Error Responses	1735
Appendix: Authenticating requests (Amazon signature version 2)	1737
Authenticating requests using the REST API (Amazon signature version 2)	1739
Signing and authenticating REST requests (Amazon signature version 2)	1742
Browser-based uploads using POST (Amazon signature version 2)	1756
Appendix: Lifecycle Configuration APIs (Deprecated)	1781
PUT Bucket lifecycle (Deprecated)	1782
GET Bucket lifecycle (Deprecated)	1798
Document history	1809

What is Amazon S3?

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.

This guide provides developer-focused information about using the Amazon S3 REST API to build applications that store and retrieve data in the cloud.

You can use any toolkit that supports HTTP to use the REST API. You can even use a browser to fetch objects, as long as they are anonymously readable.

The REST API uses the standard HTTP headers and status codes, so that standard browsers and toolkits work as expected. In some areas, we have added functionality to HTTP (for example, we added headers to support access control). In these cases, we have done our best to add the new functionality in a way that matched the style of standard HTTP usage.

Note

Support for SOAP over HTTP is deprecated, but it is still available over HTTPS. However, new Amazon S3 features will not be supported for SOAP. We recommend that you use either this REST API or the Amazon SDKs at the following link:

<https://www.amazonaws.cn/en/tools/>

This developer guide includes:

- [Common request headers](#) and [Common response headers](#)
- [Error responses](#)
- [Browser-Based Uploads Using POST \(Amazon Signature Version 4\)](#)

⚠ Important

Read the following about authentication and access control before going to specific API topics.

Requests to Amazon S3 can be authenticated or anonymous. Authenticated access requires credentials that Amazon can use to authenticate your requests.

API call recommendations

Making REST API calls directly from your code can be cumbersome. It requires you to write the necessary code to calculate a valid signature to authenticate your requests. We recommend the following alternatives instead:

- Use the [Amazon SDKs](#) to send your requests.

Also, see the [Sample Code and Libraries](#).

If you use the SDKs, you don't need to write code to calculate a signature for request authentication because the SDK clients authenticate your requests by using access keys that you provide. Unless you have a good reason not to, you should always use the Amazon SDKs.

- Use the Amazon CLI to make Amazon S3 API calls. For information about setting up the Amazon CLI and example Amazon S3 commands see the following topics:

[Set Up the Amazon CLI](#) in the *Amazon Simple Storage Service User Guide*.

[Using Amazon S3 with the Amazon Command Line Interface](#) in the *Amazon Command Line Interface User Guide*.

Making direct REST API calls

ℹ Note

The PUT request header is limited to 8 KB in size. Within the PUT request header, the system-defined metadata is limited to 2 KB in size. The size of system-defined metadata is

measured by taking the sum of the number of bytes in the US-ASCII encoding of each key and value.

If you'd like to make your own REST API calls instead of using one of the above alternatives, there are some things to keep in mind.

- To make direct REST API calls from your code, create a signature using valid credentials and include the signature in your request. For information about various authentication methods and signature calculations, see [Authenticating Requests \(Amazon Signature Version 4\)](#).
- The REST API uses standard HTTP headers and status codes, so standard browsers and toolkits work as expected. In some areas, we have added functionality to HTTP (for example, we added headers to support access control). In these cases, we have done our best to add the new functionality in a way that matches the style of standard HTTP usage. For more information about making requests, see [Making requests](#).

Permissions

You can have valid credentials to authenticate your requests, but unless you have S3 permissions from the account owner or bucket owner you cannot create or access Amazon S3 resources. These permissions are typically granted through an Amazon Identity and Access Management (IAM) [policy](#), such as a bucket policy. For example, you must have permissions to create an S3 bucket or get an object in a bucket. For a complete list of S3 permissions, see [Actions, resources, and condition keys for Amazon S3](#).

For more information about the permissions to S3 API operations by S3 resource types, see [Required permissions for Amazon S3 API operations](#) in the *Amazon Simple Storage Service User Guide*.

If you use the root user credentials of your Amazon Web Services account, you have all the permissions. However, using root user credentials is not recommended. Instead, we recommend that you create Amazon Identity and Access Management (IAM) roles in your account and manage user permissions. For more information, see [Access Management](#) in the *Amazon Simple Storage Service User Guide*.

Developing with Amazon S3

This section covers developer-related topics for using Amazon S3. For more information, review the topics below.

Topics

- [Making requests](#)
- [Developing with Amazon S3 using the Amazon CLI](#)
- [Developing with Amazon S3 using the Amazon SDKs](#)
- [Getting Amazon S3 request IDs for Amazon Web Services Support](#)
- [Supported Amazon S3 object-level API operations for S3 Tables](#)

Making requests

Amazon S3 is a REST service. You can send requests to Amazon S3 using the REST API.

Every interaction with Amazon S3 is either authenticated or anonymous. Authentication is a process of verifying the identity of the requester trying to access an Amazon Web Services (Amazon) product. Authenticated requests must include a signature value that authenticates the request sender. The signature value is, in part, generated from the requester's Amazon access keys (access key ID and secret access key). For more information about getting access keys, see [How Do I Get Security Credentials?](#) in the *Amazon Web Services General Reference*.

If you are using the Amazon SDK, the libraries compute the signature from the keys you provide. However, if you make direct REST API calls in your application, you must write the code to compute the signature and add it to the request.

Topics

- [About access keys](#)
- [Request endpoints](#)
- [Making requests to Amazon S3 over IPv6](#)
- [Making requests using the Amazon SDKs](#)
- [Making requests using the REST API](#)

About access keys

The following sections review the types of access keys that you can use to make authenticated requests.

Amazon Web Services account access keys

The account access keys provide full access to the Amazon resources owned by the account. The following are examples of access keys:

- Access key ID (a 20-character, alphanumeric string). For example: AKIAIOSFODNN7EXAMPLE
- Secret access key (a 40-character string). For example: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

The access key ID uniquely identifies an Amazon Web Services account. You can use these access keys to send authenticated requests to Amazon S3.

IAM user access keys

You can create one Amazon Web Services account for your company; however, there may be several employees in the organization who need access to your organization's Amazon resources. Sharing your Amazon Web Services account access keys reduces security, and creating individual Amazon Web Services accounts for each employee might not be practical. Also, you cannot easily share resources such as buckets and objects because they are owned by different accounts. To share resources, you must grant permissions, which is additional work.

In such scenarios, you can use Amazon Identity and Access Management (IAM) to create users under your Amazon Web Services account with their own access keys and attach IAM user policies that grant appropriate resource access permissions to these users. To better manage these users, IAM enables you to create groups of users and grant group-level permissions that apply to all users in that group.

These users are referred to as IAM users that you create and manage within Amazon. The parent account controls a user's ability to access Amazon. Any resources an IAM user creates are under the control of and paid for by the parent Amazon Web Services account. These IAM users can send authenticated requests to Amazon S3 using their own security credentials. For more information about creating and managing users under your Amazon Web Services account, go to the [Amazon Identity and Access Management product details page](#).

Temporary security credentials

In addition to creating IAM users with their own access keys, IAM also lets you grant temporary security credentials (temporary access keys and a security token) to any IAM user so they can access your Amazon services and resources. You can also manage users in your system outside Amazon. These are referred to as federated users. Additionally, users can be applications that you create to access your Amazon resources.

IAM provides the Amazon Security Token Service API for you to request temporary security credentials. You can use either the Amazon STS API or the Amazon SDK to request these credentials. The API returns temporary security credentials (access key ID and secret access key), and a security token. These credentials are valid only for the duration you specify when you request them. You use the access key ID and secret key the same way you use them when sending requests using your Amazon Web Services account or IAM user access keys. In addition, you must include the token in each request you send to Amazon S3.

An IAM user can request these temporary security credentials for their own use or hand them out to federated users or applications. When requesting temporary security credentials for federated users, you must provide a user name and an IAM policy defining the permissions you want to associate with these temporary security credentials. The federated user cannot get more permissions than the parent IAM user who requested the temporary credentials.

You can use these temporary security credentials in making requests to Amazon S3. The API libraries compute the necessary signature value using those credentials to authenticate your request. If you send requests using expired credentials, Amazon S3 denies the request.

For information on signing requests using temporary security credentials in your REST API requests, see [Signing and authenticating REST requests \(Amazon signature version 2\)](#). For information about sending requests using Amazon SDKs, see [Making requests using the Amazon SDKs](#).

For more information about IAM support for temporary security credentials, see [Temporary Security Credentials](#) in the *IAM User Guide*.

For added security, you can require multifactor authentication (MFA) when accessing your Amazon S3 resources by configuring a bucket policy. For information, see [Example bucket policies: Requiring MFA](#). After you require MFA to access your Amazon S3 resources, the only way you can access these resources is by providing temporary credentials that are created with an MFA key. For more information, see the [Amazon Multi-Factor Authentication](#) detail page and [Configuring MFA-Protected API Access](#) in the *IAM User Guide*.

Request endpoints

You send REST requests to the service's predefined endpoint. For a list of all Amazon services and their corresponding endpoints, go to [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

Making requests to Amazon S3 over IPv6

Amazon Simple Storage Service (Amazon S3) supports the ability to access S3 buckets using the Internet Protocol version 6 (IPv6), in addition to the IPv4 protocol. Amazon S3 dual-stack endpoints support requests to S3 buckets over IPv6 and IPv4. There are no additional charges for accessing Amazon S3 over IPv6. For more information about pricing, see [Amazon S3 Pricing](#).

Note

The following information is for public Amazon S3 IPv6 endpoints. For information about Amazon Virtual Private Cloud (VPC) endpoints, refer to [Amazon PrivateLink for Amazon S3](#).

Topics

- [Getting started making requests over IPv6](#)
- [Using IPv6 addresses in IAM policies](#)
- [Testing IP address compatibility](#)
- [Using Amazon S3 dual-stack endpoints](#)

Getting started making requests over IPv6

To make a request to an S3 bucket over IPv6, you need to use a dual-stack endpoint. The next section describes how to make requests over IPv6 by using dual-stack endpoints.

The following are some things you should know before trying to access a bucket over IPv6:

- The client and the network accessing the bucket must be enabled to use IPv6.
- Both virtual hosted-style and path style requests are supported for IPv6 access. For more information, see [Amazon S3 dual-stack endpoints](#).

- If you use source IP address filtering in your Amazon Identity and Access Management (IAM) user or bucket policies, you need to update the policies to include IPv6 address ranges. For more information, see [Using IPv6 addresses in IAM policies](#).
- When using IPv6, server access log files output IP addresses in an IPv6 format. You need to update existing tools, scripts, and software that you use to parse Amazon S3 log files so that they can parse the IPv6 formatted Remote IP addresses. For more information, see [Logging requests with server access logging](#).

Note

If you experience issues related to the presence of IPv6 addresses in log files, contact [Amazon Web Services Support](#).

Making requests over IPv6 by using dual-stack endpoints

You make requests with Amazon S3 API calls over IPv6 by using dual-stack endpoints. The Amazon S3 API operations work the same way whether you're accessing Amazon S3 over IPv6 or over IPv4. Performance should be the same too.

When using the REST API, you access a dual-stack endpoint directly. For more information, see [Dual-stack endpoints](#).

When using the Amazon Command Line Interface (Amazon CLI) and Amazon SDKs, you can use a parameter or flag to change to a dual-stack endpoint. You can also specify the dual-stack endpoint directly as an override of the Amazon S3 endpoint in the config file.

You can use a dual-stack endpoint to access a bucket over IPv6 from any of the following:

- The Amazon CLI, see [Using dual-stack endpoints from the Amazon CLI](#).
- The Amazon SDKs, see [Using dual-stack endpoints from the Amazon SDKs](#).
- The REST API, see [Making requests to dual-stack endpoints by using the REST API](#).

Features not available over IPv6

The following feature is currently not supported when accessing an S3 bucket over IPv6: Static website hosting from an S3 bucket.

Using IPv6 addresses in IAM policies

Before trying to access a bucket using IPv6, you must ensure that any IAM user or S3 bucket policies that are used for IP address filtering are updated to include IPv6 address ranges. IP address filtering policies that are not updated to handle IPv6 addresses may result in clients incorrectly losing or gaining access to the bucket when they start using IPv6. For more information about managing access permissions with IAM, see [Identity and Access Management for Amazon S3](#).

IAM policies that filter IP addresses use [IP Address Condition Operators](#). The following bucket policy identifies the 54.240.143.* range of allowed IPv4 addresses by using IP address condition operators. Any IP addresses outside of this range will be denied access to the bucket (examplebucket). Since all IPv6 addresses are outside of the allowed range, this policy prevents IPv6 addresses from being able to access examplebucket.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws-cn:s3:::examplebucket/*",
      "Condition": {
        "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

You can modify the bucket policy's Condition element to allow both IPv4 (54.240.143.0/24) and IPv6 (2001:DB8:1234:5678::/64) address ranges as shown in the following example. You can use the same type of Condition block shown in the example to update both your IAM user and bucket policies.

```
"Condition": {
  "IpAddress": {
```

```
    "aws:SourceIp": [
      "54.240.143.0/24",
      "2001:DB8:1234:5678::/64"
    ]
  }
}
```

Before using IPv6 you must update all relevant IAM user and bucket policies that use IP address filtering. We do not recommend using IP address filterig in bucket policies.

You can review your IAM user policies using the IAM console at <https://console.amazonaws.cn/iam/>. For more information about IAM, see the [IAM User Guide](#). For information about editing S3 bucket policies, see [Adding a bucket policy](#).

Testing IP address compatibility

If you are using use Linux/Unix or Mac OS X, you can test whether you can access a dual-stack endpoint over IPv6 by using the `curl` command as shown in the following example:

Example

```
curl -v http://s3.dualstack.us-west-2.amazonaws.com/
```

You get back information similar to the following example. If you are connected over IPv6 the connected IP address will be an IPv6 address.

```
* About to connect() to s3-us-west-2.amazonaws.com port 80 (#0)
* Trying IPv6 address... connected
* Connected to s3.dualstack.us-west-2.amazonaws.com (IPv6 address) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1 OpenSSL/1.0.1t
zlib/1.2.3
> Host: s3.dualstack.us-west-2.amazonaws.com
```

If you are using Microsoft 7 or 10, you can test whether you can access a dual-stack endpoint over IPv6 or IPv4 by using the `ping` command as shown in the following example.

```
ping ipv6.s3.dualstack.us-west-2.amazonaws.com
```

Using Amazon S3 dual-stack endpoints

Amazon S3 dual-stack endpoints support requests to S3 buckets over IPv6 and IPv4. This section describes how to use dual-stack endpoints.

Topics

- [Amazon S3 dual-stack endpoints](#)
- [Using dual-stack endpoints from the Amazon CLI](#)
- [Using dual-stack endpoints from the Amazon SDKs](#)
- [Using dual-stack endpoints from the REST API](#)

Amazon S3 dual-stack endpoints

When you make a request to a dual-stack endpoint, the bucket URL resolves to an IPv6 or an IPv4 address. For more information about accessing a bucket over IPv6, see [Making requests to Amazon S3 over IPv6](#).

When using the REST API, you directly access an Amazon S3 endpoint by using the endpoint name (URI). You can access an S3 bucket through a dual-stack endpoint by using a virtual hosted-style or a path-style endpoint name. Amazon S3 supports only regional dual-stack endpoint names, which means that you must specify the region as part of the name.

Use the following naming conventions for the dual-stack virtual hosted-style and path-style endpoint names:

- Virtual hosted-style dual-stack endpoint:

bucketname.s3.dualstack.*aws-region*.amazonaws.com

- Path-style dual-stack endpoint:

s3.dualstack.*aws-region*.amazonaws.com/*bucketname*

For more information, about endpoint name style, see [Accessing and listing an Amazon S3 bucket](#) . For a list of Amazon S3 endpoints, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

⚠ Important

You can use transfer acceleration with dual-stack endpoints. For more information, see [Getting started with Amazon S3 Transfer Acceleration](#).

ℹ Note

The two types of Virtual Private Cloud (VPC) endpoints that access Amazon S3 (*Interface VPC endpoints* and *Gateway VPC endpoints*) now have dual-stack support. For more information about VPC endpoints for Amazon S3, see [Amazon PrivateLink for Amazon S3](#).

When using the Amazon Command Line Interface (Amazon CLI) and Amazon SDKs, you can use a parameter or flag to change to a dual-stack endpoint. You can also specify the dual-stack endpoint directly as an override of the Amazon S3 endpoint in the config file. The following sections describe how to use dual-stack endpoints from the Amazon CLI and the Amazon SDKs.

Using dual-stack endpoints from the Amazon CLI

This section provides examples of Amazon CLI commands used to make requests to a dual-stack endpoint. For instructions on setting up the Amazon CLI, see [Developing with Amazon S3 using the Amazon CLI](#).

You set the configuration value `use_dualstack_endpoint` to `true` in a profile in your Amazon Config file to direct all Amazon S3 requests made by the `s3` and `s3api` Amazon CLI commands to the dual-stack endpoint for the specified region. You specify the region in the config file or in a command using the `--region` option.

When using dual-stack endpoints with the Amazon CLI, both `path` and `virtual` addressing styles are supported. The addressing style, set in the config file, controls if the bucket name is in the hostname or part of the URL. By default, the CLI will attempt to use virtual style where possible, but will fall back to path style if necessary. For more information, see [Amazon CLI Amazon S3 Configuration](#).

You can also make configuration changes by using a command, as shown in the following example, which sets `use_dualstack_endpoint` to `true` and `addressing_style` to `virtual` in the default profile.

```
$ aws configure set default.s3.use_dualstack_endpoint true
$ aws configure set default.s3.addressing_style virtual
```

If you want to use a dual-stack endpoint for specified Amazon CLI commands only (not all commands), you can use either of the following methods:

- You can use the dual-stack endpoint per command by setting the `--endpoint-url` parameter to `https://s3.dualstack.aws-region.amazonaws.com` or `http://s3.dualstack.aws-region.amazonaws.com` for any `s3` or `s3api` command.

```
$ aws s3api list-objects --bucket bucketname --endpoint-url https://s3.dualstack.aws-region.amazonaws.com
```

- You can set up separate profiles in your Amazon Config file. For example, create one profile that sets `use_dualstack_endpoint` to `true` and a profile that does not set `use_dualstack_endpoint`. When you run a command, specify which profile you want to use, depending upon whether or not you want to use the dual-stack endpoint.

Note

When using the Amazon CLI you currently cannot use transfer acceleration with dual-stack endpoints. However, support for the Amazon CLI is coming soon. For more information, see [Enabling and using S3 Transfer Acceleration](#).

Using dual-stack endpoints from the Amazon SDKs

This section provides examples of how to access a dual-stack endpoint by using the Amazon SDKs.

Amazon SDK for Java dual-stack endpoint example

The following example shows how to enable dual-stack endpoints when creating an Amazon S3 client using the Amazon SDK for Java.

For instructions on creating and testing a working Java sample, see [Getting Started](#) in the Amazon SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
```

```
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

public class DualStackEndpoints {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "**** Bucket name ****";

        try {
            // Create an Amazon S3 client with dual-stack endpoints enabled.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .withDualstackEnabled(true)
                .build();

            s3Client.listObjects(bucketName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

If you are using the Amazon SDK for Java on Windows, you might have to set the following Java virtual machine (JVM) property:

```
java.net.preferIPv6Addresses=true
```

Amazon .NET SDK dual-stack endpoint example

When using the Amazon SDK for .NET you use the `AmazonS3Config` class to enable the use of a dual-stack endpoint as shown in the following example.


```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DualStackEndpointTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            var config = new AmazonS3Config
            {
                UseDualstackEndpoint = true,
                RegionEndpoint = bucketRegion
            };
            client = new AmazonS3Client(config);
            Console.WriteLine("Listing objects stored in a bucket");
            ListingObjectsAsync().Wait();
        }

        private static async Task ListingObjectsAsync()
        {
            try
            {
                var request = new ListObjectsV2Request
                {
                    BucketName = bucketName,
                    MaxKeys = 10
                };
                ListObjectsV2Response response;
                do
                {
                    response = await client.ListObjectsV2Async(request);

                    // Process the response.
                    foreach (S3Object entry in response.S3Objects)
                }
            }
        }
    }
}
```

```
        {
            Console.WriteLine("key = {0} size = {1}",
                              entry.Key, entry.Size);
        }
        Console.WriteLine("Next Continuation Token: {0}",
response.NextContinuationToken);
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated == true);
}
catch (AmazonS3Exception amazonS3Exception)
{
    Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
amazonS3Exception.ToString());
}
catch (Exception e)
{
    Console.WriteLine("Exception: " + e.ToString());
}
}
}
```

For information about setting up and running the code examples, see [Getting Started with the Amazon SDK for .NET](#) in the *Amazon SDK for .NET Developer Guide*.

Using dual-stack endpoints from the REST API

For information about making requests to dual-stack endpoints by using the REST API, see [Making requests to dual-stack endpoints by using the REST API](#).

Making requests using the Amazon SDKs

Topics

- [Making requests using Amazon Web Services account or IAM user credentials](#)
- [Making requests using IAM user temporary credentials](#)
- [Making requests using federated user temporary credentials](#)

You can send authenticated requests to Amazon S3 using either the Amazon SDK or by making the REST API calls directly in your application. The Amazon SDK API uses the credentials that you provide to compute the signature for authentication. If you use the REST API directly in your applications, you must write the necessary code to compute the signature for authenticating your request. For a list of available Amazon SDKs go to, [Sample Code and Libraries](#).

Making requests using Amazon Web Services account or IAM user credentials

You can use your Amazon Web Services account or IAM user security credentials to send authenticated requests to Amazon S3. This section provides examples of how you can send authenticated requests using the Amazon SDK for Java, Amazon SDK for .NET, and Amazon SDK for PHP. For a list of available Amazon SDKs, go to [Sample Code and Libraries](#).

Each of these Amazon SDKs uses an SDK-specific credentials provider chain to find and use credentials and perform actions on behalf of the credentials owner. What all these credentials provider chains have in common is that they all look for your local Amazon credentials file.

For more information, see the topics below:

Topics

- [To create a local Amazon credentials file](#)
- [Sending authenticated requests using the Amazon SDKs](#)
- [Related resources](#)

To create a local Amazon credentials file

The easiest way to configure credentials for your Amazon SDKs is to use an Amazon credentials file. If you use the Amazon Command Line Interface (Amazon CLI), you may already have a local Amazon credentials file configured. Otherwise, use the following procedure to set up a credentials file:

1. Sign in to the Amazon Web Services Management Console and open the IAM console at <https://console.amazonaws.cn/iam/>.
2. Create a new user with permissions limited to the services and actions that you want your code to have access to. For more information about creating a new user, see [Creating IAM users \(Console\)](#), and follow the instructions through step 8.
3. Choose **Download .csv** to save a local copy of your Amazon credentials.
4. On your computer, navigate to your home directory, and create an `.aws` directory. On Unix-based systems, such as Linux or OS X, this is in the following location:

```
~/ .aws
```

On Windows, this is in the following location:

```
%HOMEPATH%\ .aws
```

5. In the `.aws` directory, create a new file named `credentials`.
6. Open the `credentials .csv` file that you downloaded from the IAM console, and copy its contents into the `credentials` file using the following format:

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

7. Save the `credentials` file, and delete the `.csv` file that you downloaded in step 3.

Your shared credentials file is now configured on your local computer, and it's ready to be used with the Amazon SDKs.

Sending authenticated requests using the Amazon SDKs

Use the Amazon SDKs to send authenticated requests. For more information about sending authenticated requests, see [Amazon security credentials](#) or [IAM Identity Center Authentication](#).

Java

For information about authenticating requests using the Amazon SDK for Java, see [Using shared config and credentials files to globally configure Amazon SDKs and tools](#) and

[Authentication and access using Amazon SDKs and tools](#) in the Amazon SDKs and Tools Reference Guide.

.NET

To send authenticated requests using your Amazon Web Services account or IAM user credentials:

- Create an instance of the `AmazonS3Client` class.
- Run one of the `AmazonS3Client` methods to send requests to Amazon S3. The client generates the necessary signature from the credentials that you provide and includes it in the request it sends to Amazon S3.

For more information, see [Making requests using Amazon account or IAM user credentials](#) >.

Note

- You can create the `AmazonS3Client` client without providing your security credentials. Requests sent using this client are anonymous requests, without a signature. Amazon S3 returns an error if you send anonymous requests for a resource that is not publicly available.
- You can create an Amazon Web Services account and create the required users. You can also manage credentials for those users. You need these credentials to perform the task in the following example. For more information, see [Configure Amazon credentials](#) in the *Amazon SDK for .NET Developer Guide*.

You can then also configure your application to actively retrieve profiles and credentials, and then explicitly use those credentials when creating an Amazon service client. For more information, see [Accessing credentials and profiles in an application](#) in the *Amazon SDK for .NET Developer Guide*.

The following C# example shows how to perform the preceding tasks. For information about setting up and running the code examples, see [Getting Started with the Amazon SDK for .NET](#) in the *Amazon SDK for .NET Developer Guide*.

Example

```
using Amazon;
```

```
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class MakeS3RequestTest
    {
        private const string bucketName = "**** bucket name ****";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            using (client = new AmazonS3Client(bucketRegion))
            {
                Console.WriteLine("Listing objects stored in a bucket");
                ListingObjectsAsync().Wait();
            }
        }

        static async Task ListingObjectsAsync()
        {
            try
            {
                ListObjectsRequest request = new ListObjectsRequest
                {
                    BucketName = bucketName,
                    MaxKeys = 2
                };
                do
                {
                    ListObjectsResponse response = await
client.ListObjectsAsync(request);
                    // Process the response.
                    foreach (S3Object entry in response.S3Objects)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }
                }
            }
        }
    }
}
```

```

        // If the response is truncated, set the marker to get the next
        // set of keys.
        if (response.IsTruncated)
        {
            request.Marker = response.NextMarker;
        }
        else
        {
            request = null;
        }
    } while (request != null);
}
catch (AmazonS3Exception e)
{
    Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
}
catch (Exception e)
{
    Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
}
}
}
}

```

PHP

This section explains how to use a class from version 3 of the Amazon SDK for PHP to send authenticated requests using your Amazon Web Services account or IAM user credentials. For more information about the Amazon SDK for Ruby API, go to [Amazon SDK for Ruby - Version 2](#).

The following PHP example shows how the client makes a request using your security credentials to list all of the buckets for your account.

Example

```

require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

```

```
$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
]);

// Retrieve the list of buckets.
$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // Print the list of objects to the page.
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Note

You can create the `S3Client` client without providing your security credentials. Requests sent using this client are anonymous requests, without a signature. Amazon S3 returns an error if you send anonymous requests for a resource that is not publicly available. For more information, see [Creating Anonymous Clients](#) in the [Amazon SDK for PHP Documentation](#).

Ruby

Before you can use version 3 of the Amazon SDK for Ruby to make calls to Amazon S3, you must set the Amazon access credentials that the SDK uses to verify your access to your buckets and objects. If you have shared credentials set up in the Amazon credentials profile on your local system, version 3 of the SDK for Ruby can use those credentials without your having

to declare them in your code. For more information about setting up shared credentials, see [Making requests using Amazon account or IAM user credentials](#).

The following Ruby code snippet uses the credentials in a shared Amazon credentials file on a local computer to authenticate a request to get all of the object key names in a specific bucket. It does the following:

1. Creates an instance of the `Aws::S3::Client` class.
2. Makes a request to Amazon S3 by enumerating objects in a bucket using the `list_objects_v2` method of `Aws::S3::Client`. The client generates the necessary signature value from the credentials in the Amazon credentials file on your computer, and includes it in the request it sends to Amazon S3.
3. Prints the array of object key names to the terminal.

Example

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'amzn-s3-demo-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts 'The object keys in this bucket are (first 50 objects):'
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts 'No objects found in the bucket.'
```

```

    puts 'No objects found in this bucket.'
  end

  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  false
end

# Example usage:
def run_me
  region = 'us-west-2'
  bucket_name = 'BUCKET_NAME'
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__

```

If you don't have a local Amazon credentials file, you can still create the `Aws::S3::Client` resource and run code against Amazon S3 buckets and objects. Requests that are sent using version 3 of the SDK for Ruby are anonymous, with no signature by default. Amazon S3 returns an error if you send anonymous requests for a resource that's not publicly available.

You can use and expand the previous code snippet for SDK for Ruby applications, as in the following more robust example.

```

# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'amzn-s3-demo-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."

```

```

objects = s3_client.list_objects_v2(
  bucket: bucket_name,
  max_keys: 50
)

if objects.count.positive?
  puts 'The object keys in this bucket are (first 50 objects):'
  objects.contents.each do |object|
    puts object.key
  end
else
  puts 'No objects found in this bucket.'
end

true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  false
end

# Example usage:
def run_me
  region = 'us-west-2'
  bucket_name = 'BUCKET_NAME'
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__

```

Go

Example

The following example uses Amazon credentials automatically loaded by the SDK for Go from the shared credentials file.

```

package main

import (
  "context"

```

```
"errors"
"fmt"

"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    s3Client := s3.NewFromConfig(sdkConfig)
    count := 10
    fmt.Printf("Let's list up to %v buckets for your account.\n", count)
    result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
    if err != nil {
        var ae smithy.APIError
        if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
            fmt.Println("You don't have permission to list buckets for this account.")
        } else {
            fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
        }
        return
    }
    if len(result.Buckets) == 0 {
        fmt.Println("You don't have any buckets!")
    } else {
        if count > len(result.Buckets) {
            count = len(result.Buckets)
        }
        for _, bucket := range result.Buckets[:count] {
            fmt.Printf("\t\t%v\n", *bucket.Name)
        }
    }
}
```

```
}
```

Related resources

- [Developing with Amazon S3 using the Amazon SDKs](#)
- [Amazon SDK for PHP for Amazon S3 Aws\S3\S3Client Class](#)
- [Amazon SDK for PHP Documentation](#)

Making requests using IAM user temporary credentials

An Amazon Web Services account or an IAM user can request temporary security credentials and use them to send authenticated requests to Amazon S3. This section provides examples of how to use the Amazon SDK for Java, .NET, and PHP to obtain temporary security credentials and use them to authenticate your requests to Amazon S3.

Java

An IAM user or an Amazon Web Services account can request temporary security credentials (see [Making requests](#)) using the Amazon SDK for Java and use them to access Amazon S3. These credentials expire after the specified session duration.

By default, the session duration is one hour. If you use IAM user credentials, you can specify the duration when requesting the temporary security credentials from 15 minutes to the maximum session duration for the role. For more information about temporary security credentials, see [Temporary Security Credentials](#) in the *IAM User Guide*. For more information about making requests, see [Making requests](#).

To get temporary security credentials and access Amazon S3

1. Create an instance of the `AWSSecurityTokenService` class.
2. Retrieve the temporary security credentials for the desired role by calling the `assumeRole()` method of the Security Token Service (STS) client.
3. Package the temporary security credentials into a `BasicSessionCredentials` object. You use this object to provide the temporary security credentials to your Amazon S3 client.
4. Create an instance of the `AmazonS3Client` class using the temporary security credentials. You send requests to Amazon S3 using this client. If you send requests using expired credentials, Amazon S3 will return an error.

Note

If you obtain temporary security credentials using your Amazon Web Services account security credentials, the temporary credentials are valid for only one hour. You can specify the session duration only if you use IAM user credentials to request a session.

The following example lists a set of object keys in the specified bucket. The example obtains temporary security credentials for a session and uses them to send an authenticated request to Amazon S3.

If you want to test the sample by using IAM user credentials, you must create an IAM user under your Amazon Web Services account. For more information about how to create an IAM user, see [Creating Your First IAM user and Administrators Group](#) in the *IAM User Guide*.

For instructions on creating and testing a working sample, see [Getting Started](#) in the Amazon SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.AssumeRoleRequest;
import com.amazonaws.services.securitytoken.model.AssumeRoleResult;
import com.amazonaws.services.securitytoken.model.Credentials;

public class MakingRequestsWithIAMTempCredentials {
    public static void main(String[] args) {
        String clientRegion = "**** Client region ****";
        String roleARN = "**** ARN for role to be assumed ****";
        String roleSessionName = "**** Role session name ****";
        String bucketName = "**** Bucket name ****";

        try {
            // Creating the STS client is part of your trusted code. It has
            // the security credentials you use to obtain temporary security
            credentials.
            AWSSecurityTokenService stsClient =
                AWSSecurityTokenServiceClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();
```

```

role of    // Obtain credentials for the IAM role. Note that you cannot assume the
           // an AWS root account;
           // Amazon S3 will deny access. You must use credentials for an IAM user
or an      // IAM role.
           AssumeRoleRequest roleRequest = new AssumeRoleRequest()
               .withRoleArn(roleARN)
               .withRoleSessionName(roleSessionName);
           AssumeRoleResult roleResponse = stsClient.assumeRole(roleRequest);
           Credentials sessionCredentials = roleResponse.getCredentials();

           // Create a BasicSessionCredentials object that contains the credentials
you         // just retrieved.
           BasicSessionCredentials awsCredentials = new BasicSessionCredentials(
               sessionCredentials.getAccessKeyId(),
               sessionCredentials.getSecretAccessKey(),
               sessionCredentials.getSessionToken());

           // Provide temporary security credentials so that the Amazon S3 client
           // can send authenticated requests to Amazon S3. You create the client
           // using the sessionCredentials object.
           AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
               .withCredentials(new
AWSStaticCredentialsProvider(awsCredentials))
               .withRegion(clientRegion)
               .build();

           // Verify that assuming the role worked and the permissions are set
correctly   // by getting a set of object keys from the bucket.
           ObjectListing objects = s3Client.listObjects(bucketName);
           System.out.println("No. of Objects: " +
objects.getObjectSummaries().size());
           } catch (AmazonServiceException e) {
               // The call was transmitted successfully, but Amazon S3 couldn't process
               // it, so it returned an error response.
               e.printStackTrace();
           } catch (SdkClientException e) {
               // Amazon S3 couldn't be contacted for a response, or the client
               // couldn't parse the response from Amazon S3.
               e.printStackTrace();
           }
}

```



```
}  
}
```

.NET

An IAM user or an Amazon Web Services account can request temporary security credentials using the Amazon SDK for .NET and use them to access Amazon S3. These credentials expire after the session duration.

By default, the session duration is one hour. If you use IAM user credentials, you can specify the duration when requesting the temporary security credentials from 15 minutes to the maximum session duration for the role. For more information about temporary security credentials, see [Temporary Security Credentials](#) in the *IAM User Guide*. For more information about making requests, see [Making requests](#).

To get temporary security credentials and access Amazon S3

1. Create an instance of the Amazon Security Token Service client, `AmazonSecurityTokenServiceClient`.
2. Start a session by calling the `GetSessionToken` method of the STS client you created in the preceding step. You provide session information to this method using a `GetSessionTokenRequest` object.

The method returns your temporary security credentials.

3. Package the temporary security credentials in an instance of the `SessionAWSCredentials` object. You use this object to provide the temporary security credentials to your Amazon S3 client.
4. Create an instance of the `AmazonS3Client` class by passing in the temporary security credentials. You send requests to Amazon S3 using this client. If you send requests using expired credentials, Amazon S3 returns an error.

Note

If you obtain temporary security credentials using your Amazon Web Services account security credentials, those credentials are valid for only one hour. You can specify a session duration only if you use IAM user credentials to request a session.

The following C# example lists object keys in the specified bucket. For illustration, the example obtains temporary security credentials for a default one-hour session and uses them to send authenticated request to Amazon S3.

If you want to test the sample by using IAM user credentials, you must create an IAM user under your Amazon Web Services account. For more information about how to create an IAM user, see [Creating Your First IAM user and Administrators Group](#) in the *IAM User Guide*. For more information about making requests, see [Making requests](#).

For information about setting up and running the code examples, see [Getting Started with the Amazon SDK for .NET](#) in the *Amazon SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempCredExplicitSessionStartTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
        RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
            try
            {
                // Credentials use the default AWS SDK for .NET credential search
                chain.
                // On local development machines, this is your default profile.
```

```
        Console.WriteLine("Listing objects stored in a bucket");
        SessionAWSCredentials tempCredentials = await
GetTemporaryCredentialsAsync();

        // Create a client by providing temporary security credentials.
        using (s3Client = new AmazonS3Client(tempCredentials, bucketRegion))
        {
            var listObjectRequest = new ListObjectsRequest
            {
                BucketName = bucketName
            };
            // Send request to Amazon S3.
            ListObjectsResponse response = await
s3Client.ListObjectsAsync(listObjectRequest);
            List<S3Object> objects = response.S3Objects;
            Console.WriteLine("Object count = {0}", objects.Count);
        }
    }
    catch (AmazonS3Exception s3Exception)
    {
        Console.WriteLine(s3Exception.Message, s3Exception.InnerException);
    }
    catch (AmazonSecurityTokenServiceException stsException)
    {
        Console.WriteLine(stsException.Message,
stsException.InnerException);
    }
}

private static async Task<SessionAWSCredentials>
GetTemporaryCredentialsAsync()
{
    using (var stsClient = new AmazonSecurityTokenServiceClient())
    {
        var getSessionTokenRequest = new GetSessionTokenRequest
        {
            DurationSeconds = 7200 // seconds
        };

        GetSessionTokenResponse sessionTokenResponse =
            await
stsClient.GetSessionTokenAsync(getSessionTokenRequest);

        Credentials credentials = sessionTokenResponse.Credentials;
    }
}
```

```
        var sessionCredentials =  
            new SessionAWSCredentials(credentials.AccessKeyId,  
                                       credentials.SecretAccessKey,  
                                       credentials.SessionToken);  
        return sessionCredentials;  
    }  
}  
}
```

PHP

For more information about the Amazon SDK for Ruby API, go to [Amazon SDK for Ruby - Version 2](#).

An IAM user or an Amazon Web Services account can request temporary security credentials using version 3 of the Amazon SDK for PHP. It can then use the temporary credentials to access Amazon S3. The credentials expire when the session duration expires.

By default, the session duration is one hour. If you use IAM user credentials, you can specify the duration when requesting the temporary security credentials from 15 minutes to the maximum session duration for the role. For more information about temporary security credentials, see [Temporary Security Credentials](#) in the *IAM User Guide*. For more information about making requests, see [Making requests](#).

Note

If you obtain temporary security credentials using your Amazon Web Services account security credentials, the temporary security credentials are valid for only one hour. You can specify the session duration only if you use IAM user credentials to request a session.

Example

The following PHP example lists object keys in the specified bucket using temporary security credentials. The example obtains temporary security credentials for a default one-hour session, and uses them to send authenticated request to Amazon S3. For more information about the Amazon SDK for Ruby API, go to [Amazon SDK for Ruby - Version 2](#).

If you want to test the example by using IAM user credentials, you must create an IAM user under your Amazon Web Services account. For information about how to create an IAM user, see [Creating Your First IAM user and Administrators Group](#) in the *IAM User Guide*. For examples of setting the session duration when using IAM user credentials to request a session, see [Making requests using IAM user temporary credentials](#).

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

$sessionToken = $sts->getSessionToken();

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);

$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // List objects
    foreach ($objects as $object) {
```

```
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

An IAM user or an Amazon Web Services account can request temporary security credentials using Amazon SDK for Ruby and use them to access Amazon S3. These credentials expire after the session duration.

By default, the session duration is one hour. If you use IAM user credentials, you can specify the duration when requesting the temporary security credentials from 15 minutes to the maximum session duration for the role. For more information about temporary security credentials, see [Temporary Security Credentials](#) in the *IAM User Guide*. For more information about making requests, see [Making requests](#).

Note

If you obtain temporary security credentials using your Amazon Web Services account security credentials, the temporary security credentials are valid for only one hour. You can specify session duration only if you use IAM user credentials to request a session.

The following Ruby example creates a temporary user to list the items in a specified bucket for one hour. To use this example, you must have Amazon credentials that have the necessary permissions to create new Amazon Security Token Service (Amazon STS) clients, and list Amazon S3 buckets.

```
# Prerequisites:
# - A user in AWS Identity and Access Management (IAM). This user must
#   be able to assume the following IAM role. You must run this code example
#   within the context of this user.
# - An existing role in IAM that allows all of the Amazon S3 actions for all of the
#   resources in this code example. This role must also trust the preceding IAM
#   user.
# - An existing S3 bucket.
```

```

require 'aws-sdk-core'
require 'aws-sdk-s3'
require 'aws-sdk-iam'

# Checks whether a user exists in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Boolean] true if the user exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless user_exists?(iam_client, 'my-user')
def user_exists?(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return true if response.user.user_name
rescue Aws::IAM::Errors::NoSuchEntity
  # User doesn't exist.
rescue StandardError => e
  puts 'Error while determining whether the user ' \
    "'#{user_name}' exists: #{e.message}"
end

# Creates a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS::IAM::Types::User] The new user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = create_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def create_user(iam_client, user_name)
  response = iam_client.create_user(user_name: user_name)
  response.user
rescue StandardError => e
  puts "Error while creating the user '#{user_name}': #{e.message}"
end

# Gets a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS::IAM::Types::User] The existing user.
# @example

```

```

#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def get_user(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  response.user
rescue StandardError => e
  puts "Error while getting the user '#{user_name}': #{e.message}"
end

# Checks whether a role exists in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The role's name.
# @return [Boolean] true if the role exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless role_exists?(iam_client, 'my-role')
def role_exists?(iam_client, role_name)
  response = iam_client.get_role(role_name: role_name)
  return true if response.role.role_name
rescue StandardError => e
  puts 'Error while determining whether the role ' \
    "'#{role_name}' exists: #{e.message}"
end

# Gets credentials for a role in IAM.
#
# @param sts_client [Aws::STS::Client] An initialized AWS STS client.
# @param role_arn [String] The role's Amazon Resource Name (ARN).
# @param role_session_name [String] A name for this role's session.
# @param duration_seconds [Integer] The number of seconds this session is valid.
# @return [AWS::AssumeRoleCredentials] The credentials.
# @example
#   sts_client = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_credentials(
#     sts_client,
#     'arn:aws:iam::123456789012:role/AmazonS3ReadOnly',
#     'ReadAmazonS3Bucket',
#     3600
#   )
#   exit 1 if credentials.nil?
def get_credentials(sts_client, role_arn, role_session_name, duration_seconds)
  Aws::AssumeRoleCredentials.new(

```



```

        client: sts_client,
        role_arn: role_arn,
        role_session_name: role_session_name,
        duration_seconds: duration_seconds
    )
rescue StandardError => e
    puts "Error while getting credentials: #{e.message}"
end

# Checks whether a bucket exists in Amazon S3.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The name of the bucket.
# @return [Boolean] true if the bucket exists; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless bucket_exists?(s3_client, 'amzn-s3-demo-bucket')
def bucket_exists?(s3_client, bucket_name)
    response = s3_client.list_buckets
    response.buckets.each do |bucket|
        return true if bucket.name == bucket_name
    end
rescue StandardError => e
    puts "Error while checking whether the bucket '#{bucket_name}' " \
        "exists: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'amzn-s3-demo-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
    puts "Accessing the contents of the bucket named '#{bucket_name}'..."
    response = s3_client.list_objects_v2(
        bucket: bucket_name,
        max_keys: 50
    )

    if response.count.positive?
        puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    end
end

```

```
puts 'Name => ETag'
response.contents.each do |obj|
  puts "#{obj.key} => #{obj.etag}"
end
else
  puts "No objects in the bucket named '#{bucket_name}'."
end
true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end
```

Making requests using federated user temporary credentials

You can request temporary security credentials and provide them to your federated users or applications who need to access your Amazon resources. This section provides examples of how you can use the Amazon SDK to obtain temporary security credentials for your federated users or applications and send authenticated requests to Amazon S3 using those credentials. For a list of available Amazon SDKs, see [Sample Code and Libraries](#).

Note

Both the Amazon Web Services account and an IAM user can request temporary security credentials for federated users. However, for added security, only an IAM user with the necessary permissions should request these temporary credentials to ensure that the federated user gets at most the permissions of the requesting IAM user. In some applications, you might find it suitable to create an IAM user with specific permissions for the sole purpose of granting temporary security credentials to your federated users and applications.

Java

You can provide temporary security credentials for your federated users and applications so that they can send authenticated requests to access your Amazon resources. When requesting these temporary credentials, you must provide a user name and an IAM policy that describes the resource permissions that you want to grant. By default, the session duration is one hour. You can explicitly set a different duration value when requesting the temporary security credentials for federated users and applications.

Note

For added security when requesting temporary security credentials for federated users and applications, we recommend that you use a dedicated IAM user with only the necessary access permissions. The temporary user you create can never get more permissions than the IAM user who requested the temporary security credentials. For more information, see [Amazon Identity and Access Management FAQs](#).

To provide security credentials and send authenticated request to access resources, do the following:

- Create an instance of the `AWSSecurityTokenServiceClient` class.
- Start a session by calling the `getFederationToken()` method of the Security Token Service (STS) client. Provide session information, including the user name and an IAM policy, that you want to attach to the temporary credentials. You can provide an optional session duration. This method returns your temporary security credentials.
- Package the temporary security credentials in an instance of the `BasicSessionCredentials` object. You use this object to provide the temporary security credentials to your Amazon S3 client.
- Create an instance of the `AmazonS3Client` class using the temporary security credentials. You send requests to Amazon S3 using this client. If you send requests using expired credentials, Amazon S3 returns an error.

Example

The example lists keys in the specified S3 bucket. In the example, you obtain temporary security credentials for a two-hour session for your federated user and use the credentials to send authenticated requests to Amazon S3. To run the example, you need to create an IAM user with an attached policy that allows the user to request temporary security credentials and list your Amazon resources. The following policy accomplishes this:

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

For more information about how to create an IAM user, see [Creating Your First IAM user and Administrators Group](#) in the *IAM User Guide*.

After creating an IAM user and attaching the preceding policy, you can run the following example. For instructions on creating and testing a working sample, see [Getting Started](#) in the Amazon SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.S3Actions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;

import java.io.IOException;

public class MakingRequestsWithFederatedTempCredentials {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "**** Specify bucket name ****";
        String federatedUser = "**** Federated user name ****";
        String resourceARN = "arn:aws:s3:::" + bucketName;

        try {
            AWSSecurityTokenService stsClient = AWSSecurityTokenServiceClientBuilder
                .standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();
```

```
        GetFederationTokenRequest getFederationTokenRequest = new
GetFederationTokenRequest();
        getFederationTokenRequest.setDurationSeconds(7200);
        getFederationTokenRequest.setName(federatedUser);

        // Define the policy and add it to the request.
        Policy policy = new Policy();
        policy.withStatements(new Statement(Effect.Allow)
            .withActions(S3Actions.ListObjects)
            .withResources(new Resource(resourceARN)));
        getFederationTokenRequest.setPolicy(policy.toJson());

        // Get the temporary security credentials.
        GetFederationTokenResult federationTokenResult =
stsClient.getFederationToken(getFederationTokenRequest);
        Credentials sessionCredentials = federationTokenResult.getCredentials();

        // Package the session credentials as a BasicSessionCredentials
        // object for an Amazon S3 client object to use.
        BasicSessionCredentials basicSessionCredentials = new
BasicSessionCredentials(
            sessionCredentials.getAccessKeyId(),
            sessionCredentials.getSecretAccessKey(),
            sessionCredentials.getSessionToken());
        AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
            .withCredentials(new
AWSStaticCredentialsProvider(basicSessionCredentials))
            .withRegion(clientRegion)
            .build();

        // To verify that the client works, send a listObjects request using
        // the temporary security credentials.
        ObjectListing objects = s3Client.listObjects(bucketName);
        System.out.println("No. of Objects = " +
objects.getObjectSummaries().size());
    } catch (AmazonServiceException e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.
        e.printStackTrace();
    } catch (SdkClientException e) {
        // Amazon S3 couldn't be contacted for a response, or the client
        // couldn't parse the response from Amazon S3.
        e.printStackTrace();
    }
}
```

```
}  
}
```

.NET

You can provide temporary security credentials for your federated users and applications so that they can send authenticated requests to access your Amazon resources. When requesting these temporary credentials, you must provide a user name and an IAM policy that describes the resource permissions that you want to grant. By default, the duration of a session is one hour. You can explicitly set a different duration value when requesting the temporary security credentials for federated users and applications. For information about sending authenticated requests, see [Making requests](#).

Note

When requesting temporary security credentials for federated users and applications, for added security, we suggest that you use a dedicated IAM user with only the necessary access permissions. The temporary user you create can never get more permissions than the IAM user who requested the temporary security credentials. For more information, see [Amazon Identity and Access Management FAQs](#).

You do the following:

- Create an instance of the Amazon Security Token Service client, `AmazonSecurityTokenServiceClient` class.
- Start a session by calling the `GetFederationToken` method of the STS client. You need to provide session information, including the user name and an IAM policy that you want to attach to the temporary credentials. Optionally, you can provide a session duration. This method returns your temporary security credentials.
- Package the temporary security credentials in an instance of the `SessionAWSCredentials` object. You use this object to provide the temporary security credentials to your Amazon S3 client.
- Create an instance of the `AmazonS3Client` class by passing the temporary security credentials. You use this client to send requests to Amazon S3. If you send requests using expired credentials, Amazon S3 returns an error.

Example

The following C# example lists the keys in the specified bucket. In the example, you obtain temporary security credentials for a two-hour session for your federated user (User1), and use the credentials to send authenticated requests to Amazon S3.

- For this exercise, you create an IAM user with minimal permissions. Using the credentials of this IAM user, you request temporary credentials for others. This example lists only the objects in a specific bucket. Create an IAM user with the following policy attached:

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

The policy allows the IAM user to request temporary security credentials and access permission only to list your Amazon resources. For more information about how to create an IAM user, see [Creating Your IAM user User and Administrators Group](#) in the *IAM User Guide*.

- Use the IAM user security credentials to test the following example. The example sends authenticated request to Amazon S3 using temporary security credentials. The example specifies the following policy when requesting temporary security credentials for the federated user (User1), which restricts access to listing objects in a specific bucket (YourBucketName). You must update the policy and provide your own existing bucket name.

```
{
  "Statement": [
    {
      "Sid": "1",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": "arn:aws-cn:s3:::YourBucketName"
    }
  ]
}
```


- **Example**

Update the following sample and provide the bucket name that you specified in the preceding federated user access policy. For information about setting up and running the code examples, see [Getting Started with the Amazon SDK for .NET](#) in the *Amazon SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempFederatedCredentialsTest
    {
        private const string bucketName = "**** bucket name ****";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
            try
            {
                Console.WriteLine("Listing objects stored in a bucket");
                // Credentials use the default AWS SDK for .NET credential search
chain.

                // On local development machines, this is your default profile.
                SessionAWSCredentials tempCredentials =
                    await GetTemporaryFederatedCredentialsAsync();
            }
            catch { }
        }
    }
}
```

```

        // Create a client by providing temporary security credentials.
        using (client = new AmazonS3Client(bucketRegion))
        {
            ListObjectsRequest listObjectRequest = new
ListObjectsRequest();
            listObjectRequest.BucketName = bucketName;

            ListObjectsResponse response = await
client.ListObjectsAsync(listObjectRequest);
            List<S3Object> objects = response.S3Objects;
            Console.WriteLine("Object count = {0}", objects.Count);

            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }
    }
    catch (AmazonS3Exception e)
    {
        Console.WriteLine("Error encountered ***. Message:'{0}' when
writing an object", e.Message);
    }
    catch (Exception e)
    {
        Console.WriteLine("Unknown encountered on server. Message:'{0}'
when writing an object", e.Message);
    }
}

private static async Task<SessionAWSCredentials>
GetTemporaryFederatedCredentialsAsync()
{
    AmazonSecurityTokenServiceConfig config = new
AmazonSecurityTokenServiceConfig();
    AmazonSecurityTokenServiceClient stsClient =
        new AmazonSecurityTokenServiceClient(
            config);

    GetFederationTokenRequest federationTokenRequest =
        new GetFederationTokenRequest();
    federationTokenRequest.DurationSeconds = 7200;
    federationTokenRequest.Name = "User1";
    federationTokenRequest.Policy = @"{
        ""Statement"":
        [

```

```

        {
            ""Sid"":""Stmt1311212314284"",
            ""Action"":[""s3:ListBucket""],
            ""Effect"":""Allow"",
            ""Resource"":""arn:aws:s3::" + bucketName + @"""
        }
    ]
}
";

    GetFederationTokenResponse federationTokenResponse =
        await
stsClient.GetFederationTokenAsync(federationTokenRequest);
    Credentials credentials = federationTokenResponse.Credentials;

    SessionAWSCredentials sessionCredentials =
        new SessionAWSCredentials(credentials.AccessKeyId,
                                   credentials.SecretAccessKey,
                                   credentials.SessionToken);

    return sessionCredentials;
}
}
}

```

PHP

This topic explains how to use classes from version 3 of the Amazon SDK for PHP to request temporary security credentials for federated users and applications and use them to access resources stored in Amazon S3. For more information about the Amazon SDK for Ruby API, go to [Amazon SDK for Ruby - Version 2](#).

You can provide temporary security credentials to your federated users and applications so they can send authenticated requests to access your Amazon resources. When requesting these temporary credentials, you must provide a user name and an IAM policy that describes the resource permissions that you want to grant. These credentials expire when the session duration expires. By default, the session duration is one hour. You can explicitly set a different value for the duration when requesting the temporary security credentials for federated users and applications. For more information about temporary security credentials, see [Temporary Security Credentials](#) in the *IAM User Guide*. For information about providing temporary security credentials to your federated users and applications, see [Making requests](#).

For added security when requesting temporary security credentials for federated users and applications, we recommend using a dedicated IAM user with only the necessary access permissions. The temporary user you create can never get more permissions than the IAM user who requested the temporary security credentials. For information about identity federation, see [Amazon Identity and Access Management FAQs](#).

For more information about the Amazon SDK for Ruby API, go to [Amazon SDK for Ruby - Version 2](#).

Example

The following PHP example lists keys in the specified bucket. In the example, you obtain temporary security credentials for an hour session for your federated user (User1). Then you use the temporary security credentials to send authenticated requests to Amazon S3.

For added security when requesting temporary credentials for others, you use the security credentials of an IAM user who has permissions to request temporary security credentials. To ensure that the IAM user grants only the minimum application-specific permissions to the federated user, you can also limit the access permissions of this IAM user. This example lists only objects in a specific bucket. Create an IAM user with the following policy attached:

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

The policy allows the IAM user to request temporary security credentials and access permission only to list your Amazon resources. For more information about how to create an IAM user, see [Creating Your First IAM user and Administrators Group](#) in the *IAM User Guide*.

You can now use the IAM user security credentials to test the following example. The example sends an authenticated request to Amazon S3 using temporary security credentials. When requesting temporary security credentials for the federated user (User1), the example specifies the following policy, which restricts access to list objects in a specific bucket. Update the policy with your bucket name.

```
{
  "Statement": [
    {
      "Sid": "1",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": "arn:aws-cn:s3:::YourBucketName"
    }
  ]
}
```

In the following example, when specifying the policy resource, replace `YourBucketName` with the name of your bucket.:

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

// In real applications, the following code is part of your trusted code. It has
// the security credentials that you use to obtain temporary security credentials.
$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

// Fetch the federated credentials.
$sessionToken = $sts->getFederationToken([
    'Name' => 'User1',
    'DurationSeconds' => '3600',
    'Policy' => json_encode([
        'Statement' => [
            'Sid' => 'randomstatementid' . time(),
            'Action' => ['s3:ListBucket'],
            'Effect' => 'Allow',
            'Resource' => 'arn:aws:s3:::' . $bucket
        ]
    ])
]);
```

```
// The following will be part of your less trusted code. You provide temporary
// security credentials so the code can send authenticated requests to Amazon S3.

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);

try {
    $result = $s3->listObjects([
        'Bucket' => $bucket
    ]);
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

You can provide temporary security credentials for your federated users and applications so that they can send authenticated requests to access your Amazon resources. When requesting temporary credentials from the IAM service, you must provide a user name and an IAM policy that describes the resource permissions that you want to grant. By default, the session duration is one hour. However, if you are requesting temporary credentials using IAM user credentials, you can explicitly set a different duration value when requesting the temporary security credentials for federated users and applications. For information about temporary security credentials for your federated users and applications, see [Making requests](#).

Note

For added security when you request temporary security credentials for federated users and applications, you might want to use a dedicated IAM user with only the necessary access permissions. The temporary user you create can never get more permissions than the IAM user who requested the temporary security credentials. For more information, see [Amazon Identity and Access Management FAQs](#).

Example

The following Ruby code example allows a federated user with a limited set of permissions to lists keys in the specified bucket.

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'
require 'aws-sdk-iam'
require 'json'

# Checks to see whether a user exists in IAM; otherwise,
# creates the user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Aws::IAM::Types::User] The existing or new user.
# @example
#   iam = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam, 'my-user')
#   exit 1 unless user.user_name
#   puts "User's name: #{user.user_name}"
def get_user(iam, user_name)
  puts "Checking for a user with the name '#{user_name}'..."
  response = iam.get_user(user_name: user_name)
  puts "A user with the name '#{user_name}' already exists."
  response.user
# If the user doesn't exist, create them.
rescue Aws::IAM::Errors::NoSuchEntity
  puts "A user with the name '#{user_name}' doesn't exist. Creating this user..."
  response = iam.create_user(user_name: user_name)
  iam.wait_until(:user_exists, user_name: user_name)
  puts "Created user with the name '#{user_name}'."
  response.user
rescue StandardError => e
  puts "Error while accessing or creating the user named '#{user_name}':
#{e.message}"
end

# Gets temporary AWS credentials for an IAM user with the specified permissions.
#
# @param sts [Aws::STS::Client] An initialized AWS STS client.
# @param duration_seconds [Integer] The number of seconds for valid credentials.
```

```

# @param user_name [String] The user's name.
# @param policy [Hash] The access policy.
# @return [Aws::STS::Types::Credentials] AWS credentials for API authentication.
# @example
#   sts = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_temporary_credentials(sts, duration_seconds, user_name,
#     {
#       'Version' => '2012-10-17',
#       'Statement' => [
#         'Sid' => 'Stmt1',
#         'Effect' => 'Allow',
#         'Action' => 's3:ListBucket',
#         'Resource' => 'arn:aws:s3:::amzn-s3-demo-bucket'
#       ]
#     }
#   )
#   exit 1 unless credentials.access_key_id
#   puts "Access key ID: #{credentials.access_key_id}"
def get_temporary_credentials(sts, duration_seconds, user_name, policy)
  response = sts.get_federation_token(
    duration_seconds: duration_seconds,
    name: user_name,
    policy: policy.to_json
  )
  response.credentials
rescue StandardError => e
  puts "Error while getting federation token: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'amzn-s3-demo-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

```



```

if response.count.positive?
  puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
  puts 'Name => ETag'
  response.contents.each do |obj|
    puts "#{obj.key} => #{obj.etag}"
  end
else
  puts "No objects in the bucket named '#{bucket_name}'."
end
true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end

# Example usage:
def run_me
  region = "us-west-2"
  user_name = "my-user"
  bucket_name = "amzn-s3-demo-bucket"

  iam = Aws::IAM::Client.new(region: region)
  user = get_user(iam, user_name)

  exit 1 unless user.user_name

  puts "User's name: #{user.user_name}"
  sts = Aws::STS::Client.new(region: region)
  credentials = get_temporary_credentials(sts, 3600, user_name,
    {
      'Version' => '2012-10-17',
      'Statement' => [
        'Sid' => 'Stmnt1',
        'Effect' => 'Allow',
        'Action' => 's3:ListBucket',
        'Resource' =>
"arn:aws:s3:::#{bucket_name}"
      ]
    })

  exit 1 unless credentials.access_key_id

  puts "Access key ID: #{credentials.access_key_id}"
  s3_client = Aws::S3::Client.new(region: region, credentials: credentials)

```

```
    exit 1 unless list_objects_in_bucket?(s3_client, bucket_name)
  end

  run_me if $PROGRAM_NAME == __FILE__
```

Making requests using the REST API

This section contains information on how to make requests to Amazon S3 endpoints by using the REST API. For a list of Amazon S3 endpoints, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

Constructing S3 hostnames for REST API requests

Amazon S3 endpoints follow the structure shown below:

```
s3.Region.amazonaws.com.cn
```

Amazon S3 access points endpoints and dual-stack endpoints also follow the standard structure:

- **Amazon S3 access points** - s3-accesspoint.*Region*.amazonaws.com.cn
- **Dual-stack** - s3.dualstack.*Region*.amazonaws.com.cn

For a complete list of Amazon S3 Regions and endpoints, see [Amazon S3 endpoints and quotas](#) in the *Amazon Web Services General Reference*.

Virtual hosted-style and path-style requests

When making requests by using the REST API, you can use virtual hosted-style or path-style URIs for the Amazon S3 endpoints. For more information, see [Path-style requests](#).

Example Virtual hosted-Style request

Following is an example of a virtual hosted-style request to delete the puppy.jpg file from the bucket named examplebucket in the US West (Oregon) Region. For more information about virtual hosted-style requests, see [Path-style requests](#).

```
DELETE /puppy.jpg HTTP/1.1
Host: examplebucket.s3.us-west-2.amazonaws.com.cn
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
```

Authorization: *authorization string*

Example Path-style request

Following is an example of a path-style version of the same request.

```
DELETE /examplebucket/puppy.jpg HTTP/1.1
Host: s3.us-west-2.amazonaws.com.cn
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Currently, Amazon S3 supports both virtual-hosted-style and path-style URL access in all Amazon Web Services Regions. However, path-style URLs will be discontinued in the future. For more information, see the following **Important** note.

For more information about path-style requests, see [Path-style requests](#).

Important

Update (September 23, 2020) – To make sure that customers have the time that they need to transition to virtual-hosted-style URLs, we have decided to delay the deprecation of path-style URLs. For more information, see [Amazon S3 Path Deprecation Plan – The Rest of the Story](#) in the *Amazon News Blog*.

Making requests to dual-stack endpoints by using the REST API

When using the REST API, you can directly access a dual-stack endpoint by using a virtual hosted-style or a path style endpoint name (URI). All Amazon S3 dual-stack endpoint names include the region in the name. Unlike the standard IPv4-only endpoints, both virtual hosted-style and a path-style endpoints use region-specific endpoint names.

Example Virtual hosted-Style dual-stack endpoint request

You can use a virtual hosted-style endpoint in your REST request as shown in the following example that retrieves the `puppy.jpg` object from the bucket named `examplebucket` in the US West (Oregon) Region.

```
GET /puppy.jpg HTTP/1.1
```

```
Host: examplebucket.s3.dualstack.us-west-2.amazonaws.com.cn
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Example Path-style dual-stack endpoint request

Or you can use a path-style endpoint in your request as shown in the following example.

```
GET /examplebucket/puppy.jpg HTTP/1.1
Host: s3.dualstack.us-west-2.amazonaws.com.cn
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

For more information about dual-stack endpoints, see [Using Amazon S3 dual-stack endpoints](#).

For more information about making requests using the REST API, see the following topics.

Topics

- [Request redirection and the REST API](#)
- [Request routing](#)

Request redirection and the REST API

Topics

- [Redirects and HTTP user-agents](#)
- [Redirects and 100-Continue](#)
- [Redirect example](#)

This section describes how to handle HTTP redirects by using the Amazon S3 REST API. For general information about Amazon S3 redirects, see [Making requests](#) in the Amazon Simple Storage Service API Reference.

Redirects and HTTP user-agents

Programs that use the Amazon S3 REST API should handle redirects either at the application layer or the HTTP layer. Many HTTP client libraries and user agents can be configured to correctly

handle redirects automatically; however, many others have incorrect or incomplete redirect implementations.

Before you rely on a library to fulfill the redirect requirement, test the following cases:

- Verify all HTTP request headers are correctly included in the redirected request (the second request after receiving a redirect) including HTTP standards such as Authorization and Date.
- Verify non-GET redirects, such as PUT and DELETE, work correctly.
- Verify large PUT requests follow redirects correctly.
- Verify PUT requests follow redirects correctly if the 100-continue response takes a long time to arrive.

HTTP user-agents that strictly conform to RFC 2616 might require explicit confirmation before following a redirect when the HTTP request method is not GET or HEAD. It is generally safe to follow redirects generated by Amazon S3 automatically, as the system will issue redirects only to hosts within the amazonaws.com.cn domain and the effect of the redirected request will be the same as that of the original request.

Redirects and 100-Continue

To simplify redirect handling, improve efficiencies, and avoid the costs associated with sending a redirected request body twice, configure your application to use 100-continues for PUT operations. When your application uses 100-continue, it does not send the request body until it receives an acknowledgement. If the message is rejected based on the headers, the body of the message is not sent. For more information about 100-continue, go to [RFC 2616 Section 8.2.3](#)

Note

According to RFC 2616, when using Expect: Continue with an unknown HTTP server, you should not wait an indefinite period before sending the request body. This is because some HTTP servers do not recognize 100-continue. However, Amazon S3 does recognize if your request contains an Expect: Continue and will respond with a provisional 100-continue status or a final status code. Additionally, no redirect error will occur after receiving the provisional 100 continue go-ahead. This will help you avoid receiving a redirect response while you are still writing the request body.

Redirect example

This section provides an example of client-server interaction using HTTP redirects and 100-continue.

Following is a sample PUT to the `quotes.s3.amazonaws.com.cn` bucket.

```
PUT /nelson.txt HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000
```

```
Content-Length: 6
Expect: 100-continue
```

Amazon S3 returns the following:

```
HTTP/1.1 307 Temporary Redirect
Location: http://quotes.s3-4c25d83b.amazonaws.com.cn/nelson.txt?rk=8d47490b
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Mon, 15 Oct 2007 22:18:46 GMT
```

```
Server: AmazonS3
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the
    specified temporary endpoint. Continue to use the
    original request endpoint for future requests.
  </Message>
  <Endpoint>quotes.s3-4c25d83b.amazonaws.com.cn</Endpoint>
  <Bucket>quotes</Bucket>
</Error>
```

The client follows the redirect response and issues a new request to the `quotes.s3-4c25d83b.amazonaws.com.cn` temporary endpoint.

```
PUT /nelson.txt?rk=8d47490b HTTP/1.1
Host: quotes.s3-4c25d83b.amazonaws.com.cn
```

```
Date: Mon, 15 Oct 2007 22:18:46 +0000
```

```
Content-Length: 6  
Expect: 100-continue
```

Amazon S3 returns a 100-continue indicating the client should proceed with sending the request body.

```
HTTP/1.1 100 Continue
```

The client sends the request body.

```
ha ha\n
```

Amazon S3 returns the final response.

```
HTTP/1.1 200 OK  
Date: Mon, 15 Oct 2007 22:18:48 GMT  
  
ETag: "a2c8d6b872054293afd41061e93bc289"  
Content-Length: 0  
Server: AmazonS3
```

Request routing

Programs that make requests against buckets created using the [CreateBucket](#) API that include a [CreateBucketConfiguration](#) must support redirects. Additionally, some clients that do not respect DNS TTLs might encounter issues.

This section describes routing and DNS issues to consider when designing your service or application for use with Amazon S3.

Request redirection and the REST API

Amazon S3 uses the Domain Name System (DNS) to route requests to facilities that can process them. This system works effectively, but temporary routing errors can occur. If a request arrives at the wrong Amazon S3 location, Amazon S3 responds with a temporary redirect that tells the requester to resend the request to a new endpoint. If a request is incorrectly formed, Amazon S3 uses permanent redirects to provide direction on how to perform the request correctly.

⚠ Important

To use this feature, you must have an application that can handle Amazon S3 redirect responses. The only exception is for applications that work exclusively with buckets that were created without `<CreateBucketConfiguration>`. For more information about location constraints, see [Accessing and listing an Amazon S3 bucket](#).

For all Regions that launched after March 20, 2019, if a request arrives at the wrong Amazon S3 location, Amazon S3 returns an HTTP 400 Bad Request error.

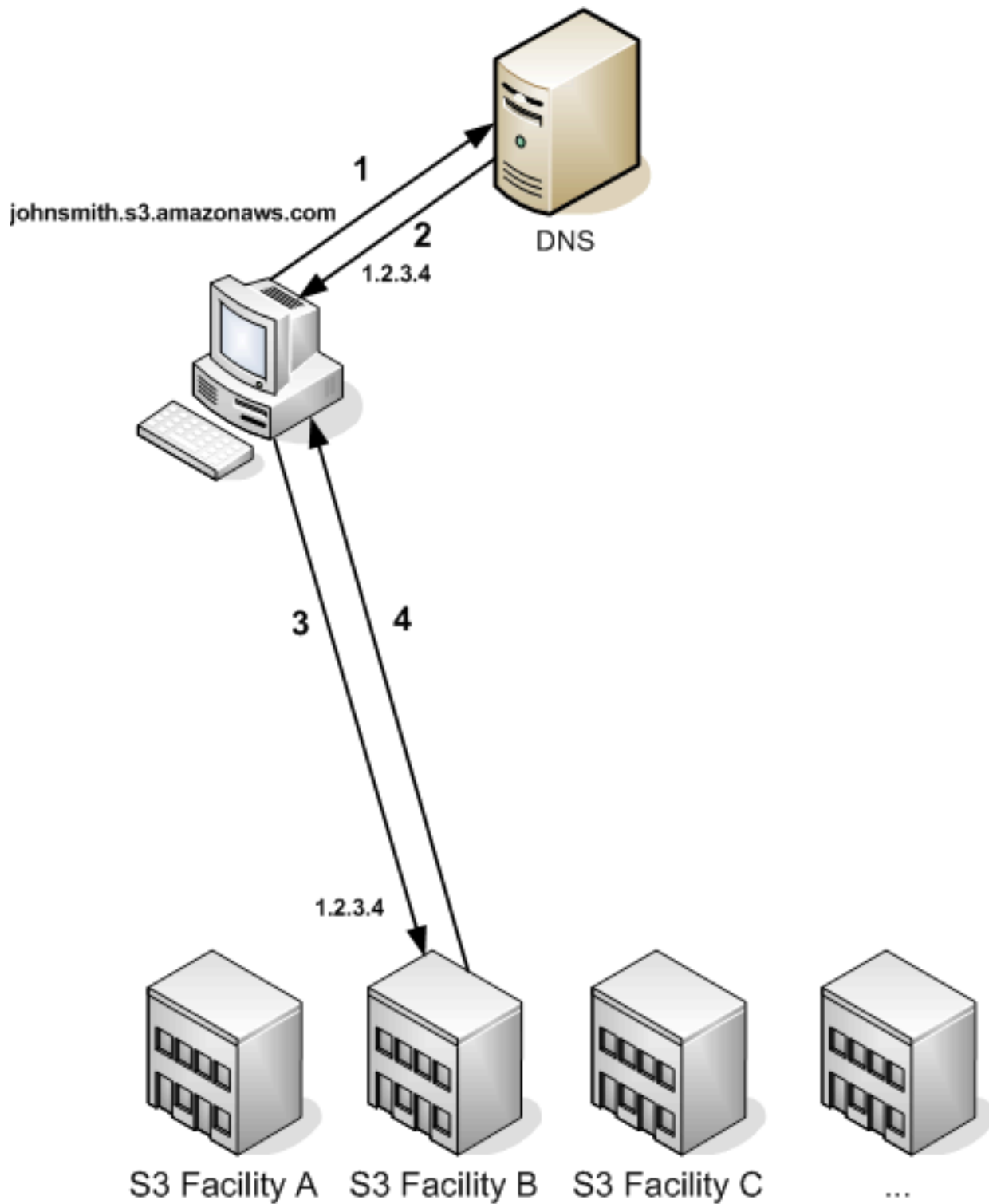
For more information about enabling or disabling an Amazon Web Services Region, see [Amazon Web Services Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

Topics

- [DNS routing](#)
- [Temporary request redirection](#)
- [Permanent request redirection](#)
- [Request redirection examples](#)

DNS routing

DNS routing routes requests to appropriate Amazon S3 facilities. The following figure and procedure show an example of DNS routing.



DNS routing request steps

1. The client makes a DNS request to get an object stored on Amazon S3.

2. The client receives one or more IP addresses for facilities that can process the request. In this example, the IP address is for Facility B.
3. The client makes a request to Amazon S3 Facility B.
4. Facility B returns a copy of the object to the client.

Temporary request redirection

A temporary redirect is a type of error response that signals to the requester that they should resend the request to a different endpoint. Due to the distributed nature of Amazon S3, requests can be temporarily routed to the wrong facility. This is most likely to occur immediately after buckets are created or deleted.

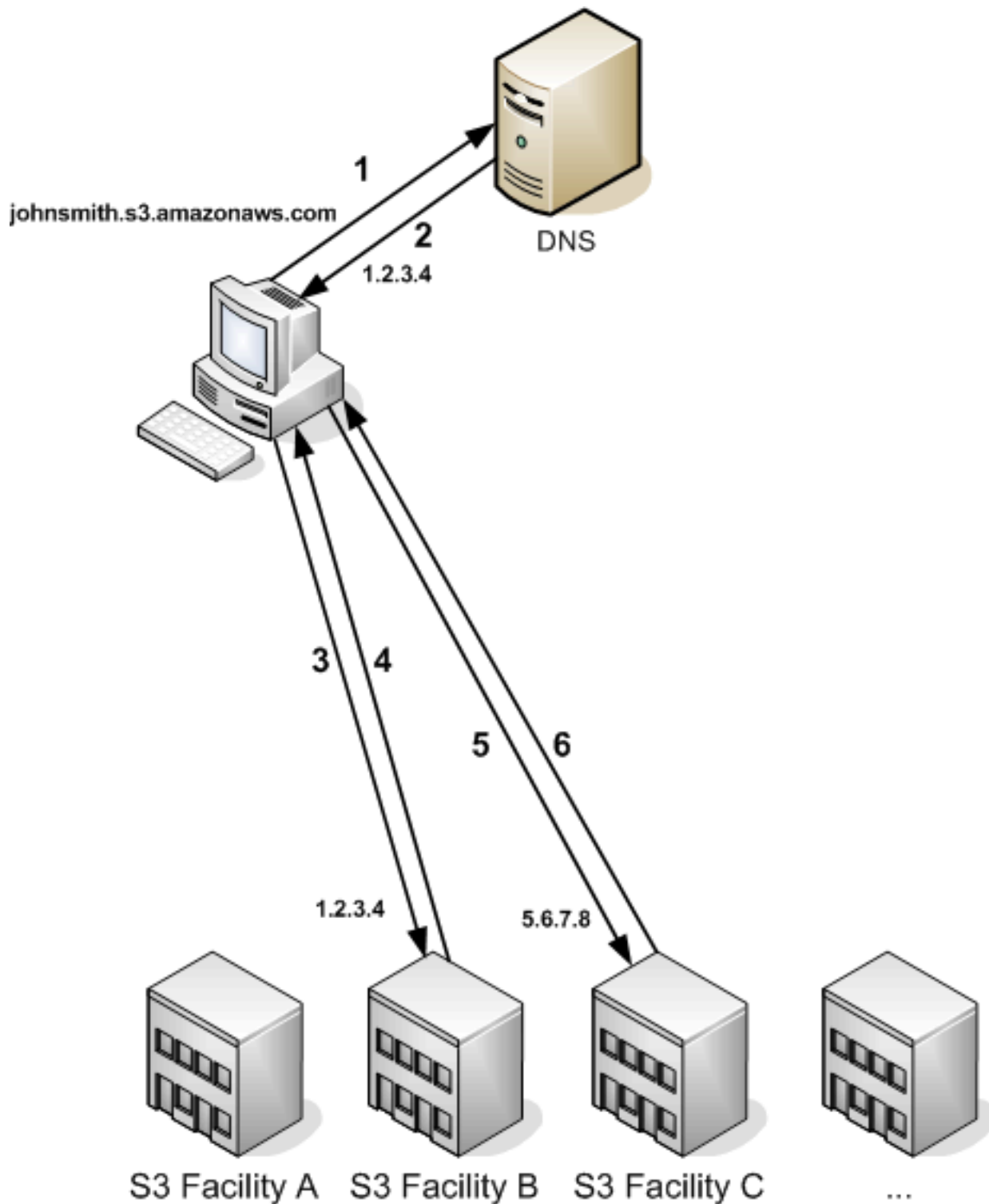
For example, if you create a new bucket and immediately make a request to the bucket, you might receive a temporary redirect, depending on the location constraint of the bucket. If you created the bucket in the US East (N. Virginia) Amazon Web Services Region, you will not see the redirect because this is also the default Amazon S3 endpoint.

However, if the bucket is created in any other Region, any requests for the bucket go to the default endpoint while the bucket's DNS entry is propagated. The default endpoint redirects the request to the correct endpoint with an HTTP 302 response. Temporary redirects contain a URI to the correct facility, which you can use to immediately resend the request.

Important

Don't reuse an endpoint provided by a previous redirect response. It might appear to work (even for long periods of time), but it might provide unpredictable results and will eventually fail without notice.

The following figure and procedure shows an example of a temporary redirect.



Temporary request redirection steps

1. The client makes a DNS request to get an object stored on Amazon S3.
2. The client receives one or more IP addresses for facilities that can process the request.

3. The client makes a request to Amazon S3 Facility B.
4. Facility B returns a redirect indicating the object is available from Location C.
5. The client resends the request to Facility C.
6. Facility C returns a copy of the object.

Permanent request redirection

A permanent redirect indicates that your request addressed a resource inappropriately. For example, permanent redirects occur if you use a path-style request to access a bucket that was created using `<CreateBucketConfiguration>`. For more information, see [Accessing and listing an Amazon S3 bucket](#).

To help you find these errors during development, this type of redirect does not contain a Location HTTP header that allows you to automatically follow the request to the correct location. Consult the resulting XML error document for help using the correct Amazon S3 endpoint.

Request redirection examples

The following are examples of temporary request redirection responses.

REST API temporary redirect response

```
HTTP/1.1 307 Temporary Redirect
Location: http://awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com.cn/photos/puppy.jpg?rk=e2c69a31
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Fri, 12 Oct 2007 01:12:56 GMT
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the specified temporary endpoint. Continue to use the original request endpoint for future requests.</Message>
  <Endpoint>awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com.cn</Endpoint>
</Error>
```

SOAP API temporary redirect response

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.TemporaryRedirect</Faultcode>
    <Faultstring>Please re-send this request to the specified temporary endpoint.
    Continue to use the original request endpoint for future requests.</Faultstring>
    <Detail>
      <Bucket>images</Bucket>
      <Endpoint>s3-gztlb4pa9sq.amazonaws.com.cn</Endpoint>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

DNS considerations

One of the design requirements of Amazon S3 is extremely high availability. One of the ways we meet this requirement is by updating the IP addresses associated with the Amazon S3 endpoint in DNS as needed. These changes are automatically reflected in short-lived clients, but not in some long-lived clients. Long-lived clients will need to take special action to re-resolve the Amazon S3 endpoint periodically to benefit from these changes. For more information about virtual machines (VMs), refer to the following:

- For Java, Sun's JVM caches DNS lookups forever by default; go to the "InetAddress Caching" section of [the InetAddress documentation](#) for information on how to change this behavior.
- For PHP, the persistent PHP VM that runs in the most popular deployment configurations caches DNS lookups until the VM is restarted. Go to [the getHostByName PHP docs](#).

Developing with Amazon S3 using the Amazon CLI

The Amazon S3 Amazon CLI commands are organized into different sets in Amazon CLI Command Reference and each has its own available commands. If you don't find the command that you're looking for in one set, check one of the other sets. The different sets are as follows:

- `s3` – Describes high-level commands for working with objects and buckets. These include copy, list and move actions. For a complete list of commands in this set, see [s3](#) in the Amazon CLI Command Reference.
- `s3api` – Describes low-level commands that manage S3 resources such as buckets, objects, sessions and multipart uploads. These commands correspond to the [Amazon S3 API](#) operations. For a complete list of CLI commands in this set, see [s3api](#) in the Amazon CLI Command Reference.
- `s3control` – Describes low-level commands that manage all other S3 resources such as Access Grants, Storage Lens groups, and Amazon S3 on Outposts buckets. These commands correspond to the [Amazon S3 Control](#) API operations. For a complete list of CLI commands in this set, see [s3control](#) in the Amazon CLI Command Reference.

If you are having issues setting up the Amazon CLI, see [How do I resolve the "Unable to locate credentials" error in Amazon S3](#) in the Amazon Web Services re:Post Knowledge Center.

Note

Services in Amazon, such as Amazon S3, require that you provide credentials when you access them. The service can then determine whether you have permissions to access the resources that it owns. The console requires your password. You can create access keys for your Amazon Web Services account to access the Amazon CLI or API. However, we don't recommend that you access Amazon using the credentials for your Amazon Web Services account. Instead, we recommend that you use Amazon Identity and Access Management (IAM). Create an IAM user, add the user to an IAM group with administrative permissions, and then grant administrative permissions to the IAM user that you created. You can then access Amazon using a special URL and the credentials of that IAM user. For instructions, go to [Creating Your First IAM user and Administrators Group](#) in the *IAM User Guide*.

Learn more about the Amazon CLI

To learn more about the Amazon, see the following resources:

- [Amazon Command Line Interface](#)
- [Amazon Command Line Interface User Guide for Version 2](#)

Developing with Amazon S3 using the Amazon SDKs

Amazon software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

Note

You can use Amazon Amplify for end-to-end fullstack development of web and mobile apps. Amplify Storage seamlessly integrates file storage and management capabilities into frontend web and mobile apps, built on top of Amazon S3. For more information, see [Storage](#) in the Amplify user guide.

SDK documentation

[Amazon CLI](#)

[Amazon SDK for Java](#)

[Amazon SDK for JavaScript](#)

[Amazon SDK for .NET](#)

[Amazon SDK for PHP](#)

[Amazon Tools for PowerShell](#)

[Amazon SDK for Python \(Boto3\)](#)

[Amazon SDK for Ruby](#)

SDK documentation

[Amazon SDK for SAP ABAP](#)

For specific examples, see [Code examples for Amazon S3 using Amazon SDKs](#).

SDK Programming interfaces

Each Amazon SDK provides one or more programmatic interfaces for working with Amazon S3. Each SDK provides a low-level interface for Amazon S3, with methods that closely resemble API operations. Some SDKs provide high-level interfaces for Amazon S3, that are abstractions intended to simplify common use cases.

For example, when you perform a multipart upload by using the low-level API operations, you use an operation to initiate the upload, another operation to upload parts, and a final operation to complete the upload. A high-level multipart upload API operation lets you to do all of the operations required for upload in a single API call. For examples, see [Uploading an object using multipart upload](#) in the *Amazon S3 User Guide*.

Low-level API operations allow greater control over the upload. We recommend that you use the low-level API operations if you need to pause and resume uploads, vary part sizes during the upload, or begin uploads when you don't know the size of the data in advance.

Specifying the Signature Version in request authentication

Amazon S3 supports only Amazon Signature Version 4 in most Amazon Web Services Regions. In some of the older Amazon Web Services Regions, Amazon S3 supports both Signature Version 4 and Signature Version 2. However, Signature Version 2 is being turned off (deprecated). For more information about the end of support for Signature Version 2, see [Amazon Signature Version 2 turned off \(deprecated\) for Amazon S3](#).

For a list of all the Amazon S3 Regions and the signature versions they support, see [Regions and Endpoints](#) in the *Amazon General Reference*.

For all Amazon Web Services Regions, Amazon SDKs use Signature Version 4 by default to authenticate requests. When using Amazon SDKs that were released before May 2016, you might be required to request Signature Version 4, as shown in the following table.

SDK	Requesting Signature Version 4 for Request Authentication
Amazon CLI	For the default profile, run the following command: <pre data-bbox="613 342 1507 443">\$ aws configure set default.s3.signature_version s3v4</pre> For a custom profile, run the following command: <pre data-bbox="613 558 1507 659">\$ aws configure set profile.your_profile_name.s3.signature_version s3v4</pre>
Java SDK	Add the following in your code: <pre data-bbox="613 789 1507 890">System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");</pre> Or, on the command line, specify the following: <pre data-bbox="613 1016 1507 1083">-Dcom.amazonaws.services.s3.enableV4</pre>
JavaScript SDK	Set the <code>signatureVersion</code> parameter to v4 when constructing the client: <pre data-bbox="613 1255 1507 1331">var s3 = new AWS.S3({signatureVersion: 'v4'});</pre>
PHP SDK	Set the <code>signature</code> parameter to v4 when constructing the Amazon S3 service client for PHP SDK v2: <pre data-bbox="613 1493 1507 1751"><?php \$client = S3Client::factory(['region' => 'YOUR-REGION', 'version' => 'latest', 'signature' => 'v4']);</pre>

SDK	Requesting Signature Version 4 for Request Authentication
	When using the PHP SDK v3, set the <code>signature_version</code> parameter to <code>v4</code> during construction of the Amazon S3 service client: <pre data-bbox="609 380 1507 653"><?php \$s3 = new Aws\S3\S3Client(['version' => '2006-03-01', 'region' => 'YOUR-REGION', 'signature_version' => 'v4']);</pre>
Ruby SDK	Ruby SDK - Version 1: Set the <code>:s3_signature_version</code> parameter to <code>v4</code> when constructing the client: <pre data-bbox="609 821 1507 936">s3 = AWS::S3::Client.new(:s3_signature_version => :v4)</pre> Ruby SDK - Version 3: Set the <code>signature_version</code> parameter to <code>v4</code> when constructing the client: <pre data-bbox="609 1094 1507 1167">s3 = Aws::S3::Client.new(signature_version: 'v4')</pre>
.NET SDK	Add the following to the code before creating the Amazon S3 client: <pre data-bbox="609 1335 1507 1409">AWSConfigsS3.UseSignatureVersion4 = true;</pre> Or, add the following to the config file: <pre data-bbox="609 1524 1507 1713"><appSettings> <add key="AWS.S3.UseSignatureVersion4" value="true" /> </appSettings></pre>

Amazon Signature Version 2 turned off (deprecated) for Amazon S3

Signature Version 2 is being turned off (deprecated) in Amazon S3. Amazon S3 will then only accept API requests that are signed using Signature Version 4.

This section provides answers to common questions regarding the end of support for Signature Version 2.

What is Signature Version 2/4, and What Does It Mean to Sign Requests?

The Signature Version 2 or Signature Version 4 signing process is used to authenticate your Amazon S3 API requests. Signing requests enables Amazon S3 to identify who is sending the request and protects your requests from bad actors.

For more information about signing Amazon requests, see [Signing Amazon API Requests](#) in the *Amazon Web Services General Reference*.

What Update Are You Making?

We currently support Amazon S3 API requests that are signed using Signature Version 2 and Signature Version 4 processes. After that, Amazon S3 will only accept requests that are signed using Signature Version 4.

For more information about signing Amazon requests, see [Changes in Signature Version 4](#) in the *Amazon Web Services General Reference*.

Why Are You Making the Update?

Signature Version 4 provides improved security by using a signing key instead of your secret access key. Signature Version 4 is currently supported in all Amazon Web Services Regions, whereas Signature Version 2 is only supported in Regions that were launched before January 2014. This update allows us to provide a more consistent experience across all Regions.

How Do I Ensure That I'm Using Signature Version 4, and What Updates Do I Need?

The signature version that is used to sign your requests is usually set by the tool or the SDK on the client side. By default, the latest versions of our Amazon SDKs use Signature Version 4. For third-party software, contact the appropriate support team for your software to confirm what version you need. If you are sending direct REST calls to Amazon S3, you must modify your application to use the Signature Version 4 signing process.

For information about which version of the Amazon SDKs to use when moving to Signature Version 4, see [Moving from Signature Version 2 to Signature Version 4](#).

For information about using Signature Version 4 with the Amazon S3 REST API, see [Authenticating Requests \(Amazon Signature Version 4\)](#) in the *Amazon Simple Storage Service API Reference*.

What Happens if I Don't Make Updates?

Requests signed with Signature Version 2 that are made after that will fail to authenticate with Amazon S3. Requesters will see errors stating that the request must be signed with Signature Version 4.

Should I Make Changes Even if I'm Using a Presigned URL That Requires Me to Sign for More than 7 Days?

If you are using a presigned URL that requires you to sign for more than 7 days, no action is currently needed. You can continue to use Amazon Signature Version 2 to sign and authenticate the presigned URL. We will follow up and provide more details on how to migrate to Signature Version 4 for a presigned URL scenario.

More Info

- For more information about using Signature Version 4, see [Signing Amazon API Requests](#).
- View the list of changes between Signature Version 2 and Signature Version 4 in [Changes in Signature Version 4](#).
- View the post [Amazon Signature Version 4 to replace Amazon Signature Version 2 for signing Amazon S3 API requests](#) in the Amazon forums.
- If you have any questions or concerns, contact [Amazon Web Services Support](#).

Moving from Signature Version 2 to Signature Version 4

If you currently use Signature Version 2 for Amazon S3 API request authentication, you should move to using Signature Version 4. Support is ending for Signature Version 2, as described in [Amazon Signature Version 2 turned off \(deprecated\) for Amazon S3](#).

For information about using Signature Version 4 with the Amazon S3 REST API, see [Authenticating Requests \(Amazon Signature Version 4\)](#) in the *Amazon Simple Storage Service API Reference*.

The following table lists the SDKs with the necessary minimum version to use Signature Version 4 (SigV4). If you are using presigned URLs with the Amazon Java, JavaScript (Node.js), or Python (Boto/CLI) SDKs, you must set the correct Amazon Web Services Region and set Signature Version 4 in the client configuration. For information about setting SigV4 in the client configuration, see [Specifying the Signature Version in request authentication](#).

If you use this SDK/ Product	Upgrade to this SDK version	Code change needed to the client to use Sigv4?	Link to SDK documentation
Amazon SDK for Java v1	Upgrade to Java 1.11.201+ or v2.	Yes	Specifying the Signature Version in request authentication
Amazon SDK for Java v2	No SDK upgrade is needed.	No	Amazon SDK for Java
Amazon SDK for .NET v1	Upgrade to 3.1.10 or later.	Yes	Amazon SDK for .NET
Amazon SDK for .NET v2	Upgrade to 3.1.10 or later.	No	Amazon SDK for .NET v2
Amazon SDK for .NET v3	Upgrade to 3.3.0.0 or later.	Yes	Amazon SDK for .NET v3
Amazon SDK for JavaScript v1	Upgrade to 2.68.0 or later.	Yes	Amazon SDK for JavaScript

If you use this SDK/Product	Upgrade to this SDK version	Code change needed to the client to use Sigv4?	Link to SDK documentation
Amazon SDK for JavaScript v2	Upgrade to 2.68.0 or later.	Yes	Amazon SDK for JavaScript
Amazon SDK for JavaScript v3	No action is currently needed. Upgrade to major version V3 in Q3 2019.	No	Amazon SDK for JavaScript
Amazon SDK for PHP v1	Recommend to upgrade to the most recent version of PHP or, at least to v2.7.4 with the signature parameter set to v4 in the S3 client's configuration.	Yes	Amazon SDK for PHP

If you use this SDK/Product	Upgrade to this SDK version	Code change needed to the client to use Sigv4?	Link to SDK documentation
Amazon SDK for PHP v2	Recommend to upgrade to the most recent version of PHP or, at least to v2.7.4 with the signature parameter set to v4 in the S3 client's configuration.	No	Amazon SDK for PHP
Amazon SDK for PHP v3	No SDK upgrade is needed.	No	Amazon SDK for PHP
Boto2	Upgrade to Boto2 v2.49.0.	Yes	Boto 2 Upgrade
Boto3	Upgrade to 1.5.71 (Botocore), 1.4.6 (Boto3).	Yes	Boto 3 - Amazon SDK for Python
Amazon CLI	Upgrade to 1.11.108.	Yes	Amazon Command Line Interface

If you use this SDK/Product	Upgrade to this SDK version	Code change needed to the client to use Sigv4?	Link to SDK documentation
Amazon CLI v2 (preview)	No SDK upgrade is needed.	No	Amazon Command Line Interface version 2
Amazon SDK for Ruby v1	Upgrade to Ruby V3.	Yes	Ruby V3 for Amazon
Amazon SDK for Ruby v2	Upgrade to Ruby V3.	Yes	Ruby V3 for Amazon
Amazon SDK for Ruby v3	No SDK upgrade is needed.	No	Ruby V3 for Amazon
Go	No SDK upgrade is needed.	No	Amazon SDK for Go
C++	No SDK upgrade is needed.	No	Amazon SDK for C++

Amazon Tools for Windows PowerShell or Amazon Tools for PowerShell Core

If you are using module versions *earlier* than 3.3.0.0, you must upgrade to 3.3.0.0.

To get the version information, use the `Get-Module` cmdlet:

```
Get-Module -Name AWSPowerShell
Get-Module -Name AWSPowerShell.NetCore
```


To update the 3.3.0.0 version, use the Update-Module cmdlet:

```
Update-Module -Name AWSPowerShell  
Update-Module -Name AWSPowerShell.NetCore
```

You can use presigned URLs that are valid for more than 7 days that you will send Signature Version 2 traffic on.

Getting Amazon S3 request IDs for Amazon Web Services Support

Whenever you contact Amazon Web Services Support because you've encountered errors or unexpected behavior in Amazon S3, you must provide the request IDs associated with the failed action. Amazon Web Services Support uses these request IDs to help resolve the problems that you're experiencing.

Request IDs come in pairs, are returned in every response that Amazon S3 processes (even the erroneous ones), and can be accessed through verbose logs. There are a number of common methods for getting your request IDs, including S3 server access logs and Amazon CloudTrail events or data events.

After you've recovered these logs, copy and retain those two values, because you'll need them when you contact Amazon Web Services Support. For information about contacting Amazon Web Services Support, see [Contact Amazon](#) or the [Amazon Web Services Support Documentation](#).

Topics

- [Using the Amazon CLI to obtain request IDs](#)
- [Using Windows PowerShell to obtain request IDs](#)
- [Using Amazon CloudTrail data events to obtain request IDs](#)
- [Using S3 server access logging to obtain request IDs](#)
- [Using HTTP to obtain request IDs](#)
- [Using a web browser to obtain request IDs](#)
- [Using the Amazon SDKs to obtain request IDs](#)

Using the Amazon CLI to obtain request IDs

To get your request IDs when using the Amazon Command Line Interface (Amazon CLI), add `--debug` to your command. For example you should see `x-amz-request-id` and `x-amz-id-2` in the debug log as shown below:

```
...
2025-04-30 14:35:28,572 - MainThread - botocore.parsers - DEBUG - Response headers:
{'x-amz-id-2': 'a+zm50vDJH3LLmCiXvwEo0u0PtPS/qCJaBvB2ZMH9dzyzTiJhiLZkBFFoRfsPf0KztUKT/
garCI=', 'x-amz-request-id': 'N4NMN0MJ4VDFZMX9', 'Date': 'Wed, 30 Apr 2025 21:35:29
GMT', 'Content-Type': 'application/xml', 'Transfer-Encoding': 'chunked', 'Server':
'AmazonS3'}
...
```

Using Windows PowerShell to obtain request IDs

For information on recovering logs with Windows PowerShell, see the [Response Logging in Amazon Tools for Windows PowerShell](#) in the *Amazon Developer Tools Blog*.

Using Amazon CloudTrail data events to obtain request IDs

An Amazon S3 bucket that is configured with CloudTrail data events to log S3 object-level API operations provides detailed information about actions that are taken by a user, role, or an Amazon service in Amazon S3. You can identify S3 request IDs by querying CloudTrail events with Athena.

For more information, see [Identifying Amazon S3 requests using CloudTrail](#) in the *Amazon Simple Storage Service User Guide*.

Using S3 server access logging to obtain request IDs

An Amazon S3 bucket configured for S3 server access logging provides detailed records for each request that is made to the bucket. You can identify S3 request IDs by querying the server access logs using Athena.

For more information, see [Querying access logs for requests by using Amazon Athena](#) in the *Amazon Simple Storage Service User Guide*.

Using HTTP to obtain request IDs

You can obtain your request IDs, `x-amz-request-id` and `x-amz-id-2` by logging the bits of an HTTP request before it reaches the target application. There are a variety of third-party tools that

can be used to recover verbose logs for HTTP requests. Choose one that you trust, and then run the tool to listen on the port that your Amazon S3 traffic travels on, as you send out another Amazon S3 HTTP request.

For HTTP requests, the pair of request IDs will look like the following:

```
x-amz-request-id: 79104EXAMPLEB723
x-amz-id-2: IOwQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km
```

Note

HTTPS requests are encrypted and hidden in most packet captures.

Using a web browser to obtain request IDs

Most web browsers have developer tools that you can use to view request headers.

For web browser-based requests that return an error, the pair of requests IDs will look like the following examples.

```
<Error><Code>AccessDenied</Code><Message>Access Denied</Message>
<RequestId>79104EXAMPLEB723</RequestId><HostId>IOwQ4fDEXAMPLEQM
+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km</HostId></Error>
```

To obtain the request ID pair from successful requests, use your browser's developer tools to look at the HTTP response headers.

Using the Amazon SDKs to obtain request IDs

The following sections include information for configuring logging by using different Amazon SDKs.

Note

Although you can enable verbose logging on every request and response, we don't recommend enabling logging in production systems, because large requests or responses can significantly slow down an application.

For Amazon SDK requests, the pair of request IDs will look like the following:

```
Status Code: 403, AWS Service: Amazon S3, AWS Request ID: 79104EXAMPLEB723
AWS Error Code: AccessDenied AWS Error Message: Access Denied
S3 Extended Request ID: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK
+Jd1vEXAMPLEa3Km
```

C++

The type of logger and the verbosity are specified during the SDK initialization in the `SDKOptions` argument. The following example specifies the verbosity level as `LogLevel::Debug`.

The default logger will write to the filesystem and the file is named using the following convention `aws_sdk_YYYY-MM-DD-HH.log`. The logger creates a new file on the hour.

```
Aws::SDKOptions options;
options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
Aws::InitAPI(options);
// ...
Aws::ShutdownAPI(options);
```

For more information, see [How do I turn on logging?](#) in the *Amazon SDK for C++ wiki on GitHub*.

Go

You can configure logging by using SDK for Go. For more information, see [Logging](#) in the *Amazon SDK for Go v2 Developer Guide*.

Java

You can enable logging for specific requests or responses to catch and return only relevant headers. To do this, import the `com.amazonaws.services.s3.S3ResponseMetadata` class. Afterward, you can store the request in a variable before performing the actual request. To get the logged request or response, call `getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()`.

```
PutObjectRequest req = new PutObjectRequest(bucketName, key, createSampleFile());
s3.putObject(req);
S3ResponseMetadata md = s3.getCachedResponseMetadata(req);
```

```
System.out.println("Host ID: " + md.getHostId() + " RequestID: " +  
md.getRequestId());
```

Alternatively, you can use verbose logging of every Java request and response. For more information, see [Verbose Wire Logging](#) in the *Amazon SDK for Java Developer Guide*.

JavaScript

The Amazon SDK for JavaScript has a built-in logger so you can log API calls you make with it. To turn on the logger and print log entries in the console, add the following statement to your code:

```
AWS.config.logger = console;
```

For more information, see [Logging Amazon SDK for JavaScript Calls](#) in the *Amazon SDK for JavaScript Developer Guide*.

Kotlin

With the Amazon SDK for Kotlin, you can specify log mode for wire-level messages using code or environment settings. You can set log mode for HTTP requests and HTTP responses.

To opt into additional logging, set the `logMode` property when you construct a service client:

```
import aws.smithy.kotlin.runtime.client.LogMode  
  
// ...  
  
val client = S3Client {  
    // ...  
    logMode = LogMode.LogRequestWithBody + LogMode.LogResponse  
}
```

Alternatively, you can set log mode using an environment variable:

```
export SDK_LOG_MODE=LogRequestWithBody|LogResponse
```

For more information, see [Logging](#) in the *Amazon SDK for Kotlin Developer Guide*.

.NET

You can configure logging with the Amazon SDK for .NET by using the built-in `System.Diagnostics` logging tool. For more information, see the [Logging with the Amazon SDK for .NET](#) *Amazon Developer Blog* post.

Note

By default, the returned log contains only error information. To get the request IDs, the config file must have `AWSLogMetrics` (and optionally, `AWSResponseLogging`) added.

PHP

You can get debug information, including the data sent over the wire, by setting the debug option to `true` in a client constructor.

```
$s3Client = new Aws\S3\S3Client([
    'region' => 'us-standard',
    'version' => '2006-03-01',
    'debug'   => true
]);
```

For more information, see [How can I see what data is sent over the wire?](#) in the *Amazon SDK for PHP Developer Guide*.

Python (Boto3)

With the Amazon SDK for Python (Boto3), you can log specific responses. You can use this feature to capture only the relevant headers. The following code shows how to log parts of the response to a file:

```
import logging
import boto3
logging.basicConfig(filename='logfile.txt', level=logging.INFO)
logger = logging.getLogger(__name__)
s3 = boto3.resource('s3')
response = s3.Bucket(bucket_name).Object(object_key).put()
logger.info("HTTPStatusCode: %s", response['ResponseMetadata']['HTTPStatusCode'])
logger.info("RequestId: %s", response['ResponseMetadata']['RequestId'])
logger.info("HostId: %s", response['ResponseMetadata']['HostId'])
```

```
logger.info("Date: %s", response['ResponseMetadata']['HTTPHeaders']['date'])
```

You can also catch exceptions and log relevant information when an exception is raised. For more information, see [Discerning useful information from error responses](#) in the *Amazon SDK for Python (Boto) API Reference*.

Additionally, you can configure Boto3 to output verbose debugging logs by using the following code:

```
import logging
import boto3
boto3.set_stream_logger('', logging.DEBUG)
```

For more information, see [set_stream_logger](#) in the *Amazon SDK for Python (Boto) API Reference*.

Ruby

You can get your request IDs using the SDK for Ruby Version 3. You can enable HTTP wire logging in your client using the following code:

```
s3 = Aws::S3::Client.new(http_wire_trace: true)
```

Note

Wire logging can include sensitive information, such as your access key ID. Sensitive information should be sanitized before sharing it with Amazon Support.

You can also find the request ID of the request context object in the request response or error:

```
# Finding the request ID from an error:
begin
  s3.put_object(bucket: 'bucket', key: 'key', body: 'test')
rescue Aws::S3::Errors::ServiceError => e
  puts e.context[:request_id]
  puts e.context[:s3_host_id]
end

# Finding the request ID from a successful call:
```

```
resp = s3.put_object(bucket: 'bucket', key: 'key', body: 'test')
puts resp.context[:request_id]
puts resp.context[:s3_host_id]
```

For more information, see [Debugging using wire trace information from an Amazon SDK for Ruby client](#) in the *Amazon SDK for Ruby Developer Guide*.

Rust

To enable logging, add the `tracing-subscriber` crate and initialize it in your Rust application.

Add the tracing library to your `Cargo.toml` file:

```
tracing-subscriber = { version = "0.3", features = ["env-filter"] }
```

Then, in your Rust code, initialize the logger in the main function before you call any SDK operation:

```
tracing_subscriber::fmt::init();
```

For more information, see [Configuring and using logging in the Amazon SDK for Rust](#) in the *Amazon SDK for Rust Developer Guide*.

Supported Amazon S3 object-level API operations for S3 Tables

The following table includes supported S3 object-level API operations and corresponding headers for S3 Tables. For a list of Amazon S3 Tables APIs, see [Amazon S3 Tables](#). For more information about Amazon S3 Tables, see [Working with Amazon S3 Tables and table buckets](#) in the *Amazon S3 User Guide*.

Object-level API	Supported headers	Notes	
AbortMultipartUpload	x-amz-expected-bucket-owner	None	

Object-level API	Supported headers	Notes	
CompleteMultipartUpload	x-amz-checksum-crc32	None	
	x-amz-checksum-crc32c	None	
	x-amz-checksum-sha1	None	
	x-amz-checksum-sha256	None	
	x-amz-expected-bucket-owner	None	
	If-Match	None	
	If-None-Match	None	
CreateMultipartUpload	x-amz-acl: ACL	For S3 Tables, the default value is bucket-owner-full-control and it can't be changed.	
	Cache-Control	None	
	Content-Disposition	None	
	Content-Encoding	None	
	Content-Language	None	
	Content-Type	None	
	Expires	None	

Object-level API	Supported headers	Notes	
	x-amz-checksum-algorithm	None	
	x-amz-storage-class	For S3 Tables, the default value is STANDARD and it can't be changed.	
	x-amz-server-side-encryption	For S3 Tables, the default value is (SSE-S3) (AES256) and it can't be changed.	
GetObject	If-Match	None	
	If-Modified-Since	None	
	If-None-Match	None	
	If-Unmodified-Since	None	
	Range	None	
	x-amz-expected-bucket-owner	None	
	x-amz-checksum-mode	None	
HeadObject	If-Match	None	
	If-Modified-Since	None	
	If-None-Match	None	

Object-level API	Supported headers	Notes	
	If-Unmodified-Since	None	
	Range	None	
	x-amz-checksum-mode	None	
	x-amz-expected-bucket-owner	None	
ListParts	x-amz-expected-bucket-owner	None	
PutObject	x-amz-acl: ACL	For S3 Tables, the default value is bucket-owner-full-control and it can't be changed.	
	Cache-Control	None	
	Content-Disposition	None	
	Content-Encoding	None	
	Content-Language	None	
	Content-Length	None	
	Content-MD5	None	
	Content-Type	None	
	x-amz-sdk-checksum-algorithm	None	

Object-level API	Supported headers	Notes	
	x-amz-checksum-crc32	None	
	x-amz-checksum-crc32c	None	
	x-amz-checksum-sha1	None	
	x-amz-checksum-sha256	None	
	Expires	None	
	If-None-Match	None	
	x-amz-expected-bucket-owner	None	
	x-amz-storage-classes	For S3 Tables, the default value is STANDARD and it can't be changed.	
	x-amz-server-side-encryption	For S3 Tables, the default value is (SSE-S3) (AES256) and it can't be changed.	
UploadPart	Content-Length	None	
	Content-MD5	None	
	x-amz-checksum-crc32	None	

Object-level API	Supported headers	Notes	
	x-amz-checksum-crc32c	None	
	x-amz-checksum-sha1	None	
	x-amz-checksum-sha256	None	
	x-amz-expected-bucket-owner	None	

Code examples for Amazon S3 using Amazon SDKs

The following code examples show how to use Amazon S3 with an Amazon software development kit (SDK).

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Code examples

- [Code examples for Amazon S3 using Amazon SDKs](#)
 - [Basic examples for Amazon S3 using Amazon SDKs](#)
 - [Hello Amazon S3](#)
 - [Learn the basics of Amazon S3 with an Amazon SDK](#)
 - [Actions for Amazon S3 using Amazon SDKs](#)
 - [Use AbortMultipartUpload with an Amazon SDK or CLI](#)
 - [Use CompleteMultipartUpload with an Amazon SDK or CLI](#)
 - [Use CopyObject with an Amazon SDK or CLI](#)
 - [Use CreateBucket with an Amazon SDK or CLI](#)
 - [Use CreateMultiRegionAccessPoint with an Amazon SDK](#)
 - [Use CreateMultipartUpload with an Amazon SDK or CLI](#)
 - [Use CreatePresignedPost with an Amazon SDK](#)
 - [Use DeleteBucket with an Amazon SDK or CLI](#)
 - [Use DeleteBucketAnalyticsConfiguration with a CLI](#)
 - [Use DeleteBucketCors with an Amazon SDK or CLI](#)
 - [Use DeleteBucketEncryption with a CLI](#)
 - [Use DeleteBucketInventoryConfiguration with a CLI](#)
 - [Use DeleteBucketLifecycle with an Amazon SDK or CLI](#)
 - [Use DeleteBucketMetricsConfiguration with a CLI](#)
 - [Use DeleteBucketPolicy with an Amazon SDK or CLI](#)
 - [Use DeleteBucketReplication with a CLI](#)
 - [Use DeleteBucketTagging with a CLI](#)

- [Use DeleteBucketWebsite with an Amazon SDK or CLI](#)
- [Use DeleteObject with an Amazon SDK or CLI](#)
- [Use DeleteObjectTagging with a CLI](#)
- [Use DeleteObjects with an Amazon SDK or CLI](#)
- [Use DeletePublicAccessBlock with a CLI](#)
- [Use GetBucketAccelerateConfiguration with a CLI](#)
- [Use GetBucketAcl with an Amazon SDK or CLI](#)
- [Use GetBucketAnalyticsConfiguration with a CLI](#)
- [Use GetBucketCors with an Amazon SDK or CLI](#)
- [Use GetBucketEncryption with an Amazon SDK or CLI](#)
- [Use GetBucketInventoryConfiguration with a CLI](#)
- [Use GetBucketLifecycleConfiguration with an Amazon SDK or CLI](#)
- [Use GetBucketLocation with an Amazon SDK or CLI](#)
- [Use GetBucketLogging with a CLI](#)
- [Use GetBucketMetricsConfiguration with a CLI](#)
- [Use GetBucketNotification with a CLI](#)
- [Use GetBucketPolicy with an Amazon SDK or CLI](#)
- [Use GetBucketPolicyStatus with a CLI](#)
- [Use GetBucketReplication with an Amazon SDK or CLI](#)
- [Use GetBucketRequestPayment with a CLI](#)
- [Use GetBucketTagging with a CLI](#)
- [Use GetBucketVersioning with a CLI](#)
- [Use GetBucketWebsite with an Amazon SDK or CLI](#)
- [Use GetObject with an Amazon SDK or CLI](#)
- [Use GetObjectAcl with an Amazon SDK or CLI](#)
- [Use GetObjectAttributes with an Amazon SDK or CLI](#)
- [Use GetObjectLegalHold with an Amazon SDK or CLI](#)
- [Use GetObjectLockConfiguration with an Amazon SDK or CLI](#)
- [Use GetObjectRetention with an Amazon SDK or CLI](#)
- [Use GetObjectTagging with a CLI](#)

- [Use GetPublicAccessBlock with a CLI](#)
- [Use HeadBucket with an Amazon SDK or CLI](#)
- [Use HeadObject with an Amazon SDK or CLI](#)
- [Use ListBucketAnalyticsConfigurations with a CLI](#)
- [Use ListBucketInventoryConfigurations with a CLI](#)
- [Use ListBuckets with an Amazon SDK or CLI](#)
- [Use ListMultipartUploads with an Amazon SDK or CLI](#)
- [Use ListObjectVersions with an Amazon SDK or CLI](#)
- [Use ListObjects with a CLI](#)
- [Use ListObjectsV2 with an Amazon SDK or CLI](#)
- [Use PutBucketAccelerateConfiguration with an Amazon SDK or CLI](#)
- [Use PutBucketAcl with an Amazon SDK or CLI](#)
- [Use PutBucketCors with an Amazon SDK or CLI](#)
- [Use PutBucketEncryption with an Amazon SDK or CLI](#)
- [Use PutBucketLifecycleConfiguration with an Amazon SDK or CLI](#)
- [Use PutBucketLogging with an Amazon SDK or CLI](#)
- [Use PutBucketNotification with a CLI](#)
- [Use PutBucketNotificationConfiguration with an Amazon SDK or CLI](#)
- [Use PutBucketPolicy with an Amazon SDK or CLI](#)
- [Use PutBucketReplication with an Amazon SDK or CLI](#)
- [Use PutBucketRequestPayment with a CLI](#)
- [Use PutBucketTagging with a CLI](#)
- [Use PutBucketVersioning with an Amazon SDK or CLI](#)
- [Use PutBucketWebsite with an Amazon SDK or CLI](#)
- [Use PutObject with an Amazon SDK or CLI](#)
- [Use PutObjectAcl with an Amazon SDK or CLI](#)
- [Use PutObjectLegalHold with an Amazon SDK or CLI](#)
- [Use PutObjectLockConfiguration with an Amazon SDK or CLI](#)
- [Use PutObjectRetention with an Amazon SDK or CLI](#)
- [Use RestoreObject with an Amazon SDK or CLI](#)

- [Use SelectObjectContent with an Amazon SDK or CLI](#)
- [Use UploadPart with an Amazon SDK or CLI](#)
- [Use UploadPartCopy with an Amazon SDK or CLI](#)
- [Scenarios for Amazon S3 using Amazon SDKs](#)
 - [Check if a bucket exists](#)
 - [Convert text to speech and back to text using an Amazon SDK](#)
 - [Create a presigned URL for Amazon S3 using an Amazon SDK](#)
 - [Create a photo asset management application that lets users manage photos using labels](#)
 - [A web page that lists Amazon S3 objects using an Amazon SDK](#)
 - [Create an Amazon Textract explorer application](#)
 - [Delete all objects in a given Amazon S3 bucket using an Amazon SDK](#)
 - [Delete incomplete multipart uploads to Amazon S3 using an Amazon SDK](#)
 - [Detect PPE in images with Amazon Rekognition using an Amazon SDK](#)
 - [Detect entities in text extracted from an image using an Amazon SDK](#)
 - [Detect faces in an image using an Amazon SDK](#)
 - [Detect objects in images with Amazon Rekognition using an Amazon SDK](#)
 - [Detect people and objects in a video with Amazon Rekognition using an Amazon SDK](#)
 - [Download S3 'directories' from an Amazon Simple Storage Service \(Amazon S3\) bucket](#)
 - [Download all objects in an Amazon Simple Storage Service \(Amazon S3\) bucket to a local directory](#)
 - [Download a stream of unknown size from an Amazon S3 object using an Amazon SDK](#)
 - [Get an Amazon S3 object from a Multi-Region Access Point by using an Amazon SDK](#)
 - [Get an object from an Amazon S3 bucket using an Amazon SDK, specifying an If-Modified-Since header](#)
 - [Get started with encryption for Amazon S3 objects using an Amazon SDK](#)
 - [Get started with tags for Amazon S3 objects using an Amazon SDK](#)
 - [Getting started with Amazon Athena](#)
 - [Getting started with Amazon EMR](#)
 - [Getting started with Amazon S3](#)
 - [Getting started with Amazon SageMaker Feature Store](#)
 - [Getting started with Amazon Textract](#)

- [Getting started with Config](#)
- [Work with Amazon S3 object lock features using an Amazon SDK](#)
- [Make Amazon S3 conditional requests using an Amazon SDK](#)
- [Manage access control lists \(ACLs\) for Amazon S3 buckets using an Amazon SDK](#)
- [Manage large Amazon SQS messages using Amazon S3 with an Amazon SDK](#)
- [Manage versioned Amazon S3 objects in batches with a Lambda function using an Amazon SDK](#)
- [Parse Amazon S3 URIs using an Amazon SDK](#)
- [Perform a multipart copy of an Amazon S3 object using an Amazon SDK](#)
- [Receive and process Amazon S3 event notifications by using an Amazon SDK](#)
- [Save EXIF and other image information using an Amazon SDK](#)
- [Send S3 event notifications to Amazon EventBridge using an Amazon SDK](#)
- [Track an Amazon S3 object upload or download using an Amazon SDK](#)
- [Transform data for your application with S3 Object Lambda](#)
- [Example approaches for unit and integration testing with an Amazon SDK](#)
- [Recursively upload a local directory to an Amazon Simple Storage Service \(Amazon S3\) bucket](#)
- [Upload or download large files to and from Amazon S3 using an Amazon SDK](#)
- [Upload a stream of unknown size to an Amazon S3 object using an Amazon SDK](#)
- [Use checksums to work with an Amazon S3 object using an Amazon SDK](#)
- [Work with Amazon S3 object integrity features using an Amazon SDK](#)
- [Work with Amazon S3 versioned objects using an Amazon SDK](#)
- [Serverless examples for Amazon S3](#)
 - [Invoke a Lambda function from an Amazon S3 trigger](#)
- [Code examples for Amazon S3 Control using Amazon SDKs](#)
 - [Basic examples for Amazon S3 Control using Amazon SDKs](#)
 - [Hello Amazon S3 Control](#)
 - [Learn the basics of Amazon S3 Control with an Amazon SDK](#)
 - [Actions for Amazon S3 Control using Amazon SDKs](#)
 - [Use CreateJob with an Amazon SDK or CLI](#)
 - [Use DeleteJobTagging with an Amazon SDK](#)

- [Use DescribeJob with an Amazon SDK or CLI](#)
- [Use GetJobTagging with an Amazon SDK](#)
- [Use PutJobTagging with an Amazon SDK](#)
- [Use UpdateJobPriority with an Amazon SDK or CLI](#)
- [Use UpdateJobStatus with an Amazon SDK or CLI](#)
- [Code examples for S3 Directory Buckets using Amazon SDKs](#)
 - [Basic examples for S3 Directory Buckets using Amazon SDKs](#)
 - [Hello Amazon S3 directory buckets](#)
 - [Learn the basics of S3 Directory Buckets with an Amazon SDK](#)
 - [Actions for S3 Directory Buckets using Amazon SDKs](#)
 - [Use AbortMultipartUpload with an Amazon SDK](#)
 - [Use CompleteMultipartUpload with an Amazon SDK](#)
 - [Use CopyObject with an Amazon SDK](#)
 - [Use CreateBucket with an Amazon SDK](#)
 - [Use CreateMultipartUpload with an Amazon SDK](#)
 - [Use CreateSession with an Amazon SDK](#)
 - [Use DeleteBucket with an Amazon SDK](#)
 - [Use DeleteBucketEncryption with an Amazon SDK](#)
 - [Use DeleteBucketPolicy with an Amazon SDK](#)
 - [Use DeleteObject with an Amazon SDK](#)
 - [Use DeleteObjects with an Amazon SDK](#)
 - [Use GetBucketEncryption with an Amazon SDK](#)
 - [Use GetBucketPolicy with an Amazon SDK](#)
 - [Use GetObject with an Amazon SDK](#)
 - [Use GetObjectAttributes with an Amazon SDK](#)
 - [Use HeadBucket with an Amazon SDK](#)
 - [Use HeadObject with an Amazon SDK](#)
 - [Use ListDirectoryBuckets with an Amazon SDK](#)
 - [Use ListMultipartUploads with an Amazon SDK](#)
 - [Use ListObjectsV2 with an Amazon SDK](#)

- [Use ListParts with an Amazon SDK](#)
- [Use PutBucketEncryption with an Amazon SDK](#)
- [Use PutBucketPolicy with an Amazon SDK](#)
- [Use PutObject with an Amazon SDK](#)
- [Use UploadPart with an Amazon SDK](#)
- [Use UploadPartCopy with an Amazon SDK](#)
- [Scenarios for S3 Directory Buckets using Amazon SDKs](#)
 - [Create a presigned URL for Amazon S3 directory buckets to get an object using an Amazon SDK](#)

Code examples for Amazon S3 using Amazon SDKs

The following code examples show how to use Amazon S3 with an Amazon software development kit (SDK).

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other Amazon Web Services services.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Amazon S3

- [Basic examples for Amazon S3 using Amazon SDKs](#)
 - [Hello Amazon S3](#)
 - [Learn the basics of Amazon S3 with an Amazon SDK](#)
 - [Actions for Amazon S3 using Amazon SDKs](#)
 - [Use AbortMultipartUpload with an Amazon SDK or CLI](#)
 - [Use CompleteMultipartUpload with an Amazon SDK or CLI](#)
 - [Use CopyObject with an Amazon SDK or CLI](#)

- [Use CreateBucket with an Amazon SDK or CLI](#)
- [Use CreateMultiRegionAccessPoint with an Amazon SDK](#)
- [Use CreateMultipartUpload with an Amazon SDK or CLI](#)
- [Use CreatePresignedPost with an Amazon SDK](#)
- [Use DeleteBucket with an Amazon SDK or CLI](#)
- [Use DeleteBucketAnalyticsConfiguration with a CLI](#)
- [Use DeleteBucketCors with an Amazon SDK or CLI](#)
- [Use DeleteBucketEncryption with a CLI](#)
- [Use DeleteBucketInventoryConfiguration with a CLI](#)
- [Use DeleteBucketLifecycle with an Amazon SDK or CLI](#)
- [Use DeleteBucketMetricsConfiguration with a CLI](#)
- [Use DeleteBucketPolicy with an Amazon SDK or CLI](#)
- [Use DeleteBucketReplication with a CLI](#)
- [Use DeleteBucketTagging with a CLI](#)
- [Use DeleteBucketWebsite with an Amazon SDK or CLI](#)
- [Use DeleteObject with an Amazon SDK or CLI](#)
- [Use DeleteObjectTagging with a CLI](#)
- [Use DeleteObjects with an Amazon SDK or CLI](#)
- [Use DeletePublicAccessBlock with a CLI](#)
- [Use GetBucketAccelerateConfiguration with a CLI](#)
- [Use GetBucketAcl with an Amazon SDK or CLI](#)
- [Use GetBucketAnalyticsConfiguration with a CLI](#)
- [Use GetBucketCors with an Amazon SDK or CLI](#)
- [Use GetBucketEncryption with an Amazon SDK or CLI](#)
- [Use GetBucketInventoryConfiguration with a CLI](#)
- [Use GetBucketLifecycleConfiguration with an Amazon SDK or CLI](#)
- [Use GetBucketLocation with an Amazon SDK or CLI](#)
- [Use GetBucketLogging with a CLI](#)
- [Use GetBucketMetricsConfiguration with a CLI](#)
- [Use GetBucketNotification with a CLI](#)

- [Use GetBucketPolicy with an Amazon SDK or CLI](#)
- [Use GetBucketPolicyStatus with a CLI](#)
- [Use GetBucketReplication with an Amazon SDK or CLI](#)
- [Use GetBucketRequestPayment with a CLI](#)
- [Use GetBucketTagging with a CLI](#)
- [Use GetBucketVersioning with a CLI](#)
- [Use GetBucketWebsite with an Amazon SDK or CLI](#)
- [Use GetObject with an Amazon SDK or CLI](#)
- [Use GetObjectAcl with an Amazon SDK or CLI](#)
- [Use GetObjectAttributes with an Amazon SDK or CLI](#)
- [Use GetObjectLegalHold with an Amazon SDK or CLI](#)
- [Use GetObjectLockConfiguration with an Amazon SDK or CLI](#)
- [Use GetObjectRetention with an Amazon SDK or CLI](#)
- [Use GetObjectTagging with a CLI](#)
- [Use GetPublicAccessBlock with a CLI](#)
- [Use HeadBucket with an Amazon SDK or CLI](#)
- [Use HeadObject with an Amazon SDK or CLI](#)
- [Use ListBucketAnalyticsConfigurations with a CLI](#)
- [Use ListBucketInventoryConfigurations with a CLI](#)
- [Use ListBuckets with an Amazon SDK or CLI](#)
- [Use ListMultipartUploads with an Amazon SDK or CLI](#)
- [Use ListObjectVersions with an Amazon SDK or CLI](#)
- [Use ListObjects with a CLI](#)
- [Use ListObjectsV2 with an Amazon SDK or CLI](#)
- [Use PutBucketAccelerateConfiguration with an Amazon SDK or CLI](#)
- [Use PutBucketAcl with an Amazon SDK or CLI](#)
- [Use PutBucketCors with an Amazon SDK or CLI](#)
- [Use PutBucketEncryption with an Amazon SDK or CLI](#)
- [Use PutBucketLifecycleConfiguration with an Amazon SDK or CLI](#)
- [Use PutBucketLogging with an Amazon SDK or CLI](#)

- [Use PutBucketNotification with a CLI](#)
- [Use PutBucketNotificationConfiguration with an Amazon SDK or CLI](#)
- [Use PutBucketPolicy with an Amazon SDK or CLI](#)
- [Use PutBucketReplication with an Amazon SDK or CLI](#)
- [Use PutBucketRequestPayment with a CLI](#)
- [Use PutBucketTagging with a CLI](#)
- [Use PutBucketVersioning with an Amazon SDK or CLI](#)
- [Use PutBucketWebsite with an Amazon SDK or CLI](#)
- [Use PutObject with an Amazon SDK or CLI](#)
- [Use PutObjectAcl with an Amazon SDK or CLI](#)
- [Use PutObjectLegalHold with an Amazon SDK or CLI](#)
- [Use PutObjectLockConfiguration with an Amazon SDK or CLI](#)
- [Use PutObjectRetention with an Amazon SDK or CLI](#)
- [Use RestoreObject with an Amazon SDK or CLI](#)
- [Use SelectObjectContent with an Amazon SDK or CLI](#)
- [Use UploadPart with an Amazon SDK or CLI](#)
- [Use UploadPartCopy with an Amazon SDK or CLI](#)
- [Scenarios for Amazon S3 using Amazon SDKs](#)
 - [Check if a bucket exists](#)
 - [Convert text to speech and back to text using an Amazon SDK](#)
 - [Create a presigned URL for Amazon S3 using an Amazon SDK](#)
 - [Create a photo asset management application that lets users manage photos using labels](#)
 - [A web page that lists Amazon S3 objects using an Amazon SDK](#)
 - [Create an Amazon Textract explorer application](#)
 - [Delete all objects in a given Amazon S3 bucket using an Amazon SDK](#)
 - [Delete incomplete multipart uploads to Amazon S3 using an Amazon SDK](#)
 - [Detect PPE in images with Amazon Rekognition using an Amazon SDK](#)
 - [Detect entities in text extracted from an image using an Amazon SDK](#)
 - [Detect faces in an image using an Amazon SDK](#)
 - [Detect objects in images with Amazon Rekognition using an Amazon SDK](#)

- [Detect people and objects in a video with Amazon Rekognition using an Amazon SDK](#)
- [Download S3 'directories' from an Amazon Simple Storage Service \(Amazon S3\) bucket](#)
- [Download all objects in an Amazon Simple Storage Service \(Amazon S3\) bucket to a local directory](#)
- [Download a stream of unknown size from an Amazon S3 object using an Amazon SDK](#)
- [Get an Amazon S3 object from a Multi-Region Access Point by using an Amazon SDK](#)
- [Get an object from an Amazon S3 bucket using an Amazon SDK, specifying an If-Modified-Since header](#)
- [Get started with encryption for Amazon S3 objects using an Amazon SDK](#)
- [Get started with tags for Amazon S3 objects using an Amazon SDK](#)
- [Getting started with Amazon Athena](#)
- [Getting started with Amazon EMR](#)
- [Getting started with Amazon S3](#)
- [Getting started with Amazon SageMaker Feature Store](#)
- [Getting started with Amazon Textract](#)
- [Getting started with Config](#)
- [Work with Amazon S3 object lock features using an Amazon SDK](#)
- [Make Amazon S3 conditional requests using an Amazon SDK](#)
- [Manage access control lists \(ACLs\) for Amazon S3 buckets using an Amazon SDK](#)
- [Manage large Amazon SQS messages using Amazon S3 with an Amazon SDK](#)
- [Manage versioned Amazon S3 objects in batches with a Lambda function using an Amazon SDK](#)
- [Parse Amazon S3 URIs using an Amazon SDK](#)
- [Perform a multipart copy of an Amazon S3 object using an Amazon SDK](#)
- [Receive and process Amazon S3 event notifications by using an Amazon SDK](#)
- [Save EXIF and other image information using an Amazon SDK](#)
- [Send S3 event notifications to Amazon EventBridge using an Amazon SDK](#)
- [Track an Amazon S3 object upload or download using an Amazon SDK](#)
- [Transform data for your application with S3 Object Lambda](#)
- [Example approaches for unit and integration testing with an Amazon SDK](#)
- [Recursively upload a local directory to an Amazon Simple Storage Service \(Amazon S3\) bucket](#)

- [Upload or download large files to and from Amazon S3 using an Amazon SDK](#)
- [Upload a stream of unknown size to an Amazon S3 object using an Amazon SDK](#)
- [Use checksums to work with an Amazon S3 object using an Amazon SDK](#)
- [Work with Amazon S3 object integrity features using an Amazon SDK](#)
- [Work with Amazon S3 versioned objects using an Amazon SDK](#)
- [Serverless examples for Amazon S3](#)
 - [Invoke a Lambda function from an Amazon S3 trigger](#)

Basic examples for Amazon S3 using Amazon SDKs

The following code examples show how to use the basics of Amazon Simple Storage Service with Amazon SDKs.

Examples

- [Hello Amazon S3](#)
- [Learn the basics of Amazon S3 with an Amazon SDK](#)
- [Actions for Amazon S3 using Amazon SDKs](#)
 - [Use AbortMultipartUpload with an Amazon SDK or CLI](#)
 - [Use CompleteMultipartUpload with an Amazon SDK or CLI](#)
 - [Use CopyObject with an Amazon SDK or CLI](#)
 - [Use CreateBucket with an Amazon SDK or CLI](#)
 - [Use CreateMultiRegionAccessPoint with an Amazon SDK](#)
 - [Use CreateMultipartUpload with an Amazon SDK or CLI](#)
 - [Use CreatePresignedPost with an Amazon SDK](#)
 - [Use DeleteBucket with an Amazon SDK or CLI](#)
 - [Use DeleteBucketAnalyticsConfiguration with a CLI](#)
 - [Use DeleteBucketCors with an Amazon SDK or CLI](#)
 - [Use DeleteBucketEncryption with a CLI](#)
 - [Use DeleteBucketInventoryConfiguration with a CLI](#)
 - [Use DeleteBucketLifecycle with an Amazon SDK or CLI](#)
 - [Use DeleteBucketMetricsConfiguration with a CLI](#)
 - [Use DeleteBucketPolicy with an Amazon SDK or CLI](#)

- [Use DeleteBucketReplication with a CLI](#)
- [Use DeleteBucketTagging with a CLI](#)
- [Use DeleteBucketWebsite with an Amazon SDK or CLI](#)
- [Use DeleteObject with an Amazon SDK or CLI](#)
- [Use DeleteObjectTagging with a CLI](#)
- [Use DeleteObjects with an Amazon SDK or CLI](#)
- [Use DeletePublicAccessBlock with a CLI](#)
- [Use GetBucketAccelerateConfiguration with a CLI](#)
- [Use GetBucketAcl with an Amazon SDK or CLI](#)
- [Use GetBucketAnalyticsConfiguration with a CLI](#)
- [Use GetBucketCors with an Amazon SDK or CLI](#)
- [Use GetBucketEncryption with an Amazon SDK or CLI](#)
- [Use GetBucketInventoryConfiguration with a CLI](#)
- [Use GetBucketLifecycleConfiguration with an Amazon SDK or CLI](#)
- [Use GetBucketLocation with an Amazon SDK or CLI](#)
- [Use GetBucketLogging with a CLI](#)
- [Use GetBucketMetricsConfiguration with a CLI](#)
- [Use GetBucketNotification with a CLI](#)
- [Use GetBucketPolicy with an Amazon SDK or CLI](#)
- [Use GetBucketPolicyStatus with a CLI](#)
- [Use GetBucketReplication with an Amazon SDK or CLI](#)
- [Use GetBucketRequestPayment with a CLI](#)
- [Use GetBucketTagging with a CLI](#)
- [Use GetBucketVersioning with a CLI](#)
- [Use GetBucketWebsite with an Amazon SDK or CLI](#)
- [Use GetObject with an Amazon SDK or CLI](#)
- [Use GetObjectAcl with an Amazon SDK or CLI](#)
- [Use GetObjectAttributes with an Amazon SDK or CLI](#)
- [Use GetObjectLegalHold with an Amazon SDK or CLI](#)
- [Use GetObjectLockConfiguration with an Amazon SDK or CLI](#)

- [Use GetObjectRetention with an Amazon SDK or CLI](#)
- [Use GetObjectTagging with a CLI](#)
- [Use GetPublicAccessBlock with a CLI](#)
- [Use HeadBucket with an Amazon SDK or CLI](#)
- [Use HeadObject with an Amazon SDK or CLI](#)
- [Use ListBucketAnalyticsConfigurations with a CLI](#)
- [Use ListBucketInventoryConfigurations with a CLI](#)
- [Use ListBuckets with an Amazon SDK or CLI](#)
- [Use ListMultipartUploads with an Amazon SDK or CLI](#)
- [Use ListObjectVersions with an Amazon SDK or CLI](#)
- [Use ListObjects with a CLI](#)
- [Use ListObjectsV2 with an Amazon SDK or CLI](#)
- [Use PutBucketAccelerateConfiguration with an Amazon SDK or CLI](#)
- [Use PutBucketAcl with an Amazon SDK or CLI](#)
- [Use PutBucketCors with an Amazon SDK or CLI](#)
- [Use PutBucketEncryption with an Amazon SDK or CLI](#)
- [Use PutBucketLifecycleConfiguration with an Amazon SDK or CLI](#)
- [Use PutBucketLogging with an Amazon SDK or CLI](#)
- [Use PutBucketNotification with a CLI](#)
- [Use PutBucketNotificationConfiguration with an Amazon SDK or CLI](#)
- [Use PutBucketPolicy with an Amazon SDK or CLI](#)
- [Use PutBucketReplication with an Amazon SDK or CLI](#)
- [Use PutBucketRequestPayment with a CLI](#)
- [Use PutBucketTagging with a CLI](#)
- [Use PutBucketVersioning with an Amazon SDK or CLI](#)
- [Use PutBucketWebsite with an Amazon SDK or CLI](#)
- [Use PutObject with an Amazon SDK or CLI](#)
- [Use PutObjectAcl with an Amazon SDK or CLI](#)
- [Use PutObjectLegalHold with an Amazon SDK or CLI](#)
- [Use PutObjectLockConfiguration with an Amazon SDK or CLI](#)

- [Use PutObjectRetention with an Amazon SDK or CLI](#)
- [Use RestoreObject with an Amazon SDK or CLI](#)
- [Use SelectObjectContent with an Amazon SDK or CLI](#)
- [Use UploadPart with an Amazon SDK or CLI](#)
- [Use UploadPartCopy with an Amazon SDK or CLI](#)

Hello Amazon S3

The following code examples show how to get started using Amazon S3.

.NET

Amazon SDK for .NET (v4)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Hello Amazon Simple Storage Service
/// (Amazon S3) example.
/// </summary>
public class HelloS3
{
    /// <summary>
    /// Main method to run the Hello S3 example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var s3Client = new AmazonS3Client();

        try
        {
            Console.WriteLine("Hello Amazon S3! Let's list your buckets:");
            Console.WriteLine(new string('-', 80));
        }
    }
}
```

```
// Use the built-in paginator to list buckets
var request = new ListBucketsRequest();
var paginator = s3Client.Paginators.ListBuckets(request);

var buckets = new List<S3Bucket>();

await foreach (var response in paginator.Responses)
{
    buckets.AddRange(response.Buckets);
}

if (buckets.Any())
{
    Console.WriteLine($"Found {buckets.Count} S3 buckets:");
    Console.WriteLine();

    foreach (var bucket in buckets)
    {
        Console.WriteLine($"- Bucket Name: {bucket.BucketName}");
        Console.WriteLine($"  Creation Date:
{bucket.CreationDate:yyyy-MM-dd HH:mm:ss UTC}");
        Console.WriteLine();
    }
}
else
{
    Console.WriteLine("No S3 buckets found in your account.");
}

Console.WriteLine("Hello S3 completed successfully.");
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"S3 service error occurred: {ex.Message}");
}
catch (Exception ex)
{
    Console.WriteLine($"Couldn't list S3 buckets. Here's why:
{ex.Message}");
}
}
```

- For API details, see [ListBuckets](#) in *Amazon SDK for .NET API Reference*.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Code for the CMakeLists.txt CMake file.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
    libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
        "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
```

```

    # Copy relevant AWS SDK for C++ libraries into the current binary directory
    for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
    may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

Code for the hello_s3.cpp source file.

```

#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 * A "Hello S3" starter application which initializes an Amazon Simple Storage
 * Service (Amazon S3) client
 * and lists the Amazon S3 buckets in the selected region.
 *
 * main function
 *
 * Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
}

```

```

int result = 0;
{
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    // You don't normally have to test that you are authenticated. But the
    S3 service permits anonymous requests, thus the s3Client will return "success"
    and 0 buckets even if you are unauthenticated, which can be confusing to a new
    user.

    auto provider =
    Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-tag");
    auto creds = provider->GetAWSCredentials();
    if (creds.IsEmpty()) {
        std::cerr << "Failed authentication" << std::endl;
    }

    Aws::S3::S3Client s3Client(clientConfig);
    auto outcome = s3Client.ListBuckets();

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() <<
std::endl;
        result = 1;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size()
        << " buckets\n";
        for (auto &bucket: outcome.GetResult().GetBuckets()) {
            std::cout << bucket.GetName() << std::endl;
        }
    }
}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

- For API details, see [ListBuckets](#) in *Amazon SDK for C++ API Reference*.

Go

SDK for Go V2

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
package main

import (
    "context"
    "errors"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    s3Client := s3.NewFromConfig(sdkConfig)
    count := 10
    fmt.Printf("Let's list up to %v buckets for your account.\n", count)
    result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
    if err != nil {
        var ae smithy.APIError
```

```

if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
    fmt.Println("You don't have permission to list buckets for this account.")
} else {
    fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
}
return
}
if len(result.Buckets) == 0 {
    fmt.Println("You don't have any buckets!")
} else {
    if count > len(result.Buckets) {
        count = len(result.Buckets)
    }
    for _, bucket := range result.Buckets[:count] {
        fmt.Printf("\t%v\n", *bucket.Name)
    }
}
}
}

```

- For API details, see [ListBuckets](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
* <p>
* For more information, see the following documentation topic:
* <p>
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    /**
     * Lists all the S3 buckets associated with the provided AWS S3 client.
     *
     * @param s3 the S3Client instance used to interact with the AWS S3 service
     */
    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [ListBuckets](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the S3 buckets in your configured AWS account.
 */
export const helloS3 = async () => {
  // When no region or credentials are provided, the SDK will use the
  // region and credentials from the local AWS config.
  const client = new S3Client({});

  try {
    /**
     * @type { import("@aws-sdk/client-s3").Bucket[] }
     */
    const buckets = [];

    for await (const page of paginateListBuckets({ client }, {})) {
      buckets.push(...page.Buckets);
    }
    console.log("Buckets: ");
    console.log(buckets.map((bucket) => bucket.Name).join("\n"));
    return buckets;
  } catch (caught) {
    // ListBuckets does not throw any modeled errors. Any error caught
    // here will be something generic like `AccessDenied`.
    if (caught instanceof S3ServiceException) {
      console.error(`${caught.name}: ${caught.message}`);
    } else {
```

```
        // Something besides S3 failed.  
        throw caught;  
    }  
}  
};
```

- For API details, see [ListBuckets](#) in *Amazon SDK for JavaScript API Reference*.

PHP

SDK for PHP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
use Aws\S3\S3Client;  
  
$client = new S3Client(['region' => 'us-west-2']);  
$results = $client->listBuckets();  
var_dump($results);
```

- For API details, see [ListBuckets](#) in *Amazon SDK for PHP API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import boto3
```

```
def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) client and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """

    # Create an S3 client.
    s3_client = boto3.client("s3")

    print("Hello, Amazon S3! Let's list your buckets:")

    # Create a paginator for the list_buckets operation.
    paginator = s3_client.get_paginator("list_buckets")

    # Use the paginator to get a list of all buckets.
    response_iterator = paginator.paginate(
        PaginationConfig={
            "PageSize": 50, # Adjust PageSize as needed.
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response.
    buckets_found = False
    for page in response_iterator:
        if "Buckets" in page and page["Buckets"]:
            buckets_found = True
            for bucket in page["Buckets"]:
                print(f"\t{bucket['Name']}")

    if not buckets_found:
        print("No buckets found!")

if __name__ == "__main__":
    hello_s3()
```

- For API details, see [ListBuckets](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets

    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end

    rescue Aws::Errors::ServiceError => e
      @logger.error("Encountered an error while listing buckets: #{e.message}")
    end
  end

  if $PROGRAM_NAME == __FILE__
    s3_client = Aws::S3::Client.new
    manager = S3Manager.new(s3_client)
  end
end
```

```
manager.list_buckets
end
```

- For API details, see [ListBuckets](#) in *Amazon SDK for Ruby API Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// S3 Hello World Example using the AWS SDK for Rust.
///
/// This example lists the objects in a bucket, uploads an object to that bucket,
/// and then retrieves the object and prints some S3 information about the
/// object.
/// This shows a number of S3 features, including how to use built-in paginators
/// for large data sets.
///
/// # Arguments
///
/// * `client` - an S3 client configured appropriately for the environment.
/// * `bucket` - the bucket name that the object will be uploaded to. Must be
///   present in the region the `client` is configured to use.
/// * `filename` - a reference to a path that will be read and uploaded to S3.
/// * `key` - the string key that the object will be uploaded as inside the
///   bucket.
async fn list_bucket_and_upload_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    filepath: &Path,
    key: &str,
) -> Result<(), S3ExampleError> {
    // List the buckets in this account
    let mut objects = client
        .list_objects_v2()
```



```

        .bucket(bucket)
        .into_paginator()
        .send();

println!("key\tetag\tlast_modified\tstorage_class");
while let Some(Ok(object)) = objects.next().await {
    for item in object.contents() {
        println!(
            "{}\t{}\t{}\t{}",
            item.key().unwrap_or_default(),
            item.e_tag().unwrap_or_default(),
            item.last_modified()
                .map(|lm| format!("{lm}"))
                .unwrap_or_default(),
            item.storage_class()
                .map(|sc| format!("{sc}"))
                .unwrap_or_default()
        );
    }
}

// Prepare a ByteStream around the file, and upload the object using that
// ByteStream.
let body = aws_sdk_s3::primitives::ByteStream::from_path(filepath)
    .await
    .map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to create bytestream for {filepath:?} ({err:?})"
        ))
    })?;
let resp = client
    .put_object()
    .bucket(bucket)
    .key(key)
    .body(body)
    .send()
    .await?;

println!(
    "Upload success. Version: {:?}",
    resp.version_id()
        .expect("S3 Object upload missing version ID")
);

```

```
// Retrieve the just-uploaded object.
let resp = client.get_object().bucket(bucket).key(key).send().await?;
println!("etag: {}", resp.e_tag().unwrap_or("(missing)"));
println!("version: {}", resp.version_id().unwrap_or("(missing)"));

Ok(())
}
```

S3ExampleError utilities.

```
/// S3ExampleError provides a From<T: ProvideErrorMetadata> impl to extract
/// client-specific error details. This serves as a consistent backup to handling
/// specific service errors, depending on what is needed by the scenario.
/// It is used throughout the code examples for the AWS SDK for Rust.
#[derive(Debug)]
pub struct S3ExampleError(String);
impl S3ExampleError {
    pub fn new(value: impl Into<String>) -> Self {
        S3ExampleError(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        S3ExampleError(format!("{}: {}", message.into(), self.0))
    }
}

impl<T: aws_sdk_s3::error::ProvideErrorMetadata> From<T> for S3ExampleError {
    fn from(value: T) -> Self {
        S3ExampleError(format!(
            "{}: {}",
            value
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
                .map(String::from)
                .unwrap_or("missing reason".into()),
        ))
    }
}
```

```
impl std::error::Error for S3ExampleError {}

impl std::fmt::Display for S3ExampleError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
```

- For API details, see [ListBuckets](#) in *Amazon SDK for Rust API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Learn the basics of Amazon S3 with an Amazon SDK

The following code examples show how to:

- Create a bucket and upload a file to it.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the bucket objects and the bucket.

.NET

Amazon SDK for .NET (v4)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Run an interactive scenario demonstrating Amazon S3 features.

```
public class S3_Basics
{
```

```
public static bool IsInteractive = true;
public static string BucketName = null!;
public static string TempFilePath = null!;
public static S3Wrapper _s3Wrapper = null!;
public static ILogger<S3_Basics> _logger = null!;

public static async Task Main(string[] args)
{
    // Set up dependency injection for the Amazon service.
    using var host = Host.CreateDefaultBuilder(args)
        .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonS3>()
                .AddTransient<S3Wrapper>()
                .AddLogging(builder => builder.AddConsole()))
        .Build();

    _logger = LoggerFactory.Create(builder => builder.AddConsole())
        .CreateLogger<S3_Basics>();

    _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();

    var sepBar = new string('-', 45);

    Console.WriteLine(sepBar);
    Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
    Console.WriteLine("procedures. This application will:");
    Console.WriteLine("\n\t1. Create a bucket");
    Console.WriteLine("\n\t2. Upload an object to the new bucket");
    Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
bucket");
    Console.WriteLine("\n\t4. List the items in the new bucket");
    Console.WriteLine("\n\t5. Delete all the items in the bucket");
    Console.WriteLine("\n\t6. Delete the bucket");
    Console.WriteLine(sepBar);

    await RunScenario(_s3Wrapper, _logger);

    Console.WriteLine(sepBar);
    Console.WriteLine("The Amazon S3 scenario has successfully completed.");
    Console.WriteLine(sepBar);
}

/// <summary>
/// Run the S3 Basics scenario with injected dependencies.
```

```
/// </summary>
/// <param name="s3Wrapper">The S3 wrapper instance.</param>
/// <param name="scenarioLogger">The logger instance.</param>
/// <returns>A Task object.</returns>
public static async Task RunScenario(S3Wrapper s3Wrapper, ILogger<S3_Basics>
scenarioLogger)
{
    string bucketName = BucketName;
    string filePath = TempFilePath;
    string keyName = string.Empty;

    var sepBar = new string('-', 45);

    try
    {
        // Create a bucket.
        Console.WriteLine($"\\n{sepBar}");
        Console.WriteLine("\\nCreate a new Amazon S3 bucket.\\n");
        Console.WriteLine(sepBar);

        if (IsInteractive)
        {
            Console.Write("Please enter a name for the new bucket: ");
            bucketName = Console.ReadLine();
        }
        else
        {
            Console.WriteLine($"Using bucket name: {bucketName}");
        }

        var success = await s3Wrapper.CreateBucketAsync(bucketName);
        if (success)
        {
            Console.WriteLine($"Successfully created bucket: {bucketName}.
\\n");
        }
        else
        {
            Console.WriteLine($"Could not create bucket: {bucketName}.\\n");
        }

        Console.WriteLine(sepBar);
        Console.WriteLine("Upload a file to the new bucket.");
        Console.WriteLine(sepBar);
    }
}
```

```
        if (IsInteractive)
        {
            // Get the local path and filename for the file to upload.
            while (string.IsNullOrEmpty(filePath))
            {
                Console.WriteLine("Please enter the path and filename of the file
to upload: ");
                filePath = Console.ReadLine();

                // Confirm that the file exists on the local computer.
                if (!File.Exists(filePath))
                {
                    Console.WriteLine($"Couldn't find {filePath}. Try again.
\n");
                    filePath = string.Empty;
                }
            }
        }
        else
        {
            // Use the public variable if set, otherwise create a temp file
            if (!string.IsNullOrEmpty(TempFilePath))
            {
                filePath = TempFilePath;
                Console.WriteLine($"Using provided test file: {filePath}");
            }
            else
            {
                // Create a temporary test file for non-interactive mode
                filePath = Path.GetTempFileName();
                var testContent = "This is a test file for S3 basics
scenario.\nGenerated on: " + DateTime.UtcNow.ToString("yyyy-MM-dd HH:mm:ss
UTC");

                await File.WriteAllTextAsync(filePath, testContent);
                Console.WriteLine($"Created temporary test file:
{filePath}");
            }
        }

        // Get the file name from the full path.
        keyName = Path.GetFileName(filePath);
```

```
        success = await s3Wrapper.UploadFileAsync(bucketName, keyName,
filePath);

        if (success)
        {
            Console.WriteLine($"Successfully uploaded {keyName} from
{filePath} to {bucketName}.\n");
        }
        else
        {
            Console.WriteLine($"Could not upload {keyName}.\n");
        }

        // Set up download path
        string downloadPath = string.Empty;

        if (IsInteractive)
        {
            // Now get a new location where we can save the file.
            while (string.IsNullOrEmpty(downloadPath))
            {
                // First get the path to which the file will be downloaded.
                Console.Write("Please enter the path where the file will be
downloaded: ");
                downloadPath = Console.ReadLine();

                // Confirm that the file doesn't already exist on the local
computer.
                if (File.Exists($"{downloadPath}\\{keyName}"))
                {
                    Console.WriteLine($"Sorry, the file already exists in
that location.\n");
                    downloadPath = string.Empty;
                }
            }
        }
        else
        {
            downloadPath = Path.GetTempPath();
            var downloadFile = Path.Combine(downloadPath, keyName);
            if (File.Exists(downloadFile))
            {
                File.Delete(downloadFile);
            }
        }
    }
}
```

```
        Console.WriteLine($"Using download path: {downloadPath}");
    }

    // Download an object from a bucket.
    success = await s3Wrapper.DownloadObjectFromBucketAsync(bucketName,
keyName, downloadPath);

    if (success)
    {
        Console.WriteLine($"Successfully downloaded {keyName}.\n");
    }
    else
    {
        Console.WriteLine($"Sorry, could not download {keyName}.\n");
    }

    // Copy the object to a different folder in the bucket.
    string folderName = string.Empty;

    if (IsInteractive)
    {
        while (string.IsNullOrEmpty(folderName))
        {
            Console.Write("Please enter the name of the folder to copy
your object to: ");
            folderName = Console.ReadLine();
        }
    }
    else
    {
        folderName = "test-folder";
        Console.WriteLine($"Using folder name: {folderName}");
    }

    await s3Wrapper.CopyObjectInBucketAsync(bucketName, keyName,
folderName);

    // List the objects in the bucket.
    await s3Wrapper.ListBucketContentsAsync(bucketName);

    // Delete the contents of the bucket.
    if (IsInteractive)
    {
```



```
        Console.WriteLine("Press <Enter> when you are ready to delete the
bucket contents.");
        _ = Console.ReadLine();
    }

    var deleteContentsSuccess = await
s3Wrapper.DeleteBucketContentsAsync(bucketName);
    if (deleteContentsSuccess)
    {
        Console.WriteLine($"Successfully deleted contents of
{bucketName}.\n");
    }
    else
    {
        Console.WriteLine($"Sorry, could not delete contents of
{bucketName}.\n");
    }

    if (IsInteractive)
    {
        // Deleting the bucket too quickly after separately deleting its
contents can
        // cause an error that the bucket isn't empty. To delete contents
and bucket in one
        // operation, use AmazonS3Util.DeleteS3BucketWithObjectsAsync
        Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
        _ = Console.ReadLine();
    }
    else
    {
        // Add a small delay for non-interactive mode to ensure objects
are fully deleted.
        Console.WriteLine("Waiting a moment for objects to be fully
deleted...");
        await Task.Delay(2000);
    }

    // Delete the bucket.
    var deleteSuccess = await s3Wrapper.DeleteBucketAsync(bucketName);
    if (deleteSuccess)
    {
        Console.WriteLine($"Successfully deleted {bucketName}.\n");
    }
}
```

```
        else
        {
            Console.WriteLine($"Sorry, could not delete {bucketName}.\n");
        }

        // Clean up temporary files in non-interactive mode
        if (!IsInteractive)
        {
            try
            {
                if (File.Exists(filePath))
                {
                    File.Delete(filePath);
                    Console.WriteLine("Cleaned up temporary test file.");
                }

                var downloadFile = Path.Combine(downloadPath, keyName);
                if (File.Exists(downloadFile))
                {
                    File.Delete(downloadFile);
                    Console.WriteLine("Cleaned up downloaded test file.");
                }
            }
            catch (Exception ex)
            {
                scenarioLogger.LogWarning(ex, "Failed to clean up temporary
files.");
            }
        }
    }
    catch (Exception ex)
    {
        scenarioLogger.LogError(ex, "An error occurred during the S3 scenario
execution.");

        // Clean up on error - delete bucket if it exists
        try
        {
            if (!string.IsNullOrEmpty(bucketName))
            {
                await s3Wrapper.DeleteBucketContentsAsync(bucketName);
                await s3Wrapper.DeleteBucketAsync(bucketName);
            }
        }
    }
}
```

```

        catch (Exception cleanupEx)
        {
            scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
        }

        // Clean up temporary files in non-interactive mode
        if (!IsInteractive)
        {
            try
            {
                if (!string.IsNullOrEmpty(filePath) && File.Exists(filePath))
                {
                    File.Delete(filePath);
                }
            }
            catch (Exception fileCleanupEx)
            {
                scenarioLogger.LogWarning(fileCleanupEx, "Failed to clean up
temporary files during error handling.");
            }
        }

        throw;
    }
}

```

A wrapper class for Amazon S3 SDK methods.

```

using Amazon.S3;
using Amazon.S3.Model;

namespace S3_Actions;

/// <summary>
/// This class contains all of the methods for working with Amazon Simple
/// Storage Service (Amazon S3) buckets.
/// </summary>
public class S3Wrapper
{
    private readonly IAmazonS3 _amazonS3;

```

```
/// <summary>
/// Initializes a new instance of the <see cref="S3Wrapper"/> class.
/// </summary>
/// <param name="amazonS3">An initialized Amazon S3 client object.</param>
public S3Wrapper(IAmazonS3 amazonS3)
{
    _amazonS3 = amazonS3;
}

/// <summary>
/// Shows how to create a new Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>A boolean value representing the success or failure of
/// the bucket creation process.</returns>
public async Task<bool> CreateBucketAsync(string bucketName)
{
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
        };

        var response = await _amazonS3.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to which the object
```

```

    /// will be uploaded.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object
    /// on the local computer to upload.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// upload procedure.</returns>
    public async Task<bool> UploadFileAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        try
        {
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await _amazonS3.PutObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Shows how to download an object from an Amazon S3 bucket to the
    /// local computer.
    /// </summary>
    /// <param name="bucketName">The name of the bucket where the object is
    /// currently stored.</param>
    /// <param name="objectName">The name of the object to download.</param>
    /// <param name="filePath">The path, including filename, where the
    /// downloaded object will be stored.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// download process.</returns>
    public async Task<bool> DownloadObjectFromBucketAsync(

```

```
        string bucketName,
        string objectName,
        string filePath)
    {
        var request = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
        };

        using GetObjectResponse response = await
            _amazonS3.GetObjectAsync(request);

        try
        {
            // Save object to local file
            await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error saving {objectName}: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Copies an object in an Amazon S3 bucket to a folder within the
    /// same bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object to copy is located.</param>
    /// <param name="objectName">The object to be copied.</param>
    /// <param name="folderName">The folder to which the object will
    /// be copied.</param>
    /// <returns>A boolean value that indicates the success or failure of
    /// the copy operation.</returns>
    public async Task<bool> CopyObjectInBucketAsync(
        string bucketName,
        string objectName,
        string folderName)
```

```

    {
        try
        {
            var request = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }
}

/// <summary>
/// Shows how to list the objects in an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket for which to list.
/// <param name="printList">True to print out the list.
/// <returns>The collection of objects.</returns>
public async Task<List<S3Object?>> ListBucketContentsAsync(string bucketName,
bool printList = true)
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 5,
        };

        if (printList)
        {
            Console.WriteLine("-----");
            Console.WriteLine($"Listing the contents of {bucketName}:");
            Console.WriteLine("-----");
        }
    }
}

```

```

    }

    var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
    {
        BucketName = bucketName,
    });
    var s3Objects = new List<S3Object>();
    await foreach (var response in listObjectsV2Paginator.Responses)
    {
        if (response.S3Objects != null)
        {
            s3Objects.AddRange(response.S3Objects);
        }
    }

    if (printList)
    {
        Console.WriteLine($"Number of Objects: {s3Objects.Count}");
        foreach (var entry in s3Objects)
        {
            Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
        }
    }

    return s3Objects;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' getting list of objects.");
    return null;
}
}

/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>

```



```

public async Task<bool> DeleteBucketContentsAsync(string bucketName)
{
    // Iterate over the contents of the bucket and delete all objects.
    try
    {
        // Delete all objects in the bucket.
        var deleteList = await ListBucketContentsAsync(bucketName, false);
        if (deleteList != null && deleteList.Any())
        {
            await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
            });
        }

        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting objects: {ex.Message}");
        return false;
    }
}

/// <summary>
/// Shows how to delete an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
/// <returns>A boolean value that represents the success or failure of
/// the delete operation.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest { BucketName = bucketName, };

        await _amazonS3.DeleteBucketAsync(request);
        return true;
    }
}

```

```

        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting bucket: {ex.Message}");
            return false;
        }
    }
}

```

- For API details, see the following topics in *Amazon SDK for .NET API Reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

#####
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#     0 - If successful.
#     1 - If an error occurred.

```

```
#####
function s3_getting_started() {
{
    if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
        cd bucket-lifecycle-operations || exit

        source ./bucket_operations.sh
        cd ..
    fi
}

echo_repeat "*" 88
echo "Welcome to the Amazon S3 getting started demo."
echo_repeat "*" 88
    echo "A unique bucket will be created by appending a Universally Unique
Identifier to a bucket name prefix."
    echo -n "Enter a prefix for the S3 bucket that will be used in this demo: "
    get_input
    bucket_name_prefix=$get_input_result
local bucket_name
bucket_name=$(generate_random_name "$bucket_name_prefix")

local region_code
region_code=$(aws configure get region)

if create_bucket -b "$bucket_name" -r "$region_code"; then
    echo "Created demo bucket named $bucket_name"
else
    errecho "The bucket failed to create. This demo will exit."
    return 1
fi

local file_name
while [ -z "$file_name" ]; do
    echo -n "Enter a file you want to upload to your bucket: "
    get_input
    file_name=$get_input_result

    if [ ! -f "$file_name" ]; then
        echo "Could not find file $file_name. Are you sure it exists?"
        file_name=""
    fi
done

```

```
local key
key="$(basename "$file_name")"

local result=0
if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
    echo "Uploaded file $file_name into bucket $bucket_name with key $key."
else
    result=1
fi

local destination_file
destination_file="$file_name.download"
if yes_no_input "Would you like to download $key to the file $destination_file?
(y/n) "; then
    if download_object_from_bucket "$bucket_name" "$destination_file" "$key";
then
        echo "Downloaded $key in the bucket $bucket_name to the file
$destination_file."
    else
        result=1
    fi
fi

if yes_no_input "Would you like to copy $key a new object key in your bucket?
(y/n) "; then
    local to_key
    to_key="demo/$key"
    if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
        echo "Copied $key in the bucket $bucket_name to the $to_key."
    else
        result=1
    fi
fi

local bucket_items
bucket_items=$(list_items_in_bucket "$bucket_name")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
    result=1
fi

echo "Your bucket contains the following items."
echo -e "Name\t\tSize"
```

```

echo "$bucket_items"

if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
(y/n) "; then
    bucket_items=$(echo "$bucket_items" | cut -f 1)

    if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
        echo "The following items were deleted from the bucket $bucket_name"
        echo "$bucket_items"
    else
        result=1
    fi

    if delete_bucket "$bucket_name"; then
        echo "Deleted the bucket $bucket_name"
    else
        result=1
    fi
fi

return $result
}

```

The Amazon S3 functions used in this scenario.

```

#####
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#     -b bucket_name  -- The name of the bucket to create.
#     -r region_code  -- The code for an AWS Region in which to
#                        create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####

```

```

function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function create_bucket"
        echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
        echo "  -b bucket_name    The name of the bucket. It must be globally
unique."
        echo "  [-r region_code]    The code for an AWS Region in which the bucket is
created."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;
            r) region_code="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done

    if [[ -z "$bucket_name" ]]; then
        errecho "ERROR: You must provide a bucket name with the -b parameter."
        usage
        return 1
    fi

    local bucket_config_arg
    # A location constraint for "us-east-1" returns an error.
    if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
        bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
    fi

```

```

iecho "Parameters:\n"
iecho "    Bucket name:  $bucket_name"
iecho "    Region code:  $region_code"
iecho ""

# If the bucket already exists, we don't want to try to create it.
if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
fi

# shellcheck disable=SC2086
response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
fi
}

#####
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file to.
#     $2 - The path and file name of the local file to copy to the bucket.
#     $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_file_to_bucket() {
    local response bucket_name source_file destination_file_name
    bucket_name=$1
    source_file=$2
    destination_file_name=$3

```

```

response=$(aws s3api put-object \
  --bucket "$bucket_name" \
  --body "$source_file" \
  --key "$destination_file_name")

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
  errecho "ERROR: AWS reports put-object operation failed.\n$response"
  return 1
fi
}

#####
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#   $1 - The name of the bucket to download the object from.
#   $2 - The path and file name to store the downloaded bucket.
#   $3 - The key (name) of the object in the bucket.
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#####
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

```



```
#####
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file from and to.
#     $2 - The key of the source file to copy.
#     $3 - The key of the destination file.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_item_in_bucket() {
    local bucket_name=$1
    local source_key=$2
    local destination_key=$3
    local response

    response=$(aws s3api copy-object \
        --bucket "$bucket_name" \
        --copy-source "$bucket_name/$source_key" \
        --key "$destination_key")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api copy-object operation failed.\n$response"
        return 1
    fi
}

#####
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     The list of files in text format.
```

```

#      And:
#      0 - If successful.
#      1 - If it fails.
#####
function list_items_in_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api list-objects \
        --bucket "$bucket_name" \
        --output text \
        --query 'Contents[].{Key: Key, Size: Size}')

    # shellcheck disable=SC2181
    if [[ ${?} -eq 0 ]]; then
        echo "$response"
    else
        errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
        return 1
    fi
}

#####
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - A list of keys in the bucket to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

    # Create the JSON for the items to delete.
    local delete_items
    delete_items="{\"Objects\":["
    for key in $keys; do

```

```

    delete_items="$delete_items{\"Key\": \"$key\"},"
done
delete_items=${delete_items%?} # Remove the final comma.
delete_items="$delete_items]}"

response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
    return 1
fi
}

#####
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api delete-bucket \
        --bucket "$bucket_name")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
        return 1
    fi
}

```

- For API details, see the following topics in *Amazon CLI Command Reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {
```

```

    //! Delete an S3 bucket.
    /*!
        \param bucketName: The S3 bucket's name.
        \param client: An S3 client.
        \return bool: Function succeeded.
    */
    static bool
    deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

    //! Delete an object in an S3 bucket.
    /*!
        \param bucketName: The S3 bucket's name.
        \param key: The key for the object in the S3 bucket.
        \param client: An S3 client.
        \return bool: Function succeeded.
    */
    static bool
    deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
&key,
                           Aws::S3::S3Client &client);
}

//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
    \param bucketNamePrefix: A prefix for a bucket name.
    \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
    \param saveFilePath: Path for saving a downloaded S3 object.
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &bucketNamePrefix,
      const Aws::String &uploadFilePath,
      const Aws::String &saveFilePath,
      const Aws::Client::ClientConfiguration
&clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: <bucketNamePrefix> + "-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();

```

```

    Aws::String bucketName = bucketNamePrefix +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(

Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
            request.WithCreateBucketConfiguration(createBucketConfiguration);
        }

        Aws::S3::Model::CreateBucketOutcome outcome =
client.CreateBucket(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: createBucket: " <<
                err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
            return false;
        } else {
            std::cout << "Created the bucket, '" << bucketName <<
                "'", in the region, '" << clientConfig.region << "'." <<
std::endl;
        }
    }

    // 2. Upload a local file to the bucket.
    Aws::String key = "key-for-test";
    {
        Aws::S3::Model::PutObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        std::shared_ptr<Aws::FStream> input_data =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                uploadFilePath,
                std::ios_base::in |

```

```

        std::ios_base::binary);

    if (!input_data->is_open()) {
        std::cerr << "Error: unable to open file, '" << uploadFilePath <<
        "'."
        << std::endl;
        AwsDoc::S3::deleteBucket(bucketName, client);
        return false;
    }

    request.SetBody(input_data);

    Aws::S3::Model::PutObjectOutcome outcome =
        client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
            outcome.GetError().GetMessage() << std::endl;
        AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
        AwsDoc::S3::deleteBucket(bucketName, client);
        return false;
    } else {
        std::cout << "Added the object with the key, '" << key
            << "', to the bucket, '"
            << bucketName << "'." << std::endl;
    }
}

// 3. Download the object to a local file.
{
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
        std::endl;
    } else {
        std::cout << "Downloaded the object with the key, '" << key

```

```

        << "'", in the bucket, '"
        << bucketName << "'.'" << std::endl;

    Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
        GetBody();
    Aws::OStream outStream(saveFilePath,
                           std::ios_base::out | std::ios_base::binary);
    if (!outStream.is_open()) {
        std::cout << "Error: unable to open file, '" << saveFilePath <<
        "'.'"
        << std::endl;
    } else {
        outStream << ioStream.rdbuf();
        std::cout << "Wrote the downloaded object to the file '"
        << saveFilePath << "'.'" << std::endl;
    }
}

// 4. Copy the object to a different "folder" in the bucket.
Aws::String copiedToKey = "test-folder/" + key;
{
    Aws::S3::Model::CopyObjectRequest request;
    request.WithBucket(bucketName)
        .WithKey(copiedToKey)
        .WithCopySource(bucketName + "/" + key);

    Aws::S3::Model::CopyObjectOutcome outcome =
        client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error: copyObject: " <<
        outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Copied the object with the key, '" << key
        << "'", to the key, '" << copiedToKey
        << "', in the bucket, '" << bucketName << "'.'" << std::endl;
    }
}

// 5. List objects in the bucket.
{
    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

```



```

    Aws::String continuationToken;
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }
        Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
            request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: ListObjects: " <<
                outcome.GetError().GetMessage() << std::endl;
            break;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                outcome.GetResult().GetContents();
            allObjects.insert(allObjects.end(), objects.begin(),
objects.end());
            continuationToken = outcome.GetResult().GetContinuationToken();
        }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " objects in the bucket, " <<
bucketName
        << ":" << std::endl;

    for (Aws::S3::Model::Object &object: allObjects) {
        std::cout << "    " << object.GetKey() << " " << std::endl;
    }
}

// 6. Delete all objects in the bucket.
// All objects in the bucket must be deleted before deleting the bucket.
AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

// 7. Delete the bucket.
return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {

```

```

    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Deleted the object with the key, '" << key
            << "', from the bucket, '"
            << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client
    &client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
    }
    return outcome.IsSuccess();
}

```

- For API details, see the following topics in *Amazon SDK for C++ API Reference*.
 - [CopyObject](#)
 - [CreateBucket](#)

- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Define a struct that wraps bucket and object actions used by the scenario.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.
```

```
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
    error) {
    var err error
    var output *s3.ListBucketsOutput
    var buckets []types.Bucket
    bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
        &s3.ListBucketsInput{})
    for bucketPaginator.HasMorePages() {
        output, err = bucketPaginator.NextPage(ctx)
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
                fmt.Println("You don't have permission to list buckets for this account.")
                err = apiErr
            } else {
                log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
            }
            break
        } else {
            buckets = append(buckets, output.Buckets...)
        }
    }
    return buckets, err
}

// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
    (bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
    exists := true
}
```

```

if err != nil {
    var apiError smithy.APIError
    if errors.As(err, &apiError) {
        switch apiError.(type) {
            case *types.NotFound:
                log.Printf("Bucket %v is available.\n", bucketName)
                exists = false
                err = nil
            default:
                log.Printf("Either you don't have access to bucket %v or another error
occurred. "+
                    "Here's what happened: %v\n", bucketName, err)
        }
    }
} else {
    log.Printf("Bucket %v exists and you already own it.", bucketName)
}

return exists, err
}

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
string) error {
    _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
        Bucket: aws.String(name),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    })
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", name)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", name)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(

```

```

    ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
if err != nil {
    log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
}
}
return err
}

// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
        log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
    } else {
        defer file.Close()
        _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
            Body:   file,
        })
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
                log.Printf("Error while uploading object to %s. The object is too large.\n"+
                    "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                    "or the multipart upload API (5TB max).", bucketName)
            } else {
                log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
                    fileName, bucketName, objectKey, err)
            }
        } else {
            err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectKey)}, time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
            }
        }
    }
    return err
}

```

```

// UploadLargeObject uses an upload manager to upload data to an object in a
// bucket.
// The upload manager breaks large data into parts and uploads the parts
// concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
    string, objectKey string, largeObject []byte) error {
    largeBuffer := bytes.NewReader(largeObject)
    var partMiBs int64 = 10
    uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
        u.PartSize = partMiBs * 1024 * 1024
    })
    _, err := uploader.Upload(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
        Body:    largeBuffer,
    })
    if err != nil {
        var apiErr smithy.APIError
        if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
            log.Printf("Error while uploading object to %s. The object is too large.\n"+
                "The maximum size for a multipart upload is 5TB.", bucketName)
        } else {
            log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
                bucketName, objectKey, err)
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }

    return err
}

// DownloadFile gets an object from a bucket and stores it in a local file.

```

```

func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
                objectKey, bucketName)
            err = noKey
        } else {
            log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
                objectKey, err)
        }
        return err
    }
    defer result.Body.Close()
    file, err := os.Create(fileName)
    if err != nil {
        log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
        return err
    }
    defer file.Close()
    body, err := io.ReadAll(result.Body)
    if err != nil {
        log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
            err)
    }
    _, err = file.Write(body)
    return err
}

// DownloadLargeObject uses a download manager to download an object from a
// bucket.
// The download manager gets the data in parts and writes them to a buffer until
// all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
    string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10

```



```

downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
{
    d.PartSize = partMiBs * 1024 * 1024
})
buffer := manager.NewWriteAtBuffer([]byte{})
_, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
    Bucket: aws.String(bucketName),
    Key:    aws.String(objectKey),
})
if err != nil {
    log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
        bucketName, objectKey, err)
}
return buffer.Bytes(), err
}

// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(ctx context.Context, bucketName string,
    objectKey string, folderName string) error {
    objectDest := fmt.Sprintf("%v/%v", folderName, objectKey)
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:    aws.String(bucketName),
        CopySource: aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
        Key:        aws.String(objectDest),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
active tier.\n",
                objectKey, bucketName)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectDest)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectDest)
        }
    }
    return err
}

```

```

}

// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
    destinationBucket string, objectKey string) error {
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(destinationBucket),
        CopySource:   aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
        Key:          aws.String(objectKey),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
                active tier.\n",
                objectKey, sourceBucket)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
                aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }
    return err
}

// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
    ([]types.Object, error) {
    var err error
    var output *s3.ListObjectsV2Output
    input := &s3.ListObjectsV2Input{
        Bucket: aws.String(bucketName),
    }
    var objects []types.Object
    objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
    for objectPaginator.HasMorePages() {

```

```

    output, err = objectPaginator.NextPage(ctx)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        }
        break
    } else {
        objects = append(objects, output.Contents...)
    }
}
return objects, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(ctx context.Context, bucketName string,
    objectKeys []string) error {
    var objectIds []types.ObjectIdentifier
    for _, key := range objectKeys {
        objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
    }
    output, err := basics.S3Client.DeleteObjects(ctx, &s3.DeleteObjectsInput{
        Bucket: aws.String(bucketName),
        Delete: &types.Delete{Objects: objectIds, Quiet: aws.Bool(true)},
    })
    if err != nil || len(output.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucketName)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
        }
        if len(output.Errors) > 0 {
            for _, outErr := range output.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *output.Errors[0].Message)
        }
    } else {
        for _, delObjs := range output.Deleted {

```

```

    err = s3.NewObjectNotExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key: delObjs.Key},
        time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to be deleted.\n",
            *delObjs.Key)
    } else {
        log.Printf("Deleted %s.\n", *delObjs.Key)
    }
}
}
return err
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is
// returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
error {
    _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
        Bucket: aws.String(bucketName)})
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        } else {
            log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
        }
    } else {
        err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
                bucketName)
        } else {
            log.Printf("Deleted %s.\n", bucketName)
        }
    }
}
return err
}

```

Run an interactive scenario that shows you how to work with S3 buckets and objects.

```
import (
    "context"
    "fmt"
    "log"
    "os"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunGetStartedScenario is an interactive example that shows you how to use
// Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
// objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Download an object to a local file.
// 4. Copy an object to a different folder in the bucket.
// 5. List objects in the bucket.
// 6. Delete all objects in the bucket.
// 7. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../..demotools folder of this repo.
func RunGetStartedScenario(ctx context.Context, sdkConfig aws.Config, questioner
    demotools.IQuestioner) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
        }
    }()
}
```

```

_, isMock := questioner.(*demotools.MockQuestioner)
if isMock || questioner.AskBool("Do you want to see the full error message (y/n)?", "y") {
    log.Println(r)
}
}
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 getting started demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}

count := 10
log.Printf("Let's list up to %v buckets for your account:", count)
buckets, err := bucketBasics.ListBuckets(ctx)
if err != nil {
    panic(err)
}
if len(buckets) == 0 {
    log.Println("You don't have any buckets!")
} else {
    if count > len(buckets) {
        count = len(buckets)
    }
    for _, bucket := range buckets[:count] {
        log.Printf("\t%v\n", *bucket.Name)
    }
}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your bucket:",
    demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {

```

```
    log.Println("Bucket created.")
}
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
    demotools.NotEmpty{})
const smallKey = "doc-example-key"
err = bucketBasics.UploadFile(ctx, bucketName, smallKey, smallFile)
if err != nil {
    panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
    demotools.NotEmpty{})
err = bucketBasics.DownloadFile(ctx, bucketName, smallKey, downloadFileName)
if err != nil {
    panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(ctx, bucketName, smallKey, folderName)
if err != nil {
    panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(ctx, bucketName)
if err != nil {
    panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
for _, object := range objects {
```

```
objKeys = append(objKeys, *object.Key)
log.Printf("\t%v\n", *object.Key)
}
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
    "contents? (y/n)", "y") {
    log.Println("Deleting objects.")
    err = bucketBasics.DeleteObjects(ctx, bucketName, objKeys)
    if err != nil {
        panic(err)
    }
    log.Println("Deleting bucket.")
    err = bucketBasics.DeleteBucket(ctx, bucketName)
    if err != nil {
        panic(err)
    }
    log.Printf("Deleting downloaded file %v.\n", downloadFileName)
    err = os.Remove(downloadFileName)
    if err != nil {
        panic(err)
    }
} else {
    log.Println("Okay. Don't forget to delete objects from your bucket to avoid
charges.")
}
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

- For API details, see the following topics in *Amazon SDK for Go API Reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)

- [PutObject](#)

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

A scenario example.

```
import java.io.IOException;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Copy the object to another Amazon S3 bucket using multi copy.
```

- * 8. Deletes the object from the Amazon S3 bucket.
- * 9. Deletes the Amazon S3 bucket.
- */

```
public class S3Scenario {

    public static Scanner scanner = new Scanner(System.in);
    static S3Actions s3Actions = new S3Actions();
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    private static final Logger logger =
LoggerFactory.getLogger(S3Scenario.class);
    public static void main(String[] args) throws IOException {
        final String usage = ""
            Usage:
                <bucketName> <key> <objectPath> <savePath> <toBucket>

        Where:
            bucketName - The name of the S3 bucket.
            key - The unique identifier for the object stored in the S3
bucket.
            objectPath - The full file path of the object within the S3
bucket (e.g., "documents/reports/annual_report.pdf").
            savePath - The local file path where the object will be
downloaded and saved (e.g., "C:/Users/username/Downloads/annual_report.pdf").
            toBucket - The name of the S3 bucket to which the object will be
copied.

        """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];

        logger.info(DASHES);
        logger.info("Welcome to the Amazon Simple Storage Service (S3) example
scenario.");
        logger.info(""
```

Amazon S3 is a highly scalable and durable object storage service provided by Amazon Web Services (AWS). It is designed to store and retrieve any amount of data, from anywhere on the web, at any time.

The `S3AsyncClient` interface in the AWS SDK for Java 2.x provides a set of methods to programmatically interact with the Amazon S3 (Simple Storage Service) service. This allows developers to automate the management and manipulation of S3 buckets and objects as part of their application deployment pipelines. With S3, teams can focus on building and deploying their applications without having to worry about the underlying storage infrastructure required to host and manage large amounts of data.

This scenario walks you through how to perform key operations for this service.

Let's get started...

```
""");
waitForInputToContinue(scanner);
logger.info(DASHES);

try {
    // Run the methods that belong to this scenario.
    runScenario(bucketName, key, objectPath, savePath, toBucket);

} catch (Throwable rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof S3Exception kmsEx) {
        logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
    } else {
        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
}

private static void runScenario(String bucketName, String key, String
objectPath, String savePath, String toBucket) throws Throwable {
    logger.info(DASHES);
    logger.info("1. Create an Amazon S3 bucket.");
    try {
```

```
        CompletableFuture<Void> future =
s3Actions.createBucketAsync(bucketName);
        future.join();
        waitForInputToContinue(scanner);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }

    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("2. Upload a local file to the Amazon S3 bucket.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<PutObjectResponse> future =
s3Actions.uploadLocalFileAsync(bucketName, key, objectPath);
        future.join();
        logger.info("File uploaded successfully to {}/{}", bucketName, key);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }

    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("3. Download the object to another local file.");
    waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<Void> future =
s3Actions.getObjectBytesAsync(bucketName, key, savePath);
            future.join();
            logger.info("Successfully obtained bytes from S3 object and wrote to
file {}", savePath);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Perform a multipart upload.");
        waitForInputToContinue(scanner);
        String multipartKey = "multiPartKey";
        try {
            // Call the multipartUpload method
            CompletableFuture<Void> future =
s3Actions.multipartUpload(bucketName, multipartKey);
            future.join();
            logger.info("Multipart upload completed successfully for bucket '{} '
and key '{}'", bucketName, multipartKey);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
logger.info(DASHES);
logger.info("5. List all objects located in the Amazon S3 bucket.");
waitForInputToContinue(scanner);
try {
    CompletableFuture<Void> future =
s3Actions.listAllObjectsAsync(bucketName);
    future.join();
    logger.info("Object listing completed successfully.");

} catch (RuntimeException rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof S3Exception s3Ex) {
        logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
    } else {
        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
    throw cause;
}
waitForInputToContinue(scanner);
logger.info(DASHES);

logger.info(DASHES);
logger.info("6. Copy the object to another Amazon S3 bucket.");
waitForInputToContinue(scanner);
try {
    CompletableFuture<String> future =
s3Actions.copyBucketObjectAsync(bucketName, key, toBucket);
    String result = future.join();
    logger.info("Copy operation result: {}", result);

} catch (RuntimeException rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof S3Exception s3Ex) {
        logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
    } else {
        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
    throw cause;
}
waitForInputToContinue(scanner);
logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("7. Copy the object to another Amazon S3 bucket using multi
copy.");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
s3Actions.performMultiCopy(toBucket, bucketName, key);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete objects from the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, key);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        throw cause;
    }
    try {
```

```

        CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, "multiPartKey");
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("9. Delete the Amazon S3 bucket.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
s3Actions.deleteBucketAsync(bucketName);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("You successfully completed the Amazon S3 scenario.");
    logger.info(DASHES);
}

private static void waitForInputToContinue(Scanner scanner) {

```



```

        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}

```

A wrapper class that contains the operations.

```

public class S3Actions {

    private static final Logger logger =
        LoggerFactory.getLogger(S3Actions.class);
    private static S3AsyncClient s3AsyncClient;

    public static S3AsyncClient getAsyncClient() {
        if (s3AsyncClient == null) {
            /*
             * The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
             * version 2,
             * and it is designed to provide a high-performance, asynchronous HTTP
             * client for interacting with AWS services.
             * It uses the Netty framework to handle the underlying network
             * communication and the Java NIO API to
             * provide a non-blocking, event-driven approach to HTTP requests and
             * responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50) // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60)) // Set the connection
            timeout.

```

```

        .readTimeout(Duration.ofSeconds(60)) // Set the read timeout.
        .writeTimeout(Duration.ofSeconds(60)) // Set the write timeout.
        .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2)) // Set the overall API
call timeout.
        .apiCallAttemptTimeout(Duration.ofSeconds(90)) // Set the
individual call attempt timeout.
        .retryStrategy(RetryMode.STANDARD)
        .build();

        s3AsyncClient = S3AsyncClient.builder()
        .region(Region.US_EAST_1)
        .httpClient(httpClient)
        .overrideConfiguration(overrideConfig)
        .build();
    }
    return s3AsyncClient;
}

/**
 * Creates an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes when the bucket is
created and ready
 * @throws RuntimeException if there is a failure while creating the bucket
 */
public CompletableFuture<Void> createBucketAsync(String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
    return response.thenCompose(resp -> {
        S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();
    });
}

```

```

        CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
            s3Waiter.waitUntilBucketExists(bucketRequestWait);
        return waiterResponseFuture.thenAccept(waiterResponse -> {
            waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                logger.info(bucketName + " is ready");
            });
        });
    }).whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to create bucket", ex);
        }
    });
}

/**
 * Uploads a local file to an AWS S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (object name) to use for the uploaded file
 * @param objectPath  the local file path of the file to be uploaded
 * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
 */
public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
    return response.whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to upload file", ex);
        }
    });
}
}

```

```

/**
 * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
and writes them to a local file.
 *
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName     the key (or name) of the S3 object to retrieve
 * @param path        the local file path where the object's bytes will be
written
 * @return a {@link CompletableFuture} that completes when the object bytes
have been written to the local file
 */
public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

    CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
    return response.thenAccept(objectBytes -> {
        try {
            byte[] data = objectBytes.asByteArray();
            Path filePath = Paths.get(path);
            Files.write(filePath, data);
            logger.info("Successfully obtained bytes from an S3 object");
        } catch (IOException ex) {
            throw new RuntimeException("Failed to write data to file", ex);
        }
    }).whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to get object bytes from S3",
ex);
        }
    });
}

/**
 * Asynchronously lists all objects in the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket to list objects for

```

```

    * @return a {@link CompletableFuture} that completes when all objects have
    been listed
    */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }

    /**
    * Asynchronously copies an object from one S3 bucket to another.
    *
    * @param fromBucket the name of the source S3 bucket
    * @param objectKey the key (name) of the object to be copied
    * @param toBucket the name of the destination S3 bucket
    * @return a {@link CompletableFuture} that completes with the copy result as
    a {@link String}
    * @throws RuntimeException if the URL could not be encoded or an S3
    exception occurred during the copy
    */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();
    }

```

```

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during
copy", ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }

    /**
     * Performs a multipart upload to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key         the key (name) of the file to be uploaded
     * @return a {@link CompletableFuture} that completes when the multipart
upload is successful
     */
    public CompletableFuture<Void> multipartUpload(String bucketName, String key)
    {
        int mB = 1024 * 1024;

        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        return
getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenCompose(createResponse -> {
                String uploadId = createResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                // Upload part 1.
                UploadPartRequest uploadPartRequest1 =
UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)

```

```
        .uploadId(uploadId)
        .partNumber(1)
        .contentType((long) (5 * MB)) // Specify the content length
        .build();

        CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
MB)))

        .thenApply(uploadPartResponse -> CompletedPart.builder()
        .partNumber(1)
        .eTag(uploadPartResponse.eTag())
        .build());

        // Upload part 2.
        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .partNumber(2)
        .contentType((long) (3 * MB))
        .build();

        CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
MB)))

        .thenApply(uploadPartResponse -> CompletedPart.builder()
        .partNumber(2)
        .eTag(uploadPartResponse.eTag())
        .build());

        // Combine the results of both parts.
        return CompletableFuture.allOf(part1Future, part2Future)
        .thenCompose(v -> {
            CompletedPart part1 = part1Future.join();
            CompletedPart part2 = part2Future.join();

            CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                .parts(part1, part2)
                .build();
```

```

        CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

        // Complete the multipart upload
        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
    });
    })
    .thenAccept(response -> System.out.println("Multipart upload
completed successfully"))
    .exceptionally(ex -> {
        System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
        throw new RuntimeException(ex);
    });
}

/**
 * Deletes an object from an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the key (file name) of the object to be deleted
 * @return a {@link CompletableFuture} that completes when the object has
been deleted
 */
public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(key + " was deleted");
        } else {

```



```

        throw new RuntimeException("An S3 exception occurred during
delete", ex);
    }
    });

    return response.thenApply(r -> null);
}

/**
 * Deletes an S3 bucket asynchronously.
 *
 * @param bucket the name of the bucket to be deleted
 * @return a {@link CompletableFuture} that completes when the bucket
deletion is successful, or throws a {@link RuntimeException}
 * if an error occurs during the deletion process
 */
public CompletableFuture<Void> deleteBucketAsync(String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(bucket + " was deleted.");
        } else {
            throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
        }
    });
    return response.thenApply(r -> null);
}

public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(toBucket)
        .key(key)
        .build();

    getAsyncClient().createMultipartUpload(createMultipartUploadRequest)

```

```

        .thenApply(createMultipartUploadResponse -> {
            String uploadId = createMultipartUploadResponse.uploadId();
            System.out.println("Upload ID: " + uploadId);

            UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                .sourceBucket(bucketName)
                .destinationBucket(toBucket)
                .sourceKey(key)
                .destinationKey(key)
                .uploadId(uploadId) // Use the valid uploadId.
                .partNumber(1) // Ensure the part number is correct.
                .copySourceRange("bytes=0-1023") // Adjust range as needed
                .build();

            return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
        })
        .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
        .whenComplete((uploadPartCopyResponse, exception) -> {
            if (exception != null) {
                // Handle any exceptions.
                logger.error("Error during upload part copy: " +
exception.getMessage());
            } else {
                // Successfully completed the upload part copy.
                System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
            }
        });
    return null;
}

private static ByteBuffer getRandomByteBuffer(int size) {
    ByteBuffer buffer = ByteBuffer.allocate(size);
    for (int i = 0; i < size; i++) {
        buffer.put((byte) (Math.random() * 256));
    }
    buffer.flip();
    return buffer;
}
}

```

- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

First, import all the necessary modules.

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "node:url";
import { readdirSync, readFileSync, writeFileSync } from "node:fs";

// Local helper utils.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
  GetObjectCommand,
  DeleteObjectsCommand,
  DeleteBucketCommand,
} from "@aws-sdk/client-s3";
```

The preceding imports reference some helper utilities. These utilities are local to the GitHub repository linked at the start of this section. For your reference, see the following implementations of those utilities.

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox, password } from "@inquirer/prompts";

export class Prompter {
  /**
   * @param {{ message: string, choices: { name: string, value: string }[] }}
   options
   */
  select(options) {
    return select(options);
  }

  /**
   * @param {{ message: string }} options
   */
  input(options) {
    return input(options);
  }

  /**
   * @param {{ message: string }} options
   */
  password(options) {
    return password({ ...options, mask: true });
  }

  /**
   * @param {string} prompt
   */
  checkContinue = async (prompt = "") => {
    const prefix = prompt && `${prompt} `;
    const ok = await this.confirm({
      message: `${prefix}Continue?`,
    });
    if (!ok) throw new Error("Exiting...");
  };
}
```

```

};

/**
 * @param {{ message: string }} options
 */
confirm(options) {
  return confirm(options);
}

/**
 * @param {{ message: string, choices: { name: string, value: string }[] }}
options
 */
checkbox(options) {
  return checkbox(options);
}
}

export const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};

```

Objects are stored in 'buckets'. Let's define a function for creating a new bucket.

```

export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};

```

Buckets contain 'objects'. This function uploads the contents of a directory to your bucket as objects.

```

export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);

```

```
const files = keys.map((key) => {
  const filePath = `${folderPath}/${key}`;
  const fileContent = readFileSync(filePath);
  return {
    Key: key,
    Body: fileContent,
  };
});

for (const file of files) {
  await s3Client.send(
    new PutObjectCommand({
      Bucket: bucketName,
      Body: file.Body,
      Key: file.Key,
    }),
  );
  console.log(`${file.Key} uploaded successfully.`);
}
};
```

After uploading objects, check to confirm that they were uploaded correctly. You can use `ListObjects` for that. You'll be using the 'Key' property, but there are other useful properties in the response also.

```
export const listFilesInBucket = async ({ bucketName }) => {
  const command = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(command);
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
  console.log("\nHere's a list of files in the bucket:");
  console.log(`${contentsList}\n`);
};
```

Sometimes you might want to copy an object from one bucket to another. Use the `CopyObject` command for that.

```
export const copyFileFromBucket = async ({ destinationBucket }) => {
  const proceed = await prompter.confirm({
    message: "Would you like to copy an object from another bucket?",
  });
};
```

```
if (!proceed) {
  return;
}
const copy = async () => {
  try {
    const sourceBucket = await prompter.input({
      message: "Enter source bucket name:",
    });
    const sourceKey = await prompter.input({
      message: "Enter source key:",
    });
    const destinationKey = await prompter.input({
      message: "Enter destination key:",
    });

    const command = new CopyObjectCommand({
      Bucket: destinationBucket,
      CopySource: `${sourceBucket}/${sourceKey}`,
      Key: destinationKey,
    });
    await s3Client.send(command);
    await copyFileFromBucket({ destinationBucket });
  } catch (err) {
    console.error("Copy error.");
    console.error(err);
    const retryAnswer = await prompter.confirm({ message: "Try again?" });
    if (retryAnswer) {
      await copy();
    }
  }
};
await copy();
};
```

There's no SDK method for getting multiple objects from a bucket. Instead, you'll create a list of objects to download and iterate over them.

```
export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
```

```
const path = await prompter.input({
  message: "Enter destination path for files:",
});

for (const content of Contents) {
  const obj = await s3Client.send(
    new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
  );
  writeFileSync(
    `${path}/${content.Key}`,
    await obj.Body.transformToByteArray(),
  );
}
console.log("Files downloaded successfully.\n");
};
```

It's time to clean up your resources. A bucket must be empty before it can be deleted. These two functions empty and delete the bucket.

```
export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log(`${bucketName} deleted successfully.\n`);
};
```

The 'main' function pulls everything together. If you run this file directly the main function will be called.


```
const main = async () => {
  const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
    import.meta.url,
  )}../../../../../resources/sample_files/.sample_media`;

  try {
    console.log(wrapText("Welcome to the Amazon S3 getting started example."));
    console.log("Let's create a bucket.");
    const bucketName = await createBucket();
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("File upload."));
    console.log(
      "I have some default files ready to go. You can edit the source code to provide your own.",
    );
    await uploadFilesToBucket({
      bucketName,
      folderPath: OBJECT_DIRECTORY,
    });

    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Copy files."));
    await copyFileFromBucket({ destinationBucket: bucketName });
    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Download files."));
    await downloadFilesFromBucket({ bucketName });

    console.log(wrapText("Clean up."));
    await emptyBucket({ bucketName });
    await deleteBucket({ bucketName });
  } catch (err) {
    console.error(err);
  }
};
```

- For API details, see the following topics in *Amazon SDK for JavaScript API Reference*.
 - [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
suspend fun main(args: Array<String>) {  
    val usage = ""  
    Usage:  
        <bucketName> <key> <objectPath> <savePath> <toBucket>  
  
    Where:  
        bucketName - The Amazon S3 bucket to create.  
        key - The key to use.  
        objectPath - The path where the file is located (for example, C:/AWS/  
book2.pdf).  
        savePath - The path where the file is saved after it's downloaded (for  
example, C:/AWS/book2.pdf).  
        toBucket - An Amazon S3 bucket to where an object is copied to (for  
example, C:/AWS/book2.pdf).  
        ""  
  
    if (args.size != 4) {  
        println(usage)  
        exitProcess(1)  
    }  
  
    val bucketName = args[0]
```

```
val key = args[1]
val objectPath = args[2]
val savePath = args[3]
val toBucket = args[4]

// Create an Amazon S3 bucket.
createBucket(bucketName)

// Update a local file to the Amazon S3 bucket.
putObject(bucketName, key, objectPath)

// Download the object to another local file.
getObjectFromMrap(bucketName, key, savePath)

// List all objects located in the Amazon S3 bucket.
listBucketObs(bucketName)

// Copy the object to another Amazon S3 bucket
copyBucketOb(bucketName, key, toBucket)

// Delete the object from the Amazon S3 bucket.
deleteBucketObs(bucketName, key)

// Delete the Amazon S3 bucket.
deleteBucket(bucketName)
println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
```

```
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }
}
```

```

S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->

    val response = s3.listObjects(request)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedOperationException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =

```

```
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- For API details, see the following topics in *Amazon SDK for Kotlin API reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

PHP

SDK for PHP

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
echo("\n");
echo("-----\n");
print("Welcome to the Amazon S3 getting started demo using PHP!\n");
echo("-----\n");

$region = 'us-west-2';

$this->s3client = new S3Client([
    'region' => $region,
]);
/* Inline declaration example
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
*/

$this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

try {
    $this->s3client->createBucket([
        'Bucket' => $this->bucketName,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $this->bucketName \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with bucket creation before continuing.");
}

$fileName = __DIR__ . "/local-file-" . uniqid();
try {
    $this->s3client->putObject([
        'Bucket' => $this->bucketName,
```

```
        'Key' => $fileName,
        'SourceFile' => __DIR__ . '/testfile.txt'
    ]);
    echo "Uploaded $fileName to $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with file upload before continuing.");
}

try {
    $file = $this->s3client->getObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}

try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
    ]);
    echo "Copied $fileName to $folder/$fileName-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with object copying before continuing.");
}

try {
    $contents = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
```



```
        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
    echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
    exit("Please fix error with listing objects before continuing.");
}

try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $this->s3client->deleteObjects([
        'Bucket' => $this->bucketName,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    if (isset($check['Contents']) && count($check['Contents']) > 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with object deletion before continuing.");
}

try {
    $this->s3client->deleteBucket([
        'Bucket' => $this->bucketName,
    ]);
    echo "Deleted bucket $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
    exit("Please fix error with bucket deletion before continuing.");
}
```

```
echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- For API details, see the following topics in *Amazon SDK for PHP API Reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError

def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)
```

```

bucket_name = f"amzn-s3-demo-bucket-{uuid.uuid4()}"
bucket = s3_resource.Bucket(bucket_name)
try:
    bucket.create(
        CreateBucketConfiguration={
            "LocationConstraint": s3_resource.meta.client.meta.region_name
        }
    )
    print(f"Created demo bucket named {bucket.name}.")
except ClientError as err:
    print(f"Tried and failed to create demo bucket {bucket_name}.")
    print(f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}")
    print(f"\nCan't continue the demo without a bucket!")
    return

file_name = None
while file_name is None:
    file_name = input("\nEnter a file you want to upload to your bucket: ")
    if not os.path.exists(file_name):
        print(f"Couldn't find file {file_name}. Are you sure it exists?")
        file_name = None

obj = bucket.Object(os.path.basename(file_name))
try:
    obj.upload_file(file_name)
    print(
        f"Uploaded file {file_name} into bucket {bucket.name} with key
{obj.key}."
    )
except S3UploadFailedError as err:
    print(f"Couldn't upload file {file_name} to {bucket.name}.")
    print(f"\t{err}")

answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
if answer.lower() == "y":
    data = io.BytesIO()
    try:
        obj.download_fileobj(data)
        data.seek(0)
        print(f"Got your object. Here are the first 20 bytes:\n")
        print(f"\t{data.read(20)}")
    except ClientError as err:
        print(f"Couldn't download {obj.key}.")

```

```

        print(
            f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
        )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")

    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)?
"
    )
    if answer.lower() == "y":
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

```

```
print("Thanks for watching!")
print("-" * 88)

if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- For API details, see the following topics in *Amazon SDK for Python (Boto3) API Reference*.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end
end
```

```

# Creates a bucket with a random name in the currently configured account and
# AWS Region.
#
# @return [Aws::S3::Bucket] The newly created bucket.
def create_bucket
  bucket = @s3_resource.create_bucket(
    bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
    create_bucket_configuration: {
      location_constraint: 'us-east-1' # NOTE: only certain regions permitted
    }
  )
  puts("Created demo bucket named #{bucket.name}.")
rescue Aws::Errors::ServiceError => e
  puts('Tried and failed to create demo bucket.')
  puts("\t#{e.code}: #{e.message}")
  puts("\nCan't continue the demo without a bucket!")
  raise
else
  bucket
end

# Requests a file name from the user.
#
# @return The name of the file.
def create_file
  File.open('demo.txt', w) { |f| f.write('This is a demo file.') }
end

# Uploads a file to an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
# @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded
file.
def upload_file(bucket)
  File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
  s3_object = bucket.object(File.basename('demo.txt'))
  s3_object.upload_file('demo.txt')
  puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't upload file demo.txt to #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
end

```

```

        raise
    else
        s3_object
    end

    # Downloads an Amazon S3 object to a file.
    #
    # @param s3_object [Aws::S3::Object] The object to download.
    def download_file(s3_object)
        puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
        answer = gets.chomp.downcase
        if answer == 'y'
            puts('Enter a name for the downloaded file: ')
            file_name = gets.chomp
            s3_object.download_file(file_name)
            puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
        end
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't download #{s3_object.key}.")
        puts("\t#{e.code}: #{e.message}")
        raise
    end

    # Copies an Amazon S3 object to a subfolder within the same bucket.
    #
    # @param source_object [Aws::S3::Object] The source object to copy.
    # @return [Aws::S3::Object, nil] The destination object.
    def copy_object(source_object)
        dest_object = nil
        puts("\nDo you want to copy #{source_object.key} to a subfolder in your
        bucket (y/n)? ")
        answer = gets.chomp.downcase
        if answer == 'y'
            dest_object = source_object.bucket.object("demo-folder/
            #{source_object.key}")
            dest_object.copy_from(source_object)
            puts("Copied #{source_object.key} to #{dest_object.key}.")
        end
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't copy #{source_object.key}.")
        puts("\t#{e.code}: #{e.message}")
        raise
    else
        dest_object
    end

```

```
end

# Lists the objects in an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to query.
def list_objects(bucket)
  puts("\nYour bucket contains the following objects:")
  bucket.objects.each do |obj|
    puts("\t#{obj.key}")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't list the objects in bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the Amazon S3 getting started demo!')
  puts('-' * 88)

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
```



```
scenario.copy_object(s3_object)
scenario.list_objects(bucket)
scenario.delete_bucket(bucket)

puts('Thanks for watching!')
puts('-' * 88)
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo!')
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME
== __FILE__
```

- For API details, see the following topics in *Amazon SDK for Ruby API Reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Code for the binary crate which runs the scenario.

```
#![allow(clippy::result_large_err)]

//! Purpose
```

```

    //! Shows how to use the AWS SDK for Rust to get started using
    //! Amazon Simple Storage Service (Amazon S3). Create a bucket, move objects
    //! into and out of it,
    //! and delete all resources at the end of the demo.
    //!
    //! This example follows the steps in "Getting started with Amazon S3" in the
    //! Amazon S3
    //! user guide.
    //! - https://docs.aws.amazon.com/AmazonS3/latest/userguide/GetStartedWithS3.html

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{config::Region, Client};
use s3_code_examples::error::S3ExampleError;
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    let shared_config =
aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);
    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name,
key, target_key).await
    {
        eprintln!("{:?}", e);
    };

    Ok(())
}

async fn run_s3_operations(
    region: Region,
    client: Client,
    bucket_name: String,
    file_name: String,
    key: String,

```

```

        target_key: String,
    ) -> Result<(), S3ExampleError> {
        s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;
        let run_example: Result<(), S3ExampleError> = (async {
            s3_code_examples::upload_object(&client, &bucket_name, &file_name,
            &key).await?;
            let _object = s3_code_examples::download_object(&client, &bucket_name,
            &key).await?;
            s3_code_examples::copy_object(&client, &bucket_name, &bucket_name, &key,
            &target_key)
                .await?;
            s3_code_examples::list_objects(&client, &bucket_name).await?;
            s3_code_examples::clear_bucket(&client, &bucket_name).await?;
            Ok(())
        })
        .await;
        if let Err(err) = run_example {
            eprintln!("Failed to complete getting-started example: {err:?}");
        }
        s3_code_examples::delete_bucket(&client, &bucket_name).await?;

        Ok(())
    }
}

```

Common actions used by the scenario.

```

pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
S3ExampleError> {
    let constraint =
aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()

```

```

        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
    task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}

pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}

pub async fn download_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::get_object::GetObjectOutput, S3ExampleError> {
    client
        .get_object()
        .bucket(bucket_name)

```

```

        .key(key)
        .send()
        .await
        .map_err(S3ExampleError::from)
    }

    /// Copy an object from one bucket to another.
    pub async fn copy_object(
        client: &aws_sdk_s3::Client,
        source_bucket: &str,
        destination_bucket: &str,
        source_object: &str,
        destination_object: &str,
    ) -> Result<(), S3ExampleError> {
        let source_key = format!("{source_bucket}/{source_object}");
        let response = client
            .copy_object()
            .copy_source(&source_key)
            .bucket(destination_bucket)
            .key(destination_object)
            .send()
            .await?;

        println!(
            "Copied from {source_key} to {destination_bucket}/{destination_object}
            with etag {}",
            response
                .copy_object_result
                .unwrap_or_else(||
                    aws_sdk_s3::types::CopyObjectResult::builder().build()
                        .e_tag()
                        .unwrap_or("missing")
                )
        );
        Ok(())
    }

    pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
    Result<(), S3ExampleError> {
        let mut response = client
            .list_objects_v2()
            .bucket(bucket.to_owned())
            .max_keys(10) // In this example, go 10 at a time.
            .into_paginator()
            .send();

```

```

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}

/// Given a bucket, remove all objects in the bucket, and then ensure no objects
/// remain in the bucket.
pub async fn clear_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<Vec<String>, S3ExampleError> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    // delete_objects no longer needs to be mutable.
    let objects_to_delete: Vec<String> = objects
        .contents()
        .iter()
        .filter_map(|obj| obj.key())
        .map(String::from)
        .collect();

    if objects_to_delete.is_empty() {
        return Ok(vec![]);
    }

    let return_keys = objects_to_delete.clone();

    delete_objects(client, bucket_name, objects_to_delete).await?;

    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");
}

```

```

        match objects.key_count {
            Some(0) => Ok(return_keys),
            _ => Err(S3ExampleError::new(
                "There were still objects left in the bucket.",
            )),
        }
    }
}

pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}
}

```

- For API details, see the following topics in *Amazon SDK for Rust API reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

SAP ABAP

SDK for SAP ABAP

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create an Amazon Simple Storage Service (Amazon S3) bucket. "
TRY.
    " determine our region from our session
    DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
    DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
    " When in the us-east-1 region, you must not specify a constraint
    " In all other regions, specify the region as the constraint
    IF lv_region = 'us-east-1'.
        CLEAR lo_constraint.
    ELSE.
        lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
    ENDIF.

    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
        io_createbucketconfiguration = lo_constraint ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
CATCH /aws1/cx_s3_bucketalrddyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
CATCH /aws1/cx_s3_bktalrddyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.

"Upload an object to an S3 bucket."
TRY.
    "Get contents of file from application server."
    DATA lv_file_content TYPE xstring.

```



```
OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
READ DATASET iv_key INTO lv_file_content.
CLOSE DATASET iv_key.

lo_s3->putobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_key
    iv_body = lv_file_content ).
MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Get an object from a bucket. "
TRY.
    DATA(lo_result) = lo_s3->getobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key ).
    DATA(lv_object_data) = lo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" Copy an object to a subfolder in a bucket. "
TRY.
    lo_s3->copyobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }|
        iv_copysource = |{ iv_bucket_name }/{ iv_key }| ).
    MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" List objects in the bucket. "
TRY.
    DATA(lo_list) = lo_s3->listobjects(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
```

```

    CATCH /aws1/cx_s3_nosuchbucket.
      MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
    DATA text TYPE string VALUE 'Object List - '.
    DATA lv_object_key TYPE /aws1/s3_objectkey.
    LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
      lv_object_key = lo_object->get_key( ).
      CONCATENATE lv_object_key ', ' INTO text.
    ENDLOOP.
    MESSAGE text TYPE 'I'.

" Delete the objects in a bucket. "
TRY.
  lo_s3->deleteobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_key ).
  lo_s3->deleteobject(
    iv_bucket = iv_bucket_name
    iv_key = |{ iv_copy_to_folder }/{ iv_key }| ).
  MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Delete the bucket. "
TRY.
  lo_s3->deletebucket(
    iv_bucket = iv_bucket_name ).
  MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- For API details, see the following topics in *Amazon SDK for SAP ABAP API reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)

- [ListObjectsV2](#)
- [PutObject](#)

Swift

SDK for Swift

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import AWSS3

import Foundation
import AWSS3
import Smithy
import ClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let configuration: S3Client.S3ClientConfiguration
    let client: S3Client

    enum HandlerError: Error {
        case getObjectBody(String)
        case readGetObjectBody(String)
        case missingContents(String)
    }

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
    drive the AWS calls
    /// used for the example.
    ///
    /// - Returns: A new ``ServiceHandler`` object, ready to be called to
    ///           execute AWS operations.
    public init() async throws {
        do {
            configuration = try await S3Client.S3ClientConfiguration()
```

```

        // configuration.region = "us-east-2" // Uncomment this to set the
region programmatically.
        client = S3Client(config: configuration)
    }
    catch {
        print("ERROR: ", dump(error, name: "Initializing S3 client"))
        throw error
    }
}

/// Create a new user given the specified name.
///
/// - Parameters:
///   - name: Name of the bucket to create.
/// Throws an exception if an error occurs.
public func createBucket(name: String) async throws {
    var input = CreateBucketInput(
        bucket: name
    )

    // For regions other than "us-east-1", you must set the
locationConstraint in the createBucketConfiguration.
    // For more information, see LocationConstraint in the S3 API guide.
    // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
    if let region = configuration.region {
        if region != "us-east-1" {
            input.createBucketConfiguration =
S3ClientTypes.CreateBucketConfiguration(locationConstraint:
S3ClientTypes.BucketLocationConstraint(rawValue: region))
        }
    }

    do {
        _ = try await client.createBucket(input: input)
    }
    catch let error as BucketAlreadyOwnedByYou {
        print("The bucket '\(name)' already exists and is owned by you. You
may wish to ignore this exception.")
        throw error
    }
    catch {
        print("ERROR: ", dump(error, name: "Creating a bucket"))
    }
}

```

```
        throw error
    }
}

/// Delete a bucket.
/// - Parameter name: Name of the bucket to delete.
public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    do {
        _ = try await client.deleteBucket(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a bucket"))
        throw error
    }
}

/// Upload a file from local storage to the bucket.
/// - Parameters:
///   - bucket: Name of the bucket to upload the file to.
///   - key: Name of the file to create.
///   - file: Path name of the file to upload.
public func uploadFile(bucket: String, key: String, file: String) async
throws {
    let fileUrl = URL(fileURLWithPath: file)
    do {
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object."))
        throw error
    }
}
```

```
/// Create a file in the specified bucket with the given name. The new
/// file's contents are uploaded from a `Data` object.
///
/// - Parameters:
///   - bucket: Name of the bucket to create a file in.
///   - key: Name of the file to create.
///   - data: A `Data` object to write into the new file.
public func createFile(bucket: String, key: String, withData data: Data)
async throws {
    let dataStream = ByteStream.data(data)

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object."))
        throw error
    }
}

/// Download the named file to the given directory on the local device.
///
/// - Parameters:
///   - bucket: Name of the bucket that contains the file to be copied.
///   - key: The name of the file to copy from the bucket.
///   - to: The path of the directory on the local device where you want to
///         download the file.
public func downloadFile(bucket: String, key: String, to: String) async
throws {
    let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        let output = try await client.getObject(input: input)
```

```
        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        try data.write(to: fileUrl)
    }
    catch {
        print("ERROR: ", dump(error, name: "Downloading a file. "))
        throw error
    }
}

/// Read the specified file from the given S3 bucket into a Swift
/// `Data` object.
///
/// - Parameters:
///   - bucket: Name of the bucket containing the file to read.
///   - key: Name of the file within the bucket to read.
///
/// - Returns: A `Data` object containing the complete file data.
public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        return data
    }
}
```

```
    }
    catch {
        print("ERROR: ", dump(error, name: "Reading a file. "))
        throw error
    }
}

/// Copy a file from one bucket to another.
///
/// - Parameters:
///   - sourceBucket: Name of the bucket containing the source file.
///   - name: Name of the source file.
///   - destBucket: Name of the bucket to copy the file into.
public func copyFile(from sourceBucket: String, name: String, to destBucket:
String) async throws {
    let srcUrl = ("\"(sourceBucket)/
\"(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

    let input = CopyObjectInput(
        bucket: destBucket,
        copySource: srcUrl,
        key: name
    )
    do {
        _ = try await client.copyObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Copying an object. "))
        throw error
    }
}

/// Deletes the specified file from Amazon S3.
///
/// - Parameters:
///   - bucket: Name of the bucket containing the file to delete.
///   - key: Name of the file to delete.
///
public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )
}
```



```
do {
    _ = try await client.deleteObject(input: input)
}
catch {
    print("ERROR: ", dump(error, name: "Deleting a file. "))
    throw error
}
}

/// Returns an array of strings, each naming one file in the
/// specified bucket.
///
/// - Parameter bucket: Name of the bucket to get a file listing for.
/// - Returns: An array of `String` objects, each giving the name of
///            one file contained in the bucket.
public func listBucketFiles(bucket: String) async throws -> [String] {
    do {
        let input = ListObjectsV2Input(
            bucket: bucket
        )

        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'continuationToken' in
        "ListObjectsV2Output".
        let output = client.listObjectsV2Paginated(input: input)
        var names: [String] = []

        for try await page in output {
            guard let objList = page.contents else {
                print("ERROR: listObjectsV2Paginated returned nil contents.")
                continue
            }

            for obj in objList {
                if let objName = obj.key {
                    names.append(objName)
                }
            }
        }

        return names
    }
}
```

```
        catch {
            print("ERROR: ", dump(error, name: "Listing objects. "))
            throw error
        }
    }
}
```

```
import AWSS3

import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String

    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String

    @Argument(help: "Directory where you want to download the file from the S3
bucket")
    var downloadDir: String

    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: """
        Performs the following Amazon S3 commands:

        * `CreateBucket`
        * `PutObject`
        * `GetObject`
        * `CopyObject`
    """
    )
}
```

```

        * `ListObjects`
        * `DeleteObjects`
        * `DeleteBucket`
        """
    )

    /// Called by ``main()`` to do the actual running of the AWS
    /// example.
    func runAsync() async throws {
        let serviceHandler = try await ServiceHandler()

        // 1. Create the bucket.
        print("Creating the bucket \(bucketName)...")
        try await serviceHandler.createBucket(name: bucketName)

        // 2. Upload a file to the bucket.
        print("Uploading the file \(uploadSource)...")
        try await serviceHandler.uploadFile(bucket: bucketName, key: objName,
file: uploadSource)

        // 3. Download the file.
        print("Downloading the file \(objName) to \(downloadDir)...")
        try await serviceHandler.downloadFile(bucket: bucketName, key: objName,
to: downloadDir)

        // 4. Copy the file to another bucket.
        print("Copying the file to the bucket \(destBucket)...")
        try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)

        // 5. List the contents of the bucket.

        print("Getting a list of the files in the bucket \(bucketName)")
        let fileList = try await serviceHandler.listBucketFiles(bucket:
bucketName)
        let numFiles = fileList.count
        if numFiles != 0 {
            print("\(numFiles) file\((numFiles > 1) ? "s" : "") in bucket
\(bucketName):")
            for name in fileList {
                print("  \(name)")
            }
        } else {
            print("No files found in bucket \(bucketName)")
        }
    }
}

```

```

    }

    // 6. Delete the objects from the bucket.

    print("Deleting the file \(objName) from the bucket \(bucketName)...")
    try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
    print("Deleting the file \(objName) from the bucket \(destBucket)...")
    try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

    // 7. Delete the bucket.
    print("Deleting the bucket \(bucketName)...")
    try await serviceHandler.deleteBucket(name: bucketName)

    print("Done.")
}
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
}

```

- For API details, see the following topics in *Amazon SDK for Swift API reference*.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)

- [ListObjectsV2](#)
- [PutObject](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Actions for Amazon S3 using Amazon SDKs

The following code examples demonstrate how to perform individual Amazon S3 actions with Amazon SDKs. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

These excerpts call the Amazon S3 API and are code excerpts from larger programs that must be run in context. You can see actions in context in [Scenarios for Amazon S3 using Amazon SDKs](#).

The following examples include only the most commonly used actions. For a complete list, see the [Amazon Simple Storage Service API Reference](#).

Examples

- [Use AbortMultipartUpload with an Amazon SDK or CLI](#)
- [Use CompleteMultipartUpload with an Amazon SDK or CLI](#)
- [Use CopyObject with an Amazon SDK or CLI](#)
- [Use CreateBucket with an Amazon SDK or CLI](#)
- [Use CreateMultiRegionAccessPoint with an Amazon SDK](#)
- [Use CreateMultipartUpload with an Amazon SDK or CLI](#)
- [Use CreatePresignedPost with an Amazon SDK](#)
- [Use DeleteBucket with an Amazon SDK or CLI](#)
- [Use DeleteBucketAnalyticsConfiguration with a CLI](#)
- [Use DeleteBucketCors with an Amazon SDK or CLI](#)
- [Use DeleteBucketEncryption with a CLI](#)
- [Use DeleteBucketInventoryConfiguration with a CLI](#)
- [Use DeleteBucketLifecycle with an Amazon SDK or CLI](#)
- [Use DeleteBucketMetricsConfiguration with a CLI](#)

- [Use DeleteBucketPolicy with an Amazon SDK or CLI](#)
- [Use DeleteBucketReplication with a CLI](#)
- [Use DeleteBucketTagging with a CLI](#)
- [Use DeleteBucketWebsite with an Amazon SDK or CLI](#)
- [Use DeleteObject with an Amazon SDK or CLI](#)
- [Use DeleteObjectTagging with a CLI](#)
- [Use DeleteObjects with an Amazon SDK or CLI](#)
- [Use DeletePublicAccessBlock with a CLI](#)
- [Use GetBucketAccelerateConfiguration with a CLI](#)
- [Use GetBucketAcl with an Amazon SDK or CLI](#)
- [Use GetBucketAnalyticsConfiguration with a CLI](#)
- [Use GetBucketCors with an Amazon SDK or CLI](#)
- [Use GetBucketEncryption with an Amazon SDK or CLI](#)
- [Use GetBucketInventoryConfiguration with a CLI](#)
- [Use GetBucketLifecycleConfiguration with an Amazon SDK or CLI](#)
- [Use GetBucketLocation with an Amazon SDK or CLI](#)
- [Use GetBucketLogging with a CLI](#)
- [Use GetBucketMetricsConfiguration with a CLI](#)
- [Use GetBucketNotification with a CLI](#)
- [Use GetBucketPolicy with an Amazon SDK or CLI](#)
- [Use GetBucketPolicyStatus with a CLI](#)
- [Use GetBucketReplication with an Amazon SDK or CLI](#)
- [Use GetBucketRequestPayment with a CLI](#)
- [Use GetBucketTagging with a CLI](#)
- [Use GetBucketVersioning with a CLI](#)
- [Use GetBucketWebsite with an Amazon SDK or CLI](#)
- [Use GetObject with an Amazon SDK or CLI](#)
- [Use GetObjectAcl with an Amazon SDK or CLI](#)
- [Use GetObjectAttributes with an Amazon SDK or CLI](#)
- [Use GetObjectLegalHold with an Amazon SDK or CLI](#)

- [Use GetObjectLockConfiguration with an Amazon SDK or CLI](#)
- [Use GetObjectRetention with an Amazon SDK or CLI](#)
- [Use GetObjectTagging with a CLI](#)
- [Use GetPublicAccessBlock with a CLI](#)
- [Use HeadBucket with an Amazon SDK or CLI](#)
- [Use HeadObject with an Amazon SDK or CLI](#)
- [Use ListBucketAnalyticsConfigurations with a CLI](#)
- [Use ListBucketInventoryConfigurations with a CLI](#)
- [Use ListBuckets with an Amazon SDK or CLI](#)
- [Use ListMultipartUploads with an Amazon SDK or CLI](#)
- [Use ListObjectVersions with an Amazon SDK or CLI](#)
- [Use ListObjects with a CLI](#)
- [Use ListObjectsV2 with an Amazon SDK or CLI](#)
- [Use PutBucketAccelerateConfiguration with an Amazon SDK or CLI](#)
- [Use PutBucketAcl with an Amazon SDK or CLI](#)
- [Use PutBucketCors with an Amazon SDK or CLI](#)
- [Use PutBucketEncryption with an Amazon SDK or CLI](#)
- [Use PutBucketLifecycleConfiguration with an Amazon SDK or CLI](#)
- [Use PutBucketLogging with an Amazon SDK or CLI](#)
- [Use PutBucketNotification with a CLI](#)
- [Use PutBucketNotificationConfiguration with an Amazon SDK or CLI](#)
- [Use PutBucketPolicy with an Amazon SDK or CLI](#)
- [Use PutBucketReplication with an Amazon SDK or CLI](#)
- [Use PutBucketRequestPayment with a CLI](#)
- [Use PutBucketTagging with a CLI](#)
- [Use PutBucketVersioning with an Amazon SDK or CLI](#)
- [Use PutBucketWebsite with an Amazon SDK or CLI](#)
- [Use PutObject with an Amazon SDK or CLI](#)
- [Use PutObjectAcl with an Amazon SDK or CLI](#)
- [Use PutObjectLegalHold with an Amazon SDK or CLI](#)

- [Use PutObjectLockConfiguration with an Amazon SDK or CLI](#)
- [Use PutObjectRetention with an Amazon SDK or CLI](#)
- [Use RestoreObject with an Amazon SDK or CLI](#)
- [Use SelectObjectContent with an Amazon SDK or CLI](#)
- [Use UploadPart with an Amazon SDK or CLI](#)
- [Use UploadPartCopy with an Amazon SDK or CLI](#)

Use AbortMultipartUpload with an Amazon SDK or CLI

The following code examples show how to use AbortMultipartUpload.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Delete incomplete multipart uploads](#)
- [Work with Amazon S3 object integrity](#)

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
```



```

        const Aws::String &uploadID,
        const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
        client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [AbortMultipartUpload](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

To abort the specified multipart upload

The following `abort-multipart-upload` command aborts a multipart upload for the key `multipart/01` in the bucket `amzn-s3-demo-bucket`.

```

aws s3api abort-multipart-upload \
  --bucket amzn-s3-demo-bucket \
  --key multipart/01 \
  --upload-
id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P

```

The upload ID required by this command is output by `create-multipart-upload` and can also be retrieved with `list-multipart-uploads`.

- For API details, see [AbortMultipartUpload](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
```

```
import java.util.Collection;
import java.util.List;
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class AbortMultipartUploadExamples {
    static final S3Client s3Client = S3Client.create();
    static final String classPathFilePath = "/multipartUploadFiles/s3-userguide.pdf";
    static final String filePath = getFullFilePath(classPathFilePath);
    private static final Logger logger =
        LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

        String key = UUID.randomUUID().toString();

        createBucket(bucketName);
        try {
            initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread");
            abortIncompleteMultipartUploadsFromList(bucketName);
            abortMultipartUploadUsingUploadId(bucketName, key);
            abortMultipartUploadsUsingLifecycleConfig(bucketName);
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources(bucketName, key);
        }
    }
}
```

```
/**
 * Aborts all incomplete multipart uploads from the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket
 */
public static void abortIncompleteMultipartUploadsFromList(String bucketName)
{
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(upload.key())
            .expectedBucketOwner(accountId)
            .uploadId(upload.uploadId())
            .build();

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
        }
    }
}

/**
 * Aborts incomplete multipart uploads older than the specified point in
time.
 *
 * @param bucketName the name of the S3 bucket
 * @param pointInTime the cutoff time; uploads initiated before this are
aborted
 */
static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {
```

```

        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
    .bucket(bucketName)
    .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }

    /**
     * Aborts a multipart upload using the upload ID.
     *
     * @param bucketName the name of the S3 bucket
     * @param key         the object key
     */
    static void abortMultipartUploadUsingUploadId(String bucketName, String key)
    {
        String uploadId = startUploadReturningUploadId(bucketName, key);
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b

```

```

        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
    }
}

/**
 * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
 *
 * @param bucketName the name of the S3 bucket
 */
static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
    Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b.
            daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());

    PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
        .bucket(bucketName)
        .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
    } else {
        logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
    }
}

/*****
Multipart upload methods
*****/

```

```

    static void initiateAndInterruptMultiPartUpload(String bucketName, String
key, String threadName) {
        Runnable upload = () -> {
            try {
                doMultipartUpload(bucketName, key);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        };
        Thread uploadThread = new Thread(upload, threadName);
        uploadThread.start();
        try {
            Thread.sleep(Duration.ofSeconds(1).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        uploadThread.interrupt();
    }

    static Instant initiateAndInterruptTwoUploads(String bucketName, String key)
{
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread1");
        try {
            Thread.sleep(Duration.ofSeconds(5).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        Instant secondUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread2");
        return secondUploadInstant;
    }

    static void doMultipartUpload(String bucketName, String key) {
        String uploadId = step1CreateMultipartUpload(bucketName, key);
        List<CompletedPart> completedParts = step2UploadParts(bucketName, key,
uploadId);
        step3CompleteMultipartUpload(bucketName, key, uploadId, completedParts);
    }

    static String step1CreateMultipartUpload(String bucketName, String key) {
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
    }

```

```
        return createMultipartUploadResponse.uploadId();
    }

    static List<CompletedPart> step2UploadParts(String bucketName, String key,
String uploadId) {
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip();
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);
                logger.info("Part {} upload", partNumber);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException | S3Exception e) {
            logger.error(e.getMessage());
            return null;
        }
        return completedParts;
    }
```



```

    }

    static void step3CompleteMultipartUpload(String bucketName, String key,
String uploadId, List<CompletedPart> completedParts) {
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

        .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }

    static String startUploadReturningUploadId(String bucketName, String key) {
        String uploadId = step1CreateMultipartUpload(bucketName, key);
        doMultipartUploadWithUploadId(bucketName, key, uploadId);
        return uploadId;
    }

    static void doMultipartUploadWithUploadId(String bucketName, String key,
String uploadId) {
        new Thread(() -> {
            try {
                List<CompletedPart> completedParts = step2UploadParts(bucketName,
key, uploadId);
                step3CompleteMultipartUpload(bucketName, key, uploadId,
completedParts);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        }, "upload thread").start();
        try {
            Thread.sleep(Duration.ofSeconds(2L).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
            System.exit(1);
        }
    }

    }

    /*****
    Resource handling methods
    *****/

    static void createBucket(String bucketName) {
        logger.info("Creating bucket: [{}]", bucketName);
    }

```

```

        s3Client.createBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
        }
        logger.info("Bucket created.");
    }

    static void deleteResources(String bucketName, String key) {
        logger.info("Deleting resources ...");
        s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
        s3Client.deleteBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
        }
        logger.info("Resources deleted.");
    }

    private static String getAccountId() {
        try (StsClient stsClient = StsClient.create()) {
            return stsClient.getCallerIdentity().account();
        }
    }

    static String getFullFilePath(String filePath) {
        URL uploadDirectoryURL =
PerformMultiPartUpload.class.getResource(filePath);
        String fullFilePath;
        try {
            fullFilePath =
Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
        return fullFilePath;
    }
}

```

- For API details, see [AbortMultipartUpload](#) in *Amazon SDK for Java 2.x API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command aborts multipart uploads created earlier than 5 days ago.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Example 2: This command aborts multipart uploads created earlier than January 2nd, 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

Example 3: This command aborts multipart uploads created earlier than January 2nd, 2014, 10:45:37.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- For API details, see [AbortMultipartUpload](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command aborts multipart uploads created earlier than 5 days ago.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Example 2: This command aborts multipart uploads created earlier than January 2nd, 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

Example 3: This command aborts multipart uploads created earlier than January 2nd, 2014, 10:45:37.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- For API details, see [AbortMultipartUpload](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use CompleteMultipartUpload with an Amazon SDK or CLI

The following code examples show how to use CompleteMultipartUpload.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Perform a multipart copy](#)
- [Use checksums](#)
- [Work with Amazon S3 object integrity](#)

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,
```

```

const Aws::String &key,

const Aws::String &uploadID,

const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
        client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

```

- For API details, see [CompleteMultipartUpload](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command completes a multipart upload for the key `multipart/01` in the bucket `amzn-s3-demo-bucket`:

```

aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket amzn-s3-demo-bucket --key 'multipart/01' --upload-
id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6YjlX.cph0gtNBtJ8P

```

The upload ID required by this command is output by `create-multipart-upload` and can also be retrieved with `list-multipart-uploads`.

The multipart upload option in the above command takes a JSON structure that describes the parts of the multipart upload that should be reassembled into the complete file. In this example, the `file://` prefix is used to load the JSON structure from a file in the local folder named `mpustruct`.

`mpustruct`:

```
{
  "Parts": [
    {
      "ETag": "e868e0f4719e394144ef36531ee6824c",
      "PartNumber": 1
    },
    {
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",
      "PartNumber": 2
    },
    {
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",
      "PartNumber": 3
    }
  ]
}
```

The ETag value for each part is output each time you upload a part using the `upload-part` command and can also be retrieved by calling `list-parts` or calculated by taking the MD5 checksum of each part.

Output:

```
{
  "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
  "Bucket": "amzn-s3-demo-bucket",
  "Location": "https://amzn-s3-demo-bucket.s3.amazonaws.com/multipart%2F01",
  "Key": "multipart/01"
}
```

- For API details, see [CompleteMultipartUpload](#) in *Amazon CLI Command Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;
```

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();
```

```

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;

    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
            .build(),
    );
}

```

- For API details, see [CompleteMultipartUpload](#) in *Amazon SDK for Rust API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use CopyObject with an Amazon SDK or CLI

The following code examples show how to use CopyObject.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Learn the basics](#)
- [Get started with encryption](#)
- [Getting started with Amazon S3](#)
- [Make conditional requests](#)

.NET

Amazon SDK for .NET (v4)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Copies an object in an Amazon S3 bucket to a folder within the
/// same bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// object to copy is located.</param>
/// <param name="objectName">The object to be copied.</param>
/// <param name="folderName">The folder to which the object will
/// be copied.</param>
/// <returns>A boolean value that indicates the success or failure of
/// the copy operation.</returns>
public async Task<bool> CopyObjectInBucketAsync(
    string bucketName,
    string objectName,
    string folderName)
{
    try
```

```
{
    var request = new CopyObjectRequest
    {
        SourceBucket = bucketName,
        SourceKey = objectName,
        DestinationBucket = bucketName,
        DestinationKey = $"{folderName}\\{objectName}",
    };
    var response = await _amazonS3.CopyObjectAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error copying object: '{ex.Message}'");
    return false;
}
}
```

- For API details, see [CopyObject](#) in *Amazon SDK for .NET API Reference*.

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Copy an object using a conditional request.

```
/// <summary>
/// Copies an object from one Amazon S3 bucket to another with a conditional
request.
/// </summary>
/// <param name="sourceKey">The key of the source object to copy.</param>
/// <param name="destKey">The key of the destination object.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="destBucket">The destination bucket of the object.</param>
/// <param name="conditionType">The type of condition to apply, e.g.
'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.</param>
```

```

    /// <param name="conditionDateValue">The value to use for the condition for
    dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
    etags.</param>
    /// <returns>True if the conditional copy is successful, False otherwise.</
returns>
    public async Task<bool> CopyObjectConditional(string sourceKey, string
destKey, string sourceBucket, string destBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
    {
        try
        {
            var copyObjectRequest = new CopyObjectRequest
            {
                DestinationBucket = destBucket,
                DestinationKey = destKey,
                SourceBucket = sourceBucket,
                SourceKey = sourceKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    copyObjectRequest.ETagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
            }

            await _amazonS3.CopyObjectAsync(copyObjectRequest);

```

```

        _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
        return true;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional copy failed: Precondition failed");
        }
        else if (e.ErrorCode == "304")
        {
            _logger.LogError("Conditional copy failed: Object not modified");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

```

- For API details, see [CopyObject](#) in *Amazon SDK for .NET API Reference*.

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {

```

```

    printf "%s\n" "$*" 1>&2
}

#####
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file from and to.
#     $2 - The key of the source file to copy.
#     $3 - The key of the destination file.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_item_in_bucket() {
    local bucket_name=$1
    local source_key=$2
    local destination_key=$3
    local response

    response=$(aws s3api copy-object \
        --bucket "$bucket_name" \
        --copy-source "$bucket_name/$source_key" \
        --key "$destination_key")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api copy-object operation failed.\n$response"
        return 1
    fi
}

```

- For API details, see [CopyObject](#) in *Amazon CLI Command Reference*.

C++

SDK for C++

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
    &fromBucket, const Aws::String &toBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
        .WithKey(objectKey)
        .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;

        } else {
            std::cout << "Successfully copied " << objectKey << " from " <<
fromBucket <<
                " to " << toBucket << "." << std::endl;
        }

        return outcome.IsSuccess();
    }
}
```

- For API details, see [CopyObject](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command copies an object from bucket-1 to bucket-2:

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --  
bucket bucket-2
```

Output:

```
{  
  "CopyObjectResult": {  
    "LastModified": "2015-11-10T01:07:25.000Z",  
    "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""  
  },  
  "VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifM1kA"  
}
```

- For API details, see [CopyObject](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"
```

```

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
    destinationBucket string, objectKey string) error {
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(destinationBucket),
        CopySource:  aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
        Key:         aws.String(objectKey),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
            active tier.\n",
                objectKey, sourceBucket)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
            aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }
}

```



```
    return err
}
```

- For API details, see [CopyObject](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Copy an object using an [S3Client](#).

```
/**
 * Asynchronously copies an object from one S3 bucket to another.
 *
 * @param fromBucket the name of the source S3 bucket
 * @param objectKey the key (name) of the object to be copied
 * @param toBucket the name of the destination S3 bucket
 * @return a {@link CompletableFuture} that completes with the copy result as
 * a {@link String}
 * @throws RuntimeException if the URL could not be encoded or an S3
 * exception occurred during the copy
 */
public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    CompletableFuture<CopyObjectResponse> response =
        getAsyncClient().copyObject(copyReq);
}
```

```

        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during
copy", ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }

```

Use an [S3TransferManager](#) to [copy an object](#) from one bucket to another. View the [complete file](#) and [test](#).

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String
bucketName,
        String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
            .sourceBucket(bucketName)
            .sourceKey(key)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .build();

        CopyRequest copyRequest = CopyRequest.builder()
            .copyObjectRequest(copyObjectRequest)
            .build();

        Copy copy = transferManager.copy(copyRequest);

```

```
CompletedCopy completedCopy = copy.completionFuture().join();
return completedCopy.response().copyObjectResult().eTag();
}
```

- For API details, see [CopyObject](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Copy the object.

```
import {
  S3Client,
  CopyObjectCommand,
  ObjectNotInActiveTierError,
  waitUntilObjectExists,
} from "@aws-sdk/client-s3";

/**
 * Copy an S3 object from one bucket to another.
 *
 * @param {{
 *   sourceBucket: string,
 *   sourceKey: string,
 *   destinationBucket: string,
 *   destinationKey: string }} config
 */
export const main = async ({
  sourceBucket,
  sourceKey,
  destinationBucket,
  destinationKey,
}) => {
```

```
const client = new S3Client({});

try {
  await client.send(
    new CopyObjectCommand({
      CopySource: `${sourceBucket}/${sourceKey}`,
      Bucket: destinationBucket,
      Key: destinationKey,
    }),
  );
  await waitUntilObjectExists(
    { client },
    { Bucket: destinationBucket, Key: destinationKey },
  );
  console.log(
    `Successfully copied ${sourceBucket}/${sourceKey} to ${destinationBucket}/${destinationKey}`,
  );
} catch (caught) {
  if (caught instanceof ObjectNotInActiveTierError) {
    console.error(
      `Could not copy ${sourceKey} from ${sourceBucket}. Object is not in the active tier.`,
    );
  } else {
    throw caught;
  }
}
};
```

Copy the object on condition its ETag does not match the one provided.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
```

```
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;
  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
        "${sourceBucketName}". No such key exists.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket
        "${sourceBucketName}": ${caught.name}: ${caught.message}`
      );
    } else {
      throw caught;
    }
  }
};
```

```
// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Copy the object on condition its ETag does not match the one provided.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfNoneMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
        "${sourceBucketName}". No such key exists.`
      );
    }
  }
}
```

```
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Unable to copy object "${sourceKeyName}" to bucket
"${sourceBucketName}": ${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
```



```

const { errors, results } = loadArgs();
if (!errors) {
  main(results.values);
} else {
  console.error(errors.join("\n"));
}
}

```

Copy the object using on condition it has been created or modified in a given timeframe.

```

import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);

  const name = data.name;
  const client = new S3Client({});
  const copySource = `${sourceBucketName}/${sourceKeyName}`;
  const copiedKey = name + sourceKeyName;

```

```
try {
  const response = await client.send(
    new CopyObjectCommand({
      CopySource: copySource,
      Bucket: destinationBucketName,
      Key: copiedKey,
      CopySourceIfModifiedSince: date,
    }),
  );
  console.log("Successfully copied object to bucket.");
} catch (caught) {
  if (caught instanceof NoSuchKey) {
    console.error(
      `Error from S3 while copying object "${sourceKeyName}" from
"${sourceBucketName}". No such key exists.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while copying object from ${sourceBucketName}.
${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
  },
```

```

    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}

```

Copy the object using on condition it has not been created or modified in a given timeframe.

```

import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,

```

```
    sourceKeyName,
    destinationBucketName,
  }) => {
    const date = new Date();
    date.setDate(date.getDate() - 1);
    const client = new S3Client({});
    const name = data.name;
    const copiedKey = name + sourceKeyName;
    const copySource = `${sourceBucketName}/${sourceKeyName}`;

    try {
      const response = await client.send(
        new CopyObjectCommand({
          CopySource: copySource,
          Bucket: destinationBucketName,
          Key: copiedKey,
          CopySourceIfUnmodifiedSince: date,
        }),
      );
      console.log("Successfully copied object to bucket.");
    } catch (caught) {
      if (caught instanceof NoSuchKey) {
        console.error(
          `Error from S3 while copying object "${sourceKeyName}" from
"${sourceBucketName}". No such key exists.`
        );
      } else if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while copying object from ${sourceBucketName}.
${caught.name}: ${caught.message}`
        );
      } else {
        throw caught;
      }
    }
  };

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";
```

```
const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- For API details, see [CopyObject](#) in *Amazon SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedOperationException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- For API details, see [CopyObject](#) in *Amazon SDK for Kotlin API reference*.

PHP

SDK for PHP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Simple copy of an object.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
```

```
try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
    ]);
    echo "Copied $fileName to $folder/$fileName-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with object copying before continuing.");
}
```

- For API details, see [CopyObject](#) in *Amazon SDK for PHP API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command copies the object "sample.txt" from bucket "test-files" to the same bucket but with a new key of "sample-copy.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
sample-copy.txt
```

Example 2: This command copies the object "sample.txt" from bucket "test-files" to the bucket "backup-files" with a key of "sample-copy.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Example 3: This command downloads the object "sample.txt" from bucket "test-files" to a local file with name "local-sample.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt
```

Example 4: Downloads the single object to the specified file. The downloaded file will be found at c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder  
c:\downloads
```

Example 5: Downloads all objects that match the specified key prefix to the local folder. The relative key hierarchy will be preserved as subfolders in the overall download location.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:  
\downloads
```

- For API details, see [CopyObject](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command copies the object "sample.txt" from bucket "test-files" to the same bucket but with a new key of "sample-copy.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey  
sample-copy.txt
```

Example 2: This command copies the object "sample.txt" from bucket "test-files" to the bucket "backup-files" with a key of "sample-copy.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -  
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Example 3: This command downloads the object "sample.txt" from bucket "test-files" to a local file with name "local-sample.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-  
sample.txt
```

Example 4: Downloads the single object to the specified file. The downloaded file will be found at c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder  
c:\downloads
```


Example 5: Downloads all objects that match the specified key prefix to the local folder. The relative key hierarchy will be preserved as subfolders in the overall download location.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:\downloads
```

Example 6: Downloads a single object to a local file using multipart parallel download for improved performance on large files.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-sample.txt -UseMultipartDownload
```

Example 7: Downloads a large object using RANGE mode with a custom 64MB part size. RANGE mode uses HTTP byte-range requests and works for all objects regardless of how they were uploaded.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Example 8: Downloads all objects matching the key prefix to a local folder using multipart download with concurrent file downloads.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Example 9: Downloads a single object using multipart download with 20 concurrent connections for maximum throughput on high-bandwidth networks.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Example 10: Downloads all objects matching the key prefix to a local folder using multipart download, continuing past individual file failures instead of aborting.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- For API details, see [CopyObject](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
        key.
                           This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
                self.object.key}
```

```

    )
    dest_object.wait_until_exists()
    logger.info(
        "Copied object from %s:%s to %s:%s.",
        self.object.bucket_name,
        self.object.key,
        dest_object.bucket_name,
        dest_object.key,
    )
except ClientError:
    logger.exception(
        "Couldn't copy object from %s/%s to %s/%s.",
        self.object.bucket_name,
        self.object.key,
        dest_object.bucket_name,
        dest_object.key,
    )
    raise

```

Copy an object using a conditional request.

```

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,

```

```

        dest_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Copies an object from one Amazon S3 bucket to another with a conditional
        request.

        :param source_key: The key of the source object to copy.
        :param dest_key: The key of the destination object.
        :param source_bucket: The source bucket of the object.
        :param dest_bucket: The destination bucket of the object.
        :param condition_type: The type of condition to apply, e.g.
            'CopySourceIfMatch', 'CopySourceIfNoneMatch',
            'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            self.s3.copy_object(
                Bucket=dest_bucket,
                Key=dest_key,
                CopySource={"Bucket": source_bucket, "Key": source_key},
                **{condition_type: condition_value},
            )
            print(
                f"\tConditional copy successful for key {dest_key} in bucket {dest_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional copy failed: Precondition failed")
            elif error_code == "304": # Not modified error code.
                print("\tConditional copy failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise

```

- For API details, see [CopyObject](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Copy an object.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for
  #                                     copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
  # target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  # nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
    why: #{e.message}"
  end
end

# Example usage:
```

```
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
  #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Copy an object and add server-side encryption to the destination object.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
  # target key, and encrypt it.
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  nil.
  def copy_object(target_bucket, target_object_key, encryption)
```

```

    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
server_side_encryption: encryption)
    target_bucket.object(target_object_key)
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
why: #{e.message}"
    end
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key,
target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
#{target_object.bucket_name}:#{target_object.key} and "\
      "encrypted the target with #{target_object.server_side_encryption}
encryption."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- For API details, see [CopyObject](#) in *Amazon SDK for Ruby API Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object}
with etag {}",
        response
            .copy_object_result
            .unwrap_or_else(||
aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
    );
    Ok(())
}
```

- For API details, see [CopyObject](#) in *Amazon SDK for Rust API reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
    lo_s3->copyobject(  
        iv_bucket = iv_dest_bucket  
        iv_key = iv_dest_object  
        iv_copysource = |{ iv_src_bucket }/{ iv_src_object }| ).  
    MESSAGE 'Object copied to another bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
    MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [CopyObject](#) in *Amazon SDK for SAP ABAP API reference*.

Swift

SDK for Swift

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import AWSS3  
  
public func copyFile(from sourceBucket: String, name: String, to destBucket:  
String) async throws {
```

```
        let srcUrl = ("\"(sourceBucket)/\n(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)\n\n        let input = CopyObjectInput(\n            bucket: destBucket,\n            copySource: srcUrl,\n            key: name\n        )\n        do {\n            _ = try await client.copyObject(input: input)\n        }\n        catch {\n            print("ERROR: ", dump(error, name: "Copying an object."))\n            throw error\n        }\n    }\n}
```

- For API details, see [CopyObject](#) in *Amazon SDK for Swift API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use CreateBucket with an Amazon SDK or CLI

The following code examples show how to use CreateBucket.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Learn the basics](#)
- [Getting started with Amazon S3](#)
- [Getting started with Amazon SageMaker Feature Store](#)
- [Getting started with Config](#)
- [Manage large messages using S3](#)
- [Work with versioned objects](#)

.NET

Amazon SDK for .NET (v4)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Shows how to create a new Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>A boolean value representing the success or failure of
/// the bucket creation process.</returns>
public async Task<bool> CreateBucketAsync(string bucketName)
{
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
        };

        var response = await _amazonS3.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}
```

- For API details, see [CreateBucket](#) in *Amazon SDK for .NET API Reference*.

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create a bucket with object lock enabled.

```
/// <summary>
/// Create a new Amazon S3 bucket with object lock actions.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <param name="enableObjectLock">True to enable object lock on the
bucket.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
enableObjectLock)
{
    Console.WriteLine($"\\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
            ObjectLockEnabledForBucket = enableObjectLock,
        };

        var response = await _amazonS3.PutBucketAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}
```

- For API details, see [CreateBucket](#) in *Amazon SDK for .NET API Reference*.

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#     -b bucket_name  -- The name of the bucket to create.
#     -r region_code  -- The code for an AWS Region in which to
```

```

#                               create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function create_bucket"
        echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
        echo "  -b bucket_name    The name of the bucket. It must be globally
unique."
        echo "  [-r region_code]    The code for an AWS Region in which the bucket is
created."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;
            r) region_code="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done

    if [[ -z "$bucket_name" ]]; then
        errecho "ERROR: You must provide a bucket name with the -b parameter."
        usage
        return 1
    fi
}

```

```

fi

local bucket_config_arg
# A location constraint for "us-east-1" returns an error.
if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
fi

iecho "Parameters:\n"
iecho "    Bucket name:  $bucket_name"
iecho "    Region code:  $region_code"
iecho ""

# If the bucket already exists, we don't want to try to create it.
if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
fi

# shellcheck disable=SC2086
response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
fi
}

```

- For API details, see [CreateBucket](#) in *Amazon CLI Command Reference*.

C++

SDK for C++

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(
            Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
            std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
            " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CreateBucket](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

Example 1: To create a bucket

The following create-bucket example creates a bucket named `amzn-s3-demo-bucket`:

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region us-east-1
```

Output:

```
{  
  "Location": "/amzn-s3-demo-bucket"  
}
```

For more information, see [Creating a bucket](#) in the *Amazon S3 User Guide*.

Example 2: To create a bucket with owner enforced

The following create-bucket example creates a bucket named `amzn-s3-demo-bucket` that uses the bucket owner enforced setting for S3 Object Ownership.

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region us-east-1 \  
  --object-ownership BucketOwnerEnforced
```

Output:

```
{  
  "Location": "/amzn-s3-demo-bucket"  
}
```

For more information, see [Controlling ownership of objects and disabling ACLs](#) in the *Amazon S3 User Guide*.

Example 3: To create a bucket outside of the ``us-east-1`` region

The following `create-bucket` example creates a bucket named `amzn-s3-demo-bucket` in the `eu-west-1` region. Regions outside of `us-east-1` require the appropriate `LocationConstraint` to be specified in order to create the bucket in the desired region.

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region eu-west-1 \  
  --create-bucket-configuration LocationConstraint=eu-west-1
```

Output:

```
{  
  "Location": "http://amzn-s3-demo-bucket.s3.amazonaws.com/"  
}
```

For more information, see [Creating a bucket](#) in the *Amazon S3 User Guide*.

- For API details, see [CreateBucket](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create a bucket with default configuration.

```
import (  
  "bytes"  
  "context"  
  "errors"  
  "fmt"  
  "io"  
  "log"  
  "os"  
  "time"
```

```
"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
string) error {
    _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
        Bucket: aws.String(name),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    })
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", name)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", name)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
        }
    }
}
```

```
}  
}  
return err  
}
```

Create a bucket with object locking and wait for it to exist.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// CreateBucketWithLock creates a new S3 bucket with optional object locking  
// enabled  
// and waits for the bucket to exist before returning.  
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,  
    region string, enableObjectLock bool) (string, error) {  
    input := &s3.CreateBucketInput{  
        Bucket: aws.String(bucket),  
        CreateBucketConfiguration: &types.CreateBucketConfiguration{  
            LocationConstraint: types.BucketLocationConstraint(region),  
        },  
    },
```

```
}

if enableObjectLock {
    input.ObjectLockEnabledForBucket = aws.Bool(true)
}

_, err := actor.S3Client.CreateBucket(ctx, input)
if err != nil {
    var owned *types.BucketAlreadyOwnedByYou
    var exists *types.BucketAlreadyExists
    if errors.As(err, &owned) {
        log.Printf("You already own bucket %s.\n", bucket)
        err = owned
    } else if errors.As(err, &exists) {
        log.Printf("Bucket %s already exists.\n", bucket)
        err = exists
    }
} else {
    err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
    }
}

return bucket, err
}
```

- For API details, see [CreateBucket](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create a bucket.

```
/**
 * Creates an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes when the bucket is
 * created and ready
 * @throws RuntimeException if there is a failure while creating the bucket
 */
public CompletableFuture<Void> createBucketAsync(String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
    return response.thenCompose(resp -> {
        S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
            s3Waiter.waitUntilBucketExists(bucketRequestWait);
        return waiterResponseFuture.thenAccept(waiterResponse -> {
            waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                logger.info(bucketName + " is ready");
            });
        });
    }).whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to create bucket", ex);
        }
    });
}
```

Create a bucket with object lock enabled.

```
// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}
```

- For API details, see [CreateBucket](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create the bucket.

```
import {
    BucketAlreadyExists,
    BucketAlreadyOwnedByYou,
    CreateBucketCommand,
    S3Client,
    waitUntilBucketExists,
} from "@aws-sdk/client-s3";
```

```
/**
 * Create an Amazon S3 bucket.
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Location } = await client.send(
      new CreateBucketCommand({
        // The name of the bucket. Bucket names are unique and have several other
        constraints.
        // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/
bucketnamingrules.html
        Bucket: bucketName,
      }),
    );
    await waitUntilBucketExists({ client }, { Bucket: bucketName });
    console.log(`Bucket created with location ${Location}`);
  } catch (caught) {
    if (caught instanceof BucketAlreadyExists) {
      console.error(
        `The bucket "${bucketName}" already exists in another AWS account. Bucket
names must be globally unique.`
      );
    }
    // WARNING: If you try to create a bucket in the North Virginia region,
    // and you already own a bucket in that region with the same name, this
    // error will not be thrown. Instead, the call will return successfully
    // and the ACL on that bucket will be reset.
    else if (caught instanceof BucketAlreadyOwnedByYou) {
      console.error(
        `The bucket "${bucketName}" already exists in this AWS account.`
      );
    } else {
      throw caught;
    }
  }
};
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).

- For API details, see [CreateBucket](#) in *Amazon SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
suspend fun createNewBucket(bucketName: String) {  
    val request =  
        CreateBucketRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.createBucket(request)  
        println("$bucketName is ready")  
    }  
}
```

- For API details, see [CreateBucket](#) in *Amazon SDK for Kotlin API reference*.

PHP

SDK for PHP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create a bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
```

```
try {
    $this->s3client->createBucket([
        'Bucket' => $this->bucketName,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $this->bucketName \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with bucket creation before continuing.");
}
```

- For API details, see [CreateBucket](#) in *Amazon SDK for PHP API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create a bucket with default settings.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def create(self, region_override=None):
```

```

        """
        Create an Amazon S3 bucket in the default Region for the account or in
the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this
is
                                not specified, the Region configured in your
shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
region})

            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
            raise error

```

Create a versioned bucket with a lifecycle configuration.

```

def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
noncurrent versions, which can slow down request performance.

```

Usage is shown in the `usage_demo_single_object` function at the end of this module.

```

:param bucket_name: The name of the bucket to create.
:param prefix: Identifies which objects are automatically expired under the
               configured lifecycle rules.
:return: The newly created bucket.
"""
try:
    bucket = s3.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            "LocationConstraint": s3.meta.client.meta.region_name
        },
    )
    logger.info("Created bucket %s.", bucket.name)
except ClientError as error:
    if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
        logger.warning("Bucket %s already exists! Using it.", bucket_name)
        bucket = s3.Bucket(bucket_name)
    else:
        logger.exception("Couldn't create bucket %s.", bucket_name)
        raise

try:
    bucket.Versioning().enable()
    logger.info("Enabled versioning on bucket %s.", bucket.name)
except ClientError:
    logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
    raise

try:
    expiration = 7
    bucket.LifecycleConfiguration().put(
        LifecycleConfiguration={
            "Rules": [
                {
                    "Status": "Enabled",
                    "Prefix": prefix,
                    "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                }
            ]
        }
    )

```

```
        }
    )
    logger.info(
        "Configured lifecycle to expire noncurrent versions after %s days "
        "on bucket %s.",
        expiration,
        bucket.name,
    )
except ClientError as error:
    logger.warning(
        "Couldn't configure lifecycle on bucket %s because %s. "
        "Continuing anyway.",
        bucket.name,
        error,
    )

return bucket
```

- For API details, see [CreateBucket](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
  # This is a client-side object until
  # create is called.
```

```
def initialize(bucket)
  @bucket = bucket
end

# Creates an Amazon S3 bucket in the specified AWS Region.
#
# @param region [String] The Region where the bucket is created.
# @return [Boolean] True when the bucket is created; otherwise, false.
def create?(region)
  @bucket.create(create_bucket_configuration: { location_constraint: region })
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create bucket. Here's why: #{e.message}"
  false
end

# Gets the Region where the bucket is located.
#
# @return [String] The location of the bucket.
def location
  if @bucket.nil?
    'None. You must create a bucket before you can get its location!'
  else
    @bucket.client.get_bucket_location(bucket:
@bucket.name).location_constraint
  end
rescue Aws::Errors::ServiceError => e
  "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [CreateBucket](#) in *Amazon SDK for Ruby API Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
S3ExampleError> {
    let constraint =
aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
    task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        }
    })
}
```

```

        } else {
            Err(S3ExampleError::from(err))
        }
    })
}

```

- For API details, see [CreateBucket](#) in *Amazon SDK for Rust API reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

TRY.
    " determine our region from our session
    DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
    DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
    " When in the us-east-1 region, you must not specify a constraint
    " In all other regions, specify the region as the constraint
    IF lv_region = 'us-east-1'.
        CLEAR lo_constraint.
    ELSE.
        lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
    ENDIF.

    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
        io_createbucketconfiguration = lo_constraint ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
    CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
    CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.

```


- For API details, see [CreateBucket](#) in *Amazon SDK for SAP ABAP API reference*.

Swift

SDK for Swift

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import AWSS3

public func createBucket(name: String) async throws {
    var input = CreateBucketInput(
        bucket: name
    )

    // For regions other than "us-east-1", you must set the
    locationConstraint in the createBucketConfiguration.
    // For more information, see LocationConstraint in the S3 API guide.
    // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
    if let region = configuration.region {
        if region != "us-east-1" {
            input.createBucketConfiguration =
S3ClientTypes.CreateBucketConfiguration(locationConstraint:
S3ClientTypes.BucketLocationConstraint(rawValue: region))
        }
    }

    do {
        _ = try await client.createBucket(input: input)
    }
    catch let error as BucketAlreadyOwnedByYou {
        print("The bucket '\(name)' already exists and is owned by you. You
may wish to ignore this exception.")
        throw error
    }
}
```

```
        catch {
            print("ERROR: ", dump(error, name: "Creating a bucket"))
            throw error
        }
    }
```

- For API details, see [CreateBucket](#) in *Amazon SDK for Swift API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use CreateMultiRegionAccessPoint with an Amazon SDK

The following code example shows how to use CreateMultiRegionAccessPoint.

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Configure the S3 control client to send request to the us-west-2 Region.

```
suspend fun createS3ControlClient(): S3ControlClient {
    // Configure your S3ControlClient to send requests to US West
    (Oregon).
    val s3Control = S3ControlClient.fromEnvironment {
        region = "us-west-2"
    }
    return s3Control
}
```

Create the Multi-Region Access Point.

```

suspend fun createMrap(
    s3Control: S3ControlClient,
    accountIdParam: String,
    bucketName1: String,
    bucketName2: String,
    mrmapName: String,
): String {
    println("Creating MRAP ...")
    val createMrapResponse: CreateMultiRegionAccessPointResponse =
        s3Control.createMultiRegionAccessPoint {
            accountId = accountIdParam
            clientToken = UUID.randomUUID().toString()
            details {
                name = mrmapName
                regions = listOf(
                    Region {
                        bucket = bucketName1
                    },
                    Region {
                        bucket = bucketName2
                    },
                )
            }
        }
    val requestToken: String? = createMrapResponse.requestTokenArn

    // Use the request token to check for the status of the
    CreateMultiRegionAccessPoint operation.
    if (requestToken != null) {
        waitForSucceededStatus(s3Control, requestToken, accountIdParam)
        println("MRAP created")
    }

    val getMrapResponse =
        s3Control.getMultiRegionAccessPoint(
            input = GetMultiRegionAccessPointRequest {
                accountId = accountIdParam
                name = mrmapName
            },
        )
    val mrmapAlias = getMrapResponse.accessPoint?.alias
    return "arn:aws:s3:$accountIdParam:accesspoint/$mrmapAlias"
}

```

Wait for the Multi-Region Access Point to become available.

```
suspend fun waitForSucceededStatus(
    s3Control: S3ControlClient,
    requestToken: String,
    accountIdParam: String,
    timeBetweenChecks: Duration = 1.minutes,
) {
    var describeResponse: DescribeMultiRegionAccessPointOperationResponse
    describeResponse = s3Control.describeMultiRegionAccessPointOperation(
        input = DescribeMultiRegionAccessPointOperationRequest {
            accountId = accountIdParam
            requestTokenArn = requestToken
        },
    )

    var status: String? = describeResponse.asyncOperation?.requestStatus
    while (status != "SUCCEEDED") {
        delay(timeBetweenChecks)
        describeResponse =
            s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )
        status = describeResponse.asyncOperation?.requestStatus
        println(status)
    }
}
```

- For more information, see [Amazon SDK for Kotlin developer guide](#).
- For API details, see [CreateMultiRegionAccessPoint](#) in *Amazon SDK for Kotlin API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use CreateMultipartUpload with an Amazon SDK or CLI

The following code examples show how to use CreateMultipartUpload.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Perform a multipart copy](#)
- [Use checksums](#)
- [Work with Amazon S3 object integrity](#)

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
&key,
                                Aws::S3::Model::ChecksumAlgorithm
checksumAlgorithm,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
}
```

```
Aws::S3::Model::CreateMultipartUploadOutcome outcome =
    client.CreateMultipartUpload(request);

Aws::String uploadID;
if (outcome.IsSuccess()) {
    uploadID = outcome.GetResult().GetUploadId();
} else {
    std::cerr << "Error creating multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
}

return uploadID;
}
```

- For API details, see [CreateMultipartUpload](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command creates a multipart upload in the bucket `amzn-s3-demo-bucket` with the key `multipart/01`:

```
aws s3api create-multipart-upload --bucket amzn-s3-demo-bucket --key
'multipart/01'
```

Output:

```
{
  "Bucket": "amzn-s3-demo-bucket",
  "UploadId":
"dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
  "Key": "multipart/01"
}
```

The completed file will be named `01` in a folder called `multipart` in the bucket `amzn-s3-demo-bucket`. Save the upload ID, key and bucket name for use with the `upload-part` command.

- For API details, see [CreateMultipartUpload](#) in *Amazon CLI Command Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
```

```

        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }

```

```

// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
    CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;

```

- For API details, see [CreateMultipartUpload](#) in *Amazon SDK for Rust API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use CreatePresignedPost with an Amazon SDK

The following code example shows how to use CreatePresignedPost.

.NET

Amazon SDK for .NET (v4)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create a presigned POST URL.

```
/// <summary>
/// Create a presigned POST URL with conditions.
/// </summary>
/// <param name="s3Client">The Amazon S3 client.</param>
/// <param name="bucketName">The name of the bucket.</param>
/// <param name="objectKey">The object key (path) where the uploaded file
will be stored.</param>
/// <param name="expires">When the presigned URL expires.</param>
/// <param name="fields">Dictionary of fields to add to the form.</param>
/// <param name="conditions">List of conditions to apply.</param>
/// <returns>A CreatePresignedPostResponse object with URL and form fields.</
returns>
public async Task<CreatePresignedPostResponse> CreatePresignedPostAsync(
    IAmazonS3 s3Client,
    string bucketName,
    string objectKey,
    DateTime expires,
    Dictionary<string, string>? fields = null,
    List<S3PostCondition>? conditions = null)
{
    var request = new CreatePresignedPostRequest
    {
        BucketName = bucketName,
        Key = objectKey,
        Expires = expires
    };
};
```

```
// Add custom fields if provided
if (fields != null)
{
    foreach (var field in fields)
    {
        request.Fields.Add(field.Key, field.Value);
    }
}

// Add conditions if provided
if (conditions != null)
{
    foreach (var condition in conditions)
    {
        request.Conditions.Add(condition);
    }
}

return await s3Client.CreatePresignedPostAsync(request);
}
```

- For API details, see [CreatePresignedPost](#) in *Amazon SDK for .NET API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucket with an Amazon SDK or CLI

The following code examples show how to use DeleteBucket.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Learn the basics](#)
- [Getting started with Amazon S3](#)
- [Getting started with Amazon SageMaker Feature Store](#)
- [Getting started with Config](#)

.NET

Amazon SDK for .NET (v4)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Shows how to delete an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
/// <returns>A boolean value that represents the success or failure of
/// the delete operation.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest { BucketName = bucketName, };

        await _amazonS3.DeleteBucketAsync(request);
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting bucket: {ex.Message}");
        return false;
    }
}
```

- For API details, see [DeleteBucket](#) in *Amazon SDK for .NET API Reference*.

Bash

Amazon CLI with Bash script

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api delete-bucket \
        --bucket "$bucket_name")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
        return 1
    fi
}
```

```
    fi  
}
```

- For API details, see [DeleteBucket](#) in *Amazon CLI Command Reference*.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,  
                               const Aws::S3::S3ClientConfiguration &clientConfig)  
{  
  
    Aws::S3::S3Client client(clientConfig);  
  
    Aws::S3::Model::DeleteBucketRequest request;  
    request.SetBucket(bucketName);  
  
    Aws::S3::Model::DeleteBucketOutcome outcome =  
        client.DeleteBucket(request);  
  
    if (!outcome.IsSuccess()) {  
        const Aws::S3::S3Error &err = outcome.GetError();  
        std::cerr << "Error: deleteBucket: " <<  
                    err.GetExceptionName() << ": " << err.GetMessage() <<  
std::endl;  
    } else {  
        std::cout << "The bucket was deleted" << std::endl;  
    }  
  
    return outcome.IsSuccess();  
}
```

- For API details, see [DeleteBucket](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command deletes a bucket named `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket --bucket amzn-s3-demo-bucket --region us-east-1
```

- For API details, see [DeleteBucket](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)
```

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is
// returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
error {
    _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
        Bucket: aws.String(bucketName)})
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        } else {
            log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
        }
    } else {
        err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
                bucketName)
        } else {
            log.Printf("Deleted %s.\n", bucketName)
        }
    }
    return err
}
```

- For API details, see [DeleteBucket](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/**
 * Deletes an S3 bucket asynchronously.
 *
 * @param bucket the name of the bucket to be deleted
 * @return a {@link CompletableFuture} that completes when the bucket
 * deletion is successful, or throws a {@link RuntimeException}
 * if an error occurs during the deletion process
 */
public CompletableFuture<Void> deleteBucketAsync(String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(bucket + " was deleted.");
        } else {
            throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
        }
    });
    return response.thenApply(r -> null);
}
```

- For API details, see [DeleteBucket](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete the bucket.

```
import {
  DeleteBucketCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Delete an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new DeleteBucketCommand({
    Bucket: bucketName,
  });

  try {
    await client.send(command);
    console.log("Bucket was deleted.");
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting bucket. The bucket doesn't exist.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting the bucket. ${caught.name}:`
        `${caught.message}`
      );
    }
  }
}
```

```
    );  
  } else {  
    throw caught;  
  }  
}  
};
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteBucket](#) in *Amazon SDK for JavaScript API Reference*.

PHP

SDK for PHP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete an empty bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);  
  
try {  
    $this->s3client->deleteBucket([  
        'Bucket' => $this->bucketName,  
    ]);  
    echo "Deleted bucket $this->bucketName.\n";  
} catch (Exception $exception) {  
    echo "Failed to delete $this->bucketName with error: " . $exception-  
>getMessage();  
    exit("Please fix error with bucket deletion before continuing.");  
}
```

- For API details, see [DeleteBucket](#) in *Amazon SDK for PHP API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command removes all objects and object versions from the bucket 'test-files' and then deletes the bucket. The command will prompt for confirmation before proceeding. Add the `-Force` switch to suppress confirmation. Note that buckets that are not empty cannot be deleted.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- For API details, see [DeleteBucket](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command removes all objects and object versions from the bucket 'test-files' and then deletes the bucket. The command will prompt for confirmation before proceeding. Add the `-Force` switch to suppress confirmation. Note that buckets that are not empty cannot be deleted.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- For API details, see [DeleteBucket](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
```

```

        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise

```

- For API details, see [DeleteBucket](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
  answer = gets.chomp.downcase
  if answer == 'y'

```

```

        bucket.objects.batch_delete!
        bucket.delete
        puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
end

```

- For API details, see [DeleteBucket](#) in *Amazon SDK for Ruby API Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}

```

```
}
```

- For API details, see [DeleteBucket](#) in *Amazon SDK for Rust API reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
  
    lo_s3->deletebucket(  
        iv_bucket = iv_bucket_name ).  
    MESSAGE 'Deleted S3 bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteBucket](#) in *Amazon SDK for SAP ABAP API reference*.

Swift

SDK for Swift

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import AWSS3
```

```
public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    do {
        _ = try await client.deleteBucket(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a bucket"))
        throw error
    }
}
```

- For API details, see [DeleteBucket](#) in *Amazon SDK for Swift API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucketAnalyticsConfiguration with a CLI

The following code examples show how to use DeleteBucketAnalyticsConfiguration.

CLI

Amazon CLI

To delete an analytics configuration for a bucket

The following delete-bucket-analytics-configuration example removes the analytics configuration for the specified bucket and ID.

```
aws s3api delete-bucket-analytics-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 1
```

This command produces no output.

- For API details, see [DeleteBucketAnalyticsConfiguration](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: The command removes the analytics filter with name 'testfilter' in the given S3 bucket.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AnalyticsId 'testfilter'
```

- For API details, see [DeleteBucketAnalyticsConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: The command removes the analytics filter with name 'testfilter' in the given S3 bucket.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AnalyticsId 'testfilter'
```

- For API details, see [DeleteBucketAnalyticsConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucketCors with an Amazon SDK or CLI

The following code examples show how to use DeleteBucketCors.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).


```
/// <summary>
/// Deletes a CORS configuration from an Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used
/// to delete the CORS configuration from the bucket.</param>
private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
{
    DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest()
    {
        BucketName = BucketName,
    };
    await client.DeleteCORSConfigurationAsync(request);
}
```

- For API details, see [DeleteBucketCors](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

The following command deletes a Cross-Origin Resource Sharing configuration from a bucket named `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-cors --bucket amzn-s3-demo-bucket
```

- For API details, see [DeleteBucketCors](#) in *Amazon CLI Command Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
        try:
            self.bucket.Cors().delete()
            logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete CORS from bucket '%s'.",
self.bucket.name)
            raise
```

- For API details, see [DeleteBucketCors](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
    #{e.message}"
    false
  end
end
```

- For API details, see [DeleteBucketCors](#) in *Amazon SDK for Ruby API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.
  lo_s3->deletebucketcors(
```

```
        iv_bucket = iv_bucket_name ).  
    MESSAGE 'Bucket CORS configuration deleted.' TYPE 'I'.  
    CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteBucketCors](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucketEncryption with a CLI

The following code examples show how to use DeleteBucketEncryption.

CLI

Amazon CLI

To delete the server-side encryption configuration of a bucket

The following delete-bucket-encryption example deletes the server-side encryption configuration of the specified bucket.

```
aws s3api delete-bucket-encryption \  
    --bucket amzn-s3-demo-bucket
```

This command produces no output.

- For API details, see [DeleteBucketEncryption](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This disables the encryption enabled for the S3 bucket provided.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
target "s3casetestbucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- For API details, see [DeleteBucketEncryption](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This disables the encryption enabled for the S3 bucket provided.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
target "s3casetestbucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- For API details, see [DeleteBucketEncryption](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucketInventoryConfiguration with a CLI

The following code examples show how to use DeleteBucketInventoryConfiguration.

CLI

Amazon CLI

To delete the inventory configuration of a bucket

The following delete-bucket-inventory-configuration example deletes the inventory configuration with ID 1 for the specified bucket.

```
aws s3api delete-bucket-inventory-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 1
```

This command produces no output.

- For API details, see [DeleteBucketInventoryConfiguration](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command removes the inventory named 'testInventoryName' corresponding to the given S3 bucket.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -  
InventoryId 'testInventoryName'
```

Output:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketInventoryConfiguration  
(DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- For API details, see [DeleteBucketInventoryConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command removes the inventory named 'testInventoryName' corresponding to the given S3 bucket.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -  
InventoryId 'testInventoryName'
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
(DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- For API details, see [DeleteBucketInventoryConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucketLifecycle with an Amazon SDK or CLI

The following code examples show how to use DeleteBucketLifecycle.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// This method removes the Lifecycle configuration from the named
/// S3 bucket.
/// </summary>
/// <param name="client">The S3 client object used to call
/// the RemoveLifecycleConfigAsync method.</param>
/// <param name="bucketName">A string representing the name of the
/// S3 bucket from which the configuration will be removed.</param>
public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client,
string bucketName)
```

```
{
    var request = new DeleteLifecycleConfigurationRequest()
    {
        BucketName = bucketName,
    };
    await client.DeleteLifecycleConfigurationAsync(request);
}
```

- For API details, see [DeleteBucketLifecycle](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

The following command deletes a lifecycle configuration from a bucket named `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-lifecycle --bucket amzn-s3-demo-bucket
```

- For API details, see [DeleteBucketLifecycle](#) in *Amazon CLI Command Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
```



```

        """
        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_lifecycle_configuration(self):
        """
        Remove the lifecycle configuration from the specified bucket.
        """
        try:
            self.bucket.LifecycleConfiguration().delete()
            logger.info(
                "Deleted lifecycle configuration for bucket '%s'.",
                self.bucket.name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete lifecycle configuration for bucket '%s'.",
                self.bucket.name,
            )
            raise

```

- For API details, see [DeleteBucketLifecycle](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

TRY.
  lo_s3->deletebucketlifecycle(
    iv_bucket = iv_bucket_name ).
  MESSAGE 'Bucket lifecycle configuration deleted.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.

```

```
MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteBucketLifecycle](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucketMetricsConfiguration with a CLI

The following code examples show how to use DeleteBucketMetricsConfiguration.

CLI

Amazon CLI

To delete a metrics configuration for a bucket

The following delete-bucket-metrics-configuration example removes the metrics configuration for the specified bucket and ID.

```
aws s3api delete-bucket-metrics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 123
```

This command produces no output.

- For API details, see [DeleteBucketMetricsConfiguration](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: The command removes the metrics filter with name 'testmetrics' in the given S3 bucket.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- For API details, see [DeleteBucketMetricsConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: The command removes the metrics filter with name 'testmetrics' in the given S3 bucket.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- For API details, see [DeleteBucketMetricsConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucketPolicy with an Amazon SDK or CLI

The following code examples show how to use DeleteBucketPolicy.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,  
                                     const Aws::S3::S3ClientConfiguration  
&clientConfig) {  
    Aws::S3::S3Client client(clientConfig);  
  
    Aws::S3::Model::DeleteBucketPolicyRequest request;  
    request.SetBucket(bucketName);
```

```
Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
client.DeleteBucketPolicy(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: deleteBucketPolicy: " <<
        err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
} else {
    std::cout << "Policy was deleted from the bucket." << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [DeleteBucketPolicy](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command deletes a bucket policy from a bucket named `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-policy --bucket amzn-s3-demo-bucket
```

- For API details, see [DeleteBucketPolicy](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the policy from
(for example, bucket1).""";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes the S3 bucket policy for the specified bucket.

```

```
*
* @param s3 the {@link S3Client} instance to use for the operation
* @param bucketName the name of the S3 bucket for which the policy should be
deleted
*
* @throws S3Exception if there is an error deleting the bucket policy
*/
public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
    DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketPolicy(delReq);
        System.out.println("Done!");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteBucketPolicy](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete the bucket policy.

```
import {
    DeleteBucketPolicyCommand,
    S3Client,
    S3ServiceException,
```

```
} from "@aws-sdk/client-s3";

/**
 * Remove the policy from an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Bucket policy deleted from "${bucketName}".`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}. The bucket
        doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}.  ${caught.name}:
        ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteBucketPolicy](#) in *Amazon SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {  
    val request =  
        DeleteBucketPolicyRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.deleteBucketPolicy(request)  
        println("Done!")  
    }  
}
```

- For API details, see [DeleteBucketPolicy](#) in *Amazon SDK for Kotlin API reference*.

PowerShell

Tools for PowerShell V4

Example 1: The command removes the bucket policy associated with the given S3 bucket.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [DeleteBucketPolicy](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: The command removes the bucket policy associated with the given S3 bucket.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```


- For API details, see [DeleteBucketPolicy](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception(
                "Couldn't delete policy for bucket '%s'.", self.bucket.name
            )
            raise
```

- For API details, see [DeleteBucketPolicy](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  # configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's
    why: #{e.message}"
    false
  end
end
```

- For API details, see [DeleteBucketPolicy](#) in *Amazon SDK for Ruby API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
    lo_s3->deletebucketpolicy(  
        iv_bucket = iv_bucket_name ).  
    MESSAGE 'Bucket policy deleted.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteBucketPolicy](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucketReplication with a CLI

The following code examples show how to use DeleteBucketReplication.

CLI

Amazon CLI

The following command deletes a replication configuration from a bucket named `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-replication --bucket amzn-s3-demo-bucket
```

- For API details, see [DeleteBucketReplication](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Deletes the replication configuration associated with the bucket named 'amzn-s3-demo-bucket'. Note that this operation requires permission for the `s3:DeleteReplicationConfiguration` action. You will be prompted for confirmation before the operation proceeds - to suppress confirmation, use the `-Force` switch.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- For API details, see [DeleteBucketReplication](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Deletes the replication configuration associated with the bucket named 'amzn-s3-demo-bucket'. Note that this operation requires permission for the `s3:DeleteReplicationConfiguration` action. You will be prompted for confirmation before the operation proceeds - to suppress confirmation, use the `-Force` switch.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- For API details, see [DeleteBucketReplication](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucketTagging with a CLI

The following code examples show how to use `DeleteBucketTagging`.

CLI

Amazon CLI

The following command deletes a tagging configuration from a bucket named `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-tagging --bucket amzn-s3-demo-bucket
```

- For API details, see [DeleteBucketTagging](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command removes all the tags associated with the given S3 bucket.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- For API details, see [DeleteBucketTagging](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command removes all the tags associated with the given S3 bucket.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- For API details, see [DeleteBucketTagging](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucketWebsite with an Amazon SDK or CLI

The following code examples show how to use DeleteBucketWebsite.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration
                                     &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
        client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
        std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteBucketWebsite](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command deletes a website configuration from a bucket named `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-website --bucket amzn-s3-demo-bucket
```

- For API details, see [DeleteBucketWebsite](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""
```

```

Usage:      <bucketName>

Where:
    bucketName - The Amazon S3 bucket to delete the website
configuration from.
    "";

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
System.out.format("Deleting website configuration for Amazon S3 bucket:
%s\n", bucketName);
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

deleteBucketWebsiteConfig(s3, bucketName);
System.out.println("Done!");
s3.close();
}

/**
 * Deletes the website configuration for an Amazon S3 bucket.
 *
 * @param s3 The {@link S3Client} instance used to interact with Amazon S3.
 * @param bucketName The name of the S3 bucket for which the website
configuration should be deleted.
 * @throws S3Exception If an error occurs while deleting the website
configuration.
 */
public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName)
{
    DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketWebsite(delReq);
    } catch (S3Exception e) {

```



```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.out.println("Failed to delete website configuration!");
        System.exit(1);
    }
}
```

- For API details, see [DeleteBucketWebsite](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete the website configuration from the bucket.

```
import {
    DeleteBucketWebsiteCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the website configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
    const client = new S3Client({});

    try {
        await client.send(
            new DeleteBucketWebsiteCommand({
                Bucket: bucketName,
            }),
        );
        // The response code will be successful for both removed configurations and
```

```
// configurations that did not exist in the first place.
console.log(
    `The bucket "${bucketName}" is not longer configured as a website, or it
never was.` ,
);
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while removing website configuration from ${bucketName}.
The bucket doesn't exist.` ,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while removing website configuration from ${bucketName}.
${caught.name}: ${caught.message}` ,
        );
    } else {
        throw caught;
    }
}
};
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteBucketWebsite](#) in *Amazon SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command disables the static website hosting property of the given S3 bucket.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Confirm
```

```
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target  
"amzn-s3-demo-bucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- For API details, see [DeleteBucketWebsite](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command disables the static website hosting property of the given S3 bucket.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target  
"amzn-s3-demo-bucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- For API details, see [DeleteBucketWebsite](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteObject with an Amazon SDK or CLI

The following code examples show how to use DeleteObject.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Work with Amazon S3 object integrity](#)
- [Work with versioned objects](#)

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete an object in a non-versioned S3 bucket.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete an object from a non-versioned Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteObject
{
    /// <summary>
    /// The Main method initializes the necessary variables and then calls
    /// the DeleteObjectNonVersionedBucketAsync method to delete the object
    /// named by the keyName parameter.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string keyName = "testfile.txt";

        // If the Amazon S3 bucket is located in an AWS Region other than the
        // Region of the default account, define the AWS Region for the
        // Amazon S3 bucket in your call to the AmazonS3Client constructor.
        // For example RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();
        await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
    }

    /// <summary>
```

```

    /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
    /// desired object from the named bucket.
    /// </summary>
    /// <param name="client">An initialized Amazon S3 client used to delete
    /// an object from an Amazon S3 bucket.</param>
    /// <param name="bucketName">The name of the bucket from which the
    /// object will be deleted.</param>
    /// <param name="keyName">The name of the object to delete.</param>
    public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
    {
        try
        {
            var deleteObjectRequest = new DeleteObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
            };

            Console.WriteLine($"Deleting object: {keyName}");
            await client.DeleteObjectAsync(deleteObjectRequest);
            Console.WriteLine($"Object: {keyName} deleted from
{bucketName}.");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' when deleting an object.");
        }
    }
}

```

Delete an object in a versioned S3 bucket.

```

using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example creates an object in an Amazon Simple Storage Service

```

```

    /// (Amazon S3) bucket and then deletes the object version that was
    /// created.
    /// </summary>
    public class DeleteObjectVersion
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "verstioned-object.txt";

            // If the AWS Region of the default user is different from the AWS
            // Region of the Amazon S3 bucket, pass the AWS Region of the
            // bucket region to the Amazon S3 client object's constructor.
            // Define it like this:
            //      RegionEndpoint bucketRegion = RegionEndpoint.USSouth1;
            IAmazonS3 client = new AmazonS3Client();

            await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// This method creates and then deletes a versioned object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// create and delete the object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// object will be created and deleted.</param>
        /// <param name="keyName">The key name of the object to create.</param>
        public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(client, bucketName,
keyName);

                // Delete the object by specifying an object key and a version
ID.

                DeleteObjectRequest request = new DeleteObjectRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID,

```

```

        };

        Console.WriteLine("Deleting an object");
        await client.DeleteObjectAsync(request);
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: {ex.Message}");
    }
}

/// <summary>
/// This method is used to create the temporary Amazon S3 object.
/// </summary>
/// <param name="client">The initialized Amazon S3 object which will be
used
/// to create the temporary Amazon S3 object.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
object
/// will be created.</param>
/// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
/// <returns>The Version ID of the created object.</returns>
public static async Task<string> PutAnObject(IAmazonS3 client, string
bucketName, string objectKey)
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = bucketName,
        Key = objectKey,
        ContentBody = "This is the content body!",
    };

    PutObjectResponse response = await client.PutObjectAsync(request);
    return response.VersionId;
}
}

```

- For API details, see [DeleteObject](#) in *Amazon SDK for .NET API Reference*.

Bash

Amazon CLI with Bash script

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_item_in_bucket
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - The key (file name) in the bucket to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_item_in_bucket() {
    local bucket_name=$1
    local key=$2
    local response

    response=$(aws s3api delete-object \
        --bucket "$bucket_name" \
        --key "$key")

    # shellcheck disable=SC2181
```



```

if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
    return 1
fi
}

```

- For API details, see [DeleteObject](#) in *Amazon CLI Command Reference*.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
        .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
        std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }
}

```

```
    return outcome.IsSuccess();  
}
```

- For API details, see [DeleteObject](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command deletes an object named `test.txt` from a bucket named `amzn-s3-demo-bucket`:

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key test.txt
```

If bucket versioning is enabled, the output will contain the version ID of the delete marker:

```
{  
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tND1WlGq",  
  "DeleteMarker": true  
}
```

For more information about deleting objects, see *Deleting Objects in the Amazon S3 Developer Guide*.

- For API details, see [DeleteObject](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (  
    "bytes"
```

```

"context"
"errors"
"fmt"
"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
    string, versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:    aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {

```

```
case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
case "InvalidArgument":
    if bypassGovernance {
        log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
        err = nil
    }
}
} else {
    err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
        time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
    } else {
        deleted = true
    }
}
return deleted, err
}
```

- For API details, see [DeleteObject](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/**
 * Deletes an object from an S3 bucket asynchronously.
```

```
*
* @param bucketName the name of the S3 bucket
* @param key         the key (file name) of the object to be deleted
* @return a {@link CompletableFuture} that completes when the object has
been deleted
*/
public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(key + " was deleted");
        } else {
            throw new RuntimeException("An S3 exception occurred during
delete", ex);
        }
    });

    return response.thenApply(r -> null);
}
```

- For API details, see [DeleteObject](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete an object.

```
import {
  DeleteObjectCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete one object from an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    await waitUntilObjectNotExists(
      { client },
      { Bucket: bucketName, Key: key },
    );
    // A successful delete, or a delete for a non-existent object, both return
    // a 204 response code.
    console.log(
      `The object "${key}" from bucket "${bucketName}" was deleted, or it didn't
exist.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}. The bucket
doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
```

```
        `Error from S3 while deleting object from ${bucketName}. ${caught.name}:  
        ${caught.message}`,  
        );  
    } else {  
        throw caught;  
    }  
}  
};
```

- For API details, see [DeleteObject](#) in *Amazon SDK for JavaScript API Reference*.

PHP

SDK for PHP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
public function deleteObject(string $bucketName, string $fileName, array  
$args = [])  
{  
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],  
$args);  
    try {  
        $this->client->deleteObject($parameters);  
        if ($this->verbose) {  
            echo "Deleted the object named: $fileName from $bucketName.\n";  
        }  
    } catch (AwsException $exception) {  
        if ($this->verbose) {  
            echo "Failed to delete $fileName from $bucketName with error:  
{ $exception->getMessage() }\n";  
            echo "Please fix error with object deletion before continuing.";  
        }  
        throw $exception;  
    }  
}
```

- For API details, see [DeleteObject](#) in *Amazon SDK for PHP API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete an object.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                               that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def delete(self):
        """
        Deletes the object.
        """
        try:
            self.object.delete()
            self.object.wait_until_not_exists()
            logger.info(
                "Deleted object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
```



```

except ClientError:
    logger.exception(
        "Couldn't delete object '%s' from bucket '%s'.",
        self.object.key,
        self.object.bucket_name,
    )
    raise

```

Roll an object back to a previous version by deleting later versions of the object.

```

def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:

```

```

        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
            f"{object_key}."
        )

```

Revive a deleted object by removing the object's active delete marker.

```

def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
    marker.
    By removing the delete marker, we make the previous version the latest
    version
    and the object then presents as not deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]

```

```

        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
        elif "Versions" in response:
            logger.warning("Got an active version for %s, nothing to do.",
                object_key)
        else:
            logger.error("Couldn't get any version info for %s.", object_key)

```

Create a Lambda handler that removes a delete marker from an S3 object. This handler can be used to efficiently clean up extraneous delete markers in a versioned bucket.

```

import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")

def lambda_handler(event, context):
    """

```

```

    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of
the
           operation. When the result code is TemporaryFailure, S3 retries the
           operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote_plus(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a
delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
marker."
            )

```

```

        logger.debug(response)
        logger.warning(result_string)
    except ClientError as error:
        delete_marker = error.response["ResponseMetadata"]
["HTTPHeaders"].get(
            "x-amz-delete-marker", "false"
        )
        if delete_marker == "true":
            logger.info(
                "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
            )
            try:
                s3.delete_object(
                    Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                )
                result_code = "Succeeded"
                result_string = (
                    f"Successfully removed delete marker "
                    f"{obj_version_id} from object {obj_key}."
                )
                logger.info(result_string)
            except ClientError as error:
                # Mark request timeout as a temporary failure so it will be
retried.

                if error.response["Error"]["Code"] == "RequestTimeout":
                    result_code = "TemporaryFailure"
                    result_string = (
                        f"Attempt to remove delete marker from "
                        f"object {obj_key} timed out."
                    )
                    logger.info(result_string)
                else:
                    raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)

```

```
finally:
    results.append(
        {
            "taskId": task_id,
            "resultCode": result_code,
            "resultString": result_string,
        }
    )
return {
    "invocationSchemaVersion": invocation_schema_version,
    "treatMissingKeysAs": "PermanentFailure",
    "invocationId": invocation_id,
    "results": results,
}
```

- For API details, see [DeleteObject](#) in *Amazon SDK for Python (Boto3) API Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// Delete an object from a bucket.
pub async fn remove_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    key: &str,
) -> Result<(), S3ExampleError> {
    client
        .delete_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await?;
```

```
// There are no modeled errors to handle when deleting an object.  
  
Ok()  
}
```

- For API details, see [DeleteObject](#) in *Amazon SDK for Rust API reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
    lo_s3->deleteobject(  
        iv_bucket = iv_bucket_name  
        iv_key = iv_object_key ).  
    MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteObject](#) in *Amazon SDK for SAP ABAP API reference*.

Swift

SDK for Swift

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import AWSS3

public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a file. "))
        throw error
    }
}
```

- For API details, see [DeleteObject](#) in *Amazon SDK for Swift API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteObjectTagging with a CLI

The following code examples show how to use DeleteObjectTagging.

CLI

Amazon CLI

To delete the tag sets of an object

The following delete-object-tagging example deletes the tag with the specified key from the object doc1.rtf.

```
aws s3api delete-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc1.rtf
```

This command produces no output.

- For API details, see [DeleteObjectTagging](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command removes all the tags associated with the object with key 'testfile.txt' in the given S3 Bucket.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

Output:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target  
"testfile.txt".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y  
testfile.txt
```

- For API details, see [DeleteObjectTagging](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command removes all the tags associated with the object with key 'testfile.txt' in the given S3 Bucket.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

Output:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target  
"testfile.txt".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

```
testfile.txt
```

- For API details, see [DeleteObjectTagging](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteObjects with an Amazon SDK or CLI

The following code examples show how to use DeleteObjects.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Learn the basics](#)
- [Delete all objects in a bucket](#)
- [Getting started with Amazon S3](#)

.NET

Amazon SDK for .NET (v4)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>
public async Task<bool> DeleteBucketContentsAsync(string bucketName)
{
```

```
// Iterate over the contents of the bucket and delete all objects.
try
{
    // Delete all objects in the bucket.
    var deleteList = await ListBucketContentsAsync(bucketName, false);
    if (deleteList != null && deleteList.Any())
    {
        await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
        {
            BucketName = bucketName,
            Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
        });
    }

    return true;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error deleting objects: {ex.Message}");
    return false;
}
}
```

- For API details, see [DeleteObjects](#) in *Amazon SDK for .NET API Reference*.

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete multiple objects in a non-versioned S3 bucket.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
```

```

    /// <summary>
    /// This example shows how to delete multiple objects from an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class DeleteMultipleObjects
    {
        /// <summary>
        /// The Main method initializes the Amazon S3 client and the name of
        /// the bucket and then passes those values to MultiObjectDeleteAsync.
        /// </summary>
        public static async Task Main()
        {
            const string bucketName = "amzn-s3-demo-bucket";

            // If the Amazon S3 bucket from which you wish to delete objects is
            not
            // located in the same AWS Region as the default user, define the
            // AWS Region for the Amazon S3 bucket as a parameter to the client
            // constructor.
            IAmazonS3 s3Client = new AmazonS3Client();

            await MultiObjectDeleteAsync(s3Client, bucketName);
        }

        /// <summary>
        /// This method uses the passed Amazon S3 client to first create and then
        /// delete three files from the named bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        call
        /// Amazon S3 methods.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where
        objects
        /// will be created and then deleted.</param>
        public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
        bucketName)
        {
            // Create three sample objects which we will then delete.
            var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);

            // Now perform the multi-object delete, passing the key names and
            // version IDs. Since we are working with a non-versioned bucket,
            // the object keys collection includes null version IDs.

```

```

        DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest
    {
        BucketName = bucketName,
        Objects = keysAndVersions,
    };

    // You can add a specific object key to the delete request using the
    // AddKey method of the multiObjectDeleteRequest.
    try
    {
        DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
        Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
    }
    catch (DeleteObjectsException e)
    {
        PrintDeletionErrorStatus(e);
    }
}

/// <summary>
/// Prints the list of errors raised by the call to DeleteObjectsAsync.
/// </summary>
/// <param name="ex">A collection of exceptions returned by the call to
/// DeleteObjectsAsync.</param>
public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
{
    DeleteObjectsResponse errorResponse = ex.Response;
    Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

    Console.WriteLine($"Successfully deleted
{errorResponse.DeletedObjects.Count}.");
    Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");

    Console.WriteLine("Printing error data...");
    foreach (DeleteError deleteError in errorResponse.DeleteErrors)
    {
        Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
    }
}

```

```
/// <summary>
/// This method creates simple text file objects that can be used in
/// the delete method.
/// </summary>
/// <param name="client">The Amazon S3 client used to call
PutObjectAsync.</param>
/// <param name="number">The number of objects to create.</param>
/// <param name="bucketName">The name of the bucket where the objects
/// will be created.</param>
/// <returns>A list of keys (object keys) and versions that the calling
/// method will use to delete the newly created files.</returns>
public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, int number, string bucketName)
{
    List<KeyVersion> keys = new List<KeyVersion>();
    for (int i = 0; i < number; i++)
    {
        string key = "ExampleObject-" + new System.Random().Next();
        PutObjectRequest request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = key,
            ContentBody = "This is the content body!",
        };

        PutObjectResponse response = await
client.PutObjectAsync(request);

        // For non-versioned bucket operations, we only need the
        // object key.
        KeyVersion keyVersion = new KeyVersion
        {
            Key = key,
        };
        keys.Add(keyVersion);
    }

    return keys;
}
}
```

Delete multiple objects in a versioned S3 bucket.

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete objects in a version-enabled Amazon
/// Simple StorageService (Amazon S3) bucket.
/// </summary>
public class DeleteMultipleObjects
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region for your Amazon S3 bucket is different from
        // the AWS Region of the default user, define the AWS Region for
        // the Amazon S3 bucket and pass it to the client constructor
        // like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USSouth1;
        IAmazonS3 s3Client;

        s3Client = new AmazonS3Client();
        await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
    }

    /// <summary>
    /// This method removes multiple versions and objects from a
    /// version-enabled Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
    {

```

```

        // Delete objects (specifying object version in the request).
        await DeleteObjectVersionsAsync(client, bucketName);

        // Delete objects (without specifying object version in the request).
        var deletedObjects = await DeleteObjectsAsync(client, bucketName);

        // Additional exercise - remove the delete markers Amazon S3 returned
from
        // the preceding response. This results in the objects reappearing
        // in the bucket (you can verify the appearance/disappearance of
        // objects in the console).
        await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
    }

    /// <summary>
    /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
    /// them again. The method returns a list of the Amazon S3 objects
deleted.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// PutObjectsAsync and NonVersionedDeleteAsync.</param>
    /// <param name="bucketName">The name of the bucket where the objects
    /// will be created and then deleted.</param>
    /// <returns>A list of DeletedObjects.</returns>
    public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
    {
        // Upload the sample objects.
        var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

        // Delete objects using only keys. Amazon S3 creates a delete marker
and
        // returns its version ID in the response.
        List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
        return deletedObjects;
    }

    /// <summary>
    /// This method creates several temporary objects and then deletes them.
    /// </summary>
    /// <param name="client">The S3 client.</param>

```



```

    /// <param name="bucketName">Name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
    {
        // Upload the sample objects.
        var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

        // Delete the specific object versions.
        await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
    }

    /// <summary>
    /// Displays the list of information about deleted files to the console.
    /// </summary>
    /// <param name="e">Error information from the delete process.</param>
    private static void DisplayDeletionErrors(DeleteObjectsException e)
    {
        var errorResponse = e.Response;
        Console.WriteLine($"No. of objects successfully deleted =
{errorResponse.DeletedObjects.Count}");
        Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");
        Console.WriteLine("Printing error data...");
        foreach (var deleteError in errorResponse.DeleteErrors)
        {
            Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
        }
    }

    /// <summary>
    /// Delete multiple objects from a version-enabled bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    /// <param name="keys">A list of key names for the objects to delete.</
param>
    private static async Task VersionedDeleteAsync(IAmazonS3 client, string
bucketName, List<KeyVersion> keys)

```

```

        {
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keys, // This includes the object keys and specific
version IDs.
            };

            try
            {
                Console.WriteLine("Executing VersionedDelete...");
                DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted all the
{response.DeletedObjects.Count} items");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Deletes multiple objects from a non-versioned Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        /// <returns>A list of the deleted objects.</returns>
        private static async Task<List<DeletedObject>>
NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
keys)
        {
            // Create a request that includes only the object key names.
            DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest();
            multiObjectDeleteRequest.BucketName = bucketName;

            foreach (var key in keys)

```

```

        {
            multiObjectDeleteRequest.AddKey(key.Key);
        }

        // Execute DeleteObjectsAsync.
        // The DeleteObjectsAsync method adds a delete marker for each
        // object deleted. You can verify that the objects were removed
        // using the Amazon S3 console.
        DeleteObjectsResponse response;
        try
        {
            Console.WriteLine("Executing NonVersionedDelete...");
            response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
            throw; // Some deletions failed. Investigate before continuing.
        }

        // This response contains the DeletedObjects list which we use to
delete the delete markers.
        return response.DeletedObjects;
    }

    /// <summary>
    /// Deletes the markers left after deleting the temporary objects.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    /// <param name="deletedObjects">A list of the objects that were
deleted.</param>
    private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
    {
        var keyVersionList = new List<KeyVersion>();

```

```

        foreach (var deletedObject in deletedObjects)
        {
            KeyVersion keyVersion = new KeyVersion
            {
                Key = deletedObject.Key,
                VersionId = deletedObject.DeleteMarkerVersionId,
            };
            keyVersionList.Add(keyVersion);
        }

        // Create another request to delete the delete markers.
        var multiObjectDeleteRequest = new DeleteObjectsRequest
        {
            BucketName = bucketName,
            Objects = keyVersionList,
        };

        // Now, delete the delete marker to bring your objects back to the
        bucket.
        try
        {
            Console.WriteLine("Removing the delete markers .....");
            var deleteObjectResponse = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine($"Successfully deleted the
{deleteObjectResponse.DeletedObjects.Count} delete markers");
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
        }
    }

    /// <summary>
    /// Create temporary Amazon S3 objects to show how object deletion works
in an
    /// Amazon S3 bucket with versioning enabled.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// PutObjectAsync to create temporary objects for the example.</param>
    /// <param name="bucketName">A string representing the name of the S3
    /// bucket where we will create the temporary objects.</param>

```

```
    /// <param name="number">The number of temporary objects to create.</  
param>  
    /// <returns>A list of the KeyVersion objects.</returns>  
    private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3  
client, string bucketName, int number)  
    {  
        var keys = new List<KeyVersion>();  
  
        for (var i = 0; i < number; i++)  
        {  
            string key = "ObjectToDelete-" + new System.Random().Next();  
            PutObjectRequest request = new PutObjectRequest  
            {  
                BucketName = bucketName,  
                Key = key,  
                ContentBody = "This is the content body!",  
            };  
  
            var response = await client.PutObjectAsync(request);  
            KeyVersion keyVersion = new KeyVersion  
            {  
                Key = key,  
                VersionId = response.VersionId,  
            };  
  
            keys.Add(keyVersion);  
        }  
  
        return keys;  
    }  
}
```

- For API details, see [DeleteObjects](#) in *Amazon SDK for .NET API Reference*.

Bash

Amazon CLI with Bash script

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - A list of keys in the bucket to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

    # Create the JSON for the items to delete.
    local delete_items
    delete_items="{\"Objects\":["
    for key in $keys; do
        delete_items="$delete_items{\"Key\": \"$key\"},"
    done
    delete_items="$delete_items\"]}"
}
```

```
done
delete_items=${delete_items%?} # Remove the final comma.
delete_items="$delete_items]}"

response=$(aws s3api delete-objects \
  --bucket "$bucket_name" \
  --delete "$delete_items")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
  errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
  return 1
fi
}
```

- For API details, see [DeleteObjects](#) in *Amazon CLI Command Reference*.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration
                               &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {
        deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }
}
```

```

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
        client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }

        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [DeleteObjects](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command deletes an object from a bucket named `amzn-s3-demo-bucket`:

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete file://delete.json
```

`delete.json` is a JSON document in the current directory that specifies the object to delete:

```
{
  "Objects": [
```



```
{
  "Key": "test1.txt"
},
"Quiet": false
}
```

Output:

```
{
  "Deleted": [
    {
      "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUP01koHwzU",
      "Key": "test1.txt",
      "DeleteMarker": true
    }
  ]
}
```

- For API details, see [DeleteObjects](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
```

```

    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }

    input := s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),
        Delete: &types.Delete{
            Objects: objects,
            Quiet:   aws.Bool(true),
        },
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
    if err != nil || len(delOut.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucket)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
        }
    } else if len(delOut.Errors) > 0 {
        for _, outErr := range delOut.Errors {
            log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
        }
    }
}

```

```

    err = fmt.Errorf("%s", *del0ut.Errors[0].Message)
}
} else {
    for _, del0bjs := range del0ut.Deleted {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: del0bjs.Key},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                *del0bjs.Key)
        } else {
            log.Printf("Deleted %s.\n", *del0bjs.Key)
        }
    }
}
return err
}

```

- For API details, see [DeleteObjects](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

```

```
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucketName>

            Where:
                bucketName - the Amazon S3 bucket name.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes multiple objects from an Amazon S3 bucket.
     *
     * @param s3 An Amazon S3 client object.
     * @param bucketName The name of the Amazon S3 bucket to delete objects from.
     */
    public static void deleteBucketObjects(S3Client s3, String bucketName) {
```

```
// Upload three sample objects to the specified Amazon S3 bucket.
ArrayList<ObjectIdentifier> keys = new ArrayList<>();
PutObjectRequest putOb;
ObjectIdentifier objectId;

for (int i = 0; i < 3; i++) {
    String keyName = "delete object example " + i;
    objectId = ObjectIdentifier.builder()
        .key(keyName)
        .build();

    putOb = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(keyName)
        .build();

    s3.putObject(putOb, RequestBody.fromString(keyName));
    keys.add(objectId);
}

System.out.println(keys.size() + " objects successfully created.");

// Delete multiple objects in one request.
Delete del = Delete.builder()
    .objects(keys)
    .build();

try {
    DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
    .bucket(bucketName)
    .delete(del)
    .build();

    s3.deleteObjects(multiObjectDeleteRequest);
    System.out.println("Multiple objects are deleted!");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteObjects](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete multiple objects.

```
import {
  DeleteObjectsCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete multiple objects from an S3 bucket.
 * @param {{ bucketName: string, keys: string[] }}
 */
export const main = async ({ bucketName, keys }) => {
  const client = new S3Client({});

  try {
    const { Deleted } = await client.send(
      new DeleteObjectsCommand({
        Bucket: bucketName,
        Delete: {
          Objects: keys.map((k) => ({ Key: k })),
        },
      }),
    );
    for (const key in keys) {
      await waitUntilObjectNotExists(
        { client },
```

```
        { Bucket: bucketName, Key: key },
      );
    }
    console.log(
      `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting objects from ${bucketName}. The bucket
doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting objects from ${bucketName}.
${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- For API details, see [DeleteObjects](#) in *Amazon SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
suspend fun deleteBucketObjects(
```

```
        bucketName: String,
        objectName: String,
    ) {
        val objectId =
            ObjectIdentifier {
                key = objectName
            }

        val delOb =
            Delete {
                objects = listOf(objectId)
            }

        val request =
            DeleteObjectsRequest {
                bucket = bucketName
                delete = delOb
            }

        S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
            s3.deleteObjects(request)
            println("$objectName was deleted from $bucketName")
        }
    }
}
```

- For API details, see [DeleteObjects](#) in *Amazon SDK for Kotlin API reference*.

PHP

SDK for PHP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete a set of objects from a list of keys.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
```



```

try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $this->s3client->deleteObjects([
        'Bucket' => $this->bucketName,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    if (isset($check['Contents']) && count($check['Contents']) > 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with object deletion before continuing.");
}

```

- For API details, see [DeleteObjects](#) in *Amazon SDK for PHP API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command removes the object "sample.txt" from bucket "test-files". You are prompted for confirmation before the command executes; to suppress the prompt use the **-Force** switch.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Example 2: This command removes the specified version of object "sample.txt" from bucket "test-files", assuming the bucket has been configured to enable object versions.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId  
HLbxx6V9omT6AQYVpks8mmFKQcejpqt
```

Example 3: This command removes objects "sample1.txt", "sample2.txt" and "sample3.txt" from bucket "test-files" as a single batch operation. The service response will list all keys processed, regardless of the success or error status of the deletion. To obtain only errors for keys that were not able to be processed by the service add the -ReportErrorsOnly parameter (this parameter can also be specified with the alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",  
"sample2.txt", "sample3.txt" )
```

Example 4: This example uses an inline expression with the -KeyCollection parameter to obtain the keys of the objects to delete. Get-S3Object returns a collection of Amazon.S3.Model.S3Object instances, each of which has a Key member of type string identifying the object.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Example 5: This example obtains all objects that have a key prefix "prefix/subprefix" in the bucket and deletes them. Note that the incoming objects are processed one at a time. For large collections consider passing the collection to the cmdlet's -InputObject (alias -S3ObjectCollection) parameter to enable the deletion to occur as a batch with a single call to the service.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

Example 6: This example pipes a collection of Amazon.S3.Model.S3ObjectVersion instances that represent delete markers to the cmdlet for deletion. Note that the incoming objects are processed one at a time. For large collections consider passing the collection to the cmdlet's -InputObject (alias -S3ObjectCollection) parameter to enable the deletion to occur as a batch with a single call to the service.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Example 7: This script shows how to perform a batch delete of a set of objects (in this case delete markers) by constructing an array of objects to be used with the `-KeyAndVersionCollection` parameter.

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
{$_ .IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
$marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- For API details, see [DeleteObjects](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command removes the object "sample.txt" from bucket "test-files". You are prompted for confirmation before the command executes; to suppress the prompt use the `-Force` switch.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Example 2: This command removes the specified version of object "sample.txt" from bucket "test-files", assuming the bucket has been configured to enable object versions.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Example 3: This command removes objects "sample1.txt", "sample2.txt" and "sample3.txt" from bucket "test-files" as a single batch operation. The service response will list all keys processed, regardless of the success or error status of the deletion. To obtain only errors for keys that were not able to be processed by the service add the `-ReportErrorsOnly` parameter (this parameter can also be specified with the alias `-Quiet`).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
"sample2.txt", "sample3.txt" )
```

Example 4: This example uses an inline expression with the `-KeyCollection` parameter to obtain the keys of the objects to delete. `Get-S3Object` returns a collection of

Amazon.S3.Model.S3Object instances, each of which has a **Key** member of type string identifying the object.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Example 5: This example obtains all objects that have a key prefix "prefix/subprefix" in the bucket and deletes them. Note that the incoming objects are processed one at a time. For large collections consider passing the collection to the cmdlet's **-InputObject** (alias **-S3ObjectCollection**) parameter to enable the deletion to occur as a batch with a single call to the service.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
Remove-S3Object -Force
```

Example 6: This example pipes a collection of **Amazon.S3.Model.S3ObjectVersion** instances that represent delete markers to the cmdlet for deletion. Note that the incoming objects are processed one at a time. For large collections consider passing the collection to the cmdlet's **-InputObject** (alias **-S3ObjectCollection**) parameter to enable the deletion to occur as a batch with a single call to the service.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Example 7: This script shows how to perform a batch delete of a set of objects (in this case delete markers) by constructing an array of objects to be used with the **-KeyAndVersionCollection** parameter.

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
{$_ .IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
$marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- For API details, see [DeleteObjects](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete a set of objects by using a list of object keys.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.

        :param bucket: The bucket that contains the objects. This is a Boto3
        Bucket
                           resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
        and any that could not be deleted.
        """
        try:
            response = bucket.delete_objects(
                Delete={"Objects": [{"Key": key} for key in object_keys]}
            )
            if "Deleted" in response:
```

```

        logger.info(
            "Deleted objects '%s' from bucket '%s'.",
            [del_obj["Key"] for del_obj in response["Deleted"]],
            bucket.name,
        )
    if "Errors" in response:
        logger.warning(
            "Could not delete objects '%s' from bucket '%s'.",
            [
                f"{del_obj['Key']}: {del_obj['Code']}"
                for del_obj in response["Errors"]
            ],
            bucket.name,
        )
    except ClientError:
        logger.exception("Couldn't delete any objects from bucket %s.",
            bucket.name)
        raise
    else:
        return response

```

Delete all objects in a bucket.

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.

```

```
:param bucket: The bucket to empty. This is a Boto3 Bucket resource.
"""
try:
    bucket.objects.delete()
    logger.info("Emptied bucket '%s'.", bucket.name)
except ClientError:
    logger.exception("Couldn't empty bucket '%s'.", bucket.name)
    raise
```

Permanently delete a versioned object by deleting all of its versions.

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

- For API details, see [DeleteObjects](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- For API details, see [DeleteObjects](#) in *Amazon SDK for Ruby API Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).


```

/// Delete the objects in a bucket.
pub async fn delete_objects(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    objects_to_delete: Vec<String>,
) -> Result<(), S3ExampleError> {
    // Push into a mut vector to use `?` early return errors while building
    object keys.
    let mut delete_object_ids: Vec<aws_sdk_s3::types::ObjectIdentifier> = vec![];
    for obj in objects_to_delete {
        let obj_id = aws_sdk_s3::types::ObjectIdentifier::builder()
            .key(obj)
            .build()
            .map_err(|err| {
                S3ExampleError::new(format!("Failed to build key for
delete_object: {err:?}"))
            })?;
        delete_object_ids.push(obj_id);
    }

    client
        .delete_objects()
        .bucket(bucket_name)
        .delete(
            aws_sdk_s3::types::Delete::builder()
                .set_objects(Some(delete_object_ids))
                .build()
                .map_err(|err| {
                    S3ExampleError::new(format!("Failed to build delete_object
input {err:?}"))
                })?,
        )
        .send()
        .await?;
    Ok(())
}

```

- For API details, see [DeleteObjects](#) in *Amazon SDK for Rust API reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
    oo_result = lo_s3->deleteobjects(          " oo_result is returned for  
testing purposes. "  
    iv_bucket = iv_bucket_name  
    io_delete = NEW /aws1/cl_s3_delete( it_objects = it_object_keys ) ).  
    MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteObjects](#) in *Amazon SDK for SAP ABAP API reference*.

Swift

SDK for Swift

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import AWSS3  
  
public func deleteObjects(bucket: String, keys: [String]) async throws {  
    let input = DeleteObjectsInput(  
        bucket: bucket,  
        delete: S3ClientTypes.Delete(  
            objects: keys.map { S3ClientTypes.ObjectIdentifier(key: $0) },
```

```
        quiet: true
    )
)

do {
    _ = try await client.deleteObjects(input: input)
} catch {
    print("ERROR: deleteObjects:", dump(error))
    throw error
}
```

- For API details, see [DeleteObjects](#) in *Amazon SDK for Swift API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeletePublicAccessBlock with a CLI

The following code examples show how to use DeletePublicAccessBlock.

CLI

Amazon CLI

To delete the block public access configuration for a bucket

The following delete-public-access-block example removes the block public access configuration on the specified bucket.

```
aws s3api delete-public-access-block \
    --bucket amzn-s3-demo-bucket
```

This command produces no output.

- For API details, see [DeletePublicAccessBlock](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command turns off the block public access setting for the given bucket.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

Output:

```
amzn-s3-demo-bucket
```

- For API details, see [DeletePublicAccessBlock](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command turns off the block public access setting for the given bucket.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

Output:

```
amzn-s3-demo-bucket
```

- For API details, see [DeletePublicAccessBlock](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketAccelerateConfiguration with a CLI

The following code examples show how to use GetBucketAccelerateConfiguration.

CLI

Amazon CLI

To retrieve the accelerate configuration of a bucket

The following `get-bucket-accelerate-configuration` example retrieves the accelerate configuration for the specified bucket.

```
aws s3api get-bucket-accelerate-configuration \
  --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "Status": "Enabled"
}
```

- For API details, see [GetBucketAccelerateConfiguration](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns the value `Enabled`, if the transfer acceleration settings is enabled for the bucket specified.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Value
-----
Enabled
```

- For API details, see [GetBucketAccelerateConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns the value `Enabled`, if the transfer acceleration settings is enabled for the bucket specified.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Value
-----
Enabled
```

- For API details, see [GetBucketAccelerateConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketAc1 with an Amazon SDK or CLI

The following code examples show how to use GetBucketAc1.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Check if a bucket exists](#)
- [Manage access control lists \(ACLs\)](#)

.NET**Amazon SDK for .NET****Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Get the access control list (ACL) for the new bucket.
/// </summary>
```

```

    /// <param name="client">The initialized client object used to get the
    /// access control list (ACL) of the bucket.</param>
    /// <param name="newBucketName">The name of the newly created bucket.</
param>
    /// <returns>An S3AccessControlList.</returns>
    public static async Task<S3AccessControlList>
    GetACLForBucketAsync(IAmazonS3 client, string newBucketName)
    {
        // Retrieve bucket ACL to show that the ACL was properly applied to
        // the new bucket.
        GetACLResponse getACLResponse = await client.GetACLAsync(new
    GetACLRequest
    {
        BucketName = newBucketName,
    });

        return getACLResponse.AccessControlList;
    }

```

- For API details, see [GetBucketAcl](#) in *Amazon SDK for .NET API Reference*.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =

```

```

        s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                    << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                        << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                            << getGranteeTypeString(grantee.GetType()) <<
std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                            << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                            << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:           "
                            << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:          "
                            << grantee.GetURI() << std::endl;
            }
        }
    }
}

```



```

        std::cout << "Permission:    " <<
            getPermissionString(grant.GetPermission()) <<
            std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
string.
/*!
\param type: Type enumeration.
\return String: Human-readable string.
*/

Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
string.
/*!
\param permission: Permission enumeration.
\return String: Human-readable string.
*/

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                "objects in this bucket, and read/write this "
                "bucket's permissions";
    }
}

```

```
    case Aws::S3::Model::Permission::NOT_SET:
        return "Permission not set";
    case Aws::S3::Model::Permission::READ:
        return "Can list objects in this bucket";
    case Aws::S3::Model::Permission::READ_ACP:
        return "Can read this bucket's permissions";
    case Aws::S3::Model::Permission::WRITE:
        return "Can create, overwrite, and delete objects in this bucket";
    case Aws::S3::Model::Permission::WRITE_ACP:
        return "Can write this bucket's permissions";
    default:
        return "Permission unknown";
}

return "Permission unknown";
}
```

- For API details, see [GetBucketAcl](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command retrieves the access control list for a bucket named `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
```

```
        "ID":  
        "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"  
    },  
    "Permission": "FULL_CONTROL"  
}  
]  
}
```

- For API details, see [GetBucketAcl](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;  
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;  
import software.amazon.awssdk.services.s3.model.Grant;  
  
import java.util.List;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 * <p>  
 * For more information, see the following documentation topic:  
 * <p>  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
  
public class GetAcl {  
    public static void main(String[] args) {
```

```

        final String usage = ""

        Usage:
            <bucketName> <objectKey>

        Where:
            bucketName - The Amazon S3 bucket to get the access control list
(ACL) for.
            objectKey - The object to get the ACL for.\s
        """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    /**
     * Retrieves the Access Control List (ACL) for an object in an Amazon S3
    bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param objectKey The key of the object for which the ACL is to be
    retrieved.
     * @param bucketName The name of the bucket containing the object.
     * @return The ID of the grantee who has permission on the object, or an
    empty string if an error occurs.
     */
    public static String getBucketACL(S3Client s3, String objectKey, String
    bucketName) {
        try {

```

```
GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .build();

GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
List<Grant> grants = aclRes.grants();
String grantee = "";
for (Grant grant : grants) {
    System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
    grantee = grant.grantee().id();
}

return grantee;
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
}
```

- For API details, see [GetBucketAcl](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get the ACL permissions.

```
import {
    GetBucketAclCommand,
    S3Client,
```

```
S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Retrieves the Access Control List (ACL) for an S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketAclCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`ACL for bucket "${bucketName}":`);
    console.log(JSON.stringify(response, null, 2));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting ACL for ${bucketName}. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting ACL for ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).
- For API details, see [GetBucketAcl](#) in *Amazon SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V5

Example 1: The command gets the details of the object owner of the S3 object.

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

Output:

```
DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- For API details, see [GetBucketAcl](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```

def get_acl(self):
    """
    Get the ACL of the bucket.

    :return: The ACL of the bucket.
    """
    try:
        acl = self.bucket.Acl()
        logger.info(
            "Got ACL for bucket %s. Owner is %s.", self.bucket.name,
            acl.owner
        )
    except ClientError:
        logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
        raise
    else:
        return acl

```

- For API details, see [GetBucketAcl](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

TRY.
    oo_result = lo_s3->getbucketacl(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved bucket ACL.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```


- For API details, see [GetBucketAcl](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketAnalyticsConfiguration with a CLI

The following code examples show how to use GetBucketAnalyticsConfiguration.

CLI

Amazon CLI

To retrieve the analytics configuration for a bucket with a specific ID

The following get-bucket-analytics-configuration example displays the analytics configuration for the specified bucket and ID.

```
aws s3api get-bucket-analytics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 1
```

Output:

```
{  
  "AnalyticsConfiguration": {  
    "StorageClassAnalysis": {},  
    "Id": "1"  
  }  
}
```

- For API details, see [GetBucketAnalyticsConfiguration](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns the details of the analytics filter with the name 'testfilter' in the given S3 bucket.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId  
'testfilter'
```

- For API details, see [GetBucketAnalyticsConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns the details of the analytics filter with the name 'testfilter' in the given S3 bucket.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId  
'testfilter'
```

- For API details, see [GetBucketAnalyticsConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketCors with an Amazon SDK or CLI

The following code examples show how to use GetBucketCors.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>  
/// Retrieve the CORS configuration applied to the Amazon S3 bucket.  
/// </summary>  
/// <param name="client">The initialized Amazon S3 client object used  
/// to retrieve the CORS configuration.</param>
```

```
/// <returns>The created CORS configuration object.</returns>
private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
{
    GetCORSConfigurationRequest request = new
GetCORSConfigurationRequest()
    {
        BucketName = BucketName,
    };
    var response = await client.GetCORSConfigurationAsync(request);
    var configuration = response.Configuration;
    PrintCORSRules(configuration);
    return configuration;
}
```

- For API details, see [GetBucketCors](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

The following command retrieves the Cross-Origin Resource Sharing configuration for a bucket named `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-cors --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "CORSRules": [
    {
      "AllowedHeaders": [
        "*"
      ],
      "ExposeHeaders": [
        "x-amz-server-side-encryption"
      ],
      "AllowedMethods": [
        "PUT",
        "POST",
```

```
        "DELETE"
      ],
      "MaxAgeSeconds": 3000,
      "AllowedOrigins": [
        "http://www.example.com"
      ]
    },
    {
      "AllowedHeaders": [
        "Authorization"
      ],
      "MaxAgeSeconds": 3000,
      "AllowedMethods": [
        "GET"
      ],
      "AllowedOrigins": [
        "*"
      ]
    }
  ]
}
```

- For API details, see [GetBucketCors](#) in *Amazon CLI Command Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get the CORS policy for the bucket.

```
import {
  GetBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
/**
 * Log the Cross-Origin Resource Sharing (CORS) configuration information
 * set for the bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new GetBucketCorsCommand({
    Bucket: bucketName,
  });

  try {
    const { CORSRules } = await client.send(command);
    console.log(JSON.stringify(CORSRules));
    CORSRules.forEach((cr, i) => {
      console.log(
        `\\nCORSRule ${i + 1}`,
        `\\n${"-"}.repeat(10)`,
        `\\nAllowedHeaders: ${cr.AllowedHeaders}`,
        `\\nAllowedMethods: ${cr.AllowedMethods}`,
        `\\nAllowedOrigins: ${cr.AllowedOrigins}`,
        `\\nExposeHeaders: ${cr.ExposeHeaders}`,
        `\\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`,
      );
    });
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).
- For API details, see [GetBucketCors](#) in *Amazon SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_cors(self):
        """
        Get the CORS rules for the bucket.

        :return The CORS rules for the specified bucket.
        """
        try:
            cors = self.bucket.Cors()
            logger.info(
                "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
                self.bucket.name
            )
        except ClientError:
```

```
        logger.exception("Couldn't get CORS for bucket %s.",
self.bucket.name))
        raise
    else:
        return cors
```

- For API details, see [GetBucketCors](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
  configuration for the bucket.
  def cors
    @bucket_cors.data
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
  why: #{e.message}"
      nil
  end
end
```

```
end

end
```

- For API details, see [GetBucketCors](#) in *Amazon SDK for Ruby API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.
    oo_result = lo_s3->getbucketcors(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved bucket CORS configuration.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see [GetBucketCors](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketEncryption with an Amazon SDK or CLI

The following code examples show how to use GetBucketEncryption.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Get and print the encryption settings of a bucket.
/// </summary>
/// <param name="bucketName">Name of the bucket.</param>
/// <returns>Async task.</returns>
public static async Task GetEncryptionSettings(string bucketName)
{
    // Check and print the bucket encryption settings.
    Console.WriteLine($"Getting encryption settings for bucket
{bucketName}.");

    try
    {
        var settings =
            await _s3Client.GetBucketEncryptionAsync(
                new GetBucketEncryptionRequest() { BucketName =
bucketName });

        foreach (var encryptionSettings in
settings?.ServerSideEncryptionConfiguration?.ServerSideEncryptionRules!)
        {
            Console.WriteLine(
                $"Algorithm:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionAlgorithm}");
            Console.WriteLine(
                $"Key:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionKeyManagementService
}
        }
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine(ex.ErrorCode == "InvalidBucketName"
```

```
        ? $"Bucket {bucketName} was not found."
        : $"Unable to get bucket encryption for bucket {bucketName},
{ex.Message}");
    }
}
```

- For API details, see [GetBucketEncryption](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To retrieve the server-side encryption configuration for a bucket

The following `get-bucket-encryption` example retrieves the server-side encryption configuration for the bucket `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-encryption \
  --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
```

- For API details, see [GetBucketEncryption](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns all the server side encryption rules associated with the given bucket.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetBucketEncryption](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns all the server side encryption rules associated with the given bucket.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetBucketEncryption](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketInventoryConfiguration with a CLI

The following code examples show how to use GetBucketInventoryConfiguration.

CLI

Amazon CLI

To retrieve the inventory configuration for a bucket

The following get-bucket-inventory-configuration example retrieves the inventory configuration for the specified bucket with ID 1.

```
aws s3api get-bucket-inventory-configuration \
```

```
--bucket amzn-s3-demo-bucket \  
--id 1
```

Output:

```
{  
  "InventoryConfiguration": {  
    "IsEnabled": true,  
    "Destination": {  
      "S3BucketDestination": {  
        "Format": "ORC",  
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",  
        "AccountId": "123456789012"  
      }  
    },  
    "IncludedObjectVersions": "Current",  
    "Id": "1",  
    "Schedule": {  
      "Frequency": "Weekly"  
    }  
  }  
}
```

- For API details, see [GetBucketInventoryConfiguration](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns the details of the inventory named 'testinventory' for the given S3 bucket.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId  
'testinventory'
```

- For API details, see [GetBucketInventoryConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns the details of the inventory named 'testinventory' for the given S3 bucket.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId  
'testinventory'
```

- For API details, see [GetBucketInventoryConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketLifecycleConfiguration with an Amazon SDK or CLI

The following code examples show how to use GetBucketLifecycleConfiguration.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>  
/// Returns a configuration object for the supplied bucket name.  
/// </summary>  
/// <param name="client">The S3 client object used to call  
/// the GetLifecycleConfigurationAsync method.</param>  
/// <param name="bucketName">The name of the S3 bucket for which a  
/// configuration will be created.</param>  
/// <returns>Returns a new LifecycleConfiguration object.</returns>  
public static async Task<LifecycleConfiguration>  
RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)  
{  
    var request = new GetLifecycleConfigurationRequest()  
    {  
        BucketName = bucketName,  
    };  
    var response = await client.GetLifecycleConfigurationAsync(request);
```

```
        var configuration = response.Configuration;
        return configuration;
    }
```

- For API details, see [GetBucketLifecycleConfiguration](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

The following command retrieves the lifecycle configuration for a bucket named `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "Rules": [
    {
      "ID": "Move rotated logs to Glacier",
      "Prefix": "rotated/",
      "Status": "Enabled",
      "Transitions": [
        {
          "Date": "2015-11-10T00:00:00.000Z",
          "StorageClass": "GLACIER"
        }
      ]
    },
    {
      "Status": "Enabled",
      "Prefix": "",
      "NoncurrentVersionTransitions": [
        {
          "NoncurrentDays": 0,
          "StorageClass": "GLACIER"
        }
      ]
    }
  ],
}
```

```

        "ID": "Move old versions to Glacier"
    }
]
}

```

- For API details, see [GetBucketLifecycleConfiguration](#) in *Amazon CLI Command Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()
            logger.info(
                "Got lifecycle rules %s for bucket '%s'.",
                config.rules,
                self.bucket.name,
            )

```

```
    )
except:
    logger.exception(
        "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
    )
    raise
else:
    return config.rules
```

- For API details, see [GetBucketLifecycleConfiguration](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.
    oo_result = lo_s3->getbucketlifecycleconf(          " oo_result is
returned for testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved bucket lifecycle configuration.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see [GetBucketLifecycleConfiguration](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketLocation with an Amazon SDK or CLI

The following code examples show how to use GetBucketLocation.

CLI

Amazon CLI

The following command retrieves the location constraint for a bucket named `amzn-s3-demo-bucket`, if a constraint exists:

```
aws s3api get-bucket-location --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "LocationConstraint": "us-west-2"
}
```

- For API details, see [GetBucketLocation](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns the location constraint for the bucket 'amzn-s3-demo-bucket', if a constraint exists.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Value
-----
ap-south-1
```

- For API details, see [GetBucketLocation](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns the location constraint for the bucket 'amzn-s3-demo-bucket', if a constraint exists.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Value
-----
ap-south-1
```

- For API details, see [GetBucketLocation](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
        }
    }
}
```

```

        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint() == Some(&region) {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}

```

- For API details, see [GetBucketLocation](#) in *Amazon SDK for Rust API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketLogging with a CLI

The following code examples show how to use GetBucketLogging.

CLI

Amazon CLI

To retrieve the logging status for a bucket

The following `get-bucket-logging` example retrieves the logging status for the specified bucket.

```
aws s3api get-bucket-logging \  
  --bucket amzn-s3-demo-bucket
```

Output:

```
{  
  "LoggingEnabled": {  
    "TargetPrefix": "",  
    "TargetBucket": "amzn-s3-demo-bucket-logs"  
  }  
}
```

- For API details, see [GetBucketLogging](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns the logging status for the specified bucket.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Output:

TargetBucketName	Grants	TargetPrefix
-----	-----	-----
testbucket1	{}	testprefix

- For API details, see [GetBucketLogging](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns the logging status for the specified bucket.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Output:

TargetBucketName	Grants	TargetPrefix
testbucket1	{}	testprefix

- For API details, see [GetBucketLogging](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketMetricsConfiguration with a CLI

The following code examples show how to use GetBucketMetricsConfiguration.

CLI

Amazon CLI

To retrieve the metrics configuration for a bucket with a specific ID

The following get-bucket-metrics-configuration example displays the metrics configuration for the specified bucket and ID.

```
aws s3api get-bucket-metrics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 123
```

Output:

```
{  
  "MetricsConfiguration": {
```

```
    "Filter": {
      "Prefix": "logs"
    },
    "Id": "123"
  }
}
```

- For API details, see [GetBucketMetricsConfiguration](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns the details about the metrics filter named 'testfilter' for the given S3 bucket.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- For API details, see [GetBucketMetricsConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns the details about the metrics filter named 'testfilter' for the given S3 bucket.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- For API details, see [GetBucketMetricsConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketNotification with a CLI

The following code examples show how to use GetBucketNotification.

CLI

Amazon CLI

The following command retrieves the notification configuration for a bucket named `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-notification --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "TopicConfiguration": {
    "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
    "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNl",
    "Event": "s3:ObjectCreated:*",
    "Events": [
      "s3:ObjectCreated:*"
    ]
  }
}
```

- For API details, see [GetBucketNotification](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example retrieves notification configuration of the given bucket

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
TopicConfigurations
```

Output:

Id	Topic
--	-----
mimo	arn:aws:sns:eu-west-1:123456789012:topic-1

- For API details, see [GetBucketNotification](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example retrieves notification configuration of the given bucket

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty  
TopicConfigurations
```

Output:

```
Id    Topic  
--    -  
mimo  arn:aws:sns:eu-west-1:123456789012:topic-1
```

- For API details, see [GetBucketNotification](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketPolicy with an Amazon SDK or CLI

The following code examples show how to use GetBucketPolicy.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,  
                                const Aws::S3::S3ClientConfiguration  
    &clientConfig) {  
    Aws::S3::S3Client s3Client(clientConfig);  
  
    Aws::S3::Model::GetBucketPolicyRequest request;  
    request.SetBucket(bucketName);
```



```

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
        s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
            << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n"
<<
            policy_stream.str() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [GetBucketPolicy](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command retrieves the bucket policy for a bucket named `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket
```

Output:

```

{
    "Policy": "{\n\"Version\": \"2008-10-17\", \"Statement\": [\n{\n\"Sid\": \"\", \"Effect\
\n\": \"Allow\", \"Principal\": \"*\", \"Action\": \"s3:GetObject\", \"Resource\":
\n\"arn:aws:s3:::amzn-s3-demo-bucket/*\"}, {\n\"Sid\": \"\", \"Effect\": \"Deny\",

```

```
\\"Principal\\":\\"*\\",\\"Action\\":\\"s3:GetObject\\",\\"Resource\\":\\"arn:aws:s3:::amzn-s3-demo-bucket/secret/*\\"}]}"
}
```

Get and put a bucket policyThe following example shows how you can download an Amazon S3 bucket policy, make modifications to the file, and then use `put-bucket-policy` to apply the modified bucket policy. To download the bucket policy to a file, you can run:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket --query Policy --output text > policy.json
```

You can then modify the `policy.json` file as needed. Finally you can apply this modified policy back to the S3 bucket by running:

`policy.json` file as needed. Finally you can apply this modified policy back to the S3 bucket by running:

file as needed. Finally you can apply this modified policy back to the S3 bucket by running:

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://policy.json
```

- For API details, see [GetBucketPolicy](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    /**
     * Retrieves the policy for the specified Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} instance to use for making the request
     */
}
```

```
    * @param bucketName the name of the S3 bucket for which to retrieve the
    policy
    * @return the policy text for the specified bucket, or an empty string if an
    error occurs
    */
    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- For API details, see [GetBucketPolicy](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get the bucket policy.

```
import {
```

```
    GetBucketPolicyCommand,
    S3Client,
    S3ServiceException,
  } from "@aws-sdk/client-s3";

/**
 * Logs the policy for a specified bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Policy } = await client.send(
      new GetBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Policy for "${bucketName}":\n${Policy}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}. The bucket
        doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}.  ${caught.name}:
        ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).
- For API details, see [GetBucketPolicy](#) in *Amazon SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- For API details, see [GetBucketPolicy](#) in *Amazon SDK for Kotlin API reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command outputs the bucket policy associated with the given S3 bucket.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetBucketPolicy](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command outputs the bucket policy associated with the given S3 bucket.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetBucketPolicy](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_policy(self):
        """
        Get the security policy of the bucket.

        :return: The security policy of the specified bucket, in JSON format.
        """
        try:
            policy = self.bucket.Policy()
            logger.info(
                "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get policy for bucket '%s'.",
                             self.bucket.name)
```

```
        raise
    else:
        return json.loads(policy.policy)
```

- For API details, see [GetBucketPolicy](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  # configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
      #{e.message}"
      nil
    end
  end
end
```


- For API details, see [GetBucketPolicy](#) in *Amazon SDK for Ruby API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
    oo_result = lo_s3->getbucketpolicy(          " oo_result is returned for  
testing purposes. "  
    iv_bucket = iv_bucket_name ).  
    DATA(lv_policy) = oo_result->get_policy( ).  
    MESSAGE 'Retrieved bucket policy.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [GetBucketPolicy](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketPolicyStatus with a CLI

The following code examples show how to use GetBucketPolicyStatus.

CLI

Amazon CLI

To retrieve the policy status for a bucket indicating whether the bucket is public

The following `get-bucket-policy-status` example retrieves the policy status for the bucket `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-policy-status \  
  --bucket amzn-s3-demo-bucket
```

Output:

```
{  
  "PolicyStatus": {  
    "IsPublic": false  
  }  
}
```

- For API details, see [GetBucketPolicyStatus](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns policy status for the given S3 bucket, indicating whether the bucket is public.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetBucketPolicyStatus](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns policy status for the given S3 bucket, indicating whether the bucket is public.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetBucketPolicyStatus](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketReplication with an Amazon SDK or CLI

The following code examples show how to use GetBucketReplication.

CLI

Amazon CLI

The following command retrieves the replication configuration for a bucket named `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-replication --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "ReplicationConfiguration": {
    "Rules": [
      {
        "Status": "Enabled",
        "Prefix": "",
        "Destination": {
          "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket-backup",
          "StorageClass": "STANDARD"
        },
        "ID": "ZmUwNzE4ZmQ4tMjVhOS00MTlkLOGI4NDkzZTIWJjNTUtYTA1"
      }
    ],
    "Role": "arn:aws:iam::123456789012:role/s3-replication-role"
  }
}
```

- For API details, see [GetBucketReplication](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/**
 * Retrieves the replication details for the specified S3 bucket.
 *
 * @param s3Client      the S3 client used to interact with the S3
service
 * @param sourceBucketName the name of the S3 bucket to retrieve the
replication details for
 *
 * @throws S3Exception if there is an error retrieving the replication
details
 */
public static void getReplicationDetails(S3Client s3Client, String
sourceBucketName) {
    GetBucketReplicationRequest getRequest =
GetBucketReplicationRequest.builder()
        .bucket(sourceBucketName)
        .build();

    try {
        ReplicationConfiguration replicationConfig =
s3Client.getBucketReplication(getRequest).replicationConfiguration();
        ReplicationRule rule = replicationConfig.rules().get(0);
        System.out.println("Retrieved destination bucket: " +
rule.destination().bucket());
        System.out.println("Retrieved priority: " + rule.priority());
        System.out.println("Retrieved source-bucket replication rule status:
" + rule.status());

    } catch (S3Exception e) {
        System.err.println("Failed to retrieve replication details: " +
e.awsErrorDetails().errorMessage());
    }
}
```

```
}
```

- For API details, see [GetBucketReplication](#) in *Amazon SDK for Java 2.x API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Returns the replication configuration information set on the bucket named 'amzn-s3-demo-bucket'.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- For API details, see [GetBucketReplication](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Returns the replication configuration information set on the bucket named 'amzn-s3-demo-bucket'.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- For API details, see [GetBucketReplication](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketRequestPayment with a CLI

The following code examples show how to use GetBucketRequestPayment.

CLI

Amazon CLI

To retrieve the request payment configuration for a bucket

The following `get-bucket-request-payment` example retrieves the requester pays configuration for the specified bucket.

```
aws s3api get-bucket-request-payment \
  --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "Payer": "BucketOwner"
}
```

- For API details, see [GetBucketRequestPayment](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Returns the request payment configuration for the bucket named 'amzn-s3-demo-bucket'. By default, the bucket owner pays for downloads from the bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- For API details, see [GetBucketRequestPayment](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Returns the request payment configuration for the bucket named 'amzn-s3-demo-bucket'. By default, the bucket owner pays for downloads from the bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- For API details, see [GetBucketRequestPayment](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketTagging with a CLI

The following code examples show how to use GetBucketTagging.

CLI

Amazon CLI

The following command retrieves the tagging configuration for a bucket named `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "TagSet": [
    {
      "Value": "marketing",
      "Key": "organization"
    }
  ]
}
```

- For API details, see [GetBucketTagging](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns all the tags associated with the given bucket.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetBucketTagging](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns all the tags associated with the given bucket.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetBucketTagging](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketVersioning with a CLI

The following code examples show how to use GetBucketVersioning.

CLI

Amazon CLI

The following command retrieves the versioning configuration for a bucket named amzn-s3-demo-bucket:

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket
```

Output:

```
{  
  "Status": "Enabled"  
}
```

- For API details, see [GetBucketVersioning](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns the status of versioning with respect to the given bucket.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetBucketVersioning](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns the status of versioning with respect to the given bucket.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetBucketVersioning](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketWebsite with an Amazon SDK or CLI

The following code examples show how to use GetBucketWebsite.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
// Get the website configuration.
GetBucketWebsiteRequest getRequest = new
GetBucketWebsiteRequest()
{
    BucketName = bucketName,
};
GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
Console.WriteLine($"Index document:
{getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
Console.WriteLine($"Error document:
{getResponse.WebsiteConfiguration.ErrorDocument}");
```

- For API details, see [GetBucketWebsite](#) in *Amazon SDK for .NET API Reference*.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::S3::S3ClientConfiguration
                                   &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
        s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                   << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult =
            outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"
                  << std::endl
                  << "Index page : "
                  << websiteResult.GetIndexDocument().GetSuffix()
                  << std::endl
                  << "Error page: "
```

```
        << websiteResult.GetErrorDocument().GetKey()
        << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [GetBucketWebsite](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command retrieves the static website configuration for a bucket named `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-website --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- For API details, see [GetBucketWebsite](#) in *Amazon CLI Command Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get the website configuration.

```
import {
  GetBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the website configuration for a bucket.
 * @param {{ bucketName }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    console.log(
      `Your bucket is set up to host a website with the following configuration:
\n${JSON.stringify(response, null, 2)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchWebsiteConfiguration"
    ) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}. The
bucket isn't configured as a website.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}.
${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- For API details, see [GetBucketWebsite](#) in *Amazon SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns the details of the static website configurations of the given S3 bucket.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetBucketWebsite](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns the details of the static website configurations of the given S3 bucket.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetBucketWebsite](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetObject with an Amazon SDK or CLI

The following code examples show how to use `GetObject`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Learn the basics](#)
- [Get an object from a bucket if it has been modified](#)
- [Get an object from a Multi-Region Access Point](#)

- [Get started with encryption](#)
- [Getting started with Amazon S3](#)
- [Make conditional requests](#)
- [Track uploads and downloads](#)

.NET

Amazon SDK for .NET (v4)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Shows how to download an object from an Amazon S3 bucket to the
/// local computer.
/// </summary>
/// <param name="bucketName">The name of the bucket where the object is
/// currently stored.</param>
/// <param name="objectName">The name of the object to download.</param>
/// <param name="filePath">The path, including filename, where the
/// downloaded object will be stored.</param>
/// <returns>A boolean value indicating the success or failure of the
/// download process.</returns>
public async Task<bool> DownloadObjectFromBucketAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    var request = new GetObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
    };

    using GetObjectResponse response = await
        _amazonS3.GetObjectAsync(request);
```

```
try
{
    // Save object to local file
    await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error saving {objectName}: {ex.Message}");
    return false;
}
}
```

- For API details, see [GetObject](#) in *Amazon SDK for .NET API Reference*.

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get an object using a conditional request.

```
/// <summary>
/// Retrieves an object from Amazon S3 with a conditional request.
/// </summary>
/// <param name="objectKey">The key of the object to retrieve.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="conditionType">The type of condition: 'IfMatch',
'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional read is successful, False otherwise.</
returns>
```

```
public async Task<bool> GetObjectConditional(string objectKey, string
sourceBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
{
    try
    {
        var getObjectRequest = new GetObjectRequest
        {
            BucketName = sourceBucket,
            Key = objectKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                getObjectRequest.EtagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                getObjectRequest.EtagToNotMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfModifiedSince:
                getObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            case S3ConditionType.IfUnmodifiedSince:
                getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            default:
                throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
        }

        var response = await _amazonS3.GetObjectAsync(getObjectRequest);
        var sampleBytes = new byte[20];
        await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
        _logger.LogInformation($"Conditional read
successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
        return true;
    }
    catch (AmazonS3Exception e)
    {

```



```

        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional read failed: Precondition failed");
        }
        else if (e.ErrorCode == "NotModified")
        {
            _logger.LogError("Conditional read failed: Object not modified");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

```

- For API details, see [GetObject](#) in *Amazon SDK for .NET API Reference*.

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function download_object_from_bucket
#

```

```
# This function downloads an object in a bucket to a file.
#
# Parameters:
#     $1 - The name of the bucket to download the object from.
#     $2 - The path and file name to store the downloaded bucket.
#     $3 - The key (name) of the object in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function download_object_from_bucket() {
    local bucket_name=$1
    local destination_file_name=$2
    local object_name=$3
    local response

    response=$(aws s3api get-object \
        --bucket "$bucket_name" \
        --key "$object_name" \
        "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}
```

- For API details, see [GetObject](#) in *Amazon CLI Command Reference*.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
            << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [GetObject](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following example uses the `get-object` command to download an object from Amazon S3:

```
aws s3api get-object --bucket text-content --key dir/
my_images.tar.bz2 my_images.tar.bz2
```

Note that the outfile parameter is specified without an option name such as "`--outfile`". The name of the output file must be the last parameter in the command.

The example below demonstrates the use of `--range` to download a specific byte range from an object. Note the byte ranges needs to be prefixed with "bytes=":

```
aws s3api get-object --bucket text-content --key dir/my_data --  
range bytes=8888-9999 my_data_range
```

For more information about retrieving objects, see *Getting Objects in the Amazon S3 Developer Guide*.

- For API details, see [GetObject](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
actions
```

```
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
                objectKey, bucketName)
            err = noKey
        } else {
            log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
                objectKey, err)
        }
        return err
    }
    defer result.Body.Close()
    file, err := os.Create(fileName)
    if err != nil {
        log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
        return err
    }
    defer file.Close()
    body, err := io.ReadAll(result.Body)
    if err != nil {
        log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
            err)
    }
    _, err = file.Write(body)
    return err
}
```

- For API details, see [GetObject](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Read data as a byte array using an [S3Client](#).

```
/**
 * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
 * and writes them to a local file.
 *
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName     the key (or name) of the S3 object to retrieve
 * @param path        the local file path where the object's bytes will be
 * written
 * @return a {@link CompletableFuture} that completes when the object bytes
 * have been written to the local file
 */
public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

    CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
    return response.thenAccept(objectBytes -> {
        try {
            byte[] data = objectBytes.asByteArray();
            Path filePath = Paths.get(path);
```

```

        Files.write(filePath, data);
        logger.info("Successfully obtained bytes from an S3 object");
    } catch (IOException ex) {
        throw new RuntimeException("Failed to write data to file", ex);
    }
}).whenComplete((resp, ex) -> {
    if (ex != null) {
        throw new RuntimeException("Failed to get object bytes from S3",
ex);
    }
});
}
}

```

Use an [S3TransferManager](#) to [download an object](#) in an S3 bucket to a local file. View the [complete file](#) and [test](#).

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String
bucketName,

                                String key, String downloadedFilePath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .destination(Paths.get(downloadedFilePath))

```

```

        .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{ }]",
downloadResult.response().contentType());
        return downloadResult.response().contentType();
    }

```

Read tags that belong to an object using an [S3Client](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:

```



```
        bucketName - The Amazon S3 bucket name.\s
        keyName - A key name that represents the object.\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listTags(s3, bucketName, keyName);
    s3.close();
}

/**
 * Lists the tags associated with an Amazon S3 object.
 *
 * @param s3 the S3Client object used to interact with the Amazon S3 service
 * @param bucketName the name of the S3 bucket that contains the object
 * @param keyName the key (name) of the S3 object
 */
public static void listTags(S3Client s3, String bucketName, String keyName) {
    try {
        GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        GetObjectTaggingResponse tags =
s3.getObjectTagging(getTaggingRequest);
        List<Tag> tagSet = tags.tagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.key());
            System.out.println(tag.value());
        }

    } catch (S3Exception e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Get a URL for an object using an [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the URL for a specific object in an Amazon S3 bucket.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the name of the object for which the URL should be
    retrieved
     * @throws S3Exception if there is an error retrieving the URL for the
    specified object
     */
    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

Get an object by using the S3Presigner client object using an [S3Client](#).

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
```

```

        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
            .region(region)
            .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    /**
     * Generates a pre-signed URL for an Amazon S3 object.
     *
     * @param presigner The {@link S3Presigner} instance to use for generating
     the pre-signed URL.
     * @param bucketName The name of the Amazon S3 bucket where the object is
     stored.
     * @param keyName The key name (file name) of the object in the Amazon S3
     bucket.
     *
     * @throws S3Exception If there is an error interacting with the Amazon S3
     service.
     * @throws IOException If there is an error opening the HTTP connection or
     reading/writing the request/response.
     */
    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
        String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            GetObjectPresignRequest getObjectPresignRequest =
                GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(60))
                    .getObjectRequest(getObjectRequest)
                    .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
                presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
                presignedGetObjectRequest.url().openConnection();

```

```

        presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
            values.forEach(value -> {
                connection.addRequestProperty(header, value);
            });
        });

        // Send any request payload that the service needs (not needed when
        // isBrowserExecutable is true).
        if (presignedGetObjectRequest.signedPayload().isPresent()) {
            connection.setDoOutput(true);

            try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
                OutputStream httpOutputStream =
connection.getOutputStream()) {
                IoUtils.copy(signedPayload, httpOutputStream);
            }
        }

        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            System.out.println("Service returned response: ");
            IoUtils.copy(content, System.out);
        }

    } catch (S3Exception | IOException e) {
        e.printStackTrace();
    }
}
}

```

Get an object by using a `ResponseTransformer` object and [S3Client](#).

```

import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;

```

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }
}
```

```
}

/**
 * Retrieves the bytes of an object stored in an Amazon S3 bucket and saves
 * them to a local file.
 *
 * @param s3 The S3Client instance used to interact with the Amazon S3
 * service.
 * @param bucketName The name of the S3 bucket where the object is stored.
 * @param keyName The key (or name) of the S3 object.
 * @param path The local file path where the object's bytes will be saved.
 * @throws IOException If an I/O error occurs while writing the bytes to the
 * local file.
 * @throws S3Exception If an error occurs while retrieving the object from
 * the S3 bucket.
 */
public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObject(objectRequest, ResponseTransformer.toBytes());
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```


- For API details, see [GetObject](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Download the object.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    // The Body object also has 'transformToArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log(str);
  } catch (caught) {
```

```
    if (caught instanceof NoSuchKey) {
        console.error(
            `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`
        );
    } else {
        throw caught;
    }
}
};
```

Download the object on condition its ETag matches the one provided.

```
import {
    GetObjectCommand,
    NoSuchKey,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
    const client = new S3Client({});

    try {
        const response = await client.send(
            new GetObjectCommand({
                Bucket: bucketName,
                Key: key,
                IfMatch: eTag,
            })
        );
    }
```

```
// The Body object also has 'transformToByteArray' and 'transformToWebStream'
methods.
const str = await response.Body.transformToString();
console.log("Success. Here is text of the file:", str);
} catch (caught) {
  if (caught instanceof NoSuchKey) {
    console.error(
      `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
```

```
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Download the object on condition its ETag does not match the one provided.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfNoneMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  }
}
```

```
    } catch (caught) {
      if (caught instanceof NoSuchKey) {
        console.error(
          `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`
        );
      } else if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`
        );
      } else {
        throw caught;
      }
    }
  };

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};
```

```
if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Download the object using on condition it has been created or modified in a given timeframe.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfModifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
```

```
        console.error(
            `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
        key: {
            type: "string",
            required: true,
        },
    };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}
```

```
}  
}
```

Download the object using on condition it has not been created or modified in a given timeframe.

```
import {  
  GetObjectCommand,  
  NoSuchKey,  
  S3Client,  
  S3ServiceException,  
} from "@aws-sdk/client-s3";  
  
/**  
 * Get a single object from a specified S3 bucket.  
 * @param {{ bucketName: string, key: string }}  
 */  
export const main = async ({ bucketName, key }) => {  
  const client = new S3Client({});  
  const date = new Date();  
  date.setDate(date.getDate() - 1);  
  try {  
    const response = await client.send(  
      new GetObjectCommand({  
        Bucket: bucketName,  
        Key: key,  
        IfUnmodifiedSince: date,  
      })),  
    );  
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'  
    methods.  
    const str = await response.Body.transformToString();  
    console.log("Success. Here is text of the file:", str);  
  } catch (caught) {  
    if (caught instanceof NoSuchKey) {  
      console.error(  
        `Error from S3 while getting object "${key}" from "${bucketName}". No  
        such key exists.`,  
      );  
    } else if (caught instanceof S3ServiceException) {  
      console.error(  

```



```
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:  
        ${caught.message}`,  
        );  
    } else {  
        throw caught;  
    }  
}  
};  
  
// Call function if run directly  
import { parseArgs } from "node:util";  
import {  
    isMain,  
    validateArgs,  
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";  
  
const loadArgs = () => {  
    const options = {  
        bucketName: {  
            type: "string",  
            required: true,  
        },  
        key: {  
            type: "string",  
            required: true,  
        },  
    };  
    const results = parseArgs({ options });  
    const { errors } = validateArgs({ options }, results);  
    return { errors, results };  
};  
  
if (isMain(import.meta.url)) {  
    const { errors, results } = loadArgs();  
    if (!errors) {  
        main(results.values);  
    } else {  
        console.error(errors.join("\n"));  
    }  
}
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).

- For API details, see [GetObject](#) in *Amazon SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
suspend fun getObjectBytes(  
    bucketName: String,  
    keyName: String,  
    path: String,  
) {  
    val request =  
        GetObjectRequest {  
            key = keyName  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.getObject(request) { resp ->  
            val myFile = File(path)  
            resp.body?.writeToFile(myFile)  
            println("Successfully read $keyName from $bucketName")  
        }  
    }  
}
```

- For API details, see [GetObject](#) in *Amazon SDK for Kotlin API reference*.

PHP

SDK for PHP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get an object.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $file = $this->s3client->getObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}
```

- For API details, see [GetObject](#) in *Amazon SDK for PHP API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command retrieves item "sample.txt" from bucket "amzn-s3-demo-bucket" and saves it to a file named "local-sample.txt" in the current location. The file "local-sample.txt" does not have to exist before this command is called.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-
sample.txt
```

Example 2: This command retrieves virtual directory "DIR" from bucket "amzn-s3-demo-bucket" and saves it to a folder named "Local-DIR" in the current location. The folder "Local-DIR" does not have to exist before this command is called.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Example 3: Downloads all objects with keys ending in '.json' from buckets with 'config' in the bucket name to files in the specified folder. The object keys are used to set the filenames.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- For API details, see [GetObject](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command retrieves item "sample.txt" from bucket "amzn-s3-demo-bucket" and saves it to a file named "local-sample.txt" in the current location. The file "local-sample.txt" does not have to exist before this command is called.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Example 2: This command retrieves virtual directory "DIR" from bucket "amzn-s3-demo-bucket" and saves it to a folder named "Local-DIR" in the current location. The folder "Local-DIR" does not have to exist before this command is called.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Example 3: Downloads all objects with keys ending in '.json' from buckets with 'config' in the bucket name to files in the specified folder. The object keys are used to set the filenames.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

Example 4: Downloads a single object using multipart parallel download for improved performance on large files.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt -UseMultipartDownload
```

Example 5: Downloads a large object using RANGE mode with a custom 64MB part size. RANGE mode uses HTTP byte-range requests and works for all objects regardless of how they were uploaded.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Example 6: Downloads all objects with the key prefix "data/" to a local folder using multipart download with concurrent file downloads.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Example 7: Downloads a single object using multipart download with 20 concurrent connections for maximum throughput on high-bandwidth networks.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Example 8: Downloads all objects with the key prefix "data/" to a local folder using multipart download, continuing past individual file failures instead of aborting.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- For API details, see [GetObject](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                               that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get(self):
        """
        Gets the object.

        :return: The object data in bytes.
        """
        try:
            body = self.object.get()["Body"].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
```

```
        raise
    else:
        return body
```

Get an object using a conditional request.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
        'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
```

```

        Key=object_key,
        **{condition_type: condition_value},
    )
    sample_bytes = response["Body"].read(20)
    print(
        f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
    )
    print(f"\t{sample_bytes}")
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    if error_code == "PreconditionFailed":
        print("\tConditional read failed: Precondition failed")
    elif error_code == "304": # Not modified error code.
        print("\tConditional read failed: Object not modified")
    else:
        logger.error(f"Unexpected error: {error_code}")
        raise

```

- For API details, see [GetObject](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get an object.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.

```



```

def initialize(object)
  @object = object
end

# Gets the object directly to a file.
#
# @param target_path [String] The path to the file where the object is
downloaded.
# @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
successful; otherwise nil.
def get_object(target_path)
  @object.get(response_target: target_path)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes) downloaded to
#{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

Get an object and report its server-side encryption state.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.

```

```

def initialize(object)
  @object = object
end

# Gets the object into memory.
#
# @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
successful; otherwise nil.
def object
  @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- For API details, see [GetObject](#) in *Amazon SDK for Ruby API Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
async fn get_object(client: Client, opt: Opt) -> Result<usize, S3ExampleError> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone()).map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to initialize file for saving S3 download: {err:?}"
        ))
    })?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
        .send()
        .await?;

    let mut byte_count = 0_usize;
    while let Some(bytes) = object.body.try_next().await.map_err(|err| {
        S3ExampleError::new(format!("Failed to read from S3 download stream:
{err:?}"))
    })? {
        let bytes_len = bytes.len();
        file.write_all(&bytes).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to write from S3 download stream to local file: {err:?}"
            ))
        })?;
        trace!("Intermediate write of {bytes_len}");
        byte_count += bytes_len;
    }
}
```

```
    Ok(byte_count)
}
```

- For API details, see [GetObject](#) in *Amazon SDK for Rust API reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.
    oo_result = lo_s3->getobject(          " oo_result is returned for
testing purposes. "
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key ).
    DATA(lv_object_data) = oo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see [GetObject](#) in *Amazon SDK for SAP ABAP API reference*.

Swift

SDK for Swift

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import AWSS3

    public func downloadFile(bucket: String, key: String, to: String) async
    throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
            }

            try data.write(to: fileUrl)
        }
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file. "))
            throw error
        }
    }
}
```

```
import AWSS3
```

```
public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        return data
    }
    catch {
        print("ERROR: ", dump(error, name: "Reading a file. "))
        throw error
    }
}
```

- For API details, see [GetObject](#) in *Amazon SDK for Swift API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetObjectAcl with an Amazon SDK or CLI

The following code examples show how to use GetObjectAcl.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Manage access control lists \(ACLs\)](#)

C++

SDK for C++

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
        s3Client.GetObjectAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObjectAcl: "
                    << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            std::cout << "For object " << objectKey << ": "
                        << std::endl << std::endl;

            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
```

```

        << getGranteeTypeString(grantee.GetType()) <<
std::endl;
    }

    if (grantee.DisplayNameHasBeenSet()) {
        std::cout << "Display name: "
        << grantee.GetDisplayName() << std::endl;
    }

    if (grantee.EmailAddressHasBeenSet()) {
        std::cout << "Email address: "
        << grantee.GetEmailAddress() << std::endl;
    }

    if (grantee.IDHasBeenSet()) {
        std::cout << "ID: "
        << grantee.GetID() << std::endl;
    }

    if (grantee.URIHasBeenSet()) {
        std::cout << "URI: "
        << grantee.GetURI() << std::endl;
    }

    std::cout << "Permission: " <<
        getPermissionString(grant.GetPermission()) <<
        std::endl << std::endl;
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
 */
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:

```



```

        return "Canonical user ID of an AWS account";
    case Aws::S3::Model::Type::Group:
        return "Predefined Amazon S3 group";
    case Aws::S3::Model::Type::NOT_SET:
        return "Not set";
    default:
        return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
    string.
    /*!
    \param permission: Permission enumeration.
    \return String: Human-readable string
    */
    Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
        switch (permission) {
            case Aws::S3::Model::Permission::FULL_CONTROL:
                return "Can read this object's data and its metadata, "
                    "and read/write this object's permissions";
            case Aws::S3::Model::Permission::NOT_SET:
                return "Permission not set";
            case Aws::S3::Model::Permission::READ:
                return "Can read this object's data and its metadata";
            case Aws::S3::Model::Permission::READ_ACP:
                return "Can read this object's permissions";
            // case Aws::S3::Model::Permission::WRITE // Not applicable.
            case Aws::S3::Model::Permission::WRITE_ACP:
                return "Can write this object's permissions";
            default:
                return "Permission unknown";
        }
    }
}

```

- For API details, see [GetObjectAcl](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command retrieves the access control list for an object in a bucket named `amzn-s3-demo-bucket`:

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key index.html
```

Output:

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    },
    {
      "Grantee": {
        "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
      },
      "Permission": "READ"
    }
  ]
}
```

- For API details, see [GetObjectAcl](#) in *Amazon CLI Command Reference*.

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- For API details, see [GetObjectAcl](#) in *Amazon SDK for Kotlin API reference*.

PowerShell

Tools for PowerShell V5

Example 1: The command gets the details of the object owner of the S3 object.

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
*).Owner
```

Output:

```

DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544

```

- For API details, see [GetObjectAcl](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                                that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
            logger.info(
                "Got ACL for object %s owned by %s.",
                self.object.key,

```

```
        acl.owner["DisplayName"],
    )
except ClientError:
    logger.exception("Couldn't get ACL for object %s.", self.object.key)
    raise
else:
    return acl
```

- For API details, see [GetObjectAcl](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.
    oo_result = lo_s3->getobjectacl(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key ).
    MESSAGE 'Retrieved object ACL.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see [GetObjectAcl](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use `GetObjectAttributes` with an Amazon SDK or CLI

The following code examples show how to use `GetObjectAttributes`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Work with Amazon S3 object integrity](#)

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object is stored.
    \param key: The unique identifier (key) of the object within the S3 bucket.
    \param hashMethod: The hashing algorithm used to calculate the hash value of
the object.
    \param[out] hashData: The retrieved hash.
    \param[out] partHashes: The part hashes if available.
    \param client: The S3 client instance used to retrieve the object.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
&key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   Aws::String &hashData,
                                   std::vector<Aws::String> *partHashes,
                                   const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
```

```

request.SetBucket(bucket);
request.SetKey(key);

if (hashMethod == MD5) {
    Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
    attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
    request.SetObjectAttributes(attributes);

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        hashData = result.GetETag();
    } else {
        std::cerr << "Error retrieving object etag attributes." <<
outcome.GetError().GetMessage() << std::endl;
        return false;
    }
} else { // hashMethod != MD5
    Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
    attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
    request.SetObjectAttributes(attributes);

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                break; // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
            case AwsDoc::S3::MD5:
                break; // MD5 is not supported.
#pragma clang diagnostic pop
            case AwsDoc::S3::SHA1:
                hashData = result.GetChecksum().GetChecksumSHA1();
                break;
            case AwsDoc::S3::SHA256:
                hashData = result.GetChecksum().GetChecksumSHA256();

```

```

        break;
    case AwsDoc::S3::CRC32:
        hashData = result.GetChecksum().GetChecksumCRC32();
        break;
    case AwsDoc::S3::CRC32C:
        hashData = result.GetChecksum().GetChecksumCRC32C();
        break;
    default:
        std::cerr << "Unknown hash method." << std::endl;
        return false;
    }
} else {
    std::cerr << "Error retrieving object checksum attributes." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

if (nullptr != partHashes) {
    attributes.clear();
    attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
    request.SetObjectAttributes(attributes);
    outcome = client.GetObjectAttributes(request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
        for (const Aws::S3::Model::ObjectPart &part: parts) {
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                    break;
                case AwsDoc::S3::MD5: // MD5 is not supported.
                    break;
                case AwsDoc::S3::SHA1:
                    partHashes->push_back(part.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:
                    partHashes->push_back(part.GetChecksumSHA256());
                    break;
                case AwsDoc::S3::CRC32:
                    partHashes->push_back(part.GetChecksumCRC32());
                    break;
                case AwsDoc::S3::CRC32C:

```



```

        partHashes->push_back(part.GetChecksumCRC32C());
        break;
    default:
        std::cerr << "Unknown hash method." << std::endl;
        return false;
    }
}
} else {
    std::cerr << "Error retrieving object attributes for object
parts." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}
}
}
return true;
}

```

- For API details, see [GetObjectAttributes](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

To retrieves metadata from an object without returning the object itself

The following `get-object-attributes` example retrieves metadata from the object `doc1.rtf`.

```

aws s3api get-object-attributes \
  --bucket amzn-s3-demo-bucket \
  --key doc1.rtf \
  --object-attributes "StorageClass" "ETag" "ObjectSize"

```

Output:

```

{
  "LastModified": "2022-03-15T19:37:31+00:00",
  "VersionId": "IuCPjXTDzHNfldAuitVBIKJpF2p1fg4P",
  "ETag": "b662d79adeb7c8d787ea7eafb9ef6207",

```

```
"StorageClass": "STANDARD",  
"ObjectSize": 405  
}
```

For more information, see [GetObjectAttributes](#) in the Amazon S3 API Reference.

- For API details, see [GetObjectAttributes](#) in *Amazon CLI Command Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetObjectLegalHold with an Amazon SDK or CLI

The following code examples show how to use `GetObjectLegalHold`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Lock Amazon S3 objects](#)

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>  
/// Get the legal hold details for an S3 object.  
/// </summary>  
/// <param name="bucketName">The bucket of the object.</param>  
/// <param name="objectKey">The object key.</param>  
/// <returns>The object legal hold details.</returns>  
public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,  
    string objectKey)  
{  
    try
```

```
{
    var request = new GetObjectLegalHoldRequest()
    {
        BucketName = bucketName,
        Key = objectKey
    };

    var response = await _amazonS3.GetObjectLegalHoldAsync(request);
    Console.WriteLine($"{objectKey} in
{bucketName}: " +
        $"{response.LegalHold.Status}");
    return response.LegalHold;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Unable to fetch legal hold: '{ex.Message}'");
    return new ObjectLockLegalHold();
}
}
```

- For API details, see [GetObjectLegalHold](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

Retrieves the Legal Hold status of an object

The following `get-object-legal-hold` example retrieves the Legal Hold status for the specified object.

```
aws s3api get-object-legal-hold \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf
```

Output:

```
{
  "LegalHold": {
    "Status": "ON"
  }
}
```

```
}
```

- For API details, see [GetObjectLegalHold](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectLegalHold retrieves the legal hold status for an S3 object.  
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key  
    string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {  
    var status *types.ObjectLockLegalHoldStatus
```

```
input := &s3.GetObjectLegalHoldInput{
    Bucket:    aws.String(bucket),
    Key:       aws.String(key),
    VersionId: aws.String(versionId),
}

output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
if err != nil {
    var noSuchKeyErr *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noSuchKeyErr) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noSuchKeyErr
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            log.Printf("Object %s does not have an object lock configuration.\n", key)
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    }
} else {
    status = &output.LegalHold.Status
}

return status, err
}
```

- For API details, see [GetObjectLegalHold](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
    }

    return null;
}
```

- For API details, see [GetObjectLegalHold](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import {
    GetObjectLegalHoldCommand,
```

```
S3Client,
S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get an object's current legal hold status.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectLegalHoldCommand({
        Bucket: bucketName,
        Key: key,
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the
bucket>",
        // VersionId: "<the specific version id of the object to check>",
      }),
    );
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in
${bucketName}. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in
${bucketName} from ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
```

```
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- For API details, see [GetObjectLegalHold](#) in *Amazon SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Put an object legal hold.

```
def get_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Get the legal hold status of a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to get the legal hold status of.
    """
    print()
    logger.info("Getting legal hold status of file [%s] in bucket [%s]", key,
bucket)
    try:
        response = s3_client.get_object_legal_hold(Bucket=bucket, Key=key)
        legal_hold_status = response["LegalHold"]["Status"]
        logger.debug(
            "Legal hold status of file [%s] in bucket [%s] is [%s]",
            key,
            bucket,
            legal_hold_status,
        )
    except Exception as e:
        logger.error(
            "Failed to get legal hold status of file [%s] in bucket [%s]: %s",
            key,
            bucket,
            e,
        )
```

- For API details, see [GetObjectLegalHold](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
    oo_result = lo_s3->getobjectlegalhold(          " oo_result is returned  
for testing purposes. "  
    iv_bucket = iv_bucket_name  
    iv_key = iv_object_key ).  
    MESSAGE 'Retrieved object legal hold status.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
    MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [GetObjectLegalHold](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetObjectLockConfiguration with an Amazon SDK or CLI

The following code examples show how to use GetObjectLockConfiguration.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Lock Amazon S3 objects](#)

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Get the object lock configuration details for an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to get details.</param>
/// <returns>The bucket's object lock configuration details.</returns>
public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
{
    try
    {
        var request = new GetObjectLockConfigurationRequest()
        {
            BucketName = bucketName
        };

        var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
        Console.WriteLine($"\\tBucket object lock config for {bucketName} in
{bucketName}: " +
                        $"\\n\\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                        $"\\n\\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

        return response.ObjectLockConfiguration;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch object lock config:
'{ex.Message}'");
        return new ObjectLockConfiguration();
    }
}
```

```
}
```

- For API details, see [GetObjectLockConfiguration](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To retrieve an object lock configuration for a bucket

The following `get-object-lock-configuration` example retrieves the object lock configuration for the specified bucket.

```
aws s3api get-object-lock-configuration \  
  --bucket amzn-s3-demo-bucket-with-object-lock
```

Output:

```
{  
  "ObjectLockConfiguration": {  
    "ObjectLockEnabled": "Enabled",  
    "Rule": {  
      "DefaultRetention": {  
        "Mode": "COMPLIANCE",  
        "Days": 50  
      }  
    }  
  }  
}
```

- For API details, see [GetObjectLockConfiguration](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectLockConfiguration retrieves the object lock configuration for an S3  
// bucket.  
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket  
    string) (*types.ObjectLockConfiguration, error) {  
    var lockConfig *types.ObjectLockConfiguration  
    input := &s3.GetObjectLockConfigurationInput{  
        Bucket: aws.String(bucket),  
    }  
}
```

```
output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
"ObjectLockConfigurationNotFoundError" {
        log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
        err = nil
    }
} else {
    lockConfig = output.ObjectLockConfiguration
}

return lockConfig, err
}
```

- For API details, see [GetObjectLockConfiguration](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .build();
}
```

```

    GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
    System.out.println("Bucket object lock config for "+bucketName+": ");
    System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().objectLockEnabled());
    System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}

```

- For API details, see [GetObjectLockConfiguration](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

import {
  GetObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Gets the Object Lock configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { ObjectLockConfiguration } = await client.send(
      new GetObjectLockConfigurationCommand({
        Bucket: bucketName,
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the
        bucket>",

```

```

    })),
  );
  console.log(
    `Object Lock Configuration:\n${JSON.stringify(ObjectLockConfiguration)}`,
  );
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while getting object lock configuration for ${bucketName}.
The bucket doesn't exist.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while getting object lock configuration for ${bucketName}.
${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
  };
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

```



```
if (isMain(import.meta.url)) {  
  const { errors, results } = loadArgs();  
  if (!errors) {  
    main(results.values);  
  } else {  
    console.error(errors.join("\n"));  
  }  
}
```

- For API details, see [GetObjectLockConfiguration](#) in *Amazon SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns the value 'Enabled' if Object lock configuration is enabled for the given S3 bucket.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Output:

```
Value  
-----  
Enabled
```

- For API details, see [GetObjectLockConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns the value 'Enabled' if Object lock configuration is enabled for the given S3 bucket.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Output:

Value

Enabled

- For API details, see [GetObjectLockConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get the object lock configuration.

```
def is_object_lock_enabled(s3_client, bucket: str) -> bool:
    """
    Check if object lock is enabled for a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket to check.

    Returns:
        True if object lock is enabled, False otherwise.
    """
    try:
        response = s3_client.get_object_lock_configuration(Bucket=bucket)
        return (
            "ObjectLockConfiguration" in response
            and response["ObjectLockConfiguration"]["ObjectLockEnabled"] ==
            "Enabled"
        )
    except s3_client.exceptions.ClientError as e:
        if e.response["Error"]["Code"] == "ObjectLockConfigurationNotFoundError":
            return False
        else:
```

```
raise
```

- For API details, see [GetObjectLockConfiguration](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
    oo_result = lo_s3->getobjectlockconfiguration(          " oo_result is  
returned for testing purposes. "  
    iv_bucket = iv_bucket_name ).  
    MESSAGE 'Retrieved object lock configuration.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [GetObjectLockConfiguration](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetObjectRetention with an Amazon SDK or CLI

The following code examples show how to use GetObjectRetention.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Lock Amazon S3 objects](#)

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Get the retention period for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object retention details.</returns>
public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };

        var response = await _amazonS3.GetObjectRetentionAsync(request);
        Console.WriteLine($"Object retention for {objectKey} in
{bucketName}: " +
                        $"{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}");
        return response.Retention;
    }
    catch (AmazonS3Exception ex)
    {

```

```
        Console.WriteLine($"\\tUnable to fetch object lock retention:
'{ex.Message}');
        return new ObjectLockRetention();
    }
}
```

- For API details, see [GetObjectRetention](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To retrieve the object retention configuration for an object

The following `get-object-retention` example retrieves the object retention configuration for the specified object.

```
aws s3api get-object-retention \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --key doc1.rtf
```

Output:

```
{
  "Retention": {
    "Mode": "GOVERNANCE",
    "RetainUntilDate": "2025-01-01T00:00:00.000Z"
  }
}
```

- For API details, see [GetObjectRetention](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectRetention retrieves the object retention configuration for an S3  
// object.  
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key  
string) (*types.ObjectLockRetention, error) {  
    var retention *types.ObjectLockRetention  
    input := &s3.GetObjectRetentionInput{  
        Bucket: aws.String(bucket),  
        Key:    aws.String(key),  
    }
```

```

}

output, err := actor.S3Client.GetObjectRetention(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have locking enabled.", bucket)
            err = nil
        }
    }
} else {
    retention = output.Retention
}

return retention, err
}

```

- For API details, see [GetObjectRetention](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {

```

```

        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("Object retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

```

- For API details, see [GetObjectRetention](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

import {
    GetObjectRetentionCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the "RetainUntilDate" for an object in an S3 bucket.
 * @param {{ bucketName: string, key: string }}

```



```
*/
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const { Retention } = await client.send(
      new GetObjectRetentionCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    console.log(
      `${key} in ${bucketName} will be retained until
      ${Retention.RetainUntilDate}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchObjectLockConfiguration"
    ) {
      console.warn(
        `The object "${key}" in the bucket "${bucketName}" does not have an
        ObjectLock configuration.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object retention settings for
        "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utills/util-node.js";

const loadArgs = () => {
  const options = {
```

```
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- For API details, see [GetObjectRetention](#) in *Amazon SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: The command returns the mode and date till the object would be retained.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- For API details, see [GetObjectRetention](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: The command returns the mode and date till the object would be retained.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- For API details, see [GetObjectRetention](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetObjectTagging with a CLI

The following code examples show how to use GetObjectTagging.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Get started with tags](#)

CLI

Amazon CLI

To retrieve the tags attached to an object

The following get-object-tagging example retrieves the values for the specified key from the specified object.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc1.rtf
```

Output:

```
{  
  "TagSet": [  
    {  
      "Value": "confidential",  
      "Key": "designation"  
    }  
  ]  
}
```

The following `get-object-tagging` example tries to retrieve the tag sets of the object `doc2.rtf`, which has no tags.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc2.rtf
```

Output:

```
{  
  "TagSet": []  
}
```

The following `get-object-tagging` example retrieves the tag sets of the object `doc3.rtf`, which has multiple tags.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc3.rtf
```

Output:

```
{  
  "TagSet": [  
    {  
      "Value": "confidential",  
      "Key": "designation"  
    },  
    {  
      "Value": "finance",  
      "Key": "department"  
    },  
    {  
      "Value": "payroll",  
      "Key": "team"  
    }  
  ]  
}
```

- For API details, see [GetObjectTagging](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: The sample returns the tags associated with the object present on the given S3 bucket.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Key  Value
---  -
test value
```

- For API details, see [GetObjectTagging](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: The sample returns the tags associated with the object present on the given S3 bucket.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Key  Value
---  -
test value
```

- For API details, see [GetObjectTagging](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetPublicAccessBlock with a CLI

The following code examples show how to use GetPublicAccessBlock.

CLI

Amazon CLI

To set or modify the block public access configuration for a bucket

The following `get-public-access-block` example displays the block public access configuration for the specified bucket.

```
aws s3api get-public-access-block \  
    --bucket amzn-s3-demo-bucket
```

Output:

```
{  
  "PublicAccessBlockConfiguration": {  
    "IgnorePublicAcls": true,  
    "BlockPublicPolicy": true,  
    "BlockPublicAcls": true,  
    "RestrictPublicBuckets": true  
  }  
}
```

- For API details, see [GetPublicAccessBlock](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: The command returns the public access block configuration of the given S3 bucket.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetPublicAccessBlock](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: The command returns the public access block configuration of the given S3 bucket.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [GetPublicAccessBlock](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use HeadBucket with an Amazon SDK or CLI

The following code examples show how to use HeadBucket.

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
#####
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#     $1 - The name of the bucket to check.
#
# Returns:
#     0 - If the bucket already exists.
#     1 - If the bucket doesn't exist.
#####
function bucket_exists() {
    local bucket_name
    bucket_name=$1

    # Check whether the bucket already exists.
    # We suppress all output - we're interested only in the return code.
```

```
if aws s3api head-bucket \  
  --bucket "$bucket_name" \  
  >/dev/null 2>&1; then  
  return 0 # 0 in Bash script means true.  
else  
  return 1 # 1 in Bash script means false.  
fi  
}
```

- For API details, see [HeadBucket](#) in *Amazon CLI Command Reference*.

CLI

Amazon CLI

The following command verifies access to a bucket named `amzn-s3-demo-bucket`:

```
aws s3api head-bucket --bucket amzn-s3-demo-bucket
```

If the bucket exists and you have access to it, no output is returned. Otherwise, an error message will be shown. For example:

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- For API details, see [HeadBucket](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (
```



```

"bytes"
"context"
"errors"
"fmt"
"io"
"log"
"os"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
    (bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
    exists := true
    if err != nil {
        var apiError smithy.APIError
        if errors.As(err, &apiError) {
            switch apiError.(type) {
            case *types.NotFound:
                log.Printf("Bucket %v is available.\n", bucketName)
                exists = false
                err = nil
            default:

```

```
    log.Printf("Either you don't have access to bucket %v or another error
occurred. "+
    "Here's what happened: %v\n", bucketName, err)
}
}
} else {
    log.Printf("Bucket %v exists and you already own it.", bucketName)
}

return exists, err
}
```

- For API details, see [HeadBucket](#) in *Amazon SDK for Go API Reference*.

PowerShell

Tools for PowerShell V5

Example 1: This command returns the output with HTTP status code 200 OK for existing bucket when user has permission to access it. BucketArn parameter is only supported for S3 directory buckets.

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

Output:

```
AccessPointAlias    : False
BucketArn           :
BucketLocationName  :
BucketLocationType  :
BucketRegion        : us-east-2
ResponseMetadata    : Amazon.Runtime.ResponseMetadata
ContentLength       : 0
HttpStatusCode       : OK
```

Example 2: This command throws error with HTTP status code NotFound for non-existent bucket.

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

Output:

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status Code NotFound. No further error information was returned by the service.
```

Example 3: This command throws error with HTTP status code Forbidden for existing bucket where user does not have permission to access it.

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

Output:

```
Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status Code Forbidden. No further error information was returned by the service.
```

- For API details, see [HeadBucket](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
def exists(self):
    """
    Determine whether the bucket exists and you have access to it.

    :return: True when the bucket exists; otherwise, False.
    """
    try:
        self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
        logger.info("Bucket %s exists.", self.bucket.name)
        exists = True
    except ClientError:
        logger.warning(
            "Bucket %s doesn't exist or you don't have access to it.",
            self.bucket.name,
        )
        exists = False
    return exists
```

- For API details, see [HeadBucket](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.
    oo_result = lo_s3->headbucket(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket exists and you have access to it.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see [HeadBucket](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use HeadObject with an Amazon SDK or CLI

The following code examples show how to use HeadObject.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Getting started with Amazon S3](#)

CLI

Amazon CLI

The following command retrieves metadata for an object in a bucket named `amzn-s3-demo-bucket`:

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

Output:

```
{
  "AcceptRanges": "bytes",
  "ContentType": "text/html",
  "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
  "ContentLength": 77,
  "VersionId": "null",
  "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
  "Metadata": {}
}
```

- For API details, see [HeadObject](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Determine the content type of an object.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    getContentType(s3, bucketName, keyName);
    s3.close();
}

/**
 * Retrieves the content type of an object stored in an Amazon S3 bucket.
 *
 * @param s3 an instance of the {@link S3Client} class, which is used to
interact with the Amazon S3 service
 * @param bucketName the name of the S3 bucket where the object is stored
 * @param keyName the key (file name) of the object in the S3 bucket
 */
public static void getContentType(S3Client s3, String bucketName, String
keyName) {
    try {
        HeadObjectRequest objectRequest = HeadObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        HeadObjectResponse objectHead = s3.headObject(objectRequest);
        String type = objectHead.contentType();
        System.out.println("The object content type is " + type);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Get the restore status of an object.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Checks the restoration status of an Amazon S3 object.
     *
     * @param s3          an instance of the {@link S3Client} class used to
     *                    interact with the Amazon S3 service
     * @param bucketName  the name of the Amazon S3 bucket where the object is
     *                    stored
     * @param keyName     the name of the Amazon S3 object to be checked

```



```
    * @throws S3Exception if an error occurs while interacting with the Amazon
    S3 service
    */
    public static void checkStatus(S3Client s3, String bucketName, String
    keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
            response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [HeadObject](#) in *Amazon SDK for Java 2.x API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
```

```

def initialize(object)
  @object = object
end

# Checks whether the object exists.
#
# @return [Boolean] True if the object exists; otherwise false.
def exists?
  @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object
#{@object.bucket.name}:#{@object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- For API details, see [HeadObject](#) in *Amazon SDK for Ruby API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use ListBucketAnalyticsConfigurations with a CLI

The following code examples show how to use ListBucketAnalyticsConfigurations.

CLI

Amazon CLI

To retrieve a list of analytics configurations for a bucket

The following `list-bucket-analytics-configurations` retrieves a list of analytics configurations for the specified bucket.

```
aws s3api list-bucket-analytics-configurations \
  --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "AnalyticsConfigurationList": [
    {
      "StorageClassAnalysis": {},
      "Id": "1"
    }
  ],
  "IsTruncated": false
}
```

- For API details, see [ListBucketAnalyticsConfigurations](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns the first 100 analytics configurations of the given S3 bucket.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [ListBucketAnalyticsConfigurations](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns the first 100 analytics configurations of the given S3 bucket.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [ListBucketAnalyticsConfigurations](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use ListBucketInventoryConfigurations with a CLI

The following code examples show how to use ListBucketInventoryConfigurations.

CLI

Amazon CLI

To retrieve a list of inventory configurations for a bucket

The following list-bucket-inventory-configurations example lists the inventory configurations for the specified bucket.

```
aws s3api list-bucket-inventory-configurations \
  --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "InventoryConfigurationList": [
    {
      "IsEnabled": true,
      "Destination": {
        "S3BucketDestination": {
          "Format": "ORC",
          "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
          "AccountId": "123456789012"
        }
      },
      "IncludedObjectVersions": "Current",
      "Id": "1",
      "Schedule": {
```

```
        "Frequency": "Weekly"
      },
    ],
    {
      "IsEnabled": true,
      "Destination": {
        "S3BucketDestination": {
          "Format": "CSV",
          "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
          "AccountId": "123456789012"
        }
      },
      "IncludedObjectVersions": "Current",
      "Id": "2",
      "Schedule": {
        "Frequency": "Daily"
      }
    }
  ],
  "IsTruncated": false
}
```

- For API details, see [ListBucketInventoryConfigurations](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns the first 100 inventory configurations of the given S3 bucket.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [ListBucketInventoryConfigurations](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns the first 100 inventory configurations of the given S3 bucket.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- For API details, see [ListBucketInventoryConfigurations](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use ListBuckets with an Amazon SDK or CLI

The following code examples show how to use ListBuckets.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
    /// Service (Amazon S3) buckets belonging to the default account.
    /// </summary>
    public class ListBuckets
    {
        private static IAmazonS3 _s3Client;

        /// <summary>
        /// Get a list of the buckets owned by the default user.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
```

```

    /// <returns>The response from the ListingBuckets call that contains a
    /// list of the buckets owned by the default user.</returns>
    public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3
client)
    {
        return await client.ListBucketsAsync();
    }

    /// <summary>
    /// This method lists the name and creation date for the buckets in
    /// the passed List of S3 buckets.
    /// </summary>
    /// <param name="bucketList">A List of S3 bucket objects.</param>
    public static void DisplayBucketList(List<S3Bucket> bucketList)
    {
        bucketList
            .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
created on: {b.CreationDate}"));
    }

    public static async Task Main()
    {
        // The client uses the AWS Region of the default user.
        // If the Region where the buckets were created is different,
        // pass the Region to the client constructor. For example:
        // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
        _s3Client = new AmazonS3Client();
        var response = await GetBuckets(_s3Client);
        DisplayBucketList(response.Buckets);
    }
}

```

- For API details, see [ListBuckets](#) in *Amazon SDK for .NET API Reference*.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

- For API details, see [ListBuckets](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command uses the `list-buckets` command to display the names of all your Amazon S3 buckets (across all regions):


```
aws s3api list-buckets --query "Buckets[].Name"
```

The query option filters the output of `list-buckets` down to only the bucket names.

For more information about buckets, see *Working with Amazon S3 Buckets* in the *Amazon S3 Developer Guide*.

- For API details, see [ListBuckets](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.
```

```
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
error) {
    var err error
    var output *s3.ListBucketsOutput
    var buckets []types.Bucket
    bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
&s3.ListBucketsInput{})
    for bucketPaginator.HasMorePages() {
        output, err = bucketPaginator.NextPage(ctx)
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
                fmt.Println("You don't have permission to list buckets for this account.")
                err = apiErr
            } else {
                log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
            }
            break
        } else {
            buckets = append(buckets, output.Buckets...)
        }
    }
    return buckets, err
}
```

- For API details, see [ListBuckets](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListBucketsIterable;
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);
    }

    /**
     * Lists all the S3 buckets available in the current AWS account.
     *
     * @param s3 The {@link S3Client} instance to use for interacting with the
     * Amazon S3 service.
     */
    public static void listAllBuckets(S3Client s3) {
        ListBucketsIterable response = s3.listBucketsPaginator();
        response.buckets().forEach(bucket ->
```

```
        System.out.println("Bucket Name: " + bucket.name());  
    }
```

- For API details, see [ListBuckets](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

List the buckets.

```
import {  
    paginateListBuckets,  
    S3Client,  
    S3ServiceException,  
} from "@aws-sdk/client-s3";  
  
/**  
 * List the Amazon S3 buckets in your account.  
 */  
export const main = async () => {  
    const client = new S3Client({});  
    /** @type {?import('@aws-sdk/client-s3').Owner} */  
    let Owner = null;  
  
    /** @type {import('@aws-sdk/client-s3').Bucket[]} */  
    const Buckets = [];  
  
    try {  
        const paginator = paginateListBuckets({ client }, {});  
  
        for await (const page of paginator) {  
            if (!Owner) {  
                Owner = page.Owner;  
            }  
        }  
    }  
}
```

```
Buckets.push(...page.Buckets);
}

console.log(
  `${Owner.DisplayName} owns ${Buckets.length} bucket${
    Buckets.length === 1 ? "" : "s"
  }:`,
);
console.log(`${Buckets.map((b) => ` • ${b.Name}`).join("\n")}`);
} catch (caught) {
  if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while listing buckets.  ${caught.name}:
${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).
- For API details, see [ListBuckets](#) in *Amazon SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command returns all S3 buckets.

```
Get-S3Bucket
```

Example 2: This command returns bucket named "test-files"

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- For API details, see [ListBuckets](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command returns all S3 buckets.

```
Get-S3Bucket
```

Example 2: This command returns bucket named "test-files"

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- For API details, see [ListBuckets](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    @staticmethod
    def list(s3_resource):
        """
        Get the buckets in all Regions for the current account.
```

```

        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
Boto3
        that contains collections and factory methods to
create
        other high-level S3 sub-resources.
    :return: The list of buckets.
    """
    try:
        buckets = list(s3_resource.buckets.all())
        logger.info("Got buckets: %s.", buckets)
    except ClientError:
        logger.exception("Couldn't get buckets.")
        raise
    else:
        return buckets

```

- For API details, see [ListBuckets](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

require 'aws-sdk-s3'

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.

```

```

#
# @param count [Integer] The maximum number of buckets to list.
def list_buckets(count)
  puts 'Found these buckets:'
  @s3_resource.buckets.each do |bucket|
    puts "\t#{bucket.name}"
    count -= 1
    break if count.zero?
  end
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list buckets. Here's why: #{e.message}"
  false
end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__

```

- For API details, see [ListBuckets](#) in *Amazon SDK for Ruby API Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

async fn show_buckets(
  strict: bool,
  client: &Client,
  region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {

```



```

let mut buckets = client.list_buckets().into_paginator().send();

let mut num_buckets = 0;
let mut in_region = 0;

while let Some(Ok(output)) = buckets.next().await {
    for bucket in output.buckets() {
        num_buckets += 1;
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint() == Some(&region) {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }
}

println!();
if strict {
    println!(
        "Found {} buckets in the {} region out of a total of {} buckets.",
        in_region, region, num_buckets
    );
} else {
    println!("Found {} buckets in all regions.", num_buckets);
}

Ok(())
}

```

- For API details, see [ListBuckets](#) in *Amazon SDK for Rust API reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
    oo_result = lo_s3->listbuckets(           " oo_result is returned for  
testing purposes. "  
    ).  
    DATA(lv_bucket_count) = lines( oo_result->get_buckets( ) ).  
    MESSAGE |Retrieved { lv_bucket_count } buckets in all regions.| TYPE 'I'.  
CATCH /aws1/cx_rt_generic.  
    MESSAGE 'Unable to list buckets.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [ListBuckets](#) in *Amazon SDK for SAP ABAP API reference*.

Swift

SDK for Swift

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import AWSS3  
  
/// Return an array containing information about every available bucket.  
///  
/// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing  
/// each bucket.  
public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {
```

```
    return try await client.listBuckets(input: ListBucketsInput())  
  }
```

- For API details, see [ListBuckets](#) in *Amazon SDK for Swift API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use ListMultipartUploads with an Amazon SDK or CLI

The following code examples show how to use ListMultipartUploads.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Delete incomplete multipart uploads](#)

CLI

Amazon CLI

The following command lists all of the active multipart uploads for a bucket named `amzn-s3-demo-bucket`:

```
aws s3api list-multipart-uploads --bucket amzn-s3-demo-bucket
```

Output:

```
{  
  "Uploads": [  
    {  
      "Initiator": {  
        "DisplayName": "username",  
        "ID": "arn:aws:iam::0123456789012:user/username"  
      },  
      "Initiated": "2015-06-02T18:01:30.000Z",  
      "UploadId":  
        "dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3",  
      "StorageClass": "STANDARD",  
    }  
  ]  
}
```

```
        "Key": "multipart/01",
        "Owner": {
            "DisplayName": "aws-account-name",
            "ID":
"100719349fc3b6dcd7c820a124bf7aecd408092c3d7b51b38494939801fc248b"
        }
    },
    "CommonPrefixes": []
}
```

In progress multipart uploads incur storage costs in Amazon S3. Complete or abort an active multipart upload to remove its parts from your account.

- For API details, see [ListMultipartUploads](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
```

```
*/

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the multipart uploads currently in progress in the specified Amazon
S3 bucket.
     *
     * @param s3 the S3Client object used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket to list the multipart
uploads for
     */
    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();
```

```
        ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            System.out.println("Upload in progress: Key = \"" + upload.key()
+ "\", id = " + upload.uploadId());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListMultipartUploads](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use ListObjectVersions with an Amazon SDK or CLI

The following code examples show how to use ListObjectVersions.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Getting started with Amazon S3](#)
- [Work with versioned objects](#)

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example lists the versions of the objects in a version enabled
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class ListObjectVersions
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region where your bucket is defined is different from
        // the AWS Region where the Amazon S3 bucket is defined, pass the
constant
        // for the AWS Region to the client constructor like this:
        //     var client = new AmazonS3Client(RegionEndpoint.USSouth1);
        IAmazonS3 client = new AmazonS3Client();
        await GetObjectListWithAllVersionsAsync(client, bucketName);
    }

    /// <summary>
    /// This method lists all versions of the objects within an Amazon S3
    /// version enabled bucket.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// ListVersionsAsync.</param>
    /// <param name="bucketName">The name of the version enabled Amazon S3
bucket
    /// for which you want to list the versions of the contained objects.</
param>
    public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
client, string bucketName)
    {
        try
        {
            // When you instantiate the ListVersionRequest, you can
            // optionally specify a key name prefix in the request
            // if you want a list of object versions of a specific object.

```

```
// For this example we set a small limit in MaxKeys to return
// a small list of versions.
ListVersionsRequest request = new ListVersionsRequest()
{
    BucketName = bucketName,
    MaxKeys = 2,
};

do
{
    ListVersionsResponse response = await
client.ListVersionsAsync(request);

    // Process response.
    foreach (S3ObjectVersion entry in response.Versions)
    {
        Console.WriteLine($"key: {entry.Key} size:
{entry.Size}");
    }

    // If response is truncated, set the marker to get the next
    // set of keys.
    if (response.IsTruncated)
    {
        request.KeyMarker = response.NextKeyMarker;
        request.VersionIdMarker = response.NextVersionIdMarker;
    }
    else
    {
        request = null;
    }
} while (request != null);
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error: '{ex.Message}'");
}
}
```

- For API details, see [ListObjectVersions](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

The following command retrieves version information for an object in a bucket named `amzn-s3-demo-bucket`:

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix index.html
```

Output:

```
{
  "DeleteMarkers": [
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": true,
      "VersionId": "B2VsEK5saUNNHKc0AJj7hIE86RozToyq",
      "Key": "index.html",
      "LastModified": "2015-11-10T00:57:03.000Z"
    },
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
      "Key": "index.html",
      "LastModified": "2015-11-09T23:32:20.000Z"
    }
  ],
  "Versions": [
    {
      "LastModified": "2015-11-10T00:20:11.000Z",
      "VersionId": "Rb_l2T8UHDkFEwCgJjhlGPOZC0qJ.vpD",
      "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
      "StorageClass": "STANDARD",
      "Key": "index.html",
    }
  ]
}
```

```

        "Owner": {
            "DisplayName": "my-username",
            "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
        },
        "IsLatest": false,
        "Size": 38
    },
    {
        "LastModified": "2015-11-09T23:26:41.000Z",
        "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAff",
        "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
        "StorageClass": "STANDARD",
        "Key": "index.html",
        "Owner": {
            "DisplayName": "my-username",
            "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
        },
        "IsLatest": false,
        "Size": 38
    },
    {
        "LastModified": "2015-11-09T22:50:50.000Z",
        "VersionId": "null",
        "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
        "StorageClass": "STANDARD",
        "Key": "index.html",
        "Owner": {
            "DisplayName": "my-username",
            "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
        },
        "IsLatest": false,
        "Size": 533823
    }
}

```

- For API details, see [ListObjectVersions](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
    ([]types.ObjectVersion, error) {
    var err error
    var output *s3.ListObjectVersionsOutput
    var versions []types.ObjectVersion
    input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
    versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
```

```

for versionPaginator.HasMorePages() {
    output, err = versionPaginator.NextPage(ctx)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
        break
    } else {
        versions = append(versions, output.Versions...)
    }
}
return versions, err
}

```

- For API details, see [ListObjectVersions](#) in *Amazon SDK for Go API Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions() {
        println!("{}", version.key().unwrap_or_default());
        println!(" version ID: {}", version.version_id().unwrap_or_default());
        println!();
    }

    Ok(())
}

```

- For API details, see [ListObjectVersions](#) in *Amazon SDK for Rust API reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
    oo_result = lo_s3->listobjectversions(           " oo_result is returned  
for testing purposes. "  
        iv_bucket = iv_bucket_name  
        iv_prefix = iv_prefix ).  
    MESSAGE 'Retrieved object versions.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [ListObjectVersions](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use ListObjects with a CLI

The following code examples show how to use ListObjects.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Create a web page that lists Amazon S3 objects](#)

CLI

Amazon CLI

The following example uses the `list-objects` command to display the names of all the objects in the specified bucket:

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

The example uses the `--query` argument to filter the output of `list-objects` down to the `key` value and `size` for each object.

For more information about objects, see *Working with Amazon S3 Objects* in the *Amazon S3 Developer Guide*.

- For API details, see [ListObjects](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command retrieves the information about all of the items in the bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Example 2: This command retrieves the information about the item "sample.txt" from bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Example 3: This command retrieves the information about all items with the prefix "sample" from bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- For API details, see [ListObjects](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command retrieves the information about all of the items in the bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Example 2: This command retrieves the information about the item "sample.txt" from bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Example 3: This command retrieves the information about all items with the prefix "sample" from bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- For API details, see [ListObjects](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use ListObjectsV2 with an Amazon SDK or CLI

The following code examples show how to use ListObjectsV2.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Learn the basics](#)
- [Delete all objects in a bucket](#)
- [Getting started with Amazon S3](#)

.NET

Amazon SDK for .NET (v4)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Shows how to list the objects in an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket for which to list.
/// <param name="printList">True to print out the list.
/// <returns>The collection of objects.</returns>
public async Task<List<S3Object?>> ListBucketContentsAsync(string bucketName,
bool printList = true)
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 5,
        };

        if (printList)
        {
            Console.WriteLine("-----");
            Console.WriteLine($"Listing the contents of {bucketName}:");
            Console.WriteLine("-----");
        }

        var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
        {
            BucketName = bucketName,
        });
        var s3Objects = new List<S3Object>();
        await foreach (var response in listObjectsV2Paginator.Responses)
```



```
        {
            if (response.S3Objects != null)
            {
                s3objects.AddRange(response.S3Objects);
            }
        }

        if (printList)
        {
            Console.WriteLine($"Number of Objects: {s3objects.Count}");
            foreach (var entry in s3objects)
            {
                Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
            }
        }

        return s3objects;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error encountered on server.
        Message: '{ex.Message}' getting list of objects.");
        return null;
    }
}
```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for .NET API Reference*.

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

List objects with a paginator.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
```

```
using Amazon.S3.Model;

/// <summary>
/// The following example lists objects in an Amazon Simple Storage
/// Service (Amazon S3) bucket.
/// </summary>
public class ListObjectsPaginator
{
    private const string BucketName = "amzn-s3-demo-bucket";

    public static async Task Main()
    {
        IAmazonS3 s3Client = new AmazonS3Client();

        Console.WriteLine($"Listing the objects contained in {BucketName}:
\n");

        await ListingObjectsAsync(s3Client, BucketName);
    }

    /// <summary>
    /// This method uses a paginator to retrieve the list of objects in an
    /// an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">An Amazon S3 client object.</param>
    /// <param name="bucketName">The name of the S3 bucket whose objects
    /// you want to list.</param>
    public static async Task ListingObjectsAsync(IAmazonS3 client, string
bucketName)
    {
        var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
ListObjectsV2Request
        {
            BucketName = bucketName,
        });

        await foreach (var response in listObjectsV2Paginator.Responses)
        {
            Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
            Console.WriteLine($"Number of Keys: {response.KeyCount}");
            foreach (var entry in response.S3Objects)
            {
                Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
            }
        }
    }
}
```

```
}
}
```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for .NET API Reference*.

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     The list of files in text format.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
```

```
function list_items_in_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api list-objects \
        --bucket "$bucket_name" \
        --output text \
        --query 'Contents[].{Key: Key, Size: Size}')

    # shellcheck disable=SC2181
    if [[ ${?} -eq 0 ]]; then
        echo "$response"
    else
        errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
        return 1
    fi
}
```

- For API details, see [ListObjectsV2](#) in *Amazon CLI Command Reference*.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;
```

```

do {
    if (!continuationToken.empty()) {
        request.SetContinuationToken(continuationToken);
    }

    auto outcome = s3Client.ListObjectsV2(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: listObjects: " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    } else {
        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();

        allObjects.insert(allObjects.end(), objects.begin(), objects.end());
        continuationToken = outcome.GetResult().GetNextContinuationToken();
    }
} while (!continuationToken.empty());

std::cout << allObjects.size() << " object(s) found:" << std::endl;

for (const auto &object: allObjects) {
    std::cout << "  " << object.GetKey() << std::endl;
    keysResult.push_back(object.GetKey());
}

return true;
}

```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

To get a list of objects in a bucket

The following `list-objects-v2` example lists the objects in the specified bucket.

```
aws s3api list-objects-v2 \
```

```
--bucket amzn-s3-demo-bucket
```

Output:

```
{
  "Contents": [
    {
      "LastModified": "2019-11-05T23:11:50.000Z",
      "ETag": "\"621503c373607d548b37cff8778d992c\"",
      "StorageClass": "STANDARD",
      "Key": "doc1.rtf",
      "Size": 391
    },
    {
      "LastModified": "2019-11-05T23:11:50.000Z",
      "ETag": "\"a2cecc36ab7c7fe3a71a273b9d45b1b5\"",
      "StorageClass": "STANDARD",
      "Key": "doc2.rtf",
      "Size": 373
    },
    {
      "LastModified": "2019-11-05T23:11:50.000Z",
      "ETag": "\"08210852f65a2e9cb999972539a64d68\"",
      "StorageClass": "STANDARD",
      "Key": "doc3.rtf",
      "Size": 399
    },
    {
      "LastModified": "2019-11-05T23:11:50.000Z",
      "ETag": "\"d1852dd683f404306569471af106988e\"",
      "StorageClass": "STANDARD",
      "Key": "doc4.rtf",
      "Size": 6225
    }
  ]
}
```

- For API details, see [ListObjectsV2](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}  
  
// ListObjects lists the objects in a bucket.
```

```
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
([]types.Object, error) {
    var err error
    var output *s3.ListObjectsV2Output
    input := &s3.ListObjectsV2Input{
        Bucket: aws.String(bucketName),
    }
    var objects []types.Object
    objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
    for objectPaginator.HasMorePages() {
        output, err = objectPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
            break
        } else {
            objects = append(objects, output.Contents...)
        }
    }
    return objects, err
}
```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/**
 * Asynchronously lists all objects in the specified S3 bucket.
```



```

    *
    * @param bucketName the name of the S3 bucket to list objects for
    * @return a {@link CompletableFuture} that completes when all objects have
    been listed
    */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3object -> {
                logger.info("Object key: " + s3object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }
}

```

List objects using pagination.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

        Where:
    
```

```
        bucketName - The Amazon S3 bucket from which objects are read.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listBucketObjects(s3, bucketName);
    s3.close();
}

/**
 * Lists the objects in the specified S3 bucket.
 *
 * @param s3 the S3Client instance used to interact with Amazon S3
 * @param bucketName the name of the S3 bucket to list the objects from
 */
public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
        listRes.stream()
            .flatMap(r -> r.contents().stream())
            .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

List all of the objects in your bucket. If there is more than one object, `IsTruncated` and `NextContinuationToken` will be used to iterate over the full list.

```
import {
  S3Client,
  S3ServiceException,
  // This command supersedes the ListObjectsCommand and is the recommended way to
  list objects.
  paginateListObjectsV2,
} from "@aws-sdk/client-s3";

/**
 * Log all of the object keys in a bucket.
 * @param {{ bucketName: string, pageSize: string }}
 */
export const main = async ({ bucketName, pageSize }) => {
  const client = new S3Client({});
  /** @type {string[][]} */
  const objects = [];
  try {
    const paginator = paginateListObjectsV2(
      { client, /* Max items per page */ pageSize: Number.parseInt(pageSize) },
      { Bucket: bucketName },
    );

    for await (const page of paginator) {
      objects.push(page.Contents.map((o) => o.Key));
    }
  }
}
```

```

    }
    objects.forEach((objectList, pageNum) => {
        console.log(
            `Page ${pageNum + 1}\n-----\n${objectList.map((o) => `•
${o}`)}.join("\n")}\n`,
        );
    });
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while listing objects for "${bucketName}". The bucket
doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while listing objects for "${bucketName}".
${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

suspend fun listBucketObjects(bucketName: String) {
    val request =

```

```
ListObjectsRequest {
    bucket = bucketName
}

S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
    val response = s3.listObjects(request)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The object is ${myObject.size?.let { calcKb(it) }} KBs")
        println("The owner is ${myObject.owner}")
    }
}

private fun calcKb(intValue: Long): Long = intValue / 1024
```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for Kotlin API reference*.

PHP

SDK for PHP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

List objects in a bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $contents = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
```

```

        echo "Failed to list objects in ${this->bucketName} with error: " .
        $exception->getMessage();
        exit("Please fix error with listing objects before continuing.");
    }

```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for PHP API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                                that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.

        :param bucket: The bucket to query. This is a Boto3 Bucket resource.
        :param prefix: When specified, only objects that start with this prefix
are listed.
        :return: The list of objects.
        """
        try:

```

```
        if not prefix:
            objects = list(bucket.objects.all())
        else:
            objects = list(bucket.objects.filter(Prefix=prefix))
        logger.info(
            "Got objects %s from bucket '%s'", [o.key for o in objects],
            bucket.name
        )
    except ClientError:
        logger.exception("Couldn't get objects for bucket '%s'.",
            bucket.name)
        raise
    else:
        return objects
```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
```

```

# @param max_objects [Integer] The maximum number of objects to list.
# @return [Integer] The number of objects listed.
def list_objects(max_objects)
  count = 0
  puts "The objects in #{@bucket.name} are:"
  @bucket.objects.each do |obj|
    puts "\t#{obj.key}"
    count += 1
    break if count == max_objects
  end
  count
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list objects in bucket #{bucket.name}. Here's why:
#{e.message}"
  0
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for Ruby API Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).


```
pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
    Result<(), S3ExampleError> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}
```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for Rust API reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

TRY.

```
oo_result = lo_s3->listobjectsv2(           " oo_result is returned for
testing purposes. "
```

```

        iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for SAP ABAP API reference*.

Swift

SDK for Swift

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

import AWSS3

public func listBucketFiles(bucket: String) async throws -> [String] {
    do {
        let input = ListObjectsV2Input(
            bucket: bucket
        )

        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'continuationToken' in
        "ListObjectsV2Output".
        let output = client.listObjectsV2Paginated(input: input)
        var names: [String] = []

        for try await page in output {
            guard let objList = page.contents else {
                print("ERROR: listObjectsV2Paginated returned nil contents.")
                continue
            }

            for obj in objList {
                if let objName = obj.key {
                    names.append(objName)
                }
            }
        }
    }
}

```

```
        }
    }
}

return names
}
catch {
    print("ERROR: ", dump(error, name: "Listing objects."))
    throw error
}
}
```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for Swift API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketAccelerateConfiguration with an Amazon SDK or CLI

The following code examples show how to use PutBucketAccelerateConfiguration.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
```

```

/// bucket-level feature that enables you to perform faster data transfers
/// to Amazon S3. This example shows how to configure Transfer
/// Acceleration.
/// </summary>
public class TransferAcceleration
{
    /// <summary>
    /// The main method initializes the client object and sets the
    /// Amazon Simple Storage Service (Amazon S3) bucket name before
    /// calling EnableAccelerationAsync.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        const string bucketName = "amzn-s3-demo-bucket";

        await EnableAccelerationAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method sets the configuration to enable transfer acceleration
    /// for the bucket referred to in the bucketName parameter.
    /// </summary>
    /// <param name="client">An Amazon S3 client used to enable the
    /// acceleration on an Amazon S3 bucket.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket for which
the
    /// method will be enabling acceleration.</param>
    private static async Task EnableAccelerationAsync(AmazonS3Client client,
string bucketName)
    {
        try
        {
            var putRequest = new PutBucketAccelerateConfigurationRequest
            {
                BucketName = bucketName,
                AccelerateConfiguration = new AccelerateConfiguration
                {
                    Status = BucketAccelerateStatus.Enabled,
                },
            };
            await client.PutBucketAccelerateConfigurationAsync(putRequest);

            var getRequest = new GetBucketAccelerateConfigurationRequest

```

```
        {
            BucketName = bucketName,
        };
        var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

        Console.WriteLine($"Acceleration state = '{response.Status}' ");
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error occurred. Message:'{ex.Message}' when
setting transfer acceleration");
    }
}
```

- For API details, see [PutBucketAccelerateConfiguration](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To set the accelerate configuration of a bucket

The following `put-bucket-accelerate-configuration` example enables the accelerate configuration for the specified bucket.

```
aws s3api put-bucket-accelerate-configuration \
    --bucket amzn-s3-demo-bucket \
    --accelerate-configuration Status=Enabled
```

This command produces no output.

- For API details, see [PutBucketAccelerateConfiguration](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command enables the transfer acceleration for the given S3 bucket.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- For API details, see [PutBucketAccelerateConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command enables the transfer acceleration for the given S3 bucket.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- For API details, see [PutBucketAccelerateConfiguration](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketAc1 with an Amazon SDK or CLI

The following code examples show how to use PutBucketAc1.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Manage access control lists \(ACLs\)](#)

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Creates an Amazon S3 bucket with an ACL to control access to the
/// bucket and the objects stored in it.
/// </summary>
/// <param name="client">The initialized client object used to create
/// an Amazon S3 bucket, with an ACL applied to the bucket.
/// </param>
/// <param name="region">The AWS Region where the bucket will be
created.</param>
/// <param name="newBucketName">The name of the bucket to create.</param>
/// <returns>A boolean value indicating success or failure.</returns>
public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
client, S3Region region, string newBucketName)
{
    try
    {
        // Create a new Amazon S3 bucket with Canned ACL.
        var putBucketRequest = new PutBucketRequest()
        {
            BucketName = newBucketName,
            BucketRegion = region,
            CannedACL = S3CannedACL.LogDeliveryWrite,
        };

        PutBucketResponse putBucketResponse = await
client.PutBucketAsync(putBucketRequest);

        return putBucketResponse.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
```

```

        {
            Console.WriteLine($"Amazon S3 error: {ex.Message}");
        }

        return false;
    }

```

- For API details, see [PutBucketAcl](#) in *Amazon SDK for .NET API Reference*.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
&ownerID,
                                const Aws::String &granteePermission,
                                const Aws::String &granteeType, const Aws::String
&granteeID,
                                const Aws::String &granteeEmailAddress,
                                const Aws::String &granteeURI, const
Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {

```



```

        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
        s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
            << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
            << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
enumeration.
/*!
    \param access: Human readable string.
    \return Permission: A Permission enum.
*/

```

```

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
    enumeration.
/*!
    \param type: Human readable string.
    \return Type: Type enumeration
*/

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}

```

- For API details, see [PutBucketAcl](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

This example grants `full control` to two Amazon users (`user1@example.com` and `user2@example.com`) and read permission to everyone:

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --grant-full-  
control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-  
read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

See <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> for details on custom ACLs (the s3api ACL commands, such as `put-bucket-acl`, use the same shorthand argument notation).

- For API details, see [PutBucketAcl](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;  
import software.amazon.awssdk.services.s3.model.Grant;  
import software.amazon.awssdk.services.s3.model.Permission;  
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.Type;  
  
import java.util.ArrayList;  
import java.util.List;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 * <p>  
 * For more information, see the following documentation topic:  
 * <p>  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */
```

```

*/
public class SetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <id>\s

            Where:
                bucketName - The Amazon S3 bucket to grant permissions on.\s
                id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    /**
     * Sets the Access Control List (ACL) for an Amazon S3 bucket.
     *
     * @param s3 the S3Client instance to be used for the operation
     * @param bucketName the name of the S3 bucket to set the ACL for
     * @param id the ID of the AWS user or account that will be granted full
control of the bucket
     * @throws S3Exception if an error occurs while setting the bucket ACL
     */
    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {

```

```
Grant ownerGrant = Grant.builder()
    .grantee(builder -> builder.id(id)
        .type(Type.CANONICAL_USER))
    .permission(Permission.FULL_CONTROL)
    .build();

List<Grant> grantList2 = new ArrayList<>();
grantList2.add(ownerGrant);

AccessControlPolicy acl = AccessControlPolicy.builder()
    .owner(builder -> builder.id(id))
    .grants(grantList2)
    .build();

PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
    .bucket(bucketName)
    .accessControlPolicy(acl)
    .build();

s3.putBucketAcl(putAclReq);

} catch (S3Exception e) {
    e.printStackTrace();
    System.exit(1);
}
}
```

- For API details, see [PutBucketAcl](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Put the bucket ACL.

```

import {
    PutBucketAclCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant read access to a user using their canonical AWS account ID.
 *
 * Most Amazon S3 use cases don't require the use of access control lists (ACLs).
 * We recommend that you disable ACLs, except in unusual circumstances where
 * you need to control access for each object individually. Consider a policy
 * instead.
 * For more information see https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-policies.html.
 * @param {{ bucketName: string, granteeCanonicalUserId: string,
 *   ownerCanonicalUserId }}
 */
export const main = async ({
    bucketName,
    granteeCanonicalUserId,
    ownerCanonicalUserId,
}) => {
    const client = new S3Client({});
    const command = new PutBucketAclCommand({
        Bucket: bucketName,
        AccessControlPolicy: {
            Grants: [
                {
                    Grantee: {
                        // The canonical ID of the user. This ID is an obfuscated form of
                        // your AWS account number.
                        // It's unique to Amazon S3 and can't be found elsewhere.
                        // For more information, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/finding-canonical-user-id.html.
                        ID: granteeCanonicalUserId,
                        Type: "CanonicalUser",
                    },
                    // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
                    // https://docs.aws.amazon.com/AmazonS3/latest/API/API\_Grant.html#AmazonS3-Type-Grant-Permission
                    Permission: "READ",
                },
            ],
        },
    });

```

```
    ],
    Owner: {
      ID: ownerCanonicalUserId,
    },
  },
});

try {
  await client.send(command);
  console.log(`Granted READ access to ${bucketName}`);
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while setting ACL for bucket ${bucketName}. The bucket
doesn't exist.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while setting ACL for bucket ${bucketName}.
${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).
- For API details, see [PutBucketAcl](#) in *Amazon SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
```



```

        bucket = bucketName
        accessControlPolicy = acl
    }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}

```

- For API details, see [PutBucketAcl](#) in *Amazon SDK for Kotlin API reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def grant_log_delivery_access(self):
        """
        Grant the AWS Log Delivery group write access to the bucket so that
        Amazon S3 can deliver access logs to the bucket. This is the only
        recommended
        use of an S3 bucket ACL.
        """

```

```

"""
try:
    acl = self.bucket.Acl()
    # Putting an ACL overwrites the existing ACL. If you want to preserve
    # existing grants, append new grants to the list of existing grants.
    grants = acl.grants if acl.grants else []
    grants.append(
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
            },
            "Permission": "WRITE",
        }
    )
    acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
    logger.info("Granted log delivery access to bucket '%s'",
self.bucket.name)
except ClientError:
    logger.exception("Couldn't add ACL to bucket '%s'.",
self.bucket.name)
    raise

```

- For API details, see [PutBucketAcl](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

TRY.

```

" Example: Grant log delivery access to a bucket
" iv_grantwrite = 'uri=http://acs.amazonaws.com/groups/s3/LogDelivery'
lo_s3->putbucketacl(
    iv_bucket = iv_bucket_name

```

```

        iv_grantwrite = iv_grantwrite ).
    MESSAGE 'Bucket ACL updated.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- For API details, see [PutBucketAcl](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketCors with an Amazon SDK or CLI

The following code examples show how to use PutBucketCors.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

    /// <summary>
    /// Add CORS configuration to the Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used
    /// to apply the CORS configuration to an Amazon S3 bucket.</param>
    /// <param name="configuration">The CORS configuration to apply.</param>
    private static async Task PutCORSConfigurationAsync(AmazonS3Client
client, CORSConfiguration configuration)
    {
        PutCORSConfigurationRequest request = new
PutCORSConfigurationRequest()
        {
            BucketName = BucketName,

```

```
        Configuration = configuration,
    };

    _ = await client.PutCORSConfigurationAsync(request);
}
```

- For API details, see [PutBucketCors](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

The following example enables PUT, POST, and DELETE requests from *www.example.com*, and enables GET requests from any domain:

```
aws s3api put-bucket-cors --bucket amzn-s3-demo-bucket --cors-configuration file://cors.json
```

cors.json:

```
{
  "CORSRules": [
    {
      "AllowedOrigins": ["http://www.example.com"],
      "AllowedHeaders": ["*"],
      "AllowedMethods": ["PUT", "POST", "DELETE"],
      "MaxAgeSeconds": 3000,
      "ExposeHeaders": ["x-amz-server-side-encryption"]
    },
    {
      "AllowedOrigins": ["*"],
      "AllowedHeaders": ["Authorization"],
      "AllowedMethods": ["GET"],
      "MaxAgeSeconds": 3000
    }
  ]
}
```

- For API details, see [PutBucketCors](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

import java.util.ArrayList;
import java.util.List;

import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
            <bucketName> <accountId>\s

        Where:
```

```

        bucketName - The Amazon S3 bucket to upload an object into.
        accountId - The id of the account that owns the Amazon S3 bucket.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String accountId = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setCorsInformation(s3, bucketName, accountId);
    getBucketCorsInformation(s3, bucketName, accountId);
    deleteBucketCorsInformation(s3, bucketName, accountId);
    s3.close();
}

/**
 * Deletes the CORS (Cross-Origin Resource Sharing) configuration for an
 * Amazon S3 bucket.
 *
 * @param s3          the {@link S3Client} instance used to interact with
 * the Amazon S3 service
 * @param bucketName  the name of the Amazon S3 bucket for which the CORS
 * configuration should be deleted
 * @param accountId   the expected AWS account ID of the bucket owner
 *
 * @throws S3Exception if an error occurs while deleting the CORS
 * configuration for the bucket
 */
public static void deleteBucketCorsInformation(S3Client s3, String
bucketName, String accountId) {
    try {
        DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)
            .build();
    }
}

```

```

        s3.deleteBucketCors(bucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**
 * Retrieves the CORS (Cross-Origin Resource Sharing) configuration for the
 * specified S3 bucket.
 *
 * @param s3 the S3Client instance to use for the operation
 * @param bucketName the name of the S3 bucket to retrieve the CORS
 * configuration for
 * @param accountId the expected bucket owner's account ID
 *
 * @throws S3Exception if there is an error retrieving the CORS configuration
 */
public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
    try {
        GetBucketCorsRequest bucketCorsRequest =
GetBucketCorsRequest.builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)
            .build();

        GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
        List<CORSRule> corsRules = corsResponse.corsRules();
        for (CORSRule rule : corsRules) {
            System.out.println("allowOrigins: " + rule.allowedOrigins());
            System.out.println("AllowedMethod: " + rule.allowedMethods());
        }

    } catch (S3Exception e) {

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**

```

```
* Sets the Cross-Origin Resource Sharing (CORS) rules for an Amazon S3
bucket.
*
* @param s3 The S3Client object used to interact with the Amazon S3 service.
* @param bucketName The name of the S3 bucket to set the CORS rules for.
* @param accountId The AWS account ID of the bucket owner.
*/
public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
    List<String> allowMethods = new ArrayList<>();
    allowMethods.add("PUT");
    allowMethods.add("POST");
    allowMethods.add("DELETE");

    List<String> allowOrigins = new ArrayList<>();
    allowOrigins.add("http://example.com");
    try {
        // Define CORS rules.
        CORSRule corsRule = CORSRule.builder()
            .allowedMethods(allowMethods)
            .allowedOrigins(allowOrigins)
            .build();

        List<CORSRule> corsRules = new ArrayList<>();
        corsRules.add(corsRule);
        CORSConfiguration configuration = CORSConfiguration.builder()
            .corsRules(corsRules)
            .build();

        PutBucketCorsRequest putBucketCorsRequest =
PutBucketCorsRequest.builder()
            .bucket(bucketName)
            .corsConfiguration(configuration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketCors(putBucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```


- For API details, see [PutBucketCors](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Add a CORS rule.

```
import {
    PutBucketCorsCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Allows cross-origin requests to an S3 bucket by setting the CORS
 * configuration.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
    const client = new S3Client({});

    try {
        await client.send(
            new PutBucketCorsCommand({
                Bucket: bucketName,
                CORSConfiguration: {
                    CORSRules: [
                        {
                            // Allow all headers to be sent to this bucket.
                            AllowedHeaders: ["*"],
                            // Allow only GET and PUT methods to be sent to this bucket.
                            AllowedMethods: ["GET", "PUT"],
                            // Allow only requests from the specified origin.
```

```

        AllowedOrigins: ["https://www.example.com"],
        // Allow the entity tag (ETag) header to be returned in the
response. The ETag header
        // The entity tag represents a specific version of the object. The
ETag reflects
        // changes only to the contents of an object, not its metadata.
ExposeHeaders: ["ETag"],
        // How long the requesting browser should cache the preflight
response. After
        // this time, the preflight request will have to be made again.
MaxAgeSeconds: 3600,
    },
],
},
)),
);
console.log(`Successfully set CORS rules for bucket: ${bucketName}`);
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while setting CORS rules for ${bucketName}. The bucket
doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while setting CORS rules for ${bucketName}.
${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).
- For API details, see [PutBucketCors](#) in *Amazon SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_cors(self, cors_rules):
        """
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that
        are
        allowed from other domains.

        :param cors_rules: The CORS rules to apply.
        """
        try:
            self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
            logger.info(
                "Put CORS rules %s for bucket '%s'.", cors_rules,
                self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't put CORS rules for bucket %s.",
                self.bucket.name)
            raise
```

- For API details, see [PutBucketCors](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  # an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
  end
end
```

```
    }  
  )  
  true  
rescue Aws::Errors::ServiceError => e  
  puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:  
#{e.message}"  
  false  
end  
  
end
```

- For API details, see [PutBucketCors](#) in *Amazon SDK for Ruby API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
  " Example: Allow PUT, POST, DELETE methods from http://www.example.com  
  lo_s3->putbucketcors(  
    iv_bucket = iv_bucket_name  
    io_corsconfiguration = NEW /aws1/cl_s3_corsconfiguration(  
      it_corsrules = it_cors_rules ) ).  
  MESSAGE 'Bucket CORS configuration set.' TYPE 'I'.  
  CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [PutBucketCors](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketEncryption with an Amazon SDK or CLI

The following code examples show how to use PutBucketEncryption.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Getting started with Amazon S3](#)

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Set the bucket server side encryption to use AWSKMS with a customer-
managed key id.
/// </summary>
/// <param name="bucketName">Name of the bucket.</param>
/// <param name="kmsKeyId">The Id of the KMS Key.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> SetBucketServerSideEncryption(string
bucketName, string kmsKeyId)
{
    var serverSideEncryptionByDefault = new ServerSideEncryptionConfiguration
    {
        ServerSideEncryptionRules = new List<ServerSideEncryptionRule>
        {
            new ServerSideEncryptionRule
            {
                ServerSideEncryptionByDefault = new
ServerSideEncryptionByDefault
```

```

        {
            ServerSideEncryptionAlgorithm =
ServerSideEncryptionMethod.AWSKMS,
            ServerSideEncryptionKeyManagementServiceKeyId = kmsKeyId
        }
    }
};
try
{
    var encryptionResponse = await _s3Client.PutBucketEncryptionAsync(new
PutBucketEncryptionRequest
    {
        BucketName = bucketName,
        ServerSideEncryptionConfiguration =
serverSideEncryptionByDefault,
    });

    return encryptionResponse.HttpStatusCode == HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine(ex.ErrorCode == "AccessDenied"
        ? $"This account does not have permission to set encryption on
{bucketName}, please try again."
        : $"Unable to set bucket encryption for bucket {bucketName},
{ex.Message}");
}
return false;
}

```

- For API details, see [PutBucketEncryption](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To configure server-side encryption for a bucket

The following `put-bucket-encryption` example sets AES256 encryption as the default for the specified bucket.

```
aws s3api put-bucket-encryption \  
  --bucket amzn-s3-demo-bucket \  
  --server-side-encryption-configuration '{"Rules":  
  [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

This command produces no output.

- For API details, see [PutBucketEncryption](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command enables default AES256 server side encryption with Amazon S3 Managed Keys(SSE-S3) on the given bucket.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
  @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- For API details, see [PutBucketEncryption](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command enables default AES256 server side encryption with Amazon S3 Managed Keys(SSE-S3) on the given bucket.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
  @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- For API details, see [PutBucketEncryption](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketLifecycleConfiguration with an Amazon SDK or CLI

The following code examples show how to use PutBucketLifecycleConfiguration.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Delete incomplete multipart uploads](#)
- [Manage large messages using S3](#)
- [Work with versioned objects](#)

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Adds lifecycle configuration information to the S3 bucket named in
/// the bucketName parameter.
/// </summary>
/// <param name="client">The S3 client used to call the
/// PutLifecycleConfigurationAsync method.</param>
/// <param name="bucketName">A string representing the S3 bucket to
/// which configuration information will be added.</param>
/// <param name="configuration">A LifecycleConfiguration object that
/// will be applied to the S3 bucket.</param>
public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
string bucketName, LifecycleConfiguration configuration)
{
    var request = new PutLifecycleConfigurationRequest()
    {
        BucketName = bucketName,
        Configuration = configuration,
    };
    var response = await client.PutLifecycleConfigurationAsync(request);
```

```
}
```

- For API details, see [PutBucketLifecycleConfiguration](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

The following command applies a lifecycle configuration to a bucket named `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --  
lifecycle-configuration file://lifecycle.json
```

The file `lifecycle.json` is a JSON document in the current folder that specifies two rules:

```
{  
  "Rules": [  
    {  
      "ID": "Move rotated logs to Glacier",  
      "Prefix": "rotated/",  
      "Status": "Enabled",  
      "Transitions": [  
        {  
          "Date": "2015-11-10T00:00:00.000Z",  
          "StorageClass": "GLACIER"  
        }  
      ]  
    },  
    {  
      "Status": "Enabled",  
      "Prefix": "",  
      "NoncurrentVersionTransitions": [  
        {  
          "NoncurrentDays": 2,  
          "StorageClass": "GLACIER"  
        }  
      ],  
      "ID": "Move old versions to Glacier"  ]  
}
```

```
    }  
  ]  
}
```

The first rule moves files with the prefix `rotated` to Glacier on the specified date. The second rule moves old object versions to Glacier when they are no longer current. For information on acceptable timestamp formats, see *Specifying Parameter Values* in the *Amazon CLI User Guide*.

- For API details, see [PutBucketLifecycleConfiguration](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;  
import software.amazon.awssdk.services.s3.model.Transition;  
import  
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;  
import  
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;  
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;  
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;  
import software.amazon.awssdk.services.s3.model.LifecycleRule;  
import software.amazon.awssdk.services.s3.model.ExpirationStatus;  
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;  
import  
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import java.util.ArrayList;  
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and deleted a
Lifecycle configuration");
        s3.close();
    }

    /**
```

```

    * Sets the lifecycle configuration for an Amazon S3 bucket.
    *
    * @param s3          The Amazon S3 client to use for the operation.
    * @param bucketName The name of the Amazon S3 bucket.
    * @param accountId   The expected owner of the Amazon S3 bucket.
    *
    * @throws S3Exception if there is an error setting the lifecycle
configuration.
    */
    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the "glacierobjects/" prefix
to the
            // S3 Glacier Flexible Retrieval storage class immediately.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("Archive immediately rule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Create a second rule.
            Transition transition2 = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            List<Transition> transitionList = new ArrayList<>();
            transitionList.add(transition2);

            LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

```

```

        LifecycleRule rule2 = LifecycleRule.builder()
            .id("Archive and then delete rule")
            .filter(ruleFilter2)
            .transitions(transitionList)
            .status(ExpirationStatus.ENABLED)
            .build();

        // Add the LifecycleRule objects to an ArrayList.
        ArrayList<LifecycleRule> ruleList = new ArrayList<>();
        ruleList.add(rule1);
        ruleList.add(rule2);

        BucketLifecycleConfiguration lifecycleConfiguration =
        BucketLifecycleConfiguration.builder()
            .rules(ruleList)
            .build();

        PutBucketLifecycleConfigurationRequest
        putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
            .builder()
            .bucket(bucketName)
            .lifecycleConfiguration(lifecycleConfiguration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the lifecycle configuration for an Amazon S3 bucket and adds a
     * new lifecycle rule to it.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket
     * @param accountId the expected owner of the Amazon S3 bucket
     */
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
    accountId) {

```

```
try {
    GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
    .builder()
    .bucket(bucketName)
    .expectedBucketOwner(accountId)
    .build();

    GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
    List<LifecycleRule> newList = new ArrayList<>();
    List<LifecycleRule> rules = response.rules();
    for (LifecycleRule rule : rules) {
        newList.add(rule);
    }

    // Add a new rule with both a prefix predicate and a tag predicate.
    LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
        .prefix("YearlyDocuments/")
        .build();

    Transition transition = Transition.builder()
        .storageClass(TransitionStorageClass.GLACIER)
        .days(3650)
        .build();

    LifecycleRule rule1 = LifecycleRule.builder()
        .id("NewRule")
        .filter(ruleFilter)
        .transitions(transition)
        .status(ExpirationStatus.ENABLED)
        .build();

    // Add the new rule to the list.
    newList.add(rule1);
    BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
        .rules(newList)
        .build();

    PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
    .builder()
```

```
        .bucket(bucketName)
        .lifecycleConfiguration(lifecycleConfiguration)
        .expectedBucketOwner(accountId)
        .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**
 * Deletes the lifecycle configuration for an Amazon S3 bucket.
 *
 * @param s3 the {@link S3Client} to use for the operation
 * @param bucketName the name of the S3 bucket
 * @param accountId the expected account owner of the S3 bucket
 *
 * @throws S3Exception if an error occurs while deleting the lifecycle
 * configuration
 */
public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
    try {
        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest
        .builder()
        .bucket(bucketName)
        .expectedBucketOwner(accountId)
        .build();

        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```


- For API details, see [PutBucketLifecycleConfiguration](#) in *Amazon SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_lifecycle_configuration(self, lifecycle_rules):
        """
        Apply a lifecycle configuration to the bucket. The lifecycle
        configuration can
            be used to archive or delete the objects in the bucket according to
        specified
            parameters, such as a number of days.

        :param lifecycle_rules: The lifecycle rules to apply.
        """
        try:
            self.bucket.LifecycleConfiguration().put(
                LifecycleConfiguration={"Rules": lifecycle_rules}
            )
            logger.info(
```

```

        "Put lifecycle rules %s for bucket '%s'.",
        lifecycle_rules,
        self.bucket.name,
    )
except ClientError:
    logger.exception(
        "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
    )
    raise

```

- For API details, see [PutBucketLifecycleConfiguration](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

TRY.
    " Example: Expire objects with prefix 'logs/' after 30 days
    lo_s3->putbucketlifecycleconf(
        iv_bucket = iv_bucket_name
        io_lifecycleconfiguration = NEW /aws1/cl_s3_bucketlcconf(
            it_rules = it_lifecycle_rule ) ).
    MESSAGE 'Bucket lifecycle configuration set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- For API details, see [PutBucketLifecycleConfiguration](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketLogging with an Amazon SDK or CLI

The following code examples show how to use PutBucketLogging.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
/// </summary>
public class ServerAccessLogging
{
    private static IConfiguration _configuration = null!;

    public static async Task Main()
    {
        LoadConfig();

        string bucketName = _configuration["BucketName"];
        string logBucketName = _configuration["LogBucketName"];
```

```

        string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
        string accountId = _configuration["AccountId"];

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();

        try
        {
            // Update bucket policy for target bucket to allow delivery of
logs to it.
            await SetBucketPolicyToAllowLogDelivery(
                client,
                bucketName,
                logBucketName,
                logObjectKeyPrefix,
                accountId);

            // Enable logging on the source bucket.
            await EnableLoggingAsync(
                client,
                bucketName,
                logBucketName,
                logObjectKeyPrefix);
        }
        catch (AmazonS3Exception e)
        {
            Console.WriteLine($"Error: {e.Message}");
        }
    }

    /// <summary>
    /// This method grants appropriate permissions for logging to the
    /// Amazon S3 bucket where the logs will be stored.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client which will be
used
    /// to apply the bucket policy.</param>
    /// <param name="sourceBucketName">The name of the source bucket.</param>
    /// <param name="logBucketName">The name of the bucket where logging
    /// information will be stored.</param>

```

```

    /// <param name="logPrefix">The logging prefix where the logs should be
    delivered.</param>
    /// <param name="accountId">The account id of the account where the
    source bucket exists.</param>
    /// <returns>Async task.</returns>
    public static async Task SetBucketPolicyToAllowLogDelivery(
        IAmazonS3 client,
        string sourceBucketName,
        string logBucketName,
        string logPrefix,
        string accountId)
    {
        var resourceArn = @""arn:aws:s3::" + logBucketName + "/" +
logPrefix + @"""";

        var newPolicy = @"{
            ""Statement"": [{
                ""Sid"": ""S3ServerAccessLogsPolicy"",
                ""Effect"": ""Allow"",
                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                ""Action"": [""s3:PutObject""],
                ""Resource"": ["" + resourceArn + @""],
                ""Condition"": {
                    ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3::" + sourceBucketName + @"""" },
                    ""StringEquals"": { ""aws:SourceAccount"": """" +
accountId + @"""" }
                }
            }]
        }";

        Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
        Console.WriteLine(newPolicy);

        PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
        {
            BucketName = logBucketName,
            Policy = newPolicy,
        };
        await client.PutBucketPolicyAsync(putRequest);
        Console.WriteLine("Policy applied.");
    }

```

```

    /// <summary>
    /// This method enables logging for an Amazon S3 bucket. Logs will be
stored
    /// in the bucket you selected for logging. Selected prefix
    /// will be prepended to each log object.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client which will be
used
    /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
    /// <param name="bucketName">The name of the Amazon S3 bucket for which
you
    /// wish to enable logging.</param>
    /// <param name="logBucketName">The name of the Amazon S3 bucket where
logging
    /// information will be stored.</param>
    /// <param name="logObjectKeyPrefix">The prefix to prepend to each
    /// object key.</param>
    /// <returns>Async task.</returns>
    public static async Task EnableLoggingAsync(
        IAmazonS3 client,
        string bucketName,
        string logBucketName,
        string logObjectKeyPrefix)
    {
        Console.WriteLine($"Enabling logging for bucket {bucketName}.");
        var loggingConfig = new S3BucketLoggingConfig
        {
            TargetBucketName = logBucketName,
            TargetPrefix = logObjectKeyPrefix,
        };

        var putBucketLoggingRequest = new PutBucketLoggingRequest
        {
            BucketName = bucketName,
            LoggingConfig = loggingConfig,
        };
        await client.PutBucketLoggingAsync(putBucketLoggingRequest);
        Console.WriteLine($"Logging enabled.");
    }

    /// <summary>
    /// Loads configuration from settings files.
    /// </summary>

```

```
public static void LoadConfig()
{
    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load settings from .json file.
        .AddJsonFile("settings.local.json", true) // Optionally, load
local settings.
        .Build();
}
}
```

- For API details, see [PutBucketLogging](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

Example 1: To set bucket policy logging

The following `put-bucket-logging` example sets the logging policy for *amzn-s3-demo-bucket*. First, grant the logging service principal permission in your bucket policy using the `put-bucket-policy` command.

```
aws s3api put-bucket-policy \
    --bucket amzn-s3-demo-bucket \
    --policy file://policy.json
```

Contents of `policy.json`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3ServerAccessLogsPolicy",
      "Effect": "Allow",
      "Principal": {"Service": "logging.s3.amazonaws.com"},
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/Logs/*",
      "Condition": {
        "ArnLike": {"aws:SourceARN": "arn:aws:s3:::SOURCE-BUCKET-NAME"},

```

```

        "StringEquals": {"aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"}
    }
}
]
}

```

To apply the logging policy, use `put-bucket-logging`.

```

aws s3api put-bucket-logging \
  --bucket amzn-s3-demo-bucket \
  --bucket-logging-status file:///logging.json

```

Contents of `logging.json`:

```

{
  "LoggingEnabled": {
    "TargetBucket": "amzn-s3-demo-bucket",
    "TargetPrefix": "Logs/"
  }
}

```

The `put-bucket-policy` command is required to grant `s3:PutObject` permissions to the logging service principal.

For more information, see [Amazon S3 Server Access Logging](#) in the *Amazon S3 User Guide*.

Example 2: To set a bucket policy for logging access to only a single user

The following `put-bucket-logging` example sets the logging policy for *amzn-s3-demo-bucket*. The Amazon user *bob@example.com* will have full control over the log files, and no one else has any access. First, grant S3 permission with `put-bucket-acl`.

```

aws s3api put-bucket-acl \
  --bucket amzn-s3-demo-bucket \
  --grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery \
  --grant-read-acp URI=http://acs.amazonaws.com/groups/s3/LogDelivery

```

Then apply the logging policy using `put-bucket-logging`.

```

aws s3api put-bucket-logging \

```



```
--bucket amzn-s3-demo-bucket \  
--bucket-logging-status file://logging.json
```

Contents of logging.json:

```
{  
  "LoggingEnabled": {  
    "TargetBucket": "amzn-s3-demo-bucket",  
    "TargetPrefix": "amzn-s3-demo-bucket-logs/",  
    "TargetGrants": [  
      {  
        "Grantee": {  
          "Type": "AmazonCustomerByEmail",  
          "EmailAddress": "bob@example.com"  
        },  
        "Permission": "FULL_CONTROL"  
      }  
    ]  
  }  
}
```

the `put-bucket-acl` command is required to grant S3's log delivery system the necessary permissions (write and read-acp permissions).

For more information, see [Amazon S3 Server Access Logging](#) in the *Amazon S3 Developer Guide*.

- For API details, see [PutBucketLogging](#) in *Amazon CLI Command Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketNotification with a CLI

The following code examples show how to use PutBucketNotification.

CLI

Amazon CLI

The applies a notification configuration to a bucket named `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-notification --bucket amzn-s3-demo-bucket --notification-configuration file://notification.json
```

The file `notification.json` is a JSON document in the current folder that specifies an SNS topic and an event type to monitor:

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

The SNS topic must have an IAM policy attached to it that allows Amazon S3 to publish to it:

```
{
  "Version": "2012-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "example-statement-ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-west-2:123456789012:amzn-s3-demo-bucket",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
        }
      }
    }
  ]
}
```

- For API details, see [PutBucketNotification](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example configures the SNS topic configuration for the S3 event ObjectRemovedDelete and enables notification for the given s3 bucket

```
$topic = [Amazon.S3.Model.TopicConfiguration] @{
    Id = "delete-event"
    Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
    Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
    $topic
```

Example 2: This example enables notifications of ObjectCreatedAll for the given bucket sending it to Lambda function.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
    LambdaFunctionConfiguration $lambdaConfig
```

Example 3: This example creates 2 different Lambda configuration on the basis of different key-suffix and configured both in a single command.

```
#Lambda Config 1
```

```

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
    Id = "ObjectCreated-dada-ps1"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".ps1"}
            )
        }
    }
}

#Lambda Config 2

$secondLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = [Amazon.S3.EventType]::ObjectCreatedAll
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifysm"
    Id = "ObjectCreated-dada-json"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".json"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondLambdaConfig

```

- For API details, see [PutBucketNotification](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example configures the SNS topic configuration for the S3 event ObjectRemovedDelete and enables notification for the given s3 bucket

```

$topic = [Amazon.S3.Model.TopicConfiguration] @{
    Id = "delete-event"
    Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
}

```

```

    Event = [Amazon.S3.EventType]::ObjectRemovedDelete
  }

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
  $topic

```

Example 2: This example enables notifications of `ObjectCreatedAll` for the given bucket sending it to Lambda function.

```

$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
  LambdaFunctionConfiguration $lambdaConfig

```

Example 3: This example creates 2 different Lambda configuration on the basis of different key-suffix and configured both in a single command.

```

#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
    Id = "ObjectCreated-dada-ps1"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".ps1"}
            )
        }
    }
}

```

```

    }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = [Amazon.S3.EventType]::ObjectCreatedAll
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
    Id = "ObjectCreated-dada-json"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".json"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig

```

- For API details, see [PutBucketNotification](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketNotificationConfiguration with an Amazon SDK or CLI

The following code examples show how to use PutBucketNotificationConfiguration.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Process S3 event notifications](#)
- [Send event notifications to EventBridge](#)

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to enable notifications for an Amazon Simple
/// Storage Service (Amazon S3) bucket.
/// </summary>
public class EnableNotifications
{
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket1";
        const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
        const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";

        IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
        await EnableNotificationAsync(client, bucketName, snsTopic,
sqsQueue);
    }

    /// <summary>
    /// This method makes the call to the PutBucketNotificationAsync method.
    /// </summary>
    /// <param name="client">An initialized Amazon S3 client used to call
    /// the PutBucketNotificationAsync method.</param>
    /// <param name="bucketName">The name of the bucket for which
    /// notifications will be turned on.</param>
```

```
/// <param name="snsTopic">The ARN for the Amazon Simple Notification
/// Service (Amazon SNS) topic associated with the S3 bucket.</param>
/// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
/// (Amazon SQS) queue to which notifications will be pushed.</param>
public static async Task EnableNotificationAsync(
    IAmazonS3 client,
    string bucketName,
    string snsTopic,
    string sqsQueue)
{
    try
    {
        // The bucket for which we are setting up notifications.
        var request = new PutBucketNotificationRequest()
        {
            BucketName = bucketName,
        };

        // Defines the topic to use when sending a notification.
        var topicConfig = new TopicConfiguration()
        {
            Events = new List<EventType> { EventType.ObjectCreatedCopy },
            Topic = snsTopic,
        };
        request.TopicConfigurations = new List<TopicConfiguration>
        {
            topicConfig,
        };
        request.QueueConfigurations = new List<QueueConfiguration>
        {
            new QueueConfiguration()
            {
                Events = new List<EventType>
{ EventType.ObjectCreatedPut },
                Queue = sqsQueue,
            },
        };

        // Now apply the notification settings to the bucket.
        PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
    }
    catch (AmazonS3Exception ex)
    {
    }
}
```



```
        Console.WriteLine($"Error: {ex.Message}");  
    }  
}  
}
```

- For API details, see [PutBucketNotificationConfiguration](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To enable the specified notifications to a bucket

The following `put-bucket-notification-configuration` example applies a notification configuration to a bucket named `amzn-s3-demo-bucket`. The file `notification.json` is a JSON document in the current folder that specifies an SNS topic and an event type to monitor.

```
aws s3api put-bucket-notification-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --notification-configuration file://notification.json
```

Contents of `notification.json`:

```
{  
  "TopicConfigurations": [  
    {  
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-  
topic",  
      "Events": [  
        "s3:ObjectCreated:*"  
      ]  
    }  
  ]  
}
```

The SNS topic must have an IAM policy attached to it that allows Amazon S3 to publish to it.

```
{
  "Version": "2012-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "example-statement-ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-west-2:123456789012:s3-notification-
topic",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
        }
      }
    }
  ]
}
```

- For API details, see [PutBucketNotificationConfiguration](#) in *Amazon CLI Command Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketPolicy with an Amazon SDK or CLI

The following code examples show how to use PutBucketPolicy.

C++

SDK for C++

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                const Aws::String &policyBody,
                                const Aws::S3::S3ClientConfiguration
                                &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
        Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
        s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}

//! Build a policy JSON string.
/*!
```

```

\param userArn: Aws user Amazon Resource Name (ARN).
    For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\_identifiers.html#identifiers-arns.
\param bucketName: Name of a bucket.
\return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                           const Aws::String &bucketName) {
    return
        "{\n"
        "  \"Version\": \"2012-10-17\", \n"
        "  \"Statement\": [\n"
        "    {\n"
        "      \"Sid\": \"1\", \n"
        "      \"Effect\": \"Allow\", \n"
        "      \"Principal\": {\n"
        "        \"AWS\": \"\"
        + userArn +
        "\"\"\"      }, \n"
        "      \"Action\": [ \"s3:getObject\" ], \n"
        "      \"Resource\": [ \"arn:aws:s3::\"
        + bucketName +
        \"/*\" ] \n"
        "    } \n"
        "  ] \n"
        "}";
}

```

- For API details, see [PutBucketPolicy](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

This example allows all users to retrieve any object in *amzn-s3-demo-bucket* except those in the *MySecretFolder*. It also grants put and delete permission to the root user of the Amazon account 1234-5678-9012:

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://policy.json
```

```
policy.json:
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/MySecretFolder/*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": [
        "s3:DeleteObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  ]
}
```

- For API details, see [PutBucketPolicy](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <polFile>

            Where:
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
Readme for an example).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
```

```

        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }

    /**
     * Sets the policy for an Amazon S3 bucket.
     *
     * @param s3          the {@link S3Client} object used to interact with the
     Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket
     * @param policyText the text of the policy to be set on the bucket
     * @throws S3Exception if there is an error setting the bucket policy
     */
    public static void setPolicy(S3Client s3, String bucketName, String
policyText) {
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                .bucket(bucketName)
                .policy(policyText)
                .build();

            s3.putBucketPolicy(policyReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }
}

```

```
/**
 * Retrieves the bucket policy from a specified file.
 *
 * @param policyFile the path to the file containing the bucket policy
 * @return the content of the bucket policy file as a string
 */
public static String getBucketPolicyFromFile(String policyFile) {
    StringBuilder fileText = new StringBuilder();
    try {
        List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
        for (String line : lines) {
            fileText.append(line);
        }

    } catch (IOException e) {
        System.out.format("Problem reading file: \"%s\"", policyFile);
        System.out.println(e.getMessage());
    }

    try {
        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
        while (parser.nextToken() != null) {
        }

    } catch (IOException jpe) {
        jpe.printStackTrace();
    }
    return fileText.toString();
}
}
```

- For API details, see [PutBucketPolicy](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Add the policy.

```
import {
  PutBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant an IAM role GetObject access to all of the objects
 * in the provided bucket.
 * @param {{ bucketName: string, iamRoleArn: string }}
 */
export const main = async ({ bucketName, iamRoleArn }) => {
  const client = new S3Client({});
  const command = new PutBucketPolicyCommand({
    // This is a resource-based policy. For more information on resource-based
    // policies,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/
    // access_policies.html#policies_resource-based.
    Policy: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            AWS: iamRoleArn,
          },
          Action: "s3:GetObject",
          Resource: `arn:aws:s3:::${bucketName}/*`,
        },
      ],
    }),
  });
```

```
// Apply the preceding policy to this bucket.
Bucket: bucketName,
});

try {
  await client.send(command);
  console.log(
    `GetObject access to the bucket "${bucketName}" was granted to the provided
IAM role.`,
  );
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "MalformedPolicy"
  ) {
    console.error(
      `Error from S3 while setting the bucket policy for the bucket
"${bucketName}". The policy was malformed.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while setting the bucket policy for the bucket
"${bucketName}". ${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).
- For API details, see [PutBucketPolicy](#) in *Amazon SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_policy(self, policy):
        """
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
            logger.info("Put policy %s for bucket '%s'.", policy,
self.bucket.name)
        except ClientError:
            logger.exception("Couldn't apply policy to bucket '%s'.",
self.bucket.name)
            raise
```

- For API details, see [PutBucketPolicy](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  # configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
    #{e.message}"
    false
  end
end
```

- For API details, see [PutBucketPolicy](#) in *Amazon SDK for Ruby API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
    " Example policy JSON string  
    " iv_policy = '{"Version":"2012-10-17",          "Statement":  
    [{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::123456789012:user/  
user"},"Action":["s3:GetObject"],"Resource":["arn:aws:s3:::bucketname/*"]}]}'  
    lo_s3->putbucketpolicy(  
        iv_bucket = iv_bucket_name  
        iv_policy = iv_policy ).  
    MESSAGE 'Bucket policy set.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [PutBucketPolicy](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketReplication with an Amazon SDK or CLI

The following code examples show how to use PutBucketReplication.

CLI

Amazon CLI

To configure replication for an S3 bucket

The following `put-bucket-replication` example applies a replication configuration to the specified S3 bucket.

```
aws s3api put-bucket-replication \  
  --bucket amzn-s3-demo-bucket1 \  
  --replication-configuration file://replication.json
```

Contents of `replication.json`:

```
{  
  "Role": "arn:aws:iam::123456789012:role/s3-replication-role",  
  "Rules": [  
    {  
      "Status": "Enabled",  
      "Priority": 1,  
      "DeleteMarkerReplication": { "Status": "Disabled" },  
      "Filter" : { "Prefix": ""},  
      "Destination": {  
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket2"  
      }  
    }  
  ]  
}
```

The destination bucket must have versioning enabled. The specified role must have permission to write to the destination bucket and have a trust relationship that allows Amazon S3 to assume the role.

Example role permission policy:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetReplicationConfiguration",  
        "s3:ListBucket"  
      ],  
      "Resource": [  
        "arn:aws:s3:::amzn-s3-demo-bucket1"  
      ]  
    }  
  ]  
}
```

```

    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete",
        "s3:ReplicateTags"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket2/*"
    }
  ]
}

```

Example trust relationship policy:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

This command produces no output.

For more information, see [This is the topic title](#) in the *Amazon Simple Storage Service Console User Guide*.

- For API details, see [PutBucketReplication](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/**
 * Sets the replication configuration for an Amazon S3 bucket.
 *
 * @param s3Client          the S3Client instance to use for the operation
 * @param sourceBucketName  the name of the source bucket
 * @param destBucketName    the name of the destination bucket
 * @param destinationBucketARN the Amazon Resource Name (ARN) of the
destination bucket
 * @param roleARN           the ARN of the IAM role to use for the
replication configuration
 */
public static void setReplication(S3Client s3Client, String sourceBucketName,
String destBucketName, String destinationBucketARN, String roleARN) {
    try {
        Destination destination = Destination.builder()
            .bucket(destinationBucketARN)
            .storageClass(StorageClass.STANDARD)
            .build();

        // Define a prefix filter for replication.
        ReplicationRuleFilter ruleFilter = ReplicationRuleFilter.builder()
            .prefix("documents/")
            .build();

        // Define delete marker replication setting.
        DeleteMarkerReplication deleteMarkerReplication =
DeleteMarkerReplication.builder()
            .status(DeleteMarkerReplicationStatus.DISABLED)
            .build();
```



```
// Create the replication rule.
ReplicationRule replicationRule = ReplicationRule.builder()
    .priority(1)
    .filter(ruleFilter)
    .status(ReplicationRuleStatus.ENABLED)
    .deleteMarkerReplication(deleteMarkerReplication)
    .destination(destination)
    .build();

List<ReplicationRule> replicationRuleList = new ArrayList<>();
replicationRuleList.add(replicationRule);

// Define the replication configuration with IAM role.
ReplicationConfiguration configuration =
ReplicationConfiguration.builder()
    .role(roleARN)
    .rules(replicationRuleList)
    .build();

// Apply the replication configuration to the source bucket.
PutBucketReplicationRequest replicationRequest =
PutBucketReplicationRequest.builder()
    .bucket(sourceBucketName)
    .replicationConfiguration(configuration)
    .build();

s3Client.putBucketReplication(replicationRequest);
System.out.println("Replication configuration set successfully.");

} catch (IllegalArgumentException e) {
    System.err.println("Configuration error: " + e.getMessage());
} catch (S3Exception e) {
    System.err.println("S3 Exception: " +
e.awsErrorDetails().errorMessage());
    System.err.println("Status Code: " + e.statusCode());
    System.err.println("Error Code: " + e.awsErrorDetails().errorCode());

} catch (SdkException e) {
    System.err.println("SDK Exception: " + e.getMessage());
}
}
```

- For API details, see [PutBucketReplication](#) in *Amazon SDK for Java 2.x API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example sets a replication configuration with a single rule enabling replication to the 'amzn-s3-demo-bucket' bucket any new objects created with the key name prefix "TaxDocs" in the bucket 'amzn-s3-demo-bucket'.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Example 2: This example sets a replication configuration with multiple rules enabling replication to the 'amzn-s3-demo-bucket' bucket any new objects created with either the key name prefix "TaxDocs" or "OtherDocs". The key prefixes must not overlap.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
```

```
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

Example 3: This example updates the replication configuration on the specified bucket to disable the rule controlling replication of objects with the key name prefix "TaxDocs" to the bucket 'amzn-s3-demo-bucket'.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- For API details, see [PutBucketReplication](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example sets a replication configuration with a single rule enabling replication to the 'amzn-s3-demo-bucket' bucket any new objects created with the key name prefix "TaxDocs" in the bucket 'amzn-s3-demo-bucket'.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
```

```
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Example 2: This example sets a replication configuration with multiple rules enabling replication to the 'amzn-s3-demo-bucket' bucket any new objects created with either the key name prefix "TaxDocs" or "OtherDocs". The key prefixes must not overlap.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

Example 3: This example updates the replication configuration on the specified bucket to disable the rule controlling replication of objects with the key name prefix "TaxDocs" to the bucket 'amzn-s3-demo-bucket'.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- For API details, see [PutBucketReplication](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketRequestPayment with a CLI

The following code examples show how to use PutBucketRequestPayment.

CLI

Amazon CLI

Example 1: To enable ``requester pays`` configuration for a bucket

The following put-bucket-request-payment example enables requester pays for the specified bucket.

```
aws s3api put-bucket-request-payment \
```

```
--bucket amzn-s3-demo-bucket \  
--request-payment-configuration '{"Payer":"Requester"}'
```

This command produces no output.

Example 2: To disable ``requester pays`` configuration for a bucket

The following `put-bucket-request-payment` example disables `requester pays` for the specified bucket.

```
aws s3api put-bucket-request-payment \  
--bucket amzn-s3-demo-bucket \  
--request-payment-configuration '{"Payer":"BucketOwner"}'
```

This command produces no output.

- For API details, see [PutBucketRequestPayment](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Updates the request payment configuration for the bucket named 'amzn-s3-demo-bucket' so that the person requesting downloads from the bucket will be charged for the download. By default the bucket owner pays for downloads. To set the request payment back to the default use 'BucketOwner' for the `RequestPaymentConfiguration_Payer` parameter.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- For API details, see [PutBucketRequestPayment](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Updates the request payment configuration for the bucket named 'amzn-s3-demo-bucket' so that the person requesting downloads from the bucket will be charged for the download. By default the bucket owner pays for downloads. To set the request payment back to the default use 'BucketOwner' for the `RequestPaymentConfiguration_Payer` parameter.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- For API details, see [PutBucketRequestPayment](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketTagging with a CLI

The following code examples show how to use PutBucketTagging.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Getting started with Amazon S3](#)

CLI

Amazon CLI

The following command applies a tagging configuration to a bucket named amzn-s3-demo-bucket:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging file://  
tagging.json
```

The file tagging.json is a JSON document in the current folder that specifies tags:

```
{  
  "TagSet": [  
    {  
      "Key": "organization",  
      "Value": "marketing"  
    }  
  ]  
}
```

Or apply a tagging configuration to `amzn-s3-demo-bucket` directly from the command line:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging  
'TagSet=[{Key=organization,Value=marketing}]'
```

- For API details, see [PutBucketTagging](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command applies two tags to a bucket named `cloudtrail-test-2018`: a tag with a key of `Stage` and a value of `Test`, and a tag with a key of `Environment` and a value of `Alpha`. To verify that the tags were added to the bucket, run `Get-S3BucketTagging -BucketName bucket_name`. The results should show the tags that you applied to the bucket in the first command. Note that `Write-S3BucketTagging` overwrites the entire existing tag set on a bucket. To add or delete individual tags, run the Resource Groups and Tagging API cmdlets, `Add-RGTResourceTag` and `Remove-RGTResourceTag`. Alternatively, use Tag Editor in the Amazon Management Console to manage S3 bucket tags.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";  
    Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Example 2: This command pipes a bucket named `cloudtrail-test-2018` into the `Write-S3BucketTagging` cmdlet. It applies tags `Stage:Production` and `Department:Finance` to the bucket. Note that `Write-S3BucketTagging` overwrites the entire existing tag set on a bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging  
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";  
    Value="Finance" } )
```

- For API details, see [PutBucketTagging](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command applies two tags to a bucket named `cloudtrail-test-2018`: a tag with a key of `Stage` and a value of `Test`, and a tag with a key of `Environment` and a value of `Alpha`. To verify that the tags were added to the bucket, run `Get-S3BucketTagging -BucketName bucket_name`. The results should show the tags that you applied to the bucket in the first command. Note that `Write-S3BucketTagging` overwrites the entire existing tag set on a bucket. To add or delete individual tags, run the Resource Groups and Tagging API cmdlets, `Add-RGTResourceTag` and `Remove-RGTResourceTag`. Alternatively, use Tag Editor in the Amazon Management Console to manage S3 bucket tags.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage"; Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Example 2: This command pipes a bucket named `cloudtrail-test-2018` into the `Write-S3BucketTagging` cmdlet. It applies tags `Stage:Production` and `Department:Finance` to the bucket. Note that `Write-S3BucketTagging` overwrites the entire existing tag set on a bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- For API details, see [PutBucketTagging](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketVersioning with an Amazon SDK or CLI

The following code examples show how to use `PutBucketVersioning`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Getting started with Amazon S3](#)

CLI

Amazon CLI

The following command enables versioning on a bucket named `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled
```

The following command enables versioning, and uses an mfa code

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled --mfa "SERIAL 123456"
```

- For API details, see [PutBucketVersioning](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/**
 * Enables bucket versioning for the specified S3 bucket.
 *
 * @param s3Client the S3 client to use for the operation
 * @param bucketName the name of the S3 bucket to enable versioning for
 */
public static void enableBucketVersioning(S3Client s3Client, String
bucketName){
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
```

```
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

s3Client.putBucketVersioning(versioningRequest);
System.out.println("Bucket versioning has been enabled for "+bucketName);
}
```

- For API details, see [PutBucketVersioning](#) in *Amazon SDK for Java 2.x API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: The command enables versioning for the given S3 bucket.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -
VersioningConfig_Status Enabled
```

- For API details, see [PutBucketVersioning](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: The command enables versioning for the given S3 bucket.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -
VersioningConfig_Status Enabled
```

- For API details, see [PutBucketVersioning](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
  " Example: Enable versioning on a bucket  
  " iv_status = 'Enabled'  
  lo_s3->putbucketversioning(  
    iv_bucket = iv_bucket_name  
    io_versioningconfiguration = NEW /aws1/cl_s3_versioningconf(  
      iv_status = iv_status ) ).  
  MESSAGE 'Bucket versioning enabled.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
  MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [PutBucketVersioning](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketWebsite with an Amazon SDK or CLI

The following code examples show how to use PutBucketWebsite.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
// Put the website configuration.
PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
{
    BucketName = bucketName,
    WebsiteConfiguration = new WebsiteConfiguration()
    {
        IndexDocumentSuffix = indexDocumentSuffix,
        ErrorDocument = errorDocument,
    },
};
PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- For API details, see [PutBucketWebsite](#) in *Amazon SDK for .NET API Reference*.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
```

```

        const Aws::String &indexPage, const Aws::String
&errorPage,
        const Aws::S3::S3ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
        client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
            << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
            << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- For API details, see [PutBucketWebsite](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The applies a static website configuration to a bucket named `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-website --bucket amzn-s3-demo-bucket --website-configuration file://website.json
```

The file `website.json` is a JSON document in the current folder that specifies index and error pages for the website:

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- For API details, see [PutBucketWebsite](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
```

```

* <p>
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucketName> [indexdoc]\s

            Where:
                bucketName    - The Amazon S3 bucket to set the website
configuration on.\s
                indexdoc - The index document, ex. 'index.html'
                        If not specified, 'index.html' will be set.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }

    /**
     * Sets the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance to use for the AWS SDK operations.
     * @param bucketName The name of the S3 bucket to configure.
     * @param indexDoc The name of the index document to use for the website
configuration.
     */
    public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {

```



```
try {
    WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
        .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
        .build();

    PutBucketWebsiteRequest pubWebsiteReq =
    PutBucketWebsiteRequest.builder()
        .bucket(bucketName)
        .websiteConfiguration(websiteConfig)
        .build();

    s3.putBucketWebsite(pubWebsiteReq);
    System.out.println("The call was successful");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [PutBucketWebsite](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Set the website configuration.

```
import {
    PutBucketWebsiteCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
/**
 * Configure an Amazon S3 bucket to serve a static website.
 * Website access must also be granted separately. For more information
 * on setting the permissions for website access, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/WebsiteAccessPermissionsReqd.html.
 *
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutBucketWebsiteCommand({
    Bucket: bucketName,
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
      },
      IndexDocument: {
        // A suffix that is appended to a request when the request is
        // for a directory.
        Suffix: "index.html",
      },
    },
  });

  try {
    await client.send(command);
    console.log(
      `The bucket "${bucketName}" has been configured as a static website.`
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while configuring the bucket "${bucketName}" as a static website. The bucket doesn't exist.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while configuring the bucket "${bucketName}" as a static website. ${caught.name}: ${caught.message}`
      );
    }
  }
}
```

```
    );  
  } else {  
    throw caught;  
  }  
}  
};
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).
- For API details, see [PutBucketWebsite](#) in *Amazon SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: The command enables website hosting for the given bucket with the index document as 'index.html' and error document as 'error.html'.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'  
  -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -  
WebsiteConfiguration_ErrorDocument 'error.html'
```

- For API details, see [PutBucketWebsite](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: The command enables website hosting for the given bucket with the index document as 'index.html' and error document as 'error.html'.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'  
  -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -  
WebsiteConfiguration_ErrorDocument 'error.html'
```

- For API details, see [PutBucketWebsite](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
  # configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the
  # website.
  # @param error_document [String] The name of the error document to show for 4XX
  # errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
  # false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
    why: #{e.message}"
    false
  end
end
```

```
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [PutBucketWebsite](#) in *Amazon SDK for Ruby API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutObject with an Amazon SDK or CLI

The following code examples show how to use PutObject.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Learn the basics](#)
- [Getting started with Amazon S3](#)
- [Make conditional requests](#)
- [Track uploads and downloads](#)
- [Work with Amazon S3 object integrity](#)

.NET

Amazon SDK for .NET (v4)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to which the object
/// will be uploaded.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object
/// on the local computer to upload.</param>
/// <returns>A boolean value indicating the success or failure of the
/// upload procedure.</returns>
public async Task<bool> UploadFileAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    try
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
        };

        var response = await _amazonS3.PutObjectAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
    }
}
```

```
        return false;
    }
}
```

- For API details, see [PutObject](#) in *Amazon SDK for .NET API Reference*.

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Upload an object with server-side encryption.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to upload an object to an Amazon Simple Storage
/// Service (Amazon S3) bucket with server-side encryption enabled.
/// </summary>
public class ServerSideEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "samplefile.txt";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();

        await WritingAnObjectAsync(client, bucketName, keyName);
    }
}
```

```

    /// <summary>
    /// Upload a sample object include a setting for encryption.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// to upload a file and apply server-side encryption.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// encrypted object will reside.</param>
    /// <param name="keyName">The name for the object that you want to
    /// create in the supplied bucket.</param>
    public static async Task WritingAnObjectAsync(IAmazonS3 client, string
bucketName, string keyName)
    {
        try
        {
            var putRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionMethod =
ServerSideEncryptionMethod.AES256,
            };

            var putResponse = await client.PutObjectAsync(putRequest);

            // Determine the encryption state of an object.
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,
            };
            GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
            ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;

            Console.WriteLine($"Encryption method used: {0}",
objectEncryption.ToString());
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error: '{ex.Message}' when writing an
object");
        }
    }
}

```



```

    }
  }
}

```

Put an object using a conditional request.

```

/// <summary>
/// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
/// </summary>
/// <param name="objectKey">The key of the object to upload.</param>
/// <param name="bucket">The source bucket of the object.</param>
/// <param name="content">The content to upload as a string.</param>
/// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
{
    try
    {
        var putObjectRequest = new PutObjectRequest
        {
            BucketName = bucket,
            Key = objectKey,
            ContentBody = content,
            IfNoneMatch = "*"
        };

        var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
        _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
        return putResult.ETag;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional write failed: Precondition
failed");
        }
        else
    }

```

```

        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return string.Empty;
    }
}

```

- For API details, see [PutObject](#) in *Amazon SDK for .NET API Reference*.

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file to.
#     $2 - The path and file name of the local file to copy to the bucket.
#     $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#     0 - If successful.

```

```
#      1 - If it fails.
#####
function copy_file_to_bucket() {
    local response bucket_name source_file destination_file_name
    bucket_name=$1
    source_file=$2
    destination_file_name=$3

    response=$(aws s3api put-object \
        --bucket "$bucket_name" \
        --body "$source_file" \
        --key "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}
```

- For API details, see [PutObject](#) in *Amazon CLI Command Reference*.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
```

```

//However, this is just a string and can be set according to your retrieval
needs.
request.SetKey(fileName);

std::shared_ptr<Aws::IOStream> inputData =
    Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                   fileName.c_str(),
                                   std::ios_base::in |
std::ios_base::binary);

if (!*inputData) {
    std::cerr << "Error unable to read file " << fileName << std::endl;
    return false;
}

request.SetBody(inputData);

Aws::S3::Model::PutObjectOutcome outcome =
    s3Client.PutObject(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: putObject: " <<
        outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Added object '" << fileName << "' to bucket '"
        << bucketName << "'.";
}

return outcome.IsSuccess();
}

```

- For API details, see [PutObject](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

Example 1: Upload an object to Amazon S3

The following put-object command example uploads an object to Amazon S3.

```
aws s3api put-object \
```

```
--bucket amzn-s3-demo-bucket \  
--key my-dir/MySampleImage.png \  
--body MySampleImage.png
```

For more information about uploading objects, see [Uploading Objects](http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html) < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html>> in the *Amazon S3 Developer Guide*.

Example 2: Upload a video file to Amazon S3

The following `put-object` command example uploads a video file.

```
aws s3api put-object \  
  --bucket amzn-s3-demo-bucket \  
  --key my-dir/big-video-file.mp4 \  
  --body /media/videos/f-sharp-3-data-services.mp4
```

For more information about uploading objects, see [Uploading Objects](http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html) < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html>> in the *Amazon S3 Developer Guide*.

- For API details, see [PutObject](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Put an object in a bucket by using the low-level API.

```
import (  
  "bytes"  
  "context"  
  "errors"  
  "fmt"
```

```
"io"
"log"
"os"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
        log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
    } else {
        defer file.Close()
        _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
            Body:   file,
        })
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
                log.Printf("Error while uploading object to %s. The object is too large.\n"+
                    "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                    "or the multipart upload API (5TB max).", bucketName)
            } else {
```

```
    log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
        fileName, bucketName, objectKey, err)
}
} else {
    err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectKey)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
    }
}
}
return err
}
```

Upload an object to a bucket by using a transfer manager.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}
```

```
// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
string, contents string) (string, error) {
    var outKey string
    input := &s3.PutObjectInput{
        Bucket:      aws.String(bucket),
        Key:         aws.String(key),
        Body:        bytes.NewReader([]byte(contents)),
        ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
    }
    output, err := actor.S3Manager.Upload(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    } else {
        err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
            Bucket: aws.String(bucket),
            Key:    aws.String(key),
        }, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
            bucket)
        } else {
            outKey = *output.Key
        }
    }
    return outKey, err
}
```

- For API details, see [PutObject](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Upload a file to a bucket using an [S3Client](#).

```
/**
 * Uploads a local file to an AWS S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (object name) to use for the uploaded file
 * @param objectPath  the local file path of the file to be uploaded
 * @return a {@link CompletableFuture} that completes with the {@link
 * PutObjectResponse} when the upload is successful, or throws a {@link
 * RuntimeException} if the upload fails
 */
public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
    return response.whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to upload file", ex);
        }
    });
}
```

Use an [S3TransferManager](#) to [upload a file](#) to a bucket. View the [complete file](#) and [test](#).

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String
bucketName,

                            String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }

```

Upload an object to a bucket and set tags using an [S3Client](#).

```

/**
 * Puts tags on an Amazon S3 object.
 *
 * @param s3 An {@link S3Client} object that represents the Amazon S3 client.
 * @param bucketName The name of the Amazon S3 bucket.
 * @param objectKey The key of the Amazon S3 object.
 * @param objectPath The file path of the object to be uploaded.
 */
public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
    try {
        Tag tag1 = Tag.builder()
            .key("Tag 1")

```

```
        .value("This is tag 1")
        .build();

    Tag tag2 = Tag.builder()
        .key("Tag 2")
        .value("This is tag 2")
        .build();

    List<Tag> tags = new ArrayList<>();
    tags.add(tag1);
    tags.add(tag2);

    Tagging allTags = Tagging.builder()
        .tagSet(tags)
        .build();

    PutObjectRequest putOb = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .tagging(allTags)
        .build();

    s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

/**
 * Updates the tags associated with an object in an Amazon S3 bucket.
 *
 * @param s3 an instance of the S3Client class, which is used to interact
with the Amazon S3 service
 * @param bucketName the name of the S3 bucket containing the object
 * @param objectKey the key (or name) of the object in the S3 bucket
 * @throws S3Exception if there is an error updating the object's tags
 */
public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
    try {
```

```
GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .build();

GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
List<Tag> obTags = getTaggingRes.tagSet();
for (Tag sinTag : obTags) {
    System.out.println("The tag key is: " + sinTag.key());
    System.out.println("The tag value is: " + sinTag.value());
}

// Replace the object's tags with two new tags.
Tag tag3 = Tag.builder()
    .key("Tag 3")
    .value("This is tag 3")
    .build();

Tag tag4 = Tag.builder()
    .key("Tag 4")
    .value("This is tag 4")
    .build();

List<Tag> tags = new ArrayList<>();
tags.add(tag3);
tags.add(tag4);

Tagging updatedTags = Tagging.builder()
    .tagSet(tags)
    .build();

PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .tagging(updatedTags)
    .build();

s3.putObjectTagging(taggingRequest1);
GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
List<Tag> modTags = getTaggingRes2.tagSet();
```

```
        for (Tag sinTag : modTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

/**
 * Retrieves the contents of a file as a byte array.
 *
 * @param filePath the path of the file to be read
 * @return a byte array containing the contents of the file, or null if an
error occurs
 */
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(byteArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    return byteArray;
}
}
```

Upload an object to a bucket and set metadata using an [S3Client](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <objectKey> <objectPath>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                objectKey - The object to upload (for example, book.pdf).
                objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
            "";

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }
    }
}
```

```
        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    /**
     * Uploads an object to an Amazon S3 bucket with metadata.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket to upload the object to
     * @param objectKey the name of the object to be uploaded
     * @param objectPath the local file path of the object to be uploaded
     */
    public static void putS3Object(S3Client s3, String bucketName, String
objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .metadata(metadata)
                .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
" + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}  
}
```

Upload an object to a bucket and set an object retention value using an [S3Client](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;  
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
  
import java.time.Instant;  
import java.time.LocalDate;  
import java.time.LocalDateTime;  
import java.time.ZoneOffset;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 * <p>  
 * For more information, see the following documentation topic:  
 * <p>  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
  
public class PutObjectRetention {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
            <key> <bucketName>\s  
  
            Where:  
            key - The name of the object (for example, book.pdf).\s  
            bucketName - The Amazon S3 bucket name that contains the object  
            (for example, bucket1).\s  
            "";  
  
        if (args.length != 2) {  
            System.out.println(usage);
```



```
        System.exit(1);
    }

    String key = args[0];
    String bucketName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setRentionPeriod(s3, key, bucketName);
    s3.close();
}

/**
 * Sets the retention period for an object in an Amazon S3 bucket.
 *
 * @param s3      the S3Client object used to interact with the Amazon S3
service
 * @param key     the key (name) of the object in the S3 bucket
 * @param bucket the name of the S3 bucket where the object is stored
 *
 * @throws S3Exception if an error occurs while setting the object retention
period
 */
public static void setRentionPeriod(S3Client s3, String key, String bucket) {
    try {
        LocalDate localDate = LocalDate.parse("2020-07-17");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        ObjectLockRetention lockRetention = ObjectLockRetention.builder()
            .mode("COMPLIANCE")
            .retainUntilDate(instant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucket)
            .key(key)
            .bypassGovernanceRetention(true)
            .retention(lockRetention)
            .build();
    }
}
```

```
        // To set Retention on an object, the Amazon S3 bucket must support
        object
            // locking, otherwise an exception is thrown.
            s3.putObjectRetention(retentionRequest);
            System.out.print("An object retention configuration was successfully
placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [PutObject](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Upload the object.

```
import { readFile } from "node:fs/promises";

import {
    PutObjectCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Upload a file to an S3 bucket.
 * @param {{ bucketName: string, key: string, filePath: string }}
 */
export const main = async ({ bucketName, key, filePath }) => {
```

```

const client = new S3Client({});
const command = new PutObjectCommand({
  Bucket: bucketName,
  Key: key,
  Body: await readFile(filePath),
});

try {
  const response = await client.send(command);
  console.log(response);
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "EntityTooLarge"
  ) {
    console.error(
      `Error from S3 while uploading object to ${bucketName}. \
The object was too large. To upload objects larger than 5GB, use the S3 console \
(160GB max) \
or the multipart upload API (5TB max).`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while uploading object to ${bucketName}. ${caught.name}: \
${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};

```

Upload the object on condition its ETag matches the one provided.

```

import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

```

```
/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
```

```
const options = {
  bucketName: {
    type: "string",
    required: true,
  },
  key: {
    type: "string",
    required: true,
  },
  eTag: {
    type: "string",
    required: true,
  },
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).
- For API details, see [PutObject](#) in *Amazon SDK for JavaScript API Reference*.

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- For API details, see [PutObject](#) in *Amazon SDK for Kotlin API reference*.

PHP

SDK for PHP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Upload an object to a bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

$fileName = __DIR__ . "/local-file-" . uniqid();
try {
```

```
$this->s3client->putObject([
    'Bucket' => $this->bucketName,
    'Key' => $fileName,
    'SourceFile' => __DIR__ . '/testfile.txt'
]);
echo "Uploaded $fileName to $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with file upload before continuing.");
}
```

- For API details, see [PutObject](#) in *Amazon SDK for PHP API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This command uploads the single file "local-sample.txt" to Amazon S3, creating an object with key "sample.txt" in bucket "test-files".

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Example 2: This command uploads the single file "sample.txt" to Amazon S3, creating an object with key "sample.txt" in bucket "test-files". If the -Key parameter is not supplied, the filename is used as the S3 object key.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Example 3: This command uploads the single file "local-sample.txt" to Amazon S3, creating an object with key "prefix/to/sample.txt" in bucket "test-files".

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -
File .\local-sample.txt
```

Example 4: This command uploads all files in the subdirectory "Scripts" to the bucket "test-files" and applies the common key prefix "SampleScripts" to each object. Each uploaded file will have a key of "SampleScripts/filename" where 'filename' varies.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\
```

Example 5: This command uploads all *.ps1 files in the local director "Scripts" to bucket "test-files" and applies the common key prefix "SampleScripts" to each object. Each uploaded file will have a key of "SampleScripts/filename.ps1" where 'filename' varies.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Example 6: This command creates a new S3 object containing the specified content string with key 'sample.txt'.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Example 7: This command uploads the specified file (the filename is used as the key) and applies the specified tags to the new object.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Example 8: This command recursively uploads the specified folder and applies the specified tags to all the new objects.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"  
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- For API details, see [PutObject](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command uploads the single file "local-sample.txt" to Amazon S3, creating an object with key "sample.txt" in bucket "test-files".

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-  
sample.txt
```


Example 2: This command uploads the single file "sample.txt" to Amazon S3, creating an object with key "sample.txt" in bucket "test-files". If the -Key parameter is not supplied, the filename is used as the S3 object key.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Example 3: This command uploads the single file "local-sample.txt" to Amazon S3, creating an object with key "prefix/to/sample.txt" in bucket "test-files".

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -  
File .\local-sample.txt
```

Example 4: This command uploads all files in the subdirectory "Scripts" to the bucket "test-files" and applies the common key prefix "SampleScripts" to each object. Each uploaded file will have a key of "SampleScripts/filename" where 'filename' varies.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\
```

Example 5: This command uploads all *.ps1 files in the local director "Scripts" to bucket "test-files" and applies the common key prefix "SampleScripts" to each object. Each uploaded file will have a key of "SampleScripts/filename.ps1" where 'filename' varies.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Example 6: This command creates a new S3 object containing the specified content string with key 'sample.txt'.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Example 7: This command uploads the specified file (the filename is used as the key) and applies the specified tags to the new object.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Example 8: This command recursively uploads the specified folder and applies the specified tags to all the new objects.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
  -Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- For API details, see [PutObject](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string.
        When this
                           argument is a string, it is interpreted as a file name,
        which is
                           opened in read bytes mode.
        """
```

```

        put_data = data
        if isinstance(data, str):
            try:
                put_data = open(data, "rb")
            except IOError:
                logger.exception("Expected file name or binary data, got '%s'."
data)
                raise

        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'."
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'."
                self.object.key,
                self.object.bucket_name,
            )
            raise
    finally:
        if getattr(put_data, "close", None):
            put_data.close()

```

Upload an object using a conditional request.

```

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """

```

```
s3_client = boto3.client("s3")
return cls(s3_client)

def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
    """
    Uploads an object to Amazon S3 with a conditional request. Prevents
    overwrite
    using an IfNoneMatch condition for the object key.

    :param object_key: The key of the object to upload.
    :param source_bucket: The source bucket of the object.
    :param data: The data to upload.
    """
    try:
        self.s3.put_object(
            Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
        )
        print(
            f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
        )
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional write failed: Precondition failed")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise
```

- For API details, see [PutObject](#) in *Amazon SDK for Python (Boto3) API Reference*.

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Upload a file using a managed uploader (Object.upload_file).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
#{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"
```

```

    wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
object_key))
    return unless wrapper.upload_file(file_path)

    puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

Upload a file using Object.put.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
#{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)

```

```
    return unless success

    puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
  end

  run_demo if $PROGRAM_NAME == __FILE__
```

Upload a file using `Object.put` and add server-side encryption.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{@object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
    object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
    #{encryption}."
```

```
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [PutObject](#) in *Amazon SDK for Ruby API Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
        aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}
```

- For API details, see [PutObject](#) in *Amazon SDK for Rust API reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
"Get contents of file from application server."
DATA lv_body TYPE xstring.
OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
READ DATASET iv_file_name INTO lv_body.
CLOSE DATASET iv_file_name.

"Upload/put an object to an S3 bucket."
TRY.
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_file_name
        iv_body = lv_body ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- For API details, see [PutObject](#) in *Amazon SDK for SAP ABAP API reference*.

Swift

SDK for Swift

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import AWSS3
import Smithy

public fun uploadFile(bucket: String, key: String, file: String) async
throws {
    let fileUrl = URL(fileURLWithPath: file)
    do {
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object."))
        throw error
    }
}
```

```
import AWSS3
import Smithy

public fun createFile(bucket: String, key: String, withData data: Data)
async throws {
    let dataStream = ByteStream.data(data)

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object."))
    }
}
```

```
        throw error
    }
}
```

- For API details, see [PutObject](#) in *Amazon SDK for Swift API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutObjectAcl with an Amazon SDK or CLI

The following code examples show how to use PutObjectAcl.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Manage access control lists \(ACLs\)](#)

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
    &objectKey, const Aws::String &ownerID,
                                const Aws::String &granteePermission, const
    Aws::String &granteeType,
                                const Aws::String &granteeID, const Aws::String
    &granteeEmailAddress,
                                const Aws::String &granteeURI, const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
```

```
owner.SetID(ownerID);

Aws::S3::Model::Grantee grantee;
grantee.SetType(setGranteeType(granteeType));

if (!granteeEmailAddress.empty()) {
    grantee.SetEmailAddress(granteeEmailAddress);
}

if (!granteeID.empty()) {
    grantee.SetID(granteeID);
}

if (!granteeURI.empty()) {
    grantee.SetURI(granteeURI);
}

Aws::S3::Model::Grant grant;
grant.SetGrantee(grantee);
grant.SetPermission(setGranteePermission(granteePermission));

Aws::Vector<Aws::S3::Model::Grant> grants;
grants.push_back(grant);

Aws::S3::Model::AccessControlPolicy acp;
acp.SetOwner(owner);
acp.SetGrants(grants);

Aws::S3::Model::PutObjectAclRequest request;
request.SetAccessControlPolicy(acp);
request.SetBucket(bucketName);
request.SetKey(objectKey);

Aws::S3::Model::PutObjectAclOutcome outcome =
    s3Client.PutObjectAcl(request);

if (!outcome.IsSuccess()) {
    auto error = outcome.GetError();
    std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
               << " - " << error.GetMessage() << std::endl;
} else {
    std::cout << "Successfully added an ACL to the object '" << objectKey
               << "' in the bucket '" << bucketName << "'." << std::endl;
}
```

```

        return outcome.IsSuccess();
    }

    //! Routine which converts a human-readable string to a built-in type
    enumeration.
    /*!
    \param access: Human readable string.
    \return Permission: Permission enumeration.
    */
    Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
        if (access == "FULL_CONTROL")
            return Aws::S3::Model::Permission::FULL_CONTROL;
        if (access == "WRITE")
            return Aws::S3::Model::Permission::WRITE;
        if (access == "READ")
            return Aws::S3::Model::Permission::READ;
        if (access == "WRITE_ACP")
            return Aws::S3::Model::Permission::WRITE_ACP;
        if (access == "READ_ACP")
            return Aws::S3::Model::Permission::READ_ACP;
        return Aws::S3::Model::Permission::NOT_SET;
    }

    //! Routine which converts a human-readable string to a built-in type
    enumeration.
    /*!
    \param type: Human readable string.
    \return Type: Type enumeration.
    */
    Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
        if (type == "Amazon customer by email")
            return Aws::S3::Model::Type::AmazonCustomerByEmail;
        if (type == "Canonical user")
            return Aws::S3::Model::Type::CanonicalUser;
        if (type == "Group")
            return Aws::S3::Model::Type::Group;
        return Aws::S3::Model::Type::NOT_SET;
    }
}

```

- For API details, see [PutObjectAcl](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command grants `full control` to two Amazon users (`user1@example.com` and `user2@example.com`) and read permission to everyone:

```
aws s3api put-object-acl --bucket amzn-s3-demo-bucket --key file.txt --grant-  
full-control emailaddress=user1@example.com,emailaddress=user2@example.com --  
grant-read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

See <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> for details on custom ACLs (the `s3api` ACL commands, such as `put-object-acl`, use the same shorthand argument notation).

- For API details, see [PutObjectAcl](#) in *Amazon CLI Command Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                               that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def put_acl(self, email):
```

```

"""
    Applies an ACL to the object that grants read access to an AWS user
    identified
    by email address.

:param email: The email address of the user to grant access.
"""
try:
    acl = self.object.Acl()
    # Putting an ACL overwrites the existing ACL, so append new grants
    # if you want to preserve existing grants.
    grants = acl.grants if acl.grants else []
    grants.append(
        {
            "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
email},
            "Permission": "READ",
        }
    )
    acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
    logger.info("Granted read access to %s.", email)
except ClientError:
    logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
    raise

```

- For API details, see [PutObjectAcl](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

TRY.

```

" Example: Grant read access to an AWS user
" iv_grantread = 'emailAddress=user@example.com'

```

```

lo_s3->putobjectacl(
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key
    iv_grantread = iv_grantread ).
MESSAGE 'Object ACL updated.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

- For API details, see [PutObjectAcl](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutObjectLegalHold with an Amazon SDK or CLI

The following code examples show how to use PutObjectLegalHold.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Lock Amazon S3 objects](#)

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

/// <summary>
/// Set or modify a legal hold on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>

```



```

    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }
}

```

- For API details, see [PutObjectLegalHold](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To apply a Legal Hold to an object

The following `put-object-legal-hold` example sets a Legal Hold on the object `doc1.rtf`.

```
aws s3api put-object-legal-hold \
```

```
--bucket amzn-s3-demo-bucket-with-object-lock \  
--key doc1.rtf \  
--legal-hold Status=ON
```

This command produces no output.

- For API details, see [PutObjectLegalHold](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}
```

```
// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
{
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }

    _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}
```

- For API details, see [PutObjectLegalHold](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
// Set or modify a legal hold on an object in an S3 bucket.
```

```
public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
    PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for " + objectKey + " in
    "+bucketName +".");
}
```

- For API details, see [PutObjectLegalHold](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import {
    PutObjectLegalHoldCommand,
    S3Client,
    S3ServiceException,
```

```

} from "@aws-sdk/client-s3";

/**
 * Apply a legal hold configuration to the specified object.
 * @param {{ bucketName: string, objectKey: string, legalHoldStatus: "ON" |
 * "OFF" }}
 */
export const main = async ({ bucketName, objectKey, legalHoldStatus }) => {
  if (!["OFF", "ON"].includes(legalHoldStatus.toUpperCase())) {
    throw new Error(
      "Invalid parameter. legalHoldStatus must be 'ON' or 'OFF'.",
    );
  }

  const client = new S3Client({});
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: legalHoldStatus,
    },
  });

  try {
    await client.send(command);
    console.log(
      `Legal hold status set to "${legalHoldStatus}" for "${objectKey}" in "${bucketName}"`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    }
  }
}

```

```
    );  
  } else {  
    throw caught;  
  }  
}  
};  
  
// Call function if run directly  
import { parseArgs } from "node:util";  
import {  
  isMain,  
  validateArgs,  
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";  
  
const loadArgs = () => {  
  const options = {  
    bucketName: {  
      type: "string",  
      required: true,  
    },  
    objectKey: {  
      type: "string",  
      required: true,  
    },  
    legalHoldStatus: {  
      type: "string",  
      default: "ON",  
    },  
  };  
};  
const results = parseArgs({ options });  
const { errors } = validateArgs({ options }, results);  
return { errors, results };  
};  
  
if (isMain(import.meta.url)) {  
  const { errors, results } = loadArgs();  
  if (!errors) {  
    main(results.values);  
  } else {  
    console.error(errors.join("\n"));  
  }  
}
```

- For API details, see [PutObjectLegalHold](#) in *Amazon SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Put an object legal hold.

```
def set_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Set a legal hold on a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to set the legal hold on.
    """
    print()
    logger.info("Setting legal hold on file [%s] in bucket [%s]", key, bucket)
    try:
        before_status = "OFF"
        after_status = "ON"
        s3_client.put_object_legal_hold(
            Bucket=bucket, Key=key, LegalHold={"Status": after_status}
        )
        logger.debug(
            "Legal hold set successfully on file [%s] in bucket [%s]", key,
            bucket
        )
        _print_legal_hold_update(bucket, key, before_status, after_status)
    except Exception as e:
        logger.error(
            "Failed to set legal hold on file [%s] in bucket [%s]: %s", key,
            bucket, e
        )
```

- For API details, see [PutObjectLegalHold](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
  " Example: Set legal hold status to ON  
  " iv_status = 'ON'  
  lo_s3->putobjectlegalhold(  
    iv_bucket = iv_bucket_name  
    iv_key = iv_object_key  
    io_legalhold = NEW /aws1/cl_s3_objlocklegalhold(  
      iv_status = iv_status ) ).  
  MESSAGE 'Object legal hold status set.' TYPE 'I'.  
  CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
  CATCH /aws1/cx_s3_nosuchkey.  
    MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [PutObjectLegalHold](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutObjectLockConfiguration with an Amazon SDK or CLI

The following code examples show how to use PutObjectLockConfiguration.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Lock Amazon S3 objects](#)

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Set the object lock configuration of a bucket.

```
/// <summary>
/// Enable object lock on an existing bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to modify.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableObjectLockOnBucket(string bucketName)
{
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
```

```

        {
            ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
        },
    };

    var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
    Console.WriteLine($"{Environment.NewLine}\tAdded an object lock policy to bucket
{bucketName}.");
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
    return false;
}
}

```

Set the default retention period of a bucket.

```

/// <summary>
/// Set or modify a retention period on an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to modify.</param>
/// <param name="retention">The retention mode.</param>
/// <param name="retainUntilDate">The date for retention until.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
{
    var enabledString = enableObjectLock ? "Enabled" : "Disabled";
    var timeDifference = retainUntilDate.Subtract(DateTime.Now);
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
            }
        }
    }
}

```

```

        Status = VersionStatus.Enabled
    }
});

var request = new PutObjectLockConfigurationRequest()
{
    BucketName = bucketName,
    ObjectLockConfiguration = new ObjectLockConfiguration()
    {
        ObjectLockEnabled = new ObjectLockEnabled(enabledString),
        Rule = new ObjectLockRule()
        {
            DefaultRetention = new DefaultRetention()
            {
                Mode = retention,
                Days = timeDifference.Days // Can be specified in
days or years but not both.
            }
        }
    }
};

var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
Console.WriteLine($"\\tAdded a default retention to bucket
{bucketName}.");
return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tError modifying object lock: '{ex.Message}'");
    return false;
}
}

```

- For API details, see [PutObjectLockConfiguration](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To set an object lock configuration on a bucket

The following `put-object-lock-configuration` example sets a 50-day object lock on the specified bucket.

```
aws s3api put-object-lock-configuration \  
  --bucket amzn-s3-demo-bucket-with-object-lock \  
  --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule":  
  { "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 50 } } }'
```

This command produces no output.

- For API details, see [PutObjectLockConfiguration](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Set the object lock configuration of a bucket.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {
```

```
S3Client  *s3.Client
S3Manager *manager.Uploader
}

// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
string) error {
    // Versioning must be enabled on the bucket before object locking is enabled.
    verInput := &s3.PutBucketVersioningInput{
        Bucket: aws.String(bucket),
        VersioningConfiguration: &types.VersioningConfiguration{
            MFADelete: types.MFADeleteDisabled,
            Status:    types.BucketVersioningStatusEnabled,
        },
    }
    _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
        return err
    }

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: types.ObjectLockEnabledEnabled,
        },
    }
    _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }

    return err
}
```

Set the default retention period of a bucket.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// ModifyDefaultBucketRetention modifies the default retention period of an
// existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    }
```

```
    },  
    },  
    },  
}  
_, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)  
if err != nil {  
    var noBucket *types.NoSuchBucket  
    if errors.As(err, &noBucket) {  
        log.Printf("Bucket %s does not exist.\n", bucket)  
        err = noBucket  
    }  
}  
  
return err  
}
```

- For API details, see [PutObjectLockConfiguration](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Set the object lock configuration of a bucket.

```
// Enable object lock on an existing bucket.  
public void enableObjectLockOnBucket(String bucketName) {  
    try {  
        VersioningConfiguration versioningConfiguration =  
VersioningConfiguration.builder()  
            .status(BucketVersioningStatus.ENABLED)  
            .build();  
  
        PutBucketVersioningRequest putBucketVersioningRequest =  
PutBucketVersioningRequest.builder()
```

```

        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

// Enable versioning on the bucket.
getClient().putBucketVersioning(putBucketVersioningRequest);
PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .build())
    .build();

getClient().putObjectLockConfiguration(request);
System.out.println("Successfully enabled object lock on
"+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
    }
}

```

Set the default retention period of a bucket.

```

// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .mfaDelete(MFADelete.DISABLED)
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    getClient().putBucketVersioning(versioningRequest);
    DefaultRetention retention = DefaultRetention.builder()

```



```
        .days(1)
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

ObjectLockRule lockRule = ObjectLockRule.builder()
    .defaultRetention(rention)
    .build();

ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
    .objectLockEnabled(ObjectLockEnabled.ENABLED)
    .rule(lockRule)
    .build();

PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(objectLockConfiguration)
    .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
    System.out.println("Added a default retention to bucket "+bucketName
+ ".");
}
```

- For API details, see [PutObjectLockConfiguration](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Set the object lock configuration of a bucket.

```
import {
```

```
PutObjectLockConfigurationCommand,
S3Client,
S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Enable S3 Object Lock for an Amazon S3 bucket.
 * After you enable Object Lock on a bucket, you can't
 * disable Object Lock or suspend versioning for that bucket.
 * @param {{ bucketName: string, enabled: boolean }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
    bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
    },
  });

  try {
    await client.send(command);
    console.log(`Object Lock for "${bucketName}" enabled.`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
        bucket "${bucketName}". The bucket doesn't exist.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
        bucket "${bucketName}". ${caught.name}: ${caught.message}`
      );
    } else {
      throw caught;
    }
  }
};
```

```
// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Set the default retention period of a bucket.

```
import {
  PutObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Change the default retention settings for an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, retentionDays: string }}
 */
```

```

export const main = async ({ bucketName, retentionDays }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: bucketName,
        // The Object Lock configuration that you want to apply to the specified
        bucket.
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
          Rule: {
            // The default Object Lock retention mode and period that you want to
            apply
            // to new objects placed in the specified bucket. Bucket settings
            require
            // both a mode and a period. The period can be either Days or Years
            but
            // you must select one.
            DefaultRetention: {
              // In governance mode, users can't overwrite or delete an object
              version
              // or alter its lock settings unless they have special permissions.
              With
              // governance mode, you protect objects against being deleted by
              most users,
              // but you can still grant some users permission to alter the
              retention settings
              // or delete the objects if necessary.
              Mode: "GOVERNANCE",
              Days: Number.parseInt(retentionDays),
            },
          },
        },
      }),
    );
    console.log(
      `Set default retention mode to "GOVERNANCE" with a retention period of
      ${retentionDays} day(s).`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    )

```

```
    ) {
      console.error(
        `Error from S3 while setting the default object retention for a bucket.
The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting the default object retention for a bucket.
${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    retentionDays: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
```

```
        console.error(errors.join("\n"));
    }
}
```

- For API details, see [PutObjectLockConfiguration](#) in *Amazon SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Put object lock configuration.

```
s3_client.put_object_lock_configuration(
    Bucket=bucket,
    ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":
{}}
)
```

- For API details, see [PutObjectLockConfiguration](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

TRY.
  " Example: Enable object lock with default retention
  " iv_enabled = 'Enabled'
  lo_s3->putobjectlockconfiguration(
    iv_bucket = iv_bucket_name
    io_objectlockconfiguration = NEW /aws1/cl_s3_objectlockconf(
      iv_objectlockenabled = iv_enabled ) ).
  MESSAGE 'Object lock configuration set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- For API details, see [PutObjectLockConfiguration](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutObjectRetention with an Amazon SDK or CLI

The following code examples show how to use PutObjectRetention.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Lock Amazon S3 objects](#)

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/// <summary>
```

```

    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }
}

```

- For API details, see [PutObjectRetention](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To set an object retention configuration for an object

The following `put-object-retention` example sets an object retention configuration for the specified object until 2025-01-01.

```
aws s3api put-object-retention \  
  --bucket amzn-s3-demo-bucket-with-object-lock \  
  --key doc1.rtf \  
  --retention '{ "Mode": "GOVERNANCE", "RetainUntilDate":  
  "2025-01-01T00:00:00" }'
```

This command produces no output.

- For API details, see [PutObjectRetention](#) in *Amazon CLI Command Reference*.

Go

SDK for Go V2

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import (  
  "bytes"  
  "context"  
  "errors"  
  "fmt"  
  "log"  
  "time"  
  
  "github.com/aws/aws-sdk-go-v2/aws"  
  "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
  "github.com/aws/aws-sdk-go-v2/service/s3"
```

```
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
    string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
    error {
    input := &s3.PutObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        Retention: &types.ObjectLockRetention{
            Mode:          retentionMode,
            RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
        },
        BypassGovernanceRetention: aws.Bool(true),
    }

    _, err := actor.S3Client.PutObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}
```

- For API details, see [PutObjectRetention](#) in *Amazon SDK for Go API Reference*.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time
    zone.
    ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

    // Define a formatter for human-readable output.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Format the ZonedDateTime object to a human-readable date string.
    String humanReadableDate = formatter.format(zonedDateTime);

    // Print the formatted date string.
    System.out.println("Formatted Date: " + humanReadableDate);
    ObjectLockRetention retention = ObjectLockRetention.builder()
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .retainUntilDate(futureInstant)
        .build();

    PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .retention(retention)
        .build();
}
```

```
getClient().putObjectRetention(retentionRequest);
System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}
```

- For API details, see [PutObjectRetention](#) in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import {
  PutObjectRetentionCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Place a 24-hour retention period on an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: key,
    BypassGovernanceRetention: false,
    Retention: {
      // In governance mode, users can't overwrite or delete an object version
      // or alter its lock settings unless they have special permissions. With
      // governance mode, you protect objects against being deleted by most
      users,
      // but you can still grant some users permission to alter the retention
      settings
      // or delete the objects if necessary.
    }
  });
```

```
        Mode: "GOVERNANCE",
        RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
    },
});

try {
    await client.send(command);
    console.log("Object Retention settings updated.");
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while modifying the governance mode and retention period
on an object. The bucket doesn't exist.`
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while modifying the governance mode and retention period
on an object. ${caught.name}: ${caught.message}`
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
        key: {
            type: "string",
            required: true,
        },
    };
};
```

```
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- For API details, see [PutObjectRetention](#) in *Amazon SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: The command enables governance retention mode until the date '31st Dec 2019 00:00:00' for 'testfile.txt' object in the given S3 bucket.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- For API details, see [PutObjectRetention](#) in *Amazon Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: The command enables governance retention mode until the date '31st Dec 2019 00:00:00' for 'testfile.txt' object in the given S3 bucket.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- For API details, see [PutObjectRetention](#) in *Amazon Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Put an object retention.

```
s3_client.put_object_retention(  
    Bucket=bucket,  
    Key=key,  
    VersionId=version_id,  
    Retention={"Mode": "GOVERNANCE", "RetainUntilDate":  
far_future_date},  
    BypassGovernanceRetention=True,  
)
```

- For API details, see [PutObjectRetention](#) in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
TRY.  
    " Example: Set retention mode to GOVERNANCE for 30 days  
    " iv_mode = 'GOVERNANCE'  
    " iv_retain_date should be a timestamp in the future  
    lo_s3->putobjectretention(  
        iv_bucket = iv_bucket_name
```

```

        iv_key = iv_object_key
        io_retention = NEW /aws1/cl_s3_objectlockret(
            iv_mode = iv_mode
            iv_retainuntildate = iv_retain_date )
        iv_bypassgovernanceretention = abap_true ).
    MESSAGE 'Object retention set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

- For API details, see [PutObjectRetention](#) in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use RestoreObject with an Amazon SDK or CLI

The following code examples show how to use RestoreObject.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to restore an archived object in an Amazon

```



```

/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class RestoreArchivedObject
{
    public static void Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string objectKey = "archived-object.txt";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        IAmazonS3 client = new AmazonS3Client(bucketRegion);
        RestoreObjectAsync(client, bucketName, objectKey).Wait();
    }

    /// <summary>
    /// This method restores an archived object from an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// RestoreObjectAsync.</param>
    /// <param name="bucketName">A string representing the name of the
    /// bucket where the object was located before it was archived.</param>
    /// <param name="objectKey">A string representing the name of the
    /// archived object to restore.</param>
    public static async Task RestoreObjectAsync(IAmazonS3 client, string
bucketName, string objectKey)
    {
        try
        {
            var restoreRequest = new RestoreObjectRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Days = 2,
            };
            RestoreObjectResponse response = await
client.RestoreObjectAsync(restoreRequest);

            // Check the status of the restoration.
            await CheckRestorationStatusAsync(client, bucketName, objectKey);
        }
        catch (AmazonS3Exception amazonS3Exception)

```

```

        {
            Console.WriteLine($"Error: {amazonS3Exception.Message}");
        }
    }

    /// <summary>
    /// This method retrieves the status of the object's restoration.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// GetObjectMetadataAsync.</param>
    /// <param name="bucketName">A string representing the name of the Amazon
    /// S3 bucket which contains the archived object.</param>
    /// <param name="objectKey">A string representing the name of the
    /// archived object you want to restore.</param>
    public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
    {
        GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
        };

        GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

        var restStatus = response.RestoreInProgress ? "in-progress" :
"finished or failed";
        Console.WriteLine($"Restoration status: {restStatus}");
    }
}

```

- For API details, see [RestoreObject](#) in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To create a restore request for an object

The following `restore-object` example restores the specified Amazon S3 Glacier object for the bucket `my-glacier-bucket` for 10 days.

```
aws s3api restore-object \  
  --bucket my-glacier-bucket \  
  --key doc1.rtf \  
  --restore-request Days=10
```

This command produces no output.

- For API details, see [RestoreObject](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.RestoreRequest;  
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;  
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.Tier;  
  
/*  
 * For more information about restoring an object, see "Restoring an archived  
 * object" at  
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html  
 *  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 */
```

```

    * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
    */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <expectedBucketOwner>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value of
Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
obtain this value from the AWS Management Console).\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }

    /**
     * Restores an S3 object from the Glacier storage class.
     *
     * @param s3 an instance of the {@link S3Client} to be used
for interacting with Amazon S3
     * @param bucketName the name of the S3 bucket where the object is
stored
     * @param keyName the key (object name) of the S3 object to be
restored
    */
}

```

```

    * @param expectedBucketOwner the AWS account ID of the expected bucket
    owner
    */
    public static void restoreS3Object(S3Client s3, String bucketName, String
    keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                .days(10)

            .glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                .expectedBucketOwner(expectedBucketOwner)
                .bucket(bucketName)
                .key(keyName)
                .restoreRequest(restoreRequest)
                .build();

            s3.restoreObject(objectRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- For API details, see [RestoreObject](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use SelectObjectContent with an Amazon SDK or CLI

The following code examples show how to use SelectObjectContent.

CLI

Amazon CLI

To filter the contents of an Amazon S3 object based on an SQL statement

The following `select-object-content` example filters the object `my-data-file.csv` with the specified SQL statement and sends output to a file.

```
aws s3api select-object-content \
  --bucket amzn-s3-demo-bucket \
  --key my-data-file.csv \
  --expression "select * from s3object limit 100" \
  --expression-type 'SQL' \
  --input-serialization '{"CSV": {}, "CompressionType": "NONE"}' \
  --output-serialization '{"CSV": {}}' "output.csv"
```

This command produces no output.

- For API details, see [SelectObjectContent](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

The following example shows a query using a JSON object. The [complete example](#) also shows the use of a CSV object.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
```

```
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import
    software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
        LoggerFactory.getLogger(SelectObjectContentExample.class);
    static String BUCKET_NAME = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/
countries/master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
        SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        }
```

```

    } catch (SdkException e) {
        logger.error(e.getMessage(), e);
        System.exit(1);
    } finally {
        SelectObjectContentExample.tearDown();
    }
}

EventStreamInfo runSelectObjectContentMethodForJSON() {
    // Set up request parameters.
    final String queryExpression = "select * from s3object[*][*] c where
c.area < 350000";
    final String fileType = FILE_JSON;

    InputSerialization inputSerialization = InputSerialization.builder()
        .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
        .compressionType(CompressionType.NONE)
        .build();

    OutputSerialization outputSerialization = OutputSerialization.builder()
        .json(JSONOutput.builder().recordDelimiter(null).build())
        .build();

    // Build the SelectObjectContentRequest.
    SelectObjectContentRequest select = SelectObjectContentRequest.builder()
        .bucket(BUCKET_NAME)
        .key(FILE_JSON)
        .expression(queryExpression)
        .expressionType(ExpressionType.SQL)
        .inputSerialization(inputSerialization)
        .outputSerialization(outputSerialization)
        .build();

    EventStreamInfo eventStreamInfo = new EventStreamInfo();
    // Call the selectObjectContent method with the request and a response
    handler.
    // Supply an EventStreamInfo object to the response handler to gather
    records and information from the response.
    s3AsyncClient.selectObjectContent(select,
    buildResponseHandler(eventStreamInfo)).join();

    // Log out information gathered while processing the response stream.
    long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record
->

```



```

        record.split("\n").length
    ).sum();
    logger.info("Total records {}: {}", fileType, recordCount);
    logger.info("Visitor onRecords for fileType {} called {} times",
fileType, eventStreamInfo.getCountOnRecordsCalled());
    logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
    logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
    return eventStreamInfo;
}

static SelectObjectContentResponseHandler
buildResponseHandler(EventStreamInfo eventStreamInfo) {
    // Use a Visitor to process the response stream. This visitor logs
information and gathers details while processing.
    final SelectObjectContentResponseHandler.Visitor visitor =
SelectObjectContentResponseHandler.Visitor.builder()
        .onRecords(r -> {
            logger.info("Record event received.");
            eventStreamInfo.addRecord(r.payload().asUtf8String());
            eventStreamInfo.incrementOnRecordsCalled();
        })
        .onCont(ce -> {
            logger.info("Continuation event received.");
            eventStreamInfo.incrementContinuationEvents();
        })
        .onProgress(pe -> {
            Progress progress = pe.details();
            logger.info("Progress event received:\n bytesScanned:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                progress.bytesScanned(),
                progress.bytesProcessed(),
                progress.bytesReturned());
        })
        .onEnd(ee -> logger.info("End event received.))
        .onStats(se -> {
            logger.info("Stats event received.");
            eventStreamInfo.addStats(se.details());
        })
        .build();

    // Build the SelectObjectContentResponseHandler with the visitor that
processes the stream.

```

```
        return SelectObjectContentResponseHandler.builder()
            .subscriber(visitor).build();
    }

    // The EventStreamInfo class is used to store information gathered while
    // processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
        private Integer countContinuationEvents = 0;
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
        }

        void incrementContinuationEvents() {
            countContinuationEvents++;
        }

        void addRecord(String record) {
            records.add(record);
        }

        void addStats(Stats stats) {
            this.stats = stats;
        }

        public List<String> getRecords() {
            return records;
        }

        public Integer getCountOnRecordsCalled() {
            return countOnRecordsCalled;
        }

        public Integer getCountContinuationEvents() {
            return countContinuationEvents;
        }

        public Stats getStats() {
            return stats;
        }
    }
}
```

- For API details, see [SelectObjectContent](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use UploadPart with an Amazon SDK or CLI

The following code examples show how to use UploadPart.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Use checksums](#)
- [Work with Amazon S3 object integrity](#)

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
    checksum algorithm,
                           ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
```

```

    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
    &bucket,

                                                         const Aws::String &key,
                                                         const Aws::String

    &uploadID,

                                                         int partNumber,

    Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,

                                                         const Aws::String

    &calculatedHash,

                                                         const

    std::shared_ptr<Aws::IOStream> &body,

                                                         const Aws::S3::S3Client

    &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }
}

```

```
    }  
  }  
  
  return client.UploadPart(request);  
}
```

- For API details, see [UploadPart](#) in *Amazon SDK for C++ API Reference*.

CLI

Amazon CLI

The following command uploads the first part in a multipart upload initiated with the `create-multipart-upload` command:

```
aws s3api upload-part --bucket amzn-s3-demo-bucket --key  
'multipart/01' --part-number 1 --body part01 --upload-id  
"dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6YjLx.cph0gtNBtJ8P"
```

The body option takes the name or path of a local file for upload (do not use the `file://` prefix). The minimum part size is 5 MB. Upload ID is returned by `create-multipart-upload` and can also be retrieved with `list-multipart-uploads`. Bucket and key are specified when you create the multipart upload.

Output:

```
{  
  "ETag": "\"e868e0f4719e394144ef36531ee6824c\""  
}
```

Save the ETag value of each part for later. They are required to complete the multipart upload.

- For API details, see [UploadPart](#) in *Amazon CLI Command Reference*.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;

    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
```

```
        .build(),
    );
}
```

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;
```

- For API details, see [UploadPart](#) in *Amazon SDK for Rust API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use UploadPartCopy with an Amazon SDK or CLI

The following code examples show how to use UploadPartCopy.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Perform a multipart copy](#)

CLI

Amazon CLI

To upload part of an object by copying data from an existing object as the data source

The following upload-part-copy example uploads a part by copying data from an existing object as a data source.

```
aws s3api upload-part-copy \  
  --bucket amzn-s3-demo-bucket \  
  --key "Map_Data_June.mp4" \  
  --copy-source "amzn-s3-demo-bucket/copy_of_Map_Data_June.mp4" \  
  --part-number 1 \  
  --upload-  
id "bq0tdE1CDpWQYRPLHuNG50xAT6pA5D.m_RiBy0gg0H6b13pVRY7QjvLlF75iFdJqp_2wztk5hvpUM2SesXgrz"
```

Output:

```
{  
  "CopyPartResult": {  
    "LastModified": "2019-12-13T23:16:03.000Z",  
    "ETag": "\"711470fc377698c393d94aed6305e245\""  
  }  
}
```

- For API details, see [UploadPartCopy](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(toBucket)
        .key(key)
        .build();

    getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
        .thenApply(createMultipartUploadResponse -> {
            String uploadId = createMultipartUploadResponse.uploadId();
            System.out.println("Upload ID: " + uploadId);

            UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                .sourceBucket(bucketName)
                .destinationBucket(toBucket)
                .sourceKey(key)
                .destinationKey(key)
                .uploadId(uploadId) // Use the valid uploadId.
                .partNumber(1) // Ensure the part number is correct.
                .copySourceRange("bytes=0-1023") // Adjust range as needed
                .build();

            return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
        })
        .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
        .whenComplete((uploadPartCopyResponse, exception) -> {
            if (exception != null) {
                // Handle any exceptions.
                logger.error("Error during upload part copy: " +
exception.getMessage());
            }
        });
}
```

```
        } else {  
            // Successfully completed the upload part copy.  
            System.out.println("Upload Part Copy completed successfully.  
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());  
        }  
    });  
    return null;  
}
```

- For API details, see [UploadPartCopy](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Scenarios for Amazon S3 using Amazon SDKs

The following code examples show you how to implement common scenarios in Amazon S3 with Amazon SDKs. These scenarios show you how to accomplish specific tasks by calling multiple functions within Amazon S3 or combined with other Amazon Web Services services. Each scenario includes a link to the complete source code, where you can find instructions on how to set up and run the code.

Scenarios target an intermediate level of experience to help you understand service actions in context.

Examples

- [Check if a bucket exists](#)
- [Convert text to speech and back to text using an Amazon SDK](#)
- [Create a presigned URL for Amazon S3 using an Amazon SDK](#)
- [Create a photo asset management application that lets users manage photos using labels](#)
- [A web page that lists Amazon S3 objects using an Amazon SDK](#)
- [Create an Amazon Textract explorer application](#)
- [Delete all objects in a given Amazon S3 bucket using an Amazon SDK](#)
- [Delete incomplete multipart uploads to Amazon S3 using an Amazon SDK](#)
- [Detect PPE in images with Amazon Rekognition using an Amazon SDK](#)

- [Detect entities in text extracted from an image using an Amazon SDK](#)
- [Detect faces in an image using an Amazon SDK](#)
- [Detect objects in images with Amazon Rekognition using an Amazon SDK](#)
- [Detect people and objects in a video with Amazon Rekognition using an Amazon SDK](#)
- [Download S3 'directories' from an Amazon Simple Storage Service \(Amazon S3\) bucket](#)
- [Download all objects in an Amazon Simple Storage Service \(Amazon S3\) bucket to a local directory](#)
- [Download a stream of unknown size from an Amazon S3 object using an Amazon SDK](#)
- [Get an Amazon S3 object from a Multi-Region Access Point by using an Amazon SDK](#)
- [Get an object from an Amazon S3 bucket using an Amazon SDK, specifying an If-Modified-Since header](#)
- [Get started with encryption for Amazon S3 objects using an Amazon SDK](#)
- [Get started with tags for Amazon S3 objects using an Amazon SDK](#)
- [Getting started with Amazon Athena](#)
- [Getting started with Amazon EMR](#)
- [Getting started with Amazon S3](#)
- [Getting started with Amazon SageMaker Feature Store](#)
- [Getting started with Amazon Textract](#)
- [Getting started with Config](#)
- [Work with Amazon S3 object lock features using an Amazon SDK](#)
- [Make Amazon S3 conditional requests using an Amazon SDK](#)
- [Manage access control lists \(ACLs\) for Amazon S3 buckets using an Amazon SDK](#)
- [Manage large Amazon SQS messages using Amazon S3 with an Amazon SDK](#)
- [Manage versioned Amazon S3 objects in batches with a Lambda function using an Amazon SDK](#)
- [Parse Amazon S3 URIs using an Amazon SDK](#)
- [Perform a multipart copy of an Amazon S3 object using an Amazon SDK](#)
- [Receive and process Amazon S3 event notifications by using an Amazon SDK](#)
- [Save EXIF and other image information using an Amazon SDK](#)
- [Send S3 event notifications to Amazon EventBridge using an Amazon SDK](#)
- [Track an Amazon S3 object upload or download using an Amazon SDK](#)
- [Transform data for your application with S3 Object Lambda](#)

- [Example approaches for unit and integration testing with an Amazon SDK](#)
- [Recursively upload a local directory to an Amazon Simple Storage Service \(Amazon S3\) bucket](#)
- [Upload or download large files to and from Amazon S3 using an Amazon SDK](#)
- [Upload a stream of unknown size to an Amazon S3 object using an Amazon SDK](#)
- [Use checksums to work with an Amazon S3 object using an Amazon SDK](#)
- [Work with Amazon S3 object integrity features using an Amazon SDK](#)
- [Work with Amazon S3 versioned objects using an Amazon SDK](#)

Check if a bucket exists

The following code example shows how to check if a bucket exists.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

You can use the following `doesBucketExists` method as a replacement for the the SDK for Java V1 [AmazonS3Client#doesBucketExistV2\(String\)](#) method.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.http.HttpStatusCode;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.utils.Validate;

public class DoesBucketExist {
    private static final Logger logger =
        LoggerFactory.getLogger(DoesBucketExist.class);

    public static void main(String[] args) {
        DoesBucketExist doesBucketExist = new DoesBucketExist();
    }
}
```

```

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with the
        bucket name that you want to check.

        boolean exists = doesBucketExist.doesBucketExist(bucketName,
s3SyncClient);
        logger.info("Bucket exists: {}", exists);
    }

    /**
     * Checks if the specified bucket exists. Amazon S3 buckets are named in a
     global namespace; use this method to
     * determine if a specified bucket name already exists, and therefore can't
     be used to create a new bucket.
     * <p>
     * Internally this method uses the <a
     * href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
     services/s3/
     S3Client.html#getBucketAcl(java.util.function.Consumer)">S3Client.getBucketAcl(String)</
     a>
     * operation to determine whether the bucket exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's <a
     * href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/
     com/amazonaws/services/s3/AmazonS3Client.html#doesBucketExistV2-
     java.lang.String-">AmazonS3Client#doesBucketExistV2(String)</a>.
     *
     * @param bucketName The name of the bucket to check.
     * @param s3SyncClient An <code>S3Client</code> instance. The method checks
     for the bucket in the AWS Region
     * configured on the instance.
     * @return The value true if the specified bucket exists in Amazon S3; the
     value false if there is no bucket in
     * Amazon S3 with that name.
     */
    public boolean doesBucketExist(String bucketName, S3Client s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
empty string.", "");
            s3SyncClient.getBucketAcl(r -> r.bucket(bucketName));
            return true;
        } catch (AwsServiceException ase) {

```

```
        // A redirect error or an AccessDenied exception means the bucket
        exists but it's not in this region
        // or we don't have permissions to it.
        if ((ase.statusCode() == HttpStatusCode.MOVED_PERMANENTLY) ||
            "AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
            return true;
        }
        if (ase.statusCode() == HttpStatusCode.NOT_FOUND) {
            return false;
        }
        throw ase;
    }
}
```

- For API details, see [GetBucketAcl](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Convert text to speech and back to text using an Amazon SDK

The following code example shows how to:

- Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file.
- Upload the audio file to an Amazon S3 bucket.
- Use Amazon Transcribe to convert the audio file to text.
- Display the text.

Rust

SDK for Rust

Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file, upload the audio file to an Amazon S3 bucket, use Amazon Transcribe to convert that audio file to text, and display the text.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Polly
- Amazon S3
- Amazon Transcribe

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Create a presigned URL for Amazon S3 using an Amazon SDK

The following code examples show how to create a presigned URL for Amazon S3 and upload an object.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Generate a presigned URL that can perform an Amazon S3 action for a limited time.

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

public class GenPresignedUrl
{
    public static void Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string objectKey = "sample.txt";
```

```

        // Specify how long the presigned URL lasts, in hours
        const double timeoutDuration = 12;

        // Specify the AWS Region of your Amazon S3 bucket. If it is
        // different from the Region defined for the default user,
        // pass the Region to the constructor for the client. For
        // example: new AmazonS3Client(RegionEndpoint.USEast1);

        // If using the Region us-east-1, and server-side encryption with AWS
        KMS, you must specify Signature Version 4.
        // Region us-east-1 defaults to Signature Version 2 unless explicitly
        set to Version 4 as shown below.
        // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingAWSSDK.html#specify-signature-version
        // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/TAWSConfigsS3.html
        AWSConfigsS3.UseSignatureVersion4 = true;
        IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

        string urlString = GeneratePresignedURL(s3Client, bucketName,
        objectKey, timeoutDuration);
        Console.WriteLine($"The generated URL is: {urlString}.");
    }

    /// <summary>
    /// Generate a presigned URL that can be used to access the file named
    /// in the objectKey parameter for the amount of time specified in the
    /// duration parameter.
    /// </summary>
    /// <param name="client">An initialized S3 client object used to call
    /// the GetPresignedUrl method.</param>
    /// <param name="bucketName">The name of the S3 bucket containing the
    /// object for which to create the presigned URL.</param>
    /// <param name="objectKey">The name of the object to access with the
    /// presigned URL.</param>
    /// <param name="duration">The length of time for which the presigned
    /// URL will be valid.</param>
    /// <returns>A string representing the generated presigned URL.</returns>
    public static string GeneratePresignedURL(IAmazonS3 client, string
    bucketName, string objectKey, double duration)
    {
        string urlString = string.Empty;
        try

```



```
        {
            var request = new GetPreSignedUrlRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Expires = DateTime.UtcNow.AddHours(duration),
            };
            urlString = client.GetPreSignedURL(request);
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error: '{ex.Message}'");
        }

        return urlString;
    }
}
```

Generate a presigned URL and perform an upload using that URL.

```
using System;
using System.IO;
using System.Net.Http;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to upload an object to an Amazon Simple Storage
/// Service (Amazon S3) bucket using a presigned URL. The code first
/// creates a presigned URL and then uses it to upload an object to an
/// Amazon S3 bucket using that URL.
/// </summary>
public class UploadUsingPresignedURL
{
    private static HttpClient httpClient = new HttpClient();

    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
```

```

        string keyName = "samplefile.txt";
        string filePath = $"source\\{keyName}";

        // Specify how long the signed URL will be valid in hours.
        double timeoutDuration = 12;

        // Specify the AWS Region of your Amazon S3 bucket. If it is
        // different from the Region defined for the default user,
        // pass the Region to the constructor for the client. For
        // example: new AmazonS3Client(RegionEndpoint.USEast1);

        // If using the Region us-east-1, and server-side encryption with AWS
        KMS, you must specify Signature Version 4.
        // Region us-east-1 defaults to Signature Version 2 unless explicitly
        set to Version 4 as shown below.
        // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingAWSSDK.html#specify-signature-version
        // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/TAWSConfigsS3.html
        AWSConfigsS3.UseSignatureVersion4 = true;
        IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);

        var url = GeneratePreSignedURL(client, bucketName, keyName,
        timeoutDuration);
        var success = await UploadObject(filePath, url);

        if (success)
        {
            Console.WriteLine("Upload succeeded.");
        }
        else
        {
            Console.WriteLine("Upload failed.");
        }
    }

    /// <summary>
    /// Uploads an object to an Amazon S3 bucket using the presigned URL
    passed in
    /// the url parameter.
    /// </summary>
    /// <param name="filePath">The path (including file name) to the local
    /// file you want to upload.</param>
    /// <param name="url">The presigned URL that will be used to upload the

```

```

    /// file to the Amazon S3 bucket.</param>
    /// <returns>A Boolean value indicating the success or failure of the
    /// operation, based on the HttpResponseMessage.</returns>
    public static async Task<bool> UploadObject(string filePath, string url)
    {
        using var streamContent = new StreamContent(
            new FileStream(filePath, FileMode.Open, FileAccess.Read));

        var response = await httpClient.PutAsync(url, streamContent);
        return response.IsSuccessStatusCode;
    }

    /// <summary>
    /// Generates a presigned URL which will be used to upload an object to
    /// an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// GetPreSignedURL.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
    /// presigned URL will point.</param>
    /// <param name="objectKey">The name of the file that will be uploaded.</
param>
    /// <param name="duration">How long (in hours) the presigned URL will
    /// be valid.</param>
    /// <returns>The generated URL.</returns>
    public static string GeneratePreSignedURL(
        IAmazonS3 client,
        string bucketName,
        string objectKey,
        double duration)
    {
        var request = new GetPreSignedUrlRequest
        {
            BucketName = bucketName,
            Key = objectKey,
            Verb = HttpVerb.PUT,
            Expires = DateTime.UtcNow.AddHours(duration),
        };

        string url = client.GetPreSignedURL(request);
        return url;
    }

```

```
}
```

Amazon SDK for .NET (v4)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create and use presigned POST URLs for direct browser uploads.

```
/// <summary>
/// Scenario demonstrating the complete workflow for presigned POST URLs:
/// 1. Create an S3 bucket
/// 2. Create a presigned POST URL
/// 3. Upload a file using the presigned POST URL
/// 4. Clean up resources
/// </summary>
public class CreatePresignedPostBasics
{
    public static ILogger<CreatePresignedPostBasics> _logger = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static UiMethods _uiMethods = null!;
    public static IHttpClientFactory _httpClientFactory = null!;
    public static bool _isInteractive = true;
    public static string? _bucketName;
    public static string? _objectKey;

    /// <summary>
    /// Set up the services and logging.
    /// </summary>
    /// <param name="host">The IHost instance.</param>
    public static void SetupServices(IHost host)
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });
        _logger = new Logger<CreatePresignedPostBasics>(loggerFactory);
    }
}
```

```

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();
        _httpClientFactory =
host.Services.GetRequiredService<IHttpClientFactory>();
        _uiMethods = new UiMethods();
    }

    /// <summary>
    /// Perform the actions defined for the Amazon S3 Presigned POST scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        _isInteractive = !args.Contains("--non-interactive");

        // Set up dependency injection for Amazon S3
        using var host =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddHttpClient()
            )
            .Build();

        SetUpServices(host);

        try
        {
            // Display overview
            _uiMethods.DisplayOverview();
            _uiMethods.PressEnter(_isInteractive);

            // Step 1: Create bucket
            await CreateBucketAsync();
            _uiMethods.PressEnter(_isInteractive);

            // Step 2: Create presigned URL
            _uiMethods.DisplayTitle("Step 2: Create presigned POST URL");
            var response = await CreatePresignedPostAsync();
            _uiMethods.PressEnter(_isInteractive);

            // Step 3: Display URL and fields
            _uiMethods.DisplayTitle("Step 3: Presigned POST URL details");

```

```

        DisplayPresignedPostFields(response);
        _uiMethods.PressEnter(_isInteractive);

        // Step 4: Upload file
        _uiMethods.DisplayTitle("Step 4: Upload test file using presigned
POST URL");
        await UploadFileAsync(response);
        _uiMethods.PressEnter(_isInteractive);

        // Step 5: Verify file exists
        await VerifyFileExistsAsync();
        _uiMethods.PressEnter(_isInteractive);

        // Step 6: Cleanup
        _uiMethods.DisplayTitle("Step 6: Clean up resources");
        await CleanupAsync();

        _uiMethods.DisplayTitle("S3 Presigned POST Scenario completed
successfully!");
        _uiMethods.PressEnter(_isInteractive);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error in scenario");
        Console.WriteLine($"Error: {ex.Message}");

        // Attempt cleanup if there was an error
        if (!string.IsNullOrEmpty(_bucketName))
        {
            _uiMethods.DisplayTitle("Cleaning up resources after error");
            await _s3Wrapper.DeleteBucketAsync(_bucketName);
            Console.WriteLine($"Cleaned up bucket: {_bucketName}");
        }
    }
}

/// <summary>
/// Create an S3 bucket for the scenario.
/// </summary>
private static async Task CreateBucketAsync()
{
    _uiMethods.DisplayTitle("Step 1: Create an S3 bucket");

    // Generate a default bucket name for the scenario

```

```
        var defaultBucketName = $"presigned-post-demo-
{DateTime.Now:yyyyMMddHHmmss}".ToLower();

        // Prompt user for bucket name or use default in non-interactive mode
        _bucketName = _uiMethods.GetUserInput(
            $"Enter S3 bucket name (or press Enter for '{defaultBucketName}'):",
            defaultBucketName,
            _isInteractive);

        // Basic validation to ensure bucket name is not empty
        if (string.IsNullOrEmpty(_bucketName))
        {
            _bucketName = defaultBucketName;
        }

        Console.WriteLine($"Creating bucket: {_bucketName}");

        await _s3Wrapper.CreateBucketAsync(_bucketName);

        Console.WriteLine($"Successfully created bucket: {_bucketName}");
    }

    /// <summary>
    /// Create a presigned POST URL.
    /// </summary>
    private static async Task<CreatePresignedPostResponse>
    CreatePresignedPostAsync()
    {
        _objectKey = "example-upload.txt";
        var expiration = DateTime.UtcNow.AddMinutes(10); // Short expiration for
        the demo

        Console.WriteLine($"Creating presigned POST URL for {_bucketName}/
{_objectKey}");
        Console.WriteLine($"Expiration: {expiration} UTC");

        var s3Client = _s3Wrapper.GetS3Client();

        var response = await _s3Wrapper.CreatePresignedPostAsync(
            s3Client, _bucketName!, _objectKey, expiration);

        Console.WriteLine("Successfully created presigned POST URL");
        return response;
    }
}
```

```
}

/// <summary>
/// Upload a file using the presigned POST URL.
/// </summary>
private static async Task UploadFileAsync(CreatePresignedPostResponse
response)
{

    // Create a temporary test file to upload
    string testFilePath = Path.GetTempFileName();
    string testContent = "This is a test file for the S3 presigned POST
scenario.";

    await File.WriteAllTextAsync(testFilePath, testContent);
    Console.WriteLine($"Created test file at: {testFilePath}");

    // Upload the file using the presigned POST URL
    Console.WriteLine("\nUploading file using the presigned POST URL...");
    var uploadResult = await UploadFileWithPresignedPostAsync(response,
testFilePath);

    // Display the upload result
    if (uploadResult.Success)
    {
        Console.WriteLine($"Upload successful! Status code:
{uploadResult.StatusCode}");
    }
    else
    {
        Console.WriteLine($"Upload failed with status code:
{uploadResult.StatusCode}");
        Console.WriteLine($"Error: {uploadResult.Response}");
        throw new Exception("File upload failed");
    }

    // Clean up the temporary file
    File.Delete(testFilePath);
    Console.WriteLine("Temporary file deleted");
}

/// <summary>
/// Helper method to upload a file using a presigned POST URL.
/// </summary>
```



```
private static async Task<(bool Success, HttpStatusCode StatusCode, string
Response)> UploadFileWithPresignedPostAsync(
    CreatePresignedPostResponse response,
    string filePath)
{
    try
    {
        _logger.LogInformation("Uploading file {filePath} using presigned
POST URL", filePath);

        using var httpClient = _httpClientFactory.CreateClient();
        using var formContent = new MultipartFormDataContent();

        // Add all the fields from the presigned POST response
        foreach (var field in response.Fields)
        {
            formContent.Add(new StringContent(field.Value), field.Key);
        }

        // Add the file content
        var fileStream = File.OpenRead(filePath);
        var fileName = Path.GetFileName(filePath);
        var fileContent = new StreamContent(fileStream);
        fileContent.Headers.ContentType = new MediaTypeHeaderValue("text/
plain");
        formContent.Add(fileContent, "file", fileName);

        // Send the POST request
        var httpResponse = await httpClient.PostAsync(response.Url,
formContent);
        var responseContent = await httpResponse.Content.ReadAsStringAsync();

        // Log and return the result
        _logger.LogInformation("Upload completed with status code
{statusCode}", httpResponse.StatusCode);

        return (httpResponse.IsSuccessStatusCode, httpResponse.StatusCode,
responseContent);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error uploading file");
        return (false, HttpStatusCode.InternalServerError, ex.Message);
    }
}
```

```
}

/// <summary>
/// Verify that the uploaded file exists in the S3 bucket.
/// </summary>
private static async Task VerifyFileExistsAsync()
{
    _uiMethods.DisplayTitle("Step 5: Verify uploaded file exists");

    Console.WriteLine($"Checking if file exists at {_bucketName}/
{_objectKey}...");

    try
    {
        var metadata = await _s3Wrapper.GetObjectMetadataAsync(_bucketName!,
            _objectKey!);

        Console.WriteLine($"File verification successful! File exists in the
bucket.");
        Console.WriteLine($"File size: {metadata.ContentLength} bytes");
        Console.WriteLine($"File type: {metadata.Headers.ContentType}");
        Console.WriteLine($"Last modified: {metadata.LastModified}");
    }
    catch (AmazonS3Exception ex) when (ex.StatusCode ==
        System.Net.HttpStatusCode.NotFound)
    {
        Console.WriteLine($"Error: File was not found in the bucket.");
        throw;
    }
}

private static void DisplayPresignedPostFields(CreatePresignedPostResponse
response)
{
    Console.WriteLine($"Presigned POST URL: {response.Url}");
    Console.WriteLine("Form fields to include:");

    foreach (var field in response.Fields)
    {
        Console.WriteLine($"  {field.Key}: {field.Value}");
    }
}

/// <summary>
```

```
/// Clean up resources created by the scenario.
/// </summary>
private static async Task CleanupAsync()
{
    if (!string.IsNullOrEmpty(_bucketName))
    {
        Console.WriteLine($"Deleting bucket {_bucketName} and its
contents...");
        bool result = await _s3Wrapper.DeleteBucketAsync(_bucketName);

        if (result)
        {
            Console.WriteLine("Bucket deleted successfully");
        }
        else
        {
            Console.WriteLine("Failed to delete bucket - it may have been
already deleted");
        }
    }
}
}
```

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Generate a pre-signed URL to download an object.

```
//! Routine which demonstrates creating a pre-signed URL to download an object
from an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
    \param bucketName: Name of the bucket.
    \param key: Name of an object key.
```

```

\param expirationSeconds: Expiration in seconds for pre-signed URL.
\param clientConfig: Aws client configuration.
\return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String
&bucketName,
                                                    const Aws::String &key,
                                                    uint64_t expirationSeconds,
                                                    const
Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_GET,
                                expirationSeconds);
}

```

Download using libcurl.

```

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
*userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
\param presignedURL: A pre-signed URL to get an object from a bucket.
\param resultString: A string to hold the result.
\return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedGetObjectUrl(const Aws::String &presignedURL,
                                                    Aws::String &resultString) {

    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

```

```

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();

    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
std::endl;
        return false;
    }

    return true;
}

```

Generate a pre-signed URL to upload an object.

```

//! Routine which demonstrates creating a pre-signed URL to upload an object to
an

```

```

//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
    \param bucketName: Name of the bucket.
    \param key: Name of an object key.
    \param clientConfig: Aws client configuration.
    \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String
    &bucketName,
                                                    const Aws::String &key,
                                                    uint64_t expirationSeconds,
                                                    const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_PUT,
                                expirationSeconds);
}

```

Upload using libcurl.

```

static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
    *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
    *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!

```

```

\param presignedURL: A pre-signed URL to put an object in a bucket.
\param data: Body of the putObject request.
\return bool: Function succeeded.
*/
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                  const Aws::String &data) {

    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {

```

```
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    std::string outString = outWriteString.str();
    if (outString.empty()) {
        std::cout << "Successfully put object." << std::endl;
        return true;
    } else {
        std::cout << "A server error was encountered, output:\n" << outString
                  << std::endl;
        return false;
    }
}
```


Go

SDK for Go V2

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create functions that wrap S3 presigning actions.

```
import (  
    "context"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
)  
  
// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign  
// actions  
// used in the examples.  
// It contains PresignClient, a client that is used to presign requests to Amazon  
// S3.  
// Presigned requests contain temporary credentials and can be made from any HTTP  
// client.  
type Presigner struct {  
    PresignClient *s3.PresignClient  
}  
  
// GetObject makes a presigned request that can be used to get an object from a  
// bucket.  
// The presigned request is valid for the specified number of seconds.  
func (presigner Presigner) GetObject(  
    ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)  
    (*v4.PresignedHTTPRequest, error) {
```

```

request, err := presigner.PresignClient.PresignGetObject(ctx,
&s3.GetObjectInput{
    Bucket: aws.String(bucketName),
    Key:    aws.String(objectKey),
}, func(opts *s3.PresignOptions) {
    opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
})
if err != nil {
    log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
        bucketName, objectKey, err)
}
return request, err
}

// PutObject makes a presigned request that can be used to put an object in a
// bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
    ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
(*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignPutObject(ctx,
&s3.PutObjectInput{
    Bucket: aws.String(bucketName),
    Key:    aws.String(objectKey),
}, func(opts *s3.PresignOptions) {
    opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
})
if err != nil {
    log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
        bucketName, objectKey, err)
}
return request, err
}

// DeleteObject makes a presigned request that can be used to delete an object
// from a bucket.
func (presigner Presigner) DeleteObject(ctx context.Context, bucketName string,
    objectKey string) (*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignDeleteObject(ctx,
&s3.DeleteObjectInput{

```

```

    Bucket: aws.String(bucketName),
    Key:     aws.String(objectKey),
})
if err != nil {
    log.Printf("Couldn't get a presigned request to delete object %v. Here's why:
    %v\n", objectKey, err)
}
return request, err
}

func (presigner Presigner) PresignPostObject(ctx context.Context, bucketName
    string, objectKey string, lifetimeSecs int64) (*s3.PresignedPostRequest, error)
{
    request, err := presigner.PresignClient.PresignPostObject(ctx,
    &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    }, func(options *s3.PresignPostOptions) {
        options.Expires = time.Duration(lifetimeSecs) * time.Second
    })
    if err != nil {
        log.Printf("Couldn't get a presigned post request to put %v:%v. Here's why: %v
        \n", bucketName, objectKey, err)
    }
    return request, nil
}

```

Run an interactive example that generates and uses presigned URLs to upload, download, and delete an S3 object.

```

import (
    "bytes"
    "context"
    "io"
    "log"
    "mime/multipart"
    "net/http"

```

```

"os"
"strings"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunPresigningScenario is an interactive example that shows you how to get
// presigned
// HTTP requests that you can use to move data into and out of Amazon Simple
// Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and
// can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
// 2. Use the net/http package to use the presigned request to upload a local
//    file to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the
//    object to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../..demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
// mocked
// during testing.
func RunPresigningScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner, httpRequester IHttpRequester) {
defer func() {
    if r := recover(); r != nil {
        log.Println("Something went wrong with the demo.")
    }
}()

```

```

_, isMock := questioner.(*demotools.MockQuestioner)
if isMock || questioner.AskBool("Do you want to see the full error message (y/n)?", "y") {
    log.Println(r)
}
}
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 presigning demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}
presignClient := s3.NewPresignClient(s3Client)
presigner := actions.Presigner{PresignClient: presignClient}

bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
    "you own or one you want to create:", demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
    demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
    demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(ctx, bucketName, uploadKey, 60)

```

```
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedPutRequest.Method,
presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
    panic(err)
}
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
    panic(err)
}
```

```
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Now we'll create a new request to put the same object using a
presigned post request")
questioner.Ask("Press Enter when you're ready.")
presignPostRequest, err := presigner.PresignPostObject(ctx, bucketName,
uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned post request to url %v with values %v\n",
presignPostRequest.URL, presignPostRequest.Values)
log.Println("Using net/http multipart to send the request...")
uploadFile, err = os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
multipartResponse, err := sendMultipartRequest(presignPostRequest.URL,
presignPostRequest.Values, uploadFile, uploadKey, httpRequester)
if err != nil {
    panic(err)
}
log.Printf("Presign post object %v with presigned URL returned %v.", uploadKey,
multipartResponse.StatusCode)

log.Println("Let's presign a request to delete the object.")
questioner.Ask("Press Enter when you're ready.")
presignedDelRequest, err := presigner.DeleteObject(ctx, bucketName, uploadKey)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedDelRequest.Method,
presignedDelRequest.URL)
log.Println("Using net/http to send the request...")
delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
if err != nil {
    panic(err)
}
```

```
log.Printf("%v object %v with presigned URL returned %v.\n",
presignedDelRequest.Method,
uploadKey, delResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

Define an HTTP request wrapper used by the example to make HTTP requests.

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
// during
// unit testing.
type IHttpRequester interface {
    Get(url string) (resp *http.Response, err error)
    Post(url, contentType string, body io.Reader) (resp *http.Response, err error)
    Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
    error)
    Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
// scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
    return http.Get(url)
}

func (httpReq HttpRequester) Post(url, contentType string, body io.Reader) (resp
    *http.Response, err error) {
    postRequest, err := http.NewRequest("POST", url, body)
    if err != nil {
        return nil, err
    }
    postRequest.Header.Set("Content-Type", contentType)
    return http.DefaultClient.Do(postRequest)
}
```



```
func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
    (resp *http.Response, err error) {
    putRequest, err := http.NewRequest("PUT", url, body)
    if err != nil {
        return nil, err
    }
    putRequest.ContentLength = contentLength
    return http.DefaultClient.Do(putRequest)
}
func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error)
{
    delRequest, err := http.NewRequest("DELETE", url, nil)
    if err != nil {
        return nil, err
    }
    return http.DefaultClient.Do(delRequest)
}
```

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

The following shows three example of how to create presigned URLs and use the URLs with HTTP client libraries:

- An HTTP GET request that uses the URL with three HTTP client libraries
- An HTTP PUT request with metadata in headers that uses the URL with three HTTP client libraries
- An HTTP PUT request with query parameters that uses the URL with one HTTP client library

Generate a pre-signed URL for an object, then download it (GET request).

Imports.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

Generate the URL.

```
/* Create a pre-signed URL to download an object in a subsequent GET request.
 */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
```

```

        .bucket(bucketName)
        .key(keyName)
        .build();

        GetObjectPresignRequest presignRequest =
GetObjectPresignRequest.builder()
        .signatureDuration(Duration.ofMinutes(10)) // The URL will
expire in 10 minutes.
        .getObjectRequest(objectRequest)
        .build();

        PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]",
presignedRequest.url().toString());
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Download the object by using any one of the following three approaches.

Use JDK `URLConnection` (since v1.1) class to do the download.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {

```

```
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

Use JDK `HttpClient` (since v11) class to do the download.

```
/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpResponse<InputStream> response = httpClient.send(requestBuilder
            .uri(presignedUrl.toURI())
            .GET()
            .build(),
            HttpResponse.BodyHandlers.ofInputStream());

        IoUtils.copy(response.body(), byteArrayOutputStream);

        logger.info("HTTP response code is " + response.statusCode());

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

Use the Amazon SDK for Java `SdkHttpClient` class to do the download.

```
/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToGet(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
```

```

        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
                sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {
                        IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                },
                () -> logger.error("No response body."));

            logger.info("HTTP Response code is {}",
                response.httpResponse().statusCode());
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }

```

Generate a pre-signed URL with metadata in headers for an upload, then upload a file (PUT request).

Imports.

```

import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;

```

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Generate the URL.

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName,
    Map<String, String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
```

```

        .build();

        PutObjectPresignRequest presignRequest =
        PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL
            expires in 10 minutes.
            .putObjectRequest(objectRequest)
            .build();

        PresignedPutObjectRequest presignedRequest =
        presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]",
        presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Upload a file object by using any one of the following three approaches.

Use the JDK `URLConnection` (since v1.1) class to do the upload.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useHttpURLConnectionToPut(String presignedUrlString, File
fileToPut, Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-
meta-" + k, v));
        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
            FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is
8k

```

```

        while (inChannel.read(buffer) > 0) {
            buffer.flip();
            for (int i = 0; i < buffer.limit(); i++) {
                out.write(buffer.get());
            }
            buffer.clear();
        }
    } catch (IOException e) {
        logger.error(e.getMessage(), e);
    }

    out.close();
    connection.getResponseCode();
    logger.info("HTTP response code is " + connection.getResponseCode());

} catch (S3Exception | IOException e) {
    logger.error(e.getMessage(), e);
}
}

```

Use the JDK `HttpClient` (since v11) class to do the upload.

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
    Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        final HttpResponse<Void> response = httpClient.send(requestBuilder
            .uri(new URL(presignedUrlString).toURI())

            .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
            .build(),
            HttpResponse.BodyHandlers.discarding());

        logger.info("HTTP response code is " + response.statusCode());
    }
}

```



```

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
}

```

Use the Amazon for Java V2 `SdkHttpClient` class to do the upload.

```

/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
    Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" +
k, v));
        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

Generate a pre-signed URL with query parameters for an upload, then upload a file (PUT request).

Imports.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.awscore.AwsRequestOverrideConfiguration;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Generate the URL.

```
/**
 * Creates a presigned URL to use in a subsequent HTTP PUT request. The code
 * adds query parameters
 * to the request instead of using headers. By using query parameters, you
 * do not need to add the
 * the parameters as headers when the PUT request is eventually sent.
 *
 * @param bucketName Bucket name where the object will be uploaded.
```

```

    * @param keyName Key name of the object that will be uploaded.
    * @param queryParams Query string parameters to be added to the presigned
    URL.
    * @return
    */
    public String createPresignedUrl(String bucketName, String keyName,
    Map<String, String> queryParams) {
        try (S3Presigner presigner = S3Presigner.create()) {
            // Create an override configuration to store the query parameters.
            AwsRequestOverrideConfiguration.Builder overrideConfigurationBuilder
            = AwsRequestOverrideConfiguration.builder();

            queryParams.forEach(overrideConfigurationBuilder::putRawQueryParameter);

            PutObjectRequest objectRequest = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)

            .overrideConfiguration(overrideConfigurationBuilder.build()) // Add the override
            configuration.
                .build();

            PutObjectPresignRequest presignRequest =
            PutObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // The URL
            expires in 10 minutes.
                .putObjectRequest(objectRequest)
                .build();

            PresignedPutObjectRequest presignedRequest =
            presigner.presignPutObject(presignRequest);
            String myURL = presignedRequest.url().toString();
            logger.info("Presigned URL to upload a file to: [{}]", myURL);
            logger.info("HTTP method: [{}]",
            presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
}

```

Use the Amazon for Java V2 SdkHttpClient class to do the upload.

```
/**
 * Use the AWS SDK for Java V2 SdkHttpClient class to execute the PUT
 * request. Since the
 * URL contains the query parameters, no headers are needed for metadata, SSE
 * settings, or ACL settings.
 *
 * @param presignedUrlString The URL for the PUT request.
 * @param fileToPut File to upload
 */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut)
{
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());

        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create a presigned URL to upload an object to a bucket.

```
import https from "node:https";

import { XMLParser } from "fast-xml-parser";
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" })
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
```

```

    return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Make a PUT request to the provided URL.
 *
 * @param {string} url
 * @param {string} data
 */
const put = (url, data) => {
  return new Promise((resolve, reject) => {
    const req = https.request(
      url,
      { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
      (res) => {
        let responseBody = "";
        res.on("data", (chunk) => {
          responseBody += chunk;
        });
        res.on("end", () => {
          const parser = new XMLParser();
          if (res.statusCode >= 200 && res.statusCode <= 299) {
            resolve(parser.parse(responseBody, true));
          } else {
            reject(parser.parse(responseBody, true));
          }
        });
      },
    );
    req.on("error", (err) => {
      reject(err);
    });
    req.write(data);
    req.end();
  });
};

/**
 * Create two presigned urls for uploading an object to an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */

```

```

export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      key,
      region,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");

    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials
        configured?\n${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

```

Create a presigned URL to download an object from a bucket.

```

import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,

```

```

    S3RequestPresigner,
  } from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(new HttpRequest(url));
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new GetObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Create two presigned urls for downloading an object from an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      region,
      key,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });
  }
};

```



```
console.log("Presigned URL without client");
console.log(noClientUrl);
console.log("\n");

console.log("Presigned URL with client");
console.log(clientUrl);
} catch (caught) {
  if (caught instanceof Error && caught.name === "CredentialsProviderError") {
    console.error(
      `There was an error getting your credentials. Are your local credentials
configured?\n${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

- For more information, see [Amazon SDK for JavaScript Developer Guide](#).

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create a `GetObject` presigned request and use the URL to download an object.

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
```

```
        bucket = bucketName
        key = keyName
    }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET
    request to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()

    return objectContents
}
```

Create a `GetObject` presigned request with advanced options.

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be
            used 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

Create a `PutObject` presigned request and use it to upload an object.

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
    HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }
            .put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- For more information, see [Amazon SDK for Kotlin developer guide](#).

PHP

SDK for PHP

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the
S3 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in
the given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
```

```

    });
    try {
        $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
        echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
next 20 minutes.\n";
        echo $linebreak;
        echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
    } catch (AwsException $exception) {
        echo $linebreak;
        echo "Something went wrong: $exception";
        die();
    }
}

$runner = new PresignedURL();
$runner->run();

namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
    }

```

```
    }
    $this->verbose = $verbose;
}

public function setVerbose($verbose)
{
    $this->verbose = $verbose;
}

public function isVerbose(): bool
{
    return $this->verbose;
}

public function getClient(): S3Client
{
    return $this->client;
}

public function setClient(S3Client $client)
{
    $this->client = $client;
}

public function emptyAndDeleteBucket($bucketName, array $args = [])
{
    try {
        $objects = $this->listAllObjects($bucketName, $args);
        $this->deleteObjects($bucketName, $objects, $args);
        if ($this->verbose) {
            echo "Deleted all objects and folders from $bucketName.\n";
        }
        $this->deleteBucket($bucketName, $args);
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "\nPlease fix error with bucket deletion before continuing.
\n";
        }
        throw $exception;
    }
}
```

```
public function createBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->createBucket($parameters);
        if ($this->verbose) {
            echo "Created the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket creation before continuing.";
        }
        throw $exception;
    }
}

public function putObject(string $bucketName, string $key, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $this->client->putObject($parameters);
        if ($this->verbose) {
            echo "Uploaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object uploading before continuing.";
        }
        throw $exception;
    }
}
```

```
public function getObject(string $bucketName, string $key, array $args = []):
Result
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $object = $this->client->getObject($parameters);
        if ($this->verbose) {
            echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object downloading before
continuing.";
        }
        throw $exception;
    }
    return $object;
}

public function copyObject($bucketName, $key, $copySource, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
    try {
        $this->client->copyObject($parameters);
        if ($this->verbose) {
            echo "Copied the object from: $copySource in $bucketName to:
$key.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object copying before continuing.";
        }
        throw $exception;
    }
}
```



```
}

    public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
            throw $exception;
        }
        return $objects;
    }

    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
        return $contents;
    }
}
```

```

    public function deleteObjects(string $bucketName, array $objects, array $args
= [])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {

```

```

        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket deletion before continuing.";
        }
        throw $exception;
    }
}

public function deleteObject(string $bucketName, string $fileName, array
$args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
    try {
        $this->client->deleteObject($parameters);
        if ($this->verbose) {
            echo "Deleted the object named: $fileName from $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}

public function listBuckets(array $args = [])
{
    try {
        $buckets = $this->client->listBuckets($args);
        if ($this->verbose) {
            echo "Retrieved all " . count($buckets) . "\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";

```

```

        echo "Please fix error with bucket lists before continuing.";
    }
    throw $exception;
}
return $buckets;
}

public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
{
    $request = $this->client->createPresignedRequest($command, $expires,
$options);
    try {
        $presignedUrl = (string)$request->getUri();
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
            echo "Please fix error with presigned urls before continuing.";
        }
        throw $exception;
    }
    return $presignedUrl;
}

public function createSession(string $bucketName)
{
    try{
        $result = $this->client->createSession([
            'Bucket' => $bucketName,
        ]);
        return $result;
    }catch(S3Exception $caught){
        if($caught->getAwsErrorType() == "NoSuchBucket"){
            echo "The specified bucket does not exist.";
        }
        throw $caught;
    }
}
}

```

```
}
```

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Generate a presigned URL that can perform an S3 action for a limited time. Use the Requests package to make a request with the URL.

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)

def generate_presigned_url(s3_client, client_method, method_parameters,
    expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method, Params=method_parameters,
            ExpiresIn=expires_in
```

```

        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method
        )
        raise
    return url

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key",
        help="For a GET operation, the key of the object in Amazon S3. For a "
        "PUT operation, the name of a file to upload.",
    )
    parser.add_argument("action", choices=("get", "put"), help="The action to perform.")
    args = parser.parse_args()

    s3_client = boto3.client("s3")
    client_action = "get_object" if args.action == "get" else "put_object"
    url = generate_presigned_url(
        s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
    )

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == "get":
        response = requests.get(url)
        if response.status_code == 200:
            with open(args.key.split("/")[-1], 'wb') as object_file:
                object_file.write(response.content)
    elif args.action == "put":
        print("Putting data to the URL.")
        try:

```

```

        with open(args.key, "rb") as object_file:
            object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(
                f"Couldn't find {args.key}. For a PUT operation, the key must be
the "
                f"name of a file that exists on your computer."
            )

        if response is not None:
            print(f"Status: {response.status_code}\nReason: {response.reason}")

        print("-" * 88)

if __name__ == "__main__":
    usage_demo()

```

Generate a presigned POST request to upload a file.

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an
AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.

```

```

        :param expires_in: The number of seconds the presigned POST is valid.
        :return: A dictionary that contains the URL and form fields that contain
                required access data.
        """
        try:
            response = self.bucket.meta.client.generate_presigned_post(
                Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
            )
            logger.info("Got presigned POST URL: %s", response["url"])
        except ClientError:
            logger.exception(
                "Couldn't get a presigned POST URL for bucket '%s' and object
                '%s'",
                self.bucket.name,
                object_key,
            )
            raise
        return response

```

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

require 'aws-sdk-s3'
require 'net/http'

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)

```



```
url = bucket.object(object_key).presigned_url(:put)
puts "Created presigned URL: #{url}"
URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's
  why: #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request('PUT', presigned_url.request_uri, object_content,
    'content_type' => '')
  end

  case response
  when Net::HTTPSuccess
    puts 'Content uploaded!'
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create presigning requests to GET S3 objects.

```
/// Generate a URL for a presigned GET request.
async fn get_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);
    let presigned_request = client
        .get_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in?))
        .await?;

    println!("Object URI: {}", presigned_request.uri());
    let valid_until = chrono::offset::Local::now() + expires_in;
    println!("Valid until: {valid_until}");

    Ok(())
}
```

Create presigning requests to PUT S3 objects.

```
async fn put_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
```

```

) -> Result<String, S3ExampleError> {
    let expires_in: std::time::Duration =
std::time::Duration::from_secs(expires_in);
    let expires_in: aws_sdk_s3::presigning::PresigningConfig =
    PresigningConfig::expires_in(expires_in).map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to convert expiration to PresigningConfig: {err:?}")
        ))
    })?;
    let presigned_request = client
        .put_object()
        .bucket(bucket)
        .key(object)
        .presigned(expires_in)
        .await?;

    Ok(presigned_request.uri().into())
}

```

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create presigned requests to GET S3 objects.

```

" iv_bucket_name is the bucket name
" iv_key is the object name like "myfile.txt"

DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

"Upload a nice Hello World file to an S3 bucket."
TRY.
    DATA(lv_contents) = cl_abap_codepage=>convert_to( 'Hello, World' ).
    lo_s3->putobject(

```

```

        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_contents
        iv_contenttype = 'text/plain' ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" now generate a presigned URL with a 600-second expiration
DATA(lo_presigner) = lo_s3->get_presigner( iv_expires_sec = 600 ).
" the presigner getobject() method has the same signature as
" lo_s3->getobject(), but it doesn't actually make the call.
" to the service. It just prepares a presigned URL for a future call
DATA(lo_presigned_req) = lo_presigner->getobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_key ).

" You can provide this URL to a web page, user, email etc so they
" can retrieve the file. The URL will expire in 10 minutes.
ov_url = lo_presigned_req->get_url( ).

```

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Create a photo asset management application that lets users manage photos using labels

The following code examples show how to create a serverless application that lets users manage photos using labels.

.NET

Amazon SDK for .NET

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

For a deep dive into the origin of this example see the post on [Amazon Community](#).

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK for C++

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

For a deep dive into the origin of this example see the post on [Amazon Community](#).

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK for Java 2.x

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

For a deep dive into the origin of this example see the post on [Amazon Community](#).

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK for JavaScript (v3)

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

For a deep dive into the origin of this example see the post on [Amazon Community](#).

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK for Kotlin

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

For a deep dive into the origin of this example see the post on [Amazon Community](#).

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK for PHP

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

For a deep dive into the origin of this example see the post on [Amazon Community](#).

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3

- Amazon SNS

Rust

SDK for Rust

Shows how to develop a photo asset management application that detects labels in images using Amazon Rekognition and stores them for later retrieval.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

For a deep dive into the origin of this example see the post on [Amazon Community](#).

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

A web page that lists Amazon S3 objects using an Amazon SDK

The following code example shows how to list Amazon S3 objects in a web page.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

The following code is the relevant React component that makes calls to the Amazon SDK. A runnable version of the application containing this component can be found at the preceding GitHub link.

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  type ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
  const [objects, setObjects] = useState<
    Required<ListObjectsCommandOutput>["Contents"]
  >([]);

  useEffect(() => {
    const client = new S3Client({
      region: "us-east-1",
      // Unless you have a public bucket, you'll need access to a private bucket.
      // One way to do this is to create an Amazon Cognito identity pool, attach
      // a role to the pool,
      // and grant the role access to the 's3:GetObject' action.
      //
      // You'll also need to configure the CORS settings on the bucket to allow
      // traffic from
      // this example site. Here's an example configuration that allows all
      // origins. Don't
      // do this in production.
      // [
      //   {
      //     "AllowedHeaders": ["*"],
      //     "AllowedMethods": ["GET"],
      //     "AllowedOrigins": ["*"],
      //     "ExposeHeaders": [],
      //   },
      // ],
      //
      credentials: fromCognitoIdentityPool({
        clientConfig: { region: "us-east-1" },
        identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
      })
    });
```

```
    }},
  });
  const command = new ListObjectsCommand({ Bucket: "bucket-name" });
  client.send(command).then(({ Contents }) => setObjects(Contents || []));
}, []);

return (
  <div className="App">
    {objects.map((o) => (
      <div key={o.ETag}>{o.Key}</div>
    ))}
  </div>
);
}

export default App;
```

- For API details, see [ListObjects](#) in *Amazon SDK for JavaScript API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Create an Amazon Textract explorer application

The following code examples show how to explore Amazon Textract output through an interactive application.

JavaScript

SDK for JavaScript (v3)

Shows how to use the Amazon SDK for JavaScript to build a React application that uses Amazon Textract to extract data from a document image and display it in an interactive web page. This example runs in a web browser and requires an authenticated Amazon Cognito identity for credentials. It uses Amazon Simple Storage Service (Amazon S3) for storage, and for notifications it polls an Amazon Simple Queue Service (Amazon SQS) queue that is subscribed to an Amazon Simple Notification Service (Amazon SNS) topic.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK for Python (Boto3)

Shows how to use the Amazon SDK for Python (Boto3) with Amazon Textract to detect text, form, and table elements in a document image. The input image and Amazon Textract output are shown in a Tkinter application that lets you explore the detected elements.

- Submit a document image to Amazon Textract and explore the output of detected elements.
- Submit images directly to Amazon Textract or through an Amazon Simple Storage Service (Amazon S3) bucket.
- Use asynchronous APIs to start a job that publishes a notification to an Amazon Simple Notification Service (Amazon SNS) topic when the job completes.
- Poll an Amazon Simple Queue Service (Amazon SQS) queue for a job completion message and display the results.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS

- Amazon Textract

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Delete all objects in a given Amazon S3 bucket using an Amazon SDK

The following code example shows how to delete all of the objects in an Amazon S3 bucket.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete all objects for a given Amazon S3 bucket.

```
import {
  DeleteObjectsCommand,
  paginateListObjectsV2,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 *
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  try {
    console.log(`Deleting all objects in bucket: ${bucketName}`);

    const paginator = paginateListObjectsV2(
      { client },
      {
        Bucket: bucketName,
```

```

    },
  );

  const objectKeys = [];
  for await (const { Contents } of paginator) {
    objectKeys.push(...Contents.map((obj) => ({ Key: obj.Key })));
  }

  const deleteCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: objectKeys },
  });

  await client.send(deleteCommand);

  console.log(`All objects deleted from bucket: ${bucketName}`);
} catch (caught) {
  if (caught instanceof Error) {
    console.error(
      `Failed to empty ${bucketName}. ${caught.name}: ${caught.message}`,
    );
  }
}
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
    bucketName: {
      type: "string",
    },
  },
};

const { values } = parseArgs({ options });
main(values);
}

```

- For API details, see the following topics in *Amazon SDK for JavaScript API Reference*.
 - [DeleteObjects](#)
 - [ListObjectsV2](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Delete incomplete multipart uploads to Amazon S3 using an Amazon SDK

The following code example shows how to delete or stop incomplete Amazon S3 multipart uploads.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

To stop multipart uploads that are in-progress or incomplete for any reason, you can get a list uploads and then delete them as shown in the following example.

```
/**
 * Aborts all incomplete multipart uploads from the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket
 */
public static void abortIncompleteMultipartUploadsFromList(String bucketName)
{
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
            .bucket(bucketName)
```

```

        .key(upload.key())
        .expectedBucketOwner(accountId)
        .uploadId(upload.uploadId())
        .build();

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
        }
    }
}

```

To delete incomplete multipart uploads that were initiated before or after a date, you can selectively delete multipart uploads based on a point in time as shown in the following example.

```

/**
 * Aborts incomplete multipart uploads older than the specified point in
time.
 *
 * @param bucketName the name of the S3 bucket
 * @param pointInTime the cutoff time; uploads initiated before this are
aborted
 */
static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
        if (upload.initiated().isBefore(pointInTime)) {

```

```

        abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(upload.key())
    .expectedBucketOwner(accountId)
    .uploadId(upload.uploadId())
    .build();

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
        }
    }
}
}

```

If you have access to the upload ID after you begin a multipart upload, you can delete the in-progress upload by using the ID.

```

/**
 * Aborts a multipart upload using the upload ID.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 */
static void abortMultipartUploadUsingUploadId(String bucketName, String key)
{
    String uploadId = startUploadReturningUploadId(bucketName, key);
    AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
    }
}

```


To consistently delete incomplete multipart uploads older than a certain number of days, set up a bucket lifecycle configuration for the bucket. The following example shows how to create a rule to delete incomplete uploads older than 7 days.

```
/**
 * Configures a lifecycle rule to abort incomplete multipart uploads after 7
 days.
 *
 * @param bucketName the name of the S3 bucket
 */
static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
    Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b.
            daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());

    PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
        .bucket(bucketName)
        .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
    } else {
        logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
    }
}
```

- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [PutBucketLifecycleConfiguration](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Detect PPE in images with Amazon Rekognition using an Amazon SDK

The following code example shows how to build an app that uses Amazon Rekognition to detect Personal Protective Equipment (PPE) in images.

Java

SDK for Java 2.x

Shows how to create an Amazon Lambda function that detects images with Personal Protective Equipment.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Detect entities in text extracted from an image using an Amazon SDK

The following code example shows how to use Amazon Comprehend to detect entities in text extracted by Amazon Textract from an image that is stored in Amazon S3.

Python

SDK for Python (Boto3)

Shows how to use the Amazon SDK for Python (Boto3) in a Jupyter notebook to detect entities in text that is extracted from an image. This example uses Amazon Textract to

extract text from an image stored in Amazon Simple Storage Service (Amazon S3) and Amazon Comprehend to detect entities in the extracted text.

This example is a Jupyter notebook and must be run in an environment that can host notebooks. For instructions on how to run the example using Amazon SageMaker AI, see the directions in [TextractAndComprehendNotebook.ipynb](#).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Comprehend
- Amazon S3
- Amazon Textract

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Detect faces in an image using an Amazon SDK

The following code example shows how to:

- Save an image in an Amazon S3 bucket.
- Use Amazon Rekognition to detect facial details, such as age range, gender, and emotion (such as smiling).
- Display those details.

Rust

SDK for Rust

Save the image in an Amazon S3 bucket with an **uploads** prefix, use Amazon Rekognition to detect facial details, such as age range, gender, and emotion (smiling, etc.), and display those details.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Rekognition
- Amazon S3

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Detect objects in images with Amazon Rekognition using an Amazon SDK

The following code examples show how to build an app that uses Amazon Rekognition to detect objects by category in images.

.NET

Amazon SDK for .NET

Shows how to use Amazon Rekognition .NET API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK for Java 2.x

Shows how to use Amazon Rekognition Java API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK for JavaScript (v3)

Shows how to use Amazon Rekognition with the Amazon SDK for JavaScript to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for objects using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK for Kotlin

Shows how to use Amazon Rekognition Kotlin API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage

Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK for Python (Boto3)

Shows you how to use the Amazon SDK for Python (Boto3) to create a web application that lets you do the following:

- Upload photos to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Rekognition to analyze and label the photos.
- Use Amazon Simple Email Service (Amazon SES) to send email reports of image analysis.

This example contains two main components: a webpage written in JavaScript that is built with React, and a REST service written in Python that is built with Flask-RESTful.

You can use the React webpage to:

- Display a list of images that are stored in your S3 bucket.
- Upload images from your computer to your S3 bucket.
- Display images and labels that identify items that are detected in the image.
- Get a report of all images in your S3 bucket and send an email of the report.

The webpage calls the REST service. The service sends requests to Amazon to perform the following actions:

- Get and filter the list of images in your S3 bucket.
- Upload photos to your S3 bucket.
- Use Amazon Rekognition to analyze individual photos and get a list of labels that identify items that are detected in the photo.

- Analyze all photos in your S3 bucket and use Amazon SES to email a report.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Detect people and objects in a video with Amazon Rekognition using an Amazon SDK

The following code examples show how to detect people and objects in a video with Amazon Rekognition.

Java

SDK for Java 2.x

Shows how to use Amazon Rekognition Java API to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK for Python (Boto3)

Use Amazon Rekognition to detect faces, objects, and people in videos by starting asynchronous detection jobs. This example also configures Amazon Rekognition to notify an Amazon Simple Notification Service (Amazon SNS) topic when jobs complete and subscribes an Amazon Simple Queue Service (Amazon SQS) queue to the topic. When the queue receives a message about a job, the job is retrieved and the results are output.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Download S3 'directories' from an Amazon Simple Storage Service (Amazon S3) bucket

The following code example shows how to download and filter the contents of Amazon S3 bucket 'directories'.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

This example shows how to use the [S3TransferManager](#) in the Amazon SDK for Java 2.x to download 'directories' from an Amazon S3 bucket. It also demonstrates how to use [DownloadFilters](#) in the request.

```
/**
 * For standard buckets, S3 provides the illusion of a directory structure
 * through the use of keys. When you upload
 *   * an object to an S3 bucket, you specify a key, which is essentially the
 * "path" to the object. The key can contain
 *   * forward slashes ("/") to make it appear as if the object is stored in a
 * directory structure, but this is just a
 *   * logical representation, not an actual directory.
 * <p><pre>
 * In this example, our S3 bucket contains the following objects:
 *
 * folder1/file1.txt
 * folder1/file2.txt
 * folder1/file3.txt
 * folder2/file1.txt
 * folder2/file2.txt
 * folder2/file3.txt
 * folder3/file1.txt
 * folder3/file2.txt
 * folder3/file3.txt
 *
 * When method `downloadS3Directories` is invoked with
 * `destinationPathURI` set to `/test`, the downloaded
 * directory looks like:
 *
 * |- test
 *   |- folder1
 *     |- file1.txt
 *     |- file2.txt
 *     |- file3.txt
 *   |- folder3
 *     |- file1.txt
 *     |- file2.txt
 *     |- file3.txt
 * </pre>
 *
 * @param transferManager An S3TransferManager instance.
 * @param destinationPathURI local directory to hold the downloaded S3
 * 'directories' and files.
```

```

    * @param bucketName      The S3 bucket that contains the 'directories' to
download.
    * @return The number of objects (files, in this case) that were downloaded.
    */
    public Integer downloadS3Directories(S3TransferManager transferManager,
                                         URI destinationPathURI, String
bucketName) {

        // Define the filters for which 'directories' we want to download.
        DownloadFilter folder1Filter = (S3Object s3Object) ->
s3Object.key().startsWith("folder1/");
        DownloadFilter folder3Filter = (S3Object s3Object) ->
s3Object.key().startsWith("folder3/");
        DownloadFilter folderFilter = s3Object ->
folder1Filter.or(folder3Filter).test(s3Object);

        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                                .destination(Paths.get(destinationPathURI))
                                .bucket(bucketName)
                                .filter(folderFilter)
                                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        Integer numFilesInFolder1 =
Paths.get(destinationPathURI).resolve("folder1").toFile().list().length;
        Integer numFilesInFolder3 =
Paths.get(destinationPathURI).resolve("folder3").toFile().list().length;

        try {
            assert numFilesInFolder1 == 3;
            assert numFilesInFolder3 == 3;
            assert !
Paths.get(destinationPathURI).resolve("folder2").toFile().exists(); // `folder2`
was not downloaded.
        } catch (AssertionError e) {
            logger.error("An assertion failed.");
        }

        completedDirectoryDownload.failedTransfers()
            .forEach(fail -> logger.warn("Object failed to transfer  [{})",
fail.exception().getMessage()));
        return numFilesInFolder1 + numFilesInFolder3;
    }

```

```
}
```

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Download all objects in an Amazon Simple Storage Service (Amazon S3) bucket to a local directory

The following code example shows how to download all objects in an Amazon Simple Storage Service (Amazon S3) bucket to a local directory.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Use an [S3TransferManager](#) to [download all S3 objects](#) in the same S3 bucket. View the [complete file](#) and [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
```

```
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
        URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
            .destination(Paths.get(destinationPathURI))
            .bucket(bucketName)
            .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
            .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

- For API details, see [DownloadDirectory](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Download a stream of unknown size from an Amazon S3 object using an Amazon SDK

The following code example shows how to download a stream of unknown size from an Amazon S3 object.

Swift

SDK for Swift

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams

/// Download a file from the specified bucket.
///
/// - Parameters:
///   - bucket: The Amazon S3 bucket name to get the file from.
///   - key: The name (or path) of the file to download from the bucket.
///   - destPath: The pathname on the local filesystem at which to store
///     the downloaded file.
func downloadFile(bucket: String, key: String, destPath: String?) async
throws {
    let fileURL: URL

    // If no destination path was provided, use the key as the name to use
    // for the file in the downloads folder.

    if destPath == nil {
        do {
            try fileURL = FileManager.default.url(
                for: .downloadsDirectory,
                in: .userDomainMask,
                appropriateFor: URL(string: key),
                create: true
            ).appendingPathComponent(key)
        } catch {
```

```
        throw TransferError.directoryError
    }
} else {
    fileURL = URL(fileURLWithPath: destPath!)
}

let config = try await S3Client.S3ClientConfiguration(region: region)
let s3Client = S3Client(config: config)

// Create a `FileHandle` referencing the local destination. Then
// create a `ByteStream` from that.

FileManager.default.createFile(atPath: fileURL.path, contents: nil,
attributes: nil)
let fileHandle = try FileHandle(forWritingTo: fileURL)

// Download the file using `GetObject`.

let getInput = GetObjectInput(
    bucket: bucket,
    key: key
)

do {
    let getOutput = try await s3Client.getObject(input: getInput)

    guard let body = getOutput.body else {
        throw TransferError.downloadError("Error: No data returned for
download")
    }

    // If the body is returned as a `Data` object, write that to the
    // file. If it's a stream, read the stream chunk by chunk,
    // appending each chunk to the destination file.

    switch body {
    case .data:
        guard let data = try await body.readData() else {
            throw TransferError.downloadError("Download error")
        }

        // Write the `Data` to the file.

        do {
```

```

        try data.write(to: fileURL)
    } catch {
        throw TransferError.writeError
    }
    break

case .stream(let stream as ReadableStream):
    while (true) {
        let chunk = try await stream.readAsync(upToCount: 5 * 1024 *
1024)

        guard let chunk = chunk else {
            break
        }

        // Write the chunk to the destination file.

        do {
            try fileHandle.write(contentsOf: chunk)
        } catch {
            throw TransferError.writeError
        }

        break
    default:
        throw TransferError.downloadError("Received data is unknown
object type")
    }
    } catch {
        throw TransferError.downloadError("Error downloading the file:
\\(error)")
    }

    print("File downloaded to \\(fileURL.path).")
}

```

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Get an Amazon S3 object from a Multi-Region Access Point by using an Amazon SDK

The following code example shows how to get an object from a Multi-Region Access Point.

Kotlin

SDK for Kotlin

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Configure the S3 client to use the Asymmetric Sigv4 (Sigv4a) signing algorithm.

```
suspend fun createS3Client(): S3Client {
    // Configure your S3Client to use the Asymmetric SigV4 (SigV4a)
    signing algorithm.
    val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)
    val s3 = S3Client.fromEnvironment {
        authSchemes = listOf(sigV4aScheme)
    }
    return s3
}
```

Use the Multi-Region Access Point ARN instead of a bucket name to retrieve the object.

```
suspend fun getObjectFromMrap(
    s3: S3Client,
    mrpArn: String,
    keyName: String,
): String? {
    val request = GetObjectRequest {
        bucket = mrpArn // Use the ARN instead of the bucket name for object
        operations.
        key = keyName
    }

    var stringObj: String? = null
```



```
s3.getObject(request) { resp ->
    stringObj = resp.body?.decodeToString()
    if (stringObj != null) {
        println("Successfully read $keyName from $mrpArn")
    }
}
return stringObj
}
```

- For more information, see [Amazon SDK for Kotlin developer guide](#).
- For API details, see [GetObject](#) in *Amazon SDK for Kotlin API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Get an object from an Amazon S3 bucket using an Amazon SDK, specifying an If-Modified-Since header

The following code example shows how to read data from an object in an S3 bucket, but only if that bucket has not been modified since the last retrieval time.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
use aws_sdk_s3::{
    error::SdkError,
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client,
};
use s3_code_examples::error::S3ExampleError;
use tracing::{error, warn};
```

```
const KEY: &str = "key";
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    tracing_subscriber::fmt::init();

    // Get a new UUID to use when creating a unique bucket name.
    let uuid = uuid::Uuid::new_v4();

    // Load the AWS configuration from the environment.
    let client = Client::new(&aws_config::load_from_env().await);

    // Generate a unique bucket name using the previously generated UUID.
    // Then create a new bucket with that name.
    let bucket_name = format!("if-modified-since-{{uuid}}");
    client
        .create_bucket()
        .bucket(bucket_name.clone())
        .send()
        .await?;

    // Create a new object in the bucket whose name is `KEY` and whose
    // contents are `BODY`.
    let put_object_output = client
        .put_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .body(ByteStream::from_static(BODY.as_bytes()))
        .send()
        .await;

    // If the `PutObject` succeeded, get the eTag string from it. Otherwise,
    // report an error and return an empty string.
```

```

let e_tag_1 = match put_object_output {
  Ok(put_object) => put_object.e_tag.unwrap(),
  Err(err) => {
    error!("{err:?}");
    String::new()
  }
};

// Request the object's headers.
let head_object_output = client
  .head_object()
  .bucket(bucket_name.as_str())
  .key(KEY)
  .send()
  .await;

// If the `HeadObject` request succeeded, create a tuple containing the
// values of the headers `last-modified` and `etag`. If the request
// failed, return the error in a tuple instead.
let (last_modified, e_tag_2) = match head_object_output {
  Ok(head_object) => (
    Ok(head_object.last_modified().cloned().unwrap()),
    head_object.e_tag.unwrap(),
  ),
  Err(err) => (Err(err), String::new()),
};

warn!("last modified: {last_modified:?}");
assert_eq!(
  e_tag_1, e_tag_2,
  "PutObject and first GetObject had differing eTags"
);

println!("First value of last_modified: {last_modified:?}");
println!("First tag: {}\n", e_tag_1);

// Send a second `HeadObject` request. This time, the `if_modified_since`
// option is specified, giving the `last_modified` value returned by the
// first call to `HeadObject`.
//
// Since the object hasn't been changed, and there are no other objects in
// the bucket, there should be no matching objects.

let head_object_output = client

```

```

        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .if_modified_since(last_modified.unwrap())
        .send()
        .await;

// If the `HeadObject` request succeeded, the result is a tuple containing
// the `last_modified` and `e_tag_1` properties. This is not the expected
// result.
//
// The expected result of the second call to `HeadObject` is an
// `SdkError::ServiceError` containing the HTTP error response. If that's
// the case and the HTTP status is 304 (not modified), the output is a
// tuple containing the values of the HTTP `last-modified` and `etag`
// headers.
//
// If any other HTTP error occurred, the error is returned as an
// `SdkError::ServiceError`.

let (last_modified, e_tag_2) = match head_object_output {
    Ok(head_object) => (
        Ok(head_object.last_modified().cloned().unwrap()),
        head_object.e_tag.unwrap(),
    ),
    Err(err) => match err {
        SdkError::ServiceError(err) => {
            // Get the raw HTTP response. If its status is 304, the
            // object has not changed. This is the expected code path.
            let http = err.raw();
            match http.status().as_u16() {
                // If the HTTP status is 304: Not Modified, return a
                // tuple containing the values of the HTTP
                // `last-modified` and `etag` headers.
                304 => (
                    Ok(DateTime::from_str(
                        http.headers().get("last-modified").unwrap(),
                        DateTimeFormat::HttpDate,
                    )
                    .unwrap()),
                    http.headers().get("etag").map(|t| t.into()).unwrap(),
                ),
                // Any other HTTP status code is returned as an
                // `SdkError::ServiceError`.
            }
        }
    }
}

```

```

        _ => (Err(SdkError::ServiceError(err)), String::new()),
    }
}
// Any other kind of error is returned in a tuple containing the
// error and an empty string.
_ => (Err(err), String::new()),
},
};

warn!("last modified: {last_modified:?}");
assert_eq!(
    e_tag_1, e_tag_2,
    "PutObject and second HeadObject had different eTags"
);

println!("Second value of last modified: {last_modified:?}");
println!("Second tag: {}", e_tag_2);

// Clean up by deleting the object and the bucket.
client
    .delete_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .send()
    .await?;

client
    .delete_bucket()
    .bucket(bucket_name.as_str())
    .send()
    .await?;

Ok(())
}

```

- For API details, see [GetObject](#) in *Amazon SDK for Rust API reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Get started with encryption for Amazon S3 objects using an Amazon SDK

The following code example shows how to get started with encryption for Amazon S3 objects.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to apply client encryption to an object in an
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class SSEClientEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "exampleobject.txt";
        string copyTargetKeyName = "examplecopy.txt";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();

        try
        {
            // Create an encryption key.
```

```

        Aes aesEncryption = Aes.Create();
        aesEncryption.KeySize = 256;
        aesEncryption.GenerateKey();
        string base64Key = Convert.ToBase64String(aesEncryption.Key);

        // Upload the object.
        PutObjectRequest putObjectRequest = await
UploadObjectAsync(client, bucketName, keyName, base64Key);

        // Download the object and verify that its contents match what
you uploaded.
        await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);

        // Get object metadata and verify that the object uses AES-256
encryption.
        await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

        // Copy both the source and target objects using server-side
encryption with
        // an encryption key.
        await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: {ex.Message}");
    }
}

/// <summary>
/// Uploads an object to an Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// PutObjectAsync.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket to which
the
/// object will be uploaded.</param>
/// <param name="keyName">The name of the object to upload to the Amazon
S3
/// bucket.</param>
/// <param name="base64Key">The encryption key.</param>

```

```

    /// <returns>The PutObjectRequest object for use by
DownloadObjectAsync.</returns>
    public static async Task<PutObjectRequest> UploadObjectAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string base64Key)
    {
        PutObjectRequest putObjectRequest = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
            ContentBody = "sample text",
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };
        PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
        return putObjectRequest;
    }

    /// <summary>
    /// Downloads an encrypted object from an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// GetObjectAsync.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
object
    /// is located.</param>
    /// <param name="keyName">The name of the Amazon S3 object to download.</
param>
    /// <param name="base64Key">The encryption key used to encrypt the
    /// object.</param>
    /// <param name="putObjectRequest">The PutObjectRequest used to upload
    /// the object.</param>
    public static async Task DownloadObjectAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string base64Key,
        PutObjectRequest putObjectRequest)
    {

```



```
GetObjectRequest getObjectRequest = new GetObjectRequest
{
    BucketName = bucketName,
    Key = keyName,

    // Provide encryption information for the object stored in Amazon
S3.
    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
    ServerSideEncryptionCustomerProvidedKey = base64Key,
};

using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
    using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
    {
        string content = reader.ReadToEnd();
        if (string.Compare(putObjectRequest.ContentBody, content) == 0)
        {
            Console.WriteLine("Object content is same as we uploaded");
        }
        else
        {
            Console.WriteLine("Error...Object content is not same.");
        }

        if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
        {
            Console.WriteLine("Object encryption method is AES256, same
as we set");
        }
        else
        {
            Console.WriteLine("Error...Object encryption method is not
the same as AES256 we set");
        }
    }

}

/// <summary>
/// Retrieves the metadata associated with an Amazon S3 object.
/// </summary>
```

```

    /// <param name="client">The initialized Amazon S3 client object used
    /// to call GetObjectMetadataAsync.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket containing
the
    /// object for which we want to retrieve metadata.</param>
    /// <param name="keyName">The name of the object for which we wish to
    /// retrieve the metadata.</param>
    /// <param name="base64Key">The encryption key associated with the
    /// object.</param>
    public static async Task GetObjectMetadataAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string base64Key)
    {
        GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
        {
            BucketName = bucketName,
            Key = keyName,

            // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };

        GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
        Console.WriteLine("The object metadata show encryption method used
is: {0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
    }

    /// <summary>
    /// Copies an encrypted object from one Amazon S3 bucket to another.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// CopyObjectAsync.</param>
    /// <param name="bucketName">The Amazon S3 bucket containing the object
    /// to copy.</param>
    /// <param name="keyName">The name of the object to copy.</param>

```

```

    /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the
object
    /// will be copied.</param>
    /// <param name="aesEncryption">The encryption type to use.</param>
    /// <param name="base64Key">The encryption key to use.</param>
    public static async Task CopyObjectAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string copyTargetKeyName,
        Aes aesEncryption,
        string base64Key)
    {
        aesEncryption.GenerateKey();
        string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

        CopyObjectRequest copyRequest = new CopyObjectRequest
        {
            SourceBucket = bucketName,
            SourceKey = keyName,
            DestinationBucket = bucketName,
            DestinationKey = copyTargetKeyName,

            // Information about the source object's encryption.
            CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

            // Information about the target object's encryption.
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
        };
        await client.CopyObjectAsync(copyRequest);
    }
}

```

- For API details, see the following topics in *Amazon SDK for .NET API Reference*.
 - [CopyObject](#)
 - [GetObject](#)

- [GetObjectMetadata](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Get started with tags for Amazon S3 objects using an Amazon SDK

The following code example shows how to get started with tags for Amazon S3 objects.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to work with tags in Amazon Simple Storage
/// Service (Amazon S3) objects.
/// </summary>
public class ObjectTag
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "newobject.txt";
        string filePath = @"*** file path ***";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
```

```

        var client = new AmazonS3Client(bucketRegion);
        await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
    }

    /// <summary>
    /// This method uploads an object with tags. It then shows the tag
    /// values, changes the tags, and shows the new tags.
    /// </summary>
    /// <param name="client">The Initialized Amazon S3 client object used
    /// to call the methods to create and change an objects tags.</param>
    /// <param name="bucketName">A string representing the name of the
    /// bucket where the object will be stored.</param>
    /// <param name="keyName">A string representing the key name of the
    /// object to be tagged.</param>
    /// <param name="filePath">The directory location and file name of the
    /// object to be uploaded to the Amazon S3 bucket.</param>
    public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
    {
        try
        {
            // Create an object with tags.
            var putRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                FilePath = filePath,
                TagSet = new List<Tag>
                {
                    new Tag { Key = "Keyx1", Value = "Value1" },
                    new Tag { Key = "Keyx2", Value = "Value2" },
                },
            };

            PutObjectResponse response = await
client.PutObjectAsync(putRequest);

            // Now retrieve the new object's tags.
            GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
            {
                BucketName = bucketName,
                Key = keyName,
            }

```

```
};

GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

// Display the tag values.
objectTags.Tagging
    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

Tagging newTagSet = new Tagging()
{
    TagSet = new List<Tag>
    {
        new Tag { Key = "Key3", Value = "Value3" },
        new Tag { Key = "Key4", Value = "Value4" },
    },
};

PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
{
    BucketName = bucketName,
    Key = keyName,
    Tagging = newTagSet,
};

PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

// Retrieve the tags again and show the values.
GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
{
    BucketName = bucketName,
    Key = keyName,
};

GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

objectTags2.Tagging
    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
}
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(
                $"Error: '{ex.Message}'");
        }
    }
}
```

- For API details, see [GetObjectTagging](#) in *Amazon SDK for .NET API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Getting started with Amazon Athena

The following code example shows how to:

- Create an S3 bucket for query results
- Create a database
- Create a table
- Run a query
- Create and use named queries
- Clean up resources

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Sample developer tutorials](#) repository.

```
#!/bin/bash
```

```
# Amazon Athena Getting Started Script
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail

# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
        echo "- Named Query: $NAMED_QUERY_ID"
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi
    echo "Exiting..."
    exit 1
}
```



```

}

# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] ||
[ ${#bucket_name} -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
        return 1
    fi
    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z_][a-zA-Z0-9_]*$ ]]; then
        return 1
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then
    handle_error "Failed to generate valid random ID"
fi

# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
        --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --
output text 2>/dev/null)
fi

```

```

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
fi

if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifier "$DATABASE_NAME"; then
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi

# Security: Validate region format - expanded regex for newer regions
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~ ^[a-z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
    CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION"
2>&1)

```

```
if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
fi

aws s3api put-bucket-tagging \
    --bucket "$S3_BUCKET" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-athena-gs}]'

# Security: Enable S3 bucket encryption with KMS validation
echo "Enabling default encryption on S3 bucket..."
if ! aws s3api put-bucket-encryption \
    --bucket "$S3_BUCKET" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' 2>&1; then
    echo "Warning: Could not enable encryption on bucket"
fi

# Security: Block public access
echo "Blocking public access to S3 bucket..."
if ! aws s3api put-public-access-block \
    --bucket "$S3_BUCKET" \
    --public-access-block-configuration \

"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
2>&1; then
    echo "Warning: Could not block public access on bucket"
fi

# Security: Enable versioning for data protection
echo "Enabling versioning on S3 bucket..."
if ! aws s3api put-bucket-versioning \
    --bucket "$S3_BUCKET" \
    --versioning-configuration Status=Enabled 2>&1; then
    echo "Warning: Could not enable versioning on bucket"
fi

echo "S3 bucket created successfully: $S3_BUCKET"
fi
```

```

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Database creation timed out"
fi

```

[illegible]

```

QUERY_ID=$(echo "$CREATE_TABLE_RESULT" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo "$CREATE_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from table creation response"
fi
echo "Table creation query ID: $QUERY_ID"

# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

```

```

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
|| echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then

```

```
        handle_error "Query execution timed out"
    fi

    # Get query results
    echo "Getting query results..."
    RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
        "$AWS_REGION" 2>&1)
    if echo "$RESULTS" | grep -qi "error\|failed"; then
        handle_error "Failed to get query results: $RESULTS"
    fi
    echo "$RESULTS"

    # Download results from S3
    echo "Downloading query results from S3..."
    S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
        --region "$AWS_REGION" 2>&1)
    if echo "$S3_PATH" | grep -qi "error\|failed"; then
        handle_error "Failed to get S3 path for results: $S3_PATH"
    fi

    if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
        handle_error "S3 path for query results is empty"
    fi

    DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
    if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
        handle_error "Failed to download query results: $DOWNLOAD_RESULT"
    fi

    # Security: Secure the downloaded file
    chmod 600 "./query-results.csv"
    echo "Query results downloaded to query-results.csv (permissions: 600)"

    # Step 4: Create a named query
    echo "Step 4: Creating a named query..."
    NAMED_QUERY_RESULT=$(aws athena create-named-query \
        --name "OS Count Query" \
        --description "Count of operating systems in CloudFront logs" \
        --database "$DATABASE_NAME" \
        --query-string "$QUERY" \
        --region "$AWS_REGION" 2>&1)

    if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
```



```

    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/
dev/null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi
echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \

```

```

--region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
|| echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "=====
echo "RESOURCES CREATED"
echo "=====
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"

```

```

echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "===== "

# Auto-confirm cleanup
echo ""
echo "===== "
echo "CLEANUP CONFIRMATION"
echo "===== "
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
"$NAMED_QUERY_ID" \
    --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else
        echo "Named query deleted successfully."
    fi

    # Drop table
    echo "Dropping table: $TABLE_NAME"
    DROP_TABLE_RESULT=$(aws athena start-query-execution \
    --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

    if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
    else
        QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*"'
| cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for table deletion to complete..."

            ELAPSED=0

```

```

        while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
            QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
            if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                echo "Table dropped successfully."
                break
            elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
                echo "Warning: Table deletion failed with status:
$QUERY_STATUS"
                break
            fi
            echo "Table deletion in progress, status: $QUERY_STATUS"
            sleep 2
            ((ELAPSED+=2))
        done
    fi
fi

# Drop database
echo "Dropping database: $DATABASE_NAME"
DROP_DB_RESULT=$(aws athena start-query-execution \
    --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
    echo "Warning: Failed to drop database: $DROP_DB_RESULT"
else
    QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' |
cut -d'"' -f4)
    if [ -n "$QUERY_ID" ]; then
        echo "Waiting for database deletion to complete..."

        ELAPSED=0
        while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
            QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
            if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then

```

```

        echo "Database dropped successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
        echo "Warning: Database deletion failed with status:
$QUERY_STATUS"
        break
    fi
    echo "Database deletion in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done
fi
fi

# Empty and delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Emptying S3 bucket: $S3_BUCKET"
    EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
    if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
    else
        echo "S3 bucket emptied successfully."
    fi

    echo "Deleting S3 bucket: $S3_BUCKET"
    DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
    if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
    else
        echo "S3 bucket deleted successfully."
    fi
else
    echo "Skipping S3 bucket deletion (shared resource)"
fi

# Security: Remove downloaded query results
if [ -f "./query-results.csv" ]; then
    if command -v shred &>/dev/null; then
        shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
    else
        rm -f "./query-results.csv"
    fi
fi

```

```
        echo "Query results file securely removed."
    fi

    echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- For API details, see the following topics in *Amazon CLI Command Reference*.
 - [Cp](#)
 - [CreateNamedQuery](#)
 - [DeleteNamedQuery](#)
 - [GetNamedQuery](#)
 - [GetQueryExecution](#)
 - [GetQueryResults](#)
 - [ListDatabases](#)
 - [ListNamedQueries](#)
 - [ListTableMetadata](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)
 - [StartQueryExecution](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Getting started with Amazon EMR

The following code example shows how to:

- Create an EC2 key pair
- Set up storage and prepare your application
- Clean up resources

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Sample developer tutorials](#) repository.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
```

```

cleanup() {
    echo ""
    echo "=====
    echo "CLEANUP IN PROGRESS"
    echo "=====
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null ||
true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
true
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ];
then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials

```



```

if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
"Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-
gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
|| true

```

```
# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to
    find the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as
    's3://emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-
    tutorial-bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
    Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red
        violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
        total_red_violations
        FROM restaurant_violations
```

```

        WHERE violation_type = 'RED'
        GROUP BY name
        ORDER BY total_red_violations DESC LIMIT 10""")

    # Write the results to the specified output URI
    top_red_violation_restaurants.write.option("header",
"true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
"Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample
data"

```

```
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
  handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output
  text 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
  echo "No EC2 key pairs found. Creating a new key pair..."
  KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
  KEY_NAME_FILE="${KEY_NAME}.pem"
  aws ec2 create-key-pair --key-name "$KEY_NAME" \
    --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
    --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
  chmod 400 "$KEY_NAME_FILE"
  echo "Created new key pair: $KEY_NAME"
else
  # Use the first available key pair
  KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
  echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
```

```

--release-label emr-6.10.0 \
--applications Name=Spark \
--ec2-attributes KeyName="$KEY_NAME" \
--instance-type m5.xlarge \
--instance-count 3 \
--use-default-roles \
--log-uri "s3://$BUCKET_NAME/logs/" \
--ebs-root-volume-size 100 \
--tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
--security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:'
[[:space:]]*"^[^"]*" | grep -o 'j-[A-Z0-9]*' || true)
fi

if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
"Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then

```

```

        handle_error "Cluster did not reach WAITING state within timeout
period"
    fi
    sleep 30
    CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --
query "Cluster.Status.State" --output text)
    echo "Current cluster state: $CLUSTER_STATE"

    # Check for error states
    if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
"TERMINATED" ]]; then
        handle_error "Cluster entered error state: $CLUSTER_STATE"
    fi
    WAIT_COUNT=$((WAIT_COUNT + 1))
done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
    --cluster-id "$CLUSTER_ID" \
    --steps Type=Spark,Name="Health Violations
Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

```

```
echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
  handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id
"$STEP_ID" --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
  handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output
files"

# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv"
| head -1 || true)
if [ -z "$RESULT_FILE" ]; then
  echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
CSV file..."
  RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv"
| head -1 || true)
  if [ -z "$RESULT_FILE" ]; then
    echo "Listing all files in results directory:"
    aws s3 ls "s3://$BUCKET_NAME/results/"
    handle_error "No result file found in the output directory"
  fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
  handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
```

```
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./}
${KEY_NAME}.pem)"

# Display summary of created resources
echo ""
echo "======"
echo "RESOURCES CREATED"
echo "======"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup

echo "Script completed successfully."
```

- For API details, see the following topics in *Amazon CLI Command Reference*.
 - [AddSteps](#)
 - [Cp](#)
 - [CreateCluster](#)
 - [CreateDefaultRoles](#)
 - [CreateKeyPair](#)
 - [DescribeCluster](#)
 - [DescribeKeyPairs](#)
 - [DescribeStep](#)
 - [Ls](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)

- [Ssh](#)
- [TerminateClusters](#)
- [Wait](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Getting started with Amazon S3

The following code example shows how to:

- Create your first S3 bucket
- Upload an object
- Enable versioning
- Configure default encryption
- Add tags to your bucket
- List objects and versions
- Clean up resources

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Sample developer tutorials](#) repository.

```
#!/bin/bash
# S3 Getting Started - Create a bucket, upload and download objects, copy to a
# folder prefix, enable versioning, configure encryption and public access
# blocking, tag the bucket, list objects and versions, and clean up.

set -eE
set -o pipefail
```

```

# =====
# Prerequisites check
# =====

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
"$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Run 'aws configure' or set
AWS_DEFAULT_REGION."
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid. Run 'aws configure'."
    exit 1
fi

# =====
# Setup: logging, temp directory, resource tracking
# =====

UNIQUE_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || true)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="s3api-${UNIQUE_ID}"
fi

TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
LOG_FILE="$TEMP_DIR/s3-gettingstarted.log"
CREATED_RESOURCES=()

exec > >(tee -a "$LOG_FILE") 2>&1

```

```

echo "=====
echo "S3 Getting Started"
echo "=====
echo "Bucket name: ${BUCKET_NAME}"
echo "Temp directory: ${TEMP_DIR}"
echo "Log file: ${LOG_FILE}"
echo ""

# =====
# Helper functions
# =====

get_region() {
    echo "${AWS_REGION:-${AWS_DEFAULT_REGION:-${CONFIGURED_REGION}}}"
}

delete_object_versions() {
    local bucket=$1
    local query=$2

    local versions
    versions=$(aws s3api list-object-versions \
        --bucket "$bucket" \
        --query "$query" \
        --output json 2>&1) || return 0

    if [ -z "$versions" ] || [ "$versions" = "null" ] || [ "$versions" = "[]" ];
    then
        return 0
    fi

    echo "$versions" | jq -r '.[ ] | "\(.Key)\t\(.VersionId)"' 2>/dev/null | while
IFS=$'\t' read -r key version_id; do
        if [ -n "$key" ] && [ "$key" != "null" ]; then
            aws s3api delete-object --bucket "$bucket" --key "$key" --version-id
"$version_id" >/dev/null 2>&1 || true
        fi
    done

    return 0
}

# =====

```

```

# Error handling and cleanup functions
# =====

cleanup() {
    echo ""
    echo "=====
    echo "CLEANUP"
    echo "=====

    if [ "$BUCKET_IS_SHARED" = "false" ]; then
        echo "Deleting all object versions in bucket..."

        delete_object_versions "$BUCKET_NAME" "Versions[.
{Key:Key,VersionId:VersionId}" || true

        delete_object_versions "$BUCKET_NAME" "DeleteMarkers[.
{Key:Key,VersionId:VersionId}" || true

        echo "Deleting bucket: ${BUCKET_NAME}"
        if ! aws s3api delete-bucket --bucket "$BUCKET_NAME" 2>/dev/null; then
            echo "WARNING: Failed to delete bucket ${BUCKET_NAME}"
        fi

        # Clean up logs bucket
        LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
        if aws s3api head-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/null; then
            echo "Deleting log bucket: ${LOG_TARGET_BUCKET}"
            if ! aws s3api delete-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/
null; then
                echo "WARNING: Failed to delete bucket ${LOG_TARGET_BUCKET}"
            fi
        fi
    else
        echo "Keeping shared bucket: ${BUCKET_NAME}"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    local line_number=$1
    echo ""
    echo "=====

```

```

    echo "ERROR on line ${line_number}"
    echo "======"
    echo ""
    echo "Resources created before error:"
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        for RESOURCE in "${CREATED_RESOURCES[@}"; do
            echo "  - ${RESOURCE}"
        done
    else
        echo "  (none)"
    fi
    echo ""
    echo "Attempting cleanup..."
    cleanup
    exit 1
}

trap 'handle_error "$LINENO"' ERR

# =====
# Step 1: Create a bucket
# =====

echo "Step 1: Creating bucket ${BUCKET_NAME}..."
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        if ! aws s3api create-bucket --bucket "$BUCKET_NAME" >/dev/null 2>&1;
        then
            echo "ERROR: Failed to create bucket $BUCKET_NAME"
            exit 1
        fi
    else
        if ! aws s3api create-bucket \
            --bucket "$BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
        2>&1; then
            echo "ERROR: Failed to create bucket $BUCKET_NAME in region $REGION"
            exit 1
        fi
    fi
    CREATED_RESOURCES+=("s3:bucket:${BUCKET_NAME}")
    echo "Bucket created."

```

```

    if ! aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag bucket"
    fi
fi
echo ""

# =====
# Step 2: Upload a sample text file
# =====

echo "Step 2: Uploading a sample text file..."

SAMPLE_FILE="${TEMP_DIR}/sample.txt"
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is a sample file for the getting started tutorial.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload sample.txt"
    exit 1
fi
echo "File uploaded."
echo ""

# =====

```

```
# Step 3: Download the object
# =====

echo "Step 3: Downloading the object..."

DOWNLOAD_FILE="${TEMP_DIR}/downloaded-sample.txt"
if ! aws s3api get-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    "$DOWNLOAD_FILE" >/dev/null 2>&1; then
    echo "ERROR: Failed to download sample.txt"
    exit 1
fi
echo "Downloaded to: ${DOWNLOAD_FILE}"
echo "Contents:"
cat "$DOWNLOAD_FILE"
echo ""

# =====
# Step 4: Copy the object to a folder prefix
# =====

echo "Step 4: Copying object to a folder prefix..."

if ! aws s3api copy-object \
    --bucket "$BUCKET_NAME" \
    --copy-source "${BUCKET_NAME}/sample.txt" \
    --key "backup/sample.txt" \
    --server-side-encryption AES256 \
    --metadata-directive COPY >/dev/null 2>&1; then
    echo "ERROR: Failed to copy object to backup/sample.txt"
    exit 1
fi
echo "Object copied to backup/sample.txt."
echo ""

# =====
# Step 5: Enable versioning and upload a second version
# =====

echo "Step 5: Enabling versioning..."

if ! aws s3api put-bucket-versioning \
    --bucket "$BUCKET_NAME" \
```

```

        --versioning-configuration Status=Enabled >/dev/null 2>&1; then
        echo "ERROR: Failed to enable versioning"
        exit 1
    fi
    echo "Versioning enabled."

    echo "Uploading a second version of sample.txt..."
    cat > "$SAMPLE_FILE" << 'EOF'
    Hello, Amazon S3! This is version 2 of the sample file.
    EOF

    if ! aws s3api put-object \
        --bucket "$BUCKET_NAME" \
        --key "sample.txt" \
        --body "$SAMPLE_FILE" \
        --server-side-encryption AES256 \
        --metadata "tutorial=s3-gettingstarted,version=2" >/dev/null 2>&1; then
        echo "ERROR: Failed to upload second version of sample.txt"
        exit 1
    fi
    echo "Second version uploaded."
    echo ""

    # =====
    # Step 6: Configure SSE-S3 encryption
    # =====

    echo "Step 6: Configuring SSE-S3 default encryption..."

    if ! aws s3api put-bucket-encryption \
        --bucket "$BUCKET_NAME" \
        --server-side-encryption-configuration '{
            "Rules": [
                {
                    "ApplyServerSideEncryptionByDefault": {
                        "SSEAlgorithm": "AES256"
                    },
                    "BucketKeyEnabled": true
                }
            ]
        }' >/dev/null 2>&1; then
        echo "ERROR: Failed to configure SSE-S3 encryption"
        exit 1
    fi

```



```

echo "SSE-S3 encryption configured."
echo ""

# =====
# Step 7: Block all public access
# =====

echo "Step 7: Blocking all public access..."

if ! aws s3api put-public-access-block \
    --bucket "$BUCKET_NAME" \
    --public-access-block-configuration '{
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to block public access"
    exit 1
fi
echo "Public access blocked."
echo ""

# =====
# Step 8: Configure bucket logging
# =====

echo "Step 8: Configuring bucket logging..."

LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        aws s3api create-bucket --bucket "$LOG_TARGET_BUCKET" >/dev/null 2>&1 ||
true
    else
        aws s3api create-bucket \
            --bucket "$LOG_TARGET_BUCKET" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
2>&1 || true
    fi

    if ! aws s3api put-bucket-tagging \

```

```

        --bucket "$LOG_TARGET_BUCKET" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag log bucket"
    fi

    aws s3api put-bucket-acl --bucket "$LOG_TARGET_BUCKET" --acl log-delivery-
write 2>/dev/null || true

    if ! aws s3api put-bucket-logging \
        --bucket "$BUCKET_NAME" \
        --bucket-logging-status '{
            "LoggingEnabled": {
                "TargetBucket": "'$LOG_TARGET_BUCKET'",
                "TargetPrefix": "logs/"
            }
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to configure bucket logging"
    else
        echo "Bucket logging configured."
    fi
else
    echo "Skipping logging configuration for shared bucket."
fi
echo ""

# =====
# Step 9: Tag the bucket
# =====

echo "Step 9: Tagging the bucket..."

if ! aws s3api put-bucket-tagging \
    --bucket "$BUCKET_NAME" \

```

```

--tagging '{
    "TagSet": [
        {
            "Key": "project",
            "Value": "doc-smith"
        },
        {
            "Key": "tutorial",
            "Value": "s3-gettingstarted"
        },
        {
            "Key": "Environment",
            "Value": "Tutorial"
        },
        {
            "Key": "Project",
            "Value": "S3-GettingStarted"
        },
        {
            "Key": "ManagedBy",
            "Value": "Bash-Tutorial"
        }
    ]
}' >/dev/null 2>&1; then
echo "ERROR: Failed to tag bucket"
exit 1
fi
echo "Bucket tagged."

echo "Verifying tags..."
if ! aws s3api get-bucket-tagging --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to retrieve bucket tags"
fi
echo ""

# =====
# Step 10: List objects and versions
# =====

echo "Step 10: Listing objects..."

if ! aws s3api list-objects-v2 --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list objects"
fi

```

```

echo ""

echo "Listing object versions..."

if ! aws s3api list-object-versions --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list object versions"
fi
echo ""

# =====
# Step 11: Cleanup
# =====

echo ""
echo "=====
echo "TUTORIAL COMPLETE"
echo "=====
echo ""
echo "Resources created:"
if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
    for RESOURCE in "${CREATED_RESOURCES[@}"; do
        echo "  - ${RESOURCE}"
    done
else
    echo "  (none)"
fi
echo ""
echo "=====
echo "CLEANUP"
echo "=====
echo "Cleaning up all created resources..."
cleanup

echo ""
echo "Done."

```

- For API details, see the following topics in *Amazon CLI Command Reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)

- [GetObject](#)
- [HeadObject](#)
- [ListObjectVersions](#)
- [ListObjectsV2](#)
- [PutBucketEncryption](#)
- [PutBucketTagging](#)
- [PutBucketVersioning](#)
- [PutObject](#)
- [PutPublicAccessBlock](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Getting started with Amazon SageMaker Feature Store

The following code example shows how to:

- Set up IAM permissions
- Create a SageMaker execution role
- Create feature groups
- Clean up resources

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Sample developer tutorials](#) repository.

```
#!/bin/bash
```

```
# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"
    done
}
```

```

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
"$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

```

```
done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
    Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2
```



```

# SageMaker execution policy
local policy1_result=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

if echo "$policy1_result" | grep -i "error" > /dev/null; then
    handle_error "Failed to attach SageMaker policy: $policy1_result"
fi

# S3 access policy
local policy2_result=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

if echo "$policy2_result" | grep -i "error" > /dev/null; then
    handle_error "Failed to attach S3 policy: $policy2_result"
fi

# Get account ID for role ARN
local account_id=$(aws sts get-caller-identity --query Account --output text)
local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

echo "Role ARN: $role_arn" >&2
echo "Waiting 10 seconds for role to propagate..." >&2
sleep 10

# Return only the role ARN to stdout
echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\\///')

```

```
ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
    echo "Creating a new role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
```

```

    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
    'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
    "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else

```

```

    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-`${RANDOM_ID}`"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://`${S3_BUCKET_NAME}`/`${PREFIX}`"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-`${RANDOM_ID}`"

```

```

echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
  --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
  --record-identifier-feature-name "customer_id" \
  --event-time-feature-name "EventTime" \
  --feature-definitions '[
    {"FeatureName": "customer_id", "FeatureType": "Integral"},
    {"FeatureName": "order_id", "FeatureType": "String"},
    {"FeatureName": "order_date", "FeatureType": "String"},
    {"FeatureName": "product", "FeatureType": "String"},
    {"FeatureName": "quantity", "FeatureType": "Integral"},
    {"FeatureName": "amount", "FeatureType": "Fractional"},
    {"FeatureName": "EventTime", "FeatureType": "Fractional"}
  ]' \
  --online-store-config '{"EnableOnlineStore": true}' \
  --offline-store-config '{
    "S3StorageConfig": {
      "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
    },
    "DisableGlueTableCreation": false
  }' \
  --role-arn "$ROLE_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to create orders feature group: $ORDERS_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."

```

```

CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "573291"},
    {"FeatureName": "name", "ValueAsString": "John Doe"},
    {"FeatureName": "age", "ValueAsString": "35"},
    {"FeatureName": "address", "ValueAsString": "123 Main St"},
    {"FeatureName": "membership_type", "ValueAsString": "premium"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
  ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "109382"},
    {"FeatureName": "name", "ValueAsString": "Jane Smith"},
    {"FeatureName": "age", "ValueAsString": "28"},
    {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
    {"FeatureName": "membership_type", "ValueAsString": "standard"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
  ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "573291"},

```

```

        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

```

```
if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
| grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"
```



```

# Display summary of created resources
echo ""
echo "=====
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "=====
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "=====
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "=====
    echo "CLEANUP CONFIRMATION"
    echo "=====
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"

```

```
    echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
$ORDERS_FEATURE_GROUP_NAME"
    echo "- S3 Bucket: $S3_BUCKET_NAME"
    if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
        echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
'IAMRole:[^[:space:]]*' | cut -d: -f2)"
    fi
    echo ""
    echo "Estimated ongoing cost: ~$0.01 per month for online store"
fi
fi

echo "Script completed at $(date)"
```

- For API details, see the following topics in *Amazon CLI Command Reference*.

- [AttachRolePolicy](#)
- [CreateBucket](#)
- [CreateFeatureGroup](#)
- [CreateRole](#)
- [DeleteBucket](#)
- [DeleteFeatureGroup](#)
- [DeleteRole](#)
- [DescribeFeatureGroup](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [GetRole](#)
- [ListFeatureGroups](#)
- [PutPublicAccessBlock](#)
- [Rm](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Getting started with Amazon Textract

The following code example shows how to:

- Create an S3 bucket
- Upload a document to S3
- Clean up resources

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Sample developer tutorials](#) repository.

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====="
echo "Amazon Textract Getting Started Tutorial"
echo "====="
echo "This script will guide you through using Amazon Textract to analyze
document text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
```

```

    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-
Security-Token\).*\/1=***REDACTED***\/g'
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to
delete document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ];
then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then

```

```

        echo "ERROR: AWS CLI is not installed."
        exit 1
    fi

    AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
    AWS_CONFIG_STATUS=$?
    if [ $AWS_CONFIG_STATUS -ne 0 ]; then
        echo "ERROR: AWS CLI is not properly configured."
        echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization
\).*\/1=***REDACTED***/'
        exit 1
    fi

    # Verify AWS region is configured and supports Textract
    AWS_REGION=$(aws configure get region)
    if [ -z "$AWS_REGION" ]; then
        echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
        default region."
        exit 1
    fi

    # Check if Textract is available in the configured region
    echo "Checking if Amazon Textract is available in region $AWS_REGION..."
    TEXTTRACT_CHECK=$(aws textract help 2>&1)
    TEXTTRACT_CHECK_STATUS=$?
    if [ $TEXTTRACT_CHECK_STATUS -ne 0 ]; then
        echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
        exit 1
    fi

    # Generate a random identifier for S3 bucket
    RANDOM_ID=$(openssl rand -hex 6)
    # Check for shared prereq bucket
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
    bucket \
        --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
    text 2>/dev/null || echo "")
    if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
        BUCKET_NAME="$PREREQ_BUCKET"
        BUCKET_IS_SHARED=true
        echo "Using shared bucket: $BUCKET_NAME"
    else
        BUCKET_IS_SHARED=false
        BUCKET_NAME="textract-${RANDOM_ID}"
    fi

```

```

fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
{"SSEAlgorithm": "AES256"}]}]' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text
50,50 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAyAAAFfcSjAAAAADULEQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"

```

```
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."
UPLOAD_OUTPUT=$(aws s3 cp "$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256
2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp "$DOCUMENT_NAME" s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
    "S3Object": {
        "Bucket": "BUCKET_PLACEHOLDER",
        "Name": "DOCUMENT_PLACEHOLDER"
    }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES", "FORMS", "SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?
```

```

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
    echo "ERROR: Document analysis failed"
    echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*\/\1=***REDACTED***\/g'
    exit 1
fi

# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "=====
echo "Analysis Summary"
echo "=====
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk
'{print $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l ||
echo 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l ||
echo 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l ||
echo 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l ||
echo 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"'
| wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' |
wc -l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"

```



```

echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""

# Cleanup confirmation
echo ""
echo "=====
echo "RESOURCES CREATED"
echo "=====
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-
results.json) has been kept."

```

```
echo ""
echo "=====
echo "Tutorial complete!"
echo "=====
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"
echo ""

trap - EXIT
```

- For API details, see the following topics in *Amazon CLI Command Reference*.
 - [AnalyzeDocument](#)
 - [Cp](#)
 - [Help](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Getting started with Config

The following code example shows how to:

- Create an Amazon S3 bucket
- Create an Amazon SNS topic
- Create an IAM role for Config
- Set up the Config configuration recorder
- Set up the Config delivery channel
- Start the configuration recorder
- Verify the Config setup

Bash

Amazon CLI with Bash script

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Sample developer tutorials](#) repository.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
    fi
}
```

```
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
"true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
"$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
"true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
}
```

```

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "=====
    echo "CREATED RESOURCES"
    echo "=====
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "=====
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then

```

```
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1" # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"
```

```
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
    "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true"
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-`${RANDOM_ID}`"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
    Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-`${RANDOM_ID}`"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "config.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

```
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam:[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
```



```

        "sns:Publish"
    ],
    "Resource": "${SNS_TOPIC_ARN}"
}
]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-records 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
    "name": "${CONFIG_RECORDER_NAME}",
    "roleARN": "${ROLE_ARN}",
    "recordingMode": {
        "recordingFrequency": "CONTINUOUS"
    }
}

```

```

}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
    "allSupported": true,
    "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
    Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
    "name": "${DELIVERY_CHANNEL_NAME}",

```

```

    "s3BucketName": "${S3_BUCKET_NAME}",
    "snsTopicARN": "${SNS_TOPIC_ARN}",
    "configSnapshotDeliveryProperties": {
        "deliveryFrequency": "Six_Hours"
    }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
    "name": "${DELIVERY_CHANNEL_NAME}",
    "s3BucketName": "${S3_BUCKET_NAME}",
    "snsTopicARN": "${SNS_TOPIC_ARN}",
    "configSnapshotDeliveryProperties": {
        "deliveryFrequency": "Six_Hours"
    }
}
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
    Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."

```

```

RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "======"
echo "CLEANUP CONFIRMATION"
echo "======"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else

```

```
    echo "Resources will not be cleaned up. You can manually clean them up
    later."
fi

echo "Script completed successfully!"
```

- For API details, see the following topics in *Amazon CLI Command Reference*.
 - [AttachRolePolicy](#)
 - [CreateBucket](#)
 - [CreateRole](#)
 - [CreateTopic](#)
 - [DeleteBucket](#)
 - [DeleteConfigurationRecorder](#)
 - [DeleteDeliveryChannel](#)
 - [DeleteRole](#)
 - [DeleteRolePolicy](#)
 - [DeleteTopic](#)
 - [DescribeConfigurationRecorderStatus](#)
 - [DescribeConfigurationRecorders](#)
 - [DescribeDeliveryChannels](#)
 - [DetachRolePolicy](#)
 - [GetCallerIdentity](#)
 - [PutConfigurationRecorder](#)
 - [PutDeliveryChannel](#)
 - [PutPublicAccessBlock](#)
 - [PutRolePolicy](#)
 - [Rm](#)
 - [StartConfigurationRecorder](#)
 - [StopConfigurationRecorder](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and [details about previous SDK versions](#).

Work with Amazon S3 object lock features using an Amazon SDK

The following code examples show how to work with S3 object lock features.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Run an interactive scenario demonstrating Amazon S3 object lock features.

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
        Before running this .NET code example, set up your development environment,
        including your credentials.

        This .NET example performs the following tasks:
        1. Create test Amazon Simple Storage Service (S3) buckets with different
        lock policies.
        2. Upload sample objects to each bucket.
        3. Set some Legal Hold and Retention Periods on objects and buckets.
        4. Investigate lock policies by viewing settings or attempting to delete
        or overwrite objects.
        5. Clean up objects and buckets.
    */
}
```

```
public static S3ActionsWrapper _s3ActionsWrapper = null!;  
public static IConfiguration _configuration = null!;  
private static string _resourcePrefix = null!;  
private static string noLockBucketName = null!;  
private static string lockEnabledBucketName = null!;  
private static string retentionAfterCreationBucketName = null!;  
private static List<string> bucketNames = new List<string>();  
private static List<string> fileNames = new List<string>();  
  
public static async Task Main(string[] args)  
{  
    // Set up dependency injection for the Amazon service.  
    using var host = Host.CreateDefaultBuilder(args)  
        .ConfigureLogging(logging =>  
            logging.AddFilter("System", LogLevel.Debug)  
                .AddFilter<DebugLoggerProvider>("Microsoft",  
LogLevel.Information)  
                .AddFilter<ConsoleLoggerProvider>("Microsoft",  
LogLevel.Trace))  
        .ConfigureServices((_, services) =>  
            services.AddAWSService<IAmazonS3>()  
                .AddTransient<S3ActionsWrapper>()  
        )  
        .Build();  
  
    _configuration = new ConfigurationBuilder()  
        .SetBasePath(Directory.GetCurrentDirectory())  
        .AddJsonFile("settings.json") // Load settings from .json file.  
        .AddJsonFile("settings.local.json",  
            true) // Optionally, load local settings.  
        .Build();  
  
    ConfigurationSetup();  
  
    ServicesSetup(host);  
  
    try  
    {  
        Console.WriteLine(new string('-', 80));  
        Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)  
Object Locking Feature Scenario.");  
        Console.WriteLine(new string('-', 80));  
        await Setup(true);  
    }
```

```

        await DemoActionChoices();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cleaning up resources.");
        Console.WriteLine(new string('-', 80));
        await Cleanup(true);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Amazon S3 Object Locking Scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"There was a problem: {ex.Message}");
        await Cleanup(true);
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Populate the services for use within the console application.
/// </summary>
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
}

/// <summary>
/// Any setup operations needed.
/// </summary>
public static void ConfigurationSetup()
{
    _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

    noLockBucketName = _resourcePrefix + "-no-lock";
    lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
    retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-creation";

    bucketNames.Add(noLockBucketName);
    bucketNames.Add(lockEnabledBucketName);
}

```



```

        bucketNames.Add(retentionAfterCreationBucketName);
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Setup(bool interactive)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create
several S3\n" +
            "buckets and files to demonstrate working with S3 locking features.
\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nS3 buckets can be created either with or without
object lock enabled.");
        await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName,
false);
        await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
true);
        await
_s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
false);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nA bucket can be configured to use object locking
with a default retention period.");
        await
_s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,
true,
            ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));

        Console.WriteLine("Press Enter to continue.");
        if (interactive)

```

```
        Console.ReadLine();

        Console.WriteLine("\nObject lock policies can also be added to existing
buckets.");
        await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        // Upload some files to the buckets.
        Console.WriteLine("\nNow let's add some test files:");
        var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";
        int fileCount = 2;
        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a bucket.");
        }

        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileCount; i++)
            {
                var numberedFileName = Path.GetFileNameWithoutExtension(fileName)
+ i + Path.GetExtension(fileName);
                fileNames.Add(numberedFileName);
                await _s3ActionsWrapper.UploadFileAsync(bucketName,
numberedFileName, fileName);
            }
        }
        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        if (!interactive)
            return true;
        Console.WriteLine("\nNow we can set some object lock policies on
individual files:");
        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileNames.Count; i++)
```

```

    {
        // No modifications to the objects in the first bucket.
        if (bucketName != bucketNames[0])
        {
            var exampleFileName = fileNames[i];
            switch (i)
            {
                case 0:
                {
                    var question =
                        $"\\nWould you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                    if (GetYesNoResponse(question))
                    {
                        // Set a legal hold.
                        await
                        _s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
                        ObjectLockLegalHoldStatus.On);

                    }
                    break;
                }
                case 1:
                {
                    var question =
                        $"\\nWould you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
                        "\\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
                    if (GetYesNoResponse(question))
                    {
                        // Set a Governance mode retention period for
1 day.
                        await
                        _s3ActionsWrapper.ModifyObjectRetentionPeriod(
                            bucketName, exampleFileName,
                            ObjectLockRetentionMode.Governance,
                            DateTime.UtcNow.AddDays(1));
                    }
                    break;
                }
            }
        }
    }
}

```

```

        }
    }
    Console.WriteLine(new string('-', 80));
    return true;
}

// <summary>
/// List all of the current buckets and objects.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>The list of buckets and objects.</returns>
public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
{
    var allObjects = new List<S3ObjectVersion>();
    foreach (var bucketName in bucketNames)
    {
        var objectsInBucket = await
        _s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
        foreach (var objectKey in objectsInBucket.Versions)
        {
            allObjects.Add(objectKey);
        }
    }

    if (interactive)
    {
        Console.WriteLine("\nCurrent buckets and objects:\n");
        int i = 0;
        foreach (var bucketObject in allObjects)
        {
            i++;
            Console.WriteLine(
                $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
        }
    }

    return allObjects;
}

/// <summary>
/// Present the user with the demo action choices.
/// </summary>

```

```

/// <returns>Async task.</returns>
public static async Task<bool> DemoActionChoices()
{
    var choices = new string[]{
        "List all files in buckets.",
        "Attempt to delete a file.",
        "Attempt to delete a file with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the object and bucket retention settings for a file.",
        "View the legal hold settings for a file.",
        "Finish the scenario."};

    var choice = 0;
    // Keep asking the user until they choose to move on.
    while (choice != 6)
    {
        Console.WriteLine(new string('-', 80));
        choice = GetChoiceResponse(
            "\nExplore the S3 locking features by selecting one of the
following choices:"
            , choices);
        Console.WriteLine(new string('-', 80));
        switch (choice)
        {
            case 0:
            {
                await ListBucketsAndObjects(true);
                break;
            }
            case 1:
            {
                Console.WriteLine("\nEnter the number of the object to
delete:");

                var allFiles = await ListBucketsAndObjects(true);
                var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
                break;
            }
            case 2:
            {

```

```

        Console.WriteLine("\nEnter the number of the object to
delete:");

        var allFiles = await ListBucketsAndObjects(true);
        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
        break;
    }
    case 3:
    {
        var allFiles = await ListBucketsAndObjects(true);
        Console.WriteLine("\nEnter the number of the object to
overwrite:");

        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        // Create the file if it does not already exist.
        if (!File.Exists(allFiles[fileChoice].Key))
        {
            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a
bucket.");
        }
        await
_s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
        break;
    }
    case 4:
    {
        var allFiles = await ListBucketsAndObjects(true);
        Console.WriteLine("\nEnter the number of the object and
bucket to view:");

        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
_s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
        await
_s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
        break;
    }

```

```

        }
        case 5:
        {
            var allFiles = await ListBucketsAndObjects(true);
            Console.WriteLine("\nEnter the number of the object to
view:");

            var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
            await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
            break;
        }
    }
}
return true;
}

// <summary>
/// Clean up the resources from the scenario.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Cleanup(bool interactive)
{
    Console.WriteLine(new string('-', 80));

    if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
    {
        // Remove all locks and delete all buckets and objects.
        var allFiles = await ListBucketsAndObjects(false);
        foreach (var fileInfo in allFiles)
        {
            // Check for a legal hold.
            var legalHold = await
_s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
            if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
            {
                await
_s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
ObjectLockLegalHoldStatus.Off);
            }
        }
    }
}

```

```

        // Check for a retention period.
        var retention = await
        _s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);
        var hasRetentionPeriod = retention?.Mode ==
        ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
        DateTime.UtcNow.Date;
        await
        _s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key,
        hasRetentionPeriod, fileInfo.VersionId);
    }

    foreach (var bucketName in bucketNames)
    {
        await _s3ActionsWrapper.DeleteBucketByName(bucketName);
    }

}
else
{
    Console.WriteLine(
        "Ok, we'll leave the resources intact.\n" +
        "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
    );
}

Console.WriteLine(new string('-', 80));
return true;
}

/// <summary>
/// Helper method to get a yes or no response from the user.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>
/// <returns>True if the user responds with a yes.</returns>
private static bool GetYesNoResponse(string question)
{
    Console.WriteLine(question);
    var ynResponse = Console.ReadLine();
    var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
    return response;
}

```



```
/// <summary>
/// Helper method to get a choice response from the user.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>
/// <param name="choices">The choices to print on the console.</param>
/// <returns>The index of the selected choice</returns>
private static int GetChoiceResponse(string? question, string[] choices)
{
    if (question != null)
    {
        Console.WriteLine(question);

        for (int i = 0; i < choices.Length; i++)
        {
            Console.WriteLine($"{i + 1}. {choices[i]}");
        }
    }

    var choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > choices.Length)
    {
        var choice = Console.ReadLine();
        Int32.TryParse(choice, out choiceNumber);
    }

    return choiceNumber - 1;
}
}
```

A wrapper class for S3 functions.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
```

```
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
    bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
    enableObjectLock)
    {
        Console.WriteLine($"\\tCreating bucket {bucketName} with object lock
    {enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
}
```

```

    }
}

/// <summary>
/// Enable object lock on an existing bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to modify.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableObjectLockOnBucket(string bucketName)
{
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
            },
        };

        var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"{bucketName}\tAdded an object lock policy to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
        return false;
    }
}

```

```

    }

    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
    retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>

```

```

    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in
days or years but not both.
                        }
                    }
                }
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"{bucketName}.");

```

```

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tError modifying object lock: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Get the retention period for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object retention details.</returns>
public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };

        var response = await _amazonS3.GetObjectRetentionAsync(request);
        Console.WriteLine($"\\tObject retention for {objectKey} in
{bucketName}: " +
                        $"\\n\\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}."");
        return response.Retention;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch object lock retention:
'{ex.Message}'");
        return new ObjectLockRetention();
    }
}

/// <summary>
/// Set or modify a legal hold on an object in an S3 bucket.
/// </summary>

```

```

    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {

```

```

        BucketName = bucketName,
        Key = objectKey
    };

    var response = await _amazonS3.GetObjectLegalHoldAsync(request);
    Console.WriteLine($"\\tObject legal hold for {objectKey} in
{bucketName}: " +
        $"\\n\\tStatus: {response.LegalHold.Status}");
    return response.LegalHold;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tUnable to fetch legal hold: '{ex.Message}'");
    return new ObjectLockLegalHold();
}
}

/// <summary>
/// Get the object lock configuration details for an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to get details.</param>
/// <returns>The bucket's object lock configuration details.</returns>
public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
{
    try
    {
        var request = new GetObjectLockConfigurationRequest()
        {
            BucketName = bucketName
        };

        var response = await
        _amazonS3.GetObjectLockConfigurationAsync(request);
        Console.WriteLine($"\\tBucket object lock config for {bucketName} in
{bucketName}: " +
            $"\\n\\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
            $"\\n\\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

        return response.ObjectLockConfiguration;
    }
    catch (AmazonS3Exception ex)

```



```

        {
            Console.WriteLine($"\\tUnable to fetch object lock config:
'{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }

    /// <summary>
    /// Upload a file from the local computer to an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object to
upload.</param>
    /// <returns>True if success.</returns>
    public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
            ChecksumAlgorithm = ChecksumAlgorithm.SHA256
        };

        var response = await _amazonS3.PutObjectAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"\\tSuccessfully uploaded {objectName} to
{bucketName}.");
            return true;
        }
        else
        {
            Console.WriteLine($"\\tCould not upload {objectName} to
{bucketName}.");
            return false;
        }
    }

    /// <summary>
    /// List bucket objects and versions.
    /// </summary>

```

```

    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <returns>The list of objects and versions.</returns>
    public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
bucketName)
    {
        var request = new ListVersionsRequest()
        {
            BucketName = bucketName
        };

        var response = await _amazonS3.ListVersionsAsync(request);
        return response;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey, bool hasRetention, string? versionId = null)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
    }

```

```

    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
        return false;
    }
}

/// <summary>
/// Delete a specific bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <param name="versionId">Optional versionId.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteBucketByName(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest() { BucketName = bucketName, };
        var response = await _amazonS3.DeleteBucketAsync(request);
        Console.WriteLine($"\\tDelete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
ex.Message);
        return false;
    }
}
}

```

- For API details, see the following topics in *Amazon SDK for .NET API Reference*.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)

- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Run an interactive scenario demonstrating Amazon S3 object lock features.

```
import (  
    "context"  
    "fmt"  
    "log"  
    "strings"  
  
    "s3_object_lock/actions"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"  
)  
  
// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.  
type ObjectLockScenario struct {  
    questioner demotools.IQuestioner  
    resources   Resources  
    s3Actions   *actions.S3Actions  
    sdkConfig   aws.Config  
}  
  
// NewObjectLockScenario constructs a new ObjectLockScenario instance.
```

```

func NewObjectLockScenario(sdkConfig aws.Config, questioner
demotools.IQuestioner) ObjectLockScenario {
    scenario := ObjectLockScenario{
        questioner: questioner,
        resources: Resources{},
        s3Actions: &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
        sdkConfig: sdkConfig,
    }
    scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
    scenario.resources.init(scenario.s3Actions, questioner)
    return scenario
}

type nameLocked struct {
    name string
    locked bool
}

var createInfo = []nameLocked{
    {"standard-bucket", false},
    {"lock-bucket", true},
    {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
    log.Println("Let's create some S3 buckets to use for this workflow.")
    success := false
    for !success {
        prefix := scenario.questioner.Ask(
            "This example creates three buckets. Enter a prefix to name your buckets
(remember bucket names must be globally unique):")

        for _, info := range createInfo {
            log.Println(fmt.Sprintf("%s.%s", prefix, info.name))
            bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx,
                fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
            if err != nil {
                switch err.(type) {
                    case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
                        log.Printf("Couldn't create bucket %s.\n", bucketName)
                    default:
                        panic(err)
                }
            }
        }
    }
}

```

```

        break
    }
    scenario.resources.demoBuckets[info.name] = &DemoBucket{
        name:      bucketName,
        objectKeys: []string{},
    }
    log.Printf("Created bucket %s.\n", bucketName)
}

if len(scenario.resources.demoBuckets) < len(createInfo) {
    scenario.resources.deleteBuckets(ctx)
} else {
    success = true
}
}

log.Println("S3 buckets created.")
log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
    log.Println("\nA bucket can be configured to use object locking.")
    scenario.questioner.Ask("Press Enter to continue.")

    var err error
    bucket := scenario.resources.demoBuckets["retention-bucket"]
    err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
        default:
            panic(err)
        }
    } else {
        log.Printf("Object locking enabled on bucket %s.", bucket.name)
    }

    log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a
bucket.
```

```

func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx
context.Context) {
    log.Println("\nA bucket can be configured to use object locking with a default
retention period.")

    bucket := scenario.resources.demoBuckets["retention-bucket"]
    retentionPeriod := scenario.questioner.AskInt("Enter the default retention
period in days: ")
    err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
types.ObjectLockRetentionModeGovernance)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("Couldn't configure a default retention period on bucket %s.\n",
bucket.name)
        default:
            panic(err)
        }
    } else {
        log.Printf("Default retention policy set on bucket %s with %d day retention
period.", bucket.name, retentionPeriod)
        bucket.retentionEnabled = true
    }

    log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
    log.Println("Uploading test objects to S3 buckets.")

    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        for i := 0; i < 2; i++ {
            key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
fmt.Sprintf("example-%d", i),
            fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
            if err != nil {
                switch err.(type) {
                case *types.NoSuchBucket:
                    log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
                default:
                    panic(err)
                }
            }
        }
    }
}

```

```

    }
    } else {
        log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
        bucket.objectKeys = append(bucket.objectKeys, key)
    }
}
}

scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test
// objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx
context.Context) {
    log.Println("Now let's set object lock configurations on individual objects.")

    buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
scenario.resources.demoBuckets["retention-bucket"]}
    for _, bucket := range buckets {
        for index, objKey := range bucket.objectKeys {
            switch index {
            case 0:
                if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold
to %s in %s (y/n)? ", objKey, bucket.name), "y") {
                    err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
types.ObjectLockLegalHoldStatusOn)
                    if err != nil {
                        switch err.(type) {
                        case *types.NoSuchKey:
                            log.Printf("Couldn't set legal hold on %s.\n", objKey)
                        default:
                            panic(err)
                        }
                    } else {
                        log.Printf("Legal hold set on %s.\n", objKey)
                    }
                }
            case 1:
                q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to
%s in %s?\n"+
                    "Reminder: Only a user with the s3:BypassGovernanceRetention permission is
able to delete this object\n"+

```



```

        "or its bucket until the retention period has expired. (y/n) ", objKey,
        bucket.name)
        if scenario.questioner.AskBool(q, "y") {
            err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
            types.ObjectLockRetentionModeGovernance, 1)
            if err != nil {
                switch err.(type) {
                    case *types.NoSuchKey:
                        log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
                        bucket.name)
                        default:
                            panic(err)
                }
            } else {
                log.Printf("Retention period set to 1 for %s.", objKey)
                bucket.retentionEnabled = true
            }
        }
    }
}
log.Println(strings.Repeat("-", 88))
}

const (
    ListAll = iota
    DeleteObject
    DeleteRetentionObject
    OverwriteObject
    ViewRetention
    ViewLegalHold
    Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
// object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
    log.Println("Now you can interact with the objects to explore the object lock
    configurations.")
    interactiveChoices := []string{
        "List all objects and buckets.",
        "Attempt to delete an object.",
        "Attempt to delete an object with retention period bypass.",
        "Attempt to overwrite a file.",
    }

```

```

"View the retention settings for an object.",
"View the legal hold settings for an object.",
"Finish the workflow."}

choice := ListAll
for choice != Finish {
    objList := scenario.GetAllObjects(ctx)
    objChoices := scenario.makeObjectChoiceList(objList)
    choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
interactiveChoices)
    switch choice {
    case ListAll:
        log.Println("The current objects in the example buckets are:")
        for _, objChoice := range objChoices {
            log.Println("\t", objChoice)
        }
    case DeleteObject, DeleteRetentionObject:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
delete:\n", objChoices)
        obj := objList[objChoice]
        deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
obj.versionId, choice == DeleteRetentionObject)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Println("Nothing to delete.")
            default:
                panic(err)
            }
        } else if deleted {
            log.Printf("Object %s deleted.\n", obj.key)
        }
    case OverwriteObject:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
        obj := objList[objChoice]
        _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Println("Couldn't upload to nonexistent bucket.")
            default:
                panic(err)
            }
        }
    }
}

```

```
    }
    } else {
        log.Printf("Uploaded new content to object %s.\n", obj.key)
    }
    case ViewRetention:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
        obj := objList[objChoice]
        retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket,
obj.key)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Can't get retention configuration for %s.\n", obj.key)
            default:
                panic(err)
            }
        } else if retention != nil {
            log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
retention.Mode, retention.RetainUntilDate)
        } else {
            log.Printf("Object %s does not have object retention configured.\n", obj.key)
        }
    case ViewLegalHold:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
        obj := objList[objChoice]
        legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket,
obj.key, obj.versionId)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
            default:
                panic(err)
            }
        } else if legalHold != nil {
            log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
        } else {
            log.Printf("Object %s does not have legal hold configured.", obj.key)
        }
    case Finish:
        log.Println("Let's clean up.")
    }
}
```

```

    log.Println(strings.Repeat("-", 88))
}
}

type BucketKeyVersionId struct {
    bucket    string
    key       string
    versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns
// them in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
[]BucketKeyVersionId {
    var objectList []BucketKeyVersionId
    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Printf("Couldn't get object versions for %s.\n", bucket.name)
            default:
                panic(err)
            }
        } else {
            for _, version := range versions {
                objectList = append(objectList,
                    BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
                    *version.VersionId})
            }
        }
    }
    return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
// are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
[]BucketKeyVersionId) []string {
    choices := make([]string, len(bucketObjects))
    for i := 0; i < len(bucketObjects); i++ {
        choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",

```

```

        bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
    }
    return choices
}

// Run runs the S3 Object Lock scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := scenario.questioner.(*demotools.MockQuestioner)
            if isMock || scenario.questioner.AskBool("Do you want to see the full error
message (y/n)?", "y") {
                log.Println(r)
            }
            scenario.resources.Cleanup(ctx)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon S3 Object Lock Feature Scenario.")
    log.Println(strings.Repeat("-", 88))

    scenario.CreateBuckets(ctx)
    scenario.EnableLockOnBucket(ctx)
    scenario.SetDefaultRetentionPolicy(ctx)
    scenario.UploadTestObjects(ctx)
    scenario.SetObjectLockConfigurations(ctx)
    scenario.InteractWithObjects(ctx)

    scenario.resources.Cleanup(ctx)

    log.Println(strings.Repeat("-", 88))
    log.Println("Thanks for watching!")
    log.Println(strings.Repeat("-", 88))
}

```

Define a struct that wraps S3 actions used in this example.

```
import (
```

```

"bytes"
"context"
"errors"
"fmt"
"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// CreateBucketWithLock creates a new S3 bucket with optional object locking
// enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
    region string, enableObjectLock bool) (string, error) {
    input := &s3.CreateBucketInput{
        Bucket: aws.String(bucket),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    }

    if enableObjectLock {
        input.ObjectLockEnabledForBucket = aws.Bool(true)
    }

    _, err := actor.S3Client.CreateBucket(ctx, input)
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", bucket)

```

```

    err = owned
} else if errors.As(err, &exists) {
    log.Printf("Bucket %s already exists.\n", bucket)
    err = exists
}
} else {
err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
    ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
if err != nil {
    log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
}
}

return bucket, err
}

// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
var status *types.ObjectLockLegalHoldStatus
input := &s3.GetObjectLegalHoldInput{
    Bucket:    aws.String(bucket),
    Key:       aws.String(key),
    VersionId: aws.String(versionId),
}

output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
if err != nil {
    var noSuchKeyErr *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noSuchKeyErr) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noSuchKeyErr
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            log.Printf("Object %s does not have an object lock configuration.\n", key)
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    }
}

```

```
    }
  } else {
    status = &output.LegalHold.Status
  }

  return status, err
}

// GetObjectLockConfiguration retrieves the object lock configuration for an S3
// bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
  string) (*types.ObjectLockConfiguration, error) {
  var lockConfig *types.ObjectLockConfiguration
  input := &s3.GetObjectLockConfigurationInput{
    Bucket: aws.String(bucket),
  }

  output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
  if err != nil {
    var noBucket *types.NoSuchBucket
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noBucket) {
      log.Printf("Bucket %s does not exist.\n", bucket)
      err = noBucket
    } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
      "ObjectLockConfigurationNotFoundError" {
      log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
      err = nil
    }
  } else {
    lockConfig = output.ObjectLockConfiguration
  }

  return lockConfig, err
}

// GetObjectRetention retrieves the object retention configuration for an S3
// object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
  string) (*types.ObjectLockRetention, error) {
```



```

var retention *types.ObjectLockRetention
input := &s3.GetObjectRetentionInput{
    Bucket: aws.String(bucket),
    Key:    aws.String(key),
}

output, err := actor.S3Client.GetObjectRetention(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have locking enabled.", bucket)
            err = nil
        }
    }
} else {
    retention = output.Retention
}

return retention, err
}

// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
{
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:    aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
}

```

```
}

_, err := actor.S3Client.PutObjectLegalHold(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    }
}

return err
}

// ModifyDefaultBucketRetention modifies the default retention period of an
// existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    }

    _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }

    return err
}
```

```
}

// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
string) error {
    // Versioning must be enabled on the bucket before object locking is enabled.
    verInput := &s3.PutBucketVersioningInput{
        Bucket: aws.String(bucket),
        VersioningConfiguration: &types.VersioningConfiguration{
            MFADelete: types.MFADeleteDisabled,
            Status:    types.BucketVersioningStatusEnabled,
        },
    }
    _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
        return err
    }

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: types.ObjectLockEnabledEnabled,
        },
    }
    _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }

    return err
}
```

```
// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
    string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
    error {
    input := &s3.PutObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        Retention: &types.ObjectLockRetention{
            Mode:          retentionMode,
            RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
        },
        BypassGovernanceRetention: aws.Bool(true),
    }

    _, err := actor.S3Client.PutObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}

// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
    string, contents string) (string, error) {
    var outKey string
    input := &s3.PutObjectInput{
        Bucket:          aws.String(bucket),
        Key:             aws.String(key),
        Body:            bytes.NewReader([]byte(contents)),
        ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
    }
    output, err := actor.S3Manager.Upload(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
        }
    }
}
```

```
    err = noBucket
}
} else {
    err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
            bucket)
    } else {
        outKey = *output.Key
    }
}
return outKey, err
}

// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
([]types.ObjectVersion, error) {
    var err error
    var output *s3.ListObjectVersionsOutput
    var versions []types.ObjectVersion
    input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
    versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
    for versionPaginator.HasMorePages() {
        output, err = versionPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
            break
        } else {
            versions = append(versions, output.Versions...)
        }
    }
    return versions, err
}
```

```
// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
string, versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "AccessDenied":
                log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
                err = nil
            case "InvalidArgument":
                if bypassGovernance {
                    log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
                    err = nil
                }
            }
        }
    } else {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
        } else {
            deleted = true
        }
    }
}
```

```

    }
}
return deleted, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }

    input := s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),
        Delete: &types.Delete{
            Objects: objects,
            Quiet:   aws.Bool(true),
        },
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
    if err != nil || len(delOut.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucket)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
        }
        } else if len(delOut.Errors) > 0 {
            for _, outErr := range delOut.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *delOut.Errors[0].Message)
        }
    } else {
        for _, delObjs := range delOut.Deleted {
            err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
                time.Minute)
        }
    }
}

```

```
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to be deleted.\n",
*del0bjs.Key)
    } else {
        log.Printf("Deleted %s.\n", *del0bjs.Key)
    }
}
}
return err
}
```

Clean up resources.

```
import (
    "context"
    "log"
    "s3_object_lock/actions"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
    name            string
    retentionEnabled bool
    objectKeys      []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario
and handles
// cleanup when the scenario finishes.
type Resources struct {
    demoBuckets map[string]*DemoBucket

    s3Actions *actions.S3Actions
}
```



```

    questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
    demotools.IQuestioner) {
    resources.s3Actions = s3Actions
    resources.questioner = questioner
    resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong during cleanup.\n%v\n", r)
            log.Println("Use the AWS Management Console to remove any remaining resources"
                " +
                "that were created for this scenario.")
        }
    }()

    wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
        resources that were created "+
        "during this demo (y/n)?", "y")
    if !wantDelete {
        log.Println("Be sure to remove resources when you're done with them to avoid
            unexpected charges!")
        return
    }

    log.Println("Removing objects from S3 buckets and deleting buckets...")
    resources.deleteBuckets(ctx)
    //resources.deleteRetentionObjects(resources.retentionBucket,
    resources.retentionObjects)

    log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
    for _, info := range createInfo {
        bucket := resources.demoBuckets[info.name]

```

```

resources.deleteObjects(ctx, bucket)
_, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
    Bucket: aws.String(bucket.name),
})
if err != nil {
    panic(err)
}
}
for _, info := range createInfo {
    bucket := resources.demoBuckets[info.name]
    err := s3.NewBucketNotExistsWaiter(resources.s3Actions.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket.name)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
            bucket.name)
    } else {
        log.Printf("Deleted %s.\n", bucket.name)
    }
}
resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket
    *DemoBucket) {
    lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx,
        bucket.name)
    if err != nil {
        panic(err)
    }
    versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("No objects to get from %s.\n", bucket.name)
        default:
            panic(err)
        }
    }
    delObjects := make([]types.ObjectIdentifier, len(versions))
    for i, version := range versions {
        if lockConfig != nil && lockConfig.ObjectLockEnabled ==
            types.ObjectLockEnabledEnabled {

```

```

    status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
*version.Key, *version.VersionId)
    if err != nil {
        switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Couldn't determine legal hold status for %s in %s.\n",
*version.Key, bucket.name)
            default:
                panic(err)
        }
    } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
        err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
*version.VersionId, types.ObjectLockLegalHoldStatusOff)
        if err != nil {
            switch err.(type) {
                case *types.NoSuchKey:
                    log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
bucket.name)
                default:
                    panic(err)
            }
        }
    }
    del0bjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
version.VersionId}
}
err = resources.s3Actions.DeleteObjects(ctx, bucket.name, del0bjects,
bucket.retentionEnabled)
if err != nil {
    switch err.(type) {
        case *types.NoSuchBucket:
            log.Println("Nothing to delete.")
        default:
            panic(err)
    }
}
}

```

- For API details, see the following topics in *Amazon SDK for Go API Reference*.
 - [GetObjectLegalHold](#)

- [GetObjectLockConfiguration](#)
- [GetObjectRetention](#)
- [PutObjectLegalHold](#)
- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Run an interactive scenario demonstrating Amazon S3 object lock features.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;
```

/*

Before running this Java V2 code example, set up your development environment, including your credentials.

For more information, see the following documentation topic:

<https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html>

This Java example performs the following tasks:

1. Create test Amazon Simple Storage Service (S3) buckets with different lock policies.
2. Upload sample objects to each bucket.
3. Set some Legal Hold and Retention Periods on objects and buckets.

4. Investigate lock policies by viewing settings or attempting to delete or overwrite objects.
5. Clean up objects and buckets.

```
*/
```

```
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <bucketName> \s

        Where:
            bucketName - The Amazon S3 bucket name.
        """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        s3LockActions = new S3LockActions();
        bucketName = args[0];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Feature Scenario.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);
    }
}
```

```
        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by
selecting one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
```

```
        case 0 -> {
            s3LockActions.listBucketsAndObjects(bucketNames, true);
        }

        case 1 -> {
            System.out.println("Enter the number of the object to
delete:");

            List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
            List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
            String[] fileKeysArray = fileKeys.toArray(new String[0]);
            int fileChoice = getChoiceResponse(null, fileKeysArray);
            String objectKey = fileKeys.get(fileChoice);
            String bucketName = allFiles.get(fileChoice).getBucketName();
            String version = allFiles.get(fileChoice).getVersion();
            s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
        }

        case 2 -> {
            System.out.println("Enter the number of the object to
delete:");

            List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
            List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
            String[] fileKeysArray = fileKeys.toArray(new String[0]);
            int fileChoice = getChoiceResponse(null, fileKeysArray);
            String objectKey = fileKeys.get(fileChoice);
            String bucketName = allFiles.get(fileChoice).getBucketName();
            String version = allFiles.get(fileChoice).getVersion();
            s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
        }

        case 3 -> {
            System.out.println("Enter the number of the object to
overwrite:");

            List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
            List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
            String[] fileKeysArray = fileKeys.toArray(new String[0]);
```

```
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();

        // Attempt to overwrite the file.
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
            writer.write("This is a modified text.");

        } catch (IOException e) {
            e.printStackTrace();
        }
        s3LockActions.uploadFile(bucketName, objectKey, objectKey);
    }

    case 4 -> {
        System.out.println("Enter the number of the object to
overwrite:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectRetention(bucketName, objectKey);
    }

    case 5 -> {
        System.out.println("Enter the number of the object to
view:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectLegalHold(bucketName, objectKey);
        s3LockActions.getBucketObjectLockConfiguration(bucketName);
    }
}
```



```
        case 6 -> {
            System.out.println("Exiting the workflow...");
            return;
        }

        default -> {
            System.out.println("Invalid choice. Please select again.");
        }
    }
}

// Clean up the resources from the scenario.
private static void cleanup() {
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
    for (S3InfoObject fileInfo : allFiles) {
        String bucketName = fileInfo.getBucketName();
        String key = fileInfo.getKeyName();
        String version = fileInfo.getVersion();
        if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
            ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
            if (legalHold != null) {
                String holdStatus = legalHold.status().name();
                System.out.println(holdStatus);
                if (holdStatus.compareTo("ON") == 0) {
                    s3LockActions.modifyObjectLegalHold(bucketName, key,
false);
                }
            }
            // Check for a retention period.
            ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
            boolean hasRetentionPeriod ;
            hasRetentionPeriod = retention != null;
            s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);
        } else {
            System.out.println(bucketName +" objects do not have a legal
lock");
        }
    }
}
```

```
s3LockActions.deleteObjectFromBucket(bucketName, key, false,
version);
    }
}

// Delete the buckets.
System.out.println("Delete "+bucketName);
for (String bucket : bucketNames){
    s3LockActions.deleteBucketByName(bucket);
}
}

private static void setup() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("""
        For this workflow, we will use the AWS SDK for Java to create
several S3
        buckets and files to demonstrate working with S3 locking
features.
        """);

    System.out.println("S3 buckets can be created either with or without
object lock enabled.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();

    // Create three S3 buckets.
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
    s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Bucket "+bucketNames.get(2) +" will be configured to
use object locking with a default retention period.");
    s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
    s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();
}
```

```
// Upload some files to the buckets.
System.out.println("Now let's add some test files:");
String fileName = "exampleFile.txt";
int fileCount = 2;
try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
    writer.write("This is a sample file for uploading to a bucket.");

} catch (IOException e) {
    e.printStackTrace();
}

for (String bucketName : bucketNames){
    for (int i = 0; i < fileCount; i++) {
        // Get the file name without extension.
        String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
        int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
        if (extensionIndex > 0) {
            fileNameWithoutExtension =
fileNameWithoutExtension.substring(0, extensionIndex);
        }

        // Create the numbered file names.
        String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
        fileNames.add(numberedFileName);
        s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
    }
}

String question = null;
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println("Now we can set some object lock policies on
individual files:");
for (String bucketName : bucketNames) {
    for (int i = 0; i < fileNames.size(); i++){

        // No modifications to the objects in the first bucket.
        if (!bucketName.equals(bucketNames.get(0))) {
            String exampleFileName = fileNames.get(i);
            switch (i) {
```

```

        case 0 -> {
            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
            System.out.println(question);
            String ans = scanner.nextLine().trim();
            if (ans.equalsIgnoreCase("y")) {
                System.out.println("**** You have selected to put
a legal hold " + exampleFileName);

                // Set a legal hold.
                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
            }
        }
        case 1 -> {
            """"
                Would you like to add a 1 day Governance
retention period to %s in %s (y/n)?
                Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
                """".formatted(exampleFileName, bucketName);
                System.out.println(question);
                String ans2 = scanner.nextLine().trim();
                if (ans2.equalsIgnoreCase("y")) {

s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                }
            }
        }
    }
}

// Get file extension.
private static String getFileExtension(String fileName) {
    int dotIndex = fileName.lastIndexOf('.');
    if (dotIndex > 0) {
        return fileName.substring(dotIndex);
    }
    return "";
}

```

```

    public static void configurationSetup() {
        String noLockBucketName = bucketName + "-no-lock";
        String lockEnabledBucketName = bucketName + "-lock-enabled";
        String retentionAfterCreationBucketName = bucketName + "-retention-after-creation";
        bucketNames.add(noLockBucketName);
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }

        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid number.");
            }
        }

        return choiceNumber - 1;
    }
}

```

A wrapper class for S3 functions.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;

```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
    software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {
```

```
private static S3Client getClient() {
    return S3Client.builder()
        .region(Region.US_EAST_1)
        .build();
}

// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time
    zone.
    ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

    // Define a formatter for human-readable output.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Format the ZonedDateTime object to a human-readable date string.
    String humanReadableDate = formatter.format(zonedDateTime);

    // Print the formatted date string.
    System.out.println("Formatted Date: " + humanReadableDate);
    ObjectLockRetention retention = ObjectLockRetention.builder()
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .retainUntilDate(futureInstant)
        .build();

    PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .retention(retention)
        .build();

    getClient().putObjectRetention(retentionRequest);
    System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}
```

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
    }

    return null;
}

// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}
```



```

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
                })
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the
updated value.
                if (interactive) {
                    System.out.println(i + ": " + s3InfoObject.getKeyName());
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();
    }

```

```
        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
    .bucket(bucketName)
    .versioningConfiguration(versioningConfiguration)
    .build();

getClient().putBucketVersioning(versioningRequest);
DefaultRetention rention = DefaultRetention.builder()
    .days(1)
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .build();

ObjectLockRule lockRule = ObjectLockRule.builder()
    .defaultRetention(rention)
    .build();

ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
    .objectLockEnabled(ObjectLockEnabled.ENABLED)
    .rule(lockRule)
    .build();

PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(objectLockConfiguration)
    .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
    System.out.println("Added a default retention to bucket "+bucketName
+ ".");
}

// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();
```

```

        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
    .bucket(bucketName)
    .versioningConfiguration(versioningConfiguration)
    .build();

        // Enable versioning on the bucket.
getClient().putBucketVersioning(putBucketVersioningRequest);
PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .build())
    .build();

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on
"+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
    }
}

public void uploadFile(String bucketName, String objectName, String filePath)
{
    Path file = Paths.get(filePath);
    PutObjectRequest request = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectName)
        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
        .build();

    PutObjectResponse response = getClient().putObject(request, file);
    if (response != null) {
        System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
    } else {
        System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
    }
}
}

```

```
// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
}

// Delete an object from a specific bucket.
public void deleteObjectFromBucket(String bucketName, String objectKey,
boolean hasRetention, String versionId) {
    try {
        DeleteObjectRequest objectRequest;
        if (hasRetention) {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
                .bypassGovernanceRetention(true)
                .build();
        } else {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
```

```
        .build();
    }

    getClient().deleteObject(objectRequest) ;
    System.out.println("The object was successfully deleted");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

}

// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

public void deleteBucketByName(String bucketName) {
    try {
        DeleteBucketRequest request = DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();

        getClient().deleteBucket(request);
        System.out.println(bucketName +" was deleted.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
    }  
}  
  
// Get the object lock configuration details for an S3 bucket.  
public void getBucketObjectLockConfiguration(String bucketName) {  
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =  
GetObjectLockConfigurationRequest.builder()  
        .bucket(bucketName)  
        .build();  
  
    GetObjectLockConfigurationResponse response =  
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);  
    System.out.println("Bucket object lock config for "+bucketName+": ");  
    System.out.println("\tEnabled:  
"+response.getObjectLockConfiguration().getObjectLockEnabled());  
    System.out.println("\tRule: "+  
response.getObjectLockConfiguration().rule().defaultRetention());  
}  
}
```

- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Entrypoint for the scenario (index.js). This orchestrates all of the steps. Visit [GitHub](#) to see the implementation details for Scenario, ScenarioInput, ScenarioOutput, and ScenarioAction.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "../welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
  confirmSetLegalHoldFileRetention,
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
  updateRetention,
  updateRetentionAction,
} from "../setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
```

```
deploy: new scenarios.Scenario(
    "S3 Object Locking - Deploy",
    [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
    ],
    initialState,
),
demo: new scenarios.Scenario(
    "S3 Object Locking - Demo",
    [loadState, replAction(scenarios, client)],
    initialState,
),
clean: new scenarios.Scenario(
    "S3 Object Locking - Destroy",
    [
        loadState,
```



```

        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
    ],
    initialState,
),
};
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
    const objectLockingScenarios = getWorkflowStages(Scenarios);
    Scenarios.parseScenarioArgs(objectLockingScenarios, {
        name: "Amazon S3 object locking workflow",
        description:
            "Work with Amazon Simple Storage Service (Amazon S3) object locking features.",
        synopsis:
            "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-v|--verbose]",
    });
}

```

Output welcome messages to the console (welcome.steps.js).

```

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
    new scenarios.ScenarioOutput(
        "welcome",

```

```

    "Welcome to the Amazon Simple Storage Service (S3) Object Locking Feature
    Scenario. For this workflow, we will use the AWS SDK for JavaScript to create
    several S3 buckets and files to demonstrate working with S3 locking features.",
    { header: true },
  );
};

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };

```

Deploy buckets, objects, and file settings (setup.steps.js).

```

import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
  GetBucketVersioningCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**

```

```

    * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
    */

    /**
     * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
     */

    /**
     * @param {Scenarios} scenarios
     */
    const getBucketPrefix = (scenarios) =>
      new scenarios.ScenarioInput(
        "bucketPrefix",
        "Provide a prefix that will be used for bucket creation.",
        { type: "input", default: "amzn-s3-demo-bucket" },
      );

    /**
     * @param {Scenarios} scenarios
     */
    const createBuckets = (scenarios) =>
      new scenarios.ScenarioOutput(
        "createBuckets",
        (state) => `The following buckets will be created:
          ${state.bucketPrefix}-no-lock with object lock False.
          ${state.bucketPrefix}-lock-enabled with object lock True.
          ${state.bucketPrefix}-retention-after-creation with object lock False.`,
        { preformatted: true },
      );

    /**
     * @param {Scenarios} scenarios
     */
    const confirmCreateBuckets = (scenarios) =>
      new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
        type: "confirm",
      });

    /**
     * @param {Scenarios} scenarios
     * @param {S3Client} client
     */
    const createBucketsAction = (scenarios, client) =>
      new scenarios.ScenarioAction("createBucketsAction", async (state) => {

```

```

const noLockBucketName = `${state.bucketPrefix}-no-lock`;
const lockEnabledBucketName = `${state.bucketPrefix}-lock-enabled`;
const retentionBucketName = `${state.bucketPrefix}-retention-after-creation`;

try {
  await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
  await waitUntilBucketExists({ client }, { Bucket: noLockBucketName });
  await client.send(
    new CreateBucketCommand({
      Bucket: lockEnabledBucketName,
      ObjectLockEnabledForBucket: true,
    }),
  );
  await waitUntilBucketExists(
    { client },
    { Bucket: lockEnabledBucketName },
  );
  await client.send(
    new CreateBucketCommand({ Bucket: retentionBucketName }),
  );
  await waitUntilBucketExists({ client }, { Bucket: retentionBucketName });

  state.noLockBucketName = noLockBucketName;
  state.lockEnabledBucketName = lockEnabledBucketName;
  state.retentionBucketName = retentionBucketName;
} catch (caught) {
  if (
    caught instanceof BucketAlreadyExists ||
    caught instanceof BucketAlreadyOwnedByYou
  ) {
    console.error(`${caught.name}: ${caught.message}`);
    state.earlyExit = true;
  } else {
    throw caught;
  }
}
});

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",

```

```

        (state) => `The following test files will be created:
            file0.txt in ${state.bucketPrefix}-no-lock.
            file1.txt in ${state.bucketPrefix}-no-lock.
            file0.txt in ${state.bucketPrefix}-lock-enabled.
            file1.txt in ${state.bucketPrefix}-lock-enabled.
            file0.txt in ${state.bucketPrefix}-retention-after-creation.
            file1.txt in ${state.bucketPrefix}-retention-after-creation.` ,
        { preformatted: true },
    );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmPopulateBuckets",
        "Populate the buckets?",
        { type: "confirm" },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
    new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
        try {
            await client.send(
                new PutObjectCommand({
                    Bucket: state.noLockBucketName,
                    Key: "file0.txt",
                    Body: "Content",
                    ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
                }),
            );
            await client.send(
                new PutObjectCommand({
                    Bucket: state.noLockBucketName,
                    Key: "file1.txt",
                    Body: "Content",
                    ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
                }),
            );
            await client.send(

```

```
        new PutObjectCommand({
            Bucket: state.lockEnabledBucketName,
            Key: "file0.txt",
            Body: "Content",
            ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
    );
    await client.send(
        new PutObjectCommand({
            Bucket: state.lockEnabledBucketName,
            Key: "file1.txt",
            Body: "Content",
            ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
    );
    await client.send(
        new PutObjectCommand({
            Bucket: state.retentionBucketName,
            Key: "file0.txt",
            Body: "Content",
            ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
    );
    await client.send(
        new PutObjectCommand({
            Bucket: state.retentionBucketName,
            Key: "file1.txt",
            Body: "Content",
            ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
    );
} catch (caught) {
    if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while uploading object.  ${caught.name}:
${caught.message}`,
        );
    } else {
        throw caught;
    }
}
});

/**
```

```

    * @param {Scenarios} scenarios
    */
const updateRetention = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateRetention",
    (state) => `A bucket can be configured to use object locking with a default
  retention period.
  A default retention period will be configured for ${state.bucketPrefix}-
  retention-after-creation.` ,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateRetention",
    "Configure default retention period?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
    await client.send(
      new PutBucketVersioningCommand({
        Bucket: state.retentionBucketName,
        VersioningConfiguration: {
          MFADelete: MFADeleteStatus.Disabled,
          Status: BucketVersioningStatus.Enabled,
        },
      }),
    );

    const getBucketVersioning = new GetBucketVersioningCommand({
      Bucket: state.retentionBucketName,
    });

    await retry({ intervalInMs: 500, maxRetries: 10 }, async () => {
      const { Status } = await client.send(getBucketVersioning);

```

```

        if (Status !== "Enabled") {
            throw new Error("Bucket versioning is not enabled.");
        }
    });

    await client.send(
        new PutObjectLockConfigurationCommand({
            Bucket: state.retentionBucketName,
            ObjectLockConfiguration: {
                ObjectLockEnabled: "Enabled",
                Rule: {
                    DefaultRetention: {
                        Mode: "GOVERNANCE",
                        Years: 1,
                    },
                },
            },
        }),
    );
});

/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
    new scenarios.ScenarioOutput(
        "updateLockPolicy",
        (state) => `Object lock policies can also be added to existing buckets.
An object lock policy will be added to ${state.bucketPrefix}-lock-enabled.`,
        { preformatted: true },
    );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateLockPolicy = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmUpdateLockPolicy",
        "Add object lock policy?",
        { type: "confirm" },
    );

/**
 * @param {Scenarios} scenarios

```



```

    * @param {S3Client} client
    */
const updateLockPolicyAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.lockEnabledBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileEnabled",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
${state.lockEnabledBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileEnabledAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        },

```

```

        })),
    );
    console.log(
        `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
    );
},
{ skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
);

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmSetRetentionPeriodFileEnabled",
        (state) =>
            `Would you like to add a 1 day Governance retention period to file1.txt in
            ${state.lockEnabledBucketName}?
            Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
            able to delete this file or its bucket until the retention period has expired.`,
        {
            type: "confirm",
        },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
    new scenarios.ScenarioAction(
        "setRetentionPeriodFileEnabledAction",
        async (state) => {
            const retentionDate = new Date();
            retentionDate.setDate(retentionDate.getDate() + 1);
            await client.send(
                new PutObjectRetentionCommand({
                    Bucket: state.lockEnabledBucketName,
                    Key: "file1.txt",
                    Retention: {
                        Mode: ObjectLockRetentionMode.GOVERNANCE,
                        RetainUntilDate: retentionDate,
                    },
                },
            );

```

```

        })),
    );
    console.log(
        `Set retention for file1.txt in ${state.lockEnabledBucketName} until
        ${retentionDate.toISOString().split("T")[0]}.`,
    );
},
{ skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
);

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmSetLegalHoldFileRetention",
        (state) =>
            `Would you like to add a legal hold to file0.txt in
            ${state.retentionBucketName}?`,
        {
            type: "confirm",
        },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
    new scenarios.ScenarioAction(
        "setLegalHoldFileRetentionAction",
        async (state) => {
            await client.send(
                new PutObjectLegalHoldCommand({
                    Bucket: state.retentionBucketName,
                    Key: "file0.txt",
                    LegalHold: {
                        Status: ObjectLockLegalHoldStatus.ON,
                    },
                }),
            );
            console.log(
                `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
            );
        },
    );

```

```

    );
  },
  { skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
);

/**
 * @param {Scenarios} scenarios
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileRetention",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
      ${state.retentionBucketName}?
      Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
      able to delete this file or its bucket until the retention period has expired.`
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileRetentionAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
          BypassGovernanceRetention: true,
        })
      );
      console.log(

```

```

        `Set retention for file1.txt in ${state.retentionBucketName} until
        ${retentionDate.toISOString().split("T")[0]}.`,
        );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },
    );

export {
  getBucketPrefix,
  createBuckets,
  confirmCreateBuckets,
  createBucketsAction,
  populateBuckets,
  confirmPopulateBuckets,
  populateBucketsAction,
  updateRetention,
  confirmUpdateRetention,
  updateRetentionAction,
  updateLockPolicy,
  confirmUpdateLockPolicy,
  updateLockPolicyAction,
  confirmSetLegalHoldFileEnabled,
  setLegalHoldFileEnabledAction,
  confirmSetRetentionPeriodFileEnabled,
  setRetentionPeriodFileEnabledAction,
  confirmSetLegalHoldFileRetention,
  setLegalHoldFileRetentionAction,
  confirmSetRetentionPeriodFileRetention,
  setRetentionPeriodFileRetentionAction,
};

```

View and delete files in the buckets (repl.steps.js).

```

import {
  ChecksumAlgorithm,
  DeleteObjectCommand,
  GetObjectLegalHoldCommand,
  GetObjectLockConfigurationCommand,
  GetObjectRetentionCommand,
  ListObjectVersionsCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";

```

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  DELETE_FILE: 2,
  DELETE_FILE_WITH_RETENTION: 3,
  OVERWRITE_FILE: 4,
  VIEW_RETENTION_SETTINGS: 5,
  VIEW_LEGAL_HOLD_SETTINGS: 6,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    "Explore the S3 locking features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
        },
        { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
        {
          name: "View the object and bucket retention settings for a file.",
          value: choices.VIEW_RETENTION_SETTINGS,
        },
        {
          name: "View the legal hold settings for a file.",
          value: choices.VIEW_LEGAL_HOLD_SETTINGS,
        },
      ],
    },
  ),
```

```

        { name: "Finish the workflow.", value: choices.EXIT },
    ],
},
);

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
    /** @type {{bucket: string, key: string, version: string}[]} */
    const files = [];
    for (const bucket of buckets) {
        const objectsResponse = await client.send(
            new ListObjectVersionsCommand({ Bucket: bucket }),
        );
        for (const version of objectsResponse.Versions || []) {
            const { Key, VersionId } = version;
            files.push({ bucket, key: Key, version: VersionId });
        }
    }

    return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
    new scenarios.ScenarioAction(
        "replAction",
        async (state) => {
            const files = await getAllFiles(client, [
                state.noLockBucketName,
                state.lockEnabledBucketName,
                state.retentionBucketName,
            ]);

            const fileInput = new scenarios.ScenarioInput(
                "selectedFile",
                "Select a file:",
                {
                    type: "select",

```

```

        choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
                file.version
            })`,
            value: index,
        })),
    },
);

const { replChoice } = state;

switch (replChoice) {
    case choices.LIST_ALL_FILES: {
        const files = await getAllFiles(client, [
            state.noLockBucketName,
            state.lockEnabledBucketName,
            state.retentionBucketName,
        ]);
        state.replOutput = files
            .map(
                (file) =>
                    `${file.bucket}: ${file.key} (version: ${file.version})`,
            )
            .join("\n");
        break;
    }
    case choices.DELETE_FILE: {
        /** @type {number} */
        const fileToDelete = await fileInput.handle(state);
        const selectedFile = files[fileToDelete];
        try {
            await client.send(
                new DeleteObjectCommand({
                    Bucket: selectedFile.bucket,
                    Key: selectedFile.key,
                    VersionId: selectedFile.version,
                }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
        } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
        }
    }
}

```



```
        break;
    }
    case choices.DELETE_FILE_WITH_RETENTION: {
        /** @type {number} */
        const fileToDelete = await fileInput.handle(state);
        const selectedFile = files[fileToDelete];
        try {
            await client.send(
                new DeleteObjectCommand({
                    Bucket: selectedFile.bucket,
                    Key: selectedFile.key,
                    VersionId: selectedFile.version,
                    BypassGovernanceRetention: true,
                }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
        } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
        }
        break;
    }
    case choices.OVERWRITE_FILE: {
        /** @type {number} */
        const fileToOverwrite = await fileInput.handle(state);
        const selectedFile = files[fileToOverwrite];
        try {
            await client.send(
                new PutObjectCommand({
                    Bucket: selectedFile.bucket,
                    Key: selectedFile.key,
                    Body: "New content",
                    ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
                }),
            );
            state.replOutput = `Overwrote ${selectedFile.key} in
${selectedFile.bucket}.`;
        } catch (err) {
            state.replOutput = `Unable to overwrite object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
        }
        break;
    }
}
```

```

    case choices.VIEW_RETENTION_SETTINGS: {
      /** @type {number} */
      const fileToView = await fileInput.handle(state);
      const selectedFile = files[fileToView];
      try {
        const retention = await client.send(
          new GetObjectRetentionCommand({
            Bucket: selectedFile.bucket,
            Key: selectedFile.key,
            VersionId: selectedFile.version,
          }),
        );
        const bucketConfig = await client.send(
          new GetObjectLockConfigurationCommand({
            Bucket: selectedFile.bucket,
          }),
        );
        state.replOutput = `Object retention for ${selectedFile.key}
in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
      } catch (err) {
        state.replOutput = `Unable to fetch object lock retention:
'${err.message}'`;
      }
      break;
    }
    case choices.VIEW_LEGAL_HOLD_SETTINGS: {
      /** @type {number} */
      const fileToView = await fileInput.handle(state);
      const selectedFile = files[fileToView];
      try {
        const legalHold = await client.send(
          new GetObjectLegalHoldCommand({
            Bucket: selectedFile.bucket,
            Key: selectedFile.key,
            VersionId: selectedFile.version,
          }),
        );
        state.replOutput = `Object legal hold for ${selectedFile.key} in
${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
      }
    }
  }

```

```

        } catch (err) {
            state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
        }
        break;
    }
    default:
        throw new Error(`Invalid replChoice: ${replChoice}`);
    }
},
{
    whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new scenarios.ScenarioOutput(
            "REPL output",
            (state) => state.replOutput,
            { preformatted: true },
        ),
    },
},
);

export { replInput, replAction, choices };

```

Destroy all created resources (clean.steps.js).

```

import {
    DeleteObjectCommand,
    DeleteBucketCommand,
    ListObjectVersionsCommand,
    GetObjectLegalHoldCommand,
    GetObjectRetentionCommand,
    PutObjectLegalHoldCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

```

```
/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
      state;

    const buckets = [
      noLockBucketName,
      lockEnabledBucketName,
      retentionBucketName,
    ];

    for (const bucket of buckets) {
      /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
      let objectsResponse;

      try {
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Object's bucket has already been deleted.");
          continue;
        }
        throw e;
      }

      for (const version of objectsResponse.Versions || []) {
        const { Key, VersionId } = version;
      }
    }
  });
```

```
try {
  const legalHold = await client.send(
    new GetObjectLegalHoldCommand({
      Bucket: bucket,
      Key,
      VersionId,
    }),
  );

  if (legalHold.LegalHold?.Status === "ON") {
    await client.send(
      new PutObjectLegalHoldCommand({
        Bucket: bucket,
        Key,
        VersionId,
        LegalHold: {
          Status: "OFF",
        },
      }),
    );
  }
} catch (err) {
  console.log(
    `Unable to fetch legal hold for ${Key} in ${bucket}:`
    `${err.message}`,
  );
}

try {
  const retention = await client.send(
    new GetObjectRetentionCommand({
      Bucket: bucket,
      Key,
      VersionId,
    }),
  );

  if (retention.Retention?.Mode === "GOVERNANCE") {
    await client.send(
      new DeleteObjectCommand({
        Bucket: bucket,
        Key,
        VersionId,
```

```
        BypassGovernanceRetention: true,
      })),
    );
  }
} catch (err) {
  console.log(
    `Unable to fetch object lock retention for ${Key} in ${bucket}:
'${err.message}'`,
  );
}

await client.send(
  new DeleteObjectCommand({
    Bucket: bucket,
    Key,
    VersionId,
  })),
);
}

await client.send(new DeleteBucketCommand({ Bucket: bucket }));
console.log(`Delete for ${bucket} complete.`);
}
});

export { confirmCleanup, cleanupAction };
```

- For API details, see the following topics in *Amazon SDK for JavaScript API Reference*.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Make Amazon S3 conditional requests using an Amazon SDK

The following code examples show how to add preconditions to Amazon S3 requests.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Run an interactive scenario demonstrating Amazon S3 conditional request features.

```
using Amazon.S3;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ConditionalRequestsScenario;

public static class S3ConditionalRequestsScenario
{
    /*
        Before running this .NET code example, set up your development environment,
        including your credentials.

        This example demonstrates the use of conditional requests for S3 operations.
        You can use conditional requests to add preconditions to S3 read requests to
        return or copy
        an object based on its Entity tag (ETag), or last modified date.
        You can use a conditional write requests to prevent overwrites by ensuring
        there is no existing object with the same key.
    */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
    public static IConfiguration _configuration = null!;
```

```
public static string _resourcePrefix = null!;  
public static string _sourceBucketName = null!;  
public static string _destinationBucketName = null!;  
public static string _sampleObjectKey = null!;  
public static string _sampleObjectEtag = null!;  
public static bool _interactive = true;  
  
public static async Task Main(string[] args)  
{  
    // Set up dependency injection for the Amazon service.  
    using var host = Host.CreateDefaultBuilder(args)  
        .ConfigureLogging(logging =>  
            logging.AddFilter("System", LogLevel.Debug)  
                .AddFilter<DebugLoggerProvider>("Microsoft",  
LogLevel.Information)  
                .AddFilter<ConsoleLoggerProvider>("Microsoft",  
LogLevel.Trace))  
        .ConfigureServices((_, services) =>  
            services.AddAWSService<IAmazonS3>()  
                .AddTransient<S3ActionsWrapper>()  
        )  
        .Build();  
  
    _configuration = new ConfigurationBuilder()  
        .SetBasePath(Directory.GetCurrentDirectory())  
        .AddJsonFile("settings.json") // Load settings from .json file.  
        .AddJsonFile("settings.local.json",  
            true) // Optionally, load local settings.  
        .Build();  
  
    ServicesSetup(host);  
  
    try  
    {  
        Console.WriteLine(new string('-', 80));  
        Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)  
Conditional Requests Feature Scenario.");  
        Console.WriteLine(new string('-', 80));  
        ConfigurationSetup();  
        _sampleObjectEtag = await Setup(_sourceBucketName,  
_destinationBucketName, _sampleObjectKey);
```



```

        await DisplayDemoChoices(_sourceBucketName, _destinationBucketName,
            _sampleObjectKey, _sampleObjectEtag, 0);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cleaning up resources.");
        Console.WriteLine(new string('-', 80));
        await Cleanup(true);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Amazon S3 Conditional Requests Feature Scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"There was a problem: {ex.Message}");
        await CleanupScenario(_sourceBucketName, _destinationBucketName);
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Populate the services for use within the console application.
/// </summary>
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
}

/// <summary>
/// Any setup operations needed.
/// </summary>
public static void ConfigurationSetup()
{
    _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

    _sourceBucketName = _resourcePrefix + "-source";
    _destinationBucketName = _resourcePrefix + "-dest";
    _sampleObjectKey = _resourcePrefix + "-sample-object.txt";
}

/// <summary>

```

```

    /// Sets up the scenario by creating a source and destination bucket, and
    uploading a test file to the source bucket.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    /// <param name="objectKey">The name of the test file to add to the source
    bucket.</param>
    /// <returns>The ETag of the uploaded test file.</returns>
    public static async Task<string> Setup(string sourceBucket, string
    destBucket, string objectKey)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create
            several S3\n" +
            "buckets and files to demonstrate working with S3 conditional
            requests.\n" +
            "This example demonstrates the use of conditional requests for S3
            operations.\r\n" +
            "You can use conditional requests to add preconditions to S3 read
            requests to return or copy\r\n" +
            "an object based on its Entity tag (ETag), or last modified date. \r
            \n" +
            "You can use a conditional write requests to prevent overwrites by
            ensuring \r\n" +
            "there is no existing object with the same key. \r\n\r\n" +
            "This example will allow you to perform conditional reads\r\n" +
            "and writes that will succeed or fail based on your selected options.
            \r\n\r\n" +
            "Sample buckets and a sample object will be created as part of the
            example.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (_interactive)
            Console.ReadLine();

        await _s3ActionsWrapper.CreateBucketWithName(sourceBucket);
        await _s3ActionsWrapper.CreateBucketWithName(destBucket);

        var eTag = await _s3ActionsWrapper.PutObjectConditional(objectKey,
            sourceBucket,
            "Test file content.");

        return eTag;
    }

```

```
}

/// <summary>
/// Cleans up the scenario by deleting the source and destination buckets.
/// </summary>
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>
public static async Task CleanupScenario(string sourceBucket, string
destBucket)
{
    await _s3ActionsWrapper.CleanupBucketByName(sourceBucket);
    await _s3ActionsWrapper.CleanupBucketByName(destBucket);
}

/// <summary>
/// Displays a list of the objects in the test buckets.
/// </summary>
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>
public static async Task DisplayBuckets(string sourceBucket, string
destBucket)
{
    await _s3ActionsWrapper.ListBucketContentsByName(sourceBucket);
    await _s3ActionsWrapper.ListBucketContentsByName(destBucket);
}

/// <summary>
/// Displays the menu of conditional request options for the user.
/// </summary>
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>
/// <param name="objectKey">The key of the test object in the source
bucket.</param>
/// <param name="etag">The ETag of the test object in the source bucket.</
param>
public static async Task DisplayDemoChoices(string sourceBucket, string
destBucket, string objectKey, string etag, int defaultChoice)
{
    var actions = new[]
    {
        "Print a list of bucket items.",
        "Perform a conditional read.",
        "Perform a conditional copy.",
        "Perform a conditional write.",
    }
}
```

```

        "Clean up and exit."
    };

    var conditions = new[]
    {
        "If-Match: using the object's ETag. This condition should succeed.",
        "If-None-Match: using the object's ETag. This condition should
fail.",
        "If-Modified-Since: using yesterday's date. This condition should
succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should
fail."
    };

    var conditionTypes = new[]
    {
        S3ConditionType.IfMatch,
        S3ConditionType.IfNoneMatch,
        S3ConditionType.IfModifiedSince,
        S3ConditionType.IfUnmodifiedSince,
    };

    var yesterdayDate = DateTime.UtcNow.AddDays(-1);

    int choice;
    while ((choice = GetChoiceResponse("\nExplore the S3 conditional request
features by selecting one of the following choices:", actions, defaultChoice)) != 4)
    {
        switch (choice)
        {
            case 0:
                Console.WriteLine("Listing the objects and buckets.");
                await DisplayBuckets(sourceBucket, destBucket);
                break;
            case 1:
                int conditionTypeIndex = GetChoiceResponse("Perform a
conditional read:", conditions, 1);
                if (conditionTypeIndex == 0 || conditionTypeIndex == 1)
                {
                    await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], null, _sampleObjectEtag);
                }
                else if (conditionTypeIndex == 2 || conditionTypeIndex == 3)

```

```

        {
            await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], yesterdayDate);
        }
        break;
    case 2:
        int copyConditionTypeIndex = GetChoiceResponse("Perform a
conditional copy:", conditions, 1);
        string destKey = GetStringResponse("Enter an object key:",
"sampleObjectKey");
        if (copyConditionTypeIndex == 0 || copyConditionTypeIndex ==
1)
        {
            await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex], null,
etag);
        }
        else if (copyConditionTypeIndex == 2 ||
copyConditionTypeIndex == 3)
        {
            await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex],
yesterdayDate);
        }
        break;
    case 3:
        Console.WriteLine("Perform a conditional write using
IfNoneMatch condition on the object key.");
        Console.WriteLine("If the key is a duplicate, the write will
fail.");
        string newObjectKey = GetStringResponse("Enter an object
key:", "newObjectKey");
        await _s3ActionsWrapper.PutObjectConditional(newObjectKey,
sourceBucket, "Conditional write example data.");
        break;
    }

    if (!_interactive)
    {
        break;
    }
}

Console.WriteLine("Proceeding to cleanup.");

```

```

    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
        {
            await _s3ActionsWrapper.CleanUpBucketByName(_sourceBucketName);
            await _s3ActionsWrapper.CleanUpBucketByName(_destinationBucketName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

```

```
}

/// <summary>
/// Helper method to get a choice response from the user.
/// </summary>
/// <param name="question">The question string to print on the console.</param>
/// <param name="choices">The choices to print on the console.</param>
/// <returns>The index of the selected choice</returns>
private static int GetChoiceResponse(string? question, string[] choices, int defaultChoice)
{
    if (question != null)
    {
        Console.WriteLine(question);

        for (int i = 0; i < choices.Length; i++)
        {
            Console.WriteLine($"{i + 1}. {choices[i]}");
        }
    }

    if (!_interactive)
        return defaultChoice;

    var choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > choices.Length)
    {
        var choice = Console.ReadLine();
        Int32.TryParse(choice, out choiceNumber);
    }

    return choiceNumber - 1;
}

/// <summary>
/// Get a string response from the user.
/// </summary>
/// <param name="question">The question to print.</param>
/// <param name="defaultAnswer">A default answer to use when not interactive.</param>
/// <returns>The string response.</returns>
public static string GetStringResponse(string? question, string defaultAnswer)
```

```
{
    string? answer = "";
    if (_interactive)
    {
        do
        {
            Console.WriteLine(question);
            answer = Console.ReadLine();
        } while (string.IsNullOrEmpty(answer));
    }
    else
    {
        answer = defaultAnswer;
    }

    return answer;
}
}
```

A wrapper class for S3 functions.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Logging;

namespace S3ConditionalRequestsScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;
    private readonly ILogger<S3ActionsWrapper> _logger;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    /// <param name="logger">The class logger.</param>
}
```



```

public S3ActionsWrapper(IAmazonS3 amazonS3, ILogger<S3ActionsWrapper> logger)
{
    _amazonS3 = amazonS3;
    _logger = logger;
}

/// <summary>
/// Retrieves an object from Amazon S3 with a conditional request.
/// </summary>
/// <param name="objectKey">The key of the object to retrieve.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="conditionType">The type of condition: 'IfMatch',
'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional read is successful, False otherwise.</
returns>
public async Task<bool> GetObjectConditional(string objectKey, string
sourceBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
{
    try
    {
        var getObjectRequest = new GetObjectRequest
        {
            BucketName = sourceBucket,
            Key = objectKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                getObjectRequest.EtagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                getObjectRequest.EtagToNotMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfModifiedSince:
                getObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
        }
    }
}

```

```

        case S3ConditionType.IfUnmodifiedSince:
            getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
            break;
        default:
            throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
    }

    var response = await _amazonS3.GetObjectAsync(getObjectRequest);
    var sampleBytes = new byte[20];
    await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
    _logger.LogInformation($"Conditional read
successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
    return true;
}
catch (AmazonS3Exception e)
{
    if (e.ErrorCode == "PreconditionFailed")
    {
        _logger.LogError("Conditional read failed: Precondition failed");
    }
    else if (e.ErrorCode == "NotModified")
    {
        _logger.LogError("Conditional read failed: Object not modified");
    }
    else
    {
        _logger.LogError($"Unexpected error: {e.ErrorCode}");
        throw;
    }
    return false;
}
}

/// <summary>
/// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
/// </summary>
/// <param name="objectKey">The key of the object to upload.</param>
/// <param name="bucket">The source bucket of the object.</param>
/// <param name="content">The content to upload as a string.</param>

```

```

    /// <returns>The ETag if the conditional write is successful, empty
    otherwise.</returns>
    public async Task<string> PutObjectConditional(string objectKey, string
    bucket, string content)
    {
        try
        {
            var putObjectRequest = new PutObjectRequest
            {
                BucketName = bucket,
                Key = objectKey,
                ContentBody = content,
                IfNoneMatch = "*"
            };

            var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
            _logger.LogInformation($"Conditional write successful for key
    {objectKey} in bucket {bucket}.");
            return putResult.ETag;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional write failed: Precondition
    failed");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return string.Empty;
        }
    }

    /// <summary>
    /// Copies an object from one Amazon S3 bucket to another with a conditional
    request.
    /// </summary>
    /// <param name="sourceKey">The key of the source object to copy.</param>
    /// <param name="destKey">The key of the destination object.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="destBucket">The destination bucket of the object.</param>

```

```

    /// <param name="conditionType">The type of condition to apply, e.g.
    'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
    'CopySourceIfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
    dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
    etags.</param>
    /// <returns>True if the conditional copy is successful, False otherwise.</
returns>
    public async Task<bool> CopyObjectConditional(string sourceKey, string
destKey, string sourceBucket, string destBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
    {
        try
        {
            var copyObjectRequest = new CopyObjectRequest
            {
                DestinationBucket = destBucket,
                DestinationKey = destKey,
                SourceBucket = sourceBucket,
                SourceKey = sourceKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    copyObjectRequest.ETagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
            }
        }
    }

```

```

        await _amazonS3.CopyObjectAsync(copyObjectRequest);
        _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
        return true;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional copy failed: Precondition failed");
        }
        else if (e.ErrorCode == "304")
        {
            _logger.LogError("Conditional copy failed: Object not modified");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

/// <summary>
/// Create a new Amazon S3 bucket with a specified name and check that the
bucket is ready.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateBucketWithName(string bucketName)
{
    Console.WriteLine($"\\tCreating bucket {bucketName}.");
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true
        };

        await _amazonS3.PutBucketAsync(request);
        var bucketReady = false;

```

```

        var retries = 5;
        while (!bucketReady && retries > 0)
        {
            Thread.Sleep(5000);
            bucketReady = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_amazonS3, bucketName);
            retries--;
        }

        return bucketReady;
    }
    catch (BucketAlreadyExistsException ex)
    {
        Console.WriteLine($"Bucket already exists: '{ex.Message}'");
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Cleans up objects and deletes the bucket by name.
/// </summary>
/// <param name="bucketName">The name of the bucket.</param>
/// <returns>Async task.</returns>
public async Task CleanupBucketByName(string bucketName)
{
    try
    {
        var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
        foreach (var obj in listObjectsResponse.S3Objects)
        {
            await _amazonS3.DeleteObjectAsync(new DeleteObjectRequest
{ BucketName = bucketName, Key = obj.Key });
        }
        await _amazonS3.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        Console.WriteLine($"Cleaned up bucket: {bucketName}.");
    }
    catch (AmazonS3Exception e)

```

```
        {
            if (e.ErrorCode == "NoSuchBucket")
            {
                Console.WriteLine($"Bucket {bucketName} does not exist, skipping
cleanup.");
            }
            else
            {
                Console.WriteLine($"Error deleting bucket: {e.ErrorCode}");
                throw;
            }
        }
    }

    /// <summary>
    /// List the contents of the bucket with their ETag.
    /// </summary>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>Async task.</returns>
    public async Task<List<S3Object>> ListBucketContentsByName(string bucketName)
    {
        var results = new List<S3Object>();
        try
        {
            Console.WriteLine($"\\t Items in bucket {bucketName}");
            var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
            if (listObjectsResponse.S3Objects.Count == 0)
            {
                Console.WriteLine("\\t\\tNo objects found.");
            }
            else
            {
                foreach (var obj in listObjectsResponse.S3Objects)
                {
                    Console.WriteLine($"\\t\\t object: {obj.Key} ETag {obj.ETag}");
                }
            }
            results = listObjectsResponse.S3Objects;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "NoSuchBucket")
```

```
        {
            _logger.LogError($"Bucket {bucketName} does not exist.");
        }
        else
        {
            _logger.LogError($"Error listing bucket and objects:
{e.ErrorCode}");
            throw;
        }
    }

    return results;
}

/// <summary>
/// Delete an object from a specific bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey)
{
    try
    {
        var request = new DeleteObjectRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };
        await _amazonS3.DeleteObjectAsync(request);
        Console.WriteLine($"Deleted {objectKey} in {bucketName}.");
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
        return false;
    }
}

/// <summary>
```



```
/// Delete a specific bucket by deleting the objects and then the bucket
itself.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <param name="versionId">Optional versionId.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CleanUpBucketByName(string bucketName)
{
    try
    {
        var allFiles = await ListBucketContentsByName(bucketName);

        foreach (var fileInfo in allFiles)
        {
            await DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key);
        }

        var request = new DeleteBucketRequest() { BucketName = bucketName, };
        var response = await _amazonS3.DeleteBucketAsync(request);
        Console.WriteLine($"\\tDelete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
            ex.Message);
        return false;
    }
}
}
```

- For API details, see the following topics in *Amazon SDK for .NET API Reference*.
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Entrypoint for the workflow (index.js). This orchestrates all of the steps. Visit GitHub to see the implementation details for Scenario, ScenarioInput, ScenarioOutput, and ScenarioAction.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "../welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
} from "../setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Conditional Requests - Deploy",
      [
```

```

        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        saveState,
    ],
    initialState,
),
demo: new scenarios.Scenario(
    "S3 Conditional Requests - Demo",
    [loadState, welcome(scenarios), replAction(scenarios, client)],
    initialState,
),
clean: new scenarios.Scenario(
    "S3 Conditional Requests - Destroy",
    [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
    ],
    initialState,
),
};
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
    const objectLockingScenarios = getWorkflowStages(Scenarios);
    Scenarios.parseScenarioArgs(objectLockingScenarios, {
        name: "Amazon S3 object locking workflow",
    });
}

```

```

    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking
      features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-v|--verbose]",
    });
  }

```

Output welcome messages to the console (welcome.steps.js).

```

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "This example demonstrates the use of conditional requests for S3
    operations." +
    " You can use conditional requests to add preconditions to S3 read requests
    to return " +
    "or copy an object based on its Entity tag (ETag), or last modified
    date.You can use " +
    "a conditional write requests to prevent overwrites by ensuring there is no
    existing " +
    "object with the same key.\n" +
    "This example will enable you to perform conditional reads and writes that
    will succeed " +
    "or fail based on your selected options.\n" +
    "Sample buckets and a sample object will be created as part of the example.
    \n" +
    "Some steps require a key name prefix to be defined by the user. Before you
    begin, you can " +
    "optionally edit this prefix in ./object_name.json. If you do so, please
    reload the scenario before you begin.",
    { header: true },
  );

```

```
/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

Deploy buckets and objects (setup.steps.js).

```
import {
  ChecksumAlgorithm,
  CreateBucketCommand,
  PutObjectCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );
/**
```

```
* @param {Scenarios} scenarios
*/
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
      ${state.bucketPrefix}-source-bucket.
      ${state.bucketPrefix}-destination-bucket.` ,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const sourceBucketName = `${state.bucketPrefix}-source-bucket`;
    const destinationBucketName = `${state.bucketPrefix}-destination-bucket`;

    try {
      await client.send(
        new CreateBucketCommand({
          Bucket: sourceBucketName,
        }),
      );
      await waitUntilBucketExists({ client }, { Bucket: sourceBucketName });
      await client.send(
        new CreateBucketCommand({
          Bucket: destinationBucketName,
        }),
      );
      await waitUntilBucketExists(
        { client },
        { Bucket: destinationBucketName },
      );
    }
  });
```

```

        state.sourceBucketName = sourceBucketName;
        state.destinationBucketName = destinationBucketName;
    } catch (caught) {
        if (
            caught instanceof BucketAlreadyExists ||
            caught instanceof BucketAlreadyOwnedByYou
        ) {
            console.error(`${caught.name}: ${caught.message}`);
            state.earlyExit = true;
        } else {
            throw caught;
        }
    }
});

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
    new scenarios.ScenarioOutput(
        "populateBuckets",
        (state) => `The following test files will be created:
            file01.txt in ${state.bucketPrefix}-source-bucket.` ,
        { preformatted: true },
    );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmPopulateBuckets",
        "Populate the buckets?",
        { type: "confirm" },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
    new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
        try {

```

```
    await client.send(
      new PutObjectCommand({
        Bucket: state.sourceBucketName,
        Key: "file01.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
  } catch (caught) {
    if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while uploading object.  ${caught.name}:
${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
});

export {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
};
```

Get, copy, and put objects using S3 conditional requests (repl.steps.js).

```
import path from "node:path";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

import {
  ListObjectVersionsCommand,
  GetObjectCommand,
  CopyObjectCommand,
  PutObjectCommand,
```



```
} from "@aws-sdk/client-s3";
import data from "./object_name.json" assert { type: "json" };
import { readFile } from "node:fs/promises";
import {
  ScenarioInput,
  Scenario,
  ScenarioAction,
  ScenarioOutput,
} from "../../libs/scenario/index.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  CONDITIONAL_READ: 2,
  CONDITIONAL_COPY: 3,
  CONDITIONAL_WRITE: 4,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new ScenarioInput(
    "replChoice",
    "Explore the S3 conditional request features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "Print list of bucket items.", value: choices.LIST_ALL_FILES },
        {
          name: "Perform a conditional read.",
          value: choices.CONDITIONAL_READ,
        },
      ],
    }
  )
```

```

        name: "Perform a conditional copy. These examples use the key name
        prefix defined in ./object_name.json.",
        value: choices.CONDITIONAL_COPY,
    },
    {
        name: "Perform a conditional write. This example use the sample file ./
        text02.txt.",
        value: choices.CONDITIONAL_WRITE,
    },
    { name: "Finish the workflow.", value: choices.EXIT },
],
},
);

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
    /** @type {{bucket: string, key: string, version: string}[]} */
    const files = [];
    for (const bucket of buckets) {
        const objectsResponse = await client.send(
            new ListObjectVersionsCommand({ Bucket: bucket })),
        );
        for (const version of objectsResponse.Versions || []) {
            const { Key } = version;
            files.push({ bucket, key: Key });
        }
    }
    return files;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 * @param {string} key
 */
const getEtag = async (client, bucket, key) => {
    const objectsResponse = await client.send(
        new GetObjectCommand({
            Bucket: bucket,
            Key: key,
        })),
    );

```

```

    );
    return objectsResponse.ETag;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
export const replAction = (scenarios, client) =>
  new ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.sourceBucketName,
        state.destinationBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file to use:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (Etag: ${
              file.version
            })`,
            value: index,
          })),
        },
      );

      const condReadOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional read action would you like to take?",
        {
          type: "select",
          choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
fail.",

```

```

        "If-Modified-Since: using yesterday's date. This condition should
        succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should
        fail.",
    ],
    },
);
const condCopyOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional copy action would you like to take?",
    {
        type: "select",
        choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
            fail.",
            "If-Modified-Since: using yesterday's date. This condition should
            succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
            fail.",
        ],
    },
);
const condWriteOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional write action would you like to take?",
    {
        type: "select",
        choices: [
            "IfNoneMatch condition on the object key: If the key is a duplicate,
            the write will fail.",
        ],
    },
);

const { replChoice } = state;

switch (replChoice) {
    case choices.LIST_ALL_FILES: {
        const files = await getAllFiles(client, [
            state.sourceBucketName,
            state.destinationBucketName,
        ]);
        state.replOutput = files

```

```

        .map(
            (file) => `Items in bucket ${file.bucket}: object: ${file.key} `,
        )
        .join("\n");
    break;
}
case choices.CONDITIONAL_READ:
{
    const selectedCondRead = await condReadOptions.handle(state);
    if (
        selectedCondRead ===
        "If-Match: using the object's ETag. This condition should succeed."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const ETag = await getEtag(client, bucket, key);

        try {
            await client.send(
                new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfMatch: ETag,
                }),
            );
            state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because ETag provided matches the object's ETag.`;
        } catch (err) {
            state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
        }
        break;
    }
    if (
        selectedCondRead ===
        "If-None-Match: using the object's ETag. This condition should
fail."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const ETag = await getEtag(client, bucket, key);

        try {
            await client.send(

```

```

        new GetObjectCommand({
            Bucket: bucket,
            Key: key,
            IfNoneMatch: ETag,
        }),
    );
    state.replOutput = `${key} in ${state.sourceBucketName} was
returned.`;
    } catch (err) {
        state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
    }
    break;
}
if (
    selectedCondRead ===
    "If-Modified-Since: using yesterday's date. This condition should
succeed."
) {
    const date = new Date();
    date.setDate(date.getDate() - 1);

    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    try {
        await client.send(
            new GetObjectCommand({
                Bucket: bucket,
                Key: key,
                IfModifiedSince: date,
            }),
        );
        state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because it has been created or modified in the last 24 hours.`;
    } catch (err) {
        state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
    }
    break;
}
if (
    selectedCondRead ===
    "If-Unmodified-Since: using yesterday's date. This condition should
fail."

```

```

    ) {
      const bucket = state.sourceBucketName;
      const key = "file01.txt";

      const date = new Date();
      date.setDate(date.getDate() - 1);
      try {
        await client.send(
          new GetObjectCommand({
            Bucket: bucket,
            Key: key,
            IfUnmodifiedSince: date,
          }),
        );
        state.replOutput = `${key} in ${state.sourceBucketName} was
read.`;
      } catch (err) {
        state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
      }
      break;
    }
  }
  break;
case choices.CONDITIONAL_COPY: {
  const selectedCondCopy = await condCopyOptions.handle(state);
  if (
    selectedCondCopy ===
    "If-Match: using the object's ETag. This condition should succeed."
  ) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    const ETag = await getEtag(client, bucket, key);

    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
    in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;
    try {
      await client.send(
        new CopyObjectCommand({
          CopySource: copySource,
          Bucket: state.destinationBucketName,

```

```

        Key: copiedKey,
        CopySourceIfMatch: ETag,
    )),
    );
    state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because ETag provided matches the object's
ETag.`;
    } catch (err) {
        state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName}: ${err.message}`;
    }
    break;
}
if (
    selectedCondCopy ===
    "If-None-Match: using the object's ETag. This condition should fail."
) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    const ETag = await getEtag(client, bucket, key);
    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
    in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;

    try {
        await client.send(
            new CopyObjectCommand({
                CopySource: copySource,
                Bucket: state.destinationBucketName,
                Key: copiedKey,
                CopySourceIfNoneMatch: ETag,
            )),
        );
        state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName}`;
    } catch (err) {
        state.replOutput = `Unable to copy object as ${key} as as
${copiedKey} to bucket ${state.destinationBucketName}: ${err.message}`;
    }
    break;
}
if (

```



```

        selectedCondCopy ===
        "If-Modified-Since: using yesterday's date. This condition should
succeed."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const copySource = `${bucket}/${key}`;
        // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
        const name = data.name;
        const copiedKey = `${name}${key}`;

        const date = new Date();
        date.setDate(date.getDate() - 1);

        try {
            await client.send(
                new CopyObjectCommand({
                    CopySource: copySource,
                    Bucket: state.destinationBucketName,
                    Key: copiedKey,
                    CopySourceIfModifiedSince: date,
                }),
            );
            state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because it has been created or modified in the
last 24 hours.`;
        } catch (err) {
            state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName} : ${err.message}`;
        }
        break;
    }
    if (
        selectedCondCopy ===
        "If-Unmodified-Since: using yesterday's date. This condition should
fail."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const copySource = `${bucket}/${key}`;
        // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
        const name = data.name;

```

```

const copiedKey = `${name}${key}`;

const date = new Date();
date.setDate(date.getDate() - 1);

try {
  await client.send(
    new CopyObjectCommand({
      CopySource: copySource,
      Bucket: state.destinationBucketName,
      Key: copiedKey,
      CopySourceIfUnmodifiedSince: date,
    }),
  );
  state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName} because it has not been created or modified in
the last 24 hours.`;
} catch (err) {
  state.replOutput = `Unable to copy object ${key} to bucket
${state.destinationBucketName}: ${err.message}`;
}
break;
}
case choices.CONDITIONAL_WRITE:
{
  const selectedCondWrite = await condWriteOptions.handle(state);
  if (
    selectedCondWrite ===
    "IfNoneMatch condition on the object key: If the key is a
duplicate, the write will fail."
  ) {
    // Optionally edit the default key name prefix of the copied object
    in ./object_name.json.
    const key = "text02.txt";
    const __filename = fileURLToPath(import.meta.url);
    const __dirname = dirname(__filename);
    const filePath = path.join(__dirname, "text02.txt");
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: `${state.destinationBucketName}`,
          Key: `${key}`,
          Body: await readFile(filePath),

```

```

                IfNoneMatch: "*",
            )),
        );
        state.replOutput = `${key} uploaded to bucket
${state.destinationBucketName} because the key is not a duplicate.`;
        } catch (err) {
            state.replOutput = `Unable to upload object to bucket
${state.destinationBucketName}:${err.message}`;
        }
        break;
    }
}
break;

default:
    throw new Error(`Invalid replChoice: ${replChoice}`);
}
},
{
    whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new ScenarioOutput("REPL output", (state) => state.replOutput, {
            preformatted: true,
        }),
    },
},
);

export { replInput, choices };

```

Destroy all created resources (clean.steps.js).

```

import {
    DeleteObjectCommand,
    DeleteBucketCommand,
    ListObjectVersionsCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

```

```
/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { sourceBucketName, destinationBucketName } = state;
    const buckets = [sourceBucketName, destinationBucketName].filter((b) => b);

    for (const bucket of buckets) {
      try {
        let objectsResponse;
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
        for (const version of objectsResponse.Versions || []) {
          const { Key, VersionId } = version;
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
              }),
            );
          } catch (err) {
            console.log(`An error occurred: ${err.message} `);
          }
        }
      }
    }
  })
```

```
    } catch (e) {
      if (e instanceof Error && e.name === "NoSuchBucket") {
        console.log("Objects and buckets have already been deleted.");
        continue;
      }
      throw e;
    }

    await client.send(new DeleteBucketCommand({ Bucket: bucket }));
    console.log(`Delete for ${bucket} complete.`);
  }
});

export { confirmCleanup, cleanupAction };
```

- For API details, see the following topics in *Amazon SDK for JavaScript API Reference*.
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Run an interactive scenario demonstrating Amazon S3 conditional requests.

```
"""
Purpose

Shows how to use AWS SDK for Python (Boto3) to get started using conditional
requests for
Amazon Simple Storage Service (Amazon S3).
```

```
"""

import logging
import random
import sys
import datetime

import boto3
from botocore.exceptions import ClientError

from s3_conditional_requests import S3ConditionalRequests

# Add relative path to include demo_tools in this code example without need for
# setup.
sys.path.append("../..")
import demo_tools.question as q # noqa

# Constants
FILE_CONTENT = "This is a test file for S3 conditional requests."
RANDOM_SUFFIX = str(random.randint(100, 999))

logger = logging.getLogger(__name__)

class ConditionalRequestsScenario:
    """Runs a scenario that shows how to use S3 Conditional Requests."""

    def __init__(self, conditional_requests, s3_client):
        """
        :param conditional_requests: An object that wraps S3 conditional request
        actions.
        :param s3_client: A Boto3 S3 client for setup and cleanup operations.
        """
        self.conditional_requests = conditional_requests
        self.s3_client = s3_client

    def setup_scenario(self, source_bucket: str, dest_bucket: str, object_key:
str):
        """
        Sets up the scenario by creating a source and destination bucket.
        Prompts the user to provide a bucket name prefix.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.

```

```

        :param object_key: The name of a test file to add to the source bucket.
        """

        # Create the buckets.
        try:
            self.s3_client.create_bucket(Bucket=source_bucket)
            self.s3_client.create_bucket(Bucket=dest_bucket)
            print(
                f"Created source bucket: {source_bucket} and destination bucket:
{dest_bucket}"
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            logger.error(f"Error creating buckets: {error_code}")
            raise

        # Upload test file into the source bucket.
        try:
            print(f"Uploading file {object_key} to bucket {source_bucket}")
            response = self.s3_client.put_object(
                Bucket=source_bucket, Key=object_key, Body=FILE_CONTENT
            )
            object_etag = response["ETag"]
            return object_etag

        except Exception as e:
            logger.error(
                f"Failed to upload file {object_key} to bucket {source_bucket}:
{e}"
            )

    def cleanup_scenario(self, source_bucket: str, dest_bucket: str):
        """
        Cleans up the scenario by deleting the source and destination buckets.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        """
        self.cleanup_bucket(source_bucket)
        self.cleanup_bucket(dest_bucket)

    def cleanup_bucket(self, bucket_name: str):
        """

```

```

        Cleans up the bucket by deleting all objects and then the bucket itself.

        :param bucket_name: The name of the bucket.
        """
        try:
            # Get list of all objects in the bucket.
            list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
            objs = list_response.get("Contents", [])
            for obj in objs:
                key = obj["Key"]
                self.s3_client.delete_object(Bucket=bucket_name, Key=key)
            self.s3_client.delete_bucket(Bucket=bucket_name)
            print(f"Cleaned up bucket: {bucket_name}.")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "NoSuchBucket":
                logger.info(f"Bucket {bucket_name} does not exist, skipping
cleanup.")
            else:
                logger.error(f"Error deleting bucket: {error_code}")
                raise

    def display_buckets(self, source_bucket: str, dest_bucket: str):
        """
        Display a list of the objects in the test buckets.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        """
        self.list_bucket_contents(source_bucket)
        self.list_bucket_contents(dest_bucket)

    def list_bucket_contents(self, bucket_name):
        """
        Display a list of the objects in the bucket.

        :param bucket_name: The name of the bucket.
        """
        try:
            # Get list of all objects in the bucket.
            print(f"\t Items in bucket {bucket_name}")
            list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
            objs = list_response.get("Contents", [])

```



```

        if not objs:
            print("\t\tNo objects found.")
        for obj in objs:
            key = obj["Key"]
            print(f"\t\t object: {key} ETag {obj['ETag']}")
        return objs
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "NoSuchBucket":
            logger.info(f"Bucket {bucket_name} does not exist.")
        else:
            logger.error(f"Error listing bucket and objects: {error_code}")
            raise

def display_menu(
    self, source_bucket: str, dest_bucket: str, object_key: str, etag: str
):
    """
    Displays the menu of conditional request options for the user.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    :param object_key: The key of the test object in the source bucket.
    :param etag: The etag of the test object in the source bucket.
    """

    actions = [
        "Print list of bucket items.",
        "Perform a conditional read.",
        "Perform a conditional copy.",
        "Perform a conditional write.",
        "Clean up and exit.",
    ]

    conditions = [
        "If-Match: using the object's ETag. This condition should succeed.",
        "If-None-Match: using the object's ETag. This condition should",
fail.",
        "If-Modified-Since: using yesterday's date. This condition should",
succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should",
fail.",
    ]

```

```

        condition_types = [
            "IfMatch",
            "IfNoneMatch",
            "IfModifiedSince",
            "IfUnmodifiedSince",
        ]
        copy_condition_types = [
            "CopySourceIfMatch",
            "CopySourceIfNoneMatch",
            "CopySourceIfModifiedSince",
            "CopySourceIfUnmodifiedSince",
        ]

        yesterday_date = datetime.datetime.utcnow() - datetime.timedelta(days=1)

        choice = 0
        while choice != 4:
            print("-" * 88)
            print("Choose an action to explore some example conditional
requests.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                print("Listing the objects and buckets.")
                self.display_buckets(source_bucket, dest_bucket)
            elif choice == 1:
                print("Perform a conditional read.")
                condition_type = q.choose("Enter the condition type : ",
conditions)
                if condition_type == 0 or condition_type == 1:
                    self.conditional_requests.get_object_conditional(
                        object_key, source_bucket,
condition_types[condition_type], etag
                    )
                elif condition_type == 2 or condition_type == 3:
                    self.conditional_requests.get_object_conditional(
                        object_key,
                        source_bucket,
                        condition_types[condition_type],
                        yesterday_date,
                    )
            elif choice == 2:
                print("Perform a conditional copy.")

```

```

        condition_type = q.choose("Enter the condition type : ",
conditions)
        dest_key = q.ask("Enter an object key: ", q.non_empty)
        if condition_type == 0 or condition_type == 1:
            self.conditional_requests.copy_object_conditional(
                object_key,
                dest_key,
                source_bucket,
                dest_bucket,
                copy_condition_types[condition_type],
                etag,
            )
        elif condition_type == 2 or condition_type == 3:
            self.conditional_requests.copy_object_conditional(
                object_key,
                dest_key,
                copy_condition_types[condition_type],
                yesterday_date,
            )
        elif choice == 3:
            print(
                "Perform a conditional write using IfNoneMatch condition on
the object key."
            )
            print("If the key is a duplicate, the write will fail.")
            object_key = q.ask("Enter an object key: ", q.non_empty)
            self.conditional_requests.put_object_conditional(
                object_key, source_bucket, b"Conditional write example data."
            )
        elif choice == 4:
            print("Proceeding to cleanup.")

def run_scenario(self):
    """
    Runs the interactive scenario.
    """
    print("-" * 88)
    print("Welcome to the Amazon S3 conditional requests example.")
    print("-" * 88)

    print(
        f"""\

```

This example demonstrates the use of conditional requests for S3 operations.

You can use conditional requests to add preconditions to S3 read requests to return or copy

an object based on its Entity tag (ETag), or last modified date.

You can use a conditional write requests to prevent overwrites by ensuring

there is no existing object with the same key.

This example will allow you to perform conditional reads and writes that will succeed or fail based on your selected options.

Sample buckets and a sample object will be created as part of the example.

```

"""
)

bucket_prefix = q.ask("Enter a bucket name prefix: ", q.non_empty)
source_bucket_name = f"{bucket_prefix}-source-{RANDOM_SUFFIX}"
dest_bucket_name = f"{bucket_prefix}-dest-{RANDOM_SUFFIX}"
object_key = "test-upload-file.txt"

try:
    etag = self.setup_scenario(source_bucket_name, dest_bucket_name,
object_key)
    self.display_menu(source_bucket_name, dest_bucket_name, object_key,
etag)
finally:
    self.cleanup_scenario(source_bucket_name, dest_bucket_name)

print("-" * 88)
print("Thanks for watching.")
print("-" * 88)

if __name__ == "__main__":
    scenario = ConditionalRequestsScenario(
        S3ConditionalRequests.from_client(), boto3.client("s3")
    )
    scenario.run_scenario()

```

A wrapper class that defines the conditional request operations.

```
import boto3
import logging

from botocore.exceptions import ClientError

# Configure logging
logger = logging.getLogger(__name__)

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
        'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
```

```

        response = self.s3.get_object(
            Bucket=source_bucket,
            Key=object_key,
            **{condition_type: condition_value},
        )
        sample_bytes = response["Body"].read(20)
        print(
            f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
        )
        print(f"\t{sample_bytes}")
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional read failed: Precondition failed")
        elif error_code == "304": # Not modified error code.
            print("\tConditional read failed: Object not modified")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise

    def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
            )
            print(
                f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]

```

```

        if error_code == "PreconditionFailed":
            print("\tConditional write failed: Precondition failed")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise

def copy_object_conditional(
    self,
    source_key: str,
    dest_key: str,
    source_bucket: str,
    dest_bucket: str,
    condition_type: str,
    condition_value: str,
):
    """
    Copies an object from one Amazon S3 bucket to another with a conditional
    request.

    :param source_key: The key of the source object to copy.
    :param dest_key: The key of the destination object.
    :param source_bucket: The source bucket of the object.
    :param dest_bucket: The destination bucket of the object.
    :param condition_type: The type of condition to apply, e.g.
        'CopySourceIfMatch', 'CopySourceIfNoneMatch',
        'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
    :param condition_value: The value to use for the condition.
    """
    try:
        self.s3.copy_object(
            Bucket=dest_bucket,
            Key=dest_key,
            CopySource={"Bucket": source_bucket, "Key": source_key},
            **{condition_type: condition_value},
        )
        print(
            f"\tConditional copy successful for key {dest_key} in bucket {dest_bucket}."
        )
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional copy failed: Precondition failed")

```

```
elif error_code == "304": # Not modified error code.
    print("\tConditional copy failed: Object not modified")
else:
    logger.error(f"Unexpected error: {error_code}")
    raise
```

- For API details, see the following topics in *Amazon SDK for Python (Boto3) API Reference*.
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Manage access control lists (ACLs) for Amazon S3 buckets using an Amazon SDK

The following code example shows how to manage access control lists (ACLs) for Amazon S3 buckets.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to manage Amazon Simple Storage Service
```



```

    /// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
    /// access.
    /// </summary>
    public class ManageACLs
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket1";
            string newBucketName = "amzn-s3-demo-bucket2";
            string keyName = "sample-object.txt";
            string emailAddress = "someone@example.com";

            // If the AWS Region where your bucket is located is different from
            // the Region defined for the default user, pass the Amazon S3
            bucket's
            // name to the client constructor. It should look like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
            IAmazonS3 client = new AmazonS3Client();

            await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
            keyName, emailAddress);
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
        ACL
        /// information and then adds a new ACL to one of the objects in the
        /// Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        call
        /// methods to create a bucket, get an ACL, and add a different ACL to
        /// one of the objects.</param>
        /// <param name="bucketName">A string representing the original Amazon S3
        /// bucket name.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new bucket that will be created.</param>
        /// <param name="keyName">A string representing the key name of an Amazon
        S3
        /// object for which we will change the ACL.</param>
        /// <param name="emailAddress">A string representing the email address
        /// belonging to the person to whom access to the Amazon S3 bucket will
        be
        /// granted.</param>

```

```

    public static async Task TestBucketObjectACLsAsync(
        IAmazonS3 client,
        string bucketName,
        string newBucketName,
        string keyName,
        string emailAddress)
    {
        try
        {
            // Create a new Amazon S3 bucket and specify canned ACL.
            var success = await CreateBucketWithCannedACLAsync(client,
newBucketName);

            // Get the ACL on a bucket.
            await GetBucketACLAsync(client, bucketName);

            // Add (replace) the ACL on an object in a bucket.
            await AddACLToExistingObjectAsync(client, bucketName, keyName,
emailAddress);
        }
        catch (AmazonS3Exception amazonS3Exception)
        {
            Console.WriteLine($"Exception: {amazonS3Exception.Message}");
        }
    }

    /// <summary>
    /// Creates a new Amazon S3 bucket with a canned ACL attached.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="newBucketName">A string representing the name of the
    /// new Amazon S3 bucket.</param>
    /// <returns>Returns a boolean value indicating success or failure.</
returns>
    public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
client, string newBucketName)
    {
        var request = new PutBucketRequest()
        {
            BucketName = newBucketName,
            BucketRegion = S3Region.EUWest1,

            // Add a canned ACL.

```

```

        CannedACL = S3CannedACL.LogDeliveryWrite,
    };

    var response = await client.PutBucketAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Retrieves the ACL associated with the Amazon S3 bucket name in the
/// bucketName parameter.
/// </summary>
/// <param name="client">The initialized client object used to call
/// PutBucketAsync.</param>
/// <param name="bucketName">The Amazon S3 bucket for which we want to
get the
/// ACL list.</param>
/// <returns>Returns an S3AccessControlList returned from the call to
/// GetACLAsync.</returns>
public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
client, string bucketName)
{
    GetACLResponse response = await client.GetACLAsync(new GetACLRequest
    {
        BucketName = bucketName,
    });

    return response.AccessControlList;
}

/// <summary>
/// Adds a new ACL to an existing object in the Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized client object used to call
/// PutBucketAsync.</param>
/// <param name="bucketName">A string representing the name of the Amazon
S3
/// bucket containing the object to which we want to apply a new ACL.</
param>
/// <param name="keyName">A string representing the name of the object
/// to which we want to apply the new ACL.</param>
/// <param name="emailAddress">The email address of the person to whom

```

```
/// we will be applying to whom access will be granted.</param>
public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
string bucketName, string keyName, string emailAddress)
{
    // Retrieve the ACL for an object.
    GetACLResponse aclResponse = await client.GetACLAsync(new
GetACLRequest
    {
        BucketName = bucketName,
        Key = keyName,
    });

    S3AccessControlList acl = aclResponse.AccessControlList;

    // Retrieve the owner.
    Owner owner = acl.Owner;

    // Clear existing grants.
    acl.Grants.Clear();

    // Add a grant to reset the owner's full permission
    // (the previous clear statement removed all permissions).
    var fullControlGrant = new S3Grant
    {
        Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
    };
    acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

    // Specify email to identify grantee for granting permissions.
    var grantUsingEmail = new S3Grant
    {
        Grantee = new S3Grantee { EmailAddress = emailAddress },
        Permission = S3Permission.WRITE_ACP,
    };

    // Specify log delivery group as grantee.
    var grantLogDeliveryGroup = new S3Grant
    {
        Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
        Permission = S3Permission.WRITE,
    };

    // Create a new ACL.
```

```
var newAcl = new S3AccessControlList
{
    Grants = new List<S3Grant> { grantUsingEmail,
grantLogDeliveryGroup },
    Owner = owner,
};

// Set the new ACL. We're throwing away the response here.
_ = await client.PutACLAsync(new PutACLRequest
{
    BucketName = bucketName,
    Key = keyName,
    AccessControlList = newAcl,
});
}
```

- For API details, see the following topics in *Amazon SDK for .NET API Reference*.
 - [GetBucketAcl](#)
 - [GetObjectAcl](#)
 - [PutBucketAcl](#)
 - [PutObjectAcl](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Manage large Amazon SQS messages using Amazon S3 with an Amazon SDK

The following code example shows how to use the Amazon SQS Extended Client Library to work with large Amazon SQS messages.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
```

```
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
        LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
            DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
            ExtendedClientConfiguration()
```

```
        .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody(largeMessage)
                .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
        }
    }
}
```



```
        if (messages.isEmpty()) {
            throw new RuntimeException("No messages received");
        }

        Message message = messages.getFirst();
        logger.info("\nMessage received.");
        logger.info("  ID: {}", message.messageId());
        logger.info("  Receipt handle: {}", message.receiptHandle());
        logger.info("  Message body size: {}", message.body().length());
        logger.info("  Message body (first 5 characters): {}",
            message.body().substring(0, 5));

        return message;
    } catch (RuntimeException e) {
        logger.error("Error during message processing: {}", e.getMessage(),
            e);
        throw e;
    }
}
```

- For more information, see [Amazon SDK for Java 2.x Developer Guide](#).
- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.
 - [CreateBucket](#)
 - [PutBucketLifecycleConfiguration](#)
 - [ReceiveMessage](#)
 - [SendMessage](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Manage versioned Amazon S3 objects in batches with a Lambda function using an Amazon SDK

The following code example shows how to manage versioned S3 objects in batches with a Lambda function.

Python

SDK for Python (Boto3)

Shows how to manipulate Amazon Simple Storage Service (Amazon S3) versioned objects in batches by creating jobs that call Amazon Lambda functions to perform processing. This example creates a version-enabled bucket, uploads the stanzas from the poem *You Are Old, Father William* by Lewis Carroll, and uses Amazon S3 batch jobs to twist the poem in various ways.

Learn how to:

- Create Lambda functions that operate on versioned objects.
- Create a manifest of objects to update.
- Create batch jobs that invoke Lambda functions to update objects.
- Delete Lambda functions.
- Empty and delete a versioned bucket.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Amazon S3

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Parse Amazon S3 URIs using an Amazon SDK

The following code example shows how to parse Amazon S3 URIs to extract important components like the bucket name and object key.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Parse an Amazon S3 URI by using the [S3Uri](#) class.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

/**
 *
 * @param s3Client      - An S3Client through which you acquire an S3Uri
instance.
 * @param s3objectUrl - A complex URL (String) that is used to demonstrate
S3Uri
 *
capabilities.
 */
public static void parseS3UriExample(S3Client s3Client, String s3objectUrl) {
    logger.info(s3objectUrl);
    // Console output:
    // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

    // Create an S3Utilities object using the configuration of the s3Client.
    S3Utilities s3Utilities = s3Client.utilities();

    // From a String URL create a URI object to pass to the parseUri()
method.
    URI uri = URI.create(s3objectUrl);
```

```
S3Uri s3Uri = s3Utilities.parseUri(uri);

// If the URI contains no value for the Region, bucket or key, the SDK
returns
// an empty Optional.
// The SDK returns decoded URI values.

Region region = s3Uri.region().orElse(null);
log("region", region);
// Console output: 'region: us-west-1'.

String bucket = s3Uri.bucket().orElse(null);
log("bucket", bucket);
// Console output: 'bucket: myBucket'.

String key = s3Uri.key().orElse(null);
log("key", key);
// Console output: 'key: resources/doc.txt'.

Boolean isPathStyle = s3Uri.isPathStyle();
log("isPathStyle", isPathStyle);
// Console output: 'isPathStyle: true'.

// If the URI contains no query parameters, the SDK returns an empty map.
Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
log("rawQueryParameters", queryParams);
// Console output: 'rawQueryParameters: {versionId=[abc123],
partNumber=[77,
// 88]]'.

// Retrieve the first or all values for a query parameter as shown in the
// following code.
String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
log("firstMatchingRawQueryParameter-versionId", versionId);
// Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
log("firstMatchingRawQueryParameter-partNumber", partNumber);
// Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
```

```

        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters,
must be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?
query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".",
the dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
            logger.info("{}: {}", s3UriElement, element);
        }
    }
}

```

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Perform a multipart copy of an Amazon S3 object using an Amazon SDK

The following code example shows how to perform a multipart copy of an Amazon S3 object.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUApiCopyObj
{
    private const string SourceBucket = "amzn-s3-demo-bucket1";
    private const string TargetBucket = "amzn-s3-demo-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";

    /// <summary>
    /// This method starts the multi-part upload.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        Console.WriteLine("Copying object...");
        await MPUCopyObjectAsync(s3Client);
    }

    /// <summary>
    /// This method uses the passed client object to perform a multipart
    /// copy operation.
    /// </summary>
    /// <param name="client">An Amazon S3 client object that will be used
```

```
/// to perform the copy.</param>
public static async Task MPUCopyObjectAsync(AmazonS3Client client)
{
    // Create a list to store the copy part responses.
    var copyResponses = new List<CopyPartResponse>();

    // Setup information required to initiate the multipart upload.
    var initiateRequest = new InitiateMultipartUploadRequest
    {
        BucketName = TargetBucket,
        Key = TargetObjectKey,
    };

    // Initiate the upload.
    InitiateMultipartUploadResponse initResponse =
        await client.InitiateMultipartUploadAsync(initiateRequest);

    // Save the upload ID.
    string uploadId = initResponse.UploadId;

    try
    {
        // Get the size of the object.
        var metadataRequest = new GetObjectMetadataRequest
        {
            BucketName = SourceBucket,
            Key = SourceObjectKey,
        };

        GetObjectMetadataResponse metadataResponse =
            await client.GetObjectMetadataAsync(metadataRequest);
        var objectSize = metadataResponse.ContentLength; // Length in
bytes.

        // Copy the parts.
        var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

        long bytePosition = 0;
        for (int i = 1; bytePosition < objectSize; i++)
        {
            var copyRequest = new CopyPartRequest
            {
                DestinationBucket = TargetBucket,
                DestinationKey = TargetObjectKey,
```

```

        SourceBucket = SourceBucket,
        SourceKey = SourceObjectKey,
        UploadId = uploadId,
        FirstByte = bytePosition,
        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
        PartNumber = i,
    };

    copyResponses.Add(await client.CopyPartAsync(copyRequest));

    bytePosition += partSize;
}

// Set up to complete the copy.
var completeRequest = new CompleteMultipartUploadRequest
{
    BucketName = TargetBucket,
    Key = TargetObjectKey,
    UploadId = initResponse.UploadId,
};
completeRequest.AddPartETags(copyResponses);

// Complete the copy.
CompleteMultipartUploadResponse completeUploadResponse =
    await client.CompleteMultipartUploadAsync(completeRequest);
}
catch (AmazonS3Exception e)
{
    Console.WriteLine($"Error encountered on server.
Message: '{e.Message}' when writing an object");
}
catch (Exception e)
{
    Console.WriteLine($"Unknown encountered on server.
Message: '{e.Message}' when writing an object");
}
}
}

```

- For API details, see the following topics in *Amazon SDK for .NET API Reference*.

- [CompleteMultipartUpload](#)
- [CreateMultipartUpload](#)
- [GetObjectMetadata](#)
- [UploadPartCopy](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Receive and process Amazon S3 event notifications by using an Amazon SDK

The following code example shows how to work with S3 event notifications in an object-oriented way.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

This example show how to process S3 notification event by using Amazon SQS.

```
/**
 * This method receives S3 event notifications by using an SqsAsyncClient.
 * After the client receives the messages it deserializes the JSON payload
and logs them. It uses
 * the S3EventNotification class (part of the S3 event notification API for
Java) to deserialize
 * the JSON payload and access the messages in an object-oriented way.
 *
 * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
notifications.
 * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification
API</a>.
 * <p>
```

```

    * To use S3 event notification serialization/deserialization to objects, add
    the following
    * dependency to your Maven pom.xml file.
    * <dependency>
    * <groupId>software.amazon.awssdk</groupId>
    * <artifactId>s3-event-notifications</artifactId>
    * <version><LATEST></version>
    * </dependency>
    * <p>
    * The S3 event notification API became available with version 2.25.11 of the
    Java SDK.
    * <p>
    * This example shows the use of the API with AWS SQS, but it can be used to
    process S3 event notifications
    * in AWS SNS or AWS Lambda as well.
    * <p>
    * Note: The S3EventNotification class does not work with messages routed
    through AWS EventBridge.
    */
    static void processS3Events(String bucketName, String queueUrl, String
    queueArn) {
        try {
            // Configure the bucket to send Object Created and Object Tagging
            notifications to an existing SQS queue.
            s3Client.putBucketNotificationConfiguration(b -> b
                .notificationConfiguration(ncb -> ncb
                    .queueConfigurations(qcb -> qcb
                        .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                            .queueArn(queueArn)))
                    .bucket(bucketName)
                ).join();

            triggerS3EventNotifications(bucketName);
            // Wait for event notifications to propagate.
            Thread.sleep(Duration.ofSeconds(5).toMillis());

            boolean didReceiveMessages = true;
            while (didReceiveMessages) {
                // Display the number of messages that are available in the
                queue.

                sqsClient.getQueueAttributes(b -> b
                    .queueUrl(queueUrl)

```

```

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                ).thenAccept(attributeResponse ->
                    logger.info("Approximate number of messages in
the queue: {}"),
attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                .join();

        // Receive the messages.
        ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
            .queueUrl(queueUrl)
            ).get();
        logger.info("Count of received messages: {}",
response.messages().size());
        didReceiveMessages = !response.messages().isEmpty();

        // Create a collection to hold the received message for deletion
        // after we log the messages.
        HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();

        // Process each message.
        response.messages().forEach(message -> {
            logger.info("Message id: {}", message.messageId());
            // Deserialize JSON message body to a S3EventNotification
object

            // to access messages in an object-oriented way.
            S3EventNotification event =
S3EventNotification.fromJson(message.body());

            // Log the S3 event notification record details.
            if (event.getRecords() != null) {
                event.getRecords().forEach(record -> {
                    String eventName = record.getEventName();
                    String key = record.getS3().getObject().getKey();
                    logger.info(record.toString());
                    logger.info("Event name is {} and key is {}",
eventName, key);

                });
            }
            // Add logged messages to collection for batch deletion.
            messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                .id(message.messageId())
                .receiptHandle(message.receiptHandle())

```

```
                .build());
            });
            // Delete messages.
            if (!messagesToDelete.isEmpty()) {

                sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(messagesToDelete)
                    .build()
                ).join();
            }
        } // End of while block.
    } catch (InterruptedException | ExecutionException e) {
        throw new RuntimeException(e);
    }
}
```

- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.
 - [DeleteMessageBatch](#)
 - [GetQueueAttributes](#)
 - [PutBucketNotificationConfiguration](#)
 - [ReceiveMessage](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Save EXIF and other image information using an Amazon SDK

The following code example shows how to:

- Get EXIF information from a a JPG, JPEG, or PNG file.
- Upload the image file to an Amazon S3 bucket.
- Use Amazon Rekognition to identify the three top attributes (labels) in the file.
- Add the EXIF and label information to an Amazon DynamoDB table in the Region.

Rust

SDK for Rust

Get EXIF information from a JPG, JPEG, or PNG file, upload the image file to an Amazon S3 bucket, use Amazon Rekognition to identify the three top attributes (*labels* in Amazon Rekognition) in the file, and add the EXIF and label information to a Amazon DynamoDB table in the Region.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Send S3 event notifications to Amazon EventBridge using an Amazon SDK

The following code example shows how to enable a bucket to send S3 event notifications to EventBridge and route notifications to an Amazon SNS topic and Amazon SQS queue.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/** This method configures a bucket to send events to AWS EventBridge and
    creates a rule
    * to route the S3 object created events to a topic and a queue.
```

```

*
* @param bucketName Name of existing bucket
* @param topicArn ARN of existing topic to receive S3 event notifications
* @param queueArn ARN of existing queue to receive S3 event notifications
*
* An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
*/
public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
    try {
        // Enable bucket to emit S3 Event notifications to EventBridge.
        s3Client.putBucketNotificationConfiguration(b -> b
            .bucket(bucketName)
            .notificationConfiguration(b1 -> b1
                .eventBridgeConfiguration(
                    SdkBuilder::build)
            ).build()).join();

        // Create an EventBridge rule to route Object Created notifications.
        PutRuleRequest putRuleRequest = PutRuleRequest.builder()
            .name(RULE_NAME)
            .eventPattern("""
                {
                    "source": ["aws.s3"],
                    "detail-type": ["Object Created"],
                    "detail": {
                        "bucket": {
                            "name": ["%s"]
                        }
                    }
                }
            """).formatted(bucketName))
            .build();

        // Add the rule to the default event bus.
        PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
            .whenComplete((r, t) -> {
                if (t != null) {
                    logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                    throw new RuntimeException(t.getCause().getMessage(),
t);

```

```

        }
        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
    }).join();

    // Add the existing SNS topic and SQS queue as targets to the rule.
    eventBridgeClient.putTargets(b -> b
        .eventBusName("default")
        .rule(RULE_NAME)
        .targets(List.of (
            Target.builder()
                .arn(queueArn)
                .id("Queue")
                .build(),
            Target.builder()
                .arn(topicArn)
                .id("Topic")
                .build()
        )
    ).join();
    return putRuleResponse.ruleArn();
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}

```

- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.
 - [PutBucketNotificationConfiguration](#)
 - [PutRule](#)
 - [PutTargets](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Track an Amazon S3 object upload or download using an Amazon SDK

The following code example shows how to track an Amazon S3 object upload or download.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Track the progress of a file upload.

```
public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .source(Paths.get(filePathURI))
        .build();
```

```
FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
```

```
fileUpload.completionFuture().join();
```

```
/*
```

The SDK provides a `LoggingTransferListener` implementation of the `TransferListener` interface.

You can also implement the interface to provide your own logic.

Configure log4j2 with settings such as the following.

```
<Configuration status="WARN">
    <Appenders>
        <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
            <PatternLayout pattern="%m%n"/>
        </Console>
    </Appenders>

    <Loggers>
        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
```



```

        <AppenderRef ref="AlignedConsoleAppender"/>
    </logger>
</Loggers>
</Configuration>

Log4J2 logs the progress. The following is example output for a 21.3
MB file upload.
    Transfer initiated...
    |                               | 0.0%
    |====                         | 21.1%
    |=====                      | 60.5%
    |=====|                    | 100.0%
    Transfer complete!

    */
}

```

Track the progress of a file download.

```

public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                               String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .destination(Paths.get(downloadedFilePath))
        .build();

    FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

    CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
    /*
        The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
        You can also implement the interface to provide your own logic.

        Configure log4J2 with settings such as the following.
        <Configuration status="WARN">
            <Appenders>

```

```

        <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
            <PatternLayout pattern="%m%n"/>
        </Console>
    </Appenders>

    <Loggers>
        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
            <AppenderRef ref="AlignedConsoleAppender"/>
        </logger>
    </Loggers>
</Configuration>

```

Log4J2 logs the progress. The following is example output for a 21.3 MB file download.

```

    Transfer initiated...
    |=====| 39.4%
    |=====| 78.8%
    |=====| 100.0%
    Transfer complete!

    */
}

```

- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.
 - [GetObject](#)
 - [PutObject](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Transform data for your application with S3 Object Lambda

The following code example shows how to transform data for your application with S3 Object Lambda.

.NET

Amazon SDK for .NET

Shows how to add custom code to standard S3 GET requests to modify the requested object retrieved from S3 so that the object suit the needs of the requesting client or application.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

Services used in this example

- Lambda
- Amazon S3

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Example approaches for unit and integration testing with an Amazon SDK

The following code example shows how to examples for best-practice techniques when writing unit and integration tests using an Amazon SDK.

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Cargo.toml for testing examples.

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
```

```

    "Doug Schwartz <dougsch@amazon.com>",
  ]
  edition = "2021"

  [dependencies]
  async-trait = "0.1.51"
  aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
  aws-credential-types = { version = "1.0.1", features = [ "hardcoded-credentials", ] }
  aws-sdk-s3 = { version = "1.4.0" }
  aws-smithy-types = { version = "1.0.1" }
  aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
  aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
  aws-types = { version = "1.0.1" }
  clap = { version = "4.4", features = ["derive"] }
  http = "0.2.9"
  mockall = "0.11.4"
  serde_json = "1"
  tokio = { version = "1.20.1", features = ["full"] }
  tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }

  [[bin]]
  name = "main"
  path = "src/main.rs"

```

Unit testing example using automock and a service wrapper.

```

use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};

#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
pub use S3Impl as S3;

#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,

```

```

}

#[cfg_attr(test, automock)]
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
    }
}

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }
    }
}

```

```

    }

    // Handle pagination, and break the loop if there are no more pages
    next_token = result.next_continuation_token.clone();
    if next_token.is_none() {
        break;
    }
}
Ok(total_size_bytes)
}

#[cfg(test)]
mod test {
    use super::*;
    use mockall::predicate::eq;

    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
    }

    #[tokio::test]
    async fn test_multiple_pages() {
        // Create the Mock instance with two pages of objects now
        let mut mock = MockS3Impl::default();
    }

```

```

    mock.expect_list_objects()
      .with(eq("test-bucket"), eq("test-prefix"), eq(None))
      .return_once(|_, _, _| {
        Ok(ListObjectsV2Output::builder()
          .set_contents(Some(vec![
            // Mock content for ListObjectsV2 response
            s3::types::Object::builder().size(5).build(),
            s3::types::Object::builder().size(2).build(),
          ]))
          .set_next_continuation_token(Some("next".to_string()))
          .build())
      });
    mock.expect_list_objects()
      .with(
        eq("test-bucket"),
        eq("test-prefix"),
        eq(Some("next".to_string()))
      )
      .return_once(|_, _, _| {
        Ok(ListObjectsV2Output::builder()
          .set_contents(Some(vec![
            // Mock content for ListObjectsV2 response
            s3::types::Object::builder().size(3).build(),
            s3::types::Object::builder().size(9).build(),
          ]))
          .build())
      });

    // Run the code we want to test with it
    let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
      .await
      .unwrap();

    assert_eq!(19, size);
  }
}

```

Integration testing example using StaticReplayClient.

```
use aws_sdk_s3 as s3;
```

```

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[allow(dead_code)]
fn make_s3_test_credentials() -> s3::config::Credentials {
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "astestsecretkey",
        Some("atestsessiontoken".to_string()),
        None,
        "",
    )
}

```



```
#[cfg(test)]
mod test {
    use super::*;
    use aws_config::BehaviorVersion;
    use aws_sdk_s3 as s3;
    use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
StaticReplayClient};
    use aws_smithy_types::body::SdkBody;

    #[tokio::test]
    async fn test_single_page() {
        let page_1 = ReplayEvent::new(
            http::Request::builder()
                .method("GET")
                .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                .body(SdkBody::empty())
                .unwrap(),
            http::Response::builder()
                .status(200)
                .body(SdkBody::from(include_str!("./testing/
response_1.xml")))
                .unwrap(),
        );
        let replay_client = StaticReplayClient::new(vec![page_1]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
        replay_client.assert_requests_match(&[]);
    }
}
```

```

#[tokio::test]
async fn test_multiple_pages() {
    let page_1 = ReplayEvent::new(
        http::Request::builder()
            .method("GET")
            .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
            .body(SdkBody::empty())
            .unwrap(),
        http::Response::builder()
            .status(200)
            .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
            .unwrap(),
    );
    let page_2 = ReplayEvent::new(
        http::Request::builder()
            .method("GET")
            .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
            .body(SdkBody::empty())
            .unwrap(),
        http::Response::builder()
            .status(200)
            .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))
            .unwrap(),
    );
    let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
    let client: s3::Client = s3::Client::from_conf(
        s3::Config::builder()
            .behavior_version(BehaviorVersion::latest())
            .credentials_provider(make_s3_test_credentials())
            .region(s3::config::Region::new("us-east-1"))
            .http_client(replay_client.clone())
            .build(),
    );

    // Run the code we want to test with it
    let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
        .await
        .unwrap();

```

```
        assert_eq!(19, size);

        replay_client.assert_requests_match(&[]);
    }
}
```

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Recursively upload a local directory to an Amazon Simple Storage Service (Amazon S3) bucket

The following code example shows how to upload a local directory recursively to an Amazon Simple Storage Service (Amazon S3) bucket.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Use an [S3TransferManager](#) to [upload a local directory](#). View the [complete file](#) and [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
```

```
import java.net.URL;
import java.nio.file.Paths;

    public Integer uploadDirectory(S3TransferManager transferManager,
        URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
        .source(Paths.get(sourceDirectory))
        .bucket(bucketName)
        .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
            .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- For API details, see [UploadDirectory](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Upload or download large files to and from Amazon S3 using an Amazon SDK

The following code examples show how to upload or download large files to and from Amazon S3.

For more information, see [Uploading an object using multipart upload](#).

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Call functions that transfer files to and from an S3 bucket using the Amazon S3 TransferUtility.

```
global using System.Text;
global using Amazon.S3;
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;

// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
    $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
    \TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");
```

```
var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");

var success = await TransferMethods.UploadSingleFileAsync(transferUtil,
    bucketName, fileToUpload, localPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
    {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil,
    bucketName, keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
    {bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];
```

```
Console.WriteLine($"Downloading {keyName} from {bucketName}.");

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
    keyName, localPath);
if (success)
{
    Console.WriteLine($"Successfully downloaded the file, {keyName} from
    {bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil,
    bucketName, s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to
    {downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}
```

```
}

// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);

    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to
an\n\t S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
directory.");
    Console.WriteLine($"
\n{sepBar}");
}

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("
");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth -
textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
```



```
        if (fileList.Length > 0)
        {
            foreach (var fileName in fileList)
            {
                Console.WriteLine(fileName);
            }
        }
    }

    // Displays a list of the files in the specified S3 bucket and prefix.
    static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
    s3Path)
    {
        ListObjectsV2Request request = new()
        {
            BucketName = bucketName,
            Prefix = s3Path,
            MaxKeys = 5,
        };

        var response = new ListObjectsV2Response();

        do
        {
            response = await client.ListObjectsV2Async(request);

            response.S3Objects
                .ForEach(obj => Console.WriteLine($"{obj.Key}"));

            // If the response is truncated, set the request ContinuationToken
            // from the NextContinuationToken property of the response.
            request.ContinuationToken = response.NextContinuationToken;
        } while (response.IsTruncated);
    }
}
```

Upload a single file.

```
/// <summary>
/// Uploads a single file from the local computer to an S3 bucket.
/// </summary>
```

```
    /// <param name="transferUtil">The transfer initialized TransferUtility
    /// object.</param>
    /// <param name="bucketName">The name of the S3 bucket where the file
    /// will be stored.</param>
    /// <param name="fileName">The name of the file to upload.</param>
    /// <param name="localPath">The local path where the file is stored.</
param>
    /// <returns>A boolean value indicating the success of the action.</
returns>
    public static async Task<bool> UploadSingleFileAsync(
        TransferUtility transferUtil,
        string bucketName,
        string fileName,
        string localPath)
    {
        if (File.Exists($"{localPath}\\{fileName}"))
        {
            try
            {
                await transferUtil.UploadAsync(new
TransferUtilityUploadRequest
                {
                    BucketName = bucketName,
                    Key = fileName,
                    FilePath = $"{localPath}\\{fileName}",
                });

                return true;
            }
            catch (AmazonS3Exception s3Ex)
            {
                Console.WriteLine($"Could not upload {fileName} from
{localPath} because:");
                Console.WriteLine(s3Ex.Message);
                return false;
            }
        }
        else
        {
            Console.WriteLine($"{fileName} does not exist in {localPath}");
            return false;
        }
    }
}
```

Upload an entire local directory.

```
/// <summary>
/// Uploads all the files in a local directory to a directory in an S3
/// bucket.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The name of the S3 bucket where the files
/// will be stored.</param>
/// <param name="keyPrefix">The key prefix is the S3 directory where
/// the files will be stored.</param>
/// <param name="localPath">The local directory that contains the files
/// to be uploaded.</param>
/// <returns>A Boolean value representing the success of the action.</
returns>
public static async Task<bool> UploadFullDirectoryAsync(
    TransferUtility transferUtil,
    string bucketName,
    string keyPrefix,
    string localPath)
{
    if (Directory.Exists(localPath))
    {
        try
        {
            await transferUtil.UploadDirectoryAsync(new
TransferUtilityUploadDirectoryRequest
            {
                BucketName = bucketName,
                KeyPrefix = keyPrefix,
                Directory = localPath,
            });

            return true;
        }
        catch (AmazonS3Exception s3Ex)
        {
            Console.WriteLine($"Can't upload the contents of {localPath}
because:");
        }
    }
}
```

```

        Console.WriteLine(s3Ex?.Message);
        return false;
    }
}
else
{
    Console.WriteLine($"The directory {localPath} does not exist.");
    return false;
}
}

```

Download a single file.

```

/// <summary>
/// Download a single file from an S3 bucket to the local computer.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The name of the S3 bucket containing the
/// file to download.</param>
/// <param name="keyName">The name of the file to download.</param>
/// <param name="localPath">The path on the local computer where the
/// downloaded file will be saved.</param>
/// <returns>A Boolean value indicating the results of the action.</
returns>
public static async Task<bool> DownloadSingleFileAsync(
    TransferUtility transferUtil,
    string bucketName,
    string keyName,
    string localPath)
{
    await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
    {
        BucketName = bucketName,
        Key = keyName,
        FilePath = $"{localPath}\\{keyName}",
    });

    return (File.Exists($"{localPath}\\{keyName}"));
}

```

Download contents of an S3 bucket.

```
/// <summary>
/// Downloads the contents of a directory in an S3 bucket to a
/// directory on the local computer.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The bucket containing the files to
download.</param>
/// <param name="s3Path">The S3 directory where the files are located.</
param>
/// <param name="localPath">The local path to which the files will be
/// saved.</param>
/// <returns>A Boolean value representing the success of the action.</
returns>
public static async Task<bool> DownloadS3DirectoryAsync(
    TransferUtility transferUtil,
    string bucketName,
    string s3Path,
    string localPath)
{
    int fileCount = 0;

    // If the directory doesn't exist, it will be created.
    if (Directory.Exists(s3Path))
    {
        var files = Directory.GetFiles(localPath);
        fileCount = files.Length;
    }

    await transferUtil.DownloadDirectoryAsync(new
TransferUtilityDownloadDirectoryRequest
    {
        BucketName = bucketName,
        LocalDirectory = localPath,
        S3Directory = s3Path,
    });
}
```

```
        if (Directory.Exists(localPath))
        {
            var files = Directory.GetFiles(localPath);
            if (files.Length > fileCount)
            {
                return true;
            }

            // No change in the number of files. Assume
            // the download failed.
            return false;
        }

        // The local directory doesn't exist. No files
        // were downloaded.
        return false;
    }
}
```

Track the progress of an upload using the `TransferUtility`.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;

/// <summary>
/// This example shows how to track the progress of a multipart upload
/// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
/// upload to an Amazon S3 bucket.
/// </summary>
public class TrackMPUUsingHighLevelAPI
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "sample_pic.png";
        string path = "filepath/directory/";
        string filePath = $"{path}{keyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
```

```
// pass the Region name to the Amazon S3 client object's constructor.
// For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
IAmazonS3 client = new AmazonS3Client();

await TrackMPUAsync(client, bucketName, filePath, keyName);
}

/// <summary>
/// Starts an Amazon S3 multipart upload and assigns an event handler to
/// track the progress of the upload.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
/// perform the multipart upload.</param>
/// <param name="bucketName">The name of the bucket to which to upload
/// the file.</param>
/// <param name="filePath">The path, including the file name of the
/// file to be uploaded to the Amazon S3 bucket.</param>
/// <param name="keyName">The file name to be used in the
/// destination Amazon S3 bucket.</param>
public static async Task TrackMPUAsync(
    IAmazonS3 client,
    string bucketName,
    string filePath,
    string keyName)
{
    try
    {
        var fileTransferUtility = new TransferUtility(client);

        // Use TransferUtilityUploadRequest to configure options.
        // In this example we subscribe to an event.
        var uploadRequest =
            new TransferUtilityUploadRequest
            {
                BucketName = bucketName,
                FilePath = filePath,
                Key = keyName,
            };

        uploadRequest.UploadProgressEvent +=
            new EventHandler<UploadProgressArgs>(
                UploadRequest_UploadPartProgressEvent);

        await fileTransferUtility.UploadAsync(uploadRequest);
    }
}
```

```

        Console.WriteLine("Upload completed");
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error:: {ex.Message}");
    }
}

/// <summary>
/// Event handler to check the progress of the multipart upload.
/// </summary>
/// <param name="sender">The object that raised the event.</param>
/// <param name="e">The object that contains multipart upload
/// information.</param>
public static void UploadRequest_UploadPartProgressEvent(object sender,
UploadProgressArgs e)
{
    // Process event.
    Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
}
}

```

Upload an object with encryption.

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
/// perform a multipart upload to an Amazon S3 bucket.
/// </summary>
public class SSECLowLevelMPUCopyObject
{
    public static async Task Main()
    {
        string existingBucketName = "amzn-s3-demo-bucket";
    }
}

```



```

        string sourceKeyName = "sample_file.txt";
        string targetKeyName = "sample_file_copy.txt";
        string filePath = $"sample\\{targetKeyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USEast1.
        IAmazonS3 client = new AmazonS3Client();

        // Create the encryption key.
        var base64Key = CreateEncryptionKey();

        await CreateSampleObjUsingClientEncryptionKeyAsync(
            client,
            existingBucketName,
            sourceKeyName,
            filePath,
            base64Key);
    }

    /// <summary>
    /// Creates the encryption key to use with the multipart upload.
    /// </summary>
    /// <returns>A string containing the base64-encoded key for encrypting
    /// the multipart upload.</returns>
    public static string CreateEncryptionKey()
    {
        Aes aesEncryption = Aes.Create();
        aesEncryption.KeySize = 256;
        aesEncryption.GenerateKey();
        string base64Key = Convert.ToBase64String(aesEncryption.Key);
        return base64Key;
    }

    /// <summary>
    /// Creates and uploads an object using a multipart upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 object used to
    /// initialize and perform the multipart upload.</param>
    /// <param name="existingBucketName">The name of the bucket to which
    /// the object will be uploaded.</param>
    /// <param name="sourceKeyName">The source object name.</param>
    /// <param name="filePath">The location of the source object.</param>

```

```
    /// <param name="base64Key">The encryption key to use with the upload.</
param>
    public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
        IAmazonS3 client,
        string existingBucketName,
        string sourceKeyName,
        string filePath,
        string base64Key)
    {
        List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

        InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
        {
            BucketName = existingBucketName,
            Key = sourceKeyName,
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };

        InitiateMultipartUploadResponse initResponse =
            await client.InitiateMultipartUploadAsync(initiateRequest);

        long contentLength = new FileInfo(filePath).Length;
        long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

        try
        {
            long filePosition = 0;
            for (int i = 1; filePosition < contentLength; i++)
            {
                UploadPartRequest uploadRequest = new UploadPartRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                    PartNumber = i,
                    PartSize = partSize,
                    FilePosition = filePosition,
                    FilePath = filePath,
                    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
```

```
        ServerSideEncryptionCustomerProvidedKey = base64Key,
    };

    // Upload part and add response to our list.
    uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));

    filePosition += partSize;
}

    CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
    {
        BucketName = existingBucketName,
        Key = sourceKeyName,
        UploadId = initResponse.UploadId,
    };
    completeRequest.AddPartETags(uploadResponses);

    CompleteMultipartUploadResponse completeUploadResponse =
        await client.CompleteMultipartUploadAsync(completeRequest);
}
catch (Exception exception)
{
    Console.WriteLine($"Exception occurred: {exception.Message}");

    // If there was an error, abort the multipart upload.
    AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
    {
        BucketName = existingBucketName,
        Key = sourceKeyName,
        UploadId = initResponse.UploadId,
    };

    await client.AbortMultipartUploadAsync(abortMPURequest);
}
}
```

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create functions that use upload and download managers to break the data into parts and transfer them concurrently.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}
```

```

// UploadLargeObject uses an upload manager to upload data to an object in a
// bucket.
// The upload manager breaks large data into parts and uploads the parts
// concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
string, objectKey string, largeObject []byte) error {
    largeBuffer := bytes.NewReader(largeObject)
    var partMiBs int64 = 10
    uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
        u.PartSize = partMiBs * 1024 * 1024
    })
    _, err := uploader.Upload(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
        Body:    largeBuffer,
    })
    if err != nil {
        var apiErr smithy.APIError
        if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
            log.Printf("Error while uploading object to %s. The object is too large.\n"+
                "The maximum size for a multipart upload is 5TB.", bucketName)
        } else {
            log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
                bucketName, objectKey, err)
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }

    return err
}

// DownloadLargeObject uses a download manager to download an object from a
// bucket.
// The download manager gets the data in parts and writes them to a buffer until
// all of

```

```
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10
    downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
    {
        d.PartSize = partMiBs * 1024 * 1024
    })
    buffer := manager.NewWriteAtBuffer([]byte{})
    _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    })
    if err != nil {
        log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return buffer.Bytes(), err
}
```

Run an interactive scenario that shows you how to use the upload and download managers in context.

```
import (
    "context"
    "crypto/rand"
    "log"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunLargeObjectScenario is an interactive example that shows you how to use
// Amazon
// Simple Storage Service (Amazon S3) to upload and download large objects.
//
// 1. Create a bucket.
```

```
// 3. Upload a large object to the bucket by using an upload manager.
// 5. Download a large object by using a download manager.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../../demotools folder of this repo.
func RunLargeObjectScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := questioner.(*demotools.MockQuestioner)
            if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
                log.Println(r)
            }
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon S3 large object demo.")
    log.Println(strings.Repeat("-", 88))

    s3Client := s3.NewFromConfig(sdkConfig)
    bucketBasics := actions.BucketBasics{S3Client: s3Client}

    bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
        demotools.NotEmpty{})
    bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
    if err != nil {
        panic(err)
    }
    if !bucketExists {
        err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
        if err != nil {
            panic(err)
        } else {

```

```
    log.Println("Bucket created.")
}
}
log.Println(strings.Repeat("-", 88))

mibs := 30
log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your
bucket. ", mibs)
questioner.Ask("Press Enter when you're ready.")
largeBytes := make([]byte, 1024*1024*mibs)
_, _ = rand.Read(largeBytes)
largeKey := "doc-example-large"
log.Println("Uploading...")
err = bucketBasics.UploadLargeObject(ctx, bucketName, largeKey, largeBytes)
if err != nil {
    panic(err)
}
log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download the %v MiB object.", mibs)
questioner.Ask("Press Enter when you're ready.")
log.Println("Downloading...")
largeDownload, err := bucketBasics.DownloadLargeObject(ctx, bucketName,
largeKey)
if err != nil {
    panic(err)
}
log.Printf("Downloaded %v bytes.", len(largeDownload))
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
"contents? (y/n)", "y") {
    log.Println("Deleting object.")
    err = bucketBasics.DeleteObjects(ctx, bucketName, []string{largeKey})
    if err != nil {
        panic(err)
    }
    log.Println("Deleting bucket.")
    err = bucketBasics.DeleteBucket(ctx, bucketName)
    if err != nil {
        panic(err)
    }
} else {
```



```
log.Println("Okay. Don't forget to delete objects from your bucket to avoid
charges.")
}
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Call functions that transfer files to and from an S3 bucket using the `S3TransferManager`.

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
        .destination(Paths.get(destinationPathURI))
        .bucket(bucketName)
        .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

Upload an entire local directory.

```
public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
    .source(Paths.get(sourceDirectory))
    .bucket(bucketName)
    .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}
```

Upload a single file.

```
public String uploadFile(S3TransferManager transferManager, String
bucketName,

    String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

The code examples use the following imports.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

Use the [S3 Transfer Manager](#) on top of the [Amazon CRT-based S3 client](#) to transparently perform a multipart upload when the size of the content exceeds a threshold. The default threshold size is 8 MB.

```
/**
 * Uploads a file to an Amazon S3 bucket using the S3TransferManager.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the file path of the file to be uploaded
 */
public void multipartUploadWithTransferManager(String bucketName, String key,
String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key))
```

```

        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}

```

Use the [S3Client API](#) to perform a multipart upload.

```

/**
 * Performs a multipart upload to Amazon S3 using the provided S3 client.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the path to the file to be uploaded
 */
public void multipartUploadWithS3Client(String bucketName, String key, String
filePath) {

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)

```

```

        .key(key)
        .uploadId(uploadId)
        .partNumber(partNumber)
        .build();

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNumber)
            .eTag(partResponse.eTag())
            .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    logger.error(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

    .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}

```

Use the [S3AsyncClient API](#) with multipart support enabled to perform a multipart upload.

```

/**
 * Uploads a file to an S3 bucket using the S3AsyncClient and enabling
 * multipart support.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the local file path of the file to be uploaded

```

```
    */
    public void multipartUploadWithS3AsyncClient(String bucketName, String key,
String filePath) {
        // Enable multipart support.
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
            .multipartEnabled(true)
            .build();

        CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b
-> b
            .bucket(bucketName)
            .key(key),
            Paths.get(filePath));

        response.join();
        logger.info("File uploaded in multiple 8 MiB parts using
S3AsyncClient.");
    }
}
```

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Upload a large file.

```
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

import {
    ProgressBar,
    logger,
} from "@aws-doc-sdk-examples/lib/utils/util-log.js";

const twentyFiveMB = 25 * 1024 * 1024;
```

```
export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

/**
 * Create a 25MB file and upload it in parts to the specified
 * Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const str = createString();
  const buffer = Buffer.from(str, "utf8");
  const progressBar = new ProgressBar({
    description: `Uploading "${key}" to "${bucketName}"`,
    barLength: 30,
  });

  try {
    const upload = new Upload({
      client: new S3Client({}),
      params: {
        Bucket: bucketName,
        Key: key,
        Body: buffer,
      },
    });

    upload.on("httpUploadProgress", ({ loaded, total }) => {
      progressBar.update({ current: loaded, total });
    });

    await upload.done();
  } catch (caught) {
    if (caught instanceof Error && caught.name === "AbortError") {
      logger.error(`Multipart upload was aborted. ${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

Download a large file.

```
import { fileURLToPath } from "node:url";
import { GetObjectCommand, NoSuchKey, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream, rmSync } from "node:fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: Number.parseInt(start),
    end: Number.parseInt(end),
    length: Number.parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };
  }
}
```



```
const { ContentRange, Body } = await getObjectRange({
  bucket,
  key,
  ...nextRange,
});
console.log(`Downloaded bytes ${nextRange.start} to ${nextRange.end}`);

writeStream.write(await Body.transformToByteArray());
rangeAndLength = getRangeAndLength(ContentRange);
}
};

/**
 * Download a large object from and Amazon S3 bucket.
 *
 * When downloading a large file, you might want to break it down into
 * smaller pieces. Amazon S3 accepts a Range header to specify the start
 * and end of the byte range to be downloaded.
 *
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  try {
    await downloadInChunks({
      bucket: bucketName,
      key: key,
    });
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(`Failed to download object. No such key "${key}".`);
      rmSync(key);
    }
  }
};
```

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create functions that transfer files using several of the available transfer manager settings. Use a callback class to write callback progress during file transfer.

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig

MB = 1024 * 1024
s3 = boto3.resource("s3")

class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.
```

Display progress during file transfer and collect per-thread transfer data. This method can be called by multiple threads, so shared instance data is protected by a thread lock.

```

"""
thread = threading.current_thread()
with self._lock:
    self._total_transferred += bytes_transferred
    if thread.ident not in self.thread_info.keys():
        self.thread_info[thread.ident] = bytes_transferred
    else:
        self.thread_info[thread.ident] += bytes_transferred

    target = self._target_size * MB
    sys.stdout.write(
        f"\r{self._total_transferred} of {target} transferred "
        f"({(self._total_transferred / target) * 100:.2f}%)."
    )
    sys.stdout.flush()

```

```

def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info

```

```

def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

```

The multipart chunk size controls the size of the chunks of data that are sent in the request. A smaller chunk size typically results in the transfer

manager using more threads for the upload.

The metadata is a set of key-value pairs that are stored with the object in Amazon S3.

```
"""
```

```
transfer_callback = TransferCallback(file_size_mb)
```

```
config = TransferConfig(multipart_chunksize=1 * MB)
```

```
extra_args = {"Metadata": metadata} if metadata else None
```

```
s3.Bucket(bucket_name).upload_file(
```

```
    local_file_path,
```

```
    object_key,
```

```
    Config=config,
```

```
    ExtraArgs=extra_args,
```

```
    Callback=transfer_callback,
```

```
)
```

```
return transfer_callback.thread_info
```

```
def upload_with_high_threshold(local_file_path, bucket_name, object_key,
    file_size_mb):
```

```
    """
```

Upload a file from a local folder to an Amazon S3 bucket, setting a multipart threshold larger than the size of the file.

Setting a multipart threshold larger than the size of the file results in the transfer manager sending the file as a standard upload instead of a multipart upload.

```
    """
```

```
transfer_callback = TransferCallback(file_size_mb)
```

```
config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
```

```
s3.Bucket(bucket_name).upload_file(
```

```
    local_file_path, object_key, Config=config, Callback=transfer_callback
```

```
)
```

```
return transfer_callback.thread_info
```

```
def upload_with_sse(
```

```
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
```

```
):
```

```
    """
```

Upload a file from a local folder to an Amazon S3 bucket, adding server-side encryption with customer-provided encryption keys to the object.

```
When this kind of encryption is specified, Amazon S3 encrypts the object
at rest and allows downloads only when the expected encryption key is
provided in the download request.
"""

transfer_callback = TransferCallback(file_size_mb)
if sse_key:
    extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
sse_key}
else:
    extra_args = None
s3.Bucket(bucket_name).upload_file(
    local_file_path, object_key, ExtraArgs=extra_args,
    Callback=transfer_callback
)
return transfer_callback.thread_info

def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

```
def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

Demonstrate the transfer manager functions and report results.

```
import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")

class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
```

```

        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
files? "
            )
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
"demonstration files? "
            )
            try:
                self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
            except ParamValidationError as err:
                print(err)
                self.demo_bucket = None
            except ClientError as err:
                print(err)
                print(
                    f"Either {self.demo_bucket} doesn't exist or you don't "
                    f"have access to it."
                )
                self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.

```



```

    Ask the user if they want to run this specific demonstration.
    If they say yes, create a file on the local path, upload it
    using the specified upload function, then download it using the
    specified download function.
    """
    if download_args is None:
        download_args = {}
    if upload_args is None:
        upload_args = {}
    question = question.format(self.file_size_mb)
    answer = input(f"{question} (y/n)")
    if answer.lower() == "y":
        local_file_path, object_key, download_file_path =
self._create_demo_file()

        file_transfer.TransferConfig = self._config_wrapper(
            TransferConfig, CONFIG_ATTRS
        )
        self._report_transfer_params(
            "Uploading", local_file_path, object_key, **upload_args
        )
        start_time = time.perf_counter()
        thread_info = upload_func(
            local_file_path,
            self.demo_bucket,
            object_key,
            self.file_size_mb,
            **upload_args,
        )
        end_time = time.perf_counter()
        self._report_transfer_result(thread_info, end_time - start_time)

        file_transfer.TransferConfig = self._config_wrapper(
            TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
        )
        self._report_transfer_params(
            "Downloading", object_key, download_file_path, **download_args
        )
        start_time = time.perf_counter()
        thread_info = download_func(
            self.demo_bucket,
            object_key,
            download_file_path,
            self.file_size_mb,

```

```

        **download_args,
    )
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

def last_name_set(self):
    """Get the name set used for the last demo."""
    return self._chore_list[-1]

def cleanup(self):
    """
    Remove files from the demo folder, and uploaded objects from the
    Amazon S3 bucket.
    """
    print("-" * self._terminal_width)
    for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
        print(f"Removing {local_file_path}")
        try:
            os.remove(local_file_path)
        except FileNotFoundError as err:
            print(err)

        print(f"Removing {downloaded_file_path}")
        try:
            os.remove(downloaded_file_path)
        except FileNotFoundError as err:
            print(err)

        if self.demo_bucket:
            print(f"Removing {self.demo_bucket}:{s3_object_key}")
            try:
self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
                except ClientError as err:
                    print(err)

def _setup_platform_specific(self):
    """Set up platform-specific command used to create a large file."""
    if platform.system() == "Windows":
        self._create_file_cmd = "fsutil file createnew {} {}"
        self._size_multiplier = MB
    elif platform.system() == "Linux" or platform.system() == "Darwin":

```

```

        self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={MB}
count={{}}"
        self._size_multiplier = 1
    else:
        raise EnvironmentError(
            f"Demo of platform {platform.system()} isn't supported."
        )

def _create_demo_file(self):
    """
    Create a file in the demo folder specified by the user. Store the local
    path, object name, and download path for later cleanup.

    Only the local file is created by this method. The Amazon S3 object and
    download file are created later during the demonstration.

    Returns:
    A tuple that contains the local file path, object name, and download
    file path.
    """
    file_name_template = "TestFile{}-{}.demo"
    local_suffix = "local"
    object_suffix = "s3object"
    download_suffix = "downloaded"
    file_tag = len(self._chore_list) + 1

    local_file_path = os.path.join(
        self.demo_folder, file_name_template.format(file_tag, local_suffix)
    )

    s3_object_key = file_name_template.format(file_tag, object_suffix)

    downloaded_file_path = os.path.join(
        self.demo_folder, file_name_template.format(file_tag,
download_suffix)
    )

    filled_cmd = self._create_file_cmd.format(
        local_file_path, self.file_size_mb * self._size_multiplier
    )

    print(
        f"Creating file of size {self.file_size_mb} MB "
        f"in {self.demo_folder} by running:"
    )

```

```

    )
    print(f"{'':4}{filled_cmd}")
    os.system(filled_cmd)

    chore = (local_file_path, s3_object_key, downloaded_file_path)
    self._chore_list.append(chore)
    return chore

def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
    """Report configuration and extra arguments used for a file transfer."""
    print("-" * self._terminal_width)
    print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
    if kwargs:
        print("With extra args:")
        for arg, value in kwargs.items():
            print(f"{'':4}{arg:<20}: {value}'")

    @staticmethod
    def ask_user(question):
        """
        Ask the user a yes or no question.

        Returns:
        True when the user answers 'y' or 'Y'; otherwise, False.
        """
        answer = input(f"{question} (y/n) ")
        return answer.lower() == "y"

    @staticmethod
    def _config_wrapper(func, config_attrs):
        def wrapper(*args, **kwargs):
            config = func(*args, **kwargs)
            print("With configuration:")
            for attr in config_attrs:
                print(f"{'':4}{attr:<20}: {getattr(config, attr)}'")
            return config

        return wrapper

    @staticmethod
    def _report_transfer_result(thread_info, elapsed):
        """Report the result of a transfer, including per-thread data."""
        print(f"\nUsed {len(thread_info)} threads.")
        for ident, byte_count in thread_info.items():

```

```
        print(f"{'':4}Thread {ident} copied {byte_count} bytes.")
        print(f"Your transfer took {elapsed:.2f} seconds.")

def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
        }
    )
```

```

    },
)

# Upload using server-side encryption with customer-provided
# encryption keys.
# Generate a 256-bit key from a passphrase.
sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
demo_manager.demo(
    "Do you want to upload and download a {} MB file using "
    "server-side encryption?",
    file_transfer.upload_with_sse,
    file_transfer.download_with_sse,
    upload_args={"sse_key": sse_key},
    download_args={"sse_key": sse_key},
)

# Download without specifying an encryption key to show that the
# encryption key must be included to download an encrypted object.
if demo_manager.ask_user(
    "Do you want to try to download the encrypted "
    "object without sending the required key?"
):
    try:
        _, object_key, download_file_path = demo_manager.last_name_set()
        file_transfer.download_with_default_configuration(
            demo_manager.demo_bucket,
            object_key,
            download_file_path,
            demo_manager.file_size_mb,
        )
    except ClientError as err:
        print(
            "Got expected error when trying to download an encrypted "
            "object without specifying encryption info:"
        )
        print(f"{'':4}{err}")

# Remove all created and downloaded files, remove all objects from
# S3 storage.
if demo_manager.ask_user(
    "Demonstration complete. Do you want to remove local files " "and S3
objects?"
):
    demo_manager.cleanup()

```

```
if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
    get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_code_examples::error::S3ExampleError;
use std::process;
```

```

use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}

async fn run_example() -> Result<(), S3ExampleError> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;

    let key = "sample.txt".to_string();
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)

```



```

        .take(256)
        .map(char::from)
        .collect();
    let return_string: String = "\n".to_string();
    file.write_all(rand_string.as_ref())
        .expect("Error writing to file.");
    file.write_all(return_string.as_ref())
        .expect("Error writing to file.");
}

let path = Path::new(&key);
let file_size = tokio::fs::metadata(path)
    .await
    .expect("it exists I swear")
    .len();

let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
let mut size_of_last_chunk = file_size % CHUNK_SIZE;
if size_of_last_chunk == 0 {
    size_of_last_chunk = CHUNK_SIZE;
    chunk_count -= 1;
}

if file_size == 0 {
    return Err(S3ExampleError::new("Bad file size."));
}
if chunk_count > MAX_CHUNKS {
    return Err(S3ExampleError::new(
        "Too many chunks! Try increasing your chunk size.",
    ));
}

let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))

```

```

        .build()
        .await
        .unwrap();

// Chunk index needs to start at 0, but part numbers start at 1.
let part_number = (chunk_index as i32) + 1;
let upload_part_res = client
    .upload_part()
    .key(&key)
    .bucket(&bucket_name)
    .upload_id(upload_id)
    .body(stream)
    .part_number(part_number)
    .send()
    .await?;

upload_parts.push(
    CompletedPart::builder()
        .e_tag(upload_part_res.e_tag.unwrap_or_default())
        .part_number(part_number)
        .build(),
);
}

// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;

let data: GetObjectOutput =
    s3_code_examples::download_object(&client, &bucket_name, &key).await?;
let data_length: u64 = data
    .content_length()
    .unwrap_or_default()

```

```
        .try_into()
        .unwrap();
    if file.metadata().unwrap().len() == data_length {
        println!("Data lengths match.");
    } else {
        println!("The data was not the same size!");
    }

    s3_code_examples::clear_bucket(&client, &bucket_name)
        .await
        .expect("Error emptying bucket.");
    s3_code_examples::delete_bucket(&client, &bucket_name)
        .await
        .expect("Error deleting bucket.");

    Ok(())
}
```

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Upload a stream of unknown size to an Amazon S3 object using an Amazon SDK

The following code examples show how to upload a stream of unknown size to an Amazon S3 object.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Use the [Amazon CRT-based S3 Client](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsync {
    private static final Logger logger =
        LoggerFactory.getLogger(PutObjectFromStreamAsync.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsync example = new PutObjectFromStreamAsync();
            S3AsyncClient s3AsyncClientCrt = S3AsyncClient.crtCreate();
            PutObjectResponse putObjectResponse =
example.putObjectFromStreamCrt(s3AsyncClientCrt, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s3CrtAsyncClient - To upload content from a stream of unknown
     size, use can the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
    */
}
```

```

    * @param key - The name of the object.
    * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
    Returns metadata pertaining to the put object operation.
    */
    public PutObjectResponse putObjectFromStreamCrt(S3AsyncClient
s3CrtAsyncClient, String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
        characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
        InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
        thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
        length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
null, executor);

        CompletableFuture<PutObjectResponse> responseFuture =
            s3CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
        response.
        logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        executor.shutdown();
        return response;
    }
}

```

Use the standard [asynchronous S3 client with multipart upload enabled](#).

```

import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;

```

```
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsyncMp {
    private static final Logger logger =
        LoggerFactory.getLogger(PutObjectFromStreamAsyncMp.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsyncMp example = new
            PutObjectFromStreamAsyncMp();
            S3AsyncClient s3AsyncClientMp =
            S3AsyncClient.builder().multipartEnabled(true).build();
            PutObjectResponse putObjectResponse =
            example.putObjectFromStreamMp(s3AsyncClientMp, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s3AsyncClientMp - To upload content from a stream of unknown size,
     use can the S3 asynchronous client with multipart enabled.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     */
}
```

```

    * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
    Returns metadata pertaining to the put object operation.
    */
    public PutObjectResponse putObjectFromStreamMp(S3AsyncClient s3AsyncClientMp,
    String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
        characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
        randomString.length());
        InputStream inputStream = new
        ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
        thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
        length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
        null, executor);

        CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClientMp.putObject(r -> r.bucket(bucketName).key(key),
        body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
        response.
        logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        executor.shutdown();
        return response;
    }
}

```

Use the [Amazon S3 Transfer Manager](#).

```

import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;

```

```
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class UploadStream {
    private static final Logger logger =
        LoggerFactory.getLogger(UploadStream.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            UploadStream example = new UploadStream();
            CompletedUpload completedUpload =
example.uploadStream(S3TransferManager.create(), bucketName, key);
            logger.info("Object {} etag: {}", key,
completedUpload.response().eTag());
            logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param transferManager - To upload content from a stream of unknown size,
you can use the S3TransferManager based on the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
result of the completed upload.
     */
}
```



```
public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

    // AsyncExampleUtils.randomString() returns a random string up to 100
    characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
    InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());

    // Executor required to handle reading from the InputStream on a separate
    thread so the main upload is not blocked.
    ExecutorService executor = Executors.newSingleThreadExecutor();
    // Specify `null` for the content length when you don't know the content
    length.
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
null, executor);

    Upload upload = transferManager.upload(builder -> builder
        .requestBody(body)
        .putObjectRequest(req -> req.bucket(bucketName).key(key))
        .build());

    CompletedUpload completedUpload = upload.completionFuture().join();
    executor.shutdown();
    return completedUpload;
}
}
```

Swift

SDK for Swift

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams

/// Upload a file to the specified bucket.
///
/// - Parameters:
///   - bucket: The Amazon S3 bucket name to store the file into.
///   - key: The name (or path) of the file to upload to in the `bucket`.
///   - sourcePath: The pathname on the local filesystem of the file to
///     upload.
func uploadFile(sourcePath: String, bucket: String, key: String?) async
throws {
    let fileURL: URL = URL(fileURLWithPath: sourcePath)
    let fileName: String

    // If no key was provided, use the last component of the filename.

    if key == nil {
        fileName = fileURL.lastPathComponent
    } else {
        fileName = key!
    }

    let s3Client = try await S3Client()

    // Create a FileHandle for the source file.

    let fileHandle = FileHandle(forReadingAtPath: sourcePath)
    guard let fileHandle = fileHandle else {
        throw TransferError.readError
    }

    // Create a byte stream to retrieve the file's contents. This uses the
    // Smithy FileStream and ByteStream types.

    let stream = FileStream(fileHandle: fileHandle)
    let body = ByteStream.stream(stream)
```

```
// Create a `PutObjectInput` with the ByteStream as the body of the
// request's data. The AWS SDK for Swift will handle sending the
// entire file in chunks, regardless of its size.

let putInput = PutObjectInput(
    body: body,
    bucket: bucket,
    key: fileName
)

do {
    _ = try await s3Client.putObject(input: putInput)
} catch {
    throw TransferError.uploadError("Error uploading the file: \(error)")
}

print("File uploaded to \(fileURL.path).")
}
```

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use checksums to work with an Amazon S3 object using an Amazon SDK

The following code example shows how to use checksums to work with an Amazon S3 object.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

The code examples use a subset of the following imports.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Specify a checksum algorithm for the `putObject` method when you [build the PutObjectRequest](#).

```
public void putObjectWithChecksum(String bucketName, String key) {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
```

```
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),  
        RequestBody.fromString("This is a test"));  
    }
```

Verify the checksum for the `getObject` method when you [build the `GetObjectRequest`](#).

```
public GetObjectResponse getObjectWithChecksum(String bucketName, String key)  
{  
    return s3Client.getObject(b -> b  
        .bucket(bucketName)  
        .key(key)  
        .checksumMode(ChecksumMode.ENABLED))  
        .response();  
}
```

Pre-calculate a checksum for the `putObject` method when you [build the `PutObjectRequest`](#).

```
public void putObjectWithPrecalculatedChecksum(String bucketName, String key,  
String filePath) {  
    String checksum = calculateChecksum(filePath, "SHA-256");  
  
    s3Client.putObject((b -> b  
        .bucket(bucketName)  
        .key(key)  
        .checksumSHA256(checksum)),  
        RequestBody.fromFile(Paths.get(filePath)));  
}
```

Use the [S3 Transfer Manager](#) on top of the [Amazon CRT-based S3 client](#) to transparently perform a multipart upload when the size of the content exceeds a threshold. The default threshold size is 8 MB.

You can specify a checksum algorithm for the SDK to use. By default, the SDK uses the CRC32 algorithm.

```
public void multipartUploadWithChecksumTm(String bucketName, String key,  
String filePath) {  
    S3TransferManager transferManager = S3TransferManager.create();
```

```

        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.SHA1))
            .source(Paths.get(filePath))
            .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }

```

Use the [S3Client API](#) or (S3AsyncClient API) to perform a multipart upload. If you specify an additional checksum, you must specify the algorithm to use on the initiation of the upload. You must also specify the algorithm for each part request and provide the checksum calculated for each part after it is uploaded.

```

    public void multipartUploadWithChecksumS3Client(String bucketName, String
    key, String filePath) {
        ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
        s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

```

```

        bb.flip(); // Swap position and limit before reading from the
buffer.

        UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .checksumAlgorithm(algorithm) // Checksum specified on each
part.

            .partNumber(partNumber)
            .build();

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNumber)
            .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
            .eTag(partResponse.eTag())
            .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    System.err.println(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

    .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}

```

- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.
 - [CompleteMultipartUpload](#)

- [CreateMultipartUpload](#)
- [UploadPart](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Work with Amazon S3 object integrity features using an Amazon SDK

The following code example shows how to work with S3 object integrity features.

C++

SDK for C++

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Run an interactive scenario demonstrating Amazon S3 object integrity features.

```
#!/ Routine which runs the S3 object integrity workflow.
/*!
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::s3ObjectIntegrityWorkflow(
    const Aws::S3::S3ClientConfiguration &clientConfiguration) {

    /*
     * Create a large file to be used for multipart uploads.
     */
    if (!createLargeFileIfNotExists()) {
        std::cerr << "Workflow exiting because large file creation failed." <<
std::endl;
        return false;
    }

    Aws::String bucketName = TEST_BUCKET_PREFIX;
```



```

    bucketName += Aws::Utils::UUID::RandomUUID();
    bucketName = Aws::Utils::StringUtils::ToLower(bucketName.c_str());

    bucketName.resize(std::min(bucketName.size(), MAX_BUCKET_NAME_LENGTH));

    introductoryExplanations(bucketName);

    if (!AwsDoc::S3::createBucket(bucketName, clientConfiguration)) {
        std::cerr << "Workflow exiting because bucket creation failed." <<
std::endl;
        return false;
    }

    Aws::S3::S3ClientConfiguration s3ClientConfiguration(clientConfiguration);
    std::shared_ptr<Aws::S3::S3Client> client =
Aws::MakeShared<Aws::S3::S3Client>("S3Client", s3ClientConfiguration);

    printAsterisksLine();
    std::cout << "Choose from one of the following checksum algorithms."
        << std::endl;

    for (HASH_METHOD hashMethod = DEFAULT; hashMethod <= SHA256; ++hashMethod) {
        std::cout << "  " << hashMethod << " - " <<
stringForHashMethod(hashMethod)
        << std::endl;
    }

    HASH_METHOD chosenHashMethod = askQuestionForIntRange("Enter an index: ",
DEFAULT,
                                SHA256);

    gUseCalculatedChecksum = !askYesNoQuestion(
        "Let the SDK calculate the checksum for you? (y/n) ");

    printAsterisksLine();

    std::cout << "The workflow will now upload a file using PutObject."
        << std::endl;
    std::cout << "Object integrity will be verified using the "
        << stringForHashMethod(chosenHashMethod) << " algorithm."
        << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout

```

```

        << "A checksum computed by this workflow will be used for object
integrity verification,"
        << std::endl;
        std::cout << "except for the TransferManager upload." << std::endl;
    } else {
        std::cout
            << "A checksum computed by the SDK will be used for object
integrity verification."
            << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();

    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                     TEST_FILE,
                                     std::ios_base::in |
                                     std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Hasher hasher;
    HASH_METHOD putObjectHashMethod = chosenHashMethod;
    if (putObjectHashMethod == DEFAULT) {
        putObjectHashMethod = MD5; // MD5 is the default hash method for
PutObject.

        std::cout << "The default checksum algorithm for PutObject is "
            << stringForHashMethod(putObjectHashMethod)
            << std::endl;
    }

    // Demonstrate in code how the hash is computed.
    if (!hasher.calculateObjectHash(*inputData, putObjectHashMethod)) {
        std::cerr << "Error calculating hash for file " << TEST_FILE <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

```

```

    Aws::String key = stringForHashMethod(putObjectHashMethod);
    key += "_";
    key += TEST_FILE_KEY;
    Aws::String localHash = hasher.getBase64HashString();

    // Upload the object with PutObject
    if (!putObjectWithHash(bucketName, key, localHash, putObjectHashMethod,
                          inputData, chosenHashMethod == DEFAULT,
                          *client)) {
        std::cerr << "Error putting file " << TEST_FILE << " to bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Aws::String retrievedHash;
    if (!retrieveObjectHash(bucketName, key,
                          putObjectHashMethod, retrievedHash,
                          nullptr, *client)) {
        std::cerr << "Error getting file " << TEST_FILE << " from bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    explainPutObjectResults();
    verifyHashingResults(retrievedHash, hasher,
                        "PutObject upload", putObjectHashMethod);

    printAsterisksLine();
    pressEnterToContinue();

    key = "tr_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    introductoryTransferManagerUploadExplanations(key);

    HASH_METHOD transferManagerHashMethod = chosenHashMethod;
    if (transferManagerHashMethod == DEFAULT) {
        transferManagerHashMethod = CRC32; // The default hash method for the
        TransferManager is CRC32.

        std::cout << "The default checksum algorithm for TransferManager is "

```

```

        << stringForHashMethod(transferManagerHashMethod)
        << std::endl;
    }

    // Upload the large file using the transfer manager.
    if (!doTransferManagerUpload(bucketName, key, transferManagerHashMethod,
        chosenHashMethod == DEFAULT,
                                client)) {
        std::cerr << "Exiting because of an error in doTransferManagerUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    std::vector<Aws::String> retrievedTransferManagerPartHashes;
    Aws::String retrievedTransferManagerFinalHash;

    // Retrieve all the hashes for the TransferManager upload.
    if (!retrieveObjectHash(bucketName, key,
                            transferManagerHashMethod,
                            retrievedTransferManagerFinalHash,
                            &retrievedTransferManagerPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
TransferManager." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    AwsDoc::S3::Hasher locallyCalculatedFinalHash;
    std::vector<Aws::String> locallyCalculatedPartHashes;

    // Calculate the hashes locally to demonstrate how TransferManager hashes are
    computed.
    if (!calculatePartHashesForFile(transferManagerHashMethod,
        MULTI_PART_TEST_FILE,
                                UPLOAD_BUFFER_SIZE,
                                locallyCalculatedFinalHash,
                                locallyCalculatedPartHashes)) {
        std::cerr << "Exiting because of an error in calculatePartHashesForFile."
<< std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

```

```

verifyHashingResults(retrievedTransferManagerFinalHash,
                    locallyCalculatedFinalHash, "TransferManager upload",
                    transferManagerHashMethod,
                    retrievedTransferManagerPartHashes,
                    locallyCalculatedPartHashes);

printAsterisksLine();

key = "mp_";
key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

multiPartUploadExplanations(key, chosenHashMethod);

pressEnterToContinue();

std::shared_ptr<Aws::IOStream> largeFileInputData =
    Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                MULTI_PART_TEST_FILE,
                                std::ios_base::in |
                                std::ios_base::binary);

if (!largeFileInputData->good()) {
    std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

HASH_METHOD multipartUploadHashMethod = chosenHashMethod;
if (multipartUploadHashMethod == DEFAULT) {
    multipartUploadHashMethod = MD5; // The default hash method for
multipart uploads is MD5.

    std::cout << "The default checksum algorithm for multipart upload is "
              << stringForHashMethod(putObjectHashMethod)
              << std::endl;
}

AwsDoc::S3::Hasher hashData;
std::vector<Aws::String> partHashes;

if (!doMultipartUpload(bucketName, key,
                      multipartUploadHashMethod,
                      largeFileInputData, chosenHashMethod == DEFAULT,
                      hashData,

```

```

        partHashes,
        *client)) {
    std::cerr << "Exiting because of an error in doMultipartUpload." <<
std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

std::cout << "Finished multipart upload of with hash method " <<
    stringForHashMethod(multipartUploadHashMethod) << std::endl;

std::cout << "Now we will retrieve the checksums from the server." <<
std::endl;

retrievedHash.clear();
std::vector<Aws::String> retrievedPartHashes;
if (!retrieveObjectHash(bucketName, key,
                        multipartUploadHashMethod,
                        retrievedHash, &retrievedPartHashes, *client)) {
    std::cerr << "Exiting because of an error in retrieveObjectHash for
multipart." << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

verifyHashingResults(retrievedHash, hashData, "MultiPart upload",
                    multipartUploadHashMethod,
                    retrievedPartHashes, partHashes);

printAsterisksLine();

if (askYesNoQuestion("Would you like to delete the resources created in this
workflow? (y/n)")) {
    return cleanUp(bucketName, clientConfiguration);
} else {
    std::cout << "The bucket " << bucketName << " was not deleted." <<
std::endl;
    return true;
}
}

//! Routine which uploads an object to an S3 bucket with different object
integrity hashing methods.
/*!
```

```

    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashData: The hash value that will be associated with the uploaded
object.
    \param hashMethod: The hashing algorithm to use when calculating the hash
value.
    \param body: The data content of the object being uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default hash
method or the one specified in the hashMethod parameter.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::putObjectWithHash(const Aws::String &bucket, const Aws::String
&key,
                                const Aws::String &hashData,
                                AwsDoc::S3::HASH_METHOD hashMethod,
                                const std::shared_ptr<Aws::IOStream> &body,
                                bool useDefaultHashMethod,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {
            request.SetChecksumAlgorithm(getChecksumAlgorithmForHashMethod(hashMethod));
        }
    }

    if (gUseCalculatedChecksum) {
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                request.SetContentMD5(hashData);
                break;
            case AwsDoc::S3::SHA1:
                request.SetChecksumSHA1(hashData);
                break;
            case AwsDoc::S3::SHA256:
                request.SetChecksumSHA256(hashData);
                break;
            case AwsDoc::S3::CRC32:
                request.SetChecksumCRC32(hashData);
                break;
            case AwsDoc::S3::CRC32C:

```

```

        request.SetChecksumCRC32C(hashData);
        break;
    default:
        std::cerr << "Unknown hash method." << std::endl;
        return false;
    }
}

request.SetBody(body);
Aws::S3::Model::PutObjectOutcome outcome = client.PutObject(request);
body->seekg(0, body->beg);
if (outcome.IsSuccess()) {
    std::cout << "Object successfully uploaded." << std::endl;
} else {
    std::cerr << "Error uploading object." <<
        outcome.GetError().GetMessage() << std::endl;
}
return outcome.IsSuccess();
}

// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object is stored.
    \param key: The unique identifier (key) of the object within the S3 bucket.
    \param hashMethod: The hashing algorithm used to calculate the hash value of
the object.
    \param[out] hashData: The retrieved hash.
    \param[out] partHashes: The part hashes if available.
    \param client: The S3 client instance used to retrieve the object.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
&key,

                                AwsDoc::S3::HASH_METHOD hashMethod,
                                Aws::String &hashData,
                                std::vector<Aws::String> *partHashes,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
    }
}

```



```

        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        hashData = result.GetETag();
    } else {
        std::cerr << "Error retrieving object etag attributes." <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
} else { // hashMethod != MD5
    Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
    attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
    request.SetObjectAttributes(attributes);

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                break; // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
            case AwsDoc::S3::MD5:
                break; // MD5 is not supported.
#pragma clang diagnostic pop
            case AwsDoc::S3::SHA1:
                hashData = result.GetChecksum().GetChecksumSHA1();
                break;
            case AwsDoc::S3::SHA256:
                hashData = result.GetChecksum().GetChecksumSHA256();
                break;
            case AwsDoc::S3::CRC32:
                hashData = result.GetChecksum().GetChecksumCRC32();
                break;
            case AwsDoc::S3::CRC32C:
                hashData = result.GetChecksum().GetChecksumCRC32C();

```

```

        break;
    default:
        std::cerr << "Unknown hash method." << std::endl;
        return false;
    }
} else {
    std::cerr << "Error retrieving object checksum attributes." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

if (nullptr != partHashes) {
    attributes.clear();
    attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
    request.SetObjectAttributes(attributes);
    outcome = client.GetObjectAttributes(request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
        for (const Aws::S3::Model::ObjectPart &part: parts) {
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                    break;
                case AwsDoc::S3::MD5: // MD5 is not supported.
                    break;
                case AwsDoc::S3::SHA1:
                    partHashes->push_back(part.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:
                    partHashes->push_back(part.GetChecksumSHA256());
                    break;
                case AwsDoc::S3::CRC32:
                    partHashes->push_back(part.GetChecksumCRC32());
                    break;
                case AwsDoc::S3::CRC32C:
                    partHashes->push_back(part.GetChecksumCRC32C());
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        }
    }
}

```

```

        }
    } else {
        std::cerr << "Error retrieving object attributes for object
parts." <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

return true;
}

//! Verifies the hashing results between the retrieved and local hashes.
/*!
\param retrievedHash The hash value retrieved from the remote source.
\param localHash The hash value calculated locally.
\param uploadtype The type of upload (e.g., "multipart", "single-part").
\param hashMethod The hashing method used (e.g., MD5, SHA-256).
\param retrievedPartHashes (Optional) The list of hashes for the individual
parts retrieved from the remote source.
\param localPartHashes (Optional) The list of hashes for the individual parts
calculated locally.
*/
void AwsDoc::S3::verifyHashingResults(const Aws::String &retrievedHash,
                                     const Hasher &localHash,
                                     const Aws::String &uploadtype,
                                     HASH_METHOD hashMethod,
                                     const std::vector<Aws::String>
&retrievedPartHashes,
                                     const std::vector<Aws::String>
&localPartHashes) {
    std::cout << "For " << uploadtype << " retrieved hash is " << retrievedHash
<< std::endl;
    if (!retrievedPartHashes.empty()) {
        std::cout << retrievedPartHashes.size() << " part hash(es) were also
retrieved."
            << std::endl;
        for (auto &retrievedPartHash: retrievedPartHashes) {
            std::cout << " Part hash " << retrievedPartHash << std::endl;
        }
    }
    Aws::String hashString;
    if (hashMethod == MD5) {

```

```

        hashString = localHash.getHexHashString();
        if (!localPartHashes.empty()) {
            hashString += "-" + std::to_string(localPartHashes.size());
        }
    } else {
        hashString = localHash.getBase64HashString();
    }

    bool allMatch = true;
    if (hashString != retrievedHash) {
        std::cerr << "For " << uploadtype << ", the main hashes do not match" <<
std::endl;
        std::cerr << "Local hash- '" << hashString << "'" << std::endl;
        std::cerr << "Remote hash - '" << retrievedHash << "'" << std::endl;
        allMatch = false;
    }

    if (hashMethod != MD5) {
        if (localPartHashes.size() != retrievedPartHashes.size()) {
            std::cerr << "For " << uploadtype << ", the number of part hashes do
not match" << std::endl;
            std::cerr << "Local number of hashes- '" << localPartHashes.size() <<
""
                << std::endl;
            std::cerr << "Remote number of hashes - '"
                << retrievedPartHashes.size()
                << "'" << std::endl;
        }

        for (int i = 0; i < localPartHashes.size(); ++i) {
            if (localPartHashes[i] != retrievedPartHashes[i]) {
                std::cerr << "For " << uploadtype << ", the part hashes do not
match for part " << i + 1
                    << "." << std::endl;
                std::cerr << "Local hash- '" << localPartHashes[i] << "'"
                    << std::endl;
                std::cerr << "Remote hash - '" << retrievedPartHashes[i] << "'"
                    << std::endl;
                allMatch = false;
            }
        }
    }

    if (allMatch) {

```

```

        std::cout << "For " << uploadtype << ", locally and remotely calculated
        hashes all match!" << std::endl;
    }

}

static void transferManagerErrorCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
                                         Aws::Transfer::TransferHandle> &,
                                         const
                                         Aws::Client::AWSError<Aws::S3::S3Errors> &err) {
    std::cerr << "Error during transfer: '" << err.GetMessage() << "'" <<
    std::endl;
}

static void transferManagerStatusCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
                                         Aws::Transfer::TransferHandle> &handle) {
    if (handle->GetStatus() == Aws::Transfer::TransferStatus::IN_PROGRESS) {
        std::cout << "Bytes transferred: " << handle->GetBytesTransferred() <<
        std::endl;
    }
}

}

//! Routine which uploads an object to an S3 bucket using the AWS C++ SDK's
    Transfer Manager.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash
    value.
    \param useDefaultHashMethod: A flag indicating whether to use the default hash
    method or the one specified in the hashMethod parameter.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool
AwsDoc::S3::doTransferManagerUpload(const Aws::String &bucket, const Aws::String
&key,
                                     AwsDoc::S3::HASH_METHOD hashMethod,
                                     bool useDefaultHashMethod,
                                     const std::shared_ptr<Aws::S3::S3Client>
&client) {

```

```

    std::shared_ptr<Aws::Utils::Threading::PooledThreadExecutor> executor =
    Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
        "executor", 25);
    Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
    transfer_config.s3Client = client;
    transfer_config.bufferSize = UPLOAD_BUFFER_SIZE;
    if (!useDefaultHashMethod) {
        if (hashMethod == MD5) {
            transfer_config.computeContentMD5 = true;
        } else {
            transfer_config.checksumAlgorithm =
getChecksumAlgorithmForHashMethod(
                hashMethod);
        }
    }
    transfer_config.errorCallback = transferManagerErrorCallback;
    transfer_config.transferStatusUpdatedCallback =
transferManagerStatusCallback;

    std::shared_ptr<Aws::Transfer::TransferManager> transfer_manager =
    Aws::Transfer::TransferManager::Create(
        transfer_config);

    std::cout << "Uploading the file..." << std::endl;
    std::shared_ptr<Aws::Transfer::TransferHandle> uploadHandle =
transfer_manager->UploadFile(MULTI_PART_TEST_FILE,

        bucket, key,

        "text/plain",

        Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success =
        uploadHandle->GetStatus() ==
    Aws::Transfer::TransferStatus::COMPLETED;
    if (!success) {
        Aws::Client::AWSError<Aws::S3::S3Errors> err = uploadHandle-
>GetLastError();
        std::cerr << "File upload failed:  " << err.GetMessage() << std::endl;
    }

    return success;
}

```

```

//! Routine which calculates the hash values for each part of a file being
//! uploaded to an S3 bucket.
/*!
    \param hashMethod: The hashing algorithm to use when calculating the hash
    values.
    \param fileName: The path to the file for which the part hashes will be
    calculated.
    \param bufferSize: The size of the buffer to use when reading the file.
    \param[out] hashDataResult: The Hasher object that will store the concatenated
    hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
    for each part of the file.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::calculatePartHashesForFile(AwsDoc::S3::HASH_METHOD hashMethod,
                                             const Aws::String &fileName,
                                             size_t bufferSize,
                                             AwsDoc::S3::Hasher &hashDataResult,
                                             std::vector<Aws::String> &partHashes)
{
    std::ifstream fileStream(fileName.c_str(), std::ifstream::binary);
    fileStream.seekg(0, std::ifstream::end);
    size_t objectSize = fileStream.tellg();
    fileStream.seekg(0, std::ifstream::beg);
    std::vector<unsigned char> totalHashBuffer;
    size_t uploadedBytes = 0;

    while (uploadedBytes < objectSize) {
        std::vector<unsigned char> buffer(bufferSize);
        std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(), objectSize -
uploadedBytes));
        fileStream.read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),

bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
            Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                           &preallocatedStreamBuf);

        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {

```

```

        std::cerr << "Error calculating hash." << std::endl;
        return false;
    }
    Aws::String base64HashString = hasher.getBase64HashString();
    partHashes.push_back(base64HashString);

    Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

    totalHashBuffer.insert(totalHashBuffer.end(),
        hashBuffer.GetUnderlyingData(),
        hashBuffer.GetUnderlyingData() +
        hashBuffer.GetLength());

    uploadedBytes += bytesToRead;
}

return hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod);
}

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
    &key,
                                Aws::S3::Model::ChecksumAlgorithm
checksumAlgorithm,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
        client.CreateMultipartUpload(request);

    Aws::String uploadID;

```



```

        if (outcome.IsSuccess()) {
            uploadID = outcome.GetResult().GetUploadId();
        } else {
            std::cerr << "Error creating multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
        }

        return uploadID;
    }

    //! Upload a part to an S3 bucket.
    /*!
        \param bucket: The name of the S3 bucket where the object will be uploaded.
        \param key: The unique identifier (key) for the object within the S3 bucket.
        \param uploadID: An upload ID string.
        \param partNumber:
        \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
        \param calculatedHash: A data integrity hash to set, depending on the
checksum algorithm,
                                ignored when it is an empty string.
        \param body: An shared_ptr IOSTream of the data to be uploaded.
        \param client: The S3 client instance used to perform the upload operation.
        \return UploadPartOutcome: The outcome.
    */

    Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
        &bucket,
                                                                const Aws::String &key,
                                                                const Aws::String
        &uploadID,
                                                                int partNumber,

        Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                                const Aws::String
        &calculatedHash,
                                                                const
        std::shared_ptr<Aws::IOStream> &body,
                                                                const Aws::S3::S3Client
        &client) {
        Aws::S3::Model::UploadPartRequest request;
        request.SetBucket(bucket);
        request.SetKey(key);
        request.SetUploadId(uploadID);
        request.SetPartNumber(partNumber);
    }

```

```

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
                                       const Aws::String &uploadID,
                                       const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

```

```

        request.SetUploadId(uploadID);

        Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

        if (outcome.IsSuccess()) {
            std::cout << "Multipart upload aborted." << std::endl;
        } else {
            std::cerr << "Error aborting multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
        }

        return outcome.IsSuccess();
    }

    /** Complete a multipart upload to an S3 bucket.
    */
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
    */
    Aws::S3::Model::CompleteMultipartUploadOutcome
    AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

        const Aws::String &key,

        const Aws::String &uploadID,

        const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

        const Aws::S3::S3Client &client) {
        Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
        completedMultipartUpload.SetParts(parts);

        Aws::S3::Model::CompleteMultipartUploadRequest request;
        request.SetBucket(bucket);
        request.SetKey(key);
        request.SetUploadId(uploadID);
        request.SetMultipartUpload(completedMultipartUpload);

        Aws::S3::Model::CompleteMultipartUploadOutcome outcome =

```

```

        client.CompleteMultipartUpload(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error completing multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
        }
        return outcome;
    }

    ///! Routine which performs a multi-part upload.
    /*!
        \param bucket: The name of the S3 bucket where the object will be uploaded.
        \param key: The unique identifier (key) for the object within the S3 bucket.
        \param hashMethod: The hashing algorithm to use when calculating the hash
value.
        \param ioStream: An IOStream for the data to be uploaded.
        \param useDefaultHashMethod: A flag indicating whether to use the default
hash method or the one specified in the hashMethod parameter.
        \param[out] hashDataResult: The Hasher object that will store the
concatenated hash value.
        \param[out] partHashes: The vector that will store the calculated hash values
for each part of the file.
        \param client: The S3 client instance used to perform the upload operation.
        \return bool: Function succeeded.
    */
    bool AwsDoc::S3::doMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
                                       AwsDoc::S3::HASH_METHOD hashMethod,
                                       const std::shared_ptr<Aws::IOStream>
&ioStream,

                                       bool useDefaultHashMethod,
                                       AwsDoc::S3::Hasher &hashDataResult,
                                       std::vector<Aws::String> &partHashes,
                                       const Aws::S3::S3Client &client) {

        // Get object size.
        ioStream->seekg(0, ioStream->end);
        size_t objectSize = ioStream->tellg();
        ioStream->seekg(0, ioStream->beg);

        Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm =
Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
        if (!useDefaultHashMethod) {
            if (hashMethod != MD5) {
                checksumAlgorithm = getChecksumAlgorithmForHashMethod(hashMethod);
            }
        }
    }

```

```

    }
}
Aws::String uploadID = createMultipartUpload(bucket, key, checksumAlgorithm,
client);
if (uploadID.empty()) {
    return false;
}

std::vector<unsigned char> totalHashBuffer;
bool uploadSucceeded = true;
std::streamsize uploadedBytes = 0;
int partNumber = 1;
Aws::Vector<Aws::S3::Model::CompletedPart> parts;
while (uploadedBytes < objectSize) {
    std::cout << "Uploading part " << partNumber << "." << std::endl;

    std::vector<unsigned char> buffer(UPLOAD_BUFFER_SIZE);
    std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(),
objectSize - uploadedBytes));
    ioStream->read((char *) buffer.data(), bytesToRead);
    Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),
bytesToRead);
    std::shared_ptr<Aws::IOStream> body =
        Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
            &preallocatedStreamBuf);

    Hasher hasher;
    if (!hasher.calculateObjectHash(*body, hashMethod)) {
        std::cerr << "Error calculating hash." << std::endl;
        uploadSucceeded = false;
        break;
    }

    Aws::String base64HashString = hasher.getBase64HashString();
    partHashes.push_back(base64HashString);

    Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

    totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),

```

```

        hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

    Aws::String calculatedHash;
    if (gUseCalculatedChecksum) {
        calculatedHash = base64HashString;
    }
    Aws::S3::Model::UploadPartOutcome uploadPartOutcome = uploadPart(bucket,
key, uploadID, partNumber,

checksumAlgorithm, base64HashString, body,

                                                                    client);

    if (uploadPartOutcome.IsSuccess()) {
        const Aws::S3::Model::UploadPartResult &uploadPartResult =
uploadPartOutcome.GetResult();
        Aws::S3::Model::CompletedPart completedPart;
        completedPart.SetETag(uploadPartResult.GetETag());
        completedPart.SetPartNumber(partNumber);
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                break; // Do nothing.
            case AwsDoc::S3::SHA1:

completedPart.SetChecksumSHA1(uploadPartResult.GetChecksumSHA1());
                break;
            case AwsDoc::S3::SHA256:

completedPart.SetChecksumSHA256(uploadPartResult.GetChecksumSHA256());
                break;
            case AwsDoc::S3::CRC32:

completedPart.SetChecksumCRC32(uploadPartResult.GetChecksumCRC32());
                break;
            case AwsDoc::S3::CRC32C:

completedPart.SetChecksumCRC32C(uploadPartResult.GetChecksumCRC32C());
                break;
            default:
                std::cerr << "Unhandled hash method for completedPart." <<
std::endl;
                break;
        }

        parts.push_back(completedPart);

```

```
    } else {
        std::cerr << "Error uploading part. " <<
            uploadPartOutcome.GetError().GetMessage() << std::endl;
        uploadSucceeded = false;
        break;
    }

    uploadedBytes += bytesToRead;
    partNumber++;
}

if (!uploadSucceeded) {
    abortMultipartUpload(bucket, key, uploadID, client);
    return false;
} else {

    Aws::S3::Model::CompleteMultipartUploadOutcome
    completeMultipartUploadOutcome = completeMultipartUpload(bucket,

        key,

        uploadID,

        parts,

        client);

    if (completeMultipartUploadOutcome.IsSuccess()) {
        std::cout << "Multipart upload completed." << std::endl;
        if (!hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod))
        {
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
    } else {
        std::cerr << "Error completing multipart upload." <<
            completeMultipartUploadOutcome.GetError().GetMessage()
            << std::endl;
    }

    return completeMultipartUploadOutcome.IsSuccess();
}
}
```

```

    //! Routine which retrieves the string for a HASH_METHOD constant.
    /*!
        \param: hashMethod: A HASH_METHOD constant.
        \return: String: A string description of the hash method.
    */
    Aws::String AwsDoc::S3::stringForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {
        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT:
                return "Default";
            case AwsDoc::S3::MD5:
                return "MD5";
            case AwsDoc::S3::SHA1:
                return "SHA1";
            case AwsDoc::S3::SHA256:
                return "SHA256";
            case AwsDoc::S3::CRC32:
                return "CRC32";
            case AwsDoc::S3::CRC32C:
                return "CRC32C";
            default:
                return "Unknown";
        }
    }

    //! Routine that returns the ChecksumAlgorithm for a HASH_METHOD constant.
    /*!
        \param: hashMethod: A HASH_METHOD constant.
        \return: ChecksumAlgorithm: The ChecksumAlgorithm enum.
    */
    Aws::S3::Model::ChecksumAlgorithm
    AwsDoc::S3::getChecksumAlgorithmForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod)
    {
        Aws::S3::Model::ChecksumAlgorithm result =
        Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT:
                std::cerr << "getChecksumAlgorithmForHashMethod- DEFAULT is not
                valid." << std::endl;
                break; // Default is not supported.
            case AwsDoc::S3::MD5:
                break; // Ignore MD5.
            case AwsDoc::S3::SHA1:
                result = Aws::S3::Model::ChecksumAlgorithm::SHA1;
                break;

```



```

        case AwsDoc::S3::SHA256:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA256;
            break;
        case AwsDoc::S3::CRC32:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32;
            break;
        case AwsDoc::S3::CRC32C:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32C;
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            break;
    }

    return result;
}

//! Routine which cleans up after the example is complete.
/*!
    \param bucket: The name of the S3 bucket where the object was uploaded.
    \param clientConfiguration: The client configuration for the S3 client.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::cleanUp(const Aws::String &bucketName,
                        const Aws::S3::S3ClientConfiguration
                        &clientConfiguration) {

    Aws::Vector<Aws::String> keysResult;
    bool result = true;
    if (AwsDoc::S3::listObjects(bucketName, keysResult, clientConfiguration)) {
        if (!keysResult.empty()) {
            result = AwsDoc::S3::deleteObjects(keysResult, bucketName,
                                                clientConfiguration);
        }
    } else {
        result = false;
    }

    return result && AwsDoc::S3::deleteBucket(bucketName, clientConfiguration);
}

//! Console interaction introducing the workflow.
/*!

```

```
\param bucketName: The name of the S3 bucket to use.
*/
void AwsDoc::S3::introductoryExplanations(const Aws::String &bucketName) {

    std::cout
        << "Welcome to the Amazon Simple Storage Service (Amazon S3) object
integrity workflow."
        << std::endl;
    printAsterisksLine();
    std::cout
        << "This workflow demonstrates how Amazon S3 uses checksum values to
verify the integrity of data\n";
    std::cout << "uploaded to Amazon S3 buckets" << std::endl;
    std::cout
        << "The AWS SDK for C++ automatically handles checksums.\n";
    std::cout
        << "By default it calculates a checksum that is uploaded with an
object.\n"
        << "The default checksum algorithm for PutObject and MultiPart upload
is an MD5 hash.\n"
        << "The default checksum algorithm for TransferManager uploads is a
CRC32 checksum."
        << std::endl;
    std::cout
        << "You can override the default behavior, requiring one of the
following checksums,\n";
    std::cout << "MD5, CRC32, CRC32C, SHA-1 or SHA-256." << std::endl;
    std::cout << "You can also set the checksum hash value, instead of letting
the SDK calculate the value."
        << std::endl;
    std::cout
        << "For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/checking-object-integrity.html."
        << std::endl;

    std::cout
        << "This workflow will locally compute checksums for files uploaded
to an Amazon S3 bucket,\n";
    std::cout << "even when the SDK also computes the checksum." << std::endl;
    std::cout
        << "This is done to provide demonstration code for how the checksums
are calculated."
        << std::endl;
```

```
        std::cout << "A bucket named '" << bucketName << "' will be created for the
        object uploads."
        << std::endl;
    }

    //! Console interaction which explains the PutObject results.
    /*
    */
    void AwsDoc::S3::explainPutObjectResults() {

        std::cout << "The upload was successful.\n";
        std::cout << "If the checksums had not matched, the upload would have
        failed."
        << std::endl;
        std::cout
        << "The checksums calculated by the server have been retrieved using
        the GetObjectAttributes."
        << std::endl;
        std::cout
        << "The locally calculated checksums have been verified against the
        retrieved checksums."
        << std::endl;
    }

    //! Console interaction explaining transfer manager uploads.
    /*
    \param objectKey: The key for the object being uploaded.
    */
    void AwsDoc::S3::introductoryTransferManagerUploadExplanations(
        const Aws::String &objectKey) {
        std::cout
        << "Now the workflow will demonstrate object integrity for
        TransferManager multi-part uploads."
        << std::endl;
        std::cout
        << "The AWS C++ SDK has a TransferManager class which simplifies
        multipart uploads."
        << std::endl;
        std::cout
        << "The following code lets the TransferManager handle much of the
        checksum configuration."
        << std::endl;

        std::cout << "An object with the key '" << objectKey
```

```

        << " will be uploaded by the TransferManager using a "
        << BUFFER_SIZE_IN_MEGABYTES << " MB buffer." << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout << "For TransferManager uploads, this demo always lets the SDK
calculate the hash value."
        << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();
}

//! Console interaction explaining multi-part uploads.
/*!
\param objectKey: The key for the object being uploaded.
\param chosenHashMethod: The hash method selected by the user.
*/
void AwsDoc::S3::multiPartUploadExplanations(const Aws::String &objectKey,
                                              HASH_METHOD chosenHashMethod) {

    std::cout
        << "Now we will provide an in-depth demonstration of multi-part
uploading by calling the multi-part upload APIs directly."
        << std::endl;
    std::cout << "These are the same APIs used by the TransferManager when
uploading large files."
        << std::endl;
    std::cout
        << "In the following code, the checksums are also calculated locally
and then compared."
        << std::endl;
    std::cout
        << "For multi-part uploads, a checksum is uploaded with each part.
The final checksum is a concatenation of"
        << std::endl;
    std::cout << "the checksums for each part." << std::endl;
    std::cout
        << "This is explained in the user guide, https://docs.aws.amazon.com/
AmazonS3/latest/userguide/checking-object-integrity.html,"
        << " in the section \"Using part-level checksums for multipart
uploads\"." << std::endl;

    std::cout << "Starting multipart upload of with hash method " <<
        stringForHashMethod(chosenHashMethod) << " uploading to with object
key\n"

```

```
        << "" << objectKey << ", " << std::endl;

    }

    //! Create a large file for doing multi-part uploads.
    /*!
    */
    bool AwsDoc::S3::createLargeFileIfNotExists() {
        // Generate a large file by writing this source file multiple times to a new
        // file.
        if (std::filesystem::exists(MULTI_PART_TEST_FILE)) {
            return true;
        }

        std::ofstream newFile(MULTI_PART_TEST_FILE, std::ios::out

                                | std::ios::binary);

        if (!newFile) {
            std::cerr << "createLargeFileIfNotExists- Error creating file " <<
MULTI_PART_TEST_FILE <<
                std::endl;
            return false;
        }

        std::ifstream input(TEST_FILE, std::ios::in

                                | std::ios::binary);

        if (!input) {
            std::cerr << "Error opening file " << TEST_FILE <<
                std::endl;
            return false;
        }
        std::stringstream buffer;
        buffer << input.rdbuf();

        input.close();

        while (newFile.tellp() < LARGE_FILE_SIZE && !newFile.bad()) {
            buffer.seekg(std::stringstream::beg);
            newFile << buffer.rdbuf();
        }

        newFile.close();
    }
```

```
    return true;  
}
```

- For API details, see the following topics in *Amazon SDK for C++ API Reference*.
 - [AbortMultipartUpload](#)
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [DeleteObject](#)
 - [GetObjectAttributes](#)
 - [PutObject](#)
 - [UploadPart](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Work with Amazon S3 versioned objects using an Amazon SDK

The following code example shows how to:

- Create a versioned S3 bucket.
- Get all versions of an object.
- Roll an object back to a previous version.
- Delete and restore a versioned object.
- Permanently delete all versions of an object.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create functions that wrap S3 actions.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
    lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                    configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
```

```

        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket

def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are

```



```

# at the end of the list even when they are interspersed in time.
versions = sorted(
    bucket.object_versions.filter(Prefix=object_key),
    key=attrgetter("last_modified"),
    reverse=True,
)

logger.debug(
    "Got versions:\n%s",
    "\n".join(
        [
            f"\t{version.version_id}, last modified {version.last_modified}"
            for version in versions
        ]
    ),
)

if version_id in [ver.version_id for ver in versions]:
    print(f"Rolling back to version {version_id}")
    for version in versions:
        if version.version_id != version_id:
            version.delete()
            print(f"Deleted version {version.version_id}")
        else:
            break

    print(f"Active version is now {bucket.Object(object_key).version_id}")
else:
    raise KeyError(
        f"{version_id} was not found in the list of versions for "
        f"{object_key}."
    )

def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
    marker.
    By removing the delete marker, we make the previous version the latest
    version
    and the object then presents as not deleted.
    """

```

Usage is shown in the `usage_demo_single_object` function at the end of this module.

```

:param bucket: The bucket that contains the object.
:param object_key: The object to revive.
"""
# Get the latest version for the object.
response = s3.meta.client.list_object_versions(
    Bucket=bucket.name, Prefix=object_key, MaxKeys=1
)

if "DeleteMarkers" in response:
    latest_version = response["DeleteMarkers"][0]
    if latest_version["IsLatest"]:
        logger.info(
            "Object %s was indeed deleted on %s. Let's revive it.",
            object_key,
            latest_version["LastModified"],
        )
        obj = bucket.Object(object_key)
        obj.Version(latest_version["VersionId"]).delete()
        logger.info(
            "Revived %s, active version is now %s with body '%s'",
            object_key,
            obj.version_id,
            obj.get()["Body"].read(),
        )
    else:
        logger.warning(
            "Delete marker is not the latest version for %s!", object_key
        )
elif "Versions" in response:
    logger.warning("Got an active version for %s, nothing to do.",
object_key)
else:
    logger.error("Couldn't get any version info for %s.", object_key)

def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

```

Usage is shown in the `usage_demo_single_object` function at the end of this module.

```
:param bucket: The bucket that contains the object.
:param object_key: The object to delete.
"""
try:
    bucket.object_versions.filter(Prefix=object_key).delete()
    logger.info("Permanently deleted all versions of object %s.", object_key)
except ClientError:
    logger.exception("Couldn't delete all versions of %s.", object_key)
    raise
```

Upload the stanza of a poem to a versioned object and perform a series of actions on it.

```
def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print(
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
    print("-" * width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
```

```

print("\nApplying some revisions to the stanza...")
obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))
obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))
obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
print(
    "The latest version of the stanza is now:",
    obj_stanza_1.get()["Body"].read().decode("utf-8"),
    sep="\n",
)

# Versions are returned in order, most recent first.
obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
print(
    "The version data of the stanza revisions:",
    *[
        f"    {version.version_id}, last modified {version.last_modified}"
        for version in obj_stanza_1_versions
    ],
    sep="\n",
)

# Rollback two versions.
print("\nRolling back two versions...")
rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
print(
    "The latest version of the stanza:",
    obj_stanza_1.get()["Body"].read().decode("utf-8"),
    sep="\n",
)

# Delete the stanza
print("\nDeleting the stanza...")
obj_stanza_1.delete()
try:
    obj_stanza_1.get()
except ClientError as error:
    if error.response["Error"]["Code"] == "NoSuchKey":
        print("The stanza is now deleted (as expected).")
    else:
        raise

```

```
# Revive the stanza
print("\nRestoring the stanza...")
revive_object(bucket, obj_stanza_1.key)
print(
    "The stanza is restored! The latest version is again:",
    obj_stanza_1.get()["Body"].read().decode("utf-8"),
    sep="\n",
)

# Permanently delete all versions of the object. This cannot be undone!
print("\nPermanently deleting all versions of the stanza...")
permanently_delete_object(bucket, obj_stanza_1.key)
obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
if len(list(obj_stanza_1_versions)) == 0:
    print("The stanza has been permanently deleted and now has no versions.")
else:
    print("Something went wrong. The stanza still exists!")

print(f"\nRemoving {bucket.name}...")
bucket.delete()
print(f"{bucket.name} deleted.")
print("Demo done!")
```

- For API details, see the following topics in *Amazon SDK for Python (Boto3) API Reference*.
 - [CreateBucket](#)
 - [DeleteObject](#)
 - [ListObjectVersions](#)
 - [PutBucketLifecycleConfiguration](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Serverless examples for Amazon S3

The following code examples show how to use Amazon S3 with Amazon SDKs.

Examples

- [Invoke a Lambda function from an Amazon S3 trigger](#)

Invoke a Lambda function from an Amazon S3 trigger

The following code examples show how to implement a Lambda function that receives an event triggered by uploading an object to an S3 bucket. The function retrieves the S3 bucket name and object key from the event parameter and calls the Amazon S3 API to retrieve and log the content type of the object.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an S3 event with Lambda using .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
// into a .NET class.
[assembly:
    LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer))]

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
```

```
public Function() : this(null)
{
}

internal Function(AmazonS3Client s3Client)
{
    _s3Client = s3Client ?? new AmazonS3Client();
}

public async Task<string> Handler(S3Event evt, ILambdaContext context)
{
    try
    {
        if (evt.Records.Count <= 0)
        {
            context.Logger.LogLine("Empty S3 Event received");
            return string.Empty;
        }

        var bucket = evt.Records[0].S3.Bucket.Name;
        var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

        context.Logger.LogLine($"Request is for {bucket} and {key}");

        var objectResult = await _s3Client.GetObjectAsync(bucket, key);

        context.Logger.LogLine($"Returning {objectResult.Key}");

        return objectResult.Key;
    }
    catch (Exception e)
    {
        context.Logger.LogLine($"Error processing request - {e.Message}");

        return string.Empty;
    }
}
}
```

Go

SDK for Go V2

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an S3 event with Lambda using Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "log"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Printf("failed to load default config: %s", err)
        return err
    }
    s3Client := s3.NewFromConfig(sdkConfig)

    for _, record := range s3Event.Records {
        bucket := record.S3.Bucket.Name
        key := record.S3.Object.URLDecodedKey
        headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
            Bucket: &bucket,
            Key:    &key,
        })
        if err != nil {
            log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
        }
    }
}
```



```
    return err
}
log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
*headOutput.ContentType)
}

return nil
}

func main() {
    lambda.Start(handler)
}
```

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an S3 event with Lambda using Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucket)
            .key(key)
            .build();
        return s3Client.headObject(headObjectRequest);
    }
}

```

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an S3 event with Lambda using JavaScript.

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

Consuming an S3 event with Lambda using TypeScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
    // Get the object from the event and show its content type
```

```
const bucket = event.Records[0].s3.bucket.name;
const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
const params = {
  Bucket: bucket,
  Key: key,
};
try {
  const { ContentType } = await s3.send(new HeadObjectCommand(params));
  console.log('CONTENT TYPE:', ContentType);
  return ContentType;
} catch (err) {
  console.log(err);
  const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your bucket is in the same region as this function.`;
  console.log(message);
  throw new Error(message);
}
};
```

PHP

SDK for PHP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an S3 event with Lambda using PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';
```

```
class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
                $bucket . '. Make sure they exist and your bucket is in the same region as this
                function.' . "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an S3 event with Lambda using Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')

def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
    encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and
        your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Ruby

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an S3 event with Lambda using Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
  Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
    and your bucket is in the same region as this function."
    raise e
  end
end
```

Rust

SDK for Rust

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an S3 event with Lambda using Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
```



```
tracing::info!(records = ?evt.payload.records.len(), "Received request from
SQS");

if evt.payload.records.len() == 0 {
    tracing::info!("Empty S3 event received");
}

let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
name to exist");
let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
exist");

tracing::info!("Request is for {} and object {}", bucket, key);

let s3_get_object_result = s3_client
    .get_object()
    .bucket(bucket)
    .key(key)
    .send()
    .await;

match s3_get_object_result {
    Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
contains a 'body' property of type ByteStream"),
    Err(_) => tracing::info!("Failure with S3 Get Object request")
}

Ok(())
}
```

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Code examples for Amazon S3 Control using Amazon SDKs

The following code examples show how to use Amazon S3 Control with an Amazon software development kit (SDK).

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Amazon S3 Control

- [Basic examples for Amazon S3 Control using Amazon SDKs](#)
 - [Hello Amazon S3 Control](#)
 - [Learn the basics of Amazon S3 Control with an Amazon SDK](#)
 - [Actions for Amazon S3 Control using Amazon SDKs](#)
 - [Use CreateJob with an Amazon SDK or CLI](#)
 - [Use DeleteJobTagging with an Amazon SDK](#)
 - [Use DescribeJob with an Amazon SDK or CLI](#)
 - [Use GetJobTagging with an Amazon SDK](#)
 - [Use PutJobTagging with an Amazon SDK](#)
 - [Use UpdateJobPriority with an Amazon SDK or CLI](#)
 - [Use UpdateJobStatus with an Amazon SDK or CLI](#)

Basic examples for Amazon S3 Control using Amazon SDKs

The following code examples show how to use the basics of Amazon S3 Control with Amazon SDKs.

Examples

- [Hello Amazon S3 Control](#)
- [Learn the basics of Amazon S3 Control with an Amazon SDK](#)
- [Actions for Amazon S3 Control using Amazon SDKs](#)
 - [Use CreateJob with an Amazon SDK or CLI](#)
 - [Use DeleteJobTagging with an Amazon SDK](#)
 - [Use DescribeJob with an Amazon SDK or CLI](#)
 - [Use GetJobTagging with an Amazon SDK](#)
 - [Use PutJobTagging with an Amazon SDK](#)

- [Use UpdateJobPriority with an Amazon SDK or CLI](#)
- [Use UpdateJobStatus with an Amazon SDK or CLI](#)

Hello Amazon S3 Control

The following code examples show how to get started using Amazon S3 Control.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlAsyncClient;
import software.amazon.awssdk.services.s3control.model.JobListDescriptor;
import software.amazon.awssdk.services.s3control.model.JobStatus;
import software.amazon.awssdk.services.s3control.model.ListJobsRequest;
import software.amazon.awssdk.services.s3control.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this example:
 * <p/>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
```

```

    * authentication for SDKs and tools, see https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs and Tools Reference Guide.
    * <p/>
    * You must have a runtime environment configured with the Java SDK.
    * See https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in the Developer Guide if this is not set up.
    */
public class HelloS3Batch {
    private static S3ControlAsyncClient asyncClient;

    public static void main(String[] args) {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        try {
            listBatchJobsAsync(accountId)
                .exceptionally(ex -> {
                    System.err.println("List batch jobs failed: " +
ex.getMessage());
                    return null;
                })
                .join();

        } catch (CompletionException ex) {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
        }
    }

    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
configuration:
     * <ul>
     * <li>Maximum concurrency: 100</li>
     * <li>Connection timeout: 60 seconds</li>
     * <li>Read timeout: 60 seconds</li>
     * <li>Write timeout: 60 seconds</li>
     * <li>API call timeout: 2 minutes</li>
     * <li>API call attempt timeout: 90 seconds</li>
     * <li>Retry policy: 3 retries</li>
     * <li>Region: US_EAST_1</li>
     * </ul>

```

```

    * <li>Credentials provider: {@link
EnvironmentVariableCredentialsProvider}</li>
    * </ul>
    *
    * @return the asynchronous S3 Control client instance
    */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    /**
    * Asynchronously lists batch jobs that have completed for the specified
    account.
    *
    * @param accountId the ID of the account to list jobs for
    * @return a CompletableFuture that completes when the job listing operation
    is finished
    */
    public static CompletableFuture<Void> listBatchJobsAsync(String accountId) {
        ListJobsRequest jobsRequest = ListJobsRequest.builder()
            .jobStatuses(JobStatus.COMPLETE)
            .accountId(accountId)
            .maxResults(10)

```

```

        .build();

        ListJobsPublisher publisher =
getAsyncClient().listJobsPaginator(jobsRequest);
        return publisher.subscribe(response -> {
            List<JobListDescriptor> jobs = response.jobs();
            for (JobListDescriptor job : jobs) {
                System.out.println("The job id is " + job.jobId());
                System.out.println("The job priority is " + job.priority());
            }
        }).thenAccept(response -> {
            System.out.println("Listing batch jobs completed");
        }).exceptionally(ex -> {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
            throw new RuntimeException(ex);
        });
    }
}

```

- For API details, see [ListJobs](#) in *Amazon SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

def list_jobs(self, account_id: str) -> None:
    """
    List all batch jobs for the account.

    Args:
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.list_jobs(
            AccountId=account_id,

```

```
        JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
    )
    jobs = response.get('Jobs', [])
    for job in jobs:
        print(f"The job id is {job['JobId']}")
        print(f"The job priority is {job['Priority']}")
except ClientError as e:
    print(f"Error listing jobs: {e}")
    raise
```

- For API details, see [ListJobs](#) in *Amazon SDK for Python (Boto3) API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Learn the basics of Amazon S3 Control with an Amazon SDK

The following code examples show how to learn core operations for Amazon S3 Control.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Learn core operations.

```
package com.example.s3.batch;

import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.IOException;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletionException;
```

```
public class S3BatchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");
    private static final String STACK_NAME = "MyS3Stack";
    public static void main(String[] args) throws IOException {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        String uuid = java.util.UUID.randomUUID().toString();
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 Batch basics scenario.");
        System.out.println("""
            S3 Batch operations enables efficient and cost-effective processing
of large-scale
            data stored in Amazon S3. It automatically scales resources to handle
varying workloads
            without the need for manual intervention.

            One of the key features of S3 Batch is its ability to perform tagging
operations on objects stored in
            S3 buckets. Users can leverage S3 Batch to apply, update, or remove
tags on thousands or millions of
            objects in a single operation, streamlining the management and
organization of their data.

            This can be particularly useful for tasks such as cost allocation,
lifecycle management, or
            metadata-driven workflows, where consistent and accurate tagging is
essential.

            S3 Batch's scalability and serverless nature make it an ideal
solution for organizations with
            growing data volumes and complex data management requirements.

            This Java program walks you through Amazon S3 Batch operations.

            Let's get started...

            """);
        waitForInputToContinue(scanner);
        // Use CloudFormation to stand up the resource required for this
scenario.
```



```
        System.out.println("Use CloudFormation to stand up the resource required
for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);

        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(STACK_NAME);
        String iamRoleArn = stackOutputs.get("S3BatchRoleArn");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setup the required bucket for this scenario.");
        waitForInputToContinue(scanner);
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
        actions.createBucket(bucketName);
        String reportBucketName = "arn:aws:s3:::"+bucketName;
        String manifestLocation = "arn:aws:s3:::"+bucketName+"/job-manifest.csv";
        System.out.println("Populate the bucket with the required files.");
        String[] fileNames = {"job-manifest.csv", "object-key-1.txt", "object-
key-2.txt", "object-key-3.txt", "object-key-4.txt"};
        actions.uploadFilesToBucket(bucketName, fileNames, actions);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a S3 Batch Job");
        System.out.println("This job tags all objects listed in the manifest file
with tags");
        waitForInputToContinue(scanner);
        String jobId ;
        try {
            jobId = actions.createS3JobAsync(accountId, iamRoleArn,
manifestLocation, reportBucketName, uuid).join();
            System.out.println("The Job id is " + jobId);

        } catch (S3Exception e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
```

```

        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update an existing S3 Batch Operations job's
priority");
        System.out.println("""
            In this step, we modify the job priority value. The higher the
number, the higher the priority.
            So, a job with a priority of `30` would have a higher priority than
a job with
            a priority of `20`. This is a common way to represent the priority
of a task
            or job, with higher numbers indicating a higher priority.

            Ensure that the job status allows for priority updates. Jobs in
certain
            states (e.g., Cancelled, Failed, or Completed) cannot have their
priorities
            updated. Only jobs in the Active or Suspended state typically allow
priority
            updates.
            """);

        try {
            actions.updateJobPriorityAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Update job priority failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to update job priority: " +
ex.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Cancel the S3 Batch job");
        System.out.print("Do you want to cancel the Batch job? (y/n): ");
        String cancelAns = scanner.nextLine();
        if (cancelAns != null && cancelAns.trim().equalsIgnoreCase("y")) {
            try {

```

```
        actions.cancelJobAsync(jobId, accountId)
            .exceptionally(ex -> {
                System.err.println("Cancel job failed: " +
ex.getMessage());
                return null;
            })
            .join();
    } catch (CompletionException ex) {
        System.err.println("Failed to cancel job: " + ex.getMessage());
    }
} else {
    System.out.println("Job " + jobId + " was not canceled.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Describe the job that was just created");
waitForInputToContinue(scanner);
try {
    actions.describeJobAsync(jobId, accountId)
        .exceptionally(ex -> {
            System.err.println("Describe job failed: " +
ex.getMessage());
            return null;
        })
        .join();
} catch (CompletionException ex) {
    System.err.println("Failed to describe job: " + ex.getMessage());
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Describe the tags associated with the job");
waitForInputToContinue(scanner);
try {
    actions.getJobTagsAsync(jobId, accountId)
        .exceptionally(ex -> {
            System.err.println("Get job tags failed: " +
ex.getMessage());
            return null;
        })
        .join();
} catch (CompletionException ex) {
    System.err.println("Failed to get job tags: " + ex.getMessage());
}
```

```
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Update Batch Job Tags");
    waitForInputToContinue(scanner);
    try {
        actions.putJobTaggingAsync(jobId, accountId)
            .exceptionally(ex -> {
                System.err.println("Put job tagging failed: " +
ex.getMessage());
                return null;
            })
            .join();
    } catch (CompletionException ex) {
        System.err.println("Failed to put job tagging: " + ex.getMessage());
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Delete the Amazon S3 Batch job tagging.");
    System.out.print("Do you want to delete Batch job tagging? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        try {
            actions.deleteBatchJobTagsAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Delete batch job tags failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to delete batch job tags: " +
ex.getMessage());
        }
    } else {
        System.out.println("Tagging was not deleted.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.print("Do you want to delete the AWS resources used in this
scenario? (y/n)");
```

```

        String delResAns = scanner.nextLine();
        if (delResAns != null && delResAns.trim().equalsIgnoreCase("y")) {
            actions.deleteFilesFromBucket(bucketName, fileNames, actions);
            actions.deleteBucketFolderAsync(bucketName);
            actions.deleteBucket(bucketName)
                .thenRun(() -> System.out.println("Bucket deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
                });
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
        } else {
            System.out.println("The AWS resources were not deleted.");
        }
        System.out.println("The Amazon S3 Batch scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println();
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println();
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
}
}

```

An action class that wraps operations.

```
public class S3BatchActions {
```

```

private static S3ControlAsyncClient asyncClient;

private static S3AsyncClient s3AsyncClient ;
/**
 * Retrieves the asynchronous S3 Control client instance.
 * <p>
 * This method creates and returns a singleton instance of the {@link
S3ControlAsyncClient}. If the instance
 * has not been created yet, it will be initialized with the following
configuration:
 * <ul>
 * <li>Maximum concurrency: 100</li>
 * <li>Connection timeout: 60 seconds</li>
 * <li>Read timeout: 60 seconds</li>
 * <li>Write timeout: 60 seconds</li>
 * <li>API call timeout: 2 minutes</li>
 * <li>API call attempt timeout: 90 seconds</li>
 * <li>Retry policy: 3 retries</li>
 * <li>Region: US_EAST_1</li>
 * <li>Credentials provider: {@link
EnvironmentVariableCredentialsProvider}</li>
 * </ul>
 *
 * @return the asynchronous S3 Control client instance
 */
private static S3ControlAsyncClient getAsyncClient() {
    if (asyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();
    }
}

```

```

        asyncClient = S3ControlAsyncClient.builder()
            .region(Region.US_EAST_1)
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return asyncClient;
}

private static S3AsyncClient getS3AsyncClient() {
    if (asyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryStrategy(RetryMode.STANDARD)
            .build();

        s3AsyncClient = S3AsyncClient.builder()
            .region(Region.US_EAST_1)
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return s3AsyncClient;
}

/**
 * Cancels a job asynchronously.
 *
 * @param jobId The ID of the job to be canceled.
 * @param accountId The ID of the account associated with the job.
 * @return A {@link CompletableFuture} that completes when the job status has
 *         been updated to "CANCELLED".
 *         If an error occurs during the update, the returned future will
 *         complete exceptionally.

```

```

    */
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
    {
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Updates the priority of a job asynchronously.
     *
     * @param jobId      the ID of the job to update
     * @param accountId the ID of the account associated with the job
     * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error
has occurred
     */
    public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
        UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .priority(60)
            .build();

        CompletableFuture<Void> future = new CompletableFuture<>();
        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
            });
    }

```



```

        future.complete(null); // Complete the CompletableFuture on
successful execution
    })
    .exceptionally(ex -> {
        System.err.println("Failed to update job priority: " +
ex.getMessage());
        future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
        return null; // Return null to handle the exception
    });

    return future;
}

/**
 * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
 *
 * @param jobId      the ID of the job for which to retrieve the tags
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
 * @throws RuntimeException if an error occurs while retrieving the job tags
 */
public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
    GetJobTaggingRequest request = GetJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();

    return asyncClient.getJobTagging(request)
        .thenAccept(response -> {
            List<S3Tag> tags = response.tags();
            if (tags.isEmpty()) {
                System.out.println("No tags found for job ID: " + jobId);
            } else {
                for (S3Tag tag : tags) {
                    System.out.println("Tag key is: " + tag.key());
                    System.out.println("Tag value is: " + tag.value());
                }
            }
        })
        .exceptionally(ex -> {

```

```

        System.err.println("Failed to get job tags: " + ex.getMessage());
        throw new RuntimeException(ex); // Propagate the exception
    });
}

/**
 * Asynchronously deletes the tags associated with a specific batch job.
 *
 * @param jobId      The ID of the batch job whose tags should be deleted.
 * @param accountId The ID of the account associated with the batch job.
 * @return A CompletableFuture that completes when the job tags have been
 * successfully deleted, or an exception is thrown if the deletion fails.
 */
public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
    DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .build();

    return asyncClient.deleteJobTagging(jobTaggingRequest)
        .thenAccept(response -> {
            System.out.println("You have successfully deleted " + jobId + "
tagging.");
        })
        .exceptionally(ex -> {
            System.err.println("Failed to delete job tags: " +
ex.getMessage());
            throw new RuntimeException(ex);
        });
}

/**
 * Asynchronously describes the specified job.
 *
 * @param jobId      the ID of the job to describe
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job
 * description is available
 * @throws RuntimeException if an error occurs while describing the job
 */
public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {

```

```
DescribeJobRequest jobRequest = DescribeJobRequest.builder()
    .jobId(jobId)
    .accountId(accountId)
    .build();

return getAsyncClient().describeJob(jobRequest)
    .thenAccept(response -> {
        System.out.println("Job ID: " + response.job().jobId());
        System.out.println("Description: " +
response.job().description());
        System.out.println("Status: " + response.job().statusAsString());
        System.out.println("Role ARN: " + response.job().roleArn());
        System.out.println("Priority: " + response.job().priority());
        System.out.println("Progress Summary: " +
response.job().progressSummary());

        // Print out details about the job manifest.
        JobManifest manifest = response.job().manifest();
        System.out.println("Manifest Location: " +
manifest.location().objectArn());
        System.out.println("Manifest ETag: " +
manifest.location().eTag());

        // Print out details about the job operation.
        JobOperation operation = response.job().operation();
        if (operation.s3PutObjectTagging() != null) {
            System.out.println("Operation: S3 Put Object Tagging");
            System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
        }

        // Print out details about the job report.
        JobReport report = response.job().report();
        System.out.println("Report Bucket: " + report.bucket());
        System.out.println("Report Prefix: " + report.prefix());
        System.out.println("Report Format: " + report.format());
        System.out.println("Report Enabled: " + report.enabled());
        System.out.println("Report Scope: " +
report.reportScopeAsString());
    })
    .exceptionally(ex -> {
        System.err.println("Failed to describe job: " + ex.getMessage());
        throw new RuntimeException(ex);
    });
```

```

    }

    /**
     * Creates an asynchronous S3 job using the AWS Java SDK.
     *
     * @param accountId      the AWS account ID associated with the job
     * @param iamRoleArn     the ARN of the IAM role to be used for the job
     * @param manifestLocation the location of the job manifest file in S3
     * @param reportBucketName the name of the S3 bucket to store the job report
     * @param uuid           a unique identifier for the job
     * @return a CompletableFuture that represents the asynchronous creation of
     the S3 job.
     *
     * The CompletableFuture will return the job ID if the job is created
     successfully,
     *
     * or throw an exception if there is an error.
     */
    public CompletableFuture<String> createS3JobAsync(String accountId, String
                                                    iamRoleArn,
                                                    String manifestLocation,
                                                    String reportBucketName, String uuid) {

        String[] bucketName = new String[]{" "};
        String[] parts = reportBucketName.split("::");
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);
            });
    }

```

```
S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
    .tagSet(tagSet)
    .build();

JobOperation jobOperation = JobOperation.builder()
    .s3PutObjectTagging(objectTaggingOperation)
    .build();

JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
    .objectArn(manifestLocation)
    .eTag(eTag)
    .build();

JobManifestSpec manifestSpec = JobManifestSpec.builder()
    .fieldsWithStrings("Bucket", "Key")
    .format("S3BatchOperations_CSV_20180820")
    .build();

JobManifest jobManifest = JobManifest.builder()
    .spec(manifestSpec)
    .location(jobManifestLocation)
    .build();

JobReport jobReport = JobReport.builder()
    .bucket(reportBucketName)
    .prefix("reports")
    .format("Report_CSV_20180820")
    .enabled(true)
    .reportScope("AllTasks")
    .build();

CreateJobRequest jobRequest = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Job created using the AWS Java SDK")
    .manifest(jobManifest)
    .operation(jobOperation)
    .report(jobReport)
    .priority(42)
    .roleArn(iamRoleArn)
    .clientRequestToken(uuid)
    .confirmationRequired(false)
```

```

        .build();

        // Create the job asynchronously.
        return getAsyncClient().createJob(jobRequest)
            .thenApply(CreateJobResponse::jobId);
    })
    .handle((jobId, ex) -> {
        if (ex != null) {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof S3ControlException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }
        return jobId;
    });
}

/**
 * Retrieves the ETag (Entity Tag) for an object stored in an Amazon S3
bucket.
 *
 * @param bucketName the name of the Amazon S3 bucket where the object is
stored
 * @param key the key (file name) of the object in the Amazon S3 bucket
 * @return the ETag of the object
 */
public String getETag(String bucketName, String key) {
    S3Client s3Client = S3Client.builder()
        .region(Region.US_EAST_1)
        .build();

    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
    return headObjectResponse.eTag();
}

```

```
/**
 * Asynchronously adds tags to a job in the system.
 *
 * @param jobId      the ID of the job to add tags to
 * @param accountId  the account ID associated with the job
 * @return a CompletableFuture that completes when the tagging operation is
finished
 */
public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
    S3Tag departmentTag = S3Tag.builder()
        .key("department")
        .value("Marketing")
        .build();

    S3Tag fiscalYearTag = S3Tag.builder()
        .key("FiscalYear")
        .value("2020")
        .build();

    PutJobTaggingRequest putJobTaggingRequest =
PutJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .tags(departmentTag, fiscalYearTag)
        .build();

    return asyncClient.putJobTagging(putJobTaggingRequest)
        .thenRun(() -> {
            System.out.println("Additional Tags were added to job " + jobId);
        })
        .exceptionally(ex -> {
            System.err.println("Failed to add tags to job: " +
ex.getMessage());
            throw new RuntimeException(ex); // Propagate the exception
        });
}

// Setup the S3 bucket required for this scenario.
/**
 * Creates an Amazon S3 bucket with the specified name.
 *
 * @param bucketName the name of the S3 bucket to create
 * @throws S3Exception if there is an error creating the bucket
 */
```

```

    */
    public void createBucket(String bucketName) {
        try {
            S3Client s3Client = S3Client.builder()
                .region(Region.US_EAST_1)
                .build();

            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Uploads a file to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param fileName the name of the file to be uploaded
     * @throws RuntimeException if an error occurs during the file upload
     */
    public void populateBucket(String bucketName, String fileName) {
        // Define the path to the directory.
        Path filePath = Paths.get("src/main/resources/batch/",
fileName).toAbsolutePath();
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)

```



```
        .build());

        CompletableFuture<PutObjectResponse> future =
getS3AsyncClient().putObject(putOb, AsyncRequestBody.fromFile(filePath));
        future.whenComplete((result, ex) -> {
            if (ex != null) {
                System.err.println("Error uploading file: " + ex.getMessage());
            } else {
                System.out.println("Successfully placed " + fileName + " into
bucket " + bucketName);
            }
        }).join();
    }

    // Update the bucketName in CSV.
    public void updateCSV(String newValue) {
        Path csvFilePath = Paths.get("src/main/resources/batch/job-
manifest.csv").toAbsolutePath();
        try {
            // Read all lines from the CSV file.
            List<String> lines = Files.readAllLines(csvFilePath);

            // Update the first value in each line.
            List<String> updatedLines = lines.stream()
                .map(line -> {
                    String[] parts = line.split(",");
                    parts[0] = newValue;
                    return String.join(",", parts);
                })
                .collect(Collectors.toList());

            // Write the updated lines back to the CSV file
            Files.write(csvFilePath, updatedLines);
            System.out.println("CSV file updated successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Deletes an object from an Amazon S3 bucket asynchronously.
     *
     * @param bucketName The name of the S3 bucket where the object is stored.
```

```

    * @param objectName The name of the object to be deleted.
    * @return A {@link CompletableFuture} that completes when the object has
    been deleted,
    *         or throws a {@link RuntimeException} if an error occurs during the
    deletion.
    */
    public CompletableFuture<Void> deleteBucketObjects(String bucketName, String
    objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
            .key(objectName)
            .build());

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(Delete.builder()
                .objects(toDelete).build())
            .build();

        return getS3AsyncClient().deleteObjects(dor)
            .thenAccept(result -> {
                System.out.println("The object was deleted!");
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Error deleting object: " +
ex.getMessage(), ex);
            });
    }

    /**
    * Deletes a folder and all its contents asynchronously from an Amazon S3
    bucket.
    *
    * @param bucketName the name of the S3 bucket containing the folder to be
    deleted
    * @return a {@link CompletableFuture} that completes when the folder and its
    contents have been deleted
    * @throws RuntimeException if any error occurs during the deletion process
    */
    public void deleteBucketFolderAsync(String bucketName) {
        String folderName = "reports/";
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(folderName)

```

```

        .build();

        CompletableFuture<ListObjectsV2Response> listObjectsFuture =
getS3AsyncClient().listObjectsV2(request);
        listObjectsFuture.thenCompose(response -> {
            List<CompletableFuture<DeleteObjectResponse>> deleteFutures =
response.contents().stream()
                .map(obj -> {
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(obj.key())
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted object: " + obj.key());
                            return deleteResponse;
                        });
                })
                .collect(Collectors.toList());

            return CompletableFuture.allOf(deleteFutures.toArray(new
CompletableFuture[0]))
                .thenCompose(v -> {
                    // Delete the folder.
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(folderName)
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted folder: " + folderName);
                            return deleteResponse;
                        });
                });
        }).join();
    }

/**
 * Deletes an Amazon S3 bucket.
 *
 * @param bucketName the name of the bucket to delete

```

```

    * @return a {@link CompletableFuture} that completes when the bucket has
    been deleted, or exceptionally if there is an error
    * @throws RuntimeException if there is an error deleting the bucket
    */
    public CompletableFuture<Void> deleteBucket(String bucketName) {
        S3AsyncClient s3Client = getS3AsyncClient();
        return s3Client.deleteBucket(DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build())
            .thenAccept(deleteBucketResponse -> {
                System.out.println(bucketName + " was deleted");
            })
            .exceptionally(ex -> {
                // Handle the exception or rethrow it.
                throw new RuntimeException("Failed to delete bucket: " +
bucketName, ex);
            });
    }

    /**
     * Uploads a set of files to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the files to
     * @param fileNames an array of file names to be uploaded
     * @param actions an instance of {@link S3BatchActions} that provides the
    implementation for the necessary S3 operations
     * @throws IOException if there's an error creating the text files or
    uploading the files to the S3 bucket
     */
    public static void uploadFilesToBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        actions.updateCSV(bucketName);
        createTextFiles(fileNames);
        for (String fileName : fileNames) {
            actions.populateBucket(bucketName, fileName);
        }
        System.out.println("All files are placed in the S3 bucket " +
bucketName);
    }

    /**
     * Deletes the specified files from the given S3 bucket.
     *
     * @param bucketName the name of the S3 bucket

```

```

    * @param fileNames an array of file names to be deleted from the bucket
    * @param actions the S3BatchActions instance to be used for the file
    deletion
    * @throws IOException if an I/O error occurs during the file deletion
    */
    public void deleteFilesFromBucket(String bucketName, String[] fileNames,
    S3BatchActions actions) throws IOException {
        for (String fileName : fileNames) {
            actions.deleteBucketObjects(bucketName, fileName)
                .thenRun(() -> System.out.println("Object deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
                });
        }
        System.out.println("All files have been deleted from the bucket " +
        bucketName);
    }

    public static void createTextFiles(String[] fileNames) {
        String currentDirectory = System.getProperty("user.dir");
        String directoryPath = currentDirectory + "\\src\\main\\resources\\
        \\batch";
        Path path = Paths.get(directoryPath);

        try {
            // Create the directory if it doesn't exist.
            if (Files.notExists(path)) {
                Files.createDirectories(path);
                System.out.println("Created directory: " + path.toString());
            } else {
                System.out.println("Directory already exists: " +
        path.toString());
            }

            for (String fileName : fileNames) {
                // Check if the file is a .txt file.
                if (fileName.endsWith(".txt")) {
                    // Define the path for the new file.
                    Path filePath = path.resolve(fileName);
                    System.out.println("Attempting to create file: " +
        filePath.toString());

                    // Create and write content to the new file.

```

```
        Files.write(filePath, "This is a test".getBytes());

        // Verify the file was created.
        if (Files.exists(filePath)) {
            System.out.println("Successfully created file: " +
filePath.toString());
        } else {
            System.out.println("Failed to create file: " +
filePath.toString());
        }
    }

    } catch (IOException e) {
        System.err.println("An error occurred: " + e.getMessage());
        e.printStackTrace();
    }
}

public String getAccountId() {
    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    GetCallerIdentityResponse callerIdentityResponse =
stsClient.getCallerIdentity();
    return callerIdentityResponse.account();
}
}
```

- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.
 - [CreateJob](#)
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)
 - [UpdateJobStatus](#)

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Learn S3 Batch Basics Scenario.

```
class S3BatchWrapper:
    """Wrapper class for managing S3 Batch Operations."""

    def __init__(self, s3_client: Any, s3control_client: Any, sts_client: Any) ->
None:
    """
    Initializes the S3BatchWrapper with AWS service clients.

    :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                        access to AWS S3 services.
    :param s3control_client: A Boto3 Amazon S3 Control client. This client
provides
                        low-level access to AWS S3 Control services.
    :param sts_client: A Boto3 AWS STS client. This client provides low-level
                        access to AWS STS services.
    """
    self.s3_client = s3_client
    self.s3control_client = s3control_client
    self.sts_client = sts_client
    # Get region from the client for bucket creation logic
    self.region_name = self.s3_client.meta.region_name

    def get_account_id(self) -> str:
        """
        Get AWS account ID.

        Returns:
            str: AWS account ID
        """
        return self.sts_client.get_caller_identity()["Account"]
```

```
def create_bucket(self, bucket_name: str) -> None:
    """
    Create an S3 bucket.

    Args:
        bucket_name (str): Name of the bucket to create

    Raises:
        ClientError: If bucket creation fails
    """
    try:
        if self.region_name and self.region_name != 'us-east-1':
            self.s3_client.create_bucket(
                Bucket=bucket_name,
                CreateBucketConfiguration={
                    'LocationConstraint': self.region_name
                }
            )
        else:
            self.s3_client.create_bucket(Bucket=bucket_name)
        print(f"Created bucket: {bucket_name}")
    except ClientError as e:
        print(f"Error creating bucket: {e}")
        raise

def upload_files_to_bucket(self, bucket_name: str, file_names: List[str]) -> str:
    """
    Upload files to S3 bucket including manifest file.

    Args:
        bucket_name (str): Target bucket name
        file_names (list): List of file names to upload

    Returns:
        str: ETag of the manifest file

    Raises:
        ClientError: If file upload fails
    """
    try:
        for file_name in file_names:
            if file_name != "job-manifest.csv":
```



```

        content = f"Content for {file_name}"
        self.s3_client.put_object(
            Bucket=bucket_name,
            Key=file_name,
            Body=content.encode('utf-8')
        )
        print(f"Uploaded {file_name} to {bucket_name}")

    manifest_content = ""
    for file_name in file_names:
        if file_name != "job-manifest.csv":
            manifest_content += f"{bucket_name},{file_name}\n"

    manifest_response = self.s3_client.put_object(
        Bucket=bucket_name,
        Key="job-manifest.csv",
        Body=manifest_content.encode('utf-8')
    )
    print(f"Uploaded manifest file to {bucket_name}")
    print(f"Manifest content:\n{manifest_content}")
    return manifest_response['ETag'].strip('')

except ClientError as e:
    print(f"Error uploading files: {e}")
    raise

def create_s3_batch_job(self, account_id: str, role_arn: str,
    manifest_location: str,
    report_bucket_name: str) -> str:
    """
    Create an S3 batch operation job.

    Args:
        account_id (str): AWS account ID
        role_arn (str): IAM role ARN for batch operations
        manifest_location (str): Location of the manifest file
        report_bucket_name (str): Bucket for job reports

    Returns:
        str: Job ID

    Raises:
        ClientError: If job creation fails
    """

```

```
try:
    bucket_name = manifest_location.split('::')[1].split('/')[0]
    manifest_key = 'job-manifest.csv'
    manifest_obj = self.s3_client.head_object(
        Bucket=bucket_name,
        Key=manifest_key
    )
    etag = manifest_obj['ETag'].strip('\"')

    response = self.s3control_client.create_job(
        AccountId=account_id,
        Operation={
            'S3PutObjectTagging': {
                'TagSet': [
                    {
                        'Key': 'BatchTag',
                        'Value': 'BatchValue'
                    },
                ]
            }
        },
        Report={
            'Bucket': report_bucket_name,
            'Format': 'Report_CSV_20180820',
            'Enabled': True,
            'Prefix': 'batch-op-reports',
            'ReportScope': 'AllTasks'
        },
        Manifest={
            'Spec': {
                'Format': 'S3BatchOperations_CSV_20180820',
                'Fields': ['Bucket', 'Key']
            },
            'Location': {
                'ObjectArn': manifest_location,
                'ETag': etag
            }
        },
        Priority=10,
        RoleArn=role_arn,
        Description='Batch job for tagging objects',
        ConfirmationRequired=True
    )
    job_id = response['JobId']
```

```

        print(f"The Job id is {job_id}")
        return job_id
    except ClientError as e:
        print(f"Error creating batch job: {e}")
        if 'Message' in str(e):
            print(f"Detailed error message: {e.response['Message']}")
        raise

    def check_job_failure_reasons(self, job_id: str, account_id: str) ->
    List[Dict[str, Any]]:
        """
        Check for any failure reasons of a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID

        Returns:
            list: List of failure reasons

        Raises:
            ClientError: If checking job failure reasons fails
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            if 'FailureReasons' in response['Job']:
                for reason in response['Job']['FailureReasons']:
                    print(f"- {reason}")
            return response['Job'].get('FailureReasons', [])
        except ClientError as e:
            print(f"Error checking job failure reasons: {e}")
            raise

    def wait_for_job_ready(self, job_id: str, account_id: str, desired_status:
    str = 'Ready') -> bool:
        """
        Wait for a job to reach the desired status.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID

```

```

        desired_status (str): Target status to wait for

Returns:
    bool: True if desired status is reached, False otherwise

Raises:
    ClientError: If checking job status fails
    """
    print(f"Waiting for job to become {desired_status}...")
    max_attempts = 60
    attempt = 0
    while attempt < max_attempts:
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")
            if current_status == desired_status:
                return True
            if current_status == 'Suspended':
                print("Job is in Suspended state, can proceed with
activation")
                return True
            if current_status in ['Active', 'Failed', 'Cancelled',
'Complete']:
                print(f"Job is in {current_status} state, cannot reach
{desired_status} status")
                if 'FailureReasons' in response['Job']:
                    print("Failure reasons:")
                    for reason in response['Job']['FailureReasons']:
                        print(f"- {reason}")
                    return False

            time.sleep(20)
            attempt += 1
        except ClientError as e:
            print(f"Error checking job status: {e}")
            raise
    print(f"Timeout waiting for job to become {desired_status}")
    return False

def update_job_priority(self, job_id: str, account_id: str) -> None:

```

```
"""
Update the priority of a batch job and start it.

Args:
    job_id (str): ID of the batch job
    account_id (str): AWS account ID
"""
try:
    response = self.s3control_client.describe_job(
        AccountId=account_id,
        JobId=job_id
    )
    current_status = response['Job']['Status']
    print(f"Current job status: {current_status}")

    if current_status in ['Ready', 'Suspended']:
        self.s3control_client.update_job_priority(
            AccountId=account_id,
            JobId=job_id,
            Priority=60
        )
        print("The job priority was updated")

    try:
        self.s3control_client.update_job_status(
            AccountId=account_id,
            JobId=job_id,
            RequestedJobStatus='Ready'
        )
        print("Job activated successfully")
    except ClientError as activation_error:
        print(f"Note: Could not activate job automatically:
{activation_error}")
        print("Job priority was updated successfully. Job may need
manual activation in the console.")
    elif current_status in ['Active', 'Completing', 'Complete']:
        print(f"Job is in '{current_status}' state - priority cannot be
updated")

    if current_status == 'Completing':
        print("Job is finishing up and will complete soon.")
    elif current_status == 'Complete':
        print("Job has already completed successfully.")
    else:
        print("Job is currently running.")
```

```

        else:
            print(f"Job is in '{current_status}' state - priority update not
allowed")

    except ClientError as e:
        print(f"Error updating job priority: {e}")
        print("Continuing with the scenario...")
        return

def cancel_job(self, job_id: str, account_id: str) -> None:
    """
    Cancel an S3 batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended', 'Active']:
            self.s3control_client.update_job_status(
                AccountId=account_id,
                JobId=job_id,
                RequestedJobStatus='Cancelled'
            )
            print(f"Job {job_id} was successfully canceled.")
        elif current_status in ['Completing', 'Complete']:
            print(f"Job is in '{current_status}' state - cannot be
cancelled")

            if current_status == 'Completing':
                print("Job is finishing up and will complete soon.")
            elif current_status == 'Complete':
                print("Job has already completed successfully.")
        else:
            print(f"Job is in '{current_status}' state - cancel not allowed")
    except ClientError as e:
        print(f"Error canceling job: {e}")
        raise

```

```

def describe_job_details(self, job_id: str, account_id: str) -> None:
    """
    Describe detailed information about a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        job = response['Job']
        print(f"Job ID: {job['JobId']}")
        print(f"Description: {job.get('Description', 'N/A')}")
        print(f"Status: {job['Status']}")
        print(f"Role ARN: {job['RoleArn']}")
        print(f"Priority: {job['Priority']}")
        if 'ProgressSummary' in job:
            progress = job['ProgressSummary']
            print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
              f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
              f"Failed={progress.get('NumberOfTasksFailed', 0)}")
    except ClientError as e:
        print(f"Error describing job: {e}")
        raise

def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Get tags associated with a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.get_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        tags = response.get('Tags', [])
    
```

```

        if tags:
            print(f"Tags for job {job_id}:")
            for tag in tags:
                print(f"  {tag['Key']}: {tag['Value']}")
        else:
            print(f"No tags found for job ID: {job_id}")
    except ClientError as e:
        print(f"Error getting job tags: {e}")
        raise

def put_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Add tags to a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.put_job_tagging(
            AccountId=account_id,
            JobId=job_id,
            Tags=[
                {'Key': 'Environment', 'Value': 'Development'},
                {'Key': 'Team', 'Value': 'DataProcessing'}
            ]
        )
        print(f"Additional tags were added to job {job_id}")
    except ClientError as e:
        print(f"Error adding job tags: {e}")
        raise

def list_jobs(self, account_id: str) -> None:
    """
    List all batch jobs for the account.

    Args:
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.list_jobs(
            AccountId=account_id,
            JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
                'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']

```



```

    )
    jobs = response.get('Jobs', [])
    for job in jobs:
        print(f"The job id is {job['JobId']}")
        print(f"The job priority is {job['Priority']}")
except ClientError as e:
    print(f"Error listing jobs: {e}")
    raise

def delete_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Delete all tags from a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.delete_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        print(f"You have successfully deleted {job_id} tagging.")
    except ClientError as e:
        print(f"Error deleting job tags: {e}")
        raise

def cleanup_resources(self, bucket_name: str, file_names: List[str]) -> None:
    """
    Clean up all resources created during the scenario.

    Args:
        bucket_name (str): Name of the bucket to clean up
        file_names (list): List of files to delete

    Raises:
        ClientError: If cleanup fails
    """
    try:
        for file_name in file_names:
            self.s3_client.delete_object(Bucket=bucket_name, Key=file_name)
            print(f"Deleted {file_name}")

        response = self.s3_client.list_objects_v2(

```

```
        Bucket=bucket_name,
        Prefix='batch-op-reports/'
    )
    if 'Contents' in response:
        for obj in response['Contents']:
            self.s3_client.delete_object(
                Bucket=bucket_name,
                Key=obj['Key']
            )
            print(f"Deleted {obj['Key']}")

    self.s3_client.delete_bucket(Bucket=bucket_name)
    print(f"Deleted bucket {bucket_name}")
except ClientError as e:
    print(f"Error in cleanup: {e}")
    raise
```

- For API details, see the following topics in *Amazon SDK for Python (Boto3) API Reference*.
 - [CreateJob](#)
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)
 - [UpdateJobStatus](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Actions for Amazon S3 Control using Amazon SDKs

The following code examples demonstrate how to perform individual Amazon S3 Control actions with Amazon SDKs. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

The following examples include only the most commonly used actions. For a complete list, see the [Amazon S3 Control API Reference](#).

Examples

- [Use CreateJob with an Amazon SDK or CLI](#)
- [Use DeleteJobTagging with an Amazon SDK](#)
- [Use DescribeJob with an Amazon SDK or CLI](#)
- [Use GetJobTagging with an Amazon SDK](#)
- [Use PutJobTagging with an Amazon SDK](#)
- [Use UpdateJobPriority with an Amazon SDK or CLI](#)
- [Use UpdateJobStatus with an Amazon SDK or CLI](#)

Use CreateJob with an Amazon SDK or CLI

The following code examples show how to use CreateJob.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Learn the basics](#)

CLI

Amazon CLI

To create an Amazon S3 batch operations job

The following create-job example creates an Amazon S3 batch operations job to tag objects as confidential in the bucket employee-records.

```
aws s3control create-job \
  --account-id 123456789012 \
  --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"confidential",
    "Value":"true"}] }}' \
  --report '{"Bucket":"arn:aws:s3:::employee-records-logs","Prefix":"batch-op-
    create-job",
    "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
  --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
    ["Bucket","Key"]},"Location":{"ObjectArn":"arn:aws:s3:::employee-records-logs/
```

```
inv-report/7a6a9be4-072c-407e-85a2-ec3e982f773e.csv", "ETag": "69f52a4e9f797e987155d9c8f5880897"}}' \
--priority 42 \
--role-arn arn:aws:iam::123456789012:role/S3BatchJobRole
```

Output:

```
{
  "JobId": "93735294-df46-44d5-8638-6356f335324e"
}
```

- For API details, see [CreateJob](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create an asynchronous S3 job.

```
/**
 * Creates an asynchronous S3 job using the AWS Java SDK.
 *
 * @param accountId      the AWS account ID associated with the job
 * @param iamRoleArn     the ARN of the IAM role to be used for the job
 * @param manifestLocation the location of the job manifest file in S3
 * @param reportBucketName the name of the S3 bucket to store the job report
 * @param uuid           a unique identifier for the job
 * @return a CompletableFuture that represents the asynchronous creation of
 * the S3 job.
 *       The CompletableFuture will return the job ID if the job is created
 * successfully,
 *       or throw an exception if there is an error.
 */
public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
```

```

String manifestLocation,
String reportBucketName, String uuid) {

    String[] bucketName = new String[]{" "};
    String[] parts = reportBucketName.split("::");
    if (parts.length > 1) {
        bucketName[0] = parts[1];
    } else {
        System.out.println("The input string does not contain the expected
format.");
    }

    return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
        .thenCompose(eTag -> {
            ArrayList<S3Tag> tagSet = new ArrayList<>();
            S3Tag s3Tag = S3Tag.builder()
                .key("keyOne")
                .value("ValueOne")
                .build();
            S3Tag s3Tag2 = S3Tag.builder()
                .key("keyTwo")
                .value("ValueTwo")
                .build();
            tagSet.add(s3Tag);
            tagSet.add(s3Tag2);

            S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
                .tagSet(tagSet)
                .build();

            JobOperation jobOperation = JobOperation.builder()
                .s3PutObjectTagging(objectTaggingOperation)
                .build();

            JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
                .objectArn(manifestLocation)
                .eTag(eTag)
                .build();

            JobManifestSpec manifestSpec = JobManifestSpec.builder()
                .fieldsWithStrings("Bucket", "Key")

```

```

        .format("S3BatchOperations_CSV_20180820")
        .build();

    JobManifest jobManifest = JobManifest.builder()
        .spec(manifestSpec)
        .location(jobManifestLocation)
        .build();

    JobReport jobReport = JobReport.builder()
        .bucket(reportBucketName)
        .prefix("reports")
        .format("Report_CSV_20180820")
        .enabled(true)
        .reportScope("AllTasks")
        .build();

    CreateJobRequest jobRequest = CreateJobRequest.builder()
        .accountId(accountId)
        .description("Job created using the AWS Java SDK")
        .manifest(jobManifest)
        .operation(jobOperation)
        .report(jobReport)
        .priority(42)
        .roleArn(iamRoleArn)
        .clientRequestToken(uuid)
        .confirmationRequired(false)
        .build();

    // Create the job asynchronously.
    return getAsyncClient().createJob(jobRequest)
        .thenApply(CreateJobResponse::jobId)
        .})
        .handle((jobId, ex) -> {
            if (ex != null) {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof S3ControlException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        })
        .return jobId;
    });

```

```
}
```

Create a compliance retention job.

```
/**
 * Creates a compliance retention job in Amazon S3 Control.
 * <p>
 * A compliance retention job in Amazon S3 Control is a feature that allows
you to
 * set a retention period for objects stored in an S3 bucket.
 * This feature is particularly useful for organizations that need to comply
with
 * regulatory requirements or internal policies that mandate the retention of
data for
 * a specific duration.
 *
 * @param s3ControlClient The S3ControlClient instance to use for the API
call.
 * @return The job ID of the created compliance retention job.
 */
public static String createComplianceRetentionJob(final S3ControlClient
s3ControlClient, String roleArn, String bucketName, String accountId) {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket/compliance-objects-manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
    JobOperation jobOperation = JobOperation.builder()
        .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                .retainUntilDate(jan2025)
                .build())
            .build())
        .build();

    JobManifestLocation manifestLocation = JobManifestLocation.builder()
        .objectArn(manifestObjectArn)
        .eTag(manifestObjectVersionId)
        .build();
```

```
JobManifestSpec manifestSpec = JobManifestSpec.builder()
    .fieldsWithStrings("Bucket", "Key")
    .format("S3BatchOperations_CSV_20180820")
    .build();

JobManifest manifestToPublicApi = JobManifest.builder()
    .location(manifestLocation)
    .spec(manifestSpec)
    .build();

// Report details.
final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
final String jobReportPrefix = "reports/compliance-objects-bops";

JobReport jobReport = JobReport.builder()
    .enabled(true)
    .reportScope(JobReportScope.ALL_TASKS)
    .bucket(jobReportBucketArn)
    .prefix(jobReportPrefix)
    .format(JobReportFormat.REPORT_CSV_20180820)
    .build();

final Boolean requiresConfirmation = true;
final int priority = 10;
CreateJobRequest request = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Set compliance retain-until to 1 Jan 2025")
    .manifest(manifestToPublicApi)
    .operation(jobOperation)
    .priority(priority)
    .roleArn(roleArn)
    .report(jobReport)
    .confirmationRequired(requiresConfirmation)
    .build();

// Create the job and get the result.
CreateJobResponse result = s3ControlClient.createJob(request);
return result.jobId();
}
```

Create a legal hold off job.


```

/**
 * Creates a compliance retention job in Amazon S3 Control.
 * <p>
 * A compliance retention job in Amazon S3 Control is a feature that allows
you to
 * set a retention period for objects stored in an S3 bucket.
 * This feature is particularly useful for organizations that need to comply
with
 * regulatory requirements or internal policies that mandate the retention of
data for
 * a specific duration.
 *
 * @param s3ControlClient The S3ControlClient instance to use for the API
call.
 * @return The job ID of the created compliance retention job.
 */
public static String createComplianceRetentionJob(final S3ControlClient
s3ControlClient, String roleArn, String bucketName, String accountId) {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket/compliance-objects-manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
    JobOperation jobOperation = JobOperation.builder()
        .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                .retainUntilDate(jan2025)
                .build())
            .build())
        .build();

    JobManifestLocation manifestLocation = JobManifestLocation.builder()
        .objectArn(manifestObjectArn)
        .eTag(manifestObjectVersionId)
        .build();

    JobManifestSpec manifestSpec = JobManifestSpec.builder()
        .fieldsWithStrings("Bucket", "Key")
        .format("S3BatchOperations_CSV_20180820")
        .build();

```

```

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

    // Report details.
    final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
    final String jobReportPrefix = "reports/compliance-objects-bops";

    JobReport jobReport = JobReport.builder()
        .enabled(true)
        .reportScope(JobReportScope.ALL_TASKS)
        .bucket(jobReportBucketArn)
        .prefix(jobReportPrefix)
        .format(JobReportFormat.REPORT_CSV_20180820)
        .build();

    final Boolean requiresConfirmation = true;
    final int priority = 10;
    CreateJobRequest request = CreateJobRequest.builder()
        .accountId(accountId)
        .description("Set compliance retain-until to 1 Jan 2025")
        .manifest(manifestToPublicApi)
        .operation(jobOperation)
        .priority(priority)
        .roleArn(roleArn)
        .report(jobReport)
        .confirmationRequired(requiresConfirmation)
        .build();

    // Create the job and get the result.
    CreateJobResponse result = s3ControlClient.createJob(request);
    return result.jobId();
}

```

Create a new governance retention job.

```

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:

```

```

*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class CreateGovernanceRetentionJob {

    public static void main(String[]args) throws ParseException {
        final String usage = ""

        Usage:
            <manifestObjectArn> <jobReportBucketArn> <roleArn> <accountId>
<manifestObjectVersionId>

        Where:
            manifestObjectArn - The Amazon Resource Name (ARN) of the S3
object that contains the manifest file for the governance objects.\s
            bucketName - The ARN of the S3 bucket where the job report will
be stored.
            roleArn - The ARN of the IAM role that will be used to perform
the governance retention operation.
            accountId - Your AWS account Id.
            manifestObjectVersionId = A unique value that is used as the
`eTag` property of the `JobManifestLocation` object.
        """;

        if (args.length != 4) {
            System.out.println(usage);
            return;
        }

        String manifestObjectArn = args[0];
        String jobReportBucketArn = args[1];
        String roleArn = args[2];
        String accountId = args[3];
        String manifestObjectVersionId = args[4];

        S3ControlClient s3ControlClient = S3ControlClient.create();
        createGovernanceRetentionJob(s3ControlClient, manifestObjectArn,
jobReportBucketArn, roleArn, accountId, manifestObjectVersionId);
    }

    public static String createGovernanceRetentionJob(final S3ControlClient
s3ControlClient, String manifestObjectArn, String jobReportBucketArn, String

```

```
roleArn, String accountId, String manifestObjectVersionId) throws ParseException
{
    final JobManifestLocation manifestLocation =
JobManifestLocation.builder()
        .objectArn(manifestObjectArn)
        .eTag(manifestObjectVersionId)
        .build();

    final JobManifestSpec manifestSpec = JobManifestSpec.builder()
        .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
        .fields(Arrays.asList(JobManifestFieldName.BUCKET,
JobManifestFieldName.KEY))
        .build();

    final JobManifest manifestToPublicApi = JobManifest.builder()
        .location(manifestLocation)
        .spec(manifestSpec)
        .build();

    final String jobReportPrefix = "reports/governance-objects";
    final JobReport jobReport = JobReport.builder()
        .enabled(true)
        .reportScope(JobReportScope.ALL_TASKS)
        .bucket(jobReportBucketArn)
        .prefix(jobReportPrefix)
        .format(JobReportFormat.REPORT_CSV_20180820)
        .build();

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan30th = format.parse("30/01/2025");

    final S3SetObjectRetentionOperation s3SetObjectRetentionOperation =
S3SetObjectRetentionOperation.builder()
        .retention(S3Retention.builder()
            .mode(S3ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(jan30th.toInstant())
            .build())
        .build();

    final JobOperation jobOperation = JobOperation.builder()
        .s3PutObjectRetention(s3SetObjectRetentionOperation)
        .build();

    final Boolean requiresConfirmation = true;
```

```

        final int priority = 10;

        final CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Put governance retention")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        final CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
}

```

- For API details, see [CreateJob](#) in *Amazon SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

def create_s3_batch_job(self, account_id: str, role_arn: str,
    manifest_location: str,
                                report_bucket_name: str) -> str:
    """
    Create an S3 batch operation job.

    Args:
        account_id (str): AWS account ID
        role_arn (str): IAM role ARN for batch operations
        manifest_location (str): Location of the manifest file
        report_bucket_name (str): Bucket for job reports
    """

```

Returns:

str: Job ID

Raises:

ClientError: If job creation fails

"""

try:

 bucket_name = manifest_location.split(':::')[1].split('/')[0]

 manifest_key = 'job-manifest.csv'

 manifest_obj = self.s3_client.head_object(

 Bucket=bucket_name,

 Key=manifest_key

)

 etag = manifest_obj['ETag'].strip('\"')'

 response = self.s3control_client.create_job(

 AccountId=account_id,

 Operation={

 'S3PutObjectTagging': {

 'TagSet': [

 {

 'Key': 'BatchTag',

 'Value': 'BatchValue'

 },

],

 }

 },

 Report={

 'Bucket': report_bucket_name,

 'Format': 'Report_CSV_20180820',

 'Enabled': True,

 'Prefix': 'batch-op-reports',

 'ReportScope': 'AllTasks'

 },

 Manifest={

 'Spec': {

 'Format': 'S3BatchOperations_CSV_20180820',

 'Fields': ['Bucket', 'Key']

 },

 'Location': {

 'ObjectArn': manifest_location,

 'ETag': etag

 }

```
        },
        Priority=10,
        RoleArn=role_arn,
        Description='Batch job for tagging objects',
        ConfirmationRequired=True
    )
    job_id = response['JobId']
    print(f"The Job id is {job_id}")
    return job_id
except ClientError as e:
    print(f"Error creating batch job: {e}")
    if 'Message' in str(e):
        print(f"Detailed error message: {e.response['Message']}")
    raise
```

- For API details, see [CreateJob](#) in *Amazon SDK for Python (Boto3) API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteJobTagging with an Amazon SDK

The following code examples show how to use DeleteJobTagging.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Learn the basics](#)

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

/**
 * Asynchronously deletes the tags associated with a specific batch job.
 *
 * @param jobId      The ID of the batch job whose tags should be deleted.
 * @param accountId The ID of the account associated with the batch job.
 * @return A CompletableFuture that completes when the job tags have been
 *         successfully deleted, or an exception is thrown if the deletion fails.
 */
public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
    DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .build();

    return asyncClient.deleteJobTagging(jobTaggingRequest)
        .thenAccept(response -> {
            System.out.println("You have successfully deleted " + jobId + "
tagging.");
        })
        .exceptionally(ex -> {
            System.err.println("Failed to delete job tags: " +
ex.getMessage());
            throw new RuntimeException(ex);
        });
}

```

- For API details, see [DeleteJobTagging](#) in *Amazon SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).


```
def delete_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Delete all tags from a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.delete_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        print(f"You have successfully deleted {job_id} tagging.")
    except ClientError as e:
        print(f"Error deleting job tags: {e}")
        raise
```

- For API details, see [DeleteJobTagging](#) in *Amazon SDK for Python (Boto3) API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DescribeJob with an Amazon SDK or CLI

The following code examples show how to use DescribeJob.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Learn the basics](#)

CLI

Amazon CLI

To describe an Amazon S3 batch operations job

The following `describe-job` provides configuration parameters and status for the specified batch operations job.

```
aws s3control describe-job \  
  --account-id 123456789012 \  
  --job-id 93735294-df46-44d5-8638-6356f335324e
```

Output:

```
{  
  "Job": {  
    "TerminationDate": "2019-10-03T21:49:53.944Z",  
    "JobId": "93735294-df46-44d5-8638-6356f335324e",  
    "FailureReasons": [],  
    "Manifest": {  
      "Spec": {  
        "Fields": [  
          "Bucket",  
          "Key"  
        ],  
        "Format": "S3BatchOperations_CSV_20180820"  
      },  
      "Location": {  
        "ETag": "69f52a4e9f797e987155d9c8f5880897",  
        "ObjectArn": "arn:aws:s3:::employee-records-logs/inv-report/7a6a9be4-072c-407e-85a2-ec3e982f773e.csv"  
      }  
    },  
    "Operation": {  
      "S3PutObjectTagging": {  
        "TagSet": [  
          {  
            "Value": "true",  
            "Key": "confidential"  
          }  
        ]  
      }  
    },  
    "RoleArn": "arn:aws:iam::123456789012:role/S3BatchJobRole",  
    "ProgressSummary": {  
      "TotalNumberOfTasks": 8,  
      "NumberOfTasksFailed": 0,  
      "NumberOfTasksSucceeded": 8  
    }  
  }  
}
```

```

    },
    "Priority": 42,
    "Report": {
        "ReportScope": "AllTasks",
        "Format": "Report_CSV_20180820",
        "Enabled": true,
        "Prefix": "batch-op-create-job",
        "Bucket": "arn:aws:s3:::employee-records-logs"
    },
    "JobArn": "arn:aws:s3:us-west-2:123456789012:job/93735294-
df46-44d5-8638-6356f335324e",
    "CreationTime": "2019-10-03T21:48:48.048Z",
    "Status": "Complete"
}
}

```

- For API details, see [DescribeJob](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

/**
 * Asynchronously describes the specified job.
 *
 * @param jobId      the ID of the job to describe
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job
description is available
 * @throws RuntimeException if an error occurs while describing the job
 */
public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {
    DescribeJobRequest jobRequest = DescribeJobRequest.builder()
        .jobId(jobId)
        .accountId(accountId)

```

```
.build();

return getAsyncClient().describeJob(jobRequest)
    .thenAccept(response -> {
        System.out.println("Job ID: " + response.job().jobId());
        System.out.println("Description: " +
response.job().description());
        System.out.println("Status: " + response.job().statusAsString());
        System.out.println("Role ARN: " + response.job().roleArn());
        System.out.println("Priority: " + response.job().priority());
        System.out.println("Progress Summary: " +
response.job().progressSummary());

        // Print out details about the job manifest.
        JobManifest manifest = response.job().manifest();
        System.out.println("Manifest Location: " +
manifest.location().objectArn());
        System.out.println("Manifest ETag: " +
manifest.location().eTag());

        // Print out details about the job operation.
        JobOperation operation = response.job().operation();
        if (operation.s3PutObjectTagging() != null) {
            System.out.println("Operation: S3 Put Object Tagging");
            System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
        }

        // Print out details about the job report.
        JobReport report = response.job().report();
        System.out.println("Report Bucket: " + report.bucket());
        System.out.println("Report Prefix: " + report.prefix());
        System.out.println("Report Format: " + report.format());
        System.out.println("Report Enabled: " + report.enabled());
        System.out.println("Report Scope: " +
report.reportScopeAsString());
    })
    .exceptionally(ex -> {
        System.err.println("Failed to describe job: " + ex.getMessage());
        throw new RuntimeException(ex);
    });
}
```

- For API details, see [DescribeJob](#) in *Amazon SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
def describe_job_details(self, job_id: str, account_id: str) -> None:
    """
    Describe detailed information about a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        job = response['Job']
        print(f"Job ID: {job['JobId']}")
        print(f"Description: {job.get('Description', 'N/A')}")
        print(f"Status: {job['Status']}")
        print(f"Role ARN: {job['RoleArn']}")
        print(f"Priority: {job['Priority']}")
        if 'ProgressSummary' in job:
            progress = job['ProgressSummary']
            print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
                  f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                  f"Failed={progress.get('NumberOfTasksFailed', 0)}")
    except ClientError as e:
        print(f"Error describing job: {e}")
        raise
```

- For API details, see [DescribeJob](#) in *Amazon SDK for Python (Boto3) API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetJobTagging with an Amazon SDK

The following code examples show how to use GetJobTagging.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Learn the basics](#)

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/**
 * Asynchronously retrieves the tags associated with a specific job in an AWS
 * account.
 *
 * @param jobId      the ID of the job for which to retrieve the tags
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job tags have
 *         been retrieved, or with an exception if the operation fails
 * @throws RuntimeException if an error occurs while retrieving the job tags
 */
public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
    GetJobTaggingRequest request = GetJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();
```

```

return asyncClient.getJobTagging(request)
    .thenAccept(response -> {
        List<S3Tag> tags = response.tags();
        if (tags.isEmpty()) {
            System.out.println("No tags found for job ID: " + jobId);
        } else {
            for (S3Tag tag : tags) {
                System.out.println("Tag key is: " + tag.key());
                System.out.println("Tag value is: " + tag.value());
            }
        }
    })
    .exceptionally(ex -> {
        System.err.println("Failed to get job tags: " + ex.getMessage());
        throw new RuntimeException(ex); // Propagate the exception
    });
}

```

- For API details, see [GetJobTagging](#) in *Amazon SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Get tags associated with a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.get_job_tagging(

```

```
        AccountId=account_id,
        JobId=job_id
    )
    tags = response.get('Tags', [])
    if tags:
        print(f"Tags for job {job_id}:")
        for tag in tags:
            print(f"    {tag['Key']}: {tag['Value']}")
    else:
        print(f"No tags found for job ID: {job_id}")
except ClientError as e:
    print(f"Error getting job tags: {e}")
    raise
```

- For API details, see [GetJobTagging](#) in *Amazon SDK for Python (Boto3) API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutJobTagging with an Amazon SDK

The following code examples show how to use PutJobTagging.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Learn the basics](#)

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/**
```



```

    * Asynchronously adds tags to a job in the system.
    *
    * @param jobId      the ID of the job to add tags to
    * @param accountId the account ID associated with the job
    * @return a CompletableFuture that completes when the tagging operation is
    finished
    */
    public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
        S3Tag departmentTag = S3Tag.builder()
            .key("department")
            .value("Marketing")
            .build();

        S3Tag fiscalYearTag = S3Tag.builder()
            .key("FiscalYear")
            .value("2020")
            .build();

        PutJobTaggingRequest putJobTaggingRequest =
PutJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .tags(departmentTag, fiscalYearTag)
            .build();

        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " +
ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
}

```

- For API details, see [PutJobTagging](#) in *Amazon SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
def put_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Add tags to a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.put_job_tagging(
            AccountId=account_id,
            JobId=job_id,
            Tags=[
                {'Key': 'Environment', 'Value': 'Development'},
                {'Key': 'Team', 'Value': 'DataProcessing'}
            ]
        )
        print(f"Additional tags were added to job {job_id}")
    except ClientError as e:
        print(f"Error adding job tags: {e}")
        raise
```

- For API details, see [PutJobTagging](#) in *Amazon SDK for Python (Boto3) API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use UpdateJobPriority with an Amazon SDK or CLI

The following code examples show how to use UpdateJobPriority.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Learn the basics](#)

CLI

Amazon CLI

To update the job priority of an Amazon S3 batch operations job

The following update-job-priority example updates the specified job to a new priority.

```
aws s3control update-job-priority \  
  --account-id 123456789012 \  
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \  
  --priority 52
```

Output:

```
{  
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386",  
  "Priority": 52  
}
```

- For API details, see [UpdateJobPriority](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```

/**
 * Updates the priority of a job asynchronously.
 *
 * @param jobId      the ID of the job to update
 * @param accountId  the ID of the account associated with the job
 * @return a {@link CompletableFuture} that represents the asynchronous
 * operation, which completes when the job priority has been updated or an error
 * has occurred
 */
public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
    UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .priority(60)
        .build();

    CompletableFuture<Void> future = new CompletableFuture<>();
    getAsyncClient().updateJobPriority(priorityRequest)
        .thenAccept(response -> {
            System.out.println("The job priority was updated");
            future.complete(null); // Complete the CompletableFuture on
successful execution
        })
        .exceptionally(ex -> {
            System.err.println("Failed to update job priority: " +
ex.getMessage());
            future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
            return null; // Return null to handle the exception
        });

    return future;
}

```

- For API details, see [UpdateJobPriority](#) in *Amazon SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
def update_job_priority(self, job_id: str, account_id: str) -> None:
    """
    Update the priority of a batch job and start it.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended']:
            self.s3control_client.update_job_priority(
                AccountId=account_id,
                JobId=job_id,
                Priority=60
            )
            print("The job priority was updated")

        try:
            self.s3control_client.update_job_status(
                AccountId=account_id,
                JobId=job_id,
                RequestedJobStatus='Ready'
            )
            print("Job activated successfully")
        except ClientError as activation_error:
```

```
        print(f"Note: Could not activate job automatically:
{activation_error}")
        print("Job priority was updated successfully. Job may need
manual activation in the console.")
        elif current_status in ['Active', 'Completing', 'Complete']:
            print(f"Job is in '{current_status}' state - priority cannot be
updated")
            if current_status == 'Completing':
                print("Job is finishing up and will complete soon.")
            elif current_status == 'Complete':
                print("Job has already completed successfully.")
            else:
                print("Job is currently running.")
        else:
            print(f"Job is in '{current_status}' state - priority update not
allowed")

    except ClientError as e:
        print(f"Error updating job priority: {e}")
        print("Continuing with the scenario...")
        return
```

- For API details, see [UpdateJobPriority](#) in *Amazon SDK for Python (Boto3) API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use UpdateJobStatus with an Amazon SDK or CLI

The following code examples show how to use UpdateJobStatus.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Learn the basics](#)

CLI

Amazon CLI

To update the status of an Amazon S3 batch operations job

The following `update-job-status` example cancels the specified job which is awaiting approval.

```
aws s3control update-job-status \  
  --account-id 123456789012 \  
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \  
  --requested-job-status Cancelled
```

Output:

```
{  
  "Status": "Cancelled",  
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386"  
}
```

The following `update-job-status` example confirms and runs the specified which is awaiting approval.

```
aws s3control update-job-status \  
  --account-id 123456789012 \  
  --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \  
  --requested-job-status Ready
```

Output::

```
{  
  "Status": "Ready",  
  "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"  
}
```

The following `update-job-status` example cancels the specified job which is running.

```
aws s3control update-job-status \  
  --account-id 123456789012 \  
  --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \  
  --requested-job-status Cancelled
```

```
--account-id 123456789012 \  
--job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \  
--requested-job-status Cancelled
```

Output::

```
{  
    "Status": "Cancelling",  
    "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"  
}
```

- For API details, see [UpdateJobStatus](#) in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
/**  
 * Cancels a job asynchronously.  
 *  
 * @param jobId The ID of the job to be canceled.  
 * @param accountId The ID of the account associated with the job.  
 * @return A {@link CompletableFuture} that completes when the job status has  
 *         been updated to "CANCELLED".  
 *         If an error occurs during the update, the returned future will  
 *         complete exceptionally.  
 */  
public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)  
{  
    UpdateJobStatusRequest updateJobStatusRequest =  
UpdateJobStatusRequest.builder()  
        .accountId(accountId)  
        .jobId(jobId)  
        .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))  
        .build();  
  
    return asyncClient.updateJobStatus(updateJobStatusRequest)
```



```
.thenAccept(updateJobStatusResponse -> {
    System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
})
.exceptionally(ex -> {
    System.err.println("Failed to cancel job: " + ex.getMessage());
    throw new RuntimeException(ex); // Propagate the exception
});
}
```

- For API details, see [UpdateJobStatus](#) in *Amazon SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
def cancel_job(self, job_id: str, account_id: str) -> None:
    """
    Cancel an S3 batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended', 'Active']:
            self.s3control_client.update_job_status(
                AccountId=account_id,
```

```
        JobId=job_id,
        RequestedJobStatus='Cancelled'
    )
    print(f"Job {job_id} was successfully canceled.")
elif current_status in ['Completing', 'Complete']:
    print(f"Job is in '{current_status}' state - cannot be
cancelled")
    if current_status == 'Completing':
        print("Job is finishing up and will complete soon.")
    elif current_status == 'Complete':
        print("Job has already completed successfully.")
else:
    print(f"Job is in '{current_status}' state - cancel not allowed")
except ClientError as e:
    print(f"Error canceling job: {e}")
    raise
```

- For API details, see [UpdateJobStatus](#) in *Amazon SDK for Python (Boto3) API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Code examples for S3 Directory Buckets using Amazon SDKs

The following code examples show how to use S3 Directory Buckets with an Amazon software development kit (SDK).

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other Amazon Web Services services.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

S3 Directory Buckets

- [Basic examples for S3 Directory Buckets using Amazon SDKs](#)
 - [Hello Amazon S3 directory buckets](#)
 - [Learn the basics of S3 Directory Buckets with an Amazon SDK](#)
 - [Actions for S3 Directory Buckets using Amazon SDKs](#)
 - [Use AbortMultipartUpload with an Amazon SDK](#)
 - [Use CompleteMultipartUpload with an Amazon SDK](#)
 - [Use CopyObject with an Amazon SDK](#)
 - [Use CreateBucket with an Amazon SDK](#)
 - [Use CreateMultipartUpload with an Amazon SDK](#)
 - [Use CreateSession with an Amazon SDK](#)
 - [Use DeleteBucket with an Amazon SDK](#)
 - [Use DeleteBucketEncryption with an Amazon SDK](#)
 - [Use DeleteBucketPolicy with an Amazon SDK](#)
 - [Use DeleteObject with an Amazon SDK](#)
 - [Use DeleteObjects with an Amazon SDK](#)
 - [Use GetBucketEncryption with an Amazon SDK](#)
 - [Use GetBucketPolicy with an Amazon SDK](#)
 - [Use GetObject with an Amazon SDK](#)
 - [Use GetObjectAttributes with an Amazon SDK](#)
 - [Use HeadBucket with an Amazon SDK](#)
 - [Use HeadObject with an Amazon SDK](#)
 - [Use ListDirectoryBuckets with an Amazon SDK](#)
 - [Use ListMultipartUploads with an Amazon SDK](#)
 - [Use ListObjectsV2 with an Amazon SDK](#)
 - [Use ListParts with an Amazon SDK](#)
 - [Use PutBucketEncryption with an Amazon SDK](#)
 - [Use PutBucketPolicy with an Amazon SDK](#)
 - [Use PutObject with an Amazon SDK](#)
 - [Use UploadPart with an Amazon SDK](#)

- [Use UploadPartCopy with an Amazon SDK](#)
- [Scenarios for S3 Directory Buckets using Amazon SDKs](#)
 - [Create a presigned URL for Amazon S3 directory buckets to get an object using an Amazon SDK](#)

Basic examples for S3 Directory Buckets using Amazon SDKs

The following code examples show how to use the basics of Amazon S3 Directory Buckets with Amazon SDKs.

Examples

- [Hello Amazon S3 directory buckets](#)
- [Learn the basics of S3 Directory Buckets with an Amazon SDK](#)
- [Actions for S3 Directory Buckets using Amazon SDKs](#)
 - [Use AbortMultipartUpload with an Amazon SDK](#)
 - [Use CompleteMultipartUpload with an Amazon SDK](#)
 - [Use CopyObject with an Amazon SDK](#)
 - [Use CreateBucket with an Amazon SDK](#)
 - [Use CreateMultipartUpload with an Amazon SDK](#)
 - [Use CreateSession with an Amazon SDK](#)
 - [Use DeleteBucket with an Amazon SDK](#)
 - [Use DeleteBucketEncryption with an Amazon SDK](#)
 - [Use DeleteBucketPolicy with an Amazon SDK](#)
 - [Use DeleteObject with an Amazon SDK](#)
 - [Use DeleteObjects with an Amazon SDK](#)
 - [Use GetBucketEncryption with an Amazon SDK](#)
 - [Use GetBucketPolicy with an Amazon SDK](#)
 - [Use GetObject with an Amazon SDK](#)
 - [Use GetObjectAttributes with an Amazon SDK](#)
 - [Use HeadBucket with an Amazon SDK](#)
 - [Use HeadObject with an Amazon SDK](#)
 - [Use ListDirectoryBuckets with an Amazon SDK](#)

- [Use ListMultipartUploads with an Amazon SDK](#)
- [Use ListObjectsV2 with an Amazon SDK](#)
- [Use ListParts with an Amazon SDK](#)
- [Use PutBucketEncryption with an Amazon SDK](#)
- [Use PutBucketPolicy with an Amazon SDK](#)
- [Use PutObject with an Amazon SDK](#)
- [Use UploadPart with an Amazon SDK](#)
- [Use UploadPartCopy with an Amazon SDK](#)

Hello Amazon S3 directory buckets

The following code example shows how to get started using Amazon S3 directory buckets.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
package com.example.s3.directorybucket;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Before running this example:
 * <p>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
 * authentication for SDKs and tools, see
 * https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs
 * and Tools Reference Guide.
 * <p>
 * You must have a runtime environment configured with the Java SDK.
 * See
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 * the Developer Guide if this is not set up.
 * <p>
 * To use S3 directory buckets, configure a gateway VPC endpoint. This is the
 * recommended method to enable directory bucket traffic without
 * requiring an internet gateway or NAT device. For more information on
 * configuring VPC gateway endpoints, visit
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-networking.html#s3-express-networking-vpc-gateway.
 * <p>
 * Directory buckets are available in specific AWS Regions and Zones. For
 * details on Regions and Zones supporting directory buckets, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-networking.html#s3-express-endpoints.
 */

public class HelloS3DirectoryBuckets {
    private static final Logger logger =
        LoggerFactory.getLogger(HelloS3DirectoryBuckets.class);

    public static void main(String[] args) {
```

```

        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.
        Region region = Region.US_WEST_2;
        String zone = "usw2-az1";
        S3Client s3Client = createS3Client(region);

        try {
            // Create the directory bucket
            createDirectoryBucket(s3Client, bucketName, zone);
            logger.info("Created bucket: {}", bucketName);

            // List all directory buckets
            List<String> bucketNames = listDirectoryBuckets(s3Client);
            bucketNames.forEach(name -> logger.info("Bucket Name: {}", name));
        } catch (S3Exception e) {
            logger.error("An error occurred during S3 operations: {} - Error
code: {}",
                e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        } finally {
            try {
                // Delete the created bucket
                deleteDirectoryBucket(s3Client, bucketName);
                logger.info("Deleted bucket: {}", bucketName);
            } catch (S3Exception e) {
                logger.error("Failed to delete the bucket due to S3 error: {} -
Error code: {}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            } catch (RuntimeException e) {
                logger.error("Failed to delete the bucket due to unexpected
error: {}", e.getMessage(), e);
            } finally {
                s3Client.close();
            }
        }
    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client    The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created

```

```

    * @param zone      The region where the bucket will be created
    * @throws S3Exception if there's an error creating the bucket
    */
    public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
        logger.info("Creating bucket: {}", bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
            .location(LocationInfo.builder()
                .type(LocationType.AVAILABILITY_ZONE)
                .name(zone).build())
            .bucket(BucketInfo.builder()
                .type(BucketType.DIRECTORY)
                .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                .build())
            .build();

        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Lists all S3 directory buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest

```



```
        ListDirectoryBucketsRequest listBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

        // Retrieve the list of buckets
        ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listBucketsRequest);

        // Extract bucket names
        List<String> bucketNames = response.buckets().stream()
            .map(Bucket::name)
            .collect(Collectors.toList());

        return bucketNames;
    } catch (S3Exception e) {
        logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

/**
 * Deletes the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the bucket to delete
 */
public static void deleteDirectoryBucket(S3Client s3Client, String
bucketName) {
    try {
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();
        s3Client.deleteBucket(deleteBucketRequest);
    } catch (S3Exception e) {
        logger.error("Failed to delete bucket: " + bucketName + " - Error
code: " + e.awsErrorDetails().errorCode(),
            e);
        throw e;
    }
}
}
```

- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.
 - [CreateBucket](#)
 - [ListDirectoryBuckets](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Learn the basics of S3 Directory Buckets with an Amazon SDK

The following code examples show how to:

- Set up a VPC and VPC Endpoint.
- Set up the Policies, Roles, and User to work with S3 directory buckets and the S3 Express One Zone storage class.
- Create two S3 Clients.
- Create two buckets.
- Create an object and copy it over.
- Demonstrate performance difference.
- Populate the buckets to show the lexicographical difference.
- Prompt the user to see if they want to clean up the resources.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Run an interactive scenario demonstrating Amazon S3 features.

```
public class S3DirectoriesScenario {
```

```
    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");

    private static final Logger logger =
    LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static S3AsyncClient mS3RegularClient;
    private static S3AsyncClient mS3ExpressClient;

    private static String mdirectoryBucketName;
    private static String mregularBucketName;

    private static String stackName = "cfn-stack-s3-express-basics--" +
    UUID.randomUUID();

    private static String regularUser = "";
    private static String vpcId = "";
    private static String expressUser = "";

    private static String vpcEndpointId = "";

    private static final S3DirectoriesActions s3DirectoriesActions = new
    S3DirectoriesActions();

    public static void main(String[] args) {
        try {
            s3ExpressScenario();
        } catch (RuntimeException e) {
            logger.info(e.getMessage());
        }
    }

    // Runs the scenario.
    private static void s3ExpressScenario() {
        logger.info(DASHES);
        logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK
    for Java V2.");
        logger.info("""
            Let's get started! First, please note that S3 Express One Zone works
    best when working within the AWS infrastructure,
            specifically when working in the same Availability Zone (AZ). To see
    the best results in this example and when you implement
```

```

        directory buckets into your infrastructure, it is best to put your
compute resources in the same AZ as your directory
        bucket.
        "");
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    // Create an optional VPC and create 2 IAM users.
    UserNames userNames = createVpcUsers();
    String expressUserName = userNames.getExpressUserName();
    String regularUserName = userNames.getRegularUserName();

    // Set up two S3 clients, one regular and one express,
    // and two buckets, one regular and one directory.
    setupClientsAndBuckets(expressUserName, regularUserName);

    // Create an S3 session for the express S3 client and add objects to the
buckets.
    logger.info("Now let's add some objects to our buckets and demonstrate
how to work with S3 Sessions.");
    waitForInputToContinue(scanner);
    String bucketObject = createSessionAdd0bjects();

    // Demonstrate performance differences between regular and directory
buckets.
    demonstratePerformance(bucketObject);

    // Populate the buckets to show the lexicographical difference between
// regular and express buckets.
    showLexicographicalDifferences(bucketObject);

    logger.info(DASHES);
    logger.info("That's it for our tour of the basic operations for S3
Express One Zone.");
    logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
    String response = scanner.next().trim().toLowerCase();
    if (response.equals("y")) {
        cleanUp(stackName);
    }
}

/*
Delete resources created by this scenario.
*/

```

```

    public static void cleanUp(String stackName) {
        try {
            if (mdirectoryBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
                }
                logger.info("Deleted directory bucket " + mdirectoryBucketName);
                mdirectoryBucketName = null;
                if (mregularBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3RegularClient,
mregularBucketName).join();
                }
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof S3Exception) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
            }

            logger.info("Deleted regular bucket " + mregularBucketName);
            mregularBucketName = null;
            CloudFormationHelper.destroyCloudFormationStack(stackName);
        }

        private static void showLexicographicalDifferences(String bucketObject) {
            logger.info(DASHES);
            logger.info("""
                7. Populate the buckets to show the lexicographical (alphabetical)
difference
                when object names are listed. Now let's explore how directory buckets
store
                objects in a different manner to regular buckets. The key is in the
name
                "Directory". Where regular buckets store their key/value pairs in a
                flat manner, directory buckets use actual directories/folders.
                This allows for more rapid indexing, traversing, and therefore
                retrieval times!

                The more segmented your bucket is, with lots of

```

```
directories, sub-directories, and objects, the more efficient it
becomes.
This structural difference also causes `ListObject` operations to
behave
differently, which can cause unexpected results. Let's add a few
more
objects in sub-directories to see how the output of
ListObjects changes.
""");

waitForInputToContinue(scanner);

// Populate a few more files in each bucket so that we can use
// ListObjects and show the difference.
String otherObject = "other/" + bucketObject;
String altObject = "alt/" + bucketObject;
String otherAltObject = "other/alt/" + bucketObject;

try {
    s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchBucketException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    try {
```

```

        // List objects in both S3 buckets.
        List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
        List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient,
mregularBucketName).join();

        logger.info("Directory bucket content");
        for (String obj : dirBucketObjects) {
            logger.info(obj);
        }

        logger.info("Regular bucket content");
        for (String obj : regBucketObjects) {
            logger.info(obj);
        }
    } catch (CompletionException e) {
        logger.error("Async operation failed: {} ",
e.getCause().getMessage());
        return;
    }

    logger.info("""
        Notice how the regular bucket lists objects in lexicographical order,
while the directory bucket does not. This is
        because the regular bucket considers the whole "key" to be the object
identifier, while the directory bucket actually
        creates directories and uses the object "key" as a path to the
object.
        """);
    waitForInputToContinue(scanner);
}

/**
 * Demonstrates the performance difference between downloading an object from
a directory bucket and a regular bucket.
 *
 * <p>This method:
 * <ul>
 *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
 *     <li>Downloads the specified object from the directory bucket and
measures the total time.</li>

```

```

    *      <li>Downloads the same object from the regular bucket and measures the
total time.</li>
    *      <li>Compares the time differences and prints the results.</li>
    * </ul>
    *
    * <p>Note: The performance difference will be more pronounced if this
example is run on an EC2 instance
    * in the same Availability Zone as the buckets.
    *
    * @param bucketObject the name of the object to download
    */
private static void demonstratePerformance(String bucketObject) {
    logger.info(DASHES);
    logger.info("6. Demonstrate the performance difference.");
    logger.info("""
        Now, let's do a performance test. We'll download the same object from
each
        bucket repeatedly and compare the total time needed.

        Note: the performance difference will be much more pronounced if this
example is run in an EC2 instance in the same Availability Zone as
the bucket.
        """);
    waitForInputToContinue(scanner);

    int downloads = 1000; // Default value.
    logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");

    // Ask if the user wants to download a different number.
    logger.info("Would you like to download the file a different number of
times? (y/n): ");
    String response = scanner.next().trim().toLowerCase();
    if (response.equals("y")) {
        int maxDownloads = 1_000_000;

        // Ask for a valid number of downloads.
        while (true) {
            logger.info("Enter a number between 1 and " + maxDownloads + "
for the number of downloads: ");
            if (scanner.hasNextInt()) {
                downloads = scanner.nextInt();
                if (downloads >= 1 && downloads <= maxDownloads) {
                    break;

```



```
        } else {
            logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
        }
    } else {
        logger.info("Invalid input. Please enter a valid integer.");
        scanner.next();
    }
}

    logger.info("You have chosen to download {} items.", downloads);
} else {
    logger.info("No changes made. Using default downloads: {}",
downloads);
}
// Simulating the download process for the directory bucket.
logger.info("Downloading from the directory bucket.");
long directoryTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
        // Get the object from the directory bucket.
        s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
    }
    return;
}
}
long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

// Download from the regular bucket.
logger.info("Downloading from the regular bucket.");
long normalTimeStart = System.nanoTime();
```

```

        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }

        long normalTimeDifference = System.nanoTime() - normalTimeStart;
        logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
        long difference = normalTimeDifference - directoryTimeDifference;
        logger.info("That's a difference of " + difference + " nanoseconds, or");
        logger.info(difference / 1_000_000_000.0 + " seconds.");

        if (difference < 0) {
            logger.info("The directory buckets were slower. This can happen if
you are not running on the cloud within a VPC.");
        }
        waitForInputToContinue(scanner);
    }

    private static String createSessionAddObjects() {
        logger.info(DASHES);
        logger.info("""
            5. Create an object and copy it.
            We'll create an object consisting of some text and upload it to the
            regular bucket.
            """);
        waitForInputToContinue(scanner);
    }

```

```

        String bucketObject = "basic-text-object.txt";
        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
            s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

            // Copy the object to the destination S3 bucket.
            s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
        logger.info("""
            It worked! This is because the S3Client that performed the copy
operation
            is the expressClient using the credentials for the user with
permission to
            work with directory buckets.

            It's important to remember the user permissions when interacting
with
            directory buckets. Instead of validating permissions on every call
as
            regular buckets do, directory buckets utilize the user credentials
and session
            token to validate. This allows for much faster connection speeds on
every call.
            For single calls, this is low, but for many concurrent calls
            this adds up to a lot of time saved.
            """);
        waitForInputToContinue(scanner);
        return bucketObject;
    }

    /**
     * Creates VPC users for the S3 Express One Zone scenario.

```

```

* <p>
* This method performs the following steps:
* <ol>
*     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
*     <li>Creates two IAM users: one with S3 Express One Zone permissions
and one without.</li>
* </ol>
*
* @return a {@link UserNames} object containing the names of the created IAM
users
*/
public static UserNames createVpcUsers() {
    /*
    Optionally create a VPC.
    Create two IAM users, one with S3 Express One Zone permissions and one
without.
    */
    logger.info(DASHES);
    logger.info("""
        1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
        directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
        """);

    logger.info("Do you want to setup a VPC Endpoint? (y/n)");
    String endpointAns = scanner.nextLine().trim();
    if (endpointAns.equalsIgnoreCase("y")) {
        logger.info("""
            Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
            """);
        try {
            s3DirectoriesActions.setupVPCAsync().join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
    }
}

```

```

        }
    }
    waitForInputToContinue(scanner);
} else {
    logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
}
logger.info(DASHES);
logger.info("""
    2. Create a RegularUser and ExpressUser by using the AWS CDK.
    One IAM User, named RegularUser, will have permissions to work only
    with regular buckets and one IAM user, named ExpressUser, will have
    permissions to work only with directory buckets.
    """);
waitForInputToContinue(scanner);

// Create two users required for this scenario.
Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
regularUser = stackOutputs.get("RegularUser");
expressUser = stackOutputs.get("ExpressUser");

UserNames names = new UserNames();
names.setRegularUserName(regularUser);
names.setExpressUserName(expressUser);
return names;
}

/**
 * Creates users using AWS CloudFormation.
 *
 * @return a {@link Map} of String keys and String values representing the
stack outputs,
 * which may include user-related information such as user names and IDs.
 */
public static Map<String, String> createUsersUsingCDK(String stackName) {
    logger.info("We'll use an AWS CloudFormation template to create the IAM
users and policies.");
    CloudFormationHelper.deployCloudFormationStack(stackName);
    return CloudFormationHelper.getStackOutputsAsync(stackName).join();
}

/**
 * Sets up the necessary clients and buckets for the S3 Express service.
 *

```

```

    * @param expressUserName the username for the user with S3 Express
permissions
    * @param regularUserName the username for the user with regular S3
permissions
    */
    public static void setupClientsAndBuckets(String expressUserName, String
regularUserName) {
        Scanner locscanner = new Scanner(System.in);
        String accessKeyIdforRegUser;
        String secretAccessforRegUser;
        try {
            CreateAccessKeyResponse keyResponse =
s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
            accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
            secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        String accessKeyIdforExpressUser;
        String secretAccessforExpressUser;
        try {
            CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
            accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
            secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
    }

```

```

    }
    return;
}

logger.info(DASHES);
logger.info("""
    3. Create two S3Clients; one uses the ExpressUser's credentials and
one uses the RegularUser's credentials.
    The 2 S3Clients will use different credentials.
    """);
waitForInputToContinue(locscanner);
try {
    mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
    mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
} catch (CompletionException ce) {
    Throwable cause = ce.getCause();
    if (cause instanceof IllegalArgumentException) {
        logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
    } else {
        logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
    }
    return;
}

logger.info("""
    We can now use the ExpressUser client to make calls to S3 Express
operations.
    """);
waitForInputToContinue(locscanner);
logger.info(DASHES);
logger.info("""
    4. Create two buckets.
    Now we will create a directory bucket which is the linchpin of the S3
Express One Zone service. Directory buckets
    behave differently from regular S3 buckets which we will explore
here. We'll also create a regular bucket, put
    an object into the regular bucket, and copy it to the directory
bucket.

```

```

        """);

        logger.info("""
            Now, let's choose an availability zone (AZ) for the directory
bucket.

            We'll choose one that is supported.
        """);
        String zoneId;
        String regularBucketName;
        try {
            zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();
            regularBucketName = "amzn-s3-demo-bucket"; // Replace with your
bucket name.
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                logger.error("EC2Exception occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }
        logger.info("""
            Now, let's create the actual directory bucket, as well as a regular
bucket."
        """);

        String directoryBucketName = "amzn-s3-demo-bucket--" + zoneId + "--x-
s3"; // Replace with your directory bucket name.
        try {
            s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
directoryBucketName, zoneId).join();
            logger.info("Created directory bucket {}", directoryBucketName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }
    }
}

```



```

        }
    }

    // Assign to the data member.
    mdirectoryBucketName = directoryBucketName;
    try {
        s3DirectoriesActions.createBucketAsync(mS3RegularClient,
regularBucketName).join();
        logger.info("Created regular bucket {} ", regularBucketName);
        mregularBucketName = regularBucketName;
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof BucketAlreadyExistsException) {
            logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            return;
        }
    }
    logger.info("Great! Both buckets were created.");
    waitForInputToContinue(locscanner);
}

/**
 * Creates an asynchronous S3 client with the specified access key and secret
access key.
 *
 * @param accessKeyId      the AWS access key ID
 * @param secretAccessKey the AWS secret access key
 * @return a {@link CompletableFuture} that asynchronously creates the S3
client
 * @throws IllegalArgumentException if the access key ID or secret access key
is null
 */
public static CompletableFuture<S3AsyncClient>
createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
    return CompletableFuture.supplyAsync(() -> {
        // Validate input parameters
        if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey
== null || secretAccessKey.isBlank()) {
            throw new IllegalArgumentException("Access Key ID and Secret
Access Key must not be null or empty");

```

```

        }

        AwsBasicCredentials awsCredentials =
        AwsBasicCredentials.create(accessKeyId, secretAccessKey);
        return S3AsyncClient.builder()

        .credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
            .region(Region.US_WEST_2)
            .build();
    });
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        logger.info("");
        logger.info("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            logger.info("Continuing with the program...");
            logger.info("");
            break;
        } else {
            logger.info("Invalid input. Please try again.");
        }
    }
}
}
}

```

A wrapper class for Amazon S3 SDK methods.

```

public class S3DirectoriesActions {

    private static IamAsyncClient iamAsyncClient;

    private static Ec2AsyncClient ec2AsyncClient;
    private static final Logger logger =
    LoggerFactory.getLogger(S3DirectoriesActions.class);

    private static IamAsyncClient getIAMAsyncClient() {
        if (iamAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()

```

```
        .maxConcurrency(100)
        .connectionTimeout(Duration.ofSeconds(60))
        .readTimeout(Duration.ofSeconds(60))
        .writeTimeout(Duration.ofSeconds(60))
        .build();

    ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2))
        .apiCallAttemptTimeout(Duration.ofSeconds(90))
        .retryStrategy(RetryMode.STANDARD)
        .build();

    iamAsyncClient = IamAsyncClient.builder()
        .httpClient(httpClient)
        .overrideConfiguration(overrideConfig)
        .build();
}
return iamAsyncClient;
}

private static Ec2AsyncClient getEc2AsyncClient() {
    if (ec2AsyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryStrategy(RetryMode.STANDARD)
            .build();

        ec2AsyncClient = Ec2AsyncClient.builder()
            .httpClient(httpClient)
            .region(Region.US_WEST_2)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return ec2AsyncClient;
}
```

```

    }

    /**
     * Deletes the specified S3 bucket and all the objects within it
     * asynchronously.
     *
     * @param s3AsyncClient the S3 asynchronous client to use for the operations
     * @param bucketName the name of the S3 bucket to be deleted
     * @return a {@link CompletableFuture} that completes with a {@link
     * WaiterResponse} containing the
     *
     *      {@link HeadBucketResponse} when the bucket has been successfully
     * deleted
     *
     * @throws CompletionException if there was an error deleting the bucket or
     * its objects
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
    ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

    return s3AsyncClient.listObjectsV2(listRequest)
        .thenCompose(listResponse -> {
            if (!listResponse.contents().isEmpty()) {
                List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()
                    .map(s3Object ->
ObjectIdentifier.builder().key(s3Object.key()).build())
                    .collect(Collectors.toList());

                DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
                    .bucket(bucketName)

.delete(Delete.builder().objects(objectIdentifiers).build())
                    .build();

                return s3AsyncClient.deleteObjects(deleteRequest)
                    .thenAccept(deleteResponse -> {
                        if (!deleteResponse.errors().isEmpty()) {
                            deleteResponse.errors().forEach(error ->
                                logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                        }
                    });
            }
        });
}

```

```

        });

        }
        return CompletableFuture.completedFuture(null);
    })
    .thenCompose(ignored -> {
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();
        return s3AsyncClient.deleteBucket(deleteBucketRequest);
    })
    .thenCompose(ignored -> {
        S3AsyncWaiter waiter = s3AsyncClient.waiter();
        HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
        return waiter.waitUntilBucketNotExists(headBucketRequest);
    })
    .whenComplete((ignored, exception) -> {
        if (exception != null) {
            Throwable cause = exception.getCause();
            if (cause instanceof S3Exception) {
                throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
            }
            throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
        }
        logger.info("Bucket deleted successfully: " + bucketName);
    });
    }

    /**
     * Lists the objects in an S3 bucket asynchronously.
     *
     * @param s3Client the S3 async client to use for the operation
     * @param bucketName the name of the S3 bucket containing the objects to list
     * @return a {@link CompletableFuture} that contains the list of object keys
in the specified bucket
     */
    public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient
s3Client, String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

```

```

        return s3Client.listObjectsV2(request)
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .toList())
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Couldn't list objects in
bucket: " + bucketName, exception);
                }
            });
    }

    /**
     * Retrieves an object from an Amazon S3 bucket asynchronously.
     *
     * @param s3Client the S3 async client to use for the operation
     * @param bucketName the name of the S3 bucket containing the object
     * @param keyName the unique identifier (key) of the object to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the
     object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
     */
    public CompletableFuture<ResponseBytes<GetObjectResponse>>
    getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        // Get the object asynchronously and transform it into a byte array
        return s3Client.getObject(objectRequest,
            AsyncResponseTransformer.toBytes())
            .exceptionally(exception -> {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchKeyException) {
                    throw new CompletionException("Failed to get the object.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to get the object",
exception);
            });
    }

    /**

```

```

    * Asynchronously copies an object from one S3 bucket to another.
    *
    * @param s3Client      the S3 async client to use for the copy
operation
    * @param sourceBucket  the name of the source bucket
    * @param sourceKey     the key of the object to be copied in the source
bucket
    * @param destinationBucket the name of the destination bucket
    * @param destinationKey the key of the copied object in the destination
bucket
    * @return a {@link CompletableFuture} that completes when the copy operation
is finished
    */
    public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey)
{
    CopyObjectRequest copyRequest = CopyObjectRequest.builder()
        .sourceBucket(sourceBucket)
        .sourceKey(sourceKey)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    return s3Client.copyObject(copyRequest)
        .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
        .whenComplete((ignored, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof S3Exception) {
                    throw new CompletionException("Couldn't
copy object '" + sourceKey + "' from bucket '" + sourceBucket + "'
to bucket '" + destinationBucket + "'. Reason: " + ((S3Exception)
cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to copy object",
exception);
            }
        });
}

/**
 * Asynchronously creates a session for the specified S3 bucket.
 *

```

```

    * @param s3Client    the S3 asynchronous client to use for creating the
    session
    * @param bucketName the name of the S3 bucket for which to create the
    session
    * @return a {@link CompletableFuture} that completes when the session is
    created, or throws a {@link CompletionException} if an error occurs
    */
    public CompletableFuture<CreateSessionResponse>
    createSessionAsync(S3AsyncClient s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createSession(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't create the
session. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(),
cause);
                    }
                    throw new CompletionException("Unexpected error occurred
while creating session", exception);
                }
                logger.info("Created session for bucket: " + bucketName);
            });
    }

    /**
    * Creates a new S3 directory bucket in a specified Zone (For example, a
    * specified Availability Zone in this code example).
    *
    * @param s3Client    The asynchronous S3 client used to create the bucket
    * @param bucketName The name of the bucket to be created
    * @param zone        The Availability Zone where the bucket will be created
    * @throws CompletionException if there's an error creating the bucket
    */
    public CompletableFuture<CreateBucketResponse>
    createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String
zone) {
        logger.info("Creating bucket: " + bucketName);
    }

```



```

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
    .location(LocationInfo.builder()
        .type(LocationType.AVAILABILITY_ZONE)
        .name(zone)
        .build())
    .bucket(BucketInfo.builder()
        .type(BucketType.DIRECTORY)
        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
        .build())
    .build();

CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
    .bucket(bucketName)
    .createBucketConfiguration(bucketConfiguration)
    .build();

return s3Client.createBucket(bucketRequest)
    .whenComplete((response, exception) -> {
        if (exception != null) {
            Throwable cause = exception.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                throw new CompletionException("The bucket already exists:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
            }
            throw new CompletionException("Unexpected error occurred
while creating bucket", exception);
        }
        logger.info("Bucket created successfully with location: " +
response.location());
    });
}

/**
 * Creates an S3 bucket asynchronously.
 *
 * @param s3Client    the S3 async client to use for the bucket creation
 * @param bucketName  the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
 *         when the bucket is successfully created
 * @throws CompletionException if there's an error creating the bucket
 */

```

```

    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    return s3Client.createBucket(bucketRequest)
        .thenCompose(response -> {
            S3AsyncWaiter s3Waiter = s3Client.waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();
            return s3Waiter.waitUntilBucketExists(bucketRequestWait);
        })
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof BucketAlreadyExistsException) {
                    throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                }
            }
            logger.info(bucketName + " is ready");
        });
}

/**
 * Uploads an object to an Amazon S3 bucket asynchronously.
 *
 * @param s3Client      the S3 async client to use for the upload
 * @param bucketName    the destination S3 bucket name
 * @param bucketObject  the name of the object to be uploaded
 * @param text          the content to be uploaded as the object
 */
public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(bucketObject)
        .build();

```

```

        return s3Client.putObject(objectRequest,
AsyncRequestBody.fromString(text))
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof NoSuchBucketException) {
                        throw new CompletionException("The S3 bucket does not
exist: " + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                    }
                }
            });
    }

    /**
     * Creates an AWS IAM access key asynchronously for the specified user name.
     *
     * @param userName the name of the IAM user for whom to create the access key
     * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
     */
    public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(userName)
            .build();

        return getIAMAsyncClient().createAccessKey(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Access Key Created.");
                } else {
                    if (exception == null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof IamException) {
                            throw new CompletionException("IAM error while
creating access key: " + cause.getMessage(), cause);
                        } else {
                            throw new CompletionException("Failed to create
access key: " + exception.getMessage(), exception);
                        }
                    }
                }
            });
    }

```

```

        }
    });
}

/**
 * Asynchronously selects an Availability Zone ID from the available EC2
 * zones.
 *
 * @return A {@link CompletableFuture} that resolves to the selected
 * Availability Zone ID.
 * @throws CompletionException if an error occurs during the request or
 * processing.
 */
public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
    DescribeAvailabilityZonesRequest zonesRequest =
DescribeAvailabilityZonesRequest.builder()
        .build();

    return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
        .thenCompose(response -> {
            List<AvailabilityZone> zonesList = response.availabilityZones();
            if (zonesList.isEmpty()) {
                logger.info("No availability zones found.");
                return CompletableFuture.completedFuture(null); // Return
null if no zones are found
            }

            List<String> zoneIds = zonesList.stream()
                .map(AvailabilityZone::zoneId) // Get the zoneId (e.g.,
"usw2-az1")
                .toList();

            return CompletableFuture.supplyAsync(() ->
promptUserForZoneSelection(zonesList, zoneIds))
                .thenApply(selectedZone -> {
                    // Return only the selected Zone ID (e.g., "usw2-az1").
                    return selectedZone.zoneId();
                });
        })
        .whenComplete((result, exception) -> {
            if (exception == null) {
                if (result != null) {
                    logger.info("Selected Availability Zone ID: " + result);
                } else {

```

```

        logger.info("No availability zone selected.");
    }
} else {
    Throwable cause = exception.getCause();
    if (cause instanceof Ec2Exception) {
        throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
    }
    throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
}
});
}

/**
 * Prompts the user to select an Availability Zone from the given list.
 *
 * @param zonesList the list of Availability Zones
 * @param zoneIds the list of zone IDs
 * @return the selected Availability Zone
 */
private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String>
zoneIds) {
    Scanner scanner = new Scanner(System.in);
    int index = -1;

    while (index < 0 || index >= zoneIds.size()) {
        logger.info("Select an availability zone:");
        IntStream.range(0, zoneIds.size()).forEach(i ->
            logger.info(i + ": " + zoneIds.get(i))
        );

        logger.info("Enter the number corresponding to your choice: ");
        if (scanner.hasNextInt()) {
            index = scanner.nextInt();
        } else {
            scanner.next();
        }
    }

    AvailabilityZone selectedZone = zonesList.get(index);
    logger.info("You selected: " + selectedZone.zoneId());
    return selectedZone;
}

```

```

    }

    /**
     * Asynchronously sets up a new VPC, including creating the VPC, finding the
     * associated route table, and
     * creating a VPC endpoint for the S3 service.
     *
     * @return a {@link CompletableFuture} that, when completed, contains a
     * AbstractMap with the
     *      * VPC ID and VPC endpoint ID.
     */
    public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
    setupVPCAsync() {
        String cidr = "10.0.0.0/16";
        CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
            .cidrBlock(cidr)
            .build();

        return getEc2AsyncClient().createVpc(vpcRequest)
            .thenCompose(vpcResponse -> {
                String vpcId = vpcResponse.vpc().vpcId();
                logger.info("VPC Created: {}", vpcId);

                Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
                DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .vpcIds(vpcId)
                    .build();

                return waiter.waitUntilVpcAvailable(request)
                    .thenApply(waiterResponse -> vpcId);
            })
            .thenCompose(vpcId -> {
                Filter filter = Filter.builder()
                    .name("vpc-id")
                    .values(vpcId)
                    .build();

                DescribeRouteTablesRequest describeRouteTablesRequest =
                DescribeRouteTablesRequest.builder()
                    .filters(filter)
                    .build();

                return
                getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)

```

```

        .thenApply(routeTablesResponse -> {
            if (routeTablesResponse.routeTables().isEmpty()) {
                throw new CompletionException("No route tables found
for VPC: " + vpcId, null);
            }
            String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
            logger.info("Route table found: {}", routeTableId);
            return new AbstractMap.SimpleEntry<>(vpcId,
routeTableId);
        });
    })
    .thenCompose(vpcAndRouteTable -> {
        String vpcId = vpcAndRouteTable.getKey();
        String routeTableId = vpcAndRouteTable.getValue();
        Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
        String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());

        CreateVpcEndpointRequest endpointRequest =
CreateVpcEndpointRequest.builder()
            .vpcId(vpcId)
            .routeTableIds(routeTableId)
            .serviceName(serviceName)
            .build();

        return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
            .thenApply(vpcEndpointResponse -> {
                String vpcEndpointId =
vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                logger.info("VPC Endpoint created: {}", vpcEndpointId);
                return new AbstractMap.SimpleEntry<>(vpcId,
vpcEndpointId);
            });
    })
    .exceptionally(exception -> {
        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
        if (cause instanceof Ec2Exception) {
            logger.error("EC2 error during VPC setup: {}",
cause.getMessage(), cause);
            throw new CompletionException("EC2 error during VPC setup: "
+ cause.getMessage(), cause);
        }
    });
}

```

```
        }

        logger.error("VPC setup failed: {}", cause.getMessage(), cause);
        throw new CompletionException("VPC setup failed: " +
            cause.getMessage(), cause);
    });
}
}
```

- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObject](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

PHP

SDK for PHP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Run a scenario demonstrating the basics of Amazon S3 directory buckets and S3 Express One Zone.

```
echo "\n";
echo "-----\n";
echo "Welcome to the Amazon S3 Express Basics demo using PHP!\n";
echo "-----\n";
```



```

        // Change these both of these values to use a different region/
availability zone.
        $region = "us-west-2";
        $az = "usw2-az1";

        $this->s3Service = new S3Service(new S3Client(['region' => $region]));
        $this->iamService = new IAMService(new IamClient(['region' => $region]));

        $uuid = uniqid();

        echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
in this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute
resources in the same AZ as your Directory
bucket.\n
INTRO;

        pressEnter();
        // 1. Configure a gateway VPC endpoint. This is the recommended method to
allow S3 Express One Zone traffic without
        // the need to pass through an internet gateway or NAT device.
        echo "\n";
        echo "1. First, we'll set up a new VPC and VPC Endpoint if this program
is running in an EC2 instance in the same AZ as your Directory buckets will be.
\n";

        $ec2Choice = testable_readline("Are you running this in an EC2 instance
located in the same AZ as your intended Directory buckets? Enter Y/y to setup a
VPC Endpoint, or N/n/blank to skip this section.");
        if($ec2Choice == "Y" || $ec2Choice == "y") {
            echo "Great! Let's set up a VPC, retrieve the Route Table from it,
and create a VPC Endpoint to connect the S3 Client to.\n";
            pressEnter();
            $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
$region]));
            $cidr = "10.0.0.0/16";
            $vpc = $this->ec2Service->createVpc($cidr);
            $this->resources['vpcId'] = $vpc['VpcId'];

            $this->ec2Service->waitForVpcAvailable($vpc['VpcId']);

            $routeTable = $this->ec2Service->describeRouteTables([], [
                [

```

```

        'Name' => "vpc-id",
        'Values' => [$vpc['VpcId']],
    ],
]);

    $serviceName = "com.amazonaws." . $this->ec2Service->getRegion() .
".s3express";
    $vpcEndpoint = $this->ec2Service->createVpcEndpoint($serviceName,
    $vpc['VpcId'], [$routeTable[0]]);
    $this->resources['vpcEndpointId'] = $vpcEndpoint['VpcEndpointId'];
} else {
    echo "Skipping the VPC setup. Don't forget to use this in production!
\n";
}

// 2. Policies, user, and roles with CDK.
echo "\n";
echo "2. Policies, users, and roles with CDK.\n";
echo "Now, we'll set up some policies, roles, and a user. This user will
only have permissions to do S3 Express One Zone actions.\n";
pressEnter();

$this->cloudFormationClient = new CloudFormationClient([]);
$stackName = "cfn-stack-s3-express-basics-" . uniqid();
$file = file_get_contents(__DIR__ . "/../../../../../resources/cfn/
s3_express_basics/s3_express_template.yml");
$result = $this->cloudFormationClient->createStack([
    'StackName' => $stackName,
    'TemplateBody' => $file,
    'Capabilities' => ['CAPABILITY_IAM'],
]);
$waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
['StackName' => $stackName]);
try {
    $waiter->promise()->wait();
} catch (CloudFormationException $caught) {
    echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
    throw $caught;
}
$this->resources['stackName'] = $stackName;
$stackInfo = $this->cloudFormationClient->describeStacks([
    'StackName' => $result['StackId'],
]);

```

```

$expressUserName = "";
$regularUserName = "";
foreach($stackInfo['Stacks'][0]['Outputs'] as $output) {
    if ($output['OutputKey'] == "RegularUser") {
        $regularUserName = $output['OutputValue'];
    }
    if ($output['OutputKey'] == "ExpressUser") {
        $expressUserName = $output['OutputValue'];
    }
}
$regularKey = $this->iamService->createAccessKey($regularUserName);
$regularCredentials = new Credentials($regularKey['AccessKeyId'],
$regularKey['SecretAccessKey']);
$expressKey = $this->iamService->createAccessKey($expressUserName);
$expressCredentials = new Credentials($expressKey['AccessKeyId'],
$expressKey['SecretAccessKey']);

// 3. Create an additional client using the credentials with S3 Express
permissions.
echo "\n";
echo "3. Create an additional client using the credentials with S3
Express permissions.\n";
echo "This client is created with the credentials associated with the
user account with the S3 Express policy attached, so it can perform S3 Express
operations.\n";
pressEnter();
$s3RegularClient = new S3Client([
    'Region' => $region,
    'Credentials' => $regularCredentials,
]);
$s3RegularService = new S3Service($s3RegularClient);
$s3ExpressClient = new S3Client([
    'Region' => $region,
    'Credentials' => $expressCredentials,
]);
$s3ExpressService = new S3Service($s3ExpressClient);
echo "All the roles and policies were created an attached to the user.
Then, a new S3 Client and Service were created using that user's credentials.
\n";
echo "We can now use this client to make calls to S3 Express operations.
Keeping permissions in mind (and adhering to least-privilege) is crucial to S3
Express.\n";
pressEnter();

```

```
// 4. Create two buckets.
echo "\n";
echo "3. Create two buckets.\n";
echo "Now we will create a Directory bucket, which is the linchpin of the
S3 Express One Zone service.\n";
echo "Directory buckets behave in different ways from regular S3 buckets,
which we will explore here.\n";
echo "We'll also create a normal bucket, put an object into the normal
bucket, and copy it over to the Directory bucket.\n";
pressEnter();

// Create a directory bucket. These are different from normal S3 buckets
in subtle ways.
$directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-
s3";
echo "Now, let's create the actual Directory bucket, as well as a regular
bucket.\n";
pressEnter();
$s3ExpressService->createBucket($directoryBucketName, [
    'CreateBucketConfiguration' => [
        'Bucket' => [
            'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
            'DataRedundancy' => "SingleAvailabilityZone",
        ],
        'Location' => [
            'Name' => $az,
            'Type' => "AvailabilityZone",
        ],
    ],
]);
$this->resources['directoryBucketName'] = $directoryBucketName;

// Create a normal bucket.
$normalBucketName = "normal-bucket-$uuid";
$s3RegularService->createBucket($normalBucketName);
$this->resources['normalBucketName'] = $normalBucketName;
echo "Great! Both buckets were created.\n";
pressEnter();

// 5. Create an object and copy it over.
echo "\n";
echo "5. Create an object and copy it over.\n";
```

```
    echo "We'll create a basic object consisting of some text and upload it
to the normal bucket.\n";
    echo "Next, we'll copy the object into the Directory bucket using the
regular client.\n";
    echo "This works fine, because Copy operations are not restricted for
Directory buckets.\n";
    pressEnter();

    $objectKey = "basic-text-object";
    $s3RegularService->putObject($normalBucketName, $objectKey, $args =
['Body' => "Look Ma, I'm a bucket!"]);
    $this->resources['objectKey'] = $objectKey;

    // Create a session to access the directory bucket. The SDK Client will
automatically refresh this as needed.
    $s3ExpressService->createSession($directoryBucketName);
    $s3ExpressService->copyObject($directoryBucketName, $objectKey,
"$normalBucketName/$objectKey");

    echo "It worked! It's important to remember the user permissions when
interacting with Directory buckets.\n";
    echo "Instead of validating permissions on every call as normal buckets
do, Directory buckets utilize the user credentials and session token to
validate.\n";
    echo "This allows for much faster connection speeds on every call. For
single calls, this is low, but for many concurrent calls, this adds up to a lot
of time saved.\n";
    pressEnter();

    // 6. Demonstrate performance difference.
    echo "\n";
    echo "6. Demonstrate performance difference.\n";
    $downloads = 1000;
    echo "Now, let's do a performance test. We'll download the same object
from each bucket $downloads times and compare the total time needed. Note: the
performance difference will be much more pronounced if this example is run in an
EC2 instance in the same AZ as the bucket.\n";
    $downloadChoice = testable_readline("If you would like to download each
object $downloads times, press enter. Otherwise, enter a custom amount and press
enter.");
    if($downloadChoice && is_numeric($downloadChoice) && $downloadChoice <
1000000){ // A million is enough. I promise.
        $downloads = $downloadChoice;
    }
}
```

```
// Download the object $downloads times from each bucket and time it to
demonstrate the speed difference.
$directoryStartTime = hrtime(true);
for($i = 0; $i < $downloads; ++$i){
    $s3ExpressService->getObject($directoryBucketName, $objectKey);
}
$directoryEndTime = hrtime(true);
$directoryTimeDiff = $directoryEndTime - $directoryStartTime;

$normalStartTime = hrtime(true);
for($i = 0; $i < $downloads; ++$i){
    $s3RegularService->getObject($normalBucketName, $objectKey);
}
$normalEndTime = hrtime(true);
$normalTimeDiff = $normalEndTime - $normalStartTime;

echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
normal bucket took $normalTimeDiff.\n";
echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
" nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
seconds.\n";
pressEnter();

// 7. Populate the buckets to show the lexicographical difference.
echo "\n";
echo "7. Populate the buckets to show the lexicographical difference.\n";
echo "Now let's explore how Directory buckets store objects in a
different manner to regular buckets.\n";
echo "The key is in the name \"Directory!\"\n";
echo "Where regular buckets store their key/value pairs in a flat manner,
Directory buckets use actual directories/folders.\n";
echo "This allows for more rapid indexing, traversing, and therefore
retrieval times!\n";
echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
echo "This structural difference also causes ListObjects to behave
differently, which can cause unexpected results.\n";
echo "Let's add a few more objects with layered directories as see how
the output of ListObjects changes.\n";
pressEnter();

// Populate a few more files in each bucket so that we can use
ListObjects and show the difference.
```

```

    $otherObject = "other/$objectKey";
    $altObject = "alt/$objectKey";
    $otherAltObject = "other/alt/$objectKey";
    $s3ExpressService->putObject($directoryBucketName, $otherObject);
    $s3RegularService->putObject($normalBucketName, $otherObject);
    $this->resources['otherObject'] = $otherObject;
    $s3ExpressService->putObject($directoryBucketName, $altObject);
    $s3RegularService->putObject($normalBucketName, $altObject);
    $this->resources['altObject'] = $altObject;
    $s3ExpressService->putObject($directoryBucketName, $otherAltObject);
    $s3RegularService->putObject($normalBucketName, $otherAltObject);
    $this->resources['otherAltObject'] = $otherAltObject;

    $listDirectoryBucket = $s3ExpressService-
>listObjects($directoryBucketName);
    $listNormalBucket = $s3RegularService->listObjects($normalBucketName);

    // Directory bucket content
    echo "Directory bucket content\n";
    foreach($listDirectoryBucket['Contents'] as $result){
        echo $result['Key'] . "\n";
    }

    // Normal bucket content
    echo "\nNormal bucket content\n";
    foreach($listNormalBucket['Contents'] as $result){
        echo $result['Key'] . "\n";
    }

    echo "Notice how the normal bucket lists objects in lexicographical
order, while the directory bucket does not. This is because the normal bucket
considers the whole \"key\" to be the object identifies, while the directory
bucket actually creates directories and uses the object \"key\" as a path to the
object.\n";
    pressEnter();

    echo "\n";
    echo "That's it for our tour of the basic operations for S3 Express One
Zone.\n";
    $cleanUp = testable_readline("Would you like to delete all the resources
created during this demo? Enter Y/y to delete all the resources.");
    if($cleanUp){
        $this->cleanUp();
    }

```

```
namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }
}
```



```
}

public function setClient(S3Client $client)
{
    $this->client = $client;
}

public function emptyAndDeleteBucket($bucketName, array $args = [])
{
    try {
        $objects = $this->listAllObjects($bucketName, $args);
        $this->deleteObjects($bucketName, $objects, $args);
        if ($this->verbose) {
            echo "Deleted all objects and folders from $bucketName.\n";
        }
        $this->deleteBucket($bucketName, $args);
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "\nPlease fix error with bucket deletion before continuing.
\n";
        }
        throw $exception;
    }
}

public function createBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->createBucket($parameters);
        if ($this->verbose) {
            echo "Created the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket creation before continuing.";
        }
    }
}
```

```
        throw $exception;
    }
}

public function putObject(string $bucketName, string $key, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
    $args);
    try {
        $this->client->putObject($parameters);
        if ($this->verbose) {
            echo "Uploaded the object named: $key to the bucket named:
    $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $key in $bucketName with error:
    {$exception->getMessage()}\n";
            echo "Please fix error with object uploading before continuing.";
        }
        throw $exception;
    }
}

public function getObject(string $bucketName, string $key, array $args = []):
Result
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
    $args);
    try {
        $object = $this->client->getObject($parameters);
        if ($this->verbose) {
            echo "Downloaded the object named: $key to the bucket named:
    $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to download $key from $bucketName with error:
    {$exception->getMessage()}\n";
        }
    }
}
```

```

        echo "Please fix error with object downloading before
continuing.";
    }
    throw $exception;
}
return $object;
}

public function copyObject($bucketName, $key, $copySource, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
    try {
        $this->client->copyObject($parameters);
        if ($this->verbose) {
            echo "Copied the object from: $copySource in $bucketName to:
$key.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object copying before continuing.";
        }
        throw $exception;
    }
}

public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
    try {
        $objects = $this->client->listObjectsV2($parameters);
        if ($this->verbose) {
            echo "Retrieved the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {

```

```
        echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
        echo "Please fix error with list objects before continuing.";
    }
    throw $exception;
}
return $objects;
}

public function listAllObjects($bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);

    $contents = [];
    $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

    foreach ($paginator as $result) {
        if($result['KeyCount'] == 0){
            break;
        }
        foreach ($result['Contents'] as $object) {
            $contents[] = $object;
        }
    }
    return $contents;
}

public function deleteObjects(string $bucketName, array $objects, array $args
= [])
{
    $listOfObjects = array_map(
        function ($object) {
            return ['Key' => $object];
        },
        array_column($objects, 'Key')
    );
    if(!$listOfObjects){
        return;
    }
}
```

```

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function deleteObject(string $bucketName, string $fileName, array
$args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);

```

```

        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }
}

public function listBuckets(array $args = [])
{
    try {
        $buckets = $this->client->listBuckets($args);
        if ($this->verbose) {
            echo "Retrieved all " . count($buckets) . "\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket lists before continuing.";
        }
        throw $exception;
    }
    return $buckets;
}

public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
{
    $request = $this->client->createPresignedRequest($command, $expires,
$options);
    try {
        $presignedUrl = (string)$request->getUri();
    }
}

```

```
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }

    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
    }
}
```

- For API details, see the following topics in *Amazon SDK for PHP API Reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Run a scenario demonstrating the basics of Amazon S3 directory buckets and S3 Express One Zone.

```
class S3ExpressScenario:
    """Runs an interactive scenario that shows how to get started with S3
    Express."""

    def __init__(
        self,
        cloud_formation_resource: ServiceResource,
        ec2_client: client,
        iam_client: client,
    ):
        self.cloud_formation_resource = cloud_formation_resource
        self.ec2_client = ec2_client
        self.iam_client = iam_client
        self.region = ec2_client.meta.region_name
        self.stack = None
        self.vpc_id = None
        self.vpc_endpoint_id = None
        self.regular_bucket_name = None
        self.directory_bucket_name = None
        self.s3_express_wrapper = None
        self.s3_regular_wrapper = None

    def s3_express_scenario(self):
        """
        Runs the scenario.
        """
        print("")
        print_dashes()
```



```

        print("Welcome to the Amazon S3 Express Basics demo using Python (Boto
3)!")
        print_dashes()
        print(
            """

```

Let's get started! First, please note that S3 Express One Zone works best when working within the AWS infrastructure, specifically when working in the same Availability Zone. To see the best results in this example and when you implement Directory buckets into your infrastructure, it is best to put your compute resources in the same AZ as your Directory bucket.

```

        """
    )
    press_enter_to_continue()

    # Create an optional VPC and create 2 IAM users.
    express_user_name, regular_user_name = self.create_vpc_and_users()

    # Set up two S3 clients, one regular and one express, and two buckets,
    one regular and one express.
    self.setup_clients_and_buckets(express_user_name, regular_user_name)

    # Create an S3 session for the express S3 client and add objects to the
    buckets.
    bucket_object = self.create_session_and_add_objects()

    # Demonstrate performance differences between regular and express
    buckets.
    self.demonstrate_performance(bucket_object)

    # Populate the buckets to show the lexicographical difference between
    regular and express buckets.
    self.show_lexicographical_differences(bucket_object)

    print("")
    print("That's it for our tour of the basic operations for S3 Express One
Zone.")

    if q.ask(
        "Would you like to delete all the resources created during this demo
(y/n)? ",
        q.is_yesno,
    ):

```

```

        self.cleanup()

    def create_vpc_and_users(self) -> None:
        """
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
        without.
        """
        # Configure a gateway VPC endpoint. This is the recommended method to
        allow S3 Express One Zone traffic without
        # the need to pass through an internet gateway or NAT device.
        print(
            """
1. First, we'll set up a new VPC and VPC Endpoint if this program is running in
an EC2 instance in the same AZ as your
Directory buckets will be. Are you running this in an EC2 instance located in the
same AZ as your intended Directory buckets?
"""
        )
        if q.ask("Do you want to setup a VPC Endpoint? (y/n) ", q.is_yesno):
            print(
                "Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to."
            )
            self.setup_vpc()
            press_enter_to_continue()
        else:
            print("Skipping the VPC setup. Don't forget to use this in
production!")
            print(
                """
2. Policies, users, and roles with CDK.
Now, we'll set up some policies, roles, and a user. This user will only have
permissions to do S3 Express One Zone actions.
"""
            )
            press_enter_to_continue()
            stack_name = f"cfn-stack-s3-express-basics--{uuid.uuid4()}"
            template_as_string = S3ExpressScenario.get_template_as_string()
            self.stack = self.deploy_cloudformation_stack(stack_name,
template_as_string)
            regular_user_name = None
            express_user_name = None
            outputs = self.stack.outputs

```

```

    for output in outputs:
        if output.get("OutputKey") == "RegularUser":
            regular_user_name = output.get("OutputValue")
        elif output.get("OutputKey") == "ExpressUser":
            express_user_name = output.get("OutputValue")
    if not regular_user_name or not express_user_name:
        error_string = f"""
        Failed to retrieve required outputs from CloudFormation stack.
        'regular_user_name'={regular_user_name},
'express_user_name'={express_user_name}
        """
        logger.error(error_string)
        raise ValueError(error_string)
    return express_user_name, regular_user_name

def setup_clients_and_buckets(
    self, express_user_name: str, regular_user_name: str
) -> None:
    """
    Set up two S3 clients, one regular and one express, and two buckets, one
    regular and one express.
    :param express_user_name: The name of the user with S3 Express
    permissions.
    :param regular_user_name: The name of the user with regular S3
    permissions.
    """
    regular_credentials = self.create_access_key(regular_user_name)
    express_credentials = self.create_access_key(express_user_name)
    # 3. Create an additional client using the credentials with S3 Express
    permissions.
    print(
        """
3. Create an additional client using the credentials with S3 Express permissions.
This client is created with the
credentials associated with the user account with the S3 Express policy attached,
so it can perform S3 Express operations.
    """
    )
    press_enter_to_continue()
    s3_regular_client = self.create_s3_client_with_access_key_credentials(
        regular_credentials
    )
    self.s3_regular_wrapper = S3ExpressWrapper(s3_regular_client)
    s3_express_client = self.create_s3_client_with_access_key_credentials(

```

```

        express_credentials
    )
    self.s3_express_wrapper = S3ExpressWrapper(s3_express_client)
    print(
        """

```

All the roles and policies were created and attached to the user. Then a new S3 Client were created using that user's credentials. We can now use this client to make calls to S3 Express operations. Keeping permissions in mind (and adhering to least-privilege) is crucial to S3 Express.

```

    """
    )
    press_enter_to_continue()
    # 4. Create two buckets.
    print(
        """

```

3. Create two buckets.

Now we will create a Directory bucket which is the linchpin of the S3 Express One Zone service. Directory buckets

behave in different ways from regular S3 buckets which we will explore here.

We'll also create a normal bucket, put an object into the normal bucket, and copy it over to the Directory bucket.

```

    """
    )

    # Create a directory bucket. These are different from normal S3 buckets
    in subtle ways.
    bucket_prefix = q.ask(
        "Enter a bucket name prefix that will be used for both buckets: ",
        q.re_match(r"[a-z0-9](?:[a-z0-9-\.]*)[a-z0-9]$",)
    )

    # Some availability zones are not supported for Directory buckets. We'll
    choose one that is supported.
    print(
        "Now, let's choose an availability zone for the Directory bucket.
    We'll choose one that is supported."
    )
    while True:
        availability_zone = self.select_availability_zone_id(self.region)
        # Construct the parts of a directory bucket name that is made unique
        with a UUID string.
        directory_bucket_suffix = f"--{availability_zone['ZoneId']}--x-s3"

```

```

        max_uuid_length = 63 - len(bucket_prefix) -
len(directory_bucket_suffix) - 1
        bucket_uuid = str(uuid.uuid4()).replace("-", "")[:max_uuid_length]
        directory_bucket_name = (
            f"{bucket_prefix}-{bucket_uuid}{directory_bucket_suffix}"
        )
        regular_bucket_name = f"{bucket_prefix}-regular-{bucket_uuid}"
        configuration = {
            "Bucket": {
                "Type": "Directory",
                "DataRedundancy": "SingleAvailabilityZone",
            },
            "Location": {
                "Name": availability_zone["ZoneId"],
                "Type": "AvailabilityZone",
            },
        }
    press_enter_to_continue()
    print(
        "Now, let's create the actual Directory bucket, as well as a
regular bucket."
    )
    press_enter_to_continue()
    try:
        self.s3_express_wrapper.create_bucket(
            directory_bucket_name, configuration
        )
        break
    except ClientError as client_error:
        if client_error.response["Error"]["Code"] == "InvalidBucketName":
            print(
                f"Bucket '{directory_bucket_name}' is invalid. This may
be because of selected availability zone."
            )
            if q.ask(
                "Would you like to select a different availability zone?"
            ),
                q.is_yesno,
            ):
                continue
            else:
                raise
        else:
            raise

```

```

        print(f"Created directory bucket, '{directory_bucket_name}'")
        self.directory_bucket_name = directory_bucket_name

        self.s3_regular_wrapper.create_bucket(regular_bucket_name)
        print(f"Created regular bucket, '{regular_bucket_name}'")
        self.regular_bucket_name = regular_bucket_name
        print("Great! Both buckets were created.")
        press_enter_to_continue()

    def create_session_and_add_objects(self) -> None:
        """
        Create a session for the express S3 client and add objects to the
        buckets.
        """
        print(
            """
5. Create an object and copy it over.
We'll create a basic object consisting of some text and upload it to the normal
bucket. Next we'll copy the object
into the Directory bucket using the regular client. This works fine because copy
operations are not restricted for
Directory buckets.
            """
        )
        press_enter_to_continue()
        bucket_object = "basic-text-object"
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, bucket_object, "Look Ma, I'm a bucket!"
        )
        self.s3_express_wrapper.create_session(self.directory_bucket_name)
        self.s3_express_wrapper.copy_object(
            self.regular_bucket_name,
            bucket_object,
            self.directory_bucket_name,
            bucket_object,
        )
        print(
            """
It worked! It's important to remember the user permissions when interacting with
Directory buckets. Instead of validating
permissions on every call as normal buckets do, Directory buckets utilize the
user credentials and session token to validate.
This allows for much faster connection speeds on every call. For single calls,
this is low, but for many concurrent calls
            """
        )

```

```
this adds up to a lot of time saved.
```

```
"""
    )
    press_enter_to_continue()
    return bucket_object

def demonstrate_performance(self, bucket_object: str) -> None:
    """
    Demonstrate performance differences between regular and Directory
    buckets.
    :param bucket_object: The name of the object to download from each
    bucket.
    """
    print("")
    print("6. Demonstrate performance difference.")
    print(
        """
```

Now, let's do a performance test. We'll download the same object from each bucket 'downloads' times

and compare the total time needed. Note: the performance difference will be much more pronounced if this

example is run in an EC2 instance in the same Availability Zone as the bucket.

```
"""
    )
    downloads = 1000
    print(
        f"The number of downloads of the same object for this example is set
    at {downloads}."
    )
    if q.ask("Would you like to download a different number? (y/n) ",
    q.is_yesno):
        max_downloads = 1000000
        downloads = q.ask(
            f"Enter a number between 1 and {max_downloads} for the number of
    downloads: ",
            q.is_int,
            q.in_range(1, max_downloads),
        )
        # Download the object 'downloads' times from each bucket and time it to
    demonstrate the speed difference.
        print("Downloading from the Directory bucket.")
        directory_time_start = time.time_ns()

        for index in range(downloads):
```

```

        if index % 10 == 0:
            print(f"Download {index} of {downloads}")

        self.s3_express_wrapper.get_object(
            self.directory_bucket_name, bucket_object
        )

    directory_time_difference = time.time_ns() - directory_time_start
    print("Downloading from the normal bucket.")
    normal_time_start = time.time_ns()

    for index in range(downloads):
        if index % 10 == 0:
            print(f"Download {index} of {downloads}")
            self.s3_regular_wrapper.get_object(self.regular_bucket_name,
            bucket_object)

    normal_time_difference = time.time_ns() - normal_time_start
    print(
        f"The directory bucket took {directory_time_difference} nanoseconds,
while the normal bucket took {normal_time_difference}."
    )
    difference = normal_time_difference - directory_time_difference
    print(f"That's a difference of {difference} nanoseconds, or")
    print(f"{{(difference) / 1000000000}} seconds.")
    if difference < 0:
        print(
            "The directory buckets were slower. This can happen if you are
not running on the cloud within a vpc."
        )
        press_enter_to_continue()

    def show_lexicographical_differences(self, bucket_object: str) -> None:
        """
        Show the lexicographical difference between Directory buckets and regular
        buckets.

        This is done by creating a few objects in each bucket and listing them to
        show the difference.

        :param bucket_object: The object to use for the listing operations.
        """
        print(
            """
7. Populate the buckets to show the lexicographical difference.

```


Now let's explore how Directory buckets store objects in a different manner to regular buckets. The key is in the name "Directory". Where regular buckets store their key/value pairs in a flat manner, Directory buckets use actual directories/folders. This allows for more rapid indexing, traversing, and therefore retrieval times! The more segmented your bucket is, with lots of directories, sub-directories, and objects, the more efficient it becomes. This structural difference also causes ListObjects to behave differently, which can cause unexpected results. Let's add a few more objects with layered directories to see how the output of ListObjects changes.

```

"""
)
press_enter_to_continue()
# Populate a few more files in each bucket so that we can use ListObjects
and show the difference.
other_object = f"other/{bucket_object}"
alt_object = f"alt/{bucket_object}"
other_alt_object = f"other/alt/{bucket_object}"
self.s3_regular_wrapper.put_object(self.regular_bucket_name,
other_object, "")
self.s3_express_wrapper.put_object(self.directory_bucket_name,
other_object, "")
self.s3_regular_wrapper.put_object(self.regular_bucket_name, alt_object,
"")
self.s3_express_wrapper.put_object(self.directory_bucket_name,
alt_object, "")
self.s3_regular_wrapper.put_object(
    self.regular_bucket_name, other_alt_object, ""
)
self.s3_express_wrapper.put_object(
    self.directory_bucket_name, other_alt_object, ""
)
)
directory_bucket_objects = self.s3_express_wrapper.list_objects(
    self.directory_bucket_name
)

regular_bucket_objects = self.s3_regular_wrapper.list_objects(
    self.regular_bucket_name
)

print("Directory bucket content")
for bucket_object in directory_bucket_objects:
    print(f"    {bucket_object['Key']}")

```

```

    print("Normal bucket content")
    for bucket_object in regular_bucket_objects:
        print(f"    {bucket_object['Key']}")
    print(
        """

```

Notice how the normal bucket lists objects in lexicographical order, while the directory bucket does not. This is because the normal bucket considers the whole "key" to be the object identifier, while the directory bucket actually creates directories and uses the object "key" as a path to the object.

```

        """
    )
    press_enter_to_continue()

def cleanup(self) -> None:
    """
    Delete resources created by this scenario.
    """
    if self.directory_bucket_name is not None:
        self.s3_express_wrapper.delete_bucket_and_objects(
            self.directory_bucket_name
        )
        print(f"Deleted directory bucket, '{self.directory_bucket_name}'")
        self.directory_bucket_name = None

    if self.regular_bucket_name is not None:

self.s3_regular_wrapper.delete_bucket_and_objects(self.regular_bucket_name)
        print(f"Deleted regular bucket, '{self.regular_bucket_name}'")
        self.regular_bucket_name = None

    if self.stack is not None:
        self.destroy_cloudformation_stack(self.stack)
        self.stack = None

    self.tear_down_vpc()

def create_access_key(self, user_name: str) -> dict[str, any]:
    """
    Creates an access key for the user.
    :param user_name: The name of the user.
    :return: The access key for the user.
    """
    try:

```

```

        access_key = self.iam_client.create_access_key(UserName=user_name)
        return access_key["AccessKey"]
    except ClientError as client_error:
        logging.error(
            "Couldn't create the access key. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def create_s3_client_with_access_key_credentials(
    self, access_key: dict[str, any]
) -> client:
    """
    Creates an S3 client with access key credentials.
    :param access_key: The access key for the user.
    :return: The S3 Express One Zone client.
    """
    try:
        s3_express_client = boto3.client(
            "s3",
            aws_access_key_id=access_key["AccessKeyId"],
            aws_secret_access_key=access_key["SecretAccessKey"],
            region_name=self.region,
        )
        return s3_express_client
    except ClientError as client_error:
        logging.error(
            "Couldn't create the S3 Express One Zone client. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def select_availability_zone_id(self, region: str) -> dict[str, any]:
    """
    Selects an availability zone.
    :param region: The region to select the availability zone from.
    :return: The availability zone dictionary.
    """
    try:
        response = self.ec2_client.describe_availability_zones(
            Filters=[{"Name": "region-name", "Values": [region]}]
        )
        availability_zones = response["AvailabilityZones"]
        zone_names = [zone["ZoneName"] for zone in availability_zones]

```

```

        index = q.choose("Select an availability zone: ", zone_names)
        return availability_zones[index]
    except ClientError as client_error:
        logging.error(
            "Couldn't describe availability zones. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def deploy_cloudformation_stack(
    self, stack_name: str, cfn_template: str
) -> ServiceResource:
    """
    Deploys prerequisite resources used by the scenario. The resources are
    defined in the associated `cfn_template.yaml` AWS CloudFormation script
    and are deployed
    as a CloudFormation stack, so they can be easily managed and destroyed.

    :param stack_name: The name of the CloudFormation stack.
    :param cfn_template: The CloudFormation template as a string.
    :return: The CloudFormation stack resource.
    """
    print(f"Deploying CloudFormation stack: {stack_name}.")
    stack = self.cloud_formation_resource.create_stack(
        StackName=stack_name,
        TemplateBody=cfn_template,
        Capabilities=["CAPABILITY_NAMED_IAM"],
    )
    print(f"CloudFormation stack creation started: {stack_name}")
    print("Waiting for CloudFormation stack creation to complete...")
    waiter = self.cloud_formation_resource.meta.client.get_waiter(
        "stack_create_complete"
    )
    waiter.wait(StackName=stack.name)
    stack.load()
    print("CloudFormation stack creation complete.")

    return stack

def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
    """
    Destroys the resources managed by the CloudFormation stack, and the
    CloudFormation
    stack itself.

```

```

        :param stack: The CloudFormation stack that manages the example
resources.
        """
        try:
            print(
                f"CloudFormation stack '{stack.name}' is being deleted. This may
take a few minutes."
            )
            stack.delete()
            waiter = self.cloud_formation_resource.meta.client.get_waiter(
                "stack_delete_complete"
            )
            waiter.wait(StackName=stack.name)
            print(f"CloudFormation stack '{stack.name}' has been deleted.")
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the CloudFormation stack. Here's why: %s",
                client_error.response["Error"]["Message"],
            )

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory,
"s3_express_template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def setup_vpc(self):
        cidr = "10.0.0.0/16"
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr)
            self.vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[self.vpc_id])
            print(f"Created vpc {self.vpc_id}")

        except ClientError as client_error:
            logging.error(

```

```

        "Couldn't create the vpc. Here's why: %s",
        client_error.response["Error"]["Message"],
    )
    raise
try:
    response = self.ec2_client.describe_route_tables(
        Filters=[{"Name": "vpc-id", "Values": [self.vpc_id]}]
    )
    route_table_id = response["RouteTables"][0]["RouteTableId"]
    service_name = f"com.amazonaws.
{self.ec2_client.meta.region_name}.s3express"

    response = self.ec2_client.create_vpc_endpoint(
        VpcId=self.vpc_id,
        RouteTableIds=[route_table_id],
        ServiceName=service_name,
    )
    self.vpc_endpoint_id = response["VpcEndpoint"]["VpcEndpointId"]
    print(f"Created vpc endpoint {self.vpc_endpoint_id}")

except ClientError as client_error:
    logging.error(
        "Couldn't create the vpc endpoint. Here's why: %s",
        client_error.response["Error"]["Message"],
    )
    raise

def tear_down_vpc(self) -> None:
    if self.vpc_endpoint_id is not None:
        try:
            self.ec2_client.delete_vpc_endpoints(
                VpcEndpointIds=[self.vpc_endpoint_id]
            )
            print(f"Deleted vpc endpoint {self.vpc_endpoint_id}.")
            self.vpc_endpoint_id = None
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the vpc endpoint %s. Here's why: %s",
                self.vpc_endpoint_id,
                client_error.response["Error"]["Message"],
            )
    if self.vpc_id is not None:
        try:
            self.ec2_client.delete_vpc(VpcId=self.vpc_id)

```

```

        print(f"Deleted vpc {self.vpc_id}")
        self.vpc_id = None
    except ClientError as client_error:
        logging.error(
            "Couldn't delete the vpc %s. Here's why: %s",
            self.vpc_id,
            client_error.response["Error"]["Message"],
        )

```

A wrapper class for Amazon S3 Express SDK functions.

```

class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
    interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                           access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def create_bucket(
        self, bucket_name: str, bucket_configuration: dict[str, any] = None
    ) -> None:

```

```

"""
Creates a bucket.
:param bucket_name: The name of the bucket.
:param bucket_configuration: The optional configuration for the bucket.
"""
try:
    params = {"Bucket": bucket_name}
    if bucket_configuration:
        params["CreateBucketConfiguration"] = bucket_configuration

    self.s3_client.create_bucket(**params)
except ClientError as client_error:
    # Do not log InvalidBucketName error because it is logged elsewhere.
    if client_error.response["Error"]["Code"] != "InvalidBucketName":
        logging.error(
            "Couldn't create the bucket %s. Here's why: %s",
            bucket_name,
            client_error.response["Error"]["Message"],
        )
    raise

def delete_bucket_and_objects(self, bucket_name: str) -> None:
    """
    Deletes a bucket and its objects.
    :param bucket_name: The name of the bucket.
    """
    try:
        # Delete the objects in the bucket first. This is required for a
        bucket to be deleted.
        paginator = self.s3_client.get_paginator("list_objects_v2")
        page_iterator = paginator.paginate(Bucket=bucket_name)
        for page in page_iterator:
            if "Contents" in page:
                delete_keys = {
                    "Objects": [{"Key": obj["Key"]} for obj in
page["Contents"]]
                }
                response = self.s3_client.delete_objects(
                    Bucket=bucket_name, Delete=delete_keys
                )
                if "Errors" in response:
                    for error in response["Errors"]:
                        logging.error(
                            "Couldn't delete object %s. Here's why: %s",

```



```

        error["Key"],
        error["Message"],
    )

    self.s3_client.delete_bucket(Bucket=bucket_name)
except ClientError as client_error:
    logging.error(
        "Couldn't delete the bucket %s. Here's why: %s",
        bucket_name,
        client_error.response["Error"]["Message"],
    )

def put_object(self, bucket_name: str, object_key: str, content: str) ->
None:
    """
    Puts an object into a bucket.
    :param bucket_name: The name of the bucket.
    :param object_key: The key of the object.
    :param content: The content of the object.
    """
    try:
        self.s3_client.put_object(Body=content, Bucket=bucket_name,
Key=object_key)
    except ClientError as client_error:
        logging.error(
            "Couldn't put the object %s into bucket %s. Here's why: %s",
            object_key,
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise

def list_objects(self, bucket: str) -> list[str]:
    """
    Lists objects in a bucket.
    :param bucket: The name of the bucket.
    :return: The list of objects in the bucket.
    """
    try:
        response = self.s3_client.list_objects_v2(Bucket=bucket)
        return response.get("Contents", [])
    except ClientError as client_error:
        logging.error(
            "Couldn't list objects in bucket %s. Here's why: %s",

```

```

        bucket,
        client_error.response["Error"]["Message"],
    )
    raise

def copy_object(
    self,
    source_bucket: str,
    source_key: str,
    destination_bucket: str,
    destination_key: str,
) -> None:
    """
    Copies an object from one bucket to another.
    :param source_bucket: The source bucket.
    :param source_key: The source key.
    :param destination_bucket: The destination bucket.
    :param destination_key: The destination key.
    :return: None
    """
    try:
        self.s3_client.copy_object(
            CopySource={"Bucket": source_bucket, "Key": source_key},
            Bucket=destination_bucket,
            Key=destination_key,
        )
    except ClientError as client_error:
        logging.error(
            "Couldn't copy object %s from bucket %s to bucket %s. Here's why:
%s",
            source_key,
            source_bucket,
            destination_bucket,
            client_error.response["Error"]["Message"],
        )
        raise

def create_session(self, bucket_name: str) -> None:
    """
    Creates an express session.
    :param bucket_name: The name of the bucket.
    """
    try:
        self.s3_client.create_session(Bucket=bucket_name)

```

```
except ClientError as client_error:
    logging.error(
        "Couldn't create the express session for bucket %s. Here's why:
%s",
        bucket_name,
        client_error.response["Error"]["Message"],
    )
    raise

def get_object(self, bucket_name: str, object_key: str) -> None:
    """
    Gets an object from a bucket.
    :param bucket_name: The name of the bucket.
    :param object_key: The key of the object.
    """
    try:
        self.s3_client.get_object(Bucket=bucket_name, Key=object_key)
    except ClientError as client_error:
        logging.error(
            "Couldn't get the object %s from bucket %s. Here's why: %s",
            object_key,
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise
```

- For API details, see the following topics in *Amazon SDK for Python (Boto3) API Reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Actions for S3 Directory Buckets using Amazon SDKs

The following code examples demonstrate how to perform individual S3 Directory Buckets actions with Amazon SDKs. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

These excerpts call the S3 Directory Buckets API and are code excerpts from larger programs that must be run in context. You can see actions in context in [Scenarios for S3 Directory Buckets using Amazon SDKs](#).

The following examples include only the most commonly used actions. For a complete list, see the [Amazon S3 Directory Buckets API Reference](#).

Examples

- [Use AbortMultipartUpload with an Amazon SDK](#)
- [Use CompleteMultipartUpload with an Amazon SDK](#)
- [Use CopyObject with an Amazon SDK](#)
- [Use CreateBucket with an Amazon SDK](#)
- [Use CreateMultipartUpload with an Amazon SDK](#)
- [Use CreateSession with an Amazon SDK](#)
- [Use DeleteBucket with an Amazon SDK](#)
- [Use DeleteBucketEncryption with an Amazon SDK](#)
- [Use DeleteBucketPolicy with an Amazon SDK](#)
- [Use DeleteObject with an Amazon SDK](#)
- [Use DeleteObjects with an Amazon SDK](#)
- [Use GetBucketEncryption with an Amazon SDK](#)
- [Use GetBucketPolicy with an Amazon SDK](#)
- [Use GetObject with an Amazon SDK](#)
- [Use GetObjectAttributes with an Amazon SDK](#)

- [Use HeadBucket with an Amazon SDK](#)
- [Use HeadObject with an Amazon SDK](#)
- [Use ListDirectoryBuckets with an Amazon SDK](#)
- [Use ListMultipartUploads with an Amazon SDK](#)
- [Use ListObjectsV2 with an Amazon SDK](#)
- [Use ListParts with an Amazon SDK](#)
- [Use PutBucketEncryption with an Amazon SDK](#)
- [Use PutBucketPolicy with an Amazon SDK](#)
- [Use PutObject with an Amazon SDK](#)
- [Use UploadPart with an Amazon SDK](#)
- [Use UploadPartCopy with an Amazon SDK](#)

Use AbortMultipartUpload with an Amazon SDK

The following code example shows how to use AbortMultipartUpload.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Abort a multipart upload in a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
```

```

import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Aborts a specific multipart upload for the specified S3 directory bucket.
 *
 * @param s3Client    The S3 client used to interact with S3
 * @param bucketName  The name of the directory bucket
 * @param objectKey   The key (name) of the object to be uploaded
 * @param uploadId    The upload ID of the multipart upload to abort
 * @return True if the multipart upload is successfully aborted, false
 * otherwise
 */
public static boolean abortDirectoryBucketMultipartUpload(S3Client s3Client,
    String bucketName,
        String objectKey, String uploadId) {
    logger.info("Aborting multipart upload: {} for bucket: {}", uploadId,
bucketName);
    try {
        // Abort the multipart upload
        AbortMultipartUploadRequest abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .uploadId(uploadId)
            .build();

        s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        logger.info("Aborted multipart upload: {} for object: {}", uploadId,
objectKey);
        return true;
    } catch (S3Exception e) {
        logger.error("Failed to abort multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}

```

- For API details, see [AbortMultipartUpload](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use CompleteMultipartUpload with an Amazon SDK

The following code example shows how to use CompleteMultipartUpload.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Complete a multipart upload in a directory bucket.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```

import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * This method completes the multipart upload request by collating all the
 * upload parts.
 *
 * @param s3Client    The S3 client used to interact with S3
 * @param bucketName  The name of the directory bucket
 * @param objectKey   The key (name) of the object to be uploaded
 * @param uploadId    The upload ID used to track the multipart upload
 * @param uploadParts The list of completed parts
 * @return True if the multipart upload is successfully completed, false
 *         otherwise
 */
public static boolean completeDirectoryBucketMultipartUpload(S3Client
s3Client, String bucketName, String objectKey,
String uploadId, List<CompletedPart> uploadParts) {
    try {
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
            .parts(uploadParts)
            .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .uploadId(uploadId)
            .multipartUpload(completedMultipartUpload)
            .build();

        CompleteMultipartUploadResponse response =
s3Client.completeMultipartUpload(completeMultipartUploadRequest);
        logger.info("Multipart upload completed. ETag: {}", response.eTag());
        return true;
    } catch (S3Exception e) {
        logger.error("Failed to complete multipart upload: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}

```


- For API details, see [CompleteMultipartUpload](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use CopyObject with an Amazon SDK

The following code example shows how to use CopyObject.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Learn the basics](#)

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Copy an object from a directory bucket to a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
```

```

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Copies an object from one S3 general purpose bucket to one S3 directory
 * bucket.
 *
 * @param s3Client      The S3 client used to interact with S3
 * @param sourceBucket The name of the source bucket
 * @param objectKey     The key (name) of the object to be copied
 * @param targetBucket The name of the target bucket
 */
public static void copyDirectoryBucketObject(S3Client s3Client, String
sourceBucket, String objectKey,
    String targetBucket) {
    logger.info("Copying object: {} from bucket: {} to bucket: {}",
objectKey, sourceBucket, targetBucket);

    try {
        // Create a CopyObjectRequest
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(objectKey)
            .destinationBucket(targetBucket)
            .destinationKey(objectKey)
            .build();

        // Copy the object
        CopyObjectResponse copyRes = s3Client.copyObject(copyReq);
        logger.info("Successfully copied {} from bucket {} into bucket {}.
CopyObjectResponse: {}",
            objectKey, sourceBucket, targetBucket,
            copyRes.copyObjectResult().toString());

    } catch (S3Exception e) {
        logger.error("Failed to copy object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

```

```
}  
}
```

- For API details, see [CopyObject](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use CreateBucket with an Amazon SDK

The following code example shows how to use CreateBucket.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Learn the basics](#)

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create an S3 directory bucket.

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.BucketInfo;  
import software.amazon.awssdk.services.s3.model.BucketType;  
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;  
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;  
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
```

```
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client The S3 client used to create the bucket
 * @param bucketName The name of the bucket to be created
 * @param zone The region where the bucket will be created
 * @throws S3Exception if there's an error creating the bucket
 */
public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
    logger.info("Creating bucket: {}", bucketName);

    CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
        .location(LocationInfo.builder()
            .type(LocationType.AVAILABILITY_ZONE)
            .name(zone).build())
        .bucket(BucketInfo.builder()
            .type(BucketType.DIRECTORY)
            .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
            .build())
        .build();

    try {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration).build();
        CreateBucketResponse response = s3Client.createBucket(bucketRequest);
        logger.info("Bucket created successfully with location: {}",
response.location());
    } catch (S3Exception e) {
        logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

```
}
```

- For API details, see [CreateBucket](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use CreateMultipartUpload with an Amazon SDK

The following code example shows how to use CreateMultipartUpload.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create a multipart upload in a directory bucket.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * This method creates a multipart upload request that generates a unique
 * upload
```

```

    * ID used to track
    * all the upload parts.
    *
    * @param s3Client The S3 client used to interact with S3
    * @param bucketName The name of the directory bucket
    * @param objectKey The key (name) of the object to be uploaded
    * @return The upload ID used to track the multipart upload
    */
    public static String createDirectoryBucketMultipartUpload(S3Client s3Client,
String bucketName, String objectKey) {
        logger.info("Creating multipart upload for object: {} in bucket: {}",
objectKey, bucketName);

        try {
            // Create a CreateMultipartUploadRequest
            CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            // Initiate the multipart upload
            CreateMultipartUploadResponse response =
s3Client.createMultipartUpload(createMultipartUploadRequest);
            String uploadId = response.uploadId();
            logger.info("Multipart upload initiated. Upload ID: {}", uploadId);
            return uploadId;

        } catch (S3Exception e) {
            logger.error("Failed to create multipart upload: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
}

```

- For API details, see [CreateMultipartUpload](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use CreateSession with an Amazon SDK

The following code example shows how to use CreateSession.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

```
class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
    interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                           access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def create_session(self, bucket_name: str) -> None:
        """
        Creates an express session.
```

```
        :param bucket_name: The name of the bucket.
        """
        try:
            self.s3_client.create_session(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't create the express session for bucket %s. Here's why:
%s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise
```

- For API details, see [CreateSession](#) in *Amazon SDK for Python (Boto3) API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucket with an Amazon SDK

The following code example shows how to use DeleteBucket.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Learn the basics](#)

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete an S3 directory bucket.


```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Deletes the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket to delete
 */
public static void deleteDirectoryBucket(S3Client s3Client, String
bucketName) {
    logger.info("Deleting bucket: {}", bucketName);

    try {
        // Create a DeleteBucketRequest
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Delete the bucket
        s3Client.deleteBucket(deleteBucketRequest);
        logger.info("Successfully deleted bucket: {}", bucketName);

    } catch (S3Exception e) {
        logger.error("Failed to delete bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- For API details, see [DeleteBucket](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucketEncryption with an Amazon SDK

The following code example shows how to use DeleteBucketEncryption.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete the encryption configuration for a directory bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Deletes the encryption configuration from an S3 bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 */
public static void deleteDirectoryBucketEncryption(S3Client s3Client, String
bucketName) {
    DeleteBucketEncryptionRequest deleteRequest =
DeleteBucketEncryptionRequest.builder()
        .bucket(bucketName)
```

```
        .build();

    try {
        s3Client.deleteBucketEncryption(deleteRequest);
        logger.info("Bucket encryption deleted for bucket: {}", bucketName);
    } catch (S3Exception e) {
        logger.error("Failed to delete bucket encryption: {} - Error code: {}", e.awsErrorDetails().errorMessage(), e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- For API details, see [DeleteBucketEncryption](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteBucketPolicy with an Amazon SDK

The following code example shows how to use DeleteBucketPolicy.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete a bucket policy for a directory bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Deletes the bucket policy for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 */
public static void deleteDirectoryBucketPolicy(S3Client s3Client, String
bucketName) {
    logger.info("Deleting policy for bucket: {}", bucketName);

    try {
        // Create a DeleteBucketPolicyRequest
        DeleteBucketPolicyRequest deletePolicyReq =
DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

        // Delete the bucket policy
        s3Client.deleteBucketPolicy(deletePolicyReq);
        logger.info("Successfully deleted bucket policy");

    } catch (S3Exception e) {
        logger.error("Failed to delete bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

```

- For API details, see [DeleteBucketPolicy](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteObject with an Amazon SDK

The following code example shows how to use DeleteObject.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Learn the basics](#)

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete an object in a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;
```

```
/**
 * Deletes an object from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be deleted
 */
public static void deleteDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
    logger.info("Deleting object: {} from bucket: {}", objectKey,
bucketName);

    try {
        // Create a DeleteObjectRequest
        DeleteObjectRequest deleteObjectRequest =
DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Delete the object
        s3Client.deleteObject(deleteObjectRequest);
        logger.info("Object {} has been deleted", objectKey);

    } catch (S3Exception e) {
        logger.error("Failed to delete object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- For API details, see [DeleteObject](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use DeleteObjects with an Amazon SDK

The following code example shows how to use DeleteObjects.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Delete multiple objects in a directory bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectsResponse;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Deletes multiple objects from the specified S3 directory bucket.
```

```

*
* @param s3Client    The S3 client used to interact with S3
* @param bucketName The name of the directory bucket
* @param objectKeys The list of keys (names) of the objects to be deleted
*/
public static void deleteDirectoryBucketObjects(S3Client s3Client, String
bucketName, List<String> objectKeys) {
    logger.info("Deleting objects from bucket: {}", bucketName);

    try {
        // Create a list of ObjectIdentifier.
        List<ObjectIdentifier> identifiers = objectKeys.stream()
            .map(key -> ObjectIdentifier.builder().key(key).build())
            .toList();

        Delete delete = Delete.builder()
            .objects(identifiers)
            .build();

        DeleteObjectsRequest deleteObjectsRequest =
DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(delete)
            .build();

        DeleteObjectsResponse deleteObjectsResponse =
s3Client.deleteObjects(deleteObjectsRequest);
        deleteObjectsResponse.deleted().forEach(deleted ->
logger.info("Deleted object: {}", deleted.key()));

    } catch (S3Exception e) {
        logger.error("Failed to delete objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

```

- For API details, see [DeleteObjects](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketEncryption with an Amazon SDK

The following code example shows how to use GetBucketEncryption.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get the encryption configuration of a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Retrieves the encryption configuration for an S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return The type of server-side encryption applied to the bucket (e.g.,
 *         AES256, aws:kms)
 */
```

```
public static String getDirectoryBucketEncryption(S3Client s3Client, String
bucketName) {
    try {
        // Create a GetBucketEncryptionRequest
        GetBucketEncryptionRequest getRequest =
        GetBucketEncryptionRequest.builder()
            .bucket(bucketName)
            .build();

        // Retrieve the bucket encryption configuration
        GetBucketEncryptionResponse response =
        s3Client.getBucketEncryption(getRequest);
        ServerSideEncryptionRule rule =
        response.getServerSideEncryptionConfiguration().rules().get(0);

        String encryptionType =
        rule.applyServerSideEncryptionByDefault().sseAlgorithmAsString();
        logger.info("Bucket encryption algorithm: {}", encryptionType);
        logger.info("KMS Customer Managed Key ID: {}",
        rule.applyServerSideEncryptionByDefault().kmsMasterKeyID());
        logger.info("Bucket Key Enabled: {}", rule.bucketKeyEnabled());

        return encryptionType;
    } catch (S3Exception e) {
        logger.error("Failed to get bucket encryption: {} - Error code: {}",
        e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- For API details, see [GetBucketEncryption](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetBucketPolicy with an Amazon SDK

The following code example shows how to use GetBucketPolicy.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get the policy of a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

/**
 * Retrieves the bucket policy for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return The bucket policy text
 */
public static String getDirectoryBucketPolicy(S3Client s3Client, String
bucketName) {
    logger.info("Getting policy for bucket: {}", bucketName);

    try {
        // Create a GetBucketPolicyRequest
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
```

```
        .build();

        // Retrieve the bucket policy
        GetBucketPolicyResponse response =
s3Client.getBucketPolicy(policyReq);

        // Print and return the policy text
        String policyText = response.policy();
        logger.info("Bucket policy: {}", policyText);
        return policyText;

    } catch (S3Exception e) {
        logger.error("Failed to get bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- For API details, see [GetBucketPolicy](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use GetObject with an Amazon SDK

The following code example shows how to use GetObject.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- [Learn the basics](#)
- [Create a presigned URL to get an object](#)

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get an object from a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.charset.StandardCharsets;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves an object from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be retrieved
 * @return The retrieved object as a ResponseInputStream
 */
```

```
public static boolean getDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
    logger.info("Retrieving object: {} from bucket: {}", objectKey,
bucketName);

    try {
        // Create a GetObjectRequest
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(objectKey)
            .bucket(bucketName)
            .build();

        // Retrieve the object as bytes
        ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Print object contents to console
        String objectContent = new String(data, StandardCharsets.UTF_8);
        logger.info("Object contents: \n{}", objectContent);

        return true;

    } catch (S3Exception e) {
        logger.error("Failed to retrieve object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}
```

- For API details, see [GetObject](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use `GetObjectAttributes` with an Amazon SDK

The following code example shows how to use `GetObjectAttributes`.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get an object attributes from a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves attributes for an object in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to retrieve attributes for
 * @return True if the object attributes are successfully retrieved, false
 *         otherwise
 */
```

```
public static boolean getDirectoryBucketObjectAttributes(S3Client s3Client,
String bucketName, String objectKey) {
    logger.info("Retrieving attributes for object: {} from bucket: {}",
objectKey, bucketName);

    try {
        // Create a GetObjectAttributesRequest
        GetObjectAttributesRequest getObjectAttributesRequest =
GetObjectAttributesRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .objectAttributes(ObjectAttributes.E_TAG,
ObjectAttributes.STORAGE_CLASS,
                ObjectAttributes.OBJECT_SIZE)
            .build();

        // Retrieve the object attributes
        GetObjectAttributesResponse response =
s3Client.getObjectAttributes(getObjectAttributesRequest);
        logger.info("Attributes for object {}:", objectKey);
        logger.info("ETag: {}", response.eTag());
        logger.info("Storage Class: {}", response.storageClass());
        logger.info("Object Size: {}", response.objectSize());
        return true;

    } catch (S3Exception e) {
        logger.error("Failed to retrieve object attributes: {} - Error code:
{}",
            e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        return false;
    }
}
```

- For API details, see [GetObjectAttributes](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use HeadBucket with an Amazon SDK

The following code example shows how to use HeadBucket.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Checks if the specified S3 directory bucket exists and is accessible.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Checks if the specified S3 directory bucket exists and is accessible.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket to check
 * @return True if the bucket exists and is accessible, false otherwise
 */
public static boolean headDirectoryBucket(S3Client s3Client, String
bucketName) {
    logger.info("Checking if bucket exists: {}", bucketName);

    try {
        // Create a HeadBucketRequest
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
```

```
        .build();
        // If the bucket doesn't exist, the following statement throws
        NoSuchBucketException,
        // which is a subclass of S3Exception.
        s3Client.headBucket(headBucketRequest);
        logger.info("Amazon S3 directory bucket: \"{}\" found.", bucketName);
        return true;

    } catch (S3Exception e) {
        logger.error("Failed to access bucket: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- For API details, see [HeadBucket](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use HeadObject with an Amazon SDK

The following code example shows how to use HeadObject.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Get metadata of an object in a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves metadata for an object in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to retrieve metadata for
 * @return True if the object exists, false otherwise
 */
public static boolean headDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
    logger.info("Retrieving metadata for object: {} from bucket: {}",
objectKey, bucketName);

    try {
        // Create a HeadObjectRequest
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Retrieve the object metadata
        HeadObjectResponse response = s3Client.headObject(headObjectRequest);
        logger.info("Amazon S3 object: \"{}\" found in bucket: \"{}\" with
ETag: \"{}\"", objectKey, bucketName,
            response.eTag());
        logger.info("Content-Type: {}", response.contentType());
        logger.info("Content-Length: {}", response.contentLength());
    }
```

```
        logger.info("Last Modified: {}", response.lastModified());
        return true;

    } catch (S3Exception e) {
        logger.error("Failed to retrieve object metadata: {} - Error code:
        {}", e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}
```

- For API details, see [HeadObject](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use ListDirectoryBuckets with an Amazon SDK

The following code example shows how to use ListDirectoryBuckets.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

List all directory buckets.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Lists all S3 directory buckets and no general purpose buckets.
 *
 * @param s3Client The S3 client used to interact with S3
 * @return A list of bucket names
 */
public static List<String> listDirectoryBuckets(S3Client s3Client) {
    logger.info("Listing all directory buckets");

    try {
        // Create a ListBucketsRequest
        ListDirectoryBucketsRequest listDirectoryBucketsRequest =
            ListDirectoryBucketsRequest.builder().build();

        // Retrieve the list of buckets
        ListDirectoryBucketsResponse response =
            s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);

        // Extract bucket names
        List<String> bucketNames = response.buckets().stream()
            .map(Bucket::name)
            .collect(Collectors.toList());

        return bucketNames;
    } catch (S3Exception e) {
        logger.error("Failed to list buckets: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        throw e;
    }
}
```

- For API details, see [ListDirectoryBuckets](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use ListMultipartUploads with an Amazon SDK

The following code example shows how to use ListMultipartUploads.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

List multipart uploads in a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
```

```
import static
com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Lists multipart uploads for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return A list of MultipartUpload objects representing the multipart
uploads
 */
public static List<MultipartUpload>
listDirectoryBucketMultipartUploads(S3Client s3Client, String bucketName) {
    logger.info("Listing in-progress multipart uploads for bucket: {}",
bucketName);

    try {
        // Create a ListMultipartUploadsRequest
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        // List the multipart uploads
        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            logger.info("In-progress multipart upload: Upload ID: {}, Key:
{}, Initiated: {}", upload.uploadId(),
                upload.key(), upload.initiated());
        }
        return uploads;

    } catch (S3Exception e) {
        logger.error("Failed to list multipart uploads: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        return List.of(); // Return an empty list if an exception is thrown
    }
}
```

- For API details, see [ListMultipartUploads](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use ListObjectsV2 with an Amazon SDK

The following code example shows how to use ListObjectsV2.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

List objects in a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

import java.nio.file.Path;
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```



```

import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Lists objects in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return A list of object keys in the bucket
 */
public static List<String> listDirectoryBucketObjectsV2(S3Client s3Client,
String bucketName) {
    logger.info("Listing objects in bucket: {}", bucketName);

    try {
        // Create a ListObjectsV2Request
        ListObjectsV2Request listObjectsV2Request =
ListObjectsV2Request.builder()
                        .bucket(bucketName)
                        .build();

        // Retrieve the list of objects
        ListObjectsV2Response response =
s3Client.listObjectsV2(listObjectsV2Request);

        // Extract and return the object keys
        return response.contents().stream()
                        .map(S3Object::key)
                        .collect(Collectors.toList());

    } catch (S3Exception e) {
        logger.error("Failed to list objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                        e.awsErrorDetails().errorCode());
        throw e;
    }
}

```

- For API details, see [ListObjectsV2](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use ListParts with an Amazon SDK

The following code example shows how to use ListParts.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

List parts of a multipart upload in a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListPartsRequest;
import software.amazon.awssdk.services.s3.model.ListPartsResponse;
import software.amazon.awssdk.services.s3.model.Part;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
```

```
import static
com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Lists the parts of a multipart upload for the specified S3 directory
 * bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object being uploaded
 * @param uploadId The upload ID used to track the multipart upload
 * @return A list of Part representing the parts of the multipart upload
 */
public static List<Part> listDirectoryBucketMultipartUploadParts(S3Client
s3Client, String bucketName,
    String objectKey, String uploadId) {
    logger.info("Listing parts for object: {} in bucket: {}", objectKey,
bucketName);

    try {
        // Create a ListPartsRequest
        ListPartsRequest listPartsRequest = ListPartsRequest.builder()
            .bucket(bucketName)
            .uploadId(uploadId)
            .key(objectKey)
            .build();

        // List the parts of the multipart upload
        ListPartsResponse response = s3Client.listParts(listPartsRequest);
        List<Part> parts = response.parts();
        for (Part part : parts) {
            logger.info("Uploaded part: Part number = \"{}\", etag = {}",
part.partNumber(), part.eTag());
        }
        return parts;

    } catch (S3Exception e) {
        logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        return List.of(); // Return an empty list if an exception is thrown
    }
}
```

- For API details, see [ListParts](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketEncryption with an Amazon SDK

The following code example shows how to use PutBucketEncryption.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Set bucket encryption to a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import
    software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsClient;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsKey;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.scheduleKeyDeletion;

/**
 * Sets the default encryption configuration for an S3 bucket as SSE-KMS.
 *
 * @param s3Client    The S3 client used to interact with S3
 * @param bucketName  The name of the directory bucket
 * @param kmsKeyId    The ID of the customer-managed KMS key
 */
public static void putDirectoryBucketEncryption(S3Client s3Client, String
bucketName, String kmsKeyId) {
    // Define the default encryption configuration to use SSE-KMS. For
    directory
    // buckets, AWS managed KMS keys aren't supported. Only customer-managed
    keys
    // are supported.
    ServerSideEncryptionByDefault encryptionByDefault =
    ServerSideEncryptionByDefault.builder()
        .sseAlgorithm(ServerSideEncryption.AWS_KMS)
        .kmsMasterKeyID(kmsKeyId)
        .build();

    // Create a server-side encryption rule to apply the default encryption
    // configuration. For directory buckets, the bucketKeyEnabled field is
    enforced
    // to be true.
    ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
        .bucketKeyEnabled(true)
        .applyServerSideEncryptionByDefault(encryptionByDefault)
        .build();

    // Create the server-side encryption configuration for the bucket
    ServerSideEncryptionConfiguration encryptionConfiguration =
    ServerSideEncryptionConfiguration.builder()
        .rules(rule)
        .build();

    // Create the PutBucketEncryption request
    PutBucketEncryptionRequest putRequest =
    PutBucketEncryptionRequest.builder()
        .bucket(bucketName)
        .serverSideEncryptionConfiguration(encryptionConfiguration)
        .build();
}
```

```
// Set the bucket encryption
try {
    s3Client.putBucketEncryption(putRequest);
    logger.info("SSE-KMS Bucket encryption configuration set for the
directory bucket: {}", bucketName);
} catch (S3Exception e) {
    logger.error("Failed to set bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
    e.awsErrorDetails().errorCode());
    throw e;
}
```

- For API details, see [PutBucketEncryption](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutBucketPolicy with an Amazon SDK

The following code example shows how to use PutBucketPolicy.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Apply a bucket policy to a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
```

```

import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;

/**
 * Sets the following bucket policy for the specified S3 directory bucket.
 * <pre>
 * {
 *     "Version": "2012-10-17",
 *     "Statement": [
 *         {
 *             "Sid": "AdminPolicy",
 *             "Effect": "Allow",
 *             "Principal": {
 *                 "AWS": "arn:aws:iam::<ACCOUNT_ID>:root"
 *             },
 *             "Action": "s3express:*",
 *             "Resource": "arn:aws:s3express:us-west-2:<ACCOUNT_ID>:bucket/
<DIR_BUCKET_NAME>
 *         }
 *     ]
 * }
 * </pre>
 * This policy grants all S3 directory bucket actions to identities in the
 * same account as the bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param policyText The policy text to be applied
 */
public static void putDirectoryBucketPolicy(S3Client s3Client, String
bucketName, String policyText) {
    logger.info("Setting policy on bucket: {}", bucketName);
    logger.info("Policy: {}", policyText);

    try {
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
            .bucket(bucketName)
            .policy(policyText)
            .build();
    }

```

```
s3Client.putBucketPolicy(policyReq);
logger.info("Bucket policy set successfully!");

} catch (S3Exception e) {
    logger.error("Failed to set bucket policy: {} - Error code: {}",
        e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode(), e);
    throw e;
}
```

- For API details, see [PutBucketPolicy](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use PutObject with an Amazon SDK

The following code example shows how to use PutObject.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- [Learn the basics](#)

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Put an object into a directory bucket.

```
import org.slf4j.Logger;
```



```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsErrorDetails;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.UncheckedIOException;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

/**
 * Puts an object into the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be placed in the bucket
 * @param filePath The path of the file to be uploaded
 */
public static void putDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey, Path filePath) {
    logger.info("Putting object: {} into bucket: {}", objectKey, bucketName);

    try {
        // Create a PutObjectRequest
        PutObjectRequest putObj = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Upload the object
        s3Client.putObject(putObj, filePath);
        logger.info("Successfully placed {} into bucket {}", objectKey,
bucketName);

    } catch (UncheckedIOException e) {
        throw S3Exception.builder().message("Failed to read the file: " +
e.getMessage()).cause(e)
    }
}
```

```
        .awsErrorDetails(AwsErrorDetails.builder()
            .errorCode("ClientSideException:FailedToReadFile")
            .errorMessage(e.getMessage())
            .build())
        .build();
    } catch (S3Exception e) {
        logger.error("Failed to put object: {}", e.getMessage(), e);
        throw e;
    }
}
```

- For API details, see [PutObject](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use UploadPart with an Amazon SDK

The following code example shows how to use UploadPart.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Upload part of a multipart upload for a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

/**
 * This method creates part requests and uploads individual parts to S3.
 * While it uses the UploadPart API to upload a single part, it does so
 * sequentially to handle multiple parts of a file, returning all the
 * completed
 * parts.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be uploaded
 * @param uploadId The upload ID used to track the multipart upload
 * @param filePath The path to the file to be uploaded
 * @return A list of uploaded parts
 * @throws IOException if an I/O error occurs
 */
public static List<CompletedPart> multipartUploadForDirectoryBucket(S3Client
s3Client, String bucketName,
    String objectKey, String uploadId, Path filePath) throws IOException
{
    logger.info("Uploading parts for object: {} in bucket: {}", objectKey,
bucketName);

    int partNumber = 1;
```

```
List<CompletedPart> uploadedParts = new ArrayList<>();
ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

// Read the local file, break down into chunks and process
try (RandomAccessFile file = new RandomAccessFile(filePath.toFile(),
"r")) {
    long fileSize = file.length();
    int position = 0;

    // Sequentially upload parts of the file
    while (position < fileSize) {
        file.seek(position);
        int read = file.getChannel().read(bb);

        bb.flip(); // Swap position and limit before reading from the
buffer

        UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .uploadId(uploadId)
            .partNumber(partNumber)
            .build();

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        // Build the uploaded part
        CompletedPart uploadedPart = CompletedPart.builder()
            .partNumber(partNumber)
            .eTag(partResponse.eTag())
            .build();

        // Add the uploaded part to the list
        uploadedParts.add(uploadedPart);

        // Log to indicate the part upload is done
        logger.info("Uploaded part number: {} with ETag: {}", partNumber,
partResponse.eTag());

        bb.clear();
        position += read;
        partNumber++;
    }
}
```

```
    } catch (S3Exception e) {
        logger.error("Failed to list parts: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        throw e;
    }
    return uploadedParts;
}
```

- For API details, see [UploadPart](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Use UploadPartCopy with an Amazon SDK

The following code example shows how to use UploadPartCopy.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Create copy parts based on source object size and copy over individual parts to a directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```

import software.amazon.awssdk.services.s3.model.UploadPartCopyRequest;
import software.amazon.awssdk.services.s3.model.UploadPartCopyResponse;

import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static
    com.example.s3.util.S3DirectoryBucketUtils.completeDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Creates copy parts based on source object size and copies over individual
 * parts.
 *
 * @param s3Client      The S3 client used to interact with S3
 * @param sourceBucket  The name of the source bucket
 * @param sourceKey     The key (name) of the source object
 * @param destinationBucket The name of the destination bucket
 * @param destinationKey The key (name) of the destination object
 * @param uploadId      The upload ID used to track the multipart upload
 * @return A list of completed parts
 */
public static List<CompletedPart>
multipartUploadCopyForDirectoryBucket(S3Client s3Client, String sourceBucket,
    String sourceKey, String destinationBucket, String destinationKey,
    String uploadId) {
    // Get the object size to track the end of the copy operation
    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(sourceBucket)
        .key(sourceKey)
        .build();

```

```

        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        long objectSize = headObjectResponse.contentLength();

        logger.info("Source Object size: {}", objectSize);

        // Copy the object using 20 MB parts
        long partSize = 20 * 1024 * 1024; // 20 MB
        long bytePosition = 0;
        int partNum = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();

        while (bytePosition < objectSize) {
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize -
1);

            logger.info("Part Number: {}, Byte Position: {}, Last Byte: {}",
partNum, bytePosition, lastByte);

            try {
                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(sourceBucket)
                    .sourceKey(sourceKey)
                    .destinationBucket(destinationBucket)
                    .destinationKey(destinationKey)
                    .uploadId(uploadId)
                    .copySourceRange("bytes=" + bytePosition + "-" +
lastByte)

                    .partNumber(partNum)
                    .build();

                UploadPartCopyResponse uploadPartCopyResponse =
s3Client.uploadPartCopy(uploadPartCopyRequest);

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNum)
                    .eTag(uploadPartCopyResponse.copyPartResult().eTag())
                    .build();
                uploadedParts.add(part);

                bytePosition += partSize;
                partNum++;
            } catch (S3Exception e) {
                logger.error("Failed to copy part number {}: {} - Error code:
{}", partNum,

```

```
        e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode());
        throw e;
    }
}

return uploadedParts;
}
```

- For API details, see [UploadPartCopy](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Scenarios for S3 Directory Buckets using Amazon SDKs

The following code examples show you how to implement common scenarios in S3 Directory Buckets with Amazon SDKs. These scenarios show you how to accomplish specific tasks by calling multiple functions within S3 Directory Buckets or combined with other Amazon Web Services services. Each scenario includes a link to the complete source code, where you can find instructions on how to set up and run the code.

Scenarios target an intermediate level of experience to help you understand service actions in context.

Examples

- [Create a presigned URL for Amazon S3 directory buckets to get an object using an Amazon SDK](#)

Create a presigned URL for Amazon S3 directory buckets to get an object using an Amazon SDK

The following code example shows how to create a presigned URL for S3 directory buckets and get an object.

Java

SDK for Java 2.x

 Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Amazon Code Examples Repository](#).

Generate a presigned GET URL for accessing an object in an S3 directory bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;

import java.nio.file.Path;
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Generates a presigned URL for accessing an object in the specified S3
 * directory bucket.
 */
```

```
    * @param s3Presigner The S3 presigner client used to generate the presigned
URL
    * @param bucketName The name of the directory bucket
    * @param objectKey The key (name) of the object to access
    * @return A presigned URL for accessing the specified object
    */
    public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
s3Presigner, String bucketName,
        String objectKey) {
        logger.info("Generating presigned URL for object: {} in bucket: {}",
objectKey, bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            // Create a GetObjectPresignRequest
            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
valid for 10 minutes
                .getObjectRequest(getObjectRequest)
                .build();

            // Generate the presigned URL
            PresignedGetObjectRequest presignedGetObjectRequest =
s3Presigner.presignGetObject(getObjectPresignRequest);

            // Get the presigned URL
            String presignedURL = presignedGetObjectRequest.url().toString();
            logger.info("Presigned URL: {}", presignedURL);
            return presignedURL;

        } catch (S3Exception e) {
            logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
}
```

- For API details, see [GetObject](#) in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see [Developing with Amazon S3 using the Amazon SDKs](#). This topic also includes information about getting started and details about previous SDK versions.

Authenticating Requests (Amazon Signature Version 4)

Topics

- [Authentication Methods](#)
- [Introduction to Signing Requests](#)
- [Authenticating Requests: Using the Authorization Header \(Amazon Signature Version 4\)](#)
- [Authenticating Requests: Using Query Parameters \(Amazon Signature Version 4\)](#)
- [Examples: Signature Calculations in Amazon Signature Version 4](#)
- [Authenticating Requests: Browser-Based Uploads Using POST \(Amazon Signature Version 4\)](#)
- [Amazon S3 Signature Version 4 Authentication Specific Policy Keys](#)

Every interaction with Amazon S3 is either authenticated or anonymous. This section explains request authentication with the Amazon Signature Version 4 algorithm.

Note

If you use the Amazon SDKs (see [Sample Code and Libraries](#)) to send your requests, you don't need to read this section because the SDK clients authenticate your requests by using access keys that you provide. Unless you have a good reason not to, you should always use the Amazon SDKs. In Regions that support both signature versions, you can request Amazon SDKs to use specific signature version. For more information, see [Sending authenticated requests using the Amazon SDKs](#). You need to read this section only if you are implementing the Amazon Signature Version 4 algorithm in your custom client.

Authentication with Amazon Signature Version 4 provides some or all of the following, depending on how you choose to sign your request:

- **Verification of the identity of the requester** – Authenticated requests require a signature that you create by using your access keys (access key ID, secret access key). For information about getting access keys, see [Understanding and Getting Your Security Credentials](#) in the *Amazon Web Services General Reference*. If you are using temporary security credentials, the signature calculations also require a security token. For more information, see [Requesting Temporary Security Credentials](#) in the *IAM User Guide*.

- **In-transit data protection** – In order to prevent tampering with a request while it is in transit, you use some of the request elements to calculate the request signature. Upon receiving the request, Amazon S3 calculates the signature by using the same request elements. If any request component received by Amazon S3 does not match the component that was used to calculate the signature, Amazon S3 will reject the request.
- **Protect against reuse of the signed portions of the request** – The signed portions (using Amazon Signatures) of requests are valid within 15 minutes of the timestamp in the request. An unauthorized party who has access to a signed request can modify the unsigned portions of the request without affecting the request's validity in the 15 minute window. Because of this, we recommend that you maximize protection by signing request headers and body, making HTTPS requests to Amazon S3, and by using the `s3:x-amz-content-sha256` condition key (see [Amazon S3 Signature Version 4 Authentication Specific Policy Keys](#)) in Amazon policies to require users to sign Amazon S3 request bodies.

Note

Amazon S3 supports Signature Version 4, a protocol for authenticating inbound API requests to Amazon services, in all Amazon Web Services Regions. At this time, Amazon Web Services Regions created before January 30, 2014 will continue to support the previous protocol, Signature Version 2. Any new Regions after January 30, 2014 will support only Signature Version 4 and therefore all requests to those Regions must be made with Signature Version 4. For more information about Amazon Signature Version 2, see [Signing and Authenticating REST Requests](#) in the *Amazon Simple Storage Service User Guide*.

Authentication Methods

You can express authentication information by using one of the following methods:

- **HTTP Authorization header** – Using the HTTP Authorization header is the most common method of authenticating an Amazon S3 request. All of the Amazon S3 REST operations (except for browser-based uploads using POST requests) require this header. For more information about the Authorization header value, and how to calculate signature and related options, see [Authenticating Requests: Using the Authorization Header \(Amazon Signature Version 4\)](#).

- **Query string parameters** – You can use a query string to express a request entirely in a URL. In this case, you use query parameters to provide request information, including the authentication information. Because the request signature is part of the URL, this type of URL is often referred to as a presigned URL. You can use presigned URLs to embed clickable links, which can be valid for up to seven days, in HTML. For more information, see [Authenticating Requests: Using Query Parameters \(Amazon Signature Version 4\)](#).

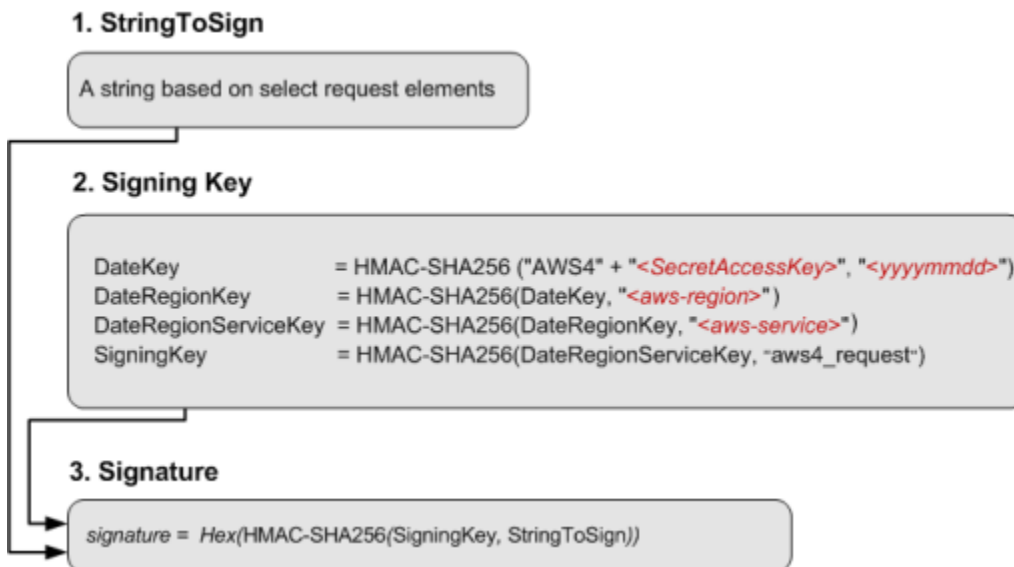
Amazon S3 also supports browser-based uploads that use HTTP POST requests. With an HTTP POST request, you can upload content to Amazon S3 directly from the browser. For information about authenticating POST requests, see [Browser-Based Uploads Using POST \(Amazon Signature Version 4\)](#).

Introduction to Signing Requests

Authentication information that you send in a request must include a signature. To calculate a signature, you first concatenate select request elements to form a string, referred to as the *string to sign*. You then use a signing key to calculate the hash-based message authentication code (HMAC) of the string to sign.

In Amazon Signature Version 4, you don't use your secret access key to sign the request. Instead, you first use your secret access key to derive a signing key. The derived signing key is specific to the date, service, and Region. For more information about how to derive a signing key in different programming languages, see [Examples of how to derive a signing key for Signature Version 4](#).

The following diagram illustrates the general process of computing a signature.



The string to sign depends on the request type. For example, when you use the HTTP Authorization header or the query parameters for authentication, you use a varying combination of request elements to create the string to sign. For an HTTP POST request, the POST policy in the request is the string you sign. For more information about computing string to sign, follow links provided at the end of this section.

For signing key, the diagram shows series of calculations, where result of each step you feed into the next step. The final step is the signing key.

Upon receiving an authenticated request, Amazon S3 servers re-create the signature by using the authentication information that is contained in the request. If the signatures match, Amazon S3 processes your request; otherwise, the request is rejected.

For more information about authenticating requests, see the following topics:

- [Authenticating Requests: Using the Authorization Header \(Amazon Signature Version 4\)](#)
- [Authenticating Requests: Using Query Parameters \(Amazon Signature Version 4\)](#)
- [Browser-Based Uploads Using POST \(Amazon Signature Version 4\)](#)

Authenticating Requests: Using the Authorization Header (Amazon Signature Version 4)

Topics

- [Overview](#)

- [Signature Calculations for the Authorization Header: Transferring Payload in a Single Chunk \(Amazon Signature Version 4\)](#)
- [Signature Calculations for the Authorization Header: Transferring Payload in Multiple Chunks \(Chunked Upload\) \(Amazon Signature Version 4\)](#)
- [Signature calculations for trailing headers \(chunked uploads\) \(Amazon Signature Version 4\)](#)

Overview

Using the HTTP Authorization header is the most common method of providing authentication information. Except for [POST requests](#) and requests that are signed by using query parameters, all Amazon S3 operations use the Authorization request header to provide authentication information.

The following is an example of the Authorization header value. Line breaks are added to this example for readability:

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

The following table describes the various components of the Authorization header value in the preceding example:

Component	Description
AWS4-HMAC-SHA256	The algorithm that was used to calculate the signature. You must provide this value when you use Amazon Signature Version 4 for authentication. The string specifies Amazon Signature Version 4 (AWS4) and the signing algorithm (HMAC-SHA256).
Credential	

Component	Description
	Your access key ID and the scope information, which includes the date, Region, and service that were used to calculate the signature. This string has the following form: <your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request Where: <date> value is specified using YYYYMMDD format. <aws-service> value is s3 when sending request to Amazon S3.
SignedHeaders	A semicolon-separated list of request headers that you used to compute Signature . The list includes header names only, and the header names must be in lowercase. For example: host;range;x-amz-date
Signature	The 256-bit signature expressed as 64 lowercase hexadecimal characters. For example: fe5f80f77d5fa3beca038a248ff027d0445342fe2855d dc963176630326f1024 Note that the signature calculations vary depending on the option you choose to transfer the payload.

The signature calculations vary depending on the method you choose to transfer the request payload. S3 supports the following options:

- **Transfer payload in a single chunk** – In this case, you have the following signature calculation options:
 - **Signed payload option** – You can optionally compute the entire payload checksum and include it in signature calculation. This provides added security but you need to read your payload twice or buffer it in memory.

For example, in order to upload a file, you need to read the file first to compute a payload hash for signature calculation and again for transmission when you create the request. For smaller payloads, this approach might be preferable. However, for large files, reading the file twice can be inefficient, so you might want to upload data in chunks instead.

We recommend you include payload checksum for added security.

- **Unsigned payload option** – Do not include payload checksum in signature calculation.

For step-by-step instructions to calculate signature and construct the Authorization header value, see [Signature Calculations for the Authorization Header: Transferring Payload in a Single Chunk \(Amazon Signature Version 4\)](#).

- **Transfer payload in multiple chunks (chunked upload)** – In this case you transfer payload in chunks. You can transfer a payload in chunks regardless of the payload size.

You can break up your payload into chunks. These can be fixed or variable-size chunks. By uploading data in chunks, you avoid reading the entire payload to calculate the signature. Instead, for the first chunk, you calculate a seed signature that uses only the request headers. The second chunk contains the signature for the first chunk, and each subsequent chunk contains the signature for the chunk that precedes it. At the end of the upload, you send a final chunk with 0 bytes of data that contains the signature of the last chunk of the payload. For more information, see [Signature Calculations for the Authorization Header: Transferring Payload in Multiple Chunks \(Chunked Upload\) \(Amazon Signature Version 4\)](#).

When signing your requests, you can use either Amazon Signature Version 4 or Amazon Signature Version 4A. The key difference between the two is determined by how the signature is calculated. With Amazon Signature Version 4A, the signature does not include Region-specific information and is calculated using the AWS4-ECDSA-P256-SHA256 algorithm.

In addition to these options, you have the option of including a trailer with your request. In order to include a trailer with your request, you need to specify that in the header by setting `x-amz-content-sha256` to the appropriate value. If you are using a trailing header, you must include `x-amz-trailer` in the header and specify the trailing header names as a string in a comma-separated list. All trailing headers are written after the final chunk. If you're uploading the data in multiple chunks, you must send a final chunk with 0 bytes of data before sending the trailing header.

When you send a request, you must tell Amazon S3 which of the preceding options you have chosen in your signature calculation, by adding the `x-amz-content-sha256` header with one of the following values:

Header value	Description
Actual payload checksum value	This value is the actual checksum of your object and is only possible when you are uploading the data in a single chunk.
UNSIGNED-PAYLOAD	Use this when you are uploading the object as a single unsigned chunk.
STREAMING-UNSIGNED-PAYLOAD-TRAILER	Use this when sending an unsigned payload over multiple chunks. In this case you also have a trailing header after the chunk is uploaded.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD	Use this when sending a payload over multiple chunks, and the chunks are signed using <code>AWS4-HMAC-SHA256</code> . This produces a SigV4 signature.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER	Use this when sending a payload over multiple chunks, and the chunks are signed using <code>AWS4-HMAC-SHA256</code> . This produces a SigV4 signature. In addition, the digest for the chunks is included as a trailing header.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD	Use this when sending a payload over multiple chunks, and the chunks are signed using <code>AWS4-ECDSA-P256-SHA256</code> . This produces a SigV4A signature.

Header value	Description
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD-TRAILER	Use this when sending a payload over multiple chunks, and the chunks are signed using AWS4-ECDSA-P256-SHA256. This produces a SigV4A signature. In addition, the digest for the chunks is included as a trailing header.

Upon receiving the request, Amazon S3 re-creates the string to sign using information in the `Authorization` header and the date header. It then verifies with authentication service the signatures match. The request date can be specified by using either the `HTTP Date` or the `x-amz-date` header. If both headers are present, `x-amz-date` takes precedence.

If the signatures match, Amazon S3 processes your request; otherwise, your request will fail.

For more information, see the following topics:

[Signature Calculations for the Authorization Header: Transferring Payload in a Single Chunk \(Amazon Signature Version 4\)](#)

[Signature Calculations for the Authorization Header: Transferring Payload in Multiple Chunks \(Chunked Upload\) \(Amazon Signature Version 4\)](#)

[Signature calculations for trailing headers \(chunked uploads\) \(Amazon Signature Version 4\)](#)

Signature Calculations for the Authorization Header: Transferring Payload in a Single Chunk (Amazon Signature Version 4)

When using the `Authorization` header to authenticate requests, the header value includes, among other things, a signature. The signature calculations vary depending on the choice you make for transferring the payload ([Overview](#)). This section explains signature calculations when you choose to transfer the payload in a single chunk. The example section (see [Examples: Signature Calculations](#)) shows signature calculations and resulting `Authorization` headers that you can use as a test suite to verify your code.

Important

When transferring payload in a single chunk, you can optionally choose to include the payload hash in the signature calculations, referred as *signed payload* (if you don't include

it, the payload is considered *unsigned*). The signing procedure discussed in the following section applies to both, but note the following differences:

- **Signed payload option** – You include the payload hash when constructing the canonical request (that then becomes part of `StringToSign`, as explained in the signature calculation section). You also specify the same value as the `x-amz-content-sha256` header value when sending the request to S3.
- **Unsigned payload option** – You include the literal string `UNSIGNED-PAYLOAD` when constructing a canonical request, and set the same value as the `x-amz-content-sha256` header value when sending the request to Amazon S3.

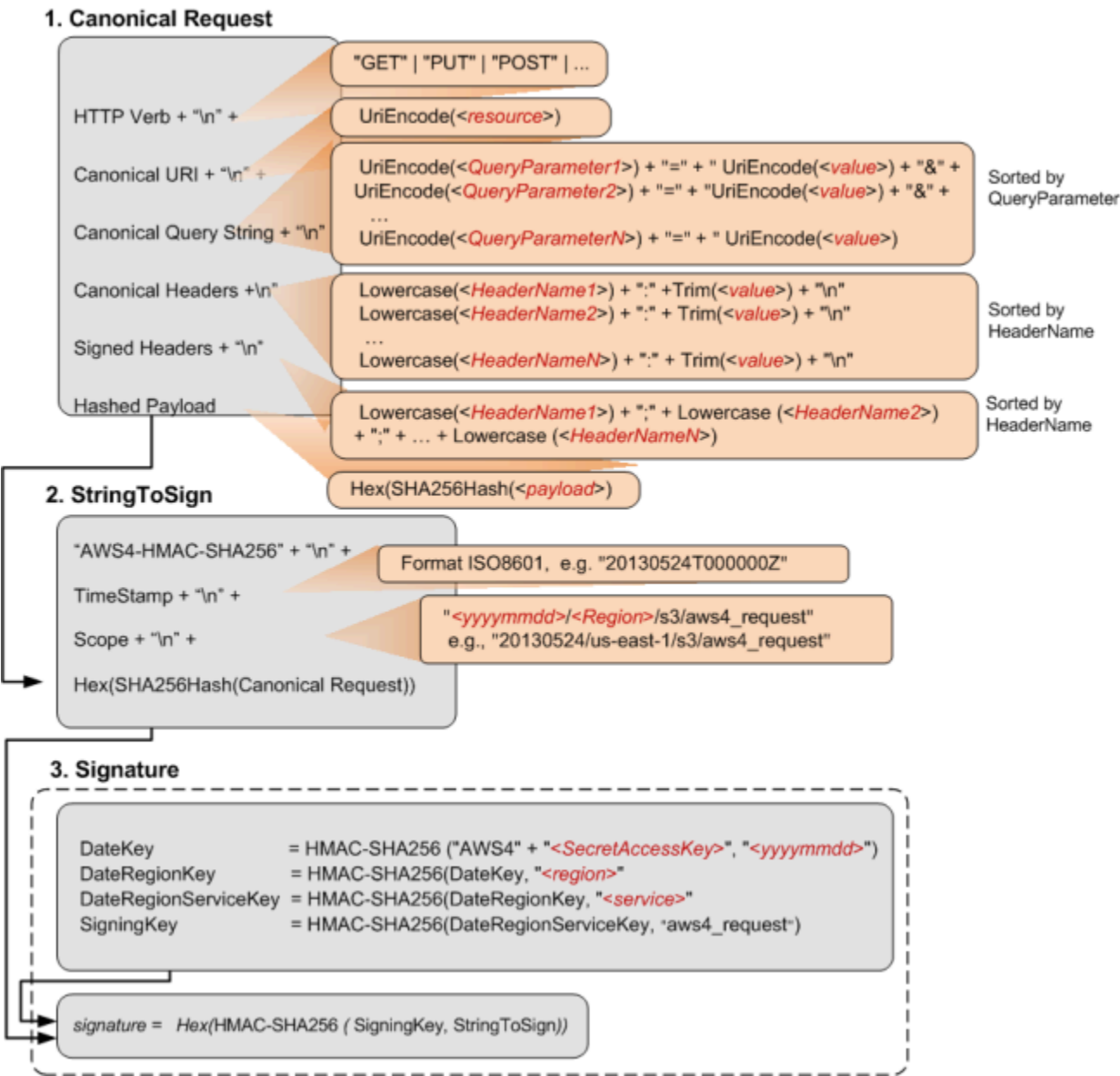
When you send your request to Amazon S3, the `x-amz-content-sha256` header value informs Amazon S3 whether the payload is signed or not. Amazon S3 can then create the signature accordingly for verification.

In both cases, because the `x-amz-content-sha256` header value is already part of your `HashedPayload`, you are not required to include the `x-amz-content-sha256` header as a canonical header.

Calculating a Signature

To calculate a signature, you first need a string to sign. You then calculate a HMAC-SHA256 hash of the string to sign by using a signing key. The following diagram illustrates the process, including the various components of the string that you create for signing

When Amazon S3 receives an authenticated request, it computes the signature and then compares it with the signature that you provided in the request. For that reason, you must compute the signature by using the same method that is used by Amazon S3. The process of putting a request in an agreed-upon form for signing is called canonicalization.



The following table describes the functions that are shown in the diagram. You need to implement code for these functions.

Function	Description
Lowercase()	Convert the string to lowercase.
Hex()	Lowercase base 16 encoding.
SHA256Hash()	Secure Hash Algorithm (SHA) cryptographic hash function.

Function	Description
<code>HMAC-SHA256()</code>	Computes HMAC by using the SHA256 algorithm with the signing key provided. This is the final signature.
<code>Trim()</code>	Remove any leading or trailing whitespace.

Function	Description
<code>UriEncode()</code>	URI encode every byte. <code>UriEncode()</code> must enforce the following rules: • URI encode every byte except the unreserved characters: 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', and '~'. • The space character is a reserved character and must be encoded as "%20" (and not as "+"). • Each URI encoded byte is formed by a '%' and the two-digit hexadecimal value of the byte. • Letters in the hexadecimal value must be uppercase, for example "%1A". • Encode the forward slash character, '/', everywhere except in the object key name. For example, if the object key name is <code>photos/Jan/sample.jpg</code>, the forward slash in the key name is not encoded.  Important The standard <code>UriEncode</code> functions provided by your development platform may not work because of differences in implementation and related ambiguity in the underlying RFCs. We recommend that you write your own custom <code>UriEncode</code> function to ensure that your encoding will work. The following is an example <code>UriEncode()</code> function in Java. <pre>public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBu ilder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i);</pre>

Function	Description
	<pre> if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Task 1: Create a Canonical Request

This section provides an overview of creating a canonical request.

The following is the canonical request format that Amazon S3 uses to calculate a signature. For signatures to match, you must create a canonical request in this format:

```

<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>

```

Where:

- *HTTPMethod* is one of the HTTP methods, for example GET, PUT, HEAD, and DELETE.
- *CanonicalURI* is the URI-encoded version of the absolute path component of the URI—everything starting with the "/" that follows the domain name and up to the end of the string or to the question mark character (?) if you have query string parameters. The URI in the following example, /examplebucket/myphoto.jpg, is the absolute path and you don't encode the "/" in the absolute path:

```
http://s3.amazonaws.com/examplebucket/myphoto.jpg
```

Note

You do not normalize URI paths for requests to Amazon S3. For example, you may have a bucket with an object named "my-object//example//photo.user". Normalizing the path changes the object name in the request to "my-object/example/photo.user". This is an incorrect path for that object.

- **CanonicalQueryString** specifies the URI-encoded query string parameters. You URI-encode name and values individually. You must also sort the parameters in the canonical query string alphabetically by key name. The sorting occurs after encoding. The query string in the following URI example is `prefix=somePrefix&marker=someMarker&max-keys=20`:

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=20
```

The canonical query string is as follows (line breaks are added to this example for readability):

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&"+  
UriEncode("max-keys")+"="+UriEncode("20") + "&" +  
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

When a request targets a subresource, the corresponding query parameter value will be an empty string (""). For example, the following URI identifies the ACL subresource on the `examplebucket` bucket:

```
http://s3.amazonaws.com/examplebucket?acl
```

The CanonicalQueryString in this case is as follows:

```
UriEncode("acl") + "=" + ""
```

If the URI does not include a '?', there is no query string in the request, and you set the canonical query string to an empty string (""). You will still need to include the "\n".

- *CanonicalHeaders* is a list of request headers with their values. Individual header name and value pairs are separated by the newline character ("\n"). Header names must be in lowercase. You must sort the header names alphabetically to construct the string, as shown in the following example:

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"  
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"  
...  
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

The Lowercase() and Trim() functions used in this example are described in the preceding section.

The *CanonicalHeaders* list must include the following:

- HTTP host header.
- If the Content-MD5 header is present in the request, you must add it to the *CanonicalHeaders* list.
- Any x-amz-* headers that you plan to include in your request must also be added. For example, if you are using temporary security credentials, you need to include x-amz-security-token in your request. You must add this header in the list of *CanonicalHeaders*.

Note

- The x-amz-content-sha256 header is required for all Amazon Signature Version 4 requests. It provides a hash of the request payload. However, you are not required to include the x-amz-content-sha256 as a canonical header because S3 will automatically use its value when calculating the payload hash sent in the request.
- If there is no payload, you must provide the hash of an empty string.
- If you do not want S3 to check against the hash of the request, you can use the literal string "UNSIGNED-PAYLOAD" instead.

The following are example CanonicalHeaders strings. The header names are in lowercase and sorted.

Example 1

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

Example 2

```
host:s3.amazonaws.com
x-amz-content-sha256:UNSIGNED-PAYLOAD
x-amz-date:20130708T220855Z
```

Example 3

```
host:s3.amazonaws.com
x-amz-date:20130708T220855Z
```

Note

For the purpose of calculating an authorization signature, the host header and all x-amz-* headers, excluding x-amz-content-sha256, are required. However, in order to prevent data tampering, you should consider including all the headers in the signature calculation.

Signing the x-amz-content-sha256 header is optional because S3 will use its value when calculating the received request payload hash.

- **SignedHeaders** is an alphabetically sorted, semicolon-separated list of lowercase request header names. The request headers in the list are the same headers that you included in the CanonicalHeaders string. In the previous examples, the value of **SignedHeaders** would be as follows:

Examples 1 and 2

```
host;x-amz-content-sha256;x-amz-date
```

Example 3

```
host;x-amz-date
```

- *HashedPayload* is the hexadecimal value of the SHA256 hash of the request payload.

```
Hex(SHA256Hash(<payload>))
```

If there is no payload in the request, you compute a hash of the empty string as follows:

```
Hex(SHA256Hash(""))
```

The hash returns the following value:

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

For example, when you upload an object by using a PUT request, you provide object data in the body. When you retrieve an object by using a GET request, you compute the empty string hash.

Task 2: Create a String to Sign

This section provides an overview of creating a string to sign. For step-by-step instructions, see [Task 2: Create a String to Sign](#) in the *Amazon Web Services General Reference*.

The string to sign is a concatenation of the following strings:

```
"AWS4-HMAC-SHA256" + "\n" +  
timestampISO8601Format + "\n" +  
<Scope> + "\n" +  
Hex(SHA256Hash(<CanonicalRequest>))
```

The constant string `AWS4-HMAC-SHA256` specifies the hash algorithm that you are using, HMAC-SHA256. The `timeStamp` is the current UTC time in ISO 8601 format (for example, `20130524T000000Z`).

Scope binds the resulting signature to a specific date, an Amazon Web Services Region, and a service. Thus, your resulting signature will work only in the specific Region and for a specific service. The signature is valid for seven days after the specified date.

```
date.Format(<YYYYMMDD>) + "/" + <region> + "/" + <service> + "/aws4_request"
```

For Amazon S3, the service string is `s3`. For a list of *region* strings, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*. The Region column in this table provides the list of valid Region strings.

The following scope restricts the resulting signature to the `us-east-1` Region and Amazon S3.

```
20130606/us-east-1/s3/aws4_request
```

Note

Scope must use the same date that you use to compute the signing key, as discussed in the following section.

Task 3: Calculate Signature

In Amazon Signature Version 4, instead of using your Amazon access keys to sign a request, you first create a signing key that is scoped to a specific Region and service. For more information about signing keys, see [Introduction to Signing Requests](#).

```
DateKey           = HMAC-SHA256("AWS4"+"<SecretAccessKey>", "<YYYYMMDD>")
DateRegionKey     = HMAC-SHA256(<DateKey>, "<aws-region>")
DateRegionServiceKey = HMAC-SHA256(<DateRegionKey>, "<aws-service>")
SigningKey        = HMAC-SHA256(<DateRegionServiceKey>, "aws4_request")
```

 **Note**

Some use cases can process signature keys for up to 7 days. For more information see [Share an Object with Others](#).

For a list of Region strings, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

Using a signing key enables you to keep your Amazon credentials in one safe place. For example, if you have multiple servers that communicate with Amazon S3, you share the signing key with those servers; you don't have to keep a copy of your secret access key on each server. Signing key is valid for up to seven days. So each time you calculate signing key you will need to share the signing key with your servers. For more information, see [Authenticating Requests \(Amazon Signature Version 4\)](#).

The final signature is the HMAC-SHA256 hash of the string to sign, using the signing key as the key.

```
HMAC-SHA256(SigningKey, StringToSign)
```

For step-by-step instructions on creating a signature, see [Task 3: Create a Signature](#) in the *Amazon Web Services General Reference*.

Examples: Signature Calculations

You can use the examples in this section as a reference to check signature calculations in your code. The calculations shown in the examples use the following data:

- Example access keys.

Parameter	Value
AWSSecretAccessKey	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- Request timestamp of 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT).
- Bucket name examplebucket.
- The bucket is assumed to be in the US East (N. Virginia) Region. The credential Scope and the Signing Key calculations use us-east-1 as the Region specifier. For information about other Regions, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.
- You can use either path-style or virtual hosted-style requests. The following examples show how to sign a virtual hosted-style request, for example:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

For more information, see [Virtual Hosting of Buckets](#) in the *Amazon Simple Storage Service User Guide*.

Example: GET Object

The following example gets the first 10 bytes of an object (test.txt) from examplebucket. For more information about the API action, see [GetObject](#).

```
GET /test.txt HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
Range: bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date: 20130524T000000Z
```

Because this GET request does not provide any body content, the x-amz-content-sha256 value can either be the hash of the empty request body or the literal string "UNSIGNED-PAYLOAD". The following steps show signature calculations and construction of the Authorization header using the hash of an empty string.

1. StringToSign

a. CanonicalRequest

```
GET
/test.txt
```



```
host:examplebucket.s3.amazonaws.com
range:bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z
```

```
host;range;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

In the canonical request string, the last line is the hash of the empty request body. The third line is empty because there are no query parameters in the request.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
7344ae5b7ee6c3e7e6b0fe0640412a37625d1fbfff95c48bbb2dc43964946972
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

4. Authorization header

The resulting Authorization header is as follows:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;range;x-amz-content-sha256;x-amz-
date,Signature=f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

Example: PUT Object

This example PUT request creates an object (test\$file.text) in examplebucket. The example assumes the following:

- You are requesting REDUCED_REDUNDANCY as the storage class by adding the x-amz-storage-class request header. For information about storage classes, see [Storage Classes](#) in the *Amazon Simple Storage Service User Guide*.
- The content of the uploaded file is a string, "Welcome to Amazon S3." The value of x-amz-content-sha256 in the request is based on this string.

For information about the API action, see [PutObject](#).

```
PUT test$file.text HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Fri, 24 May 2013 00:00:00 GMT
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
x-amz-content-sha256: 44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
```

<Payload>

The following steps show signature calculations.

1. StringToSign

a. CanonicalRequest

```
PUT
/test%24file.text

date:Fri, 24 May 2013 00:00:00 GMT
host:examplebucket.s3.amazonaws.com
x-amz-content-sha256:44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
x-amz-date:20130524T000000Z
x-amz-storage-class:REDUCED_REDUNDANCY
```

```
date;host;x-amz-content-sha256;x-amz-date;x-amz-storage-class
44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
```

In the canonical request, the third line is empty because there are no query parameters in the request. The `x-amz-content-sha256` canonical header may optionally be signed since its payload hash is already provided at the bottom of the request. The last line is the hash of the body, which should be the same as the `x-amz-content-sha256` header value sent to S3 in the HTTP request.

b. **StringToSign**

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
9e0e90d9c76de8fa5b200d8c849cd5b8dc7a3be3951ddb7f6a76b4158342019d
```

2. **SigningKey**

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. **Signature**

```
98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

4. **Authorization header**

The resulting Authorization header is as follows:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=date;host;x-amz-content-sha256;x-amz-date;x-amz-storage-
class,Signature=98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

Example: GET Bucket Lifecycle

The following GET request retrieves the lifecycle configuration of `examplebucket`. For information about the API action, see [GetBucketLifecycleConfiguration](#).

```
GET ?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256: e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

Because the request does not provide any body content, the `x-amz-content-sha256` header value is the hash of the empty request body. The following steps show signature calculations.

1. StringToSign

a. CanonicalRequest

```
GET
/
lifecycle=
host:examplebucket.s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

In the canonical request, the last line is the hash of the empty request body.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
9766c798316ff2757b517bc739a67f6213b4ab36dd5da2f94eaebf79c77395ca
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +  
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

4. Authorization header

The resulting Authorization header is as follows:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/  
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-  
date,Signature=fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

Example: Get Bucket (List Objects)

The following example retrieves a list of objects from `examplebucket` bucket. For information about the API action, see [ListObjects](#).

```
GET ?max-keys=2&prefix=J HTTP/1.1  
Host: examplebucket.s3.amazonaws.com  
Authorization: SignatureToBeCalculated  
x-amz-date: 20130524T000000Z  
x-amz-content-sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

Because the request does not provide a body, the value of `x-amz-content-sha256` is the hash of the empty request body. The following steps show signature calculations.

1. StringToSign

a. CanonicalRequest

```
GET  
/
```

```
max-keys=2&prefix=J
host:examplebucket.s3.amazonaws.com
x-amz-content-
sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

In the canonical string, the last line is the hash of the empty request body.

b. **StringToSign**

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
df57d21db20da04d7fa30298dd4488ba3a2b47ca3a489c74750e0f1e7df1b9b7
```

2. **SigningKey**

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. **Signature**

```
34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

4. **Authorization header**

The resulting Authorization header is as follows:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

Signature Calculations for the Authorization Header: Transferring Payload in Multiple Chunks (Chunked Upload) (Amazon Signature Version 4)

As described in the [Overview](#), when authenticating requests using the Authorization header, you have an option of uploading the payload in chunks. You can send data in fixed size or variable size chunks. This section describes the signature calculation process in chunked upload, how you create the chunk body, and how the delayed signing works where you first upload the chunk, and send its signature in the subsequent chunk. The example section (see [Example: PUT Object](#)) shows signature calculations and resulting Authorization headers that you can use as a test suite to verify your code.

Note

When transferring data in a series of chunks, you must do one of the following:

- Explicitly specify the total content length (object length in bytes plus metadata in each chunk) using the Content-Length HTTP header. To do this, you must pre-compute the total length of the payload, including the metadata that you send in each chunk, before starting your request.
- Specify the Transfer-Encoding HTTP header. If you include the Transfer-Encoding header and specify any value other than identity, you must omit the Content-Length header.

For all requests, you must include the x-amz-decoded-content-length header, specifying the size of the object in bytes.

Each chunk signature calculation includes the signature of the previous chunk. To begin, you create a *seed* signature using only the headers. You use the seed signature in the signature calculation of the first chunk. For each subsequent chunk, you create a chunk signature that includes the signature of the previous chunk. Thus, the chunk signatures are chained together; that is, the signature of chunk n is a function $F(\text{chunk } n, \text{signature}(\text{chunk } n-1))$. The chaining ensures that you send the chunks in the correct order.

To perform a chunked upload, do the following:

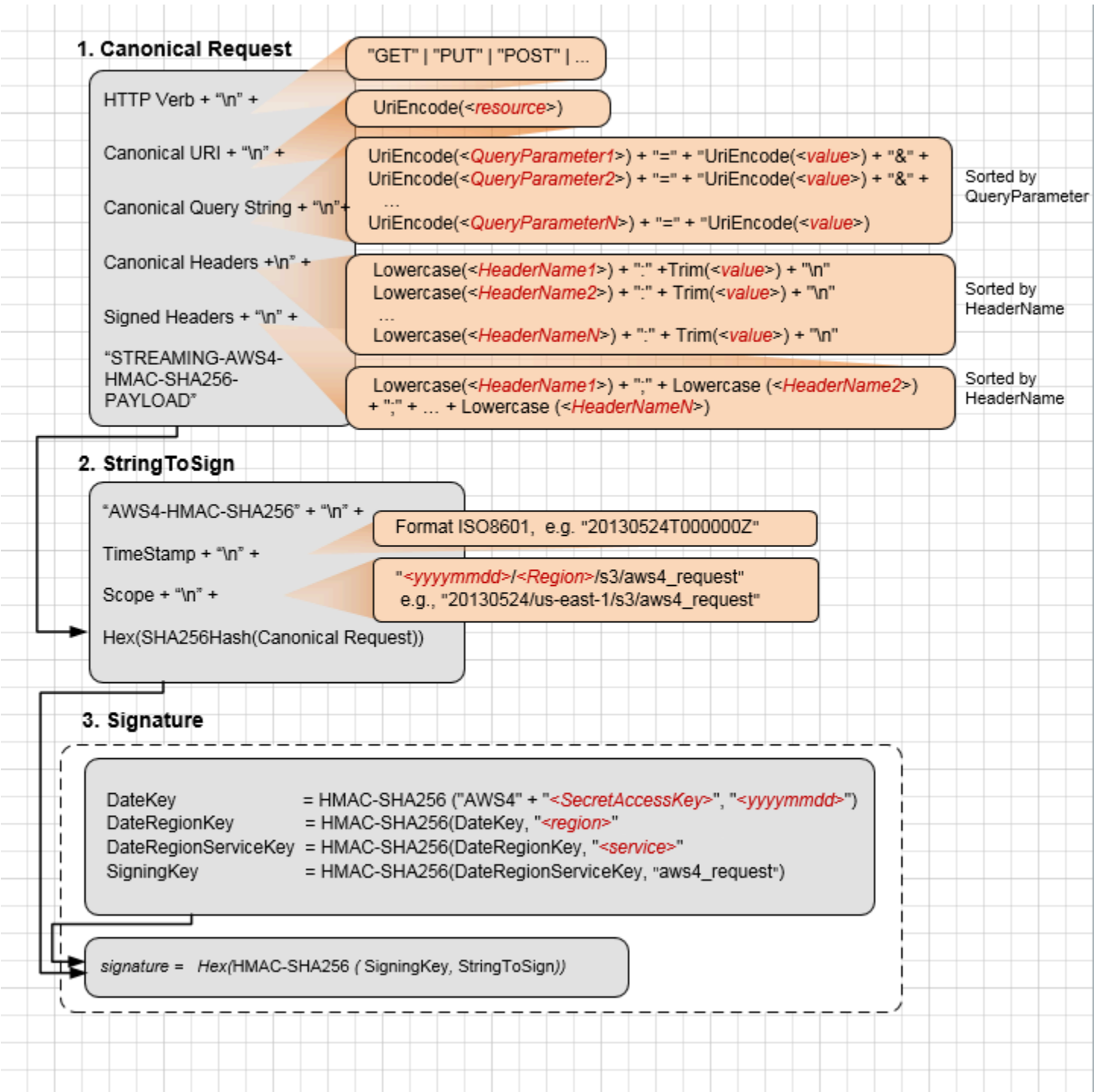
1. Decide the payload chunk size. You need this when you write the code.

The chunk size must be at least 8 KB. We recommend a chunk size of at least 64 KB for better performance. This chunk size applies to all chunks except the last one. The last chunk you send can be smaller than 8 KB. If your payload is small and can fit into one chunk, then it can be smaller than the 8 KB.

2. Create the seed signature for inclusion in the first chunk. For more information, see [Calculating the Seed Signature](#).
3. Create the first chunk and stream it. For more information, see [Defining the Chunk Body](#).
4. For each subsequent chunk, calculate the chunk signature that includes the previous signature in the string you sign, construct the chunk, and send it. For more information, see [Defining the Chunk Body](#).
5. Send the final additional chunk, which is the same as the other chunks in the construction, but it has zero data bytes. For more information, see [Defining the Chunk Body](#).

Calculating the Seed Signature

The following diagram illustrates the process of calculating the seed signature.



The following table describes the functions that are shown in the diagram. You need to implement code for these functions.

Function	Description
<code>Lowercase()</code>	Convert the string to lowercase.
<code>Hex()</code>	Lowercase base 16 encoding.
<code>SHA256Hash()</code>	Secure Hash Algorithm (SHA) cryptographic hash function.
<code>HMAC-SHA256()</code>	Computes HMAC by using the SHA256 algorithm with the signing key provided. This is the final signature.
<code>Trim()</code>	Remove any leading or trailing whitespace.
<code>UriEncode()</code>	URI encode every byte. <code>UriEncode()</code> must enforce the following rules: • URI encode every byte except the unreserved characters: 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', and '~'. • The space character is a reserved character and must be encoded as "%20" (and not as "+"). • Each URI encoded byte is formed by a '%' and the two-digit hexadecimal value of the byte. • Letters in the hexadecimal value must be uppercase, for example "%1A". • Encode the forward slash character, '/', everywhere except in the object key name. For example, if the object key name is <code>photos/Jan/sample.jpg</code>, the forward slash in the key name is not encoded.

 **Important**

The standard `UriEncode` functions provided by your development platform may not work because of differences in implementation and related ambiguity in the underlying RFCs. We recommend that you write

Function	Description
	your own custom UriEncode function to ensure that your encoding will work. The following is an example UriEncode() function in Java. <pre data-bbox="613 499 1450 1245"> public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBui lder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

For information about the signing process, see [Signature Calculations for the Authorization Header: Transferring Payload in a Single Chunk \(Amazon Signature Version 4\)](#). The process is the same, except that the creation of CanonicalRequest differs as follows:

- In addition to the request headers you plan to add, you must include the following headers:

Header	Description
<code>x-amz-content-sha256</code>	This header is required for all Amazon Signature Version 4 requests. Set the value to <code>STREAMING-AWS4-HMAC-SHA256-PAYLOAD</code> to indicate that the signature covers only headers and that there is no payload.
<code>Content-Encoding</code>	Set the value to <code>aws-chunked</code> . Amazon S3 supports multiple content encoding values. You can specify your custom content-encoding when using the Signature Version 4 streaming API. For example: Content-Encoding : <code>aws-chunked,gzip</code> Amazon S3 stores the resulting object without the <code>aws-chunked</code> value in the <code>content-encoding</code> header. If <code>aws-chunked</code> is the only value that you pass in the <code>content-encoding</code> header, S3 considers the <code>content-encoding</code> header empty and does not return this header when you retrieve the object.
<code>x-amz-decoded-content-length</code>	Set the value to the length, in bytes, of the data to be chunked, without counting any metadata. For example, if you are uploading a 4 GB file, set the value to 4294967296. This is the raw size of the object to be uploaded (data you want to store in Amazon S3).

Header	Description
Content-Length	Set the value to the actual size of the transmitted HTTP body, which includes the length of your data (value set for <code>x-amz-decoded-content-length</code>), plus chunk metadata. Each chunk has metadata, such as the signature of the previous chunk. Chunk calculations are discussed in the following section. If you include the <code>Transfer-Encoding</code> header and specify any value other than <code>identity</code> , you must not include the <code>Content-Length</code> header.

You send the first chunk with the seed signature. You must construct the chunk as described in the following section.

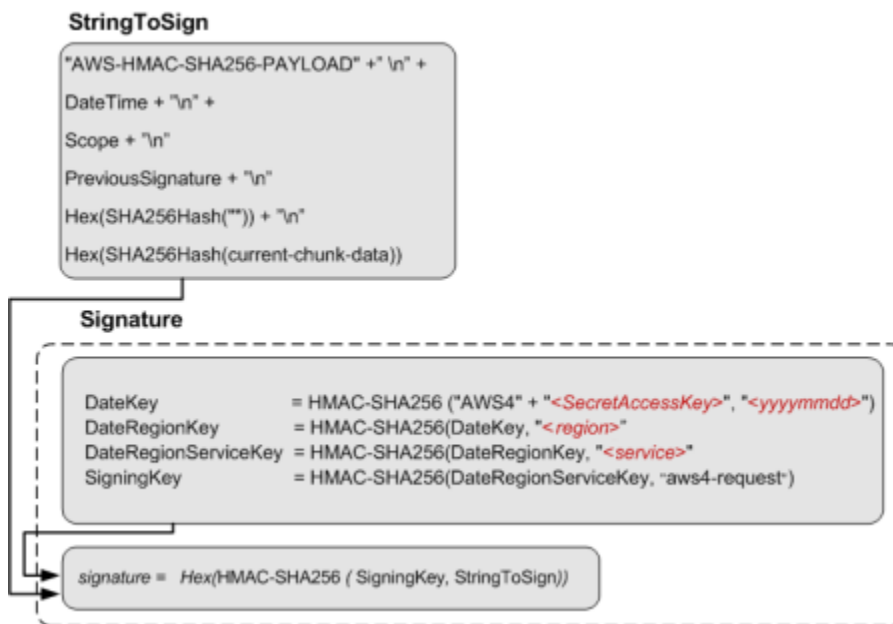
Defining the Chunk Body

All chunks include some metadata. Each chunk must conform to the following structure:

```
string(IntHexBase(chunk-size)) + ";chunk-signature=" + signature + \r\n + chunk-data + \r\n
```

Where:

- `IntHexBase()` is a function that you write to convert an integer chunk-size to hexadecimal. For example, if chunk-size is 65536, hexadecimal string is "10000".
- chunk-size* is the size, in bytes, of the chunk-data, without metadata. For example, if you are uploading a 65 KB object and using a chunk size of 64 KB, you upload the data in three chunks: the first would be 64 KB, the second 1 KB, and the final chunk with 0 bytes.
- signature* For each chunk, you calculate the signature using the following string to sign. For the first chunk, you use the seed-signature as the previous signature.



The size of the final chunk data that you send is 0, although the chunk body still contains metadata, including the signature of the previous chunk.

Example: PUT Object

You can use the examples in this section as a reference to check signature calculations in your code. Before you review the examples, note the following:

- The signature calculations in these examples use the following example security credentials.

Parameter	Value
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- All examples use the request timestamp 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT).
- All examples use examplebucket as the bucket name.

- The bucket is assumed to be in the US East (N. Virginia) Region, and the credential Scope and the Signing Key calculations use `us-east-1` as the Region specifier. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.
- You can use either path style or virtual-hosted style requests. The following examples use virtual-hosted style requests, for example:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

For more information, see [Virtual Hosting of Buckets](#) in the *Amazon Simple Storage Service User Guide*.

The following example sends a PUT request to upload an object. The signature calculations assume the following:

- You are uploading a 65 KB text file, and the file content is a one-character string made up of the letter 'a'.
- The chunk size is 64 KB. As a result, the payload is uploaded in three chunks, 64 KB, 1 KB, and the final chunk with 0 bytes of chunk data.
- The resulting object has the key name `chunkObject.txt`.
- You are requesting `REDUCED_REDUNDANCY` as the storage class by adding the `x-amz-storage-class` request header.

For information about the API action, see [PutObject](#). The general request syntax is as follows:

```
PUT /examplebucket/chunkObject.txt HTTP/1.1
Host: s3.amazonaws.com
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
Authorization: SignatureToBeCalculated
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 66560
Content-Length: 66824
<Payload>
```

The following steps show signature calculations.

1. Seed signature — Create String to Sign

a. CanonicalRequest

```
PUT
/examplebucket/chunk0bject.txt

content-encoding:aws-chunked
content-length:66824
host:s3.amazonaws.com
x-amz-content-sha256:STREAMING-AWS4-HMAC-SHA256-PAYLOAD
x-amz-date:20130524T000000Z
x-amz-decoded-content-length:66560
x-amz-storage-class:REDUCED_REDUNDANCY

content-encoding;content-length;host;x-amz-content-sha256;x-amz-date;x-amz-
decoded-content-length;x-amz-storage-class
STREAMING-AWS4-HMAC-SHA256-PAYLOAD
```

In the canonical request, the third line is empty because there are no query parameters in the request. The last line is the constant string provided as the value of the hashed Payload, which should be same as the value of `x-amz-content-sha256` header.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
cee3fed04b70f867d036f722359b0b1f2f0e5dc0efadbc082b76c4c60e316455
```

Note

For information about each of line in the string to sign, see the diagram that explains seed signature calculation.

2. SigningKey


```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Seed Signature

```
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

4. Authorization header

The resulting Authorization header is as follows:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=content-encoding;content-length;host;x-amz-
content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-
class,Signature=4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

5. Chunk 1: (65536 bytes, with value 97 for letter 'a')

a. Chunk string to sign:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

For information about each line in the string to sign, see the preceding diagram that shows various components of the string to sign (for example, the last three lines are, previous-signature, hash(""), and hash(current-chunk-data)).

b. Chunk signature:

```
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
```

c. Chunk data sent:

```
10000;chunk-  
signature=ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648  
<65536-bytes>
```

6. Chunk 2: (1024 bytes, with value 97 for letter 'a')

a. Chunk string to sign:

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648  
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855  
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Chunk signature:

```
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
```

c. Chunk data sent:

```
400;chunk-  
signature=0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
<1024 bytes>
```

7. Chunk 3: (0 byte data)

a. Chunk string to sign:

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855  
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Chunk signature:

```
b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

c. **Chunk data sent:**

```
0; chunk-  
signature=b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

Signature calculations for trailing headers (chunked uploads) (Amazon Signature Version 4)

When authenticating requests using the Authorization header, you can also upload the payload in chunks. When you send the data for the object in chunks, you also have the option of including trailing headers. (For more information, see [Signature Calculations for the Authorization Header: Transferring Payload in Multiple Chunks \(Chunked Upload\) \(Amazon Signature Version 4\)](#).) This section describes the steps you need to take when you want to include a trailing header at the end of your multiple chunk upload.

Important

When you are including trailing headers, you must send the following in your initial header:

- You must set `x-amz-content-sha256` to an appropriate value that indicates a trailer will be included. To see the acceptable values for `x-amz-content-sha256`, see [Authenticating Requests: Using the Authorization Header \(Amazon Signature Version 4\)](#).
- You must set `x-amz-trailer` to indicate the contents you are including in your trailing header.

Trailing headers are only sent after the chunks have been uploaded. Previous chunks are sent as normal and signed as described in the previous sections, including sending the final chunk with a payload of 0 bytes. The trailing headers are included as their own chunk and sent after the final chunk with a payload of 0 bytes. For example, if your data ended with a 100 KB chunk, you would send the following:

- Previous data chunks
- 100 KB final chunk of the object
- 0 bytes chunk signifying the end of the object
- Trailing headers chunk

Examples: Checking signature calculations

You can use the examples in this section as a reference to check signature calculations in your code. Before you review the examples, note the following:

- The signature calculations in these examples use the following example security credentials.

Parameter	Value
AWSSecretAccessKey	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

- All examples use the request timestamp 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT).
- All examples use `examplebucket` as the bucket name.
- The bucket is assumed to be in the US East (N. Virginia) Region, and the credential Scope and the Signing Key calculations use `us-east-1` as the Region specifier. For more information, see [Regions and endpoints](#) in the *Amazon Web Services General Reference*.
- You can use either path style or virtual-hosted style requests. The following examples use virtual-hosted style requests, for example:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

For more information, see [Virtual Hosting of Buckets](#) in the *Amazon Simple Storage Service User Guide*.

The following example sends a PUT request to upload an object. The signature calculations assume the following:

- You are uploading a 65 KB text file, and the file content is a one-character string made up of the letter 'a'.
- The chunk size is 64 KB. As a result, the payload is uploaded in three chunks, 64 KB, 1 KB, and the final chunk with 0 bytes of chunk data.
- The resulting object has the key name `chunkObject.txt`.
- You are requesting `REDUCED_REDUNDANCY` as the storage class by adding the `x-amz-storage-class` request header.
- The transfer is including a CRC32C checksum value as a trailing header.

For information about the API action, see [PutObject](#). The general request syntax is as follows:

```
PUT /examplebucket/chunkObject.txt HTTP/1.1
Host: s3.amazonaws.com
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
Authorization: SignatureToBeCalculated
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 66560
x-amz-trailer: x-amz-checksum-crc32c
Content-Length: 66946
<Payload>
```

The following steps show signature calculations.

1. Seed signature — Create String to Sign

a. CanonicalRequest

```
PUT
/examplebucket/chunkObject.txt

content-encoding:aws-chunked
host:s3.amazonaws.com
x-amz-content-sha256:STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
x-amz-date:20130524T000000Z
x-amz-decoded-content-length:66560
x-amz-storage-class:REDUCED_REDUNDANCY
x-amz-trailer:x-amz-checksum-crc32c

content-encoding;host;x-amz-content-sha256;x-amz-date;x-amz-decoded-content-
length;x-amz-storage-class;x-amz-trailer
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
```

In the canonical request, the third line is empty because there are no query parameters in the request. The last line is the constant string provided as the value of the hashed Payload, which should be same as the value of `x-amz-content-sha256` header.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
44d48b8c2f70eae815a0198cc73d7a546a73a93359c070abbaa5e6c7de112559
```

Note

For information about each of line in the string to sign, see the diagram that explains seed signature calculation.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Seed Signature

```
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

4. Authorization header

The resulting Authorization header is as follows:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=content-encoding;content-length;host;x-amz-
content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-
class,Signature=106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

5. Chunk 1: (65536 bytes, with value 97 for letter 'a')

a. Chunk string to sign:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
```

```
20130524/us-east-1/s3/aws4_request
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

For information about each line in the string to sign, see the diagram in the preceding topic ([Calculating the Seed Signature](#)) that shows various components of the string to sign. For example, the last three lines consist of the following:

- *previous-signature*
- `hash("")`
- `hash(current-chunk-data)`

b. Chunk signature:

```
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
```

c. Chunk data sent:

```
10000; chunk-
signature=b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
<65536-bytes>
```

6. Chunk 2: (1024 bytes, with value 97 for letter 'a')

a. Chunk string to sign:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Chunk signature:

```
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
```


c. Chunk data sent:

```
400;chunk-  
signature=1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7  
<1024-bytes>
```

7. Chunk 3: (0 byte data)

a. Chunk string to sign:

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7  
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855  
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Chunk signature:

```
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

c. Chunk data sent:

```
0;chunk-  
signature=2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

8. Chunk 4: Trailing headers

a. Trailer chunk string to sign:

```
AWS4-HMAC-SHA256-TRAILER  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992  
1e376db7e1a34a8ef1c4bcee131a2d60a1cb62503747488624e10995f448d774
```

Note

The last two lines are previous-signature (the 0 byte data chunk signature), hash(*trailing-checksum-header-name:base64-encoded-trailing-checksum-value*\n).

The hash is calculated as follows with no whitespace:

- The trailing checksum header name
- A colon (:)
- The base64-encoded trailing checksum value
- A newline character (\n).

In this example, where we are using the `crc32c` hash algorithm, with the base64-encoded checksum value `s008/Q==`, we can represent the computation as follows:
`hash('x-amz-checksum-crc32c:s008/Q==\n')`.

b. Chunk signature:

```
d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

c. Chunk data sent:

```
x-amz-checksum-crc32c:s008/Q==  
x-amz-trailer-  
signature:d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

Authenticating Requests: Using Query Parameters (Amazon Signature Version 4)

As described in the authentication overview (see [Authentication Methods](#)), you can provide authentication information using query string parameters. Using query parameters to authenticate requests is useful when you want to express a request entirely in a URL. This method is also referred as presigning a URL.

A use case scenario for presigned URLs is that you can grant temporary access to your Amazon S3 resources. For example, you can embed a presigned URL on your website or alternatively use it in command line client (such as Curl) to download objects.

 Note

You can also use the Amazon CLI to create presigned URLs. For more information, see [presign](#) in the *Amazon CLI Command Reference*.

The following is an example presigned URL.

```
https://examplebucket.s3.amazonaws.com/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=<your-access-key-id>/20130721/us-east-1/s3/aws4_request
&X-Amz-Date=20130721T201207Z
&X-Amz-Expires=86400
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

In the example URL, note the following:

- The line feeds are added for readability.
- The X-Amz-Credential value in the URL shows the "/" character only for readability. In practice, it should be encoded as %2F. For example:

```
&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request
```

The following table describes the query parameters in the URL that provide authentication information.

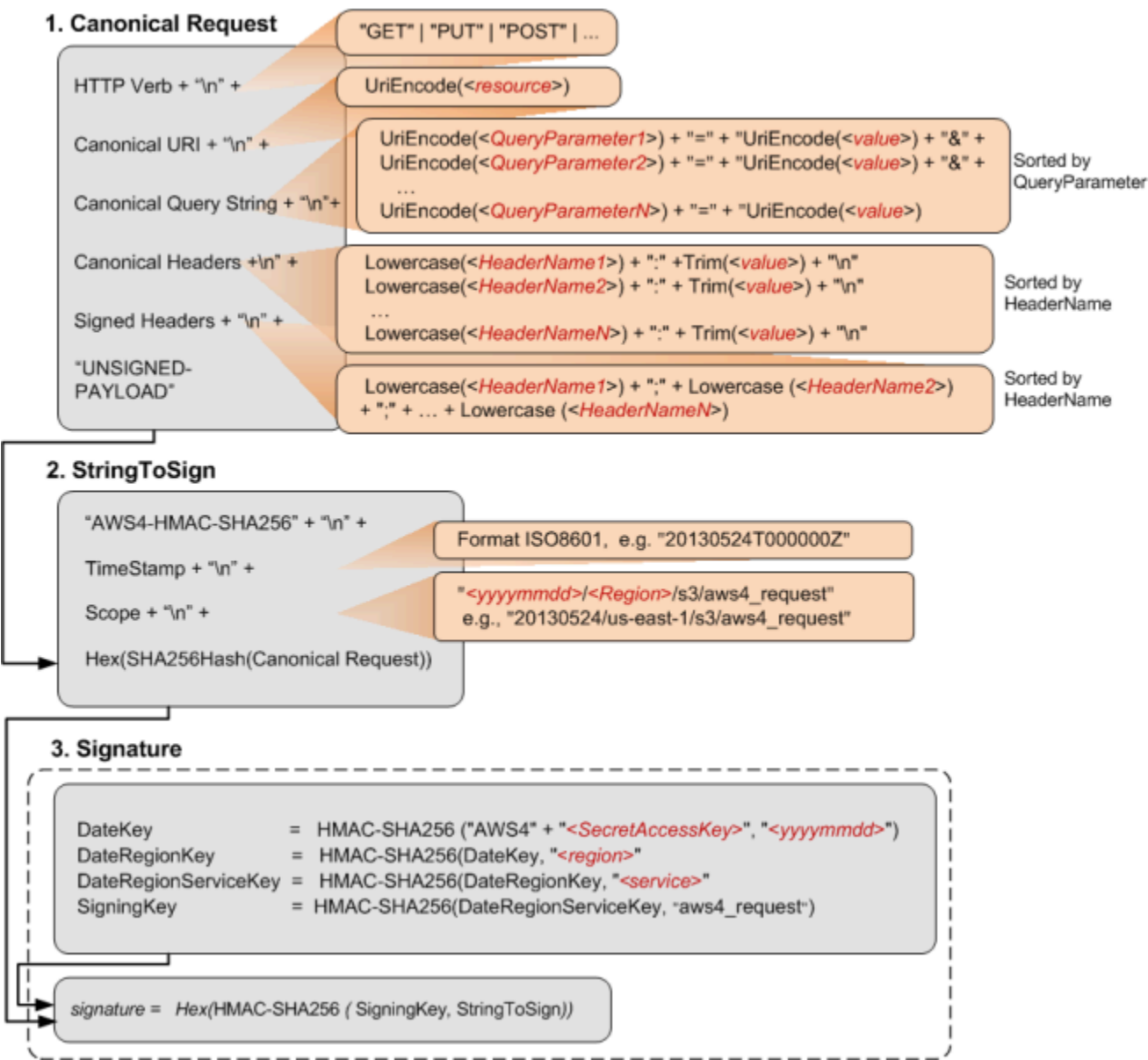
Query String Parameter Name	Example Value
X-Amz-Algorithm	Identifies the version of Amazon Signature and the algorithm that you used to calculate the signature. For Amazon Signature Version 4, you set this parameter value to <code>AWS4-HMAC-SHA256</code> . This string identifies Amazon Sig

Query String Parameter Name	Example Value
	nature Version 4 (Amazon4) and the HMAC-SHA256 algorithm (HMAC-SHA256).
X-Amz-Credential	In addition to your access key ID, this parameter also provides scope (Amazon Web Services Region and service) for which the signature is valid. This value must match the scope you use in signature calculations, discussed in the following section. The general form for this parameter value is as follows: <your-access-key-id> /<date>/<Amazon Web Services Region>/<Amazon-service> /aws4_request For example: AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request For Amazon S3, the <i>Amazon-service</i> string is <code>s3</code>. For a list of S3 AWS-region strings, see Regions and Endpoints in the <i>Amazon Web Services General Reference</i>.
X-Amz-Date	The date and time format must follow the ISO 8601 standard, and must be formatted with the "<i>yyyyMMddTHHmmsZ</i>" format. For example if the date and time was "08/01/2016 15:32:41.982-700" then it must first be converted to UTC (Coordinated Universal Time) and then submitted as "20160801T223241Z".

Query String Parameter Name	Example Value
X-Amz-Expires	Provides the time period, in seconds, for which the generated presigned URL is valid. For example, 86400 (24 hours). This value is an integer. The minimum value you can set is 1, and the maximum is 604800 (seven days). A presigned URL can be valid for a maximum of seven days because the signing key you use in signature calculation is valid for up to seven days.
X-Amz-SignedHeaders	Lists the headers that you used to calculate the signature. The following headers are required in the signature calculations: • The HTTP host header. • Any x-amz-* headers that you plan to add to the request.  Note For added security, you should sign all the request headers that you plan to include in your request.
X-Amz-Signature	Provides the signature to authenticate your request. This signature must match the signature Amazon S3 calculates; otherwise, Amazon S3 denies the request. For example, 733255ef022bec3f2a8701cd61d4b371f3f 28c9f193a1f02279211d48d5193d7 Signature calculations are described in the following section.
X-Amz-Security-Token	Optional credential parameter if using credentials sourced from the STS service.

Calculating a Signature

The following diagram illustrates the signature calculation process.



The following table describes the functions that are shown in the diagram. You need to implement code for these functions.

Function	Description
Lowercase()	Convert the string to lowercase.

Function	Description
Hex()	Lowercase base 16 encoding.
SHA256Hash()	Secure Hash Algorithm (SHA) cryptographic hash function.
HMAC-SHA256()	Computes HMAC by using the SHA256 algorithm with the signing key provided. This is the final signature.
Trim()	Remove any leading or trailing whitespace.

Function	Description
<code>UriEncode()</code>	URI encode every byte. <code>UriEncode()</code> must enforce the following rules: • URI encode every byte except the unreserved characters: 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', and '~'. • The space character is a reserved character and must be encoded as "%20" (and not as "+"). • Each URI encoded byte is formed by a '%' and the two-digit hexadecimal value of the byte. • Letters in the hexadecimal value must be uppercase, for example "%1A". • Encode the forward slash character, '/', everywhere except in the object key name. For example, if the object key name is <code>photos/Jan/sample.jpg</code>, the forward slash in the key name is not encoded.  Important The standard <code>UriEncode</code> functions provided by your development platform may not work because of differences in implementation and related ambiguity in the underlying RFCs. We recommend that you write your own custom <code>UriEncode</code> function to ensure that your encoding will work. The following is an example <code>UriEncode()</code> function in Java. <pre>public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBu ilder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i);</pre>

Function	Description
	<pre> if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

For more information about the signing process (details of creating a canonical request, string to sign, and signature calculations), see [Signature Calculations for the Authorization Header: Transferring Payload in a Single Chunk \(Amazon Signature Version 4\)](#). The process is generally the same except that the creation of **CanonicalRequest** in a presigned URL differs as follows:

- You don't include a payload hash in the **Canonical Request**, because when you create a presigned URL, you don't know the payload content because the URL is used to upload an arbitrary payload. Instead, you use a constant string `UNSIGNED-PAYLOAD`.
- The **Canonical Query String** must include all the query parameters from the preceding table except for `X-Amz-Signature`.
- For `S3`, you must include the `X-Amz-Security-Token` query parameter in the URL if using credentials sourced from the STS service.
- **Canonical Headers** must include the HTTP host header. If you plan to include any of the `x-amz-*` headers, these headers must also be added for signature calculation. You can optionally add all other headers that you plan to include in your request. For added security, you should sign as many headers as possible. If you add a signed header that is also a signed query parameter, and they differ in value, you will receive an `InvalidRequest` error as the input is conflicting.

An Example

Suppose you have an object `test.txt` in your `examplebucket` bucket. You want to share this object with others for a period of 24 hours (86400 seconds) by creating a presigned URL.

```
https://examplebucket.s3.amazonaws.com/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request
&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

The following steps illustrate first the signature calculations and then construction of the presigned URL. The example makes the following additional assumptions:

- Request timestamp is `Fri, 24 May 2013 00:00:00 GMT`.
- The bucket is in the US East (N. Virginia) region, and the credential Scope and the Signing Key calculations use `us-east-1` as the region specifier. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

You can use this example as a test case to verify the signature that your code calculates; however, you must use the same bucket name, object key, time stamp, and the following example credentials:

Parameter	Value
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

1. StringToSign
- a. CanonicalRequest

```
GET
```

```
/test.txt
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE
%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
host:examplebucket.s3.amazonaws.com

host
UNSIGNED-PAYLOAD
```

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
3bfa292879f6447bbcd47001decf97f4a54dc650c8942174ae0a9121cf58ad04
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
aeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```

Now you have all information to construct a presigned URL. The resulting URL for this example is shown as follows (you can use this to compare your presigned URL):

```
https://examplebucket.s3.amazonaws.com/test.txt?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=aeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```

Example 2

The following is an example (unrelated to the previous example) showing a presigned URL with the X-Amz-Security-Token parameter.

```
https://examplebucket.s3.us-east-1.amazonaws.com/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request
&X-Amz-Date=20200524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEMv%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJGMEQCIBSubVdj9YGs2g0HkHs0HFdkw0ozjARSKHL987NhhOC8AiBPepRU1obMvIbGU0T%2BWphFPgK%2Fqpxaf5Snm5M57XFkCqlAgjz%2F%2F%2F%2F%2F%2F%2F%2F%2F8BEAAaDDQ3MjM4NTU0NDY2MCIM83pULBe5%2F%2BNm1GZBKvkBVslSaJVgwSef7SsoZCJlfJ56weYl3QCWEGr2F4BmCZZyFpmWEYzWnhNK1AnHMj5nkfKlKBx30XAT5PZGVr%2F3HhM0kpdanMXn%2B4PY8lvM8RgnzSu90j0UpGXEOAo%2F6G80qlMim3%2BZmaQmasn4VYRVeSEd7072QGZ3%2BvDnDVnss0lSYjlV8PP7IujnvHzRnj0WoeOyMe1lL0wTG%2Fa9usH5HE52w%2FYUJccOn00aZuyR0uVsRV4Q70sbWQHuvYUt%2B0tUMKzm8vsFOp4BaNZFqobbjtbt36Y92v%2Bx5kY6i0s8QE886jJtUWMP5ldMziClGx3p0mN5dzsYlM3GyiJ%2F01mWkPQDwg3mtSp0A9oeeuAMPTA7qmQy9RNuTKBDsx9EW27wvpZBum3SJhEfXv48euadKgriX3Z79ruQFSQ0c9LUrDjr%2B4SoWAJqK%2BGX8Q3vPSjsLxhqhemWD6U4TXcm7ku3gxMbzbqfT8NDg%3D
&X-Amz-Signature=<signature-value>
```

Examples: Signature Calculations in Amazon Signature Version 4

Topics

- [Signature Calculation Examples Using Java \(Amazon Signature Version 4\)](#)
- [Examples of Signature Calculations Using C# \(Amazon Signature Version 4\)](#)

For authenticated requests, unless you are using the Amazon SDKs, you have to write code to calculate signatures that provide authentication information in your requests. Signature calculation in Amazon Signature Version 4 (see [Authenticating Requests \(Amazon Signature Version 4\)](#)) can be a complex undertaking, and we recommend that you use the Amazon SDKs whenever possible.

This section provides examples of signature calculations written in Java and C#. The code samples send the following requests and use the HTTP Authorization header to provide authentication information:

- **PUT object** – Separate examples illustrate both uploading the full payload at once and uploading the payload in chunks. For information about using the Authorization header for authentication, see [Authenticating Requests: Using the Authorization Header \(Amazon Signature Version 4\)](#).
- **GET object** – This example generates a presigned URL to get an object. Query parameters provide the signature and other authentication information. Users can paste a presigned URL in their browser to retrieve the object, or you can use the URL to create a clickable link. For information about using query parameters for authentication, see [Authenticating Requests: Using Query Parameters \(Amazon Signature Version 4\)](#).

The rest of this section describes the examples in Java and C#. The topics include instructions for downloading the samples and for executing them.

Signature Calculation Examples Using Java (Amazon Signature Version 4)

The Java sample that shows signature calculation can be downloaded at <https://docs.amazonaws.cn/AmazonS3/latest/API/samples/AWSS3SigV4JavaSamples.zip>. Note that these samples use Amazon SDK for Java v1, and we recommend using [Amazon SDK for Java v2](#) for new applications. In `RunAllSamples.java`, the `main()` function executes sample requests to create an object, retrieve an object, and create a presigned URL for the object. The sample creates an object from the text string provided in the code:

```
PutS3ObjectSample.putS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
GetS3ObjectSample.getS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
PresignedUrlSample.getPresignedUrlToS3Object(bucketName, regionName, awsAccessKey,
awsSecretKey);
PutS3ObjectChunkedSample.putS3ObjectChunked(bucketName, regionName, awsAccessKey,
awsSecretKey);
```

To test the examples on a Linux-based computer

The following instructions are for the Linux operating system.

1. In a terminal, navigate to the directory that contains `AWSS3SigV4JavaSamples.zip`.
2. Extract the .zip file.
3. In a text editor, open the file `./com/amazonaws/services/s3/samples/RunAllSamples.java`. Update code with the following information:

- The name of a bucket where the new object can be created.

 Note

The examples use a virtual-hosted style request to access the bucket. To avoid potential errors, ensure that your bucket name conforms to the bucket naming rules as explained in [Bucket Restrictions and Limitations](#) in the *Amazon Simple Storage Service User Guide*.

- Amazon Web Services Region where the bucket resides.

If bucket is in the US East (N. Virginia) region, use `us-east-1` to specify the region. For a list of other Amazon Web Services Regions, go to [Amazon Simple Storage Service \(S3\)](#) in the *Amazon Web Services General Reference*.

4. Compile the source code and store the compiled classes into the `bin/` directory.

```
javac -d bin -source 6 -verbose com
```

5. Change the directory to `bin/`, and then run `RunAllSamples`.

```
java com.amazonaws.services.s3.sample.RunAllSamples
```

The code runs all the methods in `main()`. For each request, the output will show the canonical request, the string to sign, and the signature.

Examples of Signature Calculations Using C# (Amazon Signature Version 4)

The C# sample that shows signature calculation can be downloaded at https://docs.amazonaws.cn/AmazonS3/latest/API/samples/AmazonS3SigV4_Samples_CSharp.zip. Note that these samples use Amazon SDK for .NET v1, and we recommend using [Amazon SDK for .NET v3](#) for new applications. In `Program.cs`, the `main()` function executes sample requests to create an object, retrieve an object, and create a presigned URL for the object. The code for signature calculation is in the `\Signers` folder.

```
PutS3ObjectSample.Run(awsRegion, bucketName, "MySampleFile.txt");

Console.WriteLine("\n\n*****");
PutS3ObjectChunkedSample.Run(awsRegion, bucketName, "MySampleFileChunked.txt");

Console.WriteLine("\n\n*****");
GetS3ObjectSample.Run(awsRegion, bucketName, "MySampleFile.txt");

Console.WriteLine("\n\n*****");
PresignedUrlSample.Run(awsRegion, bucketName, "MySampleFile.txt");
```

To test the examples with Microsoft Visual Studio 2010 or later

1. Extract the .zip file.
2. Start Visual Studio, and then open the .sln file.
3. Update the App.config file with valid security credentials.
4. Update the code as follows:
 - In Program.cs, provide the bucket name and the Amazon Web Services Region where the bucket resides. The sample creates an object in this bucket.
5. Run the code.
6. To verify that the object was created, copy the presigned URL that the program creates, and then paste it in a browser window.

Authenticating Requests: Browser-Based Uploads Using POST (Amazon Signature Version 4)

Amazon S3 supports HTTP POST requests so that users can upload content directly to Amazon S3. Using HTTP POST to upload content simplifies uploads and reduces upload latency where users upload data to store in Amazon S3. This section describes how you authenticate HTTP POST requests. For more information about HTTP POST requests, how to create a form, create a POST policy, and an example, see [Browser-Based Uploads Using POST \(Amazon Signature Version 4\)](#).

To authenticate an HTTP POST request you do the following:

1. The form must include the following fields to provide signature and relevant information that Amazon S3 can use to re-calculate the signature upon receiving the request:

Element Name	Description
policy	The Base64-encoded security policy that describes what is permitted in the request. For signature calculation this policy is the string you sign. Amazon S3 must get this policy so it can re-calculate the signature.
x-amz-algorithm	The signing algorithm used. For Amazon Signature Version 4, the value is AWS4-HMAC-SHA256 .
x-amz-credential	In addition to your access key ID, this provides scope information you used in calculating the signing key for signature calculation. It is a string of the following form: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> For example: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .</pre> For Amazon S3, the <i>aws-service</i> string is s3. For a list of Amazon S3 <i>aws-region</i> strings, see Regions and Endpoints in the <i>Amazon Web Services General Reference</i>.

Element Name	Description
x-amz-date	It is the date value in ISO8601 format. For example, 20130728T000000Z . It is the same date you used in creating the signing key. This must also be the same value you provide in the policy (x-amz-date) that you signed.
x-amz-signature	(Amazon Signature Version 4) The HMAC-SHA256 hash of the security policy. For more information on options for the signature , see Add the signature to the HTTP request in the <i>Amazon Web Services General Reference</i>.

2. The POST policy must include the following elements:

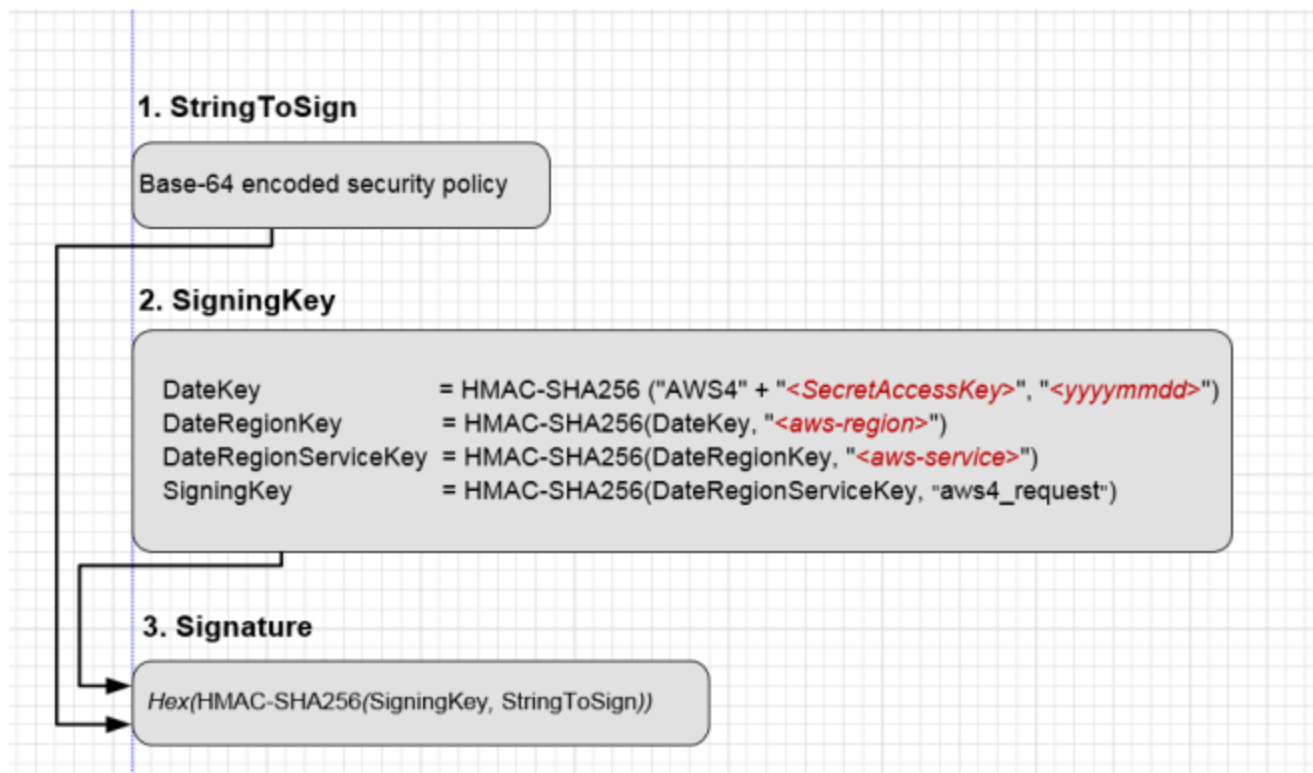
Element Name	Description
x-amz-algorithm	The signing algorithm that you used to calculation the signature. For Amazon Signature Version 4, the value is AWS4-HMAC-SHA256 .
x-amz-credential	In addition to your access key ID, this provides scope information you used in calculating the signing key for signature calculation. It is a string of the following form: <i><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</i> For example, AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .

Element Name	Description
x-amz-date	The date value specified in the ISO8601 formatted string. For example, "20130728T000000Z". The date must be the same that you used in creating the signing key for signature calculation.

3. For signature calculation the POST policy is the string to sign.

Calculating a Signature

The following diagram illustrates the signature calculation process.



To Calculate a signature

1. Create a policy using UTF-8 encoding.
2. Convert the UTF-8-encoded policy to Base64. The result is the string to sign.

3. Create the signature as an HMAC-SHA256 hash of the string to sign. You will provide the signing key as key to the hash function.
4. Encode the signature by using hex encoding.

For more information about creating HTML forms, security policies, and an example, see the following subtopics:

- [Creating an HTML Form \(Using Amazon Signature Version 4\)](#)
- [POST Policy](#)
- [Example: Browser-Based Upload using HTTP POST \(Using Amazon Signature Version 4\)](#)

Amazon S3 Signature Version 4 Authentication Specific Policy Keys

The following table shows the policy keys related Amazon S3 Signature Version 4 authentication that can be in Amazon S3 policies. In a bucket policy, you can add these conditions to enforce specific behavior when requests are authenticated by using Signature Version 4. For example policies, see [Bucket policy examples using Signature Version 4 related condition keys](#).

Applicable Keys	Description
s3:signatureversion	Identifies the version of Amazon Signature that you want to support for authenticated requests. For authenticated requests, Amazon S3 supports both Signature Version 4 and Signature Version 2. You can add this condition in your bucket policy to require a specific signature version. Valid values: "AWS" identifies Signature Version 2 "AWS4-HMAC-SHA256" identifies Signature Version 4

Applicable Keys	Description
s3:authType	Amazon S3 supports various methods of authentication (see Authenticating Requests (Amazon Signature Version 4)). You can optionally use this condition key to restrict incoming requests to use a specific authentication method. For example, you can allow only the HTTP Authorization header to be used in request authentication. Valid values: REST-HEADER REST-QUERY-STRING POST

Applicable Keys	Description
s3:signatureAge	The length of time, in milliseconds, that a signature is valid in an authenticated request. This condition works for: • <i>Presigned URLs</i> — where the most restrictive condition wins. For more information, see Working with presigned URLs. • <i>Presigned POST</i> — upload files directly to S3 using pre-signed POST. For more information, see Amazon S3 POST Policy. In Signature Version 2, this value is always set to 0. In Signature Version 4, the signing key is valid for up to seven days. Therefore, the signatures are also valid for up to seven days. You can use this condition to further limit the signature age. For more information, see Introduction to Signing Requests. Example value: 100

Applicable Keys	Description
s3:x-amz-content-sha256	You can use this condition key to disallow unsigned content in your bucket. When you use Signature Version 4, for requests that use the <code>Authorization</code> header, you add the <code>x-amz-content-sha256</code> header in the signature calculation and then set its value to the hash payload. Note that this condition key doesn't support the <code>x-amz-content-sha256</code> header as a query string parameter. You can use this condition key in your bucket policy to deny any uploads where payloads are not signed. For example, you can deny uploads that use the <code>Authorization</code> header to authenticate requests but don't sign the payload. For more information, see Signature Calculations for the Authorization Header: Transferring Payload in a Single Chunk (Amazon Signature Version 4). Valid value: UNSIGNED-PAYLOAD

Bucket policy examples using Signature Version 4 related condition keys

The following bucket policy denies any Amazon S3 presigned URL request on objects in `examplebucket` if the signature is more than ten minutes old.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```

    {
      "Sid": "Deny a presigned URL request if the signature is more than 10
min old",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws-cn:s3:::examplebucket3/*",
      "Condition": {
        "NumericGreaterThan": {
          "s3:signatureAge": 600000
        }
      }
    }
  ]
}

```

The following bucket policy allows only requests that use the Authorization header for request authentication. Any POST or presigned URL requests will be denied.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow only requests that use Authorization header for
request authentication. Deny POST or presigned URL requests.",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws-cn:s3:::examplebucket3/*",
      "Condition": {
        "StringNotEquals": {
          "s3:authType": "REST-HEADER"
        }
      }
    }
  ]
}

```

The following bucket policy denies requests that use presigned URLs for request authentication:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyUploadsUsingPresignedURL",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws-cn:s3:::amzn-s3-demo-bucket1/*",
      "Condition": {
        "StringEquals": {
          "s3:authType": "REST-query-string"
        }
      }
    }
  ]
}
```

The following bucket policy denies any uploads with unsigned payloads:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Deny uploads with unsigned payloads that use the
Authorization header.",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws-cn:s3:::examplebucket3/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-content-sha256": "UNSIGNED-PAYLOAD"
        }
      }
    }
  ]
}
```



```
    ]
  }
```

Browser-Based Uploads Using POST (Amazon Signature Version 4)

This section discusses how to upload files directly to Amazon S3 through a browser using HTTP POST requests. It also contains information about how to use the Amazon Amplify JavaScript library for browser-based file uploads to Amazon S3.

Topics

- [POST Object](#)
- [POST Object restore](#)
- [Browser-Based Uploads Using HTTP POST](#)
- [Calculating a Signature](#)
- [Creating an HTML Form \(Using Amazon Signature Version 4\)](#)
- [POST Policy](#)
- [Example: Browser-Based Upload using HTTP POST \(Using Amazon Signature Version 4\)](#)
- [Browser-based uploads to Amazon S3 using the Amazon Amplify library](#)

POST Object

Description

The POST operation adds an object to a specified bucket by using HTML forms. POST is an alternate form of PUT that enables browser-based uploads as a way of putting objects in buckets. Parameters that are passed to PUT through HTTP headers are instead passed as form fields to POST in the multipart/form-data encoded message body. To add an object to a bucket, you must have WRITE access on the bucket. Amazon S3 never stores partial objects. If you receive a successful response, you can be confident that the entire object was stored.

Amazon S3 is a distributed system. Unless you've enabled versioning for a bucket, if Amazon S3 receives multiple write requests for the same object simultaneously, only the last version of the object written is stored.

To ensure that data is not corrupted while traversing the network, use the Content-MD5 form field. When you use this form field, Amazon S3 checks the object against the provided MD5 value. If they do not match, Amazon S3 returns an error. Additionally, you can calculate the MD5 value while posting an object to Amazon S3 and compare the returned ETag to the calculated MD5 value. The ETag reflects only changes to the contents of an object, not its metadata.

Note

To configure your application to send the request headers before sending the request body, use the HTTP status code 100 (Continue). For POST operations, using this status code helps you avoid sending the message body if the message is rejected based on the headers (for example, because of an authentication failure or redirect). For more information about the HTTP status code 100 (Continue), go to Section 8.2.3 of <http://www.ietf.org/rfc/rfc2616.txt>.

Amazon S3 automatically encrypts all new objects that are uploaded to an S3 bucket. The encryption setting of an uploaded object depends on the default encryption configuration of the destination bucket. By default, all buckets have a default encryption configuration that uses server-side encryption with Amazon S3 managed keys (SSE-S3).

If the destination bucket has an encryption configuration that uses server-side encryption with an Amazon Key Management Service (Amazon KMS) key (SSE-KMS), dual-layer server-side encryption

with an Amazon KMS key (DSSE-KMS), or a customer-provided encryption key (SSE-C), Amazon S3 uses the corresponding KMS key or customer-provided key to encrypt the uploaded object. When uploading an object, if you want to change the encryption setting of the uploaded object, you can specify the type of server-side encryption. You can configure SSE-S3, SSE-KMS, DSSE-KMS, or SSE-C. For more information, see [Protecting data using server-side encryption](#) in the *Amazon Simple Storage Service User Guide*.

Important

When constructing your request, make sure that the `file` field is the last field in the form.

Versioning

If you enable versioning for a bucket, POST automatically generates a unique version ID for the object being added. Amazon S3 returns this ID in the response by using the `x-amz-version-id` response header.

If you suspend versioning for a bucket, Amazon S3 always uses `null` as the version ID of the object stored in a bucket.

For more information about returning the versioning state of a bucket, see [GetBucketVersioning](#).

Amazon S3 is a distributed system. If you enable versioning for a bucket and Amazon S3 receives multiple write requests for the same object simultaneously, all versions of the object are stored.

To see sample requests that use versioning, see [Sample Request](#).

Requests

Syntax

```
POST / HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
User-Agent: browser_data
Accept: file_types
Accept-Language: Regions
Accept-Encoding: encoding
Accept-Charset: character_set
Keep-Alive: 300
Connection: keep-alive
```

```
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: length

--9431149156168
Content-Disposition: form-data; name="key"

acl
--9431149156168
Content-Disposition: form-data; name="tagging"

<Tagging><TagSet><Tag><Key>Tag Name</Key><Value>Tag Value</Value></Tag></TagSet></
Tagging>
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

success_redirect
--9431149156168
Content-Disposition: form-data; name="Content-Type"

content_type
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

uuid
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

metadata
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

access-key-id
--9431149156168
Content-Disposition: form-data; name="Policy"

encoded_policy
--9431149156168
Content-Disposition: form-data; name="Signature"

signature=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg
```

```
file_content
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

Request Parameters

This implementation of the operation does not use request parameters.

Form Fields

This operation can use the following form fields.

Name	Description	Required
AWSAccessKeyId	The Amazon access key ID of the owner of the bucket who grants an Anonymous user access for a request that satisfies the set of constraints in the policy. Type: String Default: None Constraints: Required if a policy document is included with the request.	Conditional
acl	The specified Amazon S3 access control list (ACL). If the specified ACL is not valid, an error is generated. For more information about ACLs, see Access control list (ACL) overview in the <i>Amazon Simple Storage Service User Guide</i>. Type: String Default: private Valid Values: private public-read public-read-write aws-exec-read	No

Name	Description	Required
	authenticated-read bucket-owner-read bucket-owner-full-control	
Cache-Control , Content-Type , Content-Disposition , Content-Encoding , Expires	The REST-specific headers. For more information, see PutObject. Type: String Default: None	No
file	The file or text content. The file or text content must be the last field in the form. You cannot upload more than one file at a time. Type: File or text content Default: None	Yes
key	The name of the uploaded key. To use the file name provided by the user, use the <code>\${filename}</code> variable. For example, if a user named Mary uploads the file <code>example.jpg</code> and you specify <code>/user/mary/\${filename}</code> , the key name is <code>/user/mary/example.jpg</code> . For more information, see Object key and metadata in the <i>Amazon Simple Storage Service User Guide</i>. Type: String Default: None	Yes

Name	Description	Required
policy	The security policy that describes what is permitted in the request. Requests without a security policy are considered anonymous and work only on publicly writable buckets. For more information, see HTML forms and Upload examples in the <i>Amazon Simple Storage Service User Guide</i>. Type: String Default: None Constraints: A security policy is required if the bucket is not publicly writable.	Conditional

Name	Description	Required
success_action_redirect , redirect	The URL to which the client is redirected upon a successful upload. If <code>success_action_redirect</code> is not specified , Amazon S3 returns the empty document type specified in the <code>success_action_status</code> field. If Amazon S3 cannot interpret the URL, it acts as if the field is not present. If the upload fails, Amazon S3 displays an error and does not redirect the user to a URL. Type: String Default: None  Note The <code>redirect</code> field name is deprecated, and support for the <code>redirect</code> field name will be removed in the future.	No

Name	Description	Required
success_action_status	If you don't specify <code>success_action_redirect</code>, the status code is returned to the client when the upload succeeds. This field accepts the values 200, 201, or 204 (the default). If the value is set to 200 or 204, Amazon S3 returns an empty document with a 200 or 204 status code. If the value is set to 201, Amazon S3 returns an XML document with a 201 status code. If the value is not set or if it is set to a value that is not valid, Amazon S3 returns an empty document with a 204 status code. Type: String Default: None	No

Name	Description	Required
tagging	The specified set of tags to add to the object. To add tags, use the following encoding scheme. <pre><Tagging> <TagSet> <Tag> <Key>TagName</Key> <Value>TagValue</Value> </Tag> ... </TagSet> </Tagging></pre> For more information, see Object tagging in the <i>Amazon Simple Storage Service User Guide</i>. Type: String Default: None	No
x-amz-storage-class	The storage class to use for storing the object. If you don't specify a class, Amazon S3 uses the default storage class, STANDARD. Amazon S3 supports other storage classes. For more information, see Storage classes in the <i>Amazon Simple Storage Service User Guide</i>. Type: String Default: STANDARD Valid values: REDUCED_REDUNDANCY EXPRESS_ONEZONE DEEP_ARCHIVE GLACIER GLACIER_IR INTELLIGENT_TIERING ONEZONE_IA STANDARD STANDARD_IA	No

Name	Description	Required
x-amz-meta-*	Headers starting with this prefix are user-defined metadata. Each one is stored and returned as a set of key-value pairs. Amazon S3 doesn't validate or interpret user-defined metadata. For more information, see PutObject. Type: String Default: None	No
x-amz-security-token	The Amazon DevPay security token. Each request that uses Amazon DevPay requires two x-amz-security-token form fields: one for the product token and one for the user token. Type: String Default: None	No
x-amz-signature	(Amazon Signature Version 4) The HMAC-SHA256 hash of the security policy. Type: String Default: None	Conditional

Name	Description	Required
<code>x-amz-website-redirect-location</code>	If the bucket is configured as a website, this field redirects requests for this object to another object in the same bucket or to an external URL. Amazon S3 stores the value of this header in the object metadata. For information about object metadata, see Object key and metadata in the <i>Amazon Simple Storage Service User Guide</i>. In the following example, the request header sets the redirect to an object (<code>anotherPage.html</code>) in the same bucket: <pre>x-amz-website-redirect-location: /anotherPage.html</pre> In the following example, the request header sets the object redirect to another website: <pre>x-amz-website-redirect-location: http://www.example.com/</pre> For more information about website hosting in Amazon S3, see Hosting websites on Amazon S3 and How to configure website page redirects in the <i>Amazon Simple Storage Service User Guide</i>. Type: String Default: None Constraints: The value must be prefixed by <code>/</code>, <code>http://</code>, or <code>https://</code>. The length of the value is limited to 2 KB.	No

Additional Checksum Request Form Fields

When uploading an object, you can specify various checksums that you would like to use to verify your data integrity. You can specify one additional checksum algorithm for Amazon S3 to use. For more information about additional checksum values, see [Checking object integrity](#) in the *Amazon Simple Storage Service User Guide*.

Name	Description	Required
x-amz-checksum-algorithm	Indicates the algorithm used to create the checksum for the object. If a value is specified, you must include the matching checksum header. Otherwise, your request will generate a 400 error. Possible values include CRC32, CRC32C, SHA1, and SHA256.	No
x-amz-checksum-crc32	Specifies the base64-encoded, 32-bit CRC32 checksum of the object. This parameter is required if the value of x-amz-checksum-algorithm is CRC32.	Conditional
x-amz-checksum-crc32c	Specifies the base64-encoded, 32-bit CRC32C checksum of the object. This parameter is required if the value of x-amz-checksum-algorithm is CRC32C.	Conditional
x-amz-checksum-sha1	Specifies the base64-encoded, 160-bit SHA-1 digest of the object. This parameter is required if the value of x-amz-checksum-algorithm is SHA1.	Conditional
x-amz-checksum-sha256	Specifies the base64-encoded, 256-bit SHA-256 digest of the object. This parameter is required if the value of x-amz-checksum-algorithm is SHA256.	Conditional

Server-Side Encryption Specific Request Form Fields

Server-side encryption is data encryption at rest. Amazon S3 encrypts your data while writing it to disks in Amazon data centers and decrypts your data when you access it. When uploading an object, you can specify the type of server-side encryption that you want Amazon S3 to use for encrypting the object.

There are four types of server-side encryption:

- **Server-side encryption with Amazon S3 managed keys (SSE-S3)** – Starting May 2022, all Amazon S3 buckets have encryption configured by default. The default option for server-side encryption is with SSE-S3. Each object is encrypted with a unique key. As an additional safeguard, SSE-S3 encrypts the key itself with a root key that it regularly rotates. SSE-S3 uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data.
- **Server-side encryption with Amazon KMS keys (SSE-KMS)** – SSE-KMS is provided through an integration of the Amazon KMS service with Amazon S3. With Amazon KMS, you have more control over your keys. For example, you can view separate keys, edit control policies, and follow the keys in Amazon CloudTrail. Additionally, you can create and manage customer managed keys or use Amazon managed keys that are unique to you, your service, and your Region.
- **Dual-layer server-side encryption with Amazon KMS keys (DSSE-KMS)** – Dual-layer server-side encryption with Amazon KMS keys (DSSE-KMS) is similar to SSE-KMS, but applies two individual layers of object-level encryption instead of one layer.
- **Server-side encryption with customer-provided keys (SSE-C)** – With SSE-C, you manage the encryption keys, and Amazon S3 manages the encryption as it writes to disks, and the decryption when you access your objects.

Note

If you have server-side encryption with customer-provided keys (SSE-C) blocked for your general purpose bucket, you will get an HTTP 403 Access Denied error when you specify the SSE-C request headers while writing new data to your bucket. For more information, see [Blocking or unblocking SSE-C for a general purpose bucket](#).

For more information, see [Protecting data using server-side encryption](#) in the *Amazon Simple Storage Service User Guide*.

Depending on which type of server-side encryption you want to use, specify the following form fields.

- **Use SSE-S3, SSE-KMS, or DSSE-KMS** – If you want to use these types of server-side encryption, specify the following form fields in the request.

Name	Description	Required
<code>x-amz-server-side-encryption</code>	Specifies the server-side encryption algorithm to use when Amazon S3 creates an object. To use SSE-S3, specify AES256. To use SSE-KMS, specify <code>aws:kms</code>. To use DSSE-KMS, specify <code>aws:kms:dsse</code> . Type: String Valid Value: <code>aws:kms</code>, AES256, <code>aws:kms:dsse</code>	Yes
<code>x-amz-server-side-encryption-aws-kms-key-id</code>	If the <code>x-amz-server-side-encryption</code> header has a valid value of <code>aws:kms</code> or <code>aws:kms:dsse</code> , this header specifies the ID of the Amazon KMS key that was used to encrypt the object. Type: String	Yes, if the value of <code>x-amz-server-side-encryption</code> is <code>aws:kms</code> or <code>aws:kms:dsse</code>
<code>x-amz-server-side-encryption-context</code>	If <code>x-amz-server-side-encryption</code> has a valid value of <code>aws:kms</code> or <code>aws:kms:dsse</code> , this header specifies the encryption context for the object. The value of this header is a base64-encoded UTF-8 string that contains JSON-formatted key-value pairs for the encryption context. Type: String	No

Name	Description	Required
<code>x-amz-server-side-encryption-bucket-key-enabled</code>	If <code>x-amz-server-side-encryption</code> has a valid value of <code>aws:kms</code> or <code>aws:kms:dsse</code>, this header specifies whether Amazon S3 should use an S3 Bucket Key with SSE-KMS or DSSE-KMS. Setting this header to <code>true</code> causes Amazon S3 to use an S3 Bucket Key for object encryption with SSE-KMS or DSSE-KMS. Type: Boolean	No

Note

If you specify `x-amz-server-side-encryption:aws:kms` or `x-amz-server-side-encryption:aws:kms:dsse`, but do not provide `x-amz-server-side-encryption-aws-kms-key-id`, Amazon S3 uses the Amazon managed key (`aws/S3`) to protect the data.

- **Use SSE-C** – If you want to manage your own encryption keys, you must provide all the following form fields in the request.

Note

If you use SSE-C, the ETag value that Amazon S3 returns in the response is not the MD5 of the object.

Name	Description	Required
<code>x-amz-server-side-encryption-customer-algorithm</code>	Specifies the algorithm to use to when encrypting the object. Type: String Default: None Valid Value: AES256	Yes

Name	Description	Required
	Constraints: Must be accompanied by valid <code>x-amz-server-side-encryption-customer-key</code> and <code>x-amz-server-side-encryption-customer-key-MD5</code> fields.	
<code>x-amz-server-side-encryption-customer-key</code>	Specifies the customer-provided base64-encoded encryption key for Amazon S3 to use in encrypting data. This value is used to store the object, and then it is discarded. Amazon does not store the encryption key. The key must be appropriate for use with the algorithm specified in the <code>x-amz-server-side-encryption-customer-algorithm</code> header. Type: String Default: None Constraints: Must be accompanied by valid <code>x-amz-server-side-encryption-customer-algorithm</code> and <code>x-amz-server-side-encryption-customer-key-MD5</code> fields.	Yes
<code>x-amz-server-side-encryption-customer-key-MD5</code>	Specifies the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. Amazon S3 uses this header for a message-integrity check to ensure that the encryption key was transmitted without error. Type: String Default: None Constraints: Must be accompanied by valid <code>x-amz-server-side-encryption-customer-algorithm</code> and <code>x-amz-server-side-encryption-customer-key</code> fields.	Yes

Responses

Response Headers

This implementation of the operation can include the following response headers in addition to the response headers common to all responses. For more information, see [Common Response Headers](#).

Name	Description
x-amz-checksum-crc32	The base64-encoded, 32-bit CRC32 checksum of the object. Type: String
x-amz-checksum-crc32c	The base64-encoded, 32-bit CRC32C checksum of the object. Type: String
x-amz-checksum-sha1	The base64-encoded, 160-bit SHA-1 digest of the object. Type: String
x-amz-checksum-sha256	The base64-encoded, 256-bit SHA-256 digest of the object. Type: String
x-amz-expiration	If an <code>Expiration</code> action is configured for the object as part of the bucket's lifecycle configuration, Amazon S3 returns this header. The header value includes an <code>expiry-date</code> component and a URL-encoded <code>rule-id</code> component. For version-enabled buckets, this header applies only to current versions. Amazon S3 does not provide a header to indicate when a noncurrent version is eligible for permanent deletion. For more information, see PutBucketLifecycleConfiguration .

Name	Description
	Type: String
success_action_redirect, redirect	The URL to which the client is redirected on a successful upload. Type: String Ancestor: PostResponse
x-amz-server-side-encryption	The server-side encryption algorithm that was used when storing this object in Amazon S3 (for example, AES256, <code>aws:kms</code>, <code>aws:kms:dsse</code>). Type: String
x-amz-server-side-encryption-aws-kms-key-id	If the <code>x-amz-server-side-encryption</code> header has a valid value of <code>aws:kms</code>, this header specifies the ID of the KMS key that was used to encrypt the object. Type: String
x-amz-server-side-encryption-bucket-key-enabled	If <code>x-amz-server-side-encryption</code> has a valid value of <code>aws:kms</code>, this header indicates whether the object is encrypted with SSE-KMS by using an S3 Bucket Key. If this header is set to <code>true</code>, the object uses an S3 Bucket Key with SSE-KMS. Type: Boolean
x-amz-server-side-encryption-customer-algorithm	If SSE-C was requested, the response includes this header, which confirms the encryption algorithm that was used. Type: String Valid Values: AES256

Name	Description
x-amz-server-side-encryption-customer-key-MD5	If SSE-C was requested, the response includes this header to verify round-trip message integrity of the customer-provided encryption key. Type: String
x-amz-version-id	Version of the object. Type: String

Response Elements

Name	Description
Bucket	The name of the bucket that the object was stored in. Type: String Ancestor: PostResponse
ETag	The entity tag (ETag) is an MD5 hash of the object that you can use to do conditional GET operations by using the <code>If-Modified</code> request tag with the GET request operation. ETag reflects changes only to the contents of an object, not to its metadata. Type: String Ancestor: PostResponse
Key	The object key name. Type: String Ancestor: PostResponse
Location	The URI of the object.

Name	Description
	Type: String
	Ancestor: PostResponse

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see [Error Responses](#).

Examples

Sample Request

```
POST /Neo HTTP/1.1
Content-Length: 4
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: authorization string
Content-Type: text/plain
Expect: the 100-continue HTTP status code
```

ObjectContent

Sample Response with Versioning Suspended

The following is a sample response when bucket versioning is suspended:

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdm0dAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiI0Ix1+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: default
Date: Wed, 12 Oct 2009 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: AmazonS3
```

In this response, the version ID is null.

Sample Response with Versioning Enabled

The following is a sample response when bucket versioning is enabled.

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdm0dAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiI0Ix1+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: 43jfkodU8493jnFJD9fjj3HHNVfdsQUIFDNsidf038jfdsjGFDSIRp
Date: Wed, 01 Mar 2006 12:00:00 GMT
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"
Content-Length: 0
Connection: close
Server: AmazonS3
```

Related Resources

- [CopyObject](#)
- [POST Object](#)
- [GetObject](#)

POST Object restore

Description

This operation performs the following types of requests:

- `select` – Perform a select query on an archived object
- `restore an archive` – Restore an archived object

To use this operation, you must have permissions to perform the `s3:RestoreObject` and `s3:GetObject` actions. The bucket owner has this permission by default and can grant this permission to others. For more information about permissions, see [Permissions Related to Bucket Subresource Operations](#) and [Managing Access Permissions to Your Amazon S3 Resources](#) in the *Amazon Simple Storage Service User Guide*.

Querying Archives with Select Requests

You use a select type of request to perform SQL queries on archived objects. The archived objects that are being queried by the select request must be formatted as uncompressed comma-separated values (CSV) files. You can run queries and custom analytics on your archived data without having to restore your data to a hotter Amazon S3 tier. For an overview about select requests, see [Querying Archived Objects](#) in the *Amazon Simple Storage Service User Guide*.

When making a select request, do the following:

- Define an output location for the select query's output. This must be an Amazon S3 bucket in the same Amazon Web Services Region as the bucket that contains the archive object that is being queried. The Amazon Web Services account that initiates the job must have permissions to write to the S3 bucket. You can specify the storage class and encryption for the output objects stored in the bucket. For more information about output, see [Querying Archived Objects](#) in the *Amazon Simple Storage Service User Guide*.

For more information about the S3 structure in the request body, see the following:

- [PutObject](#)
- [Managing Access with ACLs](#) in the *Amazon Simple Storage Service User Guide*
- [Protecting Data Using Server-Side Encryption](#) in the *Amazon Simple Storage Service User Guide*

- Define the SQL expression for the SELECT type of restoration for your query in the request body's `SelectParameters` structure. You can use expressions like the following examples.
- The following expression returns all records from the specified object.

```
SELECT * FROM Object
```

- Assuming that you are not using any headers for data stored in the object, you can specify columns with positional headers.

```
SELECT s._1, s._2 FROM Object s WHERE s._3 > 100
```

- If you have headers and you set the `fileHeaderInfo` in the CSV structure in the request body to `USE`, you can specify headers in the query. (If you set the `fileHeaderInfo` field to `IGNORE`, the first row is skipped for the query.) You cannot mix ordinal positions with header column names.

```
SELECT s.Id, s.FirstName, s.SSN FROM S3Object s
```

For more information about using SQL with Amazon Glacier Select restore, see [SQL Reference for Amazon S3 Select and Amazon Glacier Select](#) in the *Amazon Simple Storage Service User Guide*.

When making a select request, you can also do the following:

- To expedite your queries, specify the Expedited tier. For more information about tiers, see "Restoring Archives," later in this topic.
- Specify details about the data serialization format of both the input object that is being queried and the serialization of the CSV-encoded query results.

The following are additional important facts about the select feature:

- The output results are new Amazon S3 objects. Unlike archive retrievals, they are stored until explicitly deleted—manually or through a lifecycle policy.
- You can issue more than one select request on the same Amazon S3 object. Amazon S3 doesn't deduplicate requests, so avoid issuing duplicate requests.
- Amazon S3 accepts a select request even if the object has already been restored. A select request doesn't return error response 409.

Restoring Archives

Objects in the GLACIER and DEEP_ARCHIVE storage classes are archived. To access an archived object, you must first initiate a restore request. This restores a temporary copy of the archived object. In a restore request, you specify the number of days that you want the restored copy to exist. After the specified period, Amazon S3 deletes the temporary copy but the object remains archived in the GLACIER or DEEP_ARCHIVE storage class that object was restored from.

To restore a specific object version, you can provide a version ID. If you don't provide a version ID, Amazon S3 restores the current version.

The time it takes restore jobs to finish depends on which storage class the object is being restored from and which data access tier you specify.

When restoring an archived object (or using a select request), you can specify one of the following data access tier options in the `Tier` element of the request body:

- **Expedited** - Expedited retrievals allow you to quickly access your data stored in the GLACIER storage class when occasional urgent requests for a subset of archives are required. For all but the largest archived objects (250 MB+), data accessed using Expedited retrievals are typically made available within 1–5 minutes. Provisioned capacity ensures that retrieval capacity for Expedited retrievals is available when you need it. Expedited retrievals and provisioned capacity are not available for the DEEP_ARCHIVE storage class.
- **Standard** - Standard retrievals allow you to access any of your archived objects within several hours. This is the default option for the GLACIER and DEEP_ARCHIVE retrieval requests that do not specify the retrieval option. Standard retrievals typically complete within 3-5 hours from the GLACIER storage class and typically complete within 12 hours from the DEEP_ARCHIVE storage class.
- **Bulk** - Bulk retrievals are Amazon Glacier's lowest-cost retrieval option, enabling you to retrieve large amounts, even petabytes, of data inexpensively in a day. Bulk retrievals typically complete within 5-12 hours from the GLACIER storage class and typically complete within 48 hours from the DEEP_ARCHIVE storage class.

For more information about archive retrieval options and provisioned capacity for Expedited data access, see [Restoring Archived Objects](#) in the *Amazon Simple Storage Service User Guide*.

You can use Amazon S3 restore speed upgrade to change the restore speed to a faster speed while it is in progress. You upgrade the speed of an in-progress restoration by issuing another

restore request to the same object, setting a new `Tier` request element. When issuing a request to upgrade the restore tier, you must choose a tier that is faster than the tier that the in-progress restore is using. You must not change any other parameters, such as the `Days` request element. For more information, see [Upgrading the Speed of an In-Progress Restore](#) in the *Amazon Simple Storage Service User Guide*.

To get the status of object restoration, you can send a HEAD request. Operations return the `x-amz-restore` header, which provides information about the restoration status, in the response. You can use Amazon S3 event notifications to notify you when a restore is initiated or completed. For more information, see [Configuring Amazon S3 Event Notifications](#) in the *Amazon Simple Storage Service User Guide*.

After restoring an archived object, you can update the restoration period by reissuing the request with a new period. Amazon S3 updates the restoration period relative to the current time and charges only for the request—there are no data transfer charges. You cannot update the restoration period when Amazon S3 is actively processing your current restore request for the object.

If your bucket has a lifecycle configuration with a rule that includes an expiration action, the object expiration overrides the life span that you specify in a restore request. For example, if you restore an object copy for 10 days, but the object is scheduled to expire in 3 days, Amazon S3 deletes the object in 3 days. For more information about lifecycle configuration, see [PutBucketLifecycleConfiguration](#) and [Object Lifecycle Management](#) in *Amazon Simple Storage Service User Guide*.

Requests

Syntax

```
POST /ObjectName?restore&versionId=VersionID HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests \(Amazon Signature Version 4\))
Content-MD5: MD5

request body
```

 **Note**

The syntax shows some of the request headers. For a complete list, see "Request Headers," later in this topic.

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

Name	Description	Required
Content-MD5	The base64-encoded 128-bit MD5 digest of the data. You must use this header as a message integrity check to verify that the request body was not corrupted in transit. For more information, see RFC 1864. Type: String Default: None	Yes

Request Elements

The following is an XML example of a request body for restoring an archive.

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Bulk</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

The following table explains the XML for archive restoration in the request body.

Name	Description	Required
RestoreRequest	Container for restore information. Type: Container	Yes
Days	Lifetime of the restored (active) copy. The minimum number of days that you can restore an object from Amazon Glacier is 1. After the object copy reaches the specified lifetime, Amazon S3 removes it from the bucket. If you are restoring an archive, this element is required. Do not use this element with a SELECT type of request. Type: Positive integer Ancestors: RestoreRequest	Yes, if restoring an archive
GlacierJobParameters	Container for S3 Glacier storage class job parameters. Do not use this element with a SELECT type of request. Type: Container Ancestors: RestoreRequest	No
Tier	The data access tier to use when restoring the archive. Standard is the default. Type: Enum Valid values: Expedited Standard Bulk Ancestors: GlacierJobParameters	No

The following XML is the request body for a select query on an archived object:

```

<RestoreRequest>
  <Type>SELECT</Type>
  <Tier>Expedited</Tier>
  <Description>Job description</Description>
  <SelectParameters>
    <Expression>Select * from Object</Expression>
    <ExpressionType>SQL</ExpressionType>
    <InputSerialization>
      <CSV>
        <FileHeaderInfo>IGNORE</FileHeaderInfo>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <Comments>#</Comments>
      </CSV>
    </InputSerialization>
    <OutputSerialization>
      <CSV>
        <QuoteFields>ASNEEDED</QuoteFields>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
      </CSV>
    </OutputSerialization>
  </SelectParameters>
  <OutputLocation>
    <S3>
      <BucketName>Name of bucket</BucketName>
      <Prefix>Key prefix</Prefix>
      <CannedACL>Canned ACL string</CannedACL>
      <AccessControlList>
        <Grantee>
          <Type>Grantee Type</Type>
        </Grantee>
      </AccessControlList>
      <Encryption>
    </S3>
  </OutputLocation>
  <ID>Grantee identifier</ID>
  <URI>Grantee URI</URI>
  <Permission>Granted permission</Permission>
  <DisplayNmae>Display Name</DisplayName>
  <EmailAddress>email</EmailAddress>
</RestoreRequest>

```

```

    <EncryptionType>Encryption type</EncryptionType>
    <KMSKeyId>KMS Key ID</KMSKeyId>
    <KMSContext>Base64-encoded JSON<KMSContext>
    </Encryption>
  <UserMetadata>
    <MetadataEntry>
      <Name>Key</Name>
      <Value>Value</Value>
    </MetadataEntry>
  </UserMetadata>
  <Tagging>
    <TagSet>
      <Tag>
        <Key>Tag name</Key>
        <Value>Tag value</Value>
      </Tag>
    </TagSet>
  </Tagging>
  <StorageClass>Storage class</StorageClass>
</S3>
  </OutputLocation>
</RestoreRequest>
```

The following tables explain the XML for a SELECT type of restoration in the request body.

Name	Description	Required
RestoreRequest	Container for restore information. Type: Container	Yes
Tier	The data access tier to use when restoring the archive. Standard is the default. Type: Enum Valid values: Expedited Standard Bulk Ancestors: RestoreRequest	No

Name	Description	Required
Description	The optional description for the request. Type: String Ancestors: RestoreRequest	No
SelectParameters	Describes the parameters for the select job request. Type: Container Ancestors: RestoreRequest	Yes, if request type is SELECT
OutputLocation	Describes the location that receives the results of the select restore request. Type: Container for Amazon S3 Ancestors: RestoreRequest	Yes, if request type is SELECT

The **SelectParameters** container element contains the following elements.

Name	Description	Required
Expression	The SQL expression. For example: - The following SQL expression retrieves the first column of the data from the object stored in CSV format: <pre>SELECT s._1 FROM Object s</pre> - The following SQL expression returns everything from the object: <pre>SELECT * FROM Object</pre>	Yes

Name	Description	Required
	Type: String Ancestors: SelectParameters	
ExpressionType	Identifies the expression type. Type: String Valid values: SQL Ancestors: SelectParameters	Yes
InputSerialization	Describes the serialization format of the object. Type: Container for CSV Ancestors: SelectParameters	Yes
OutputSerialization	Describes how the results of the select job are serialized. Type: Container for CSV Ancestors: SelectParameters	Yes

The CSV container element in the **InputSerialization** element contains the following elements.

Name	Description	Required
RecordDelimiter	A single character used to separate individual records in the input. Instead of the default value, you can specify an arbitrary delimiter. Type: String Default: \n	No

Name	Description	Required
	Ancestors: CSV	
FieldDelimiter	A single character used to separate individual fields in a record. You can specify an arbitrary delimiter. Type: String Default: , Ancestors: CSV	No
QuoteCharacter	A single character used for escaping when the field delimiter is part of the value. Consider this example in a CSV file: "a, b" Wrapping the value in quotation marks makes this value a single field. If you don't use the quotation marks, the comma is a field delimiter (which makes it two separate field values, a and b). Type: String Default: " Ancestors: CSV	No

Name	Description	Required
QuoteEscapeCharacter	A single character used for escaping the quotation mark character inside an already escaped value. For example, the value <code>"" a , b ""</code> is parsed as <code>" a , b "</code>. Type: String Default: <code>"</code> Ancestors: CSV	No
FileHeaderInfo	Describes the first line in the input data. It is one of the ENUM values. • NONE: First line is not a header. • IGNORE: First line is a header, but you can't use the header values to indicate the column in an expression. You can use column position (such as <code>_1, _2, ...</code>) to indicate the column (<code>SELECT s._1 FROM OBJECT s</code>). • Use: First line is a header, and you can use the header value to identify a column in an expression (<code>SELECT "name" FROM OBJECT</code>). Type: Enum Valid values: NONE USE IGNORE Ancestors: CSV	No

Name	Description	Required
Comments	A single character used to indicate that a row should be ignored when the character is present at the start of that row. You can specify any character to indicate a comment line. Type: String Ancestors: CSV	No

The **CSV** container element (in the **OutputSerialization** elements) contains the following elements.

Name	Description	Required
QuoteFields	Indicates whether to use quotation marks around output fields. • ALWAYS: Always use quotation marks for output fields. • ASNEEDED: Use quotation marks for output fields when needed. Type: Enum Valid values: ALWAYS ASNEEDED Default: AsNeeded Ancestors: CSV	No
RecordDelimiter	A single character used to separate individual records in the output. Instead of the default value, you can specify an arbitrary delimiter.	No

Name	Description	Required
	Type: String Default: \n Ancestors: CSV	
FieldDelimiter	A single character used to separate individual fields in a record. You can specify an arbitrary delimiter. Type: String Default: , Ancestors: CSV	No
QuoteCharacter	A single character used for escaping when the field delimiter is part of the value. For example, if the value is a , b, Amazon S3 wraps this field value in quotation marks, as follows: " a , b " . Type: String Default: " Ancestors: CSV	No
QuoteEscapeCharacter	A single character used for escaping the quotation mark character inside an already escaped value. For example, if the value is " a , b " , Amazon S3 wraps the value in quotation marks, as follows: "" " a , b "" . Type: String Ancestors: CSV	No

The **S3** container element (in the **OutputLocation** element) contains the following elements.

Name	Description	Required
AccessControlList	A list of grants that control access to the staged results. Type: Container for Grant Ancestors: S3	No
BucketName	The name of the S3 bucket where the select restore results are stored. The bucket must be in the same Amazon Web Services Region as the bucket that contains the input archive object. Type: String Ancestors: S3	Yes
CannedACL	The canned access control list (ACL) to apply to the select restore results. Type: String Valid values: <code>private</code> <code>public-read</code> <code>public-read-write</code> <code>aws-exec-read</code> <code>authenticated-read</code> <code>bucket-owner-read</code> <code>bucket-owner-full-control</code> Ancestors: S3	No
Encryption	Contains encryption information for the stored results. Type: Container for Encryption Ancestors: S3	No
Prefix		Yes

Name	Description	Required
	The prefix that is prepended to the select restore results. The maximum length for the prefix is 512 bytes. Type: String Ancestors: S3	
StorageClass	The class of storage used to store the select request results. Type: String Valid values: STANDARD REDUCED_REDUNDANCY STANDARD_IA ONEZONE_IA Ancestors: S3	No
Tagging	Container for tag information. Type: Tag structure Ancestors: S3	No
UserMetadata	Contains a list of metadata to store with the select restore results. Type: MetadataEntry structure Ancestors: S3	No

The **Grantee** container element (in the **AccessControlList** element) contains the following elements.

Name	Description	Required
DisplayName	The screen name of the grantee.	No

Name	Description	Required
	Type: String Ancestors: Grantee	
EmailAddress	The email address of the grantee. Type: String Ancestors: Grantee	No
ID	The canonical user ID of the grantee. Type: String Ancestors: Grantee	No
Type	The type of the grantee. Type: String Ancestors: Grantee	No
URI	The URI of the grantee group. Type: String Ancestors: Grantee	No
Permission	Granted permission. Type: String Ancestors: Grantee	No

The Encryption container element (in S3) contains the following elements.

Name	Description	Required
EncryptionType	The server-side encryption algorithm used when storing job results. The default is no encryption. Type: String Valid Values <code>aws:kms</code> <code>AES256</code> Ancestors: Encryption	No
KMSContext	Optional. If the encryption type is <code>aws:kms</code>, you can use this value to specify the encryption context for the select restore results. Type: String Ancestors: Encryption	No
KMSKeyId	The Amazon Key Management Service (Amazon KMS) key ID to use for object encryption. Type: String Ancestors: Encryption	No

The TagSet container element (in the Tagging element) contains the following element.

Name	Description	Required
Tag	Contains tags. Type: Container Ancestors: TagSet	No

The **Tag** container element (in the **TagSet** element) contains the following elements.

Name	Description	Required
Key	Name of the tag. Type: String Ancestors: Tag	No
Value	Value of the tag. Type: String Ancestors: Tag	No

The **MetadataEntry** container element (in the **UserMetadata** element) contains the following key-value pair elements to store with an object.

Name	Description	Required
MetadataKey	The metadata key. Type: String Ancestors:	No
MetadataEntry	The metadata value. Type: String Ancestors:	No

Responses

A successful operation returns either the **200 OK** or **202 Accepted** status code.

- If the object copy is not previously restored, then Amazon S3 returns 202 Accepted in the response.
- If the object copy is previously restored, Amazon S3 returns 200 OK in the response.

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see [Common Response Headers](#).

Response Elements

This operation does not return response elements.

Special Errors

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
RestoreAlreadyInProgress	Object restore is already in progress. (This error does not apply to SELECT type requests.)	409 Conflict	Client
GlacierExpeditedRetrievalNotAvailable	S3 Glacier storage class expedited retrievals are currently not available. Try again later. (Returned if there is insufficient capacity to process the Expedited request. This error applies only to Expedited retrievals and not to Standard or Bulk retrievals.)	503	N/A

Examples

Restore an Object for Two Days Using the Expedited Retrieval Option

The following restore request restores a copy of the `photo1.jpg` object from Amazon Glacier for a period of two days using the expedited retrieval option.

```
POST /photo1.jpg?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Mon, 22 Oct 2012 01:49:52 GMT
Authorization: authorization string
Content-Length: content length
```

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Expedited</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

If the `examplebucket` does not have a restored copy of the object, Amazon S3 returns the following `202 Accepted` response.

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZ1zYQvPiBlZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

If a copy of the object is already restored, Amazon S3 returns a `200 OK` response, and updates only the restored copy's expiry time.

Query an Archive with a SELECT Request

The following is an example select restore request.

```
POST /object-one.csv?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Sat, 20 Oct 2012 23:54:05 GMT
Authorization: authorization string
```

Content-Length: *content length*

```
<RestoreRequest xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Type>SELECT</Type>
  <Tier>Expedited</Tier>
  <Description>this is a description</Description>
  <SelectParameters>
    <InputSerialization>
      <CSV>
        <FileHeaderInfo>IGNORE</FileHeaderInfo>
        <Comments>#</Comments>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
      </CSV>
    </InputSerialization>
    <ExpressionType>SQL</ExpressionType>
    <Expression>select * from object</Expression>
    <OutputSerialization>
      <CSV>
        <QuoteFields>ALWAYS</QuoteFields>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>\t</FieldDelimiter>
        <QuoteCharacter>\'</QuoteCharacter>
      </CSV>
    </OutputSerialization>
  </SelectParameters>
  <OutputLocation>
    <S3>
      <BucketName>example-output-bucket</BucketName>
      <Prefix>test-s3</Prefix>
      <AccessControlList>
        <Grant>
          <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail">
            <EmailAddress>jane-doe@example.com</EmailAddress>
          </Grantee>
          <Permission>FULL_CONTROL</Permission>
        </Grant>
      </AccessControlList>
      <UserMetadata>
        <MetadataEntry>
```

```
<Name>test</Name>
<Value>test-value</Value>
</MetadataEntry>
<MetadataEntry>
  <Name>other</Name>
  <Value>something else</Value>
</MetadataEntry>
</UserMetadata>
<StorageClass>STANDARD</StorageClass>
</S3>
</OutputLocation>
</RestoreRequest>
```

Amazon S3 returns the following 202 Accepted response.

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZ1zYQvPiB1ZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
x-amz-restore-output-path: js-test-s3/qE8nk5M0XIj-LuZE2HXNw6empQm3znLkH1MWInRYPS-
0r12W0uj6LyYm-neTvm1-btz3wbBxfMhPykd3jkl-1vZE7w42/
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

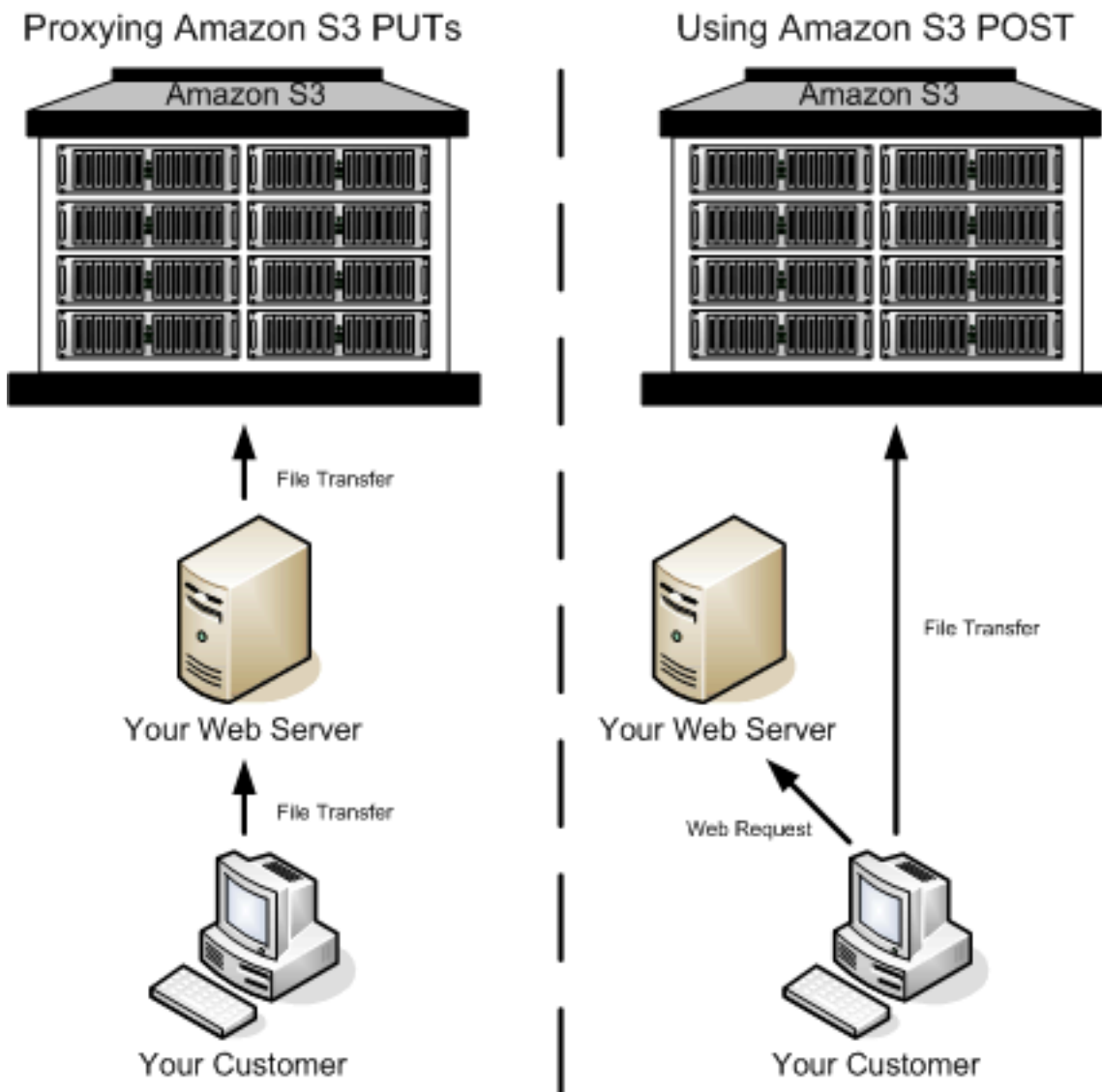
More Info

- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)
- [SQL Reference for Amazon S3 Select and Amazon Glacier Select](#) in the *Amazon Simple Storage Service User Guide*

Browser-Based Uploads Using HTTP POST

Amazon S3 supports HTTP POST requests so that users can upload content directly to Amazon S3. By using POST, end users can authenticate requests without having to pass data through a secure intermediary node that protects your credentials. Thus, HTTP POST has the potential to reduce latency.

The following figure shows an Amazon S3 upload using a POST request.



1. The user accesses your page from a web browser.
2. Your webpage contains an HTML form that contains all the information necessary for the user to upload content to Amazon S3.
3. The user uploads content to Amazon S3 through the web browser.

The process for sending browser-based POST requests is as follows:

1. Create a security policy specifying conditions that restrict what you want to allow in the request, such as the bucket name where objects can be uploaded, and key name prefixes that you want to allow for the object that is being created.

2. Create a signature that is based on the policy. For authenticated requests, the form must include a valid signature and the policy.
3. Create an HTML form that your users can access in order to upload objects to your Amazon S3 bucket.

The following section describes how to create a signature to authenticate a request. For information about creating forms and security policies, see [Creating an HTML Form \(Using Amazon Signature Version 4\)](#).

Calculating a Signature

For authenticated requests, the HTML form must include fields for a security policy and a signature.

- A security policy (see [POST Policy](#)) controls what is allowed in the request.
- The security policy is the StringToSign (see [Introduction to Signing Requests](#)) in your signature calculation.



To Calculate a signature

1. Create a policy using UTF-8 encoding.
2. Convert the UTF-8-encoded policy bytes to base64. The result is the `StringToSign`.
3. Create a signing key.
4. Use the signing key to sign the `StringToSign` using HMAC-SHA256 signing algorithm.

For more information about creating HTML forms, security policies, and an example, see the following:

- [Creating an HTML Form \(Using Amazon Signature Version 4\)](#)
- [POST Policy](#)
- [Example: Browser-Based Upload using HTTP POST \(Using Amazon Signature Version 4\)](#)

Creating an HTML Form (Using Amazon Signature Version 4)

Topics

- [HTML Form Declaration](#)
- [HTML Form Fields](#)

To allow users to upload content to Amazon S3 by using their browsers (HTTP POST requests), you use HTML forms. HTML forms consist of a form declaration and form fields. The form declaration contains high-level information about the request. The form fields contain detailed request information.

This section describes how to create HTML forms. For a working example of browser-based upload using HTTP POST and related signature calculations for request authentication, see [Example: Browser-Based Upload using HTTP POST \(Using Amazon Signature Version 4\)](#).

The form and policy must be UTF-8 encoded. You can apply UTF-8 encoding to the form by specifying `charset=UTF-8` in the content attribute. The following is an example of UTF-8 encoding in the HTML heading.

```
<html>
```

```
<head>
  ...
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  ...
</head>
<body>
```

Following is an example of UTF-8 encoding in a request header.

```
Content-Type: text/html; charset=UTF-8
```

 Note

The form data and boundaries (excluding the contents of the file) cannot exceed 20KB.

HTML Form Declaration

The HTML form declaration has the following three attributes:

- **action** – The URL that processes the request, which must be set to the URL of the bucket. For example, if the name of your bucket is `examplebucket`, the URL is `http://examplebucket.s3.amazonaws.com/`.

 Note

The key name is specified in a form field.

- **method** – The method must be `POST`.
- **enctype** – The enclosure type (`enctype`) must be set to `multipart/form-data` for both file uploads and text area uploads. For more information about `enctype`, see [RFC 1867](#).

This is a form declaration for the bucket `examplebucket`.

```
<form action="http://examplebucket.s3.amazonaws.com/" method="post"

  enctype="multipart/form-data">
```

HTML Form Fields

The following table describes a list of fields that you can use within a form. Among other fields, there is a signature field that you can use to authenticate requests. There are fields for you to specify the signature calculation algorithm (`x-amz-algorithm`), the credential scope (`x-amz-credential`) that you used to generate the signing key, and the date (`x-amz-date`) used to calculate the signature. Amazon S3 uses this information to re-create the signature. If the signatures match, Amazon S3 processes the request.

 Note

The variable `${filename}` is automatically replaced with the name of the file provided by the user and is recognized by all form fields. If the browser or client provides a full or partial path to the file, only the text following the last slash (/) or backslash (\) is used (for example, `C:\Program Files\directory1\file.txt` is interpreted as `file.txt`). If no file or file name is provided, the variable is replaced with an empty string.

If you don't provide elements required for authenticated requests, such as the `policy` element, the request is assumed to be anonymous and will succeed only if you have configured the bucket for public read and write.

Element Name	Description	Required
<code>acl</code>	An Amazon S3 access control list (ACL). If an invalid ACL is specified, Amazon S3 denies the request. For more information about ACLs, see Using Amazon S3 ACLs. Type: String Default: private Valid Values: <code>private</code> <code>public-read</code> <code>public-read-write</code> <code>aws-exec-read</code> <code>authenticated-read</code>	No

Element Name	Description	Required
	bucket-owner-read bucket-owner-full-control	
Cache-Control	REST-specific headers. For more information, see PutObject .	No
Content-Type		
Content-Disposition		
Content-Encoding		
Expires		
key	The key name of the uploaded object. To use the file name provided by the user, use the <code>\${filename}</code> variable. For example, if you upload a file <code>photo1.jpg</code> and you specify <code>/user/user1/\${filename}</code> as key name, the file is stored as <code>/user/user1/photo1.jpg</code>. For more information, see Object Key and Metadata in the <i>Amazon Simple Storage Service User Guide</i>.	Yes
policy	The base64-encoded security policy that describes what is permitted in the request. For authenticated requests, a policy is required. Requests without a security policy are considered anonymous and will succeed only on a publicly writable bucket.	Required for authenticated requests

Element Name	Description	Required
<code>success_action_redirect</code>	The URL to which the client is redirected upon successful upload. If <code>success_action_redirect</code> is not specified, or Amazon S3 cannot interpret the URL, Amazon S3 returns the empty document type that is specified in the <code>success_action_status</code> field. If the upload fails, Amazon S3 returns an error and does not redirect the user to another URL.	No

Element Name	Description	Required
success_action_status	The status code returned to the client upon successful upload if success_action_redirect is not specified. Valid values are 200, 201, or 204 (default). If the value is set to 200 or 204, Amazon S3 returns an empty document with the specified status code. If the value is set to 201, Amazon S3 returns an XML document with a 201 status code. For information about the content of the XML document, see POST Object. If the value is not set or is invalid, Amazon S3 returns an empty document with a 204 status code.  Note Some versions of the Adobe Flash player do not properly handle HTTP responses with an empty body. To support uploads through Adobe Flash, we recommend setting success_action_status to 201.	No

Element Name	Description	Required
x-amz-algorithm	The signing algorithm used to authenticate the request. For Amazon Signature Version 4, the value is <code>AWS4-HMAC-SHA256</code> . This field is required if a policy document is included with the request.	Required for authenticated requests
x-amz-credential	In addition to your access key ID, this field also provides scope information identifying region and service for which the signature is valid. This should be the same scope you used in calculating the signing key for signature calculation. It is a string of the following form: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> For example: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> For Amazon S3, the <i>aws-service</i> string is <code>s3</code>. For a list of Amazon S3 <i>aws-region</i> strings, see Regions and Endpoints in the <i>Amazon Web Services General Reference</i>. This is required if a policy document is included with the request.	Required for authenticated requests

Element Name	Description	Required
x-amz-date	It is the date value in ISO8601 format. For example, 20130728T000000Z . It is the same date you used in creating the signing key (for example, 20130728). This must also be the same value you provide in the policy (x-amz-date) that you signed. This is required if a policy document is included with the request.	Required for authenticated requests
x-amz-security-token	A security token used by Amazon DevPay and session credentials If the request is using Amazon DevPay, it requires two x-amz-security-token form fields: one for the product token and one for the user token. For more information, see Using DevPay in the <i>Amazon Simple Storage Service User Guide</i>. If the request is using session credentials, it requires one x-amz-security-token form. For more information, see Requesting Temporary Security Credentials in the <i>IAM User Guide</i>.	No
x-amz-signature	(Amazon Signature Version 4) The HMAC-SHA256 hash of the security policy. This field is required if a policy document is included with the request.	Required for authenticated requests

Element Name	Description	Required
x-amz-meta-*	Field names starting with this prefix are user-defined metadata. Each one is stored and returned as a set of key-value pairs. Amazon S3 doesn't validate or interpret user-defined metadata. For more information, see PutObject .	No
x-amz-*	See POST Object (POST Object for other x-amz-* headers.	No
file	File or text content. The file or content must be the last field in the form. You cannot upload more than one file at a time.	Yes

Conditional items are required for authenticated requests and are optional for anonymous requests.

Now that you know how to create forms, next you can create a security policy that you can sign. For more information, see [POST Policy](#).

POST Policy

Topics

- [Expiration](#)
- [Condition Matching](#)
- [Conditions](#)
- [Character Escaping](#)

The policy required for making authenticated requests using HTTP POST is a UTF-8 and base64-encoded document written in JavaScript Object Notation (JSON) that specifies conditions that the request must meet. Depending on how you design your policy document, you can control the access granularity per-upload, per-user, for all uploads, or according to other designs that meet your needs.

This section describes the POST policy. For example signature calculations using POST policy, see [Example: Browser-Based Upload using HTTP POST \(Using Amazon Signature Version 4\)](#).

Note

Although the policy document is optional, we highly recommend that you use one in order to control what is allowed in the request. If you make the bucket publicly writable, you have no control at all over which users can write to your bucket.

The following is an example of a POST policy document.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

The POST policy always contains the `expiration` and `conditions` elements. The example policy uses two condition matching types (exact matching and starts-with matching). The following sections describe these elements.

Expiration

The `expiration` element specifies the expiration date and time of the POST policy in ISO8601 GMT date format. For example, `2013-08-01T12:00:00.000Z` specifies that the POST policy is not valid after midnight GMT on August 1, 2013.

Condition Matching

Following is a table that describes condition matching types that you can use to specify POST policy conditions (described in the next section). Although you must specify at least one condition

for each form field that you specify in the form, you can create more complex matching criteria by specifying multiple conditions for a form field.

Condition Match Type	Description
Exact Matches	The form field value must match the value specified. This example indicates that the ACL must be set to public-read: <pre>{"acl": "public-read" }</pre> This example is an alternate way to indicate that the ACL must be set to public-read: <pre>["eq", "\$acl", "public-read"]</pre>
Starts With	The value must start with the specified value. This example indicates that the object key must start with user/user1: <pre>["starts-with", "\$key", "user/user1/"]</pre>
Matching Content-Types in a Comma-Separated List	Content-Types values for a starts-with condition that include commas are interpreted as lists. Each value in the list must meet the condition for the whole condition to pass. For example,given the following condition: <pre>["starts-with", "\$Content-Type", "image/"]</pre> The following value would pass the condition: <pre>"image/jpg,image/png,image/gif"</pre> The following value would not pass the condition: <pre>["image/jpg,text/plain"]</pre>

Condition Match Type	Description
	Note Data elements other than Content-Type are treated as strings, regardless of the presence of commas.
Matching Any Content	To configure the POST policy to allow any content within a form field, use starts-with with an empty value (""). This example allows any value for success_action_redirect : ["starts-with", "\$success_action_redirect", ""]
Specifying Ranges	For form fields that accept a range, separate the upper and lower limit with a comma. This example allows a file size from 1 to 10 MiB: ["content-length-range", 1048576, 10485760]

The specific conditions supported in a POST policy are described in [Conditions](#).

Conditions

The conditions in a POST policy is an array of objects, each of which is used to validate the request. You can use these conditions to restrict what is allowed in the request. For example, the preceding policy conditions require the following:

- Request must specify the johnsmith bucket name.
- Object key name must have the user/eric prefix.
- Object ACL must be set to public-read.

Each form field that you specify in a form (except x-amz-signature, file, policy, and field names that have an x-ignore- prefix) must appear in the list of conditions.

 Note

All variables within the form are expanded prior to validating the POST policy. Therefore, all condition matching should be against the expanded form fields. Suppose that you want to restrict your object key name to a specific prefix (user/user1). In this case, you set the key form field to user/user1/\${filename}. Your POST policy should be ["starts-with", "\$key", "user/user1/"] (do not enter ["starts-with", "\$key", "user/user1/\${filename}"]). For more information, see [Condition Matching](#).

Policy document conditions are described in the following table.

Element Name	Description
acl	Specifies the ACL value that must be used in the form submission. This condition supports exact matching and starts-with condition match type discussed in the following section.
bucket	Specifies the acceptable bucket name. This condition supports exact matching condition match type.
content-length-range	The minimum and maximum allowable size for the uploaded content. This condition supports content-length-range condition match type.
Cache-Control	REST-specific headers. For more information, see POST Object .
Content-Type	
Content-Disposition	

Element Name	Description
Content-Encoding	This condition supports exact matching and <code>starts-with</code> condition match type.
Expires	
key	The acceptable key name or a prefix of the uploaded object. This condition supports exact matching and <code>starts-with</code> condition match type.
success_action_redirect	The URL to which the client is redirected upon successful upload.
redirect	This condition supports exact matching and <code>starts-with</code> condition match type.
success_action_status	The status code returned to the client upon successful upload if <code>success_action_redirect</code> is not specified. This condition supports exact matching.
x-amz-algorithm	The signing algorithm that must be used during signature calculation. For Amazon Signature Version 4, the value is <code>AWS4-HMAC-SHA256</code> . This condition supports exact matching.

Element Name	Description
x-amz-credential	The credentials that you used to calculate the signature. It provides access key ID and scope information identifying region and service for which the signature is valid. This should be the same scope you used in calculating the signing key for signature calculation. It is a string of the following form: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> For example: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> For Amazon S3, the aws-service string is s3. For a list of Amazon S3 aws-region strings, see Regions and Endpoints in the <i>Amazon Web Services General Reference</i>. This is required if a POST policy document is included with the request. This condition supports exact matching.
x-amz-date	The date value specified in the ISO8601 formatted string. For example, 20130728T000000Z . The date must be same that you used in creating the signing key for signature calculation. This is required if a POST policy document is included with the request. This condition supports exact matching.

Element Name	Description
<code>x-amz-security-token</code>	Amazon DevPay security token. Each request that uses Amazon DevPay requires two <code>x-amz-security-token</code> form fields: one for the product token and one for the user token. As a result, the values must be separated by commas. For example, if the user token is <code>eW91dHVhZQ==</code> and the product token is <code>b0hnNVNKWVJIQTA=</code>, you set the POST policy entry to: <code>{ "x-amz-security-token": "eW91dHVhZQ==,b0hnNVNKWVJIQTA=" }</code>. For more information about Amazon DevPay, see Using DevPay in the <i>Amazon Simple Storage Service User Guide</i>.
<code>x-amz-meta-*</code>	User-specified metadata. This condition supports exact matching and <code>starts-with</code> condition match type.
<code>x-amz-*</code>	See POST Object (POST Object for other <code>x-amz-*</code> headers. This condition supports exact matching.

Note

If your toolkit adds more form fields (for example, Flash adds `filename`), you must add them to the POST policy document. If you can control this functionality, prefix `x-ignore-` to the field so Amazon S3 ignores the feature and it won't affect future versions of this feature.

Character Escaping

Characters that must be escaped within a POST policy document are described in the following table.

Escape Sequence	Description
\\	Backslash
\\$	Dollar symbol
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\uXXXX	All Unicode characters

Now that you are acquainted with forms and policies, and understand how signing works, you can try a POST upload example. You need to write the code to calculate the signature. The example provides a sample form, and a POST policy that you can use to test your signature calculations. For more information, see [Example: Browser-Based Upload using HTTP POST \(Using Amazon Signature Version 4\)](#).

Example: Browser-Based Upload using HTTP POST (Using Amazon Signature Version 4)

This section shows an example of using an HTTP POST request to upload content directly to Amazon S3.

For more information on Signature Version 4, see [Signature Version 4 Signing Process](#).

Uploading a File to Amazon S3 Using HTTP POST

This example provides a sample POST policy and a form that you can use to upload a file. The topic uses the example policy and fictitious credentials to show you the workflow and resulting signature and policy hash. You can use this data as test suite to verify your signature calculation code.

The example uses the following example credentials the signature calculations. You can use these credentials to verify your signature calculation code. However, you must then replace these with your own credentials when sending requests to Amazon.

Parameter	Value
AWSSecretAccessKey	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

Sample Policy and Form

The following POST policy supports uploads to Amazon S3 with specific conditions.

```
{ "expiration": "2015-12-30T12:00:00.000Z",
  "conditions": [
    {"bucket": "sigv4examplebucket"},
    ["starts-with", "$key", "user/user1/"],
    {"acl": "public-read"},
    {"success_action_redirect": "http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"}]
```

```
{
  "x-amz-server-side-encryption": "AES256"},
  ["starts-with", "$x-amz-meta-tag", ""],

  {"x-amz-credential": "AKIAIOSFODNN7EXAMPLE/20151229/us-east-1/s3/aws4_request"},
  {"x-amz-algorithm": "AWS4-HMAC-SHA256"},
  {"x-amz-date": "20151229T000000Z" }
]
}
```

This POST policy sets the following conditions on the request:

- The upload must occur before noon UTC on December 30, 2015.
- The content can be uploaded only to the `sigv4examplebucket`. The bucket must be in the region that you specified in the credential scope (`x-amz-credential` form parameter), because the signature you provided is valid only within this scope.
- You can provide any key name that starts with `user/user1`. For example, `user/user1/MyPhoto.jpg`.
- The ACL must be set to `public-read`.
- If the upload succeeds, the user's browser is redirected to `http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html`.
- The object must be an image file.
- The `x-amz-meta-uuid` tag must be set to `14365123651274`.
- The `x-amz-meta-tag` can contain any value.

The following is a Base64-encoded version of this POST policy. You use this value as your `StringToSign` in signature calculation.

```
eyJhZiZlZGhwaXJhdGlvbiI6IClIyMDE1LTEyLTMwVDEyOjAwOjAwLjAwMFoiLA0KICAiY29uZGl0aW9ucyI6IFsNCiAgICB7ImJ
```

When you copy/paste the preceding policy, it should have carriage returns and new lines for your computed hash to match this value (ie. ASCII text, with CRLF line terminators).

Using example credentials to create a signature, the signature value is as follows (in signature calculation, the date is same as the `x-amz-date` in the policy (20151229):

```
8afdbf4008c03f22c2cd3cdb72e4afbb1f6a588f3255ac628749a66d7f09699e
```

The following example form specifies the preceding POST policy and supports a POST request to the `sigv4examplebucket`. Copy/paste the content in a text editor and save it as `exampleform.html`. You can then upload image files to the specific bucket using the `exampleform.html`. Your request will succeed if the signature you provide matches the signature Amazon S3 calculates.

Note

You must update the bucket name, dates, credential, policy, and signature with valid values for this to successfully upload to S3.

```
<html>
  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

  </head>
  <body>

    <form action="http://sigv4examplebucket.s3.amazonaws.com/" method="post"
    enctype="multipart/form-data">
      Key to upload:
      <input type="input" name="key" value="user/user1/${filename}" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect" value="http://
sigv4examplebucket.s3.amazonaws.com/successful_upload.html" />
      Content-Type:
      <input type="input" name="Content-Type" value="image/jpeg" /><br />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      <input type="hidden" name="x-amz-server-side-encryption" value="AES256" />
      <input type="text" name="X-Amz-Credential" value="AKIAIOSFODNN7EXAMPLE/20151229/
us-east-1/s3/aws4_request" />
      <input type="text" name="X-Amz-Algorithm" value="AWS4-HMAC-SHA256" />
      <input type="text" name="X-Amz-Date" value="20151229T000000Z" />

      Tags for File:
      <input type="input" name="x-amz-meta-tag" value="" /><br />
      <input type="hidden" name="Policy" value='<Base64-encoded policy string>' />
      <input type="hidden" name="X-Amz-Signature" value="<signature-value>" />
      File:
      <input type="file" name="file" /> <br />
```

```
<!-- The elements after this will be ignored -->
<input type="submit" name="submit" value="Upload to Amazon S3" />
</form>

</html>
```

The post parameters are case insensitive. For example, you can specify `x-amz-signature` or `X-Amz-Signature`.

Browser-based uploads to Amazon S3 using the Amazon Amplify library

This section describes how to upload files to Amazon S3 using the Amazon Amplify JavaScript library.

For information about setting up the Amazon Amplify library, see [Amazon Amplify Installation and Configuration](#).

Using the Amazon Amplify JavaScript library to Upload Files to Amazon S3

The Amazon Amplify library Storage module gives a simple browser-based upload mechanism for managing user content in public or private Amazon S3 storage.

Example: Amazon Amplify Manual Setup

The following example shows the manual setup for using the Amazon Amplify Storage module. The default implementation of the Storage module uses Amazon S3.

```
import Amplify from 'aws-amplify';
Amplify.configure(
  Auth: {
    identityPoolId: 'XX-XXXX-X:XXXXXXXXX-XXXX-1234-abcd-1234567890ab', //REQUIRED -
    Amazon Cognito Identity Pool ID
    region: 'XX-XXXX-X', // REQUIRED - Amazon Cognito Region
    userPoolId: 'XX-XXXX-X_abcd1234', //OPTIONAL - Amazon Cognito User Pool ID
    userPoolWebClientId: 'XX-XXXX-X_abcd1234', //OPTIONAL - Amazon Cognito Web
    Client ID
  },
  Storage: {
```

```
    bucket: '', //REQUIRED - Amazon S3 bucket
    region: 'XX-XXXX-X', //OPTIONAL - Amazon service region
  }
};
```

Example: Put data into Amazon S3

The following example shows how to put public data into Amazon S3.

```
Storage.put('test.txt', 'Hello')
  .then (result => console.log(result))
  .catch(err => console.log(err));
```

The following example shows how to put private data into Amazon S3.

```
Storage.put('test.txt', 'Private Content', {
  level: 'private',
  contentType: 'text/plain'
})
  .then (result => console.log(result))
  .catch(err => console.log(err));
```

For more information about using the Amazon Amplify Storage module, see [Amazon Amplify Storage](#).

More Info

[Amazon Amplify Quick Start](#)

Common request headers

The following table describes headers that can be used by various types of Amazon S3 REST requests.

Header Name	Description
Authorization	The information required for request authentication. For more information, go to The Authentication Header in the <i>Amazon Simple Storage Service Developer Guide</i> . For anonymous requests this header is not required.
Access-Control-Request-Method	A list of HTTP methods that is sent as a pre-flight CORS request. If the pre-flight CORS evaluation is successful, then the specified methods are allowed to be used in the following CORS request.
Content-Length	Length of the message (without the headers) according to RFC 2616. This header is required for PUTs and operations that load XML, such as logging and ACLs.
Content-Type	The content type of the resource in case the request has content in the body. Example: text/plain
Content-MD5	The base64 encoded 128-bit MD5 digest of the message (without the headers) according to RFC 1864. This header can be used as a message integrity check to verify that the data is the same data that was originally sent. Although it is optional, we recommend using the Content-MD5 mechanism as an end-to-end integrity check. For more information about REST request authentication, go to REST Authentication in the <i>Amazon Simple Storage Service Developer Guide</i> .
Date	The date that can be used to create the signature contained in the Authorization header. If the Date header is to be used for signing it must be specified in the ISO 8601 basic format. In this case, the x-amz-date header is not needed.

Header Name	Description
	Note that when <code>x-amz-date</code> is present, it always overrides the value of the Date header. If the Date header is not used for signing, it can be one of the full date formats specified by RFC 2616, section 3.3. For example, the date/time <code>Wed, 01 Mar 2006 12:00:00 GMT</code> is a valid date/time header for use with Amazon S3. If you are using the Date header for signing, then it must be in the ISO 8601 basic <code>YYYYMMDD'T'HHMMSS'Z'</code> format. If Date is specified but is not in ISO 8601 basic format, then you must also include the <code>x-amz-date</code> header. If Date is specified in ISO 8601 basic format, then this is sufficient for signing requests and you do not need the <code>x-amz-date</code> header. For more information, see Handling Dates in Signature Version 4 in the <i>Amazon Web Services Glossary</i>.
Expect	When your application uses 100-continue, it does not send the request body until it receives an acknowledgment. If the message is rejected based on the headers, the body of the message is not sent. This header can be used only if you are sending a body. Valid Values: 100-continue
Host	For path-style requests, the value is <code>s3.amazonaws.com</code>. For virtual-style requests, the value is <code>BucketName.s3.amazonaws.com</code>. For more information, go to Virtual Hosting in the <i>Amazon Simple Storage Service User Guide</i>. This header is required for HTTP 1.1 (most toolkits add this header automatically); optional for HTTP/1.0 requests.
Origin	An endpoint that specifies the server name of the initial requester.

Header Name	Description
x-amz-content-sha256	When using signature version 4 to authenticate request, this header provides a hash of the request payload. For more information see Signature Calculations for the Authorization Header: Transferring Payload in a Single Chunk (Amazon Signature Version 4). When uploading object in chunks, you set the value to <code>STREAMING-AWS4-HMAC-SHA256-PAYLOAD</code> to indicate that the signature covers only headers and that there is no payload. For more information, see Signature Calculations for the Authorization Header: Transferring Payload in Multiple Chunks (Chunked Upload) (Amazon Signature Version 4).
x-amz-date	The date used to create the signature in the Authorization header. The format must be ISO 8601 basic in the <code>YYYYMMDD'T'HHMMSS'Z'</code> format. For example, the date/time <code>20170210T120000Z</code> is a valid <code>x-amz-date</code> for use with Amazon S3. <code>x-amz-date</code> is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, then <code>x-amz-date</code> is not needed. When <code>x-amz-date</code> is present, it always overrides the value of the Date header. For more information, see Handling Dates in Signature Version 4 in the <i>Amazon Web Services Glossary</i>.

Header Name	Description
x-amz-security-token	This header can be used in the following scenarios: • To provide security tokens for Amazon DevPay operations - Each request that uses Amazon DevPay requires two x-amz-security-token headers: one for the product token and one for the user token. When Amazon S3 receives an authenticated request, it compares the computed signature with the provided signature. Improperly formatted multi-value headers that are used to calculate a signature can cause authentication issues. • To provide a security token when using temporary security credentials - When making requests using temporary security credentials that you obtained from IAM, you must provide a security token by using this header. To learn more about temporary security credentials, see Making Requests. This header is required for requests that use Amazon DevPay and requests that are signed by using temporary security credentials.

Common response headers

The following table describes response headers that are common to most Amazon S3 responses.

Name	Description
Access-Control-Allow-Credentials	A Boolean that determines if the server allows CORS requests to contain credentials. If the <code>Access-Control-Allow-Origin</code> request header is set to <code>'*'</code> then the <code>Access-Control-Allow-Credentials</code> response header will be omitted, else it is set to <code>true</code> when CORS evaluation is successful. Type: Boolean Default: None
Access-Control-Allow-Headers	A list of HTTP headers allowed for your CORS requests. The <code>Access-Control-Allow-Headers</code> response header is returned for successful CORS evaluations and explicitly specifies all allowed <code>Access-Control-Request-Headers</code>. Type: String Default: None
Access-Control-Allow-Methods	A list that specifies which HTTP methods are allowed. Amazon S3 will only allow CORS requests from allowed CORS methods when the CORS evaluation is successful. Type: String Default: None
Access-Control-Allow-Origin	The location of the allowed origin. Amazon S3 will only send the <code>Access-Control-Allow-Origin</code> response header when the CORS evaluation is successful. If the request origin matches <code>'*'</code> in the CORS configuration's allowed origins then the <code>'*'</code> is returned in this response header instead of the original origin.

Name	Description
	Type: String Default: None
Access-Control-Expose-Headers	A list that allows a server to identify a response header that exposes access for applications when the CORS evaluation is successful. Type: String Default: None
Access-Control-Max-Age	The time in seconds that your browser can cache the response for a CORS pre-flight request as identified by the resource, the HTTP method, and the origin. The Access-Control-Max-Age response header is only returned when the CORS evaluation is successful. Type: Integer Default: None
Vary	A list that indicates which request headers the CORS evaluation result varies on. The Vary response header is only returned when the CORS evaluation is successful. Type: String Default: None
Content-Length	The length in bytes of the body in the response. Type: String Default: None
Content-Type	The MIME type of the content. For example, Content-Type: text/html; charset=utf-8 . Type: String Default: None

Name	Description
Connection	A value that specifies whether the connection to the server is open or closed. Type: Enum Valid Values: open close Default: None
Date	The date and time that Amazon S3 responded; for example, Wed, 01 Mar 2006 12:00:00 GMT. Type: String Default: None
ETag	The entity tag (ETag) represents a specific version of the object. The ETag reflects changes only to the contents of an object, not its metadata. The ETag might or might not be an MD5 digest of the object data. Whether or not it is depends on how the object was created and how it is encrypted, as follows: • Objects created through the Amazon Web Services Management Console or by the PUT Object, POST Object, or Copy operation: • Objects that are plaintext or encrypted by server-side encryption with Amazon S3 managed keys (SSE-S3) have ETags that are an MD5 digest of their data. • Objects encrypted by server-side encryption with customer-provided keys (SSE-C) or Amazon Key Management Service (Amazon KMS) keys (SSE-KMS) have ETags that are not an MD5 digest of their object data. • Objects created by either the Multipart Upload or Upload Part Copy operation have ETags that are not MD5 digests, regardless of the method of encryption. Type: String

Name	Description
Server	The name of the server that created the response. Type: String Default: AmazonS3
x-amz-delete-marker	A value that specifies whether the object returned was (true) or was not (false) a delete marker. Type: Boolean Valid Values: true false Default: false
x-amz-id-2	A special token that is used together with the x-amz-request-id header to help Amazon troubleshoot problems. For information about Amazon Web Services Support using these request IDs, see Troubleshooting Amazon S3. Type: String Default: None
x-amz-request-id	A value created by Amazon S3 that uniquely identifies the request. This value is used together with the x-amz-id-2 header to help Amazon troubleshoot problems. For information about Amazon Web Services Support using these request IDs, see Troubleshooting Amazon S3. Type: String Default: None
x-amz-server-side-encryption	The server-side encryption algorithm used when storing this object in Amazon S3 (for example, AES256, aws:kms). Valid Values: AES256 aws:kms

Name	Description
x-amz-version-id	The version of the object. When you enable versioning, Amazon S3 generates a random number for objects added to a bucket. The value is UTF-8 encoded and URL ready. When you PUT an object in a bucket where versioning has been suspended, the version ID is always null. Type: String Valid Values: null any URL-ready, UTF-8 encoded string Default: null

Error responses

This section provides reference information about Amazon S3 errors.

Note

- In general, S3 bucket owners are billed for requests with HTTP 200 OK successful responses and HTTP 4XX client error responses. Bucket owners aren't billed for HTTP 5XX server error responses, such as HTTP 503 Slow Down errors. For more information on S3 error codes under HTTP 3XX and 4XX status codes that aren't billed, see [Billing for Amazon S3 error responses](#) in the Amazon S3 User Guide. For more information about billing charges if your bucket is configured as a Requester Pays bucket, see [How Requester Pays charges work](#) in the Amazon S3 User Guide.
- SOAP support over HTTP is deprecated, but SOAP is still available over HTTPS. New Amazon S3 features are not supported for SOAP. Instead of using SOAP, we recommend that you use either the REST API or the Amazon SDKs.

Topics

- [REST error responses](#)
- [List of error codes](#)
- [List of SELECT Object Content Error Codes](#)
- [List of Replication-related error codes](#)
- [List of Tagging-related error codes](#)
- [List of Amazon S3 on Outposts error codes](#)
- [List of Amazon S3 Storage Lens error codes](#)
- [List of Amazon S3 Object Lambda error codes](#)
- [List of Amazon S3 asynchronous error codes](#)
- [List of Amazon S3 Access Grants error codes](#)
- [List of Amazon FSx-related error codes](#)
- [List of Amazon S3 Tables error codes](#)
- [Amazon S3 error best practices](#)

REST error responses

When an error occurs, the header information contains the following:

- Content-Type: application/xml
- An appropriate 3xx, 4xx, or 5xx HTTP status code

The body of the response also contains information about the error. The following sample error response shows the structure of response elements common to all REST error responses.

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/mybucket/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

The following table explains the REST error response elements.

Name	Description
Code	The error code is a string that uniquely identifies an error condition. It is meant to be read and understood by programs that detect and handle errors by type. For more information, see List of error codes. Type: String Ancestor: Error
Error	Container for all error elements. Type: Container Ancestor: None
Message	The error message contains a generic description of the error condition in English. It is intended for a human audience. Simple programs display the message directly to the end user if they encounter an error condition they

Name	Description
	don't know how or don't care to handle. Sophisticated programs with more exhaustive error handling and proper internationalization are more likely to ignore the error message. Type: String Ancestor: Error
RequestId	ID of the request associated with the error. Type: String Ancestor: Error
Resource	The bucket or object that is involved in the error. Type: String Ancestor: Error

Many error responses contain additional structured data meant to be read and understood by a developer diagnosing programming errors. For example, if you send a Content-MD5 header with a REST PUT request that doesn't match the digest calculated on the server, you receive a `BadDigest` error. The error response also includes as detail elements the digest that the server calculated, and the digest that you told the server to expect. During development, you can use this information to diagnose the error. In production, a well-behaved program might include this information in its error log.

For information about general response elements, go to [Error responses](#).

List of error codes

The following table lists Amazon S3 error codes.

Error code	Description	HTTP status code	SOAP fault code prefix
AccessControlListNotSupported	The bucket does not allow ACLs.	400 Bad Request	Client
AccessDenied	Access Denied	403 Forbidden	Client
AccessPointAlreadyOwnedByYou	An access point with an identical name already exists in your account.	409 Conflict	Client
AccountProblem	There is a problem with your Amazon Web Services account that prevents the operation from completing successfully. For further assistance, see Contact Us .	403 Forbidden	Client
AllAccessDisabled	All access to this Amazon S3 resource has been disabled. For further assistance, see Contact Us .	403 Forbidden	Client
AmbiguousGrantByEmailAddress	The email address that you provided is associated with more than one account.	400 Bad Request	Client
AuthorizationHeaderMalformed	The authorization header that you provided is not valid.	400 Bad Request	N/A
AuthorizationQueryParametersError	The authorization query parameters that you provided are not valid.	400 Bad Request	N/A

Error code	Description	HTTP status code	SOAP fault code prefix
BadDigest	The Content-MD5 or checksum value that you specified did not match what the server received.	400 Bad Request	Client
BucketAlreadyExists	The requested bucket name is not available. The bucket namespace is shared by all users of the system. Specify a different name and try again.	409 Conflict	Client
BucketAlreadyOwnedByYou	The bucket that you tried to create already exists, and you own it. Amazon S3 returns this error in all Amazon Web Services Regions except in the US East (N. Virginia) Region (us-east-1). For legacy compatibility, if you re-create an existing bucket that you already own in us-east-1, Amazon S3 returns 200 OK and resets the bucket access control lists (ACLs). For Amazon S3 on Outposts, the bucket that you tried to create already exists in your Outpost and you own it.	409 Conflict (in all Regions except us-east-1)	Client
BucketHasAccessPointsAttached	The bucket you tried to delete has access points attached. Delete your access points before deleting your bucket.	400 Bad Request	Client

Error code	Description	HTTP status code	SOAP fault code prefix
BucketNotEmpty	The bucket that you tried to delete is not empty.	409 Conflict	Client
ClientTokenConflict	Your Multi-Region Access Point idempotency token was already used for a different request.	409 Conflict	Client
ConnectionClosedByRequester	Returned to the original caller when an error is encountered while reading the WriteGetObjectResponse body.	400 Bad Request	Client
ConditionalRequestConflict	A conflicting operation occurred. If using PutObject you can retry the request. If using multipart upload you should initiate another CreateMultipartUpload request and re-upload each part.	409 Conflict	Client
CredentialsNotSupported	This request does not support credentials.	400 Bad Request	Client
CrossLocationLoggingProhibited	Cross-Region logging is not allowed. Buckets in one Amazon Web Services Region cannot log information to a bucket in another Region.	403 Forbidden	Client
DeviceNotActiveError	The device is not currently active.	400 Bad Request	Client

Error code	Description	HTTP status code	SOAP fault code prefix
EndpointNotFound	Direct requests to the correct endpoint.	400 Bad Request	Client
EntityTooSmall	Your proposed upload is smaller than the minimum allowed object size.	400 Bad Request	Client
EntityTooLarge	Your proposed upload exceeds the maximum allowed object size. For more information, see Amazon Simple Storage Service endpoints and quotas in the <i>Amazon Web Services General Reference</i> .	400 Bad Request	Client
	Your proposed download exceeds the maximum allowed size.	405 Method Not Allowed	Client
ExpiredToken	The provided token has expired.	400 Bad Request	Client

Error code	Description	HTTP status code	SOAP fault code prefix
<code>IllegalLocationConstraintException</code>	This error might occur for the following reasons: - You are trying to access a bucket from a different Region than where the bucket exists. - You attempt to create a bucket with a location constraint that corresponds to a different region than the regional endpoint the request was sent to.	400 Bad Request	Client
<code>IllegalVersioningConfigurationException</code>	The versioning configuration specified in the request is not valid.	400 Bad Request	Client
<code>IncompleteBody</code>	You did not provide the number of bytes specified by the Content-Length HTTP header.	400 Bad Request	Client
<code>IncorrectEndpoint</code>	The specified bucket exists in another Region. Direct requests to the correct endpoint.	400 Bad Request	Client
<code>IncorrectNumberOfFilesInPostRequest</code>	POST requires exactly one file upload per request.	400 Bad Request	Client
<code>InlineDataTooLarge</code>	The inline data exceeds the maximum allowed size.	400 Bad Request	Client

Error code	Description	HTTP status code	SOAP fault code prefix
InternalError	An internal error occurred. Try again.	500 Internal Server Error	Server
InvalidAccessKeyId	The Amazon access key ID that you provided does not exist in our records.	403 Forbidden	Client
InvalidAccessPoint	The specified access point name or account is not valid.	400 Bad Request	Client
InvalidAccessPointAliasError	The specified access point alias name is not valid.	400 Bad Request	Client
InvalidAddressingHeader	You must specify the Anonymous role.	N/A	Client

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidArgument	This error might occur for the following reasons: • A ListBuckets request is made to a Regional endpoint that is different from the Region specified in the bucket-region parameter. • The specified argument was not valid. • The request was missing a required header. • The specified argument was incomplete or in the wrong format. • The specified argument must have a length greater than or equal to 3.	400 Bad Request	Client
InvalidBucketAclWithObjectOwnership	Bucket cannot have ACLs set with ObjectOwnership's BucketOwner Enforced setting.	400 Bad Request	Client
InvalidBucketName	The specified bucket is not valid.	400 Bad Request	Client

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidBucketOwner AWSAccountID	The value of the expected bucket owner parameter must be an Amazon Web Services account ID.	400 Bad Request	Client
InvalidBucketState	The request is not valid for the current state of the bucket.	409 Conflict	Client
InvalidDigest	The Content-MD5 or checksum value that you specified is not valid.	400 Bad Request	Client
InvalidEncryptionAlgorithmError	The encryption request that you specified is not valid. The valid value is AES256.	400 Bad Request	Client
InvalidHostHeader	The host headers provided in the request used the incorrect style addressing.	400 Bad Request	Client
InvalidHttpMethod	The request is made using an unexpected HTTP method.	400 Bad Request	Client
InvalidLocationConstraint	The specified location (Region) constraint is not valid. For more information about selecting a Region for your buckets, see Buckets overview .	400 Bad Request	Client
InvalidObjectState	The operation is not valid for the current state of the object.	403 Forbidden	Client

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidPart	One or more of the specified parts could not be found. The part might not have been uploaded, or the specified entity tag might not have matched the part's entity tag.	400 Bad Request	Client
InvalidPartOrder	The list of parts was not in ascending order. The parts list must be specified in order by part number.	400 Bad Request	Client
InvalidPayer	All access to this object has been disabled. For further assistance, see Contact Us .	403 Forbidden	Client
InvalidPolicyDocument	The content of the form does not meet the conditions specified in the policy document.	400 Bad Request	Client
InvalidRange	The requested range cannot be satisfied.	416 Requested Range Not Satisfiable	Client
InvalidRegion	You've attempted to create a Multi-Region Access Point in a Region that you haven't opted in to.	403 Forbidden	Client

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidRequest	This error might occur for the following reasons: • An unpaginated ListBuckets request is made from an account that has an approved general purpose bucket quota higher than 10,000. You must make paginated requests to list the buckets in an account with more than 10,000 buckets. • The request is using the wrong signature version. Use AWS4-HMAC-SHA256 (Signature Version 4). • An access point can be created only for an existing bucket. • The access point is not in a state where it can be deleted. • An access point can be listed only for an existing bucket. • The next token is not valid. • At least one action must be specified in a lifecycle rule. •	400 Bad Request	Client

Error code	Description	HTTP status code	SOAP fault code prefix
	At least one lifecycle rule must be specified. • The number of lifecycle rules must not exceed the allowed limit of 1000 rules. • The range for the MaxResults parameter is not valid. • SOAP requests must be made over an HTTPS connection. • Amazon S3 Transfer Acceleration is not supported for buckets with non-DNS compliant names. • Amazon S3 Transfer Acceleration is not supported for buckets with periods (.) in their names. • The Amazon S3 Transfer Acceleration endpoint supports only virtual style requests. • Amazon S3 Transfer Acceleration is not configured on this bucket. • Amazon S3 Transfer Acceleration is disabled on this bucket. •		

Error code	Description	HTTP status code	SOAP fault code prefix
	Amazon S3 Transfer Acceleration is not supported on this bucket. For assistance, contact Amazon Web Services Support. • Amazon S3 Transfer Acceleration cannot be enabled on this bucket. For assistance, contact Amazon Web Services Support. • Conflicting values provided in HTTP headers and query parameters. • Conflicting values provided in HTTP headers and POST form fields. • CopyObject request made on objects larger than 5GB in size.		
InvalidSessionException	Returned if the session doesn't exist anymore because it timed out or expired.	400 Bad Request	Client
InvalidSignature	The request signature that the server calculated does not match the signature that you provided. Check your Amazon secret access key and signing method. For more information, see Signing and authenticating REST requests .	400 Bad Request	Client

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidSecurity	The provided security credentials are not valid.	403 Forbidden	Client
InvalidSOAPRequest	The SOAP request body is not valid.	400 Bad Request	Client
InvalidStorageClass	The storage class that you specified is not valid.	400 Bad Request	Client
InvalidTargetBucketForLogging	The target bucket for logging either does not exist, is not owned by you, or does not have the appropriate grants for the log-delivery group.	400 Bad Request	Client
InvalidToken	The provided token is malformed or otherwise not valid.	400 Bad Request	Client
InvalidURI	The specified URI couldn't be parsed.	400 Bad Request	Client
KeyTooLongError	Your key is too long.	400 Bad Request	Client
KMS.DisabledException	The request was rejected because the specified KMS key is not enabled.	400 Bad Request	Client

Error code	Description	HTTP status code	SOAP fault code prefix
KMS.InvalidKeyUsageException	The request was rejected for one of the following reasons: • The KeyUsage value of the KMS key is incompatible with the API operation. • The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (KeySpec). For encrypting, decrypting, re-encrypting, and generating data keys, the KeyUsage must be ENCRYPT_DECRYPT. For signing and verifying messages, the KeyUsage must be SIGN_VERIFY. For generating and verifying message authentication codes (MACs), the KeyUsage must be GENERATE_VERIFY_MAC. For deriving key agreement secrets, the KeyUsage must be KEY_AGREEMENT. To find the KeyUsage of a KMS key, use the DescribeKey operation. To find the encryption or signing algorithms supported for a	400 Bad Request	Client

Error code	Description	HTTP status code	SOAP fault code prefix
	particular KMS key, use the DescribeKey operation.		
KMS.KMSInvalidStateException	The request was rejected because the state of the specified resource is not valid for this request. This exception means one of the following: - The key state of the KMS key is not compatible with the operation. To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each KMS operation, see Key states of Amazon KMS keys in the <i>Amazon Key Management Service Developer Guide</i>. - For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.	400 Bad Request	Client

Error code	Description	HTTP status code	SOAP fault code prefix
<code>KMS.NotFoundException</code>	The request was rejected because the specified entity or resource could not be found.	400 Bad Request	Client
<code>MalformedACLError</code>	The ACL that you provided was not well formed or did not validate against our published schema.	400 Bad Request	Client
<code>MalformedPOSTRequest</code>	The body of your POST request is not well-formed multipart/form-data.	400 Bad Request	Client
<code>MalformedXML</code>	The XML that you provided was not well formed or did not validate against our published schema.	400 Bad Request	Client
<code>MaxMessageLengthExceeded</code>	Your request was too large.	400 Bad Request	Client
<code>MaxPostPreDataLengthExceededError</code>	Your POST request fields preceding the upload file were too large.	400 Bad Request	Client
<code>MetadataTooLarge</code>	Your metadata headers exceed the maximum allowed metadata size.	400 Bad Request	Client
<code>MethodNotAllowed</code>	The specified method is not allowed against this resource.	405 Method Not Allowed	Client

Error code	Description	HTTP status code	SOAP fault code prefix
MissingAttachment	A SOAP attachment was expected, but none was found.	400 Bad Request	Client
MissingAuthenticationToken	The request was not signed.	403 Forbidden	Client
MissingContentLength	You must provide the Content-Length HTTP header.	411 Length Required	Client
MissingRequestBodyError	You sent an empty XML document as a request.	400 Bad Request	Client
MissingSecurityElement	The SOAP 1.1 request is missing a security element.	400 Bad Request	Client
MissingSecurityHeader	Your request is missing a required header.	400 Bad Request	Client
NoLoggingStatusForKey	There is no such thing as a logging status subresource for a key.	400 Bad Request	Client
NoSuchAsyncRequest	The specified request was not found.	404 Not Found	Client
NoSuchBucket	The specified bucket does not exist.	404 Not Found	Client

Error code	Description	HTTP status code	SOAP fault code prefix
NoSuchBucketPolicy	The specified bucket does not have a bucket policy.	404 Not Found	Client
NoSuchCORSConfiguration	The specified bucket does not have a CORS configuration.	404 Not Found	Client
NoSuchKey	The specified key does not exist.	404 Not Found	Client
NoSuchLifecycleConfiguration	The specified lifecycle configuration does not exist.	404 Not Found	Client
NoSuchMultiRegionAccessPoint	The specified Multi-Region Access Point does not exist.	404 Not Found	Client
NoSuchObjectLockConfiguration	The specified object does not have an ObjectLock configuration.	404 Not Found	Client
NoSuchWebsiteConfiguration	The specified bucket does not have a website configuration.	404 Not Found	Client
NoSuchTagSet	The specified tag does not exist.	404 Not Found	Client

Error code	Description	HTTP status code	SOAP fault code prefix
NoSuchUpload	The specified multipart upload does not exist. The upload ID might not be valid, or the multipart upload might have been aborted or completed.	404 Not Found	Client
NoSuchVersion	The version ID specified in the request does not match an existing version.	404 Not Found	Client
NotDeviceOwnerError	The device that generated the token is not owned by the authenticated user.	400 Bad Request	Client
NotImplemented	A header that you provided implies functionality that is not implemented.	501 Not Implemented	Server
NotModified	The resource was not changed.	304 Not Modified	Server
NoTransformationDefined	No transformation found for this Object Lambda Access Point.	404 Not Found	Client
NotSignedUp	Your account is not signed up for the Amazon S3 service. You must sign up before you can use Amazon S3. You can sign up at the following URL: http://www.amazonaws.cn/s3	403 Forbidden	Client

Error code	Description	HTTP status code	SOAP fault code prefix
ObjectLockConfigurationNotFoundError	The Object Lock configuration does not exist for this bucket.	404 Not Found	Client
OwnershipControlsNotFoundError	The bucket ownership controls were not found.	404 Not Found	Client
OperationAborted	A conflicting conditional operation is currently in progress against this resource. Try again.	409 Conflict	Client
PermanentRedirect	The bucket that you are attempting to access must be addressed using the specified endpoint. Send all future requests to this endpoint.	301 Moved Permanently	Client
PermanentRedirectControlError	The API operation you are attempting to access must be addressed using the specified endpoint. Send all future requests to this endpoint.	301 Moved Permanently	Client
PreconditionFailed	At least one of the preconditions that you specified did not hold.	412 Precondition Failed	Client
Redirect	Temporary redirect. You are being redirected to the bucket while the Domain Name System (DNS) server is being updated.	307 Temporary Redirect	Client

Error code	Description	HTTP status code	SOAP fault code prefix
RequestHeaderSectionTooLarge	The request header and query parameters used to make the request exceed the maximum allowed size.	400 Bad Request	Client
RequestIsNotMultipartContent	A bucket POST request must be of the enclosure-type multipart/form-data.	412 Precondition Failed	Client
RequestTimeout	Your socket connection to the server was not read from or written to within the timeout period.	400 Bad Request	Client
RequestTimeTooSkewed	The difference between the request time and the server's time is too large.	403 Forbidden	Client
RequestTorrentOfBucketError	Requesting the torrent file of a bucket is not permitted.	400 Bad Request	Client
ResponseInterrupted	Returned to the original caller when an error is encountered while reading the WriteGetObjectResponse body.	400 Bad Request	Client
RestoreAlreadyInProgress	The object restore is already in progress.	409 Conflict	Client

Error code	Description	HTTP status code	SOAP fault code prefix
ServerSideEncryptionConfigurationNotFound	The server-side encryption configuration was not found.	400 Bad Request	Client
ServiceUnavailable	Service is unable to handle request.	503 Service Unavailable	Server
SignatureDoesNotMatch	The request signature that the server calculated does not match the signature that you provided. Check your Amazon secret access key and signing method. For more information, see REST Authentication and SOAP Authentication .	403 Forbidden	Client
SlowDown	Please reduce your request rate.	503 Slow Down	Server
503 SlowDown	Slow Down	503 Slow Down	Server
TagPolicyException	The tag policy does not allow the specified value for the following tag key.	400 Bad Request	Client
TemporaryRedirect	You are being redirected to the bucket while the Domain Name System (DNS) server is being updated.	307 Temporary Redirect	Client

Error code	Description	HTTP status code	SOAP fault code prefix
TokenCodeInvalidError	The serial number and/or token code you provided is not valid.	400 Bad Request	Client
TokenRefreshRequired	The provided token must be refreshed.	400 Bad Request	Client
TooManyAccessPoints	You have attempted to create more access points than are allowed for an account. For more information, see Amazon Simple Storage Service endpoints and quotas in the <i>Amazon Web Services General Reference</i> .	400 Bad Request	Client
TooManyBuckets	You have attempted to create more buckets than are allowed for an account. For more information, see Amazon Simple Storage Service endpoints and quotas in the <i>Amazon Web Services General Reference</i> .	400 Bad Request	Client
TooManyMultiRegionAccessPointregionsError	You have attempted to create a Multi-Region Access Point with more Regions than are allowed for an account. For more information, see Amazon Simple Storage Service endpoints and quotas in the <i>Amazon Web Services General Reference</i> .	400 Bad Request	Client

Error code	Description	HTTP status code	SOAP fault code prefix
TooManyMultiRegionAccessPoints	You have attempted to create more Multi-Region Access Points than are allowed for an account. For more information, see Amazon Simple Storage Service endpoints and quotas in the <i>Amazon Web Services General Reference</i> .	400 Bad Request	Client
UnauthorizedAccessError	Applicable in China Regions only. Returned when a request is made to a bucket that doesn't have an ICP license. For more information, see ICP Recordal .	403 Forbidden	Client
UnexpectedContent	This request contains unsupported content.	400 Bad Request	Client
UnexpectedIPError	Applicable in China Regions only. This request was rejected because the IP was unexpected.	403 Forbidden	Client
UnsupportedArgument	The request contained an unsupported argument.	400 Bad Request	Client
UnsupportedSignature	The provided request is signed with an unsupported STS Token version or the signature version is not supported.	400 Bad Request	Client

Error code	Description	HTTP status code	SOAP fault code prefix
UnresolvableGrantByEmailAddress	The email address that you provided does not match any account on record.	400 Bad Request	Client
UserKeyMustBeSpecified	The bucket POST request must contain the specified field name. If it is specified, check the order of the fields.	400 Bad Request	Client
NoSuchAccessPoint	The specified access point does not exist.	404 Not Found	Client
InvalidTag	Your request contains tag input that is not valid. For example, your request might contain duplicate keys, keys or values that are too long, or system tags.	400 Bad Request	Client
MalformedPolicy	Your policy contains a principal that is not valid.	400 Bad Request	Client

List of SELECT Object Content Error Codes

Important

Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. [Learn more](#)

The following table contains special errors that `SELECT Object Content` might return. For general information about Amazon S3 errors and a list of error codes, see [Error responses](#).

Error code	Description	HTTP status code	SOAP fault code prefix
<code>AmbiguousFieldName</code>	The field name matches to multiple fields in the file. Check the SQL expression and the file, and try again.	400	Client
<code>Busy</code>	The service is unavailable. Try again later.	503	Client
<code>CastFailed</code>	An attempt to convert from one data type to another using <code>CAST</code> failed in the SQL expression.	400	Client
<code>ColumnTooLong</code>	The length of a column in the result is greater than <code>maxCharsPerColumn</code> of 1 MB.	400	Client
<code>CSVEscapingRecordDelimiter</code>	A quoted record delimiter was found in the file. To allow quoted record delimiters, set <code>AllowQuotedRecordDelimiter</code> to <code>'TRUE'</code> .	400	Client
<code>CSVParsingError</code>	An error occurred while parsing the CSV file. Check the file and try again.	400	Client
<code>CSVUnescapedQuote</code>	An unescaped quote was found while parsing the CSV file. To allow quoted record delimiters, set <code>AllowQuotedRecordDelimiter</code> to <code>'TRUE'</code> .	400	Client

Error code	Description	HTTP status code	SOAP fault code prefix
EmptyRequestBody	The request body cannot be empty.	400	Client
EvaluatorBindingDoesNotExist	A column name or a path provided does not exist in the SQL expression.	400	Client
EvaluatorInvalidArguments	There is an incorrect number of arguments in the function call in the SQL expression.	400	Client
EvaluatorInvalidTimestampFormatPattern	The timestamp format string in the SQL expression is not valid.	400	Client
EvaluatorInvalidTimestampFormatPatternSymbol	The timestamp format pattern contains a symbol in the SQL expression that is not valid.	400	Client
EvaluatorInvalidTimestampFormatPatternSymbolForParsing	The timestamp format pattern contains a valid format symbol that cannot be applied to timestamp parsing in the SQL expression.	400	Client
EvaluatorInvalidTimestampFormatPatternToken	The timestamp format pattern contains a token in the SQL expression that is not valid.	400	Client
EvaluatorLikePatternInvalidEscapeSequence	An argument given to the LIKE expression was not valid.	400	Client
EvaluatorNegativeLimit	LIMIT must not be negative.	400	Client

Error code	Description	HTTP status code	SOAP fault code prefix
EvaluatorTimestampFormatPatternDuplicateFields	The timestamp format pattern contains multiple format specifiers representing the timestamp field in the SQL expression.	400	Client
EvaluatorTimestampFormatPatternHourClockAmPmMismatch	The timestamp format pattern contains a 12-hour hour of day format symbol but doesn't also contain an AM/PM field, or it contains a 24-hour hour of day format specifier and contains an AM/PM field in the SQL expression.	400	Client
EvaluatorUnterminatedTimestampFormatPatternToken	The timestamp format pattern contains an unterminated token in the SQL expression.	400	Client
ExpressionTooLong	The SQL expression is too long. The maximum byte-length for an SQL expression is 256 KB.	400	Client
ExternalEvalException	The query cannot be evaluated. Check the file and try again.	400	Client
IllegalSqlFunctionArgument	An illegal argument was used in the SQL function.	400	Client
IncorrectSqlFunctionArgumentType	An incorrect argument type was specified in a function call in the SQL expression.	400	Client
IntegerOverflow	An integer overflow or underflow occurred in the SQL expression.	400	Client

Error code	Description	HTTP status code	SOAP fault code prefix
<code>InternalError</code>	An internal error occurred.	500	Client
<code>InvalidCast</code>	An attempt to convert from one data type to another using <code>CAST</code> failed in the SQL expression.	400	Client
<code>InvalidColumnIndex</code>	The column index in the SQL expression is not valid.	400	Client
<code>InvalidCompressionFormat</code>	The file is not in a supported compression format. Only GZIP and BZIP2 are supported.	400	Client
<code>InvalidDataSource</code>	The data source type is not valid. Only CSV, JSON, and Parquet are supported.	400	Client
<code>InvalidDataType</code>	The SQL expression contains a data type that is not valid.	400	Client
<code>InvalidExpressionType</code>	The <code>ExpressionType</code> value is not valid. Only SQL expressions are supported.	400	Client
<code>InvalidFileHeaderInfo</code>	The <code>FileHeaderInfo</code> value is not valid. Only NONE, USE, and IGNORE are supported.	400	Client
<code>InvalidJsonType</code>	The <code>JsonType</code> value is not valid. Only DOCUMENT and LINES are supported.	400	Client
<code>InvalidKeyPath</code>	The key path in the SQL expression is not valid.	400	Client

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidQuoteFields	The QuoteFields value is not valid. Only ALWAYS and ASNEEDED are supported.	400	Client
InvalidRequestParameter	The value of a parameter in the SelectRequest element is not valid. Check the service API documentation and try again.	400	Client
InvalidScanRange	The provided scan range is not valid.	400	Client
InvalidTableAlias	The SQL expression contains a table alias that is not valid.	400	Client
InvalidTextEncoding	The encoding type is not valid. Only UTF-8 encoding is supported.	400	Client
JSONParsingError	An error occurred while parsing the JSON file. Check the file and try again.	400	Client
LexerInvalidChar	The SQL expression contains a character that is not valid.	400	Client
LexerInvalidIONLiteral	The SQL expression contains an operator that is not valid.	400	Client
LexerInvalidLiteral	The SQL expression contains an operator that is not valid.	400	Client
LexerInvalidOperator	The SQL expression contains a literal that is not valid.	400	Client

Error code	Description	HTTP status code	SOAP fault code prefix
LikeInvalidInputs	The argument given to the LIKE clause in the SQL expression is not valid.	400	Client
MalformedXML	The XML provided was not well formed or did not validate against our published schema. Check the service documentation and try again.	400	Client
MaxOperatorsExceeded	Failed to parse SQL expression, try reducing complexity. For example, reduce number of operators used.	400	Client
MethodNotAllowed	The specified method is not allowed against this resource.	405 Method Not Allowed	Client
MissingRequiredParameter	The SelectRequest entity is missing a required parameter. Check the service documentation and try again.	400	Client
MultipleDataSourceUnsupported	Multiple data sources are not supported.	400	Client
NumberFormatError	An error occurred while parsing a number. This error can be caused by underflow or overflow of integers.	400	Client

Error code	Description	HTTP status code	SOAP fault code prefix
ObjectSerializationConflict	InputSerialization specifies more than one format (CSV, JSON, or Parquet), or OutputSerialization specifies more than one format (CSV or JSON). For InputSerialization and OutputSerialization, you can specify only one format for each.	400	Client
OverMaxColumn	The number of columns in the result is greater than the maximum allowable number of columns.	400	Client
OverMaxParquetBlockSize	The Parquet file is above the max row group size.	400	Client
OverMaxRecordSize	The length of a record in the input or result is greater than the maxCharsPerRecord limit of 1 MB.	400	Client
ParquetParsingError	An error occurred while parsing the Parquet file. Check the file and try again.	400	Client
ParquetUnsupportedCompressionCodec	The specified Parquet compression codec is not supported.	400	Client
ParseAsteriskIsNotAloneInSelectList	Other expressions are not allowed in the SELECT list when * is used without dot notation in the SQL expression.	400	Client

Error code	Description	HTTP status code	SOAP fault code prefix
ParseCannotMixSqbandWildcardInSelectList	Cannot mix [] and * in the same expression in a SELECT list in the SQL expression.	400	Client
ParseCastArity	The SQL expression CAST has incorrect arity.	400	Client
ParseEmptySelect	The SQL expression contains an empty SELECT clause.	400	Client
ParseExpected2TokenTypes	The expected token in the SQL expression was not found.	400	Client
ParseExpectedArgumentDelimiter	The expected argument delimiter in the SQL expression was not found.	400	Client
ParseExpectedDatePart	The expected date part in the SQL expression was not found.	400	Client
ParseExpectedExpression	The expected SQL expression was not found.	400	Client
ParseExpectedIdentifierForAlias	The expected identifier for the alias in the SQL expression was not found.	400	Client
ParseExpectedIdentifierForAt	The expected identifier for AT name in the SQL expression was not found.	400	Client
ParseExpectedIdentifierForGroupName	GROUP is not supported in the SQL expression.	400	Client
ParseExpectedKeyword	The expected keyword in the SQL expression was not found.	400	Client

Error code	Description	HTTP status code	SOAP fault code prefix
ParseExpectedLeftParenAfterCast	The expected left parenthesis after CAST in the SQL expression was not found.	400	Client
ParseExpectedLeftParenBuiltinFunctionCall	The expected left parenthesis in the SQL expression was not found.	400	Client
ParseExpectedLeftParenValueConstructor	The expected left parenthesis in the SQL expression was not found.	400	Client
ParseExpectedMember	The SQL expression contains an unsupported use of MEMBER.	400	Client
ParseExpectedNumber	The expected number in the SQL expression was not found.	400	Client
ParseExpectedRightParenBuiltinFunctionCall	The expected right parenthesis character in the SQL expression was not found.	400	Client
ParseExpectedToken Type	The expected token in the SQL expression was not found.	400	Client
ParseExpectedTypeName	The expected type name in the SQL expression was not found.	400	Client
ParseExpectedWhenClause	The expected WHEN clause in the SQL expression was not found. CASE is not supported.	400	Client
ParseInvalidContextForWildcardInSelectList	The use of * in the SELECT list in the SQL expression is not valid.	400	Client

Error code	Description	HTTP status code	SOAP fault code prefix
ParseInvalidPathComponent	The SQL expression contains a path component that is not valid.	400	Client
ParseInvalidTypeParameter	The SQL expression contains a parameter value that is not valid.	400	Client
ParseMalformedJoin	JOIN is not supported in the SQL expression.	400	Client
ParseMissingIdentifierAfterAt	The expected identifier after the @ symbol in the SQL expression was not found.	400	Client
ParseNonUnaryAggregateFunctionCall	Only one argument is supported for aggregate functions in the SQL expression.	400	Client
ParseSelectMissingFrom	The SQL expression contains a missing FROM after the SELECT list.	400	Client
ParseUnexpectedKeyword	The SQL expression contains an unexpected keyword.	400	Client
ParseUnexpectedOperator	The SQL expression contains an unexpected operator.	400	Client
ParseUnexpectedTerm	The SQL expression contains an unexpected term.	400	Client
ParseUnexpectedToken	The SQL expression contains an unexpected token.	400	Client
ParseUnknownOperator	The SQL expression contains an operator that is not valid.	400	Client

Error code	Description	HTTP status code	SOAP fault code prefix
ParseUnsupportedAlias	The SQL expression contains an unsupported use of ALIAS.	400	Client
ParseUnsupportedCallWithStar	Only COUNT with (*) as a parameter is supported in the SQL expression.	400	Client
ParseUnsupportedCase	The SQL expression contains an unsupported use of CASE.	400	Client
ParseUnsupportedCaseClause	The SQL expression contains an unsupported use of CASE.	400	Client
ParseUnsupportedLiteralsGroupBy	The SQL expression contains an unsupported use of GROUP BY .	400	Client
ParseUnsupportedSelect	The SQL expression contains an unsupported use of SELECT.	400	Client
ParseUnsupportedSyntax	The SQL expression contains unsupported syntax.	400	Client
ParseUnsupportedToken	The SQL expression contains an unsupported token.	400	Client
TruncatedInput	Object decompression failed. Check that the object is properly compressed using the format specified in the request.	400	Client
UnauthorizedAccess	You are not authorized to perform this operation.	401	Client
UnrecognizedFormatException	We encountered a record type that is not valid.	400	Client

Error code	Description	HTTP status code	SOAP fault code prefix
UnsupportedFunction	We encountered an unsupported SQL function.	400	Client
UnsupportedParquet Type	The specified Parquet type is not supported.	400	Client
UnsupportedRangeHeader	A range header is not supported for this operation.	400	Client
UnsupportedScanRangeInput	Scan range queries are not supported on this type of object.	400	Client
UnsupportedSqlOperation	We encountered an unsupported SQL operation.	400	Client
UnsupportedSqlStructure	We encountered an unsupported SQL structure. Check the SQL Reference.	400	Client
UnsupportedStorageClass	We encountered a storage class that is not supported. Only STANDARD, STANDARD_IA , and ONEZONE_IA storage classes are supported.	400	Client
UnsupportedSyntax	We encountered syntax that is not valid.	400	Client
UnsupportedTypeForQuerying	Your query contains an unsupported type for comparison (e.g. verifying that a Parquet INT96 column type is greater than 0).	400	Client
ValueParseFailure	A timestamp parse failure occurred in the SQL expression.	400	Client

List of Replication-related error codes

The following table contains special errors that the Replication operation might return. For general information about Amazon S3 errors and a list of error codes, see [Error responses](#).

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidArgument	This error might occur for the following reasons: • The <Account> element is empty. It must contain a valid account ID. • The Amazon Web Services account specified in the <Account> element must match the destination bucket owner. • ReplicationTime-Status must contain a value. • ReplicationTime-ReplicationTimeValue must contain a value. • Replication-ReplicationTimeValue-Minutes value must be 15. • ReplicationMetrics must contain a Status. •	400	Client

Error code	Description	HTTP status code	SOAP fault code prefix
	ReplicationMetrics must contain an EventThreshold . • EventThreshold-ReplicationTimeValue-Minutes value must be 15. • Rule ID must not contain non-ASCII characters.		

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidRequest	This error might occur for the following reasons: - The <Owner> in <AccessControlTranslation> has a value, so the <Account> element must be specified. - The <Account> element is empty. It must contain a valid account ID. - The replication destination must contain both ReplicationTime and Metrics, or neither. - ReplicationTime and ReplicationMetrics must have the same status. - S3 Replication Time Control (S3 RTC) is not supported in this Amazon Web Services Region.	400	Client
ReplicationConfigurationNotFoundError	There is no replication configuration for this bucket.	404 Not Found	Client

List of Tagging-related error codes

The following table contains special errors that the `TagResource`, `UntagResource`, and `ListTagsForResource` operations might return for Storage Lens groups. For general information about general Amazon S3 errors and a list of error codes, see [Error responses](#).

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
<code>InvalidRequest</code>	The Amazon Web Services Region in the resource ARN doesn't match the Region that's specified in this request. The Amazon account in the resource ARN doesn't match the account ID that's specified in this request. The Amazon partition in the resourceArn is invalid.	400 Bad Request	Not supported
<code>InvalidTag</code>	This request contains a tag key or value that isn't valid. Valid characters include the following : [a-zA-Z+ -= . _ : /] . Tag keys can contain up to 128 characters. Tag values can contain up to 256 characters. There are duplicate tag keys in your request. User-defined tag keys can't start with <code>aws:</code> .	400 Bad Request	Not supported
<code>NoSuchResource</code>	The specified resource doesn't exist.	404 Not Found	Not supported
<code>TagPolicyException</code>	The tag policy does not allow the specified value for the following tag key.	400 Bad Request	Not supported

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
TooManyTags	The number of tags exceeds the limit of 50 tags.	400 Bad Request	Not supported

List of Amazon S3 on Outposts error codes

The following table contains special errors that an Amazon S3 on Outposts operation might return. For general information about Amazon S3 errors and a list of error codes, see [Error responses](#).

Error code	Description	HTTP status code	SOAP fault code prefix
BadRequest	The bucket is in a transitional state because of a previous deletion attempt. Try again later.	400 Bad Request	Not supported
InvalidRequest	This error might occur for the following reasons: - Amazon VPC configuration is required. - Public access is not allowed on S3 on Outposts access points.	400 Bad Request	Client
InvalidOutpostState	The request is not valid for the current state of the Outpost.	409 Conflict	Not supported
InvalidRequest	The access point is not in a state where it can be deleted.	400 Bad Request	Not supported
NoSuchOutpost	The specified Outpost does not exist.	404 Not Found	Not supported

Error code	Description	HTTP status code	SOAP fault code prefix
UnsupportedOperation	The specified action was not supported.	404 Not Found	Not supported
InsufficientCapacity	Insufficient capacity.	507 Insufficient Storage	Not supported

List of Amazon S3 Storage Lens error codes

The following table contains special errors that Amazon S3 Storage Lens operations might return. For general information about general Amazon S3 errors and a list of error codes, see [Error responses](#).

Error code	Description	HTTP status code	SOAP fault code prefix
AccessDenied	This Region is not supported as a home Region for S3 Storage Lens.	403 Forbidden	Not supported
AccountNotAuthorized	This account not authorized to use Amazon Organizations. Use your management account or delegated administrator account.	403 Forbidden	Not supported
ActivityMetricsMustEnabled	Activity metrics must be enabled.	400 Bad Request	Not supported
AWSOrganizationsNotInUseException	This account is not part of your organization.	403 Forbidden	Not supported
DefaultConfigurationDeleteForbidden	The Default configuration cannot be deleted.	403 Forbidden	Not supported

Error code	Description	HTTP status code	SOAP fault code prefix
DuplicateStorageLensGroupARN	There are two or more entries of the same Storage Lens group ARN in this configuration.	400 Bad Request	Not supported
EmptyExcludeContainer	This error occurs for the following reasons: • The exclude container cannot be empty. • The exclude container cannot have zero buckets. • The exclude container cannot have zero Regions.	400 Bad Request	Not supported
EmptyExcludeElement	You must specify a Storage Lens group with your Exclude element.	400 Bad Request	Not supported
EmptyIncludeContainer	This error occurs for the following reasons: • The include container cannot be empty. • The include container cannot have zero buckets. • The include container cannot have zero Regions.	400 Bad Request	Not supported

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidAWSOrgArn	There is a malformed Amazon Organizations ARN in the configuration.	400 Bad Request	Not supported
EmptyIncludeElement	You must specify a Storage Lens group with your Include element.	400 Bad Request	Not supported
InvalidBucketFilter	Organization-level configurations do not support bucket filters.	400 Bad Request	Not supported
InvalidConfigId	The configuration ID is not valid.	400 Bad Request	Not supported
InvalidDestination	The S3 bucket ARN is malformed.	400 Bad Request	Not supported
InvalidEncryptionMethod	Only one encryption method can be specified.	400 Bad Request	Not supported
InvalidFilterForDefaultConfiguration	The default configuration must not include any filters.	400 Bad Request	Not supported
InvalidIncludeExcludeContainers	You can specify either an Include container or an Exclude container in a configuration. You cannot specify both in a configuration.	400 Bad Request	Not supported
InvalidIncludeExcludeElements	Only one Include or Exclude element is allowed. At least one Include or Exclude element must be present.	400 Bad Request	Not supported
InvalidKMSEncryptionKeyId	The KMS key ID ARN is not valid.	400 Bad Request	Not supported

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidMaximumPrefixDepth	MaxDepth must be within the range [1,10].	400 Bad Request	Not supported
InvalidMinimumStorageBytesPercentage	MinStorageBytesPercentage must be within the range [1.00,100.00].	400 Bad Request	Not supported
InvalidOrganizationARN	The Amazon Organizations ARN in the configuration is not valid.	400 Bad Request	Not supported
InvalidOrganizationForDefaultConfiguration	The default configuration does not support organization-level metrics.	400 Bad Request	Not supported
InvalidRegionForDefaultConfiguration	The specified Region is not supported for default configuration.	400 Bad Request	Not supported
InvalidRegionName	The Region name is not valid.	400 Bad Request	Not supported
InvalidStorageLensArn	The S3 Storage Lens ARN is not required in input.	400 Bad Request	Not supported

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidStorageLensGroupARN	This Storage Lens group ARN isn't valid or only Storage Lens groups in your account are allowed. Additionally, you must follow the Storage Lens group ARN structure :arn::s3:::storage-lens-group/ and adhere to the 64 character limit. Storage Lens group names can also contain only the following characters: a-z, A-Z, 0-9, hyphens (-), and underscores (_).	400 Bad Request	Not supported
MissingAccountLevelActivityMetrics	Activity metrics must be enabled at the account level when activity metrics are enabled at the bucket level.	400 Bad Request	Not supported
MissingBucketLevelActivityMetrics	Activity metrics must be enabled at the bucket level when activity metrics are enabled at the account level.	400 Bad Request	Not supported
MissingEncryptionMethod	The encryption method cannot be blank. Specify either SSE-KMS or SSE-S3.	400 Bad Request	Not supported
MissingPrefixLevelStorageMetrics	Storage metrics at the prefix level are mandatory when the prefix level is enabled.	400 Bad Request	Not supported
OrganizationAccessDenied	This account is not authorized to add Amazon Organizations.	403 Forbidden	Not supported

Error code	Description	HTTP status code	SOAP fault code prefix
OrgConfigurationNotSupported	The specified Region does not support Amazon Organizations in the configuration.	403 Forbidden	Not supported
ServiceNotEnabledForOrg	The S3 Storage Lens service-linked role is not enabled for the organization.	403 Forbidden	Not supported
StorageMetricsMustEnabled	Prefix-level storage metrics must be enabled.	400 Bad Request	Not supported
TooManyBuckets	The buckets container cannot have more than 50 buckets.	400 Bad Request	Not supported
TooManyRegions	The Regions container cannot have more than 50 Regions.	400 Bad Request	Not supported
TooManyStorageLensGroups	You can't attach more than 50 Storage Lens groups to your Storage Lens dashboard.	400 Bad Request	Not supported

The following table contains special errors that S3 Storage Lens groups operations might return. For general information about general Amazon S3 errors and a list of error codes, see [Error responses](#).

Error code	Description	HTTP status code	SOAP fault code prefix
AccessDenied	You don't have permission to perform Storage Lens group actions. This Region is not supported as home Region for S3 Storage Lens groups.	403 Forbidden	Not supported

Error code	Description	HTTP status code	SOAP fault code prefix
ConfigurationAlreadyExists	The specified configuration already exists.	409 Conflict	Not supported
DuplicateElement	Tags must be unique. The And logical operator includes duplicate tag keys. The Or logical operator includes duplicate tags. Logical operator includes duplicate prefixes or suffixes.	400 Bad Request	Not supported
InvalidAge	DaysLessThan and DaysGreaterThan must be positive numbers.	400 Bad Request	Not supported
InvalidFilter	A filter must include one of the following elements: And, Or, MatchAnyTag , MatchAnyPrefix , MatchAnySuffix , MatchObjectAge , MatchObjectSize .	400 Bad Request	Not supported
InvalidLogicalOperator	At least two sub elements must be present in the logical operators And or Or.	400 Bad Request	Not supported
InvalidMatchAnyPrefix	The MatchAnyPrefix parameter can't be empty.	400 Bad Request	Not supported
InvalidMatchAnySuffix	The MatchAnySuffix parameter can't be empty.	400 Bad Request	Not supported
InvalidMatchAnyTag	The MatchAnyTag parameter can't be empty.	400 Bad Request	Not supported

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidMatchObjectAge	The MatchObjectAge parameter can't be empty.	400 Bad Request	Not supported
InvalidMatchObjectSize	The MatchObjectSize parameter can't be empty.	400 Bad Request	Not supported
InvalidName	Storage Lens group Name parameter must be between 1 and 64 characters. The Storage Lens group Name parameter must use the <code>^[a-zA-Z0-9\-_]+\$</code> pattern.	400 Bad Request	Not supported
InvalidNumericCombination	This object age or object size combination isn't valid.	400 Bad Request	Not supported
InvalidPrefix	The maximum length of a prefix is 1,024 characters. The prefix string can't be empty.	400 Bad Request	Not supported
InvalidSize	BytesLessThan and BytesGreaterThan must be positive numbers. The maximum object size can't exceed 50 TB. The minimum object size can't be greater than or equal to 50 TB.	400 Bad Request	Not supported
InvalidSuffix	The maximum length of a suffix is 1,024 characters. The suffix string can't be empty.	400 Bad Request	Not supported

Error code	Description	HTTP status code	SOAP fault code prefix
InvalidTag	The object tag key can't exceed 128 characters. The object tag key string can't be null or empty. The maximum length of a tag value is 256 characters. The object tag key contains characters that aren't valid. The object tag key must contain only a-z, A-Z, 0-9, spaces, and the following characters: ^(_ . : / = + \ - @] *) \$.	400 Bad Request	Not supported
MismatchedName	The name specified in the request doesn't match the Storage Lens group name.	400 Bad Request	Not supported
TooManyConfigurations	You have attempted to create more Storage Lens group configurations than the 50 allowed.	400 Bad Request	Not supported
TooManyElements	The Element exceeds the maximum number of elements allowed within a logical operator. Only 10 prefixes, suffixes, or tags are allowed.	400 Bad Request	Not supported

List of Amazon S3 Object Lambda error codes

The following table contains special errors that S3 Object Lambda might return. For information about general Amazon S3 errors and a list of error codes, see [Error responses](#).

Error responses received from the supporting access points during non-GetObject requests are sent to the caller unaltered.

Error code	Description	HTTP status code	
LambdaInvalidResponse	Returned to the original caller when <code>WriteGetObjectResponse</code> responds with <code>ValidationError</code> to Amazon Lambda. See the <code>ValidationError</code> message for more details. Not all cases of <code>ValidationError</code> result in a <code>LambdaInvalidResponse</code> error.	400 Bad Request	
LambdaInvocationFailed	Lambda function invocation failed. Callers might receive the following error when S3 Object Lambda is unable to successfully invoke the configured Lambda function. The error message might contain details about an eventual error returned by the Amazon Lambda service when invoking the function (for example, status code, error code, error message and request ID).	400 Bad Request	
LambdaNotFound	The Amazon Lambda function was not found. The configured Lambda function, version, or alias was not found when attempting to invoke it. Ensure that the S3 Object Lambda Access Point configuration points	404 Not Found	

Error code	Description	HTTP status code	
	to the correct Lambda function ARN. The error message might contain details about an eventual error returned by the Amazon Lambda service when invoking the function (for example, status code, error code, error message and request ID).		
LambdaPermissionError	The caller is not authorized to invoke the Lambda function. The caller must have permission to invoke the Lambda function. Check the policies attached to the caller and ensure that they've been allowed to use <code>lambda:Invoke</code> for the configured function. The error message might contain details about an eventual error returned by the Amazon Lambda service when invoking the function (for example, status code, error code, error message and request ID).	403 Forbidden	

Error code	Description	HTTP status code	
LambdaResponseNotReceived	The Lambda function exited without successfully calling <code>WriteGetObjectResponse</code> . <code>GetObject</code> response data is provided by the Lambda function by calling the <code>WriteGetObjectResponse</code> API operation . The Amazon CloudWatch logs for the function might provide more insight into why the function did not successfully call this API operation despite exiting normally.	500 Internal Service Error	
LambdaRuntimeError	The Lambda function failed during execution. An explicit error was received from the Lambda function. For details about the failure, check the Amazon CloudFormation logs.	500 Internal Service Error	
LambdaTimeout	The Lambda function did not respond in the allowed time. The Lambda function failed to complete its call to <code>WriteGetObjectResponse</code> within 60 seconds.	500 Internal Service Error	

Error code	Description	HTTP status code	
SlowDown	Reduce your request rate for operations involving Amazon Lambda. The function invocation was throttled by Amazon Lambda, perhaps because it has reached its configured concurrency limit. For more information, see Managing concurrency for a Lambda function in the <i>Amazon Lambda Developer Guide</i>. The error message might contain details about an eventual error returned by the Amazon Lambda service when invoking the function (for example, status code, error code, error message and request ID).	503 Slow Down	
ValidationError	Validation errors might be returned from the WriteGetObjectResponse API operation and can occur for numerous reasons. See the error message for more details.	400 Bad Request	

List of Amazon S3 asynchronous error codes

The following table contains special errors that asynchronous requests might return. For general information about Amazon S3 errors and a list of error codes, see [Error responses](#).

These errors are returned when you query about the state of an asynchronous request, such as by using `DescribeMultiRegionAccessPointOperation`. Because these requests are asynchronous, all of these errors have a status code of `200 OK`.

Error code	Description	HTTP status code	
<code>AccessDenied</code>	Access denied.	<code>200 OK</code>	
<code>InternalErrors</code>	An internal server error occurred.	<code>200 OK</code>	
<code>MalformedPolicy</code>	The specified policy syntax is not valid.	<code>200 OK</code>	
<code>MultiRegionAccessPointAlreadyOwnedByYou</code>	You already have a Multi-Region Access Point with the same name.	<code>200 OK</code>	
<code>MultiRegionAccessPointModifiedByAnotherRequest</code>	The action failed because another request is modifying the specified resource. Try resubmitting your request after the previous request has been completed.	<code>200 OK</code>	
<code>MultiRegionAccessPointNotReady</code>	The specified Multi-Region Access Point is not ready to be updated.	<code>200 OK</code>	
<code>MultiRegionAccessPointSameBucketRegion</code>	The buckets used to create a Multi-Region Access Point cannot be in the same Region.	<code>200 OK</code>	
<code>MultiRegionAccessPointUnsupportedRegion</code>	One of the buckets supplied to create the Multi-Region Access Point is in a Region that is not supported.	<code>200 OK</code>	
<code>NoSuchBucket</code>	The specified bucket does not exist.	<code>200 OK</code>	

Error code	Description	HTTP status code	
NoSuchMultiRegionAccessPoint	The specified Multi-Region Access Point does not exist.	200 OK	

List of Amazon S3 Access Grants error codes

The following table contains special errors that S3 Access Grants requests might return. For general information about Amazon S3 errors and a list of error codes, see [Error responses](#).

Error Code	Description	HTTP Status Code	
AccessGrantAlreadyExists	The specified access grant already exists	409	
AccessGrantsInstanceAlreadyExists	Access Grants Instance already exists	409	
AccessGrantsInstanceNotEmptyError	Please clean up locations before deleting the access grants instance	400	
AccessGrantsInstanceDoesNotExistError	Access Grants Instance does not exist	404	
AccessGrantsInstanceResourcePolicyDoesNotExist	Access Grants Instance Resource Policy does not exist	404	
AccessGrantsLocationAlreadyExistsError	The specified access grants location already exists	409	
AccessGrantsLocationNotEmptyError	Please clean up access grants before deleting access grants location	400	

Error Code	Description	HTTP Status Code	
AccessGrantsLocationsQuotaExceededError	The access grants location quota has been exceeded. Access Grants Locations Quota: <i><value></i> . Please reach out to S3 if an increase is required.	409	
AccessGrantsQuotaExceededError	The access grants quota has been exceeded. Access Grants Quota: <i><value></i> . Please reach out to S3 if an increase is required.	409	
InvalidTag	There are duplicate tag keys in your request. Remove the duplicate tag keys and try again.	400	
InvalidAccessGrant	The specified Access Grant is invalid	400	
InvalidAccessGrantLocation	The specified Access Grants Location is invalid	400	
InvalidIamRole	The specified IAM Role is invalid	400	
InvalidIdentityCenterInstance	The specified identity center instance is invalid	400	
InvalidResourcePolicy	The specified Resource Policy is invalid	400	
InvalidResourcePolicy	The specified Resource Policy is invalid	400	

Error Code	Description	HTTP Status Code	
InvalidTag	This request contains a tag key or value that isn't valid. Valid characters include the following: [a-zA-Z+-. _:/]. Tag keys can contain up to 128 characters. Tag values can contain up to 256 characters.	400	
NoSuchAccessGrantError	The specified access grant does not exist	404	
NoSuchAccessGrantsLocationError	The specified access grants location does not exist	404	
AccessDenied	You do not have <i><requested permission></i> permissions to the requested S3 Prefix: <i><requested target></i>	403 Forbidden	
StsNotAuthorizedError	An error occurred (StsNotAuthorizedError) when calling the GetDataAccess operation: User: access-grants.s3.amazonaws.com is not authorized to perform: sts:AssumeRole on resource: <i><IAM Role ARN></i>	403	
StsPackedPolicyTooLargeError	An error occurred (StsPackedPolicyTooLargeError) when calling the GetDataAccess operation : Serialized token too large for session	400	
StsValidationError	<i>The error message varies depending on the validation error.</i>	400	

Error Code	Description	HTTP Status Code	
InvalidTags	Tag keys cannot start with Amazon reserved prefix for system tags."	400	
TooManyTags	The number of tags exceeds the limit of 50 tags. Remove some tags and try again.	400	

List of Amazon FSx-related error codes

The following table contains special errors that requests made to Amazon FSx through an Amazon S3 access point might return. For general information about Amazon S3 errors and a list of error codes, see [Error responses](#).

Error Code	Description	HTTP Status Code	
AccessDenied	This error might occur for the following reasons: • Caller is not authorized to preform specified S3 operation. • Caller is not authorized to access the Amazon FSx resource for the specified key.	403 Forbidden	
InsufficientCapacity	Maximum storage capacity of file system has been reached.	507 Insufficient Capacity	
InvalidKey	The specified key is not valid.	400 Bad Request	

Error Code	Description	HTTP Status Code	
SlowDown	Please reduce your request rate.	503 Slow Down	

List of Amazon S3 Tables error codes

The following table contains special errors that Amazon S3 Tables operations might return. For general information about Amazon S3 errors and a list of error codes, see [Error responses](#).

Error Code	Description	HTTP Status Code
AccessDeniedException	The action cannot be performed because you do not have the required permission.	403 Forbidden
BadRequestException	The request is invalid or malformed.	400 Bad Request
ConflictException	The request failed because there is a conflict with a previous write. You can retry the request.	409 Conflict
ForbiddenException	The caller isn't authorized to make the request.	403 Forbidden
InternalServerErrorException	The request failed due to an internal server error.	500 Internal Server Error
NotFoundException	The request was rejected because the specified resource could not be found.	404 Not Found

Error Code	Description	HTTP Status Code
TooManyRequestsException	The limit on the number of requests per second was exceeded.	429 Too Many Requests

Amazon S3 error best practices

Many error responses contain additional structured data meant to be read and understood by a developer diagnosing programming errors. For example, if you send a Content-MD5 header with a REST PUT request that doesn't match the digest calculated on the server, you receive a BadDigest error. The error response also includes as detail elements the digest we calculated, and the digest you told us to expect. During development, you can use this information to diagnose the error. In production, a well-behaved program might include this information in its error log.

When designing an application for use with Amazon S3, it is important to handle Amazon S3 errors appropriately. This section describes issues to consider when designing your application.

Retry InternalErrors

Internal errors are errors that occur within the Amazon S3 environment.

Requests that receive an InternalError response might not have processed. For example, if a PUT request returns InternalError, a subsequent GET might retrieve the old value or the updated value.

If Amazon S3 returns an InternalError response, retry the request.

Tune application for repeated SlowDown errors

As with any distributed system, S3 has protection mechanisms which detect intentional or unintentional resource over-consumption and react accordingly. SlowDown errors can occur when a high request rate triggers one of these mechanisms. Reducing your request rate will decrease or eliminate errors of this type. Generally speaking, most users will not experience these errors regularly; however, if you would like more information or are experiencing high or unexpected SlowDown errors, please post to our [Amazon S3 developer forum](https://forums.aws.amazon.com/thread.jspa?threadID=234567) or sign up for Amazon Web Services Support <http://www.amazonaws.cn/support-plans/>.

Isolate errors

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

Amazon S3 provides a set of error codes that are used by both the SOAP and REST API. The SOAP API returns standard Amazon S3 error codes. The REST API is designed to look like a standard HTTP server and interact with existing HTTP clients (e.g., browsers, HTTP client libraries, proxies, caches, and so on). To ensure the HTTP clients handle errors properly, we map each Amazon S3 error to an HTTP status code.

HTTP status codes are less expressive than Amazon S3 error codes and contain less information about the error. For example, the `NoSuchKey` and `NoSuchBucket` Amazon S3 errors both map to the HTTP `404 Not Found` status code.

Although the HTTP status codes contain less information about the error, clients that understand HTTP, but not the Amazon S3 API, will usually handle the error correctly.

Therefore, when handling errors or reporting Amazon S3 errors to end users, use the Amazon S3 error code instead of the HTTP status code as it contains the most information about the error. Additionally, when debugging your application, you should also consult the human readable `<Details>` element of the XML error response.

Amazon Glossary

For the latest Amazon terminology, see the [Amazon glossary](#) in the *Amazon Web Services Glossary Reference*.

Amazon S3 Resources

Following is a table that lists related resources that you'll find useful as you work with this service.

Resource	Description
Amazon Simple Storage Service User Guide	The getting started guide provides a quick tutorial of the service based on a simple use case.
Amazon Simple Storage Service User Guide	The developer guide describes how to accomplish tasks using Amazon S3 operations.
Amazon S3 Technical FAQ	The FAQ covers the top 20 questions developers have asked about this product.
Amazon S3 Release Notes	The Release Notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
Tools for Amazon Web Services	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with Amazon SDKs and tools.
Amazon Web Services Management Console	The console allows you to perform most of the functions of Amazon S3 without programming.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
Amazon Web Services Support Center	The home page for Amazon Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and Premium Support.
Amazon Web Services Support	The primary web page for information about Amazon Web Services Support, a one-on-one, fast-response support channel to help you build and run applications on Amazon Infrastructure Services.

Resource	Description
Amazon S3 product information	The primary web page for information about Amazon S3.
Contact Us	A central contact point for inquiries concerning Amazon billing, account, events, abuse, etc.
Conditions of Use	Detailed information about the copyright and trademark usage at Amazon.com and other topics.

Appendix

Topics

- [Appendix: SelectObjectContent Response](#)
- [Appendix: OPTIONS object](#)
- [Appendix: SOAP API](#)
- [Appendix: Authenticating requests \(Amazon signature version 2\)](#)
- [Appendix: Lifecycle Configuration APIs \(Deprecated\)](#)

Appendix: SelectObjectContent Response

Description

The Amazon S3 Select operation filters the contents of an Amazon S3 object based on a simple structured query language (SQL) statement. Given the response size of this operation is unknown, Amazon S3 Select streams the response as a series of messages and includes a Transfer-Encoding header with **chunked** as its value in the response.

For more information about Amazon S3 Select, see [Selecting Content from Objects](#) in the *Amazon Simple Storage Service User Guide*.

For more information about using SQL with Amazon S3 Select, see [SQL Reference for Amazon S3 Select and Amazon Glacier Select](#) in the *Amazon Simple Storage Service User Guide*.

Responses

A successful Amazon S3 Select Operation returns 200 OK status code.

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see [Common Response Headers](#).

Response Body

Since the Amazon S3 Select response size is unknown, Amazon S3 streams the response as a series of messages and includes a Transfer-Encoding header with **chunked** as its value in the response. The following example shows the response format at the top level:

```
<Message 1>
<Message 2>
<Message 3>
.....
<Message n>
```

Each message consists of two sections: the prelude and the data. The prelude section consists of 1) the total byte-length of the message, and 2) the combined byte-length of all the headers. The data section consists of 1) the headers, and 2) a payload.

Each section ends with a 4-byte big-endian integer checksum (CRC). Amazon S3 Select uses CRC32 (often referred to as GZIP CRC32) to calculate both CRCs. For more information about CRC32, see [GZIP file format specification version 4.3](#).

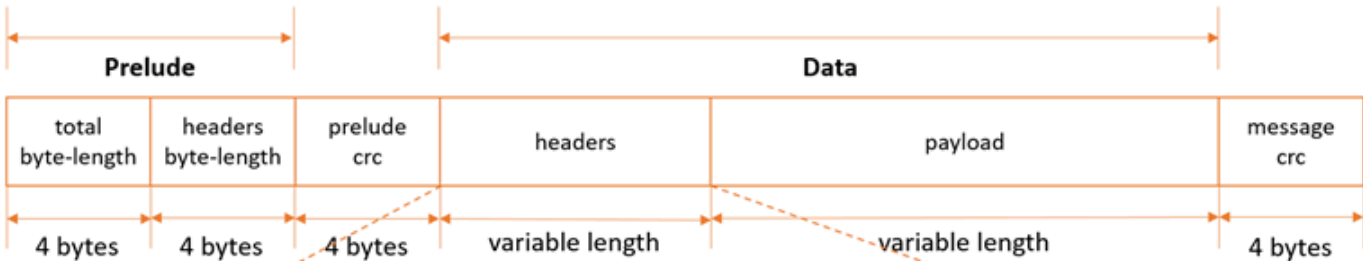
Total message overhead including the prelude and both checksums is 16 bytes.



Note
All integer values within messages are in network byte order, or big-endian order.

The following diagram shows the components that make up a message and a header. Note that there are multiple headers per message.

Message:



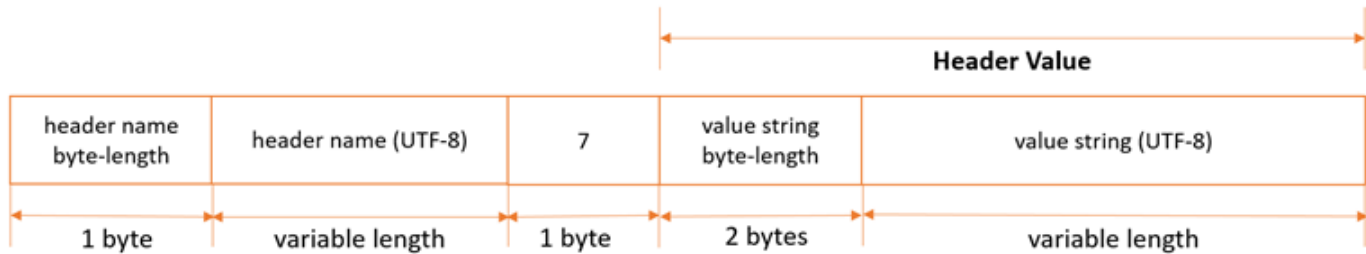
Headers (multiple headers per message):





Note
For Amazon S3 Select, the header value type is always 7 (type=String). For this type, the header value consists of two components, a 2-byte big-endian integer length, and a UTF-8 string that is of that byte-length. The following diagram shows the components that make up Amazon S3 Select headers.

Amazon S3 Select Headers (type=String):



Payload byte-length calculations (these two calculations are equivalent):

- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - \text{sizeof}(\text{total\_length}) - \text{sizeof}(\text{header\_length}) - \text{sizeof}(\text{prelude\_crc}) - \text{sizeof}(\text{message\_crc})$
- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - 16$

Each message contains the following components:

- **Prelude:** Always fixed size of 8 bytes (two fields of 4 bytes each):
 - *First four bytes:* Total byte-length: Big-endian integer byte-length of the entire message (including the 4-byte total length field itself).
 - *Second four bytes:* Headers byte-length: Big-endian integer byte-length of the headers portion of the message (excluding the headers length field itself).
- **Prelude CRC:** 4-byte big-endian integer checksum (CRC) for the prelude portion of the message (excluding the CRC itself). The prelude has a separate CRC from the message CRC (see below), to ensure that corrupted byte-length information can be detected immediately, without causing pathological buffering behavior.
- **Headers:** A set of metadata annotating the message, such as the message type, payload format, and so on. Messages can have multiple headers, so this portion of the message can have different byte-lengths depending on the message type. Headers are key-value pairs, where both the key and value are UTF-8 strings. Headers can appear in any order within the headers portion of the message, and any given header type can only appear once.

For Amazon S3 Select, following is a list of header names and the set of valid values depending on the message type.

- *MessageType Header:*

- **HeaderName** => ":message-type"
- **Valid HeaderValues** => "error", "event"
- **EventType Header:**
 - **HeaderName** => ":event-type"
 - **Valid HeaderValues** => "Records", "Cont", "Progress", "Stats", "End"
- **ErrorCode Header:**
 - **HeaderName** => ":error-code"
 - **Valid HeaderValues** => Error Code from the table in the [List of SELECT Object Content Error Codes](#) section.
- **ErrorMessage Header:**
 - **HeaderName** => ":error-message"
 - **Valid HeaderValues** => Error message returned by the service, to help diagnose request-level errors.
- **Payload:** Can be anything.
- **Message CRC:** 4-byte big-endian integer checksum (CRC) from the start of the message to the start of the checksum (that is, everything in the message excluding the message CRC itself).

Each header contains the following components. There can be multiple headers per message.

- **Header Name Byte-Length:** Byte-length of the header name.
- **Header Name:** Name of the header, indicating the header type. Valid values: ":message-type" ":event-type" ":error-code" ":error-message"
- **Header Value Type:** Enum indicating the header value type. For Amazon S3 Select, this is always 7.
- **Value String Byte-Length:** (For Amazon S3 Select) Byte-length of the header value string.
- **Header Value String:** (For Amazon S3 Select) Value of the header string. Valid values for this field vary based on the type of the header. See the sections below for valid values for each header type and message type.

For Amazon S3 Select, responses can be messages of the following types:

- **Records message:** Can contain a single record, partial records, or multiple records. Depending on the size of the result, a response can contain one or more of these messages.

- **Continuation message:** Amazon S3 periodically sends this message to keep the TCP connection open. These messages appear in responses at random. The client must detect the message type and process accordingly.
- **Progress message:** Amazon S3 periodically sends this message, if requested. It contains information about the progress of a query that has started but has not yet completed.
- **Stats message:** Amazon S3 sends this message at the end of the request. It contains statistics about the query.
- **End message:** Indicates that the request is complete, and no more messages will be sent. You should not assume that the request is complete until the client receives an End message.
- **RequestLevelError message:** Amazon S3 sends this message if the request failed for any reason. It contains the error code and error message for the failure. If Amazon S3 sends a `RequestLevelError` message, it doesn't send an End message.

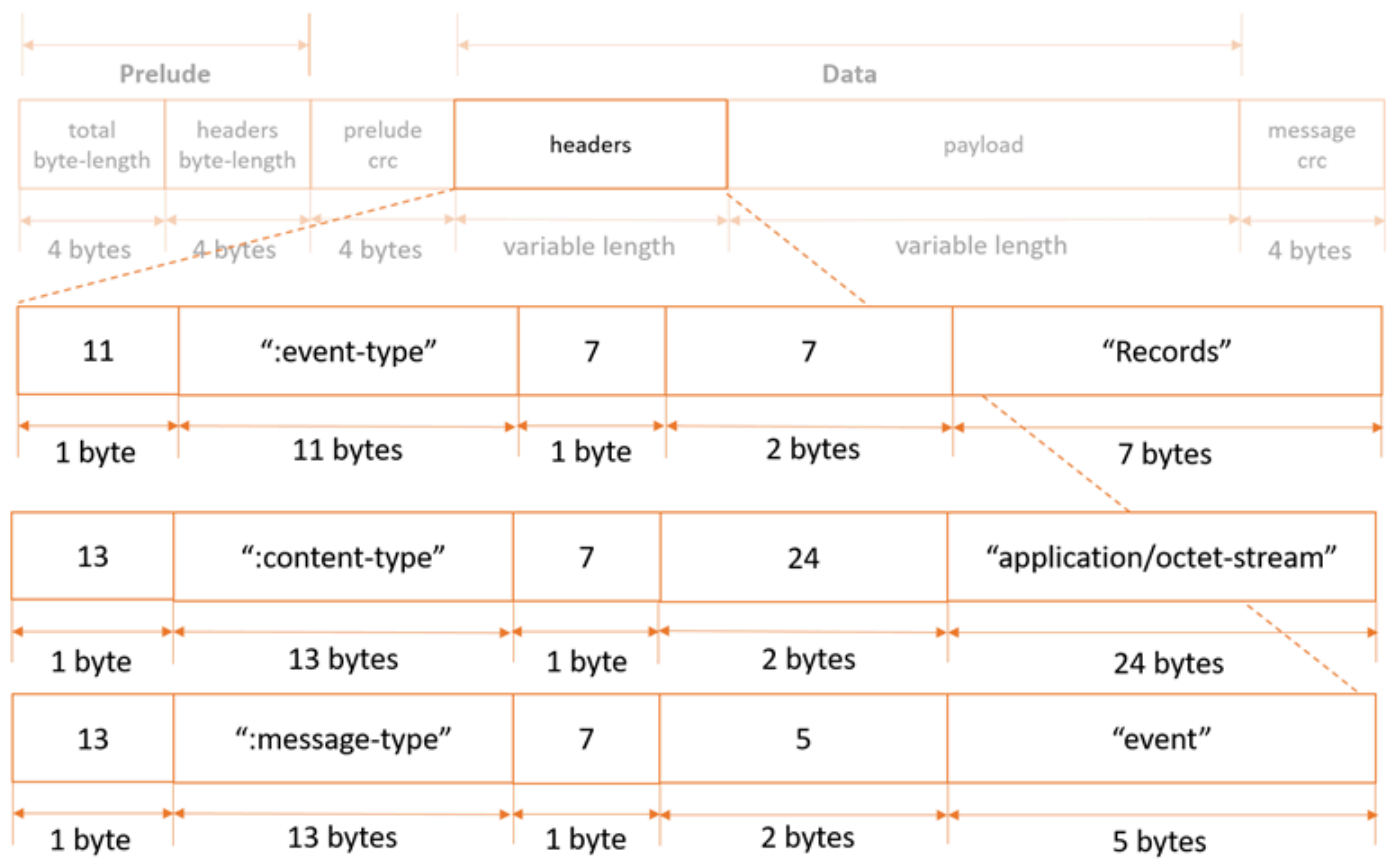
The following sections explain the structure of each message type in more detail.

For sample code and unit tests that use this protocol, see [Amazon C Event Stream](#) on the GitHub website.

Records Message

Header specification

Records messages contain three headers, as follows:



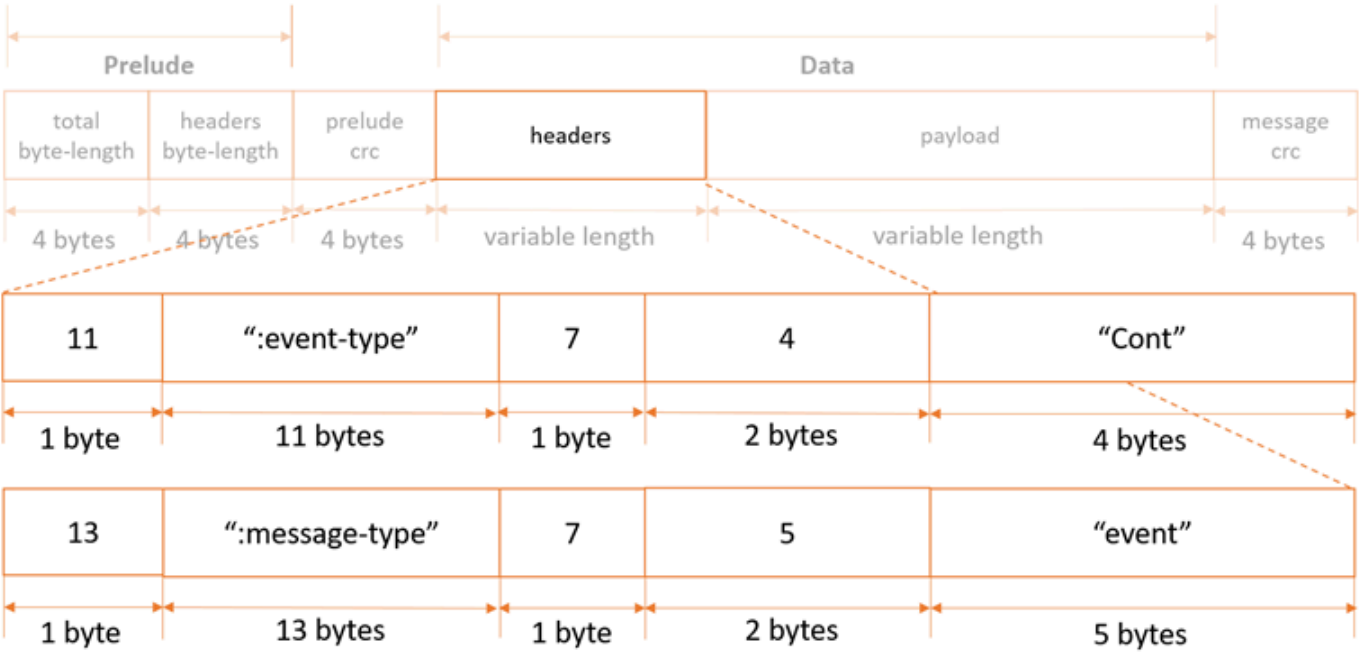
Payload specification

Records message payloads can contain a single record, partial records, or multiple records.

Continuation Message

Header specification

Continuation messages contain two headers, as follows:



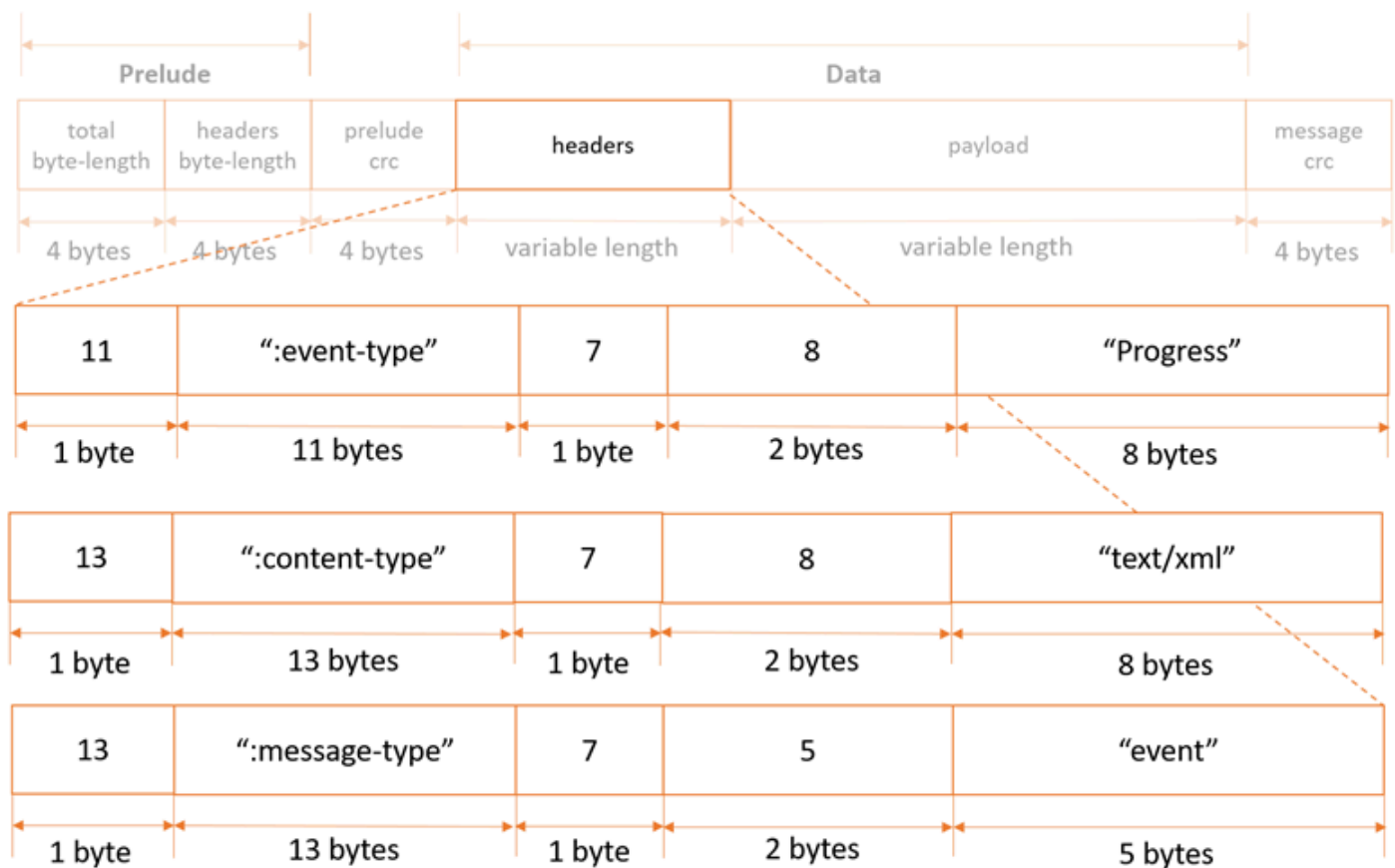
Payload specification

Continuation messages have no payload.

Progress Message

Header specification

Progress messages contain three headers, as follows:



Payload specification

Progress message payload is an XML document containing information about the progress of a request.

- *BytesScanned* => Number of bytes that have been processed before being uncompressed (if the file is compressed).
- *BytesProcessed* => Number of bytes that have been processed after being uncompressed (if the file is compressed).
- *BytesReturned* => Current number of bytes of records payload data returned by Amazon S3.

For uncompressed files, *BytesScanned* and *BytesProcessed* are equal.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

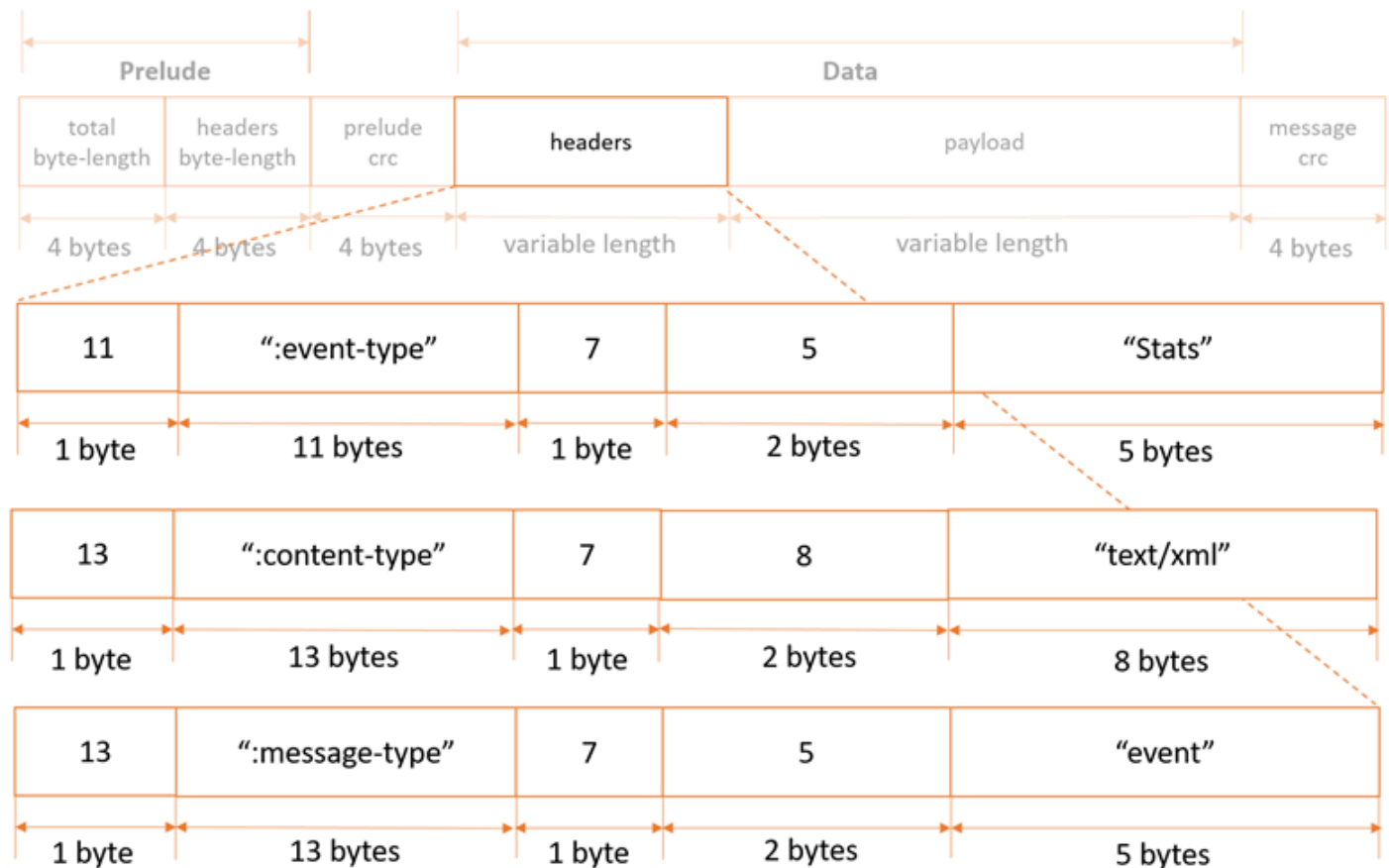
<Progress>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Progress>

```

Stats Message

Header specification

Stats messages contain three headers, as follows:



Payload specification

Stats message payload is an XML document containing information about a request's stats when processing is complete.

- *BytesScanned* => Number of bytes that have been processed before being uncompressed (if the file is compressed).

- *BytesProcessed* => Number of bytes that have been processed after being uncompressed (if the file is compressed).
- *BytesReturned* => Total number of bytes of records payload data returned by Amazon S3.

For uncompressed files, BytesScanned and BytesProcessed are equal.

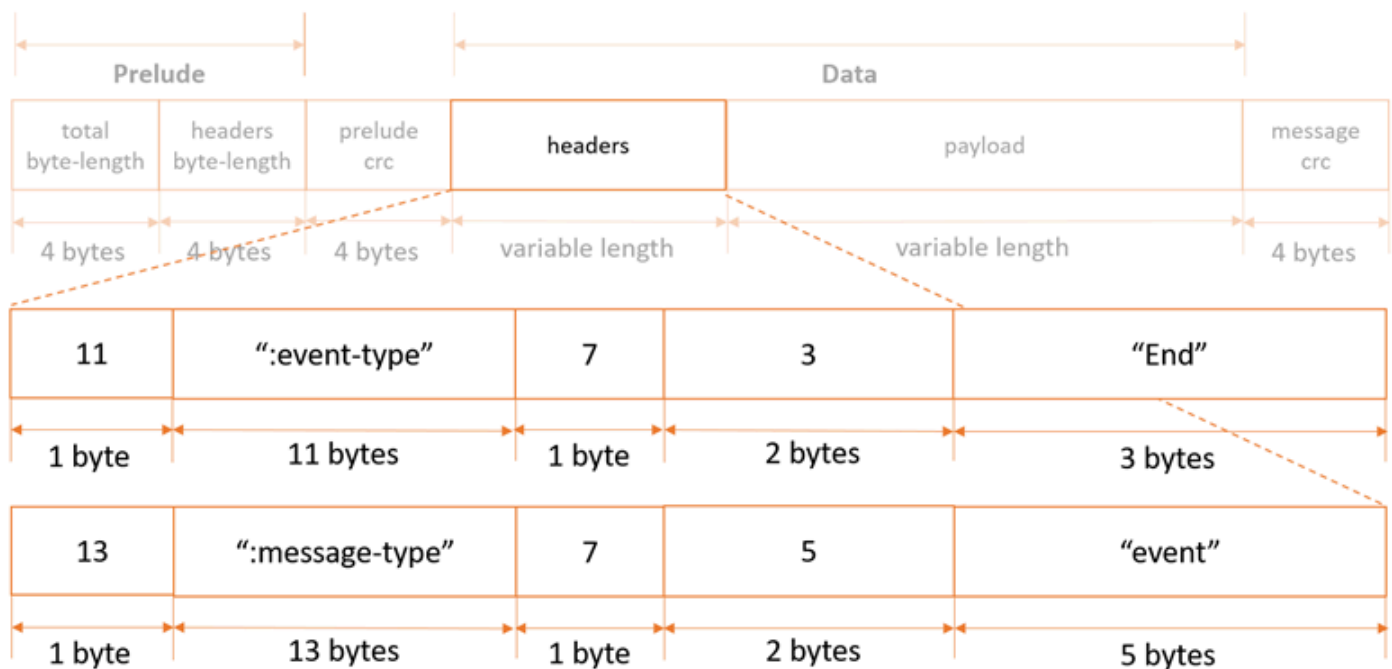
Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Stats>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Stats>
```

End Message

Header specification

End messages contain two headers, as follows:



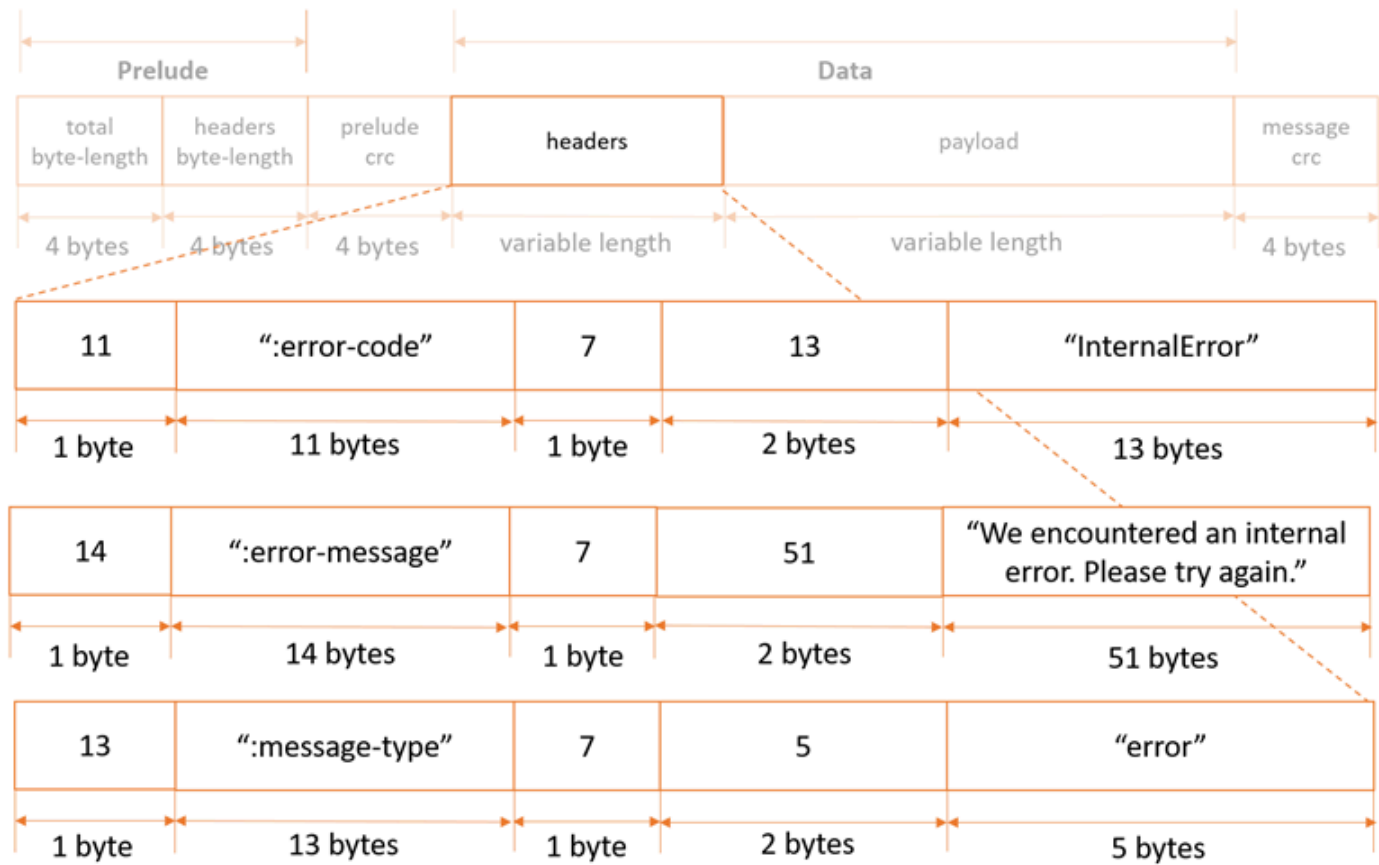
Payload specification

End messages have no payload.

Request Level Error Message

Header specification

Request-level error messages contain three headers, as follows:



For a list of possible error codes and error messages, see the [List of SELECT Object Content Error Codes](#).

Payload specification

Request-level error messages have no payload.

Related Resources

- [SelectObjectContent](#)

- [GetObject](#)
- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)

Appendix: OPTIONS object

Description

A browser can send this preflight request to Amazon S3 to determine if it can send an actual request with the specific origin, HTTP method, and headers.

Amazon S3 supports cross-origin resource sharing (CORS) by enabling you to add a `cors` subresource on a bucket. When a browser sends this preflight request, Amazon S3 responds by evaluating the rules that are defined in the `cors` configuration.

If `cors` is not enabled on the bucket, then Amazon S3 returns a 403 Forbidden response.

For more information about CORS, go to [Enabling Cross-Origin Resource Sharing](#) in the *Amazon Simple Storage Service User Guide*.

Requests

Syntax

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
```

Request Parameters

This operation does not introduce any specific request parameters, but it may contain any request parameters that are required by the actual request.

Request Headers

Name	Description	Required
Origin	Identifies the origin of the cross-origin request to Amazon S3. For example, <code>http://www.example.com</code> . Type: String	Yes

Name	Description	Required
	Default: None	
Access-Control-Request-Method	Identifies what HTTP method will be used in the actual request. Type: String Default: None	Yes
Access-Control-Request-Headers	A comma-delimited list of HTTP headers that will be sent in the actual request. For example, to put an object with server-side encryption, this preflight request will determine if it can include the <code>x-amz-server-side-encryption</code> header with the request. Type: String Default: None	No

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

Header	Description
Access-Control-Allow-Origin	The origin you sent in your request. If the origin in your request is not allowed, Amazon S3 will not include this header in the response.

Header	Description
	Type: String
Access-Control-Max-Age	How long, in seconds, the results of the preflight request can be cached. Type: String
Access-Control-Allow-Methods	The HTTP method that was sent in the original request. If the method in the request is not allowed, Amazon S3 will not include this header in the response. Type: String
Access-Control-Allow-Headers	A comma-delimited list of HTTP headers that the browser can send in the actual request. If any of the requested headers is not allowed, Amazon S3 will not include that header in the response, nor will the response contain any of the headers with the Access-Control prefix. Type: String
Access-Control-Expose-Headers	A comma-delimited list of HTTP headers. This header provides the JavaScript client with access to these headers in the response to the actual request. Type: String

Response Elements

This implementation of the operation does not return response elements.

Examples

Example : Send a preflight OPTIONS request to a cors enabled bucket

A browser can send this preflight request to Amazon S3 to determine if it can send the actual PUT request from `http://www.example.com` origin to the Amazon S3 bucket named `examplebucket`.

Sample Request

```
OPTIONS /exampleobject HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Origin: http://www.example.com
Access-Control-Request-Method: PUT
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: 6SvaESv3VULYPLik5LL171SPPtSnBvDdGmnk1X1HfU17uS2m1DF6td6KWKNjYMXZ
x-amz-request-id: BDC4B83DF5096BBE
Date: Wed, 21 Aug 2012 23:09:55 GMT
Etag: "1f1a1af1f111111111111111c11aed1da1"
Access-Control-Allow-Origin: http://www.example.com
Access-Control-Allow-Methods: PUT
Access-Control-Expose-Headers: x-amz-request-id
Content-Length: 0
Server: AmazonS3
```

Related Resources

- [GetBucketCors](#)
- [DeleteBucketCors](#)
- [PutBucketCors](#)

Appendix: SOAP API

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

This section describes the SOAP API with respect to service, bucket, and object operations. Note that SOAP requests, both authenticated and anonymous, must be sent to Amazon S3 using SSL. Amazon S3 returns an error when you send a SOAP request over HTTP.

The latest Amazon S3 WSDL is available at docs.amazonaws.cn/2006-03-01/AmazonS3.wsdl.

Topics

- [Operations on the Service \(SOAP API\)](#)
- [Operations on Buckets \(SOAP API\)](#)
- [Operations on Objects \(SOAP API\)](#)
- [Authenticating SOAP requests](#)
- [Setting access policy with SOAP](#)
- [Common elements](#)
- [SOAP Error Responses](#)

Operations on the Service (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

This section describes operations you can perform on the Amazon S3 service.

Topics

- [ListAllMyBuckets \(SOAP API\)](#)

ListAllMyBuckets (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The `ListAllMyBuckets` operation returns a list of all buckets owned by the sender of the request.

Example

Sample Request

```
<ListAllMyBuckets xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListAllMyBuckets>
```

Sample Response

```
<ListAllMyBucketsResult xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <Owner>
    <ID>bcaf1fffd86f41161ca5fb16fd081034f</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>quotes</Name>
      <CreationDate>2006-02-03T16:45:09.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>samples</Name>
      <CreationDate>2006-02-03T16:41:58.000Z</CreationDate>
    </Bucket>
  </Buckets>
```

```
</ListAllMyBucketsResult>
```

Response Body

- **Owner :**

This provides information that Amazon S3 uses to represent your identity for purposes of authentication and access control. ID is a unique and permanent identifier for the developer who made the request. DisplayName is a human-readable name representing the developer who made the request. It is not unique, and might change over time. We recommend that you match your DisplayName to your Forum name.

- **Name :**

The name of a bucket. Note that if one of your buckets was recently deleted, the name of the deleted bucket might still be present in this list for a period of time.

- **CreationDate:**

The time that the bucket was created.

Access Control

You must authenticate with a valid Amazon Access Key ID. Anonymous requests are never allowed to list buckets, and you can only list buckets for which you are the owner.

Operations on Buckets (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

This section describes operations you can perform on Amazon S3 buckets.

Topics

- [CreateBucket \(SOAP API\)](#)
- [DeleteBucket \(SOAP API\)](#)

- [ListBucket \(SOAP API\)](#)
- [GetBucketAccessControlPolicy \(SOAP API\)](#)
- [SetBucketAccessControlPolicy \(SOAP API\)](#)
- [GetBucketLoggingStatus \(SOAP API\)](#)
- [SetBucketLoggingStatus \(SOAP API\)](#)

CreateBucket (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The CreateBucket operation creates a bucket. Not every string is an acceptable bucket name. For information on bucket naming restrictions, see [Working with Amazon S3 Buckets](#).

Note

To determine whether a bucket name exists, use ListBucket and set MaxKeys to 0. A NoSuchBucket response indicates that the bucket is available, an AccessDenied response indicates that someone else owns the bucket, and a Success response indicates that you own the bucket or have permission to access it.

Example Create a bucket named "quotes"

Sample Request

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

Sample Response

```
<CreateBucketResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <CreateBucketResponse>
    <Bucket>quotes</Bucket>
  </CreateBucketResponse>
</CreateBucketResponse>
```

Elements

- **Bucket**: The name of the bucket you are trying to create.
- **AccessControlList**: The access control list for the new bucket. This element is optional. If not provided, the bucket is created with an access policy that give the requester FULL_CONTROL access.

Access Control

You must authenticate with a valid Amazon Access Key ID. Anonymous requests are never allowed to create buckets.

Related Resources

- [ListBucket \(SOAP API\)](#)

DeleteBucket (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The DeleteBucket operation deletes a bucket. All objects in the bucket must be deleted before the bucket itself can be deleted.

Example

This example deletes the "quotes" bucket.

Sample Request

```
<DeleteBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteBucket>
```

Sample Response

```
<DeleteBucketResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <DeleteBucketResponse>
    <Code>204</Code>
    <Description>No Content</Description>
  </DeleteBucketResponse>
</DeleteBucketResponse>
```

Elements

- **Bucket**: The name of the bucket you want to delete.

Access Control

Only the owner of a bucket is allowed to delete it, regardless the access control policy on the bucket.

ListBucket (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The `ListBucket` operation returns information about some of the items in the bucket.

For a general introduction to the list operation, see the [Listing Object Keys](#).

Requests

This example lists up to 1000 keys in the "quotes" bucket that have the prefix "notes."

Syntax

```
<ListBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Prefix>notes</Prefix>
  <Delimiter></Delimiter>
  <MaxKeys>1000</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Parameters

Name	Description	Required
prefix	Limits the response to keys which begin with the indicated prefix. You can use prefixes to separate a bucket into different sets of keys in a way similar to how a file system uses folders. Type: String Default: None	No
marker	Indicates where in the bucket to begin listing. The list will only include keys that occur lexicographically after marker. This is convenient for pagination: To get the next page of results use the last key of the current page as the marker. Type: String Default: None	No
max-keys	The maximum number of keys you'd like to see in the response body. The server might return fewer than this many keys, but will not return more.	No

Name	Description	Required
	Type: String Default: None	
delimiter	Causes keys that contain the same string between the prefix and the first occurrence of the delimiter to be rolled up into a single result element in the CommonPrefixes collection. These rolled-up keys are not returned elsewhere in the response. Type: String Default: None	No

Success Response

This response assumes the bucket contains the following keys:

```
notes/todos.txt
notes/2005-05-23/customer_mtg_notes.txt
notes/2005-05-23/phone_notes.txt
notes/2005-05-28/sales_notes.txt
```

Syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>backups</Name>
  <Prefix>notes</Prefix>
  <MaxKeys>1000</MaxKeys>
  <Delimiter></Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>notes/todos.txt</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <Size>5126</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
```

```
<DisplayName>webfile</DisplayName>
</Owner>
<StorageClass>STANDARD</StorageClass>
</Contents>
<CommonPrefixes>
  <Prefix>notes/2005-05-23/</Prefix>
</CommonPrefixes>
<CommonPrefixes>
  <Prefix>notes/2005-05-28/</Prefix>
</CommonPrefixes>
</ListBucketResult>
```

As you can see, many of the fields in the response echo the request parameters. `IsTruncated`, `Contents`, and `CommonPrefixes` are the only response elements that can contain new information.

Response Elements

Name	Description
Contents	Metadata about each object returned. Type: XML metadata Ancestor: ListBucketResult
CommonPrefixes	A response can contain <code>CommonPrefixes</code> only if you specify a <code>delimiter</code> . When you do, <code>CommonPrefixes</code> contains all (if there are any) keys between <code>Prefix</code> and the next occurrence of the string specified by <code>delimiter</code> . In effect, <code>CommonPrefixes</code> lists keys that act like subdirectories in the directory specified by <code>Prefix</code> . For example, if <code>prefix</code> is <code>notes/</code> and <code>delimiter</code> is a slash (/), in <code>notes/summer/july</code> , the common prefix is <code>notes/summer/</code> . Type: String Ancestor: ListBucketResult
Delimiter	Causes keys that contain the same string between the prefix and the first occurrence of the delimiter to be rolled up into a single result element in the

Name	Description
	CommonPrefixes collection. These rolled-up keys are not returned elsewhere in the response. Type: String Ancestor: ListBucketResult
IsTruncated	Specifies whether (true) or not (false) all of the results were returned. All of the results may not be returned if the number of results exceeds that specified by MaxKeys. Type: String Ancestor: boolean
Marker	Indicates where in the bucket to begin listing. Type: String Ancestor: ListBucketResult
MaxKeys	The maximum number of keys returned in the response body. Type: String Ancestor: ListBucketResult
Name	Name of the bucket. Type: String Ancestor: ListBucketResult
Prefix	Keys that begin with the indicated prefix. Type: String Ancestor: ListBucketResult

Response Body

For information about the list response, see [Listing Keys Response](#).

Access Control

To list the keys of a bucket you need to have been granted READ access on the bucket.

GetBucketAccessControlPolicy (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The `GetBucketAccessControlPolicy` operation fetches the access control policy for a bucket.

Example

This example retrieves the access control policy for the "quotes" bucket.

Sample Request

```
<GetBucketAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetBucketAccessControlPolicy>
```

Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd2441bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
```



```
<ID>a9a7b886d6f41bf9b1c61be666e9</ID>
  <DisplayName>chriscustomer</DisplayName>
</Grantee>
<Permission>FULL_CONTROL</Permission>
</Grant>
<Grant>
  <Grantee xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
</AccessControlList>
<AccessControlPolicy>
```

Response Body

The response contains the access control policy for the bucket. For an explanation of this response, see [SOAP Access Policy](#).

Access Control

You must have READ_ACP rights to the bucket in order to retrieve the access control policy for a bucket.

SetBucketAccessControlPolicy (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The SetBucketAccessControlPolicy operation sets the Access Control Policy for an existing bucket. If successful, the previous Access Control Policy for the bucket is entirely replaced with the specified Access Control Policy.

Example

Give the specified user (usually the owner) FULL_CONTROL access to the "quotes" bucket.

Sample Request

```
<SetBucketAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b8863000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</SetBucketAccessControlPolicy >
```

Sample Response

```
<GetBucketAccessControlPolicyResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetBucketAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </GetBucketAccessControlPolicyResponse>
</GetBucketAccessControlPolicyResponse>
```

Access Control

You must have WRITE_ACP rights to the bucket in order to set the access control policy for a bucket.

GetBucketLoggingStatus (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The `GetBucketLoggingStatus` retrieves the logging status for an existing bucket.

For a general introduction to this feature, see [Server Logs](#).

Example

Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
    <soap:Body>
      <GetBucketLoggingStatus xmlns="https://doc.s3.amazonaws.com/2006-03-01">
        <Bucket>mybucket</Bucket>
        <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
        <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
        <Signature>YOUR_SIGNATURE_HERE</Signature>
      </GetBucketLoggingStatus>
    </soap:Body>
  </soap:Envelope>
```

Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" >
    <soapenv:Header>
    </soapenv:Header>
    <soapenv:Body>
      <GetBucketLoggingStatusResponse xmlns="https://s3.amazonaws.com/
doc/2006-03-01">
        <GetBucketLoggingStatusResponse>
          <LoggingEnabled>
            <TargetBucket>mylogs</TargetBucket>
            <TargetPrefix>mybucket-access_log</TargetPrefix>
          </LoggingEnabled>
        </GetBucketLoggingStatusResponse>
      </GetBucketLoggingStatusResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

Access Control

Only the owner of a bucket is permitted to invoke this operation.

SetBucketLoggingStatus (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The SetBucketLoggingStatus operation updates the logging status for an existing bucket.

For a general introduction to this feature, see [Server Logs](#).

Example

This sample request enables server access logging for the 'mybucket' bucket, and configures the logs to be delivered to 'mylogs' under prefix 'access_log-'

Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
    <soap:Body>
      <SetBucketLoggingStatus xmlns="https://doc.s3.amazonaws.com/2006-03-01">
        <Bucket>myBucket</Bucket>
        <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
        <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
        <Signature>YOUR_SIGNATURE_HERE</Signature>
        <BucketLoggingStatus>
          <LoggingEnabled>
            <TargetBucket>mylogs</TargetBucket>
            <TargetPrefix>mybucket-access_log-</TargetPrefix>
          </LoggingEnabled>
        </BucketLoggingStatus>
      </SetBucketLoggingStatus>
    </soap:Body>
  </soap:Envelope>
```

```
</soap:Body>
:</soap:Envelope>
```

Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" >
    <soapenv:Header>
    </soapenv:Header>
    <soapenv:Body>
      <SetBucketLoggingStatusResponse xmlns="https://s3.amazonaws.com/
doc/2006-03-01"/>
    </soapenv:Body>
  </soapenv:Envelope>
```

Access Control

Only the owner of a bucket is permitted to invoke this operation.

Operations on Objects (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

This section describes operations you can perform on Amazon S3 objects.

Topics

- [PutObjectInline \(SOAP API\)](#)
- [PutObject \(SOAP API\)](#)
- [CopyObject \(SOAP API\)](#)

- [GetObject \(SOAP API\)](#)
- [GetObjectExtended \(SOAP API\)](#)
- [DeleteObject \(SOAP API\)](#)
- [GetObjectAccessControlPolicy \(SOAP API\)](#)
- [SetObjectAccessControlPolicy \(SOAP API\)](#)

PutObjectInline (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The `PutObjectInline` operation adds an object to a bucket. The data for the object is provided in the body of the SOAP message.

If an object already exists in a bucket, the new object will overwrite it because Amazon S3 stores the last write request. However, Amazon S3 is a distributed system. If Amazon S3 receives multiple write requests for the same object nearly simultaneously, all of the objects might be stored, even though only one wins in the end. Amazon S3 does not provide object locking; if you need this, make sure to build it into your application layer.

To ensure an object is not corrupted over the network, you can calculate the MD5 of an object, PUT it to Amazon S3, and compare the returned Etag to the calculated MD5 value.

`PutObjectInline` is not suitable for use with large objects. The system limits this operation to working with objects 1MB or smaller. `PutObjectInline` will fail with the `InlineDataTooLargeError` status code if the `Data` parameter encodes an object larger than 1MB. To upload large objects, consider using the non-inline `PutObject` API, or the REST API instead.

Example

This example writes some text and metadata into the "Nelson" object in the "quotes" bucket, give a user (usually the owner) `FULL_CONTROL` access to the object, and make the object readable by anonymous parties.

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fde241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2006-01-01T12:00:00.000Z</lastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

Elements

- **Bucket**: The bucket in which to add the object.
- **Key**: The key to assign to the object.
- **Metadata**: You can provide name-value metadata pairs in the metadata element. These will be stored with the object.
- **Data**: The base 64 encoded form of the data.
- **ContentLength**: The length of the data in bytes.
- **AccessControlList**: An Access Control List for the resource. This element is optional. If omitted, the requester is given FULL_CONTROL access to the object. If the object already exists, the preexisting access control policy is replaced.

Responses

- **ETag**: The entity tag is an MD5 hash of the object that you can use to do conditional fetches of the object using `GetObjectExtended`. The ETag only reflects changes to the contents of an object, not its metadata.
- **LastModified**: The Amazon S3 timestamp for the saved object.

Access Control

You must have WRITE access to the bucket in order to put objects into the bucket.

Related Resources

- [PutObject \(SOAP API\)](#)
- [CopyObject \(SOAP API\)](#)

PutObject (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The `PutObject` operation adds an object to a bucket. The data for the object is attached as a DIME attachment.

To ensure an object is not corrupted over the network, you can calculate the MD5 of an object, PUT it to Amazon S3, and compare the returned Etag to the calculated MD5 value.

If an object already exists in a bucket, the new object will overwrite it because Amazon S3 stores the last write request. However, Amazon S3 is a distributed system. If Amazon S3 receives multiple write requests for the same object nearly simultaneously, all of the objects might be stored, even though only one wins in the end. Amazon S3 does not provide object locking; if you need this, make sure to build it into your application layer.

Example

This example puts some data and metadata in the "Nelson" object of the "quotes" bucket, give a user (usually the owner) `FULL_CONTROL` access to the object, and make the object readable by anonymous parties. In this sample, the actual attachment is not shown.

Sample Request

```
<PutObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
```

```

<ContentLength>5</ContentLength>
<AccessControlList>
  <Grant>
    <Grantee xsi:type="CanonicalUser">
      <ID>a9a7b886d6241bf9b1c61be666e9</ID>
      <DisplayName>chriscustomer</DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
  <Grant>
    <Grantee xsi:type="Group">
      <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
    </Grantee>
    <Permission>READ</Permission>
  </Grant>
</AccessControlList>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2007-05-11T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObject>

```

Sample Response

```

<PutObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2006-03-01T12:00:00.183Z</LastModified>
  </PutObjectResponse>
</PutObjectResponse>

```

Elements

- **Bucket**: The bucket in which to add the object.
- **Key**: The key to assign to the object.
- **Metadata**: You can provide name-value metadata pairs in the metadata element. These will be stored with the object.
- **ContentLength**: The length of the data in bytes.
- **AccessControlList**: An Access Control List for the resource. This element is optional. If omitted, the requester is given FULL_CONTROL access to the object. If the object already exists, the preexisting Access Control Policy is replaced.

Responses

- **ETag**: The entity tag is an MD5 hash of the object that you can use to do conditional fetches of the object using `GetObjectExtended`. The ETag only reflects changes to the contents of an object, not its metadata.
- **LastModified**: The Amazon S3 timestamp for the saved object.

Access Control

To put objects into a bucket, you must have `WRITE` access to the bucket.

Related Resources

- [CopyObject \(SOAP API\)](#)

CopyObject (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

Description

The `CopyObject` operation creates a copy of an object when you specify the key and bucket of a source object and the key and bucket of a target destination.

When copying an object, you can preserve all metadata (default) or specify new metadata. However, the ACL is not preserved and is set to `private` for the user making the request. To override the default ACL setting, specify a new ACL when generating a copy request. For more information, see [Using ACLs](#).

All copy requests must be authenticated. Additionally, you must have *read* access to the source object and *write* access to the destination bucket. For more information, see [Using Auth Access](#).

To only copy an object under certain conditions, such as whether the Etag matches or whether the object was modified before or after a specified date, use the request parameters

CopySourceIfUnmodifiedSince, CopyIfUnmodifiedSince, CopySourceIfMatch, or CopySourceIfNoneMatch.

Note

You might need to configure the SOAP stack socket timeout for copying large objects.

Request Syntax

```
<CopyObject xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <SourceBucket>source_bucket</SourceBucket>
  <SourceObject>source_object</SourceObject>
  <DestinationBucket>destination_bucket</DestinationBucket>
  <DestinationObject>destination_object</DestinationObject>
  <MetadataDirective>{REPLACE | COPY}</MetadataDirective>
  <Metadata>
    <Name>metadata_name</Name>
    <Value>metadata_value</Value>
  </Metadata>
  ...
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="user_type">
        <ID>user_id</ID>
        <DisplayName>display_name</DisplayName>
      </Grantee>
      <Permission>permission</Permission>
    </Grant>
    ...
  </AccessControlList>
  <CopySourceIfMatch>etag</CopySourceIfMatch>
  <CopySourceIfNoneMatch>etag</CopySourceIfNoneMatch>
  <CopySourceIfModifiedSince>date_time</CopySourceIfModifiedSince>
  <CopySourceIfUnmodifiedSince>date_time</CopySourceIfUnmodifiedSince>
  <AWSAccessKeyId>AWSAccessKeyId</AWSAccessKeyId>
  <Timestamp>TimeStamp</Timestamp>
  <Signature>Signature</Signature>
</CopyObject>
```

Request Parameters

Name	Description	Required
SourceBucket	The name of the source bucket. Type: String Default: None Constraints: A valid source bucket.	Yes
SourceKey	The key name of the source object. Type: String Default: None Constraints: The key for a valid source object to which you have READ access.	Yes
DestinationBucket	The name of the destination bucket. Type: String Default: None Constraints: You must have WRITE access to the destination bucket.	Yes
DestinationKey	The key of the destination object. Type: String Default: None Constraints: You must have WRITE access to the destination bucket.	Yes
MetadataDirective	Specifies whether the metadata is copied from the source object or replaced with metadata provided in the request.	No

Name	Description	Required
	Type: String Default: COPY Valid values: COPY REPLACE Constraints: Values other than COPY or REPLACE will result in an immediate error. You cannot copy an object to itself unless the MetadataDirective header is specified and its value set to REPLACE.	
Metadata	Specifies metadata name-value pairs to set for the object.If MetadataDirective is set to COPY, all metadata is ignored. Type: String Default: None Constraints: None.	No
AccessControlList	Grants access to users by e-mail addresses or canonical user ID. Type: String Default: None Constraints: None	No

Name	Description	Required
CopySourceIfMatch	Copies the object if its entity tag (ETag) matches the specified tag; otherwise return a PreconditionFailed. Type: String Default: None Constraints: None. If the Etag does not match, the object is not copied.	No
CopySourceIfNoneMatch	Copies the object if its entity tag (ETag) is different than the specified Etag; otherwise returns an error. Type: String Default: None Constraints: None.	No
CopySourceIfUnmodifiedSince	Copies the object if it hasn't been modified since the specified time; otherwise returns a PreconditionFailed. Type: dateTime Default: None	No
CopySourceIfModifiedSince	Copies the object if it has been modified since the specified time; otherwise returns an error. Type: dateTime Default: None	No

Response Syntax

```
<CopyObjectResponse xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"etag"</ETag>
    <LastModified>timestamp</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Response Elements

Following is a list of response elements.

 **Note**

The SOAP API does not return extra whitespace. Extra whitespace is only returned by the REST API.

Name	Description
Etag	Returns the etag of the new object. The ETag only reflects changes to the contents of an object, not its metadata. Type: String Ancestor: CopyObjectResult
LastModified	Returns the date the object was last modified. Type: String Ancestor: CopyObjectResult

For information about general response elements, see [Using REST Error Response Headers](#).

Special Errors

There are no special errors for this operation. For information about general Amazon S3 errors, see [List of error codes](#).

Examples

This example copies the `flotsam` object from the `pacific` bucket to the `jetsam` object of the `atlantic` bucket, preserving its metadata.

Sample Request

```
<CopyObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <SourceBucket>pacific</SourceBucket>
  <SourceObject>flotsam</SourceObject>
  <DestinationBucket>atlantic</DestinationBucket>
  <DestinationObject>jetsam</DestinationObject>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2008-02-18T13:54:10.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbq7RrtSFmw=</Signature>
</CopyObject>
```

Sample Response

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

This example copies the `"tweedledee"` object from the `wonderland` bucket to the `"tweedledum"` object of the `wonderland` bucket, replacing its metadata.

Sample Request

```
<CopyObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <SourceBucket>wonderland</SourceBucket>
  <SourceObject>tweedledee</SourceObject>
  <DestinationBucket>wonderland</DestinationBucket>
  <DestinationObject>tweedledum</DestinationObject>
  <MetadataDirective>REPLACE</MetadataDirective>
</CopyObject>
```

```
<Metadata>
  <Name>Content-Type</Name>
  <Value>text/plain</Value>
</Metadata>
<Metadata>
  <Name>relationship</Name>
  <Value>twins</Value>
</Metadata>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2008-02-18T13:54:10.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbq7RrtSFmw=</Signature>
</CopyObject>
```

Sample Response

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Related Resources

- [PutObject \(SOAP API\)](#)
- [PutObjectInline \(SOAP API\)](#)

GetObject (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The `GetObject` operation returns the current version of an object. If you try to `GetObject` an object that has a delete marker as its current version, S3 returns a 404 error. You cannot use the SOAP API to retrieve a specified version of an object. To do that, use the REST API. For more information, see [Versioning](#). For more options, use the [GetObjectExtended \(SOAP API\)](#) operation.

Note

Object key names with the value "soap" aren't supported for [virtual-hosted-style requests](#). For object key name values where "soap" is used, a [path-style URL](#) must be used instead.

Example

This example gets the "Nelson" object from the "quotes" bucket.

Sample Request

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Sample Response

```
<GetObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>aGEtaGE=</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

```
</GetObjectResponse>  
</GetObjectResponse>
```

Elements

- **Bucket**: The bucket from which to retrieve the object.
- **Key**: The key that identifies the object.
- **GetMetadata**: The metadata is returned with the object if this is true.
- **GetData**: The object data is returned if this is true.
- **InlineData**: If this is true, then the data is returned, base 64-encoded, as part of the SOAP body of the response. If false, then the data is returned as a SOAP attachment. The `InlineData` option is not suitable for use with large objects. The system limits this operation to working with 1MB of data or less. A `GetObject` request with the `InlineData` flag set will fail with the `InlineDataTooLargeError` status code if the resulting `Data` parameter would have encoded more than 1MB. To download large objects, consider calling `GetObject` without setting the `InlineData` flag, or use the REST API instead.

Returned Elements

- **Metadata**: The name-value paired metadata stored with the object.
- **Data**: If `InlineData` was true in the request, this contains the base 64 encoded object data.
- **LastModified**: The time that the object was stored in Amazon S3.
- **ETag**: The object's entity tag. This is a hash of the object that can be used to do conditional gets. The ETag only reflects changes to the contents of an object, not its metadata.

Access Control

You can read an object only if you have been granted READ access to the object.

SOAP Chunked and Resumable Downloads

To provide GET flexibility, Amazon S3 supports chunked and resumable downloads.

Select from the following:

- For large object downloads, you might want to break them into smaller chunks. For more information, see [Range GETs](#)

- For GET operations that fail, you can design your application to download the remainder instead of the entire file. For more information, see [REST GET Error Recovery](#)

Range GETs

For some clients, you might want to break large downloads into smaller downloads. To break a GET into smaller units, use Range.

Before you can break a GET into smaller units, you must determine its size. For example, the following request gets the size of the bigfile object.

```
<ListBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Prefix>bigfile</Prefix>
  <MaxKeys>1</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Amazon S3 returns the following response.

```
<ListBucketResult xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <Name>quotes</Name>
  <Prefix>N</Prefix>
  <MaxKeys>1</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>bigfile</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <Size>2023276</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>bcaf1ffd86f41161ca5fb16fd081034f</ID>
      <DisplayName>bigfile</DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

Following is a request that downloads the first megabyte from the bigfile object.

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <ByteRangeStart>0</ByteRangeStart>
  <ByteRangeEnd>1048576</ByteRangeEnd>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Amazon S3 returns the first megabyte of the file and the Etag of the file.

```
<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>--first megabyte of bigfile--</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

To ensure the file did not change since the previous portion was downloaded, specify the `IfMatch` element. Although the `IfMatch` element is not required, it is recommended for content that is likely to change.

The following is a request that gets the remainder of the file, using the `IfMatch` request header.

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
```

```

<Bucket>bigbucket</Bucket>
<Key>bigfile</Key>
<GetMetadata>true</GetMetadata>
<GetData>true</GetData>
<InlineData>true</InlineData>
<ByteRangeStart>10485761</ByteRangeStart>
<ByteRangeEnd>2023276</ByteRangeEnd>
<IfMatch>"828ef3fdfa96f00ad9f27c383fc9ac7f"</IfMatch>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>

```

Amazon S3 returns the following response and the remainder of the file.

```

<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>>Muntz</Value>
    </Metadata>
    <Data>--remainder of bigfile--</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>

```

Versioned GetObject

The following request returns the specified version of the object in the bucket.

```

<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <GetMetadata>true</GetMetadata>

```

```
<GetData>true</GetData>
<InlineData>true</InlineData>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Sample Response

```
<GetObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>aGEtaGE=</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

REST GET Error Recovery

If an object GET fails, you can get the rest of the file by specifying the range to download. To do so, you must get the size of the object using `ListBucket` and perform a range GET on the remainder of the file. For more information, see [GetObjectExtended \(SOAP API\)](#).

Related Resources

[Operations on Objects \(SOAP API\)](#)

GetObjectExtended (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

GetObjectExtended is exactly like [GetObject \(SOAP API\)](#), except that it supports the following additional elements that can be used to accomplish much of the same functionality provided by HTTP GET headers (go to <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>).

GetObjectExtended supports the following elements in addition to those supported by GetObject:

- **ByteRangeStart**, **ByteRangeEnd**: These elements specify that only a portion of the object data should be retrieved. They follow the behavior of the HTTP byte ranges (go to <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35>).
- **IfModifiedSince**: Return the object only if the object's timestamp is later than the specified timestamp. (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.25>)
- **IfUnmodifiedSince**: Return the object only if the object's timestamp is earlier than or equal to the specified timestamp. (go to <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.28>)
- **IfMatch**: Return the object only if its ETag matches the supplied tag(s). (go to <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.24>)
- **IfNoneMatch**: Return the object only if its ETag does not match the supplied tag(s). (go to <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.26>)
- **ReturnCompleteObjectOnConditionFailure**: **ReturnCompleteObjectOnConditionFailure**: If true, then if the request includes a range element and one or both of **IfUnmodifiedSince**/**IfMatch** elements, and the condition fails, return the entire object rather than a fault. This enables the **If-Range** functionality (go to <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.27>).

DeleteObject (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The DeleteObject operation removes the specified object from Amazon S3. Once deleted, there is no method to restore or undelete an object.

Note

If you delete an object that does not exist, Amazon S3 will return a success (not an error message).

Example

This example deletes the "Nelson" object from the "quotes" bucket.

Sample Request

```
<DeleteObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteObject>
```

Sample Response

```
<DeleteObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <DeleteObjectResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </DeleteObjectResponse>
</DeleteObjectResponse>
```

Elements

- **Bucket**: The bucket that holds the object.
- **Key**: The key that identifies the object.

Access Control

You can delete an object only if you have **WRITE** access to the bucket, regardless of who owns the object or what rights are granted to it.

GetObjectAccessControlPolicy (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The `GetObjectAccessControlPolicy` operation fetches the access control policy for an object.

Example

This example retrieves the access control policy for the "Nelson" object from the "quotes" bucket.

Sample Request

```
<GetObjectAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObjectAccessControlPolicy>
```

Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd24a541bf9b1c61be666e9</ID>
```

```
<DisplayName>chriscustomer</DisplayName>
</Owner>
<AccessControlList>
  <Grant>
    <Grantee xsi:type="CanonicalUser">
      <ID>a9a7b841bf9b1c61be666e9</ID>
      <DisplayName>chriscustomer</DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
  <Grant>
    <Grantee xsi:type="Group">
      <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
    </Grantee>
    <Permission>READ</Permission>
  </Grant>
</AccessControlList>
</AccessControlPolicy>
```

Response Body

The response contains the access control policy for the bucket. For an explanation of this response, [SOAP Access Policy](#).

Access Control

You must have READ_ACP rights to the object in order to retrieve the access control policy for an object.

SetObjectAccessControlPolicy (SOAP API)

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

The `SetObjectAccessControlPolicy` operation sets the access control policy for an existing object. If successful, the previous access control policy for the object is entirely replaced with the specified access control policy.

Example

This example gives the specified user (usually the owner) FULL_CONTROL access to the "Nelson" object from the "quotes" bucket.

Sample Request

```
<SetObjectAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE= </Signature>
</SetObjectAccessControlPolicy>
```

Sample Response

```
<SetObjectAccessControlPolicyResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <SetObjectAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </SetObjectAccessControlPolicyResponse>
</SetObjectAccessControlPolicyResponse>
```

Key

Access Control

You must have WRITE_ACP rights to the object in order to set the access control policy for a bucket.

Authenticating SOAP requests

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

Every non-anonymous request must contain authentication information to establish the identity of the principal making the request. In SOAP, the authentication information is put into the following elements of the SOAP request:

- Your Amazon Access Key ID

Note

When making authenticated SOAP requests, temporary security credentials are not supported. For more information about types of credentials, see [Making requests](#).

- **Timestamp:** This must be a dateTime (go to <http://www.w3.org/TR/xmlschema-2/#dateTime>) in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as 2009-01-01T12:00:00.000Z. Authorization will fail if this timestamp is more than 15 minutes away from the clock on Amazon S3 servers.
- **Signature:** The RFC 2104 HMAC-SHA1 digest (go to <http://www.ietf.org/rfc/rfc2104.txt>) of the concatenation of "AmazonS3" + OPERATION + Timestamp, using your Amazon Secret Access Key as the key. For example, in the following CreateBucket sample request, the signature element would contain the HMAC-SHA1 digest of the value "AmazonS3CreateBucket2009-01-01T12:00:00.000Z":

For example, in the following CreateBucket sample request, the signature element would contain the HMAC-SHA1 digest of the value "AmazonS3CreateBucket2009-01-01T12:00:00.000Z":

Example

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Acl>private</Acl>
```

```
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2009-01-01T12:00:00.000Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

Note

SOAP requests, both authenticated and anonymous, must be sent to Amazon S3 using SSL. Amazon S3 returns an error when you send a SOAP request over HTTP.

Important

Due to different interpretations regarding how extra time precision should be dropped, .NET users should take care not to send Amazon S3 overly specific time stamps. This can be accomplished by manually constructing `DateTime` objects with only millisecond precision.

Setting access policy with SOAP

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

Access control can be set at the time a bucket or object is written by including the "AccessControlList" element with the request to `CreateBucket`, `PutObjectInline`, or `PutObject`. The `AccessControlList` element is described in [Identity and Access Management for Amazon S3](#). If no access control list is specified with these operations, the resource is created with a default access policy that gives the requester `FULL_CONTROL` access (this is the case even if the request is a `PutObjectInline` or `PutObject` request for an object that already exists).

Following is a request that writes data to an object, makes the object readable by anonymous principals, and gives the specified user `FULL_CONTROL` rights to the bucket (Most developers will want to give themselves `FULL_CONTROL` access to their own bucket).

Example

Following is a request that writes data to an object and makes the object readable by anonymous principals.

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
```



```
</PutObjectInlineResponse>
```

The access control policy can be read or set for an existing bucket or object using the `GetBucketAccessControlPolicy`, `GetObjectAccessControlPolicy`, `SetBucketAccessControlPolicy`, and `SetObjectAccessControlPolicy` methods. For more information, see the detailed explanation of these methods.

Common elements

You can include the following authorization-related elements with any SOAP request:

- **AWSSignatureVersion**: The Amazon Access Key ID of the requester
- **Timestamp**: The current time on your system
- **Signature**: The signature for the request

SOAP Error Responses

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

In SOAP, an error result is returned to the client as a SOAP fault, with the HTTP response code 500. If you do not receive a SOAP fault, then your request was successful. The Amazon S3 SOAP fault code is comprised of a standard SOAP 1.1 fault code (either "Server" or "Client") concatenated with the Amazon S3-specific error code. For example: "Server.InternalError" or "Client.NoSuchBucket". The SOAP fault string element contains a generic, human readable error message in English. Finally, the SOAP fault detail element contains miscellaneous information relevant to the error.

For example, if you attempt to delete the object "Fred", which does not exist, the body of the SOAP response contains a "NoSuchKey" SOAP fault.

The following example shows a sample SOAP error response.

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.NoSuchKey</Faultcode>
```

```
<Faultstring>The specified key does not exist.</Faultstring>
<Detail>
  <Key>Fred</Key>
</Detail>
</soapenv:Fault>
</soapenv:Body>
```

The following table explains the SOAP error response elements

Name	Description
Detail	Container for the key involved in the error Type: Container Ancestor: Body.Fault
Fault	Container for error information. Type: Container Ancestor: Body
Faultcode	The fault code is a string that uniquely identifies an error condition. It is meant to be read and understood by programs that detect and handle errors by type. For more information, see List of Error Codes. Type: String Ancestor: Body.Fault
Faultstring	The fault string contains a generic description of the error condition in English. It is intended for a human audience. Simple programs display the message directly to the end user if they encounter an error condition they don't know how or don't care to handle. Sophisticated programs with more exhaustive error handling and proper internationalization are more likely to ignore the fault string. Type: String Ancestor: Body.Fault

Name	Description
Key	Identifies the key involved in the error
	Type: String
	Ancestor: Body.Fault

Appendix: Authenticating requests (Amazon signature version 2)

Important

This section describes how to authenticate requests using Amazon Signature Version 2. Signature Version 2 is being turned off (deprecated), Amazon S3 will only accept API requests that are signed using Signature Version 4. For more information, see [Amazon Signature Version 2 turned off \(deprecated\) for Amazon S3](#). Signature Version 4 is supported in all Amazon Web Services Regions, and it is the only version that is supported for new Regions. For more information, see [Authenticating Requests \(Amazon Signature Version 4\)](#) in the *Amazon Simple Storage Service API Reference*. Amazon S3 offers you the ability to identify what API signature version was used to sign a request. It is important to identify if any of your workflows are utilizing Signature Version 2 signing and upgrading them to use Signature Version 4 to prevent impact to your business.

- If you are using CloudTrail event logs(recommended option), please see [Identifying Amazon S3 Signature Version 2 requests by using CloudTrail](#) on how to query and identify such requests.
- If you are using the Amazon S3 Server Access logs, see [Using Amazon S3 server access logs to identify requests](#)

Topics

- [Authenticating requests using the REST API \(Amazon signature version 2\)](#)
- [Signing and authenticating REST requests \(Amazon signature version 2\)](#)
- [Browser-based uploads using POST \(Amazon signature version 2\)](#)

Authenticating requests using the REST API (Amazon signature version 2)

When accessing Amazon S3 using REST, you must provide the following items in your request so the request can be authenticated:

Request elements

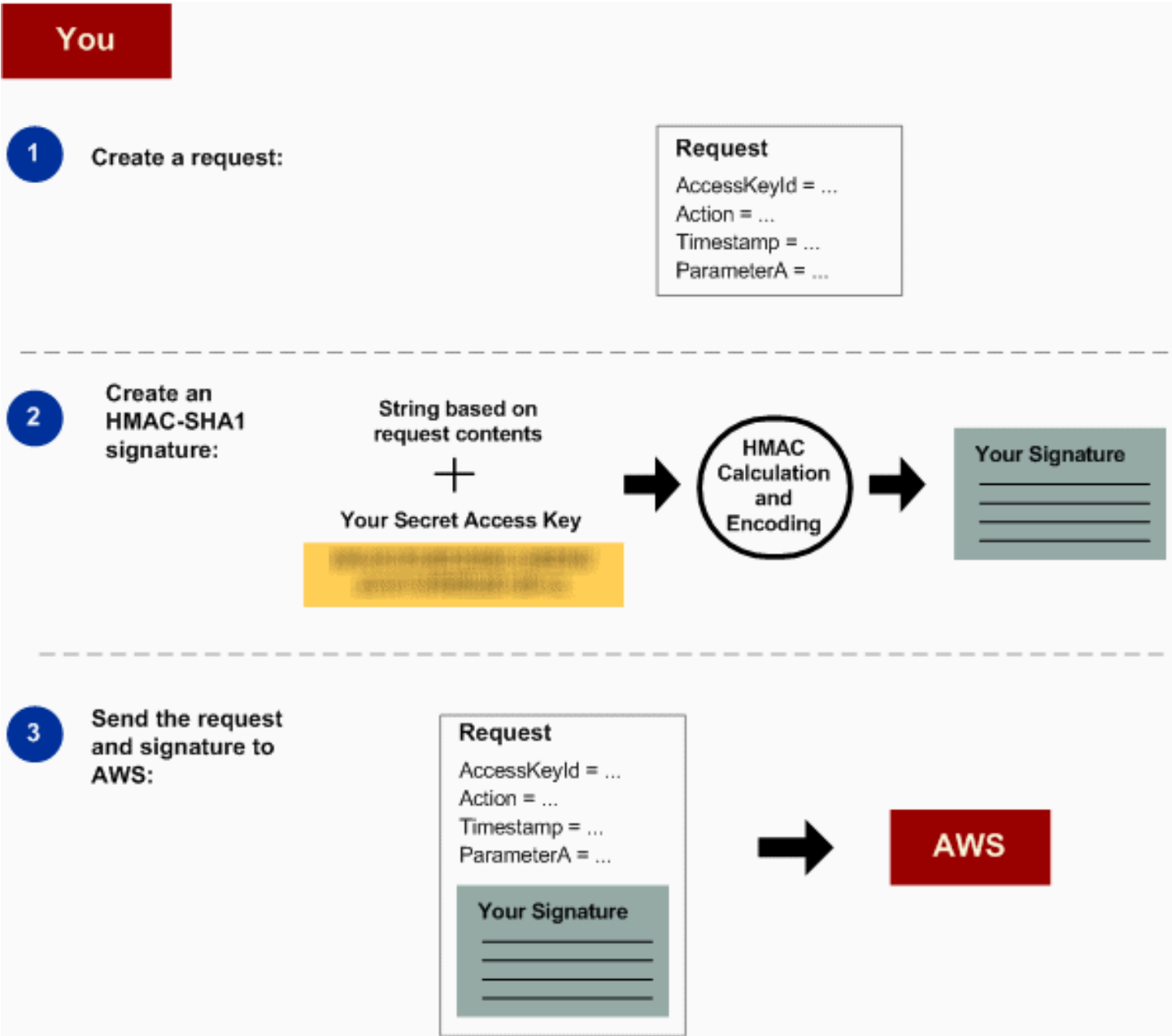
- **Amazon access key Id** – Each request must contain the access key ID of the identity you are using to send your request.
- **Signature** – Each request must contain a valid request signature, or the request is rejected.

A request signature is calculated using your secret access key, which is a shared secret known only to you and Amazon.

- **Time stamp** – Each request must contain the date and time the request was created, represented as a string in UTC.
- **Date** – Each request must contain the time stamp of the request.

Depending on the API action you're using, you can provide an expiration date and time for the request instead of or in addition to the time stamp. See the authentication topic for the particular action to determine what it requires.

Following are the general steps for authenticating requests to Amazon S3. It is assumed you have the necessary security credentials, access key ID and secret access key.

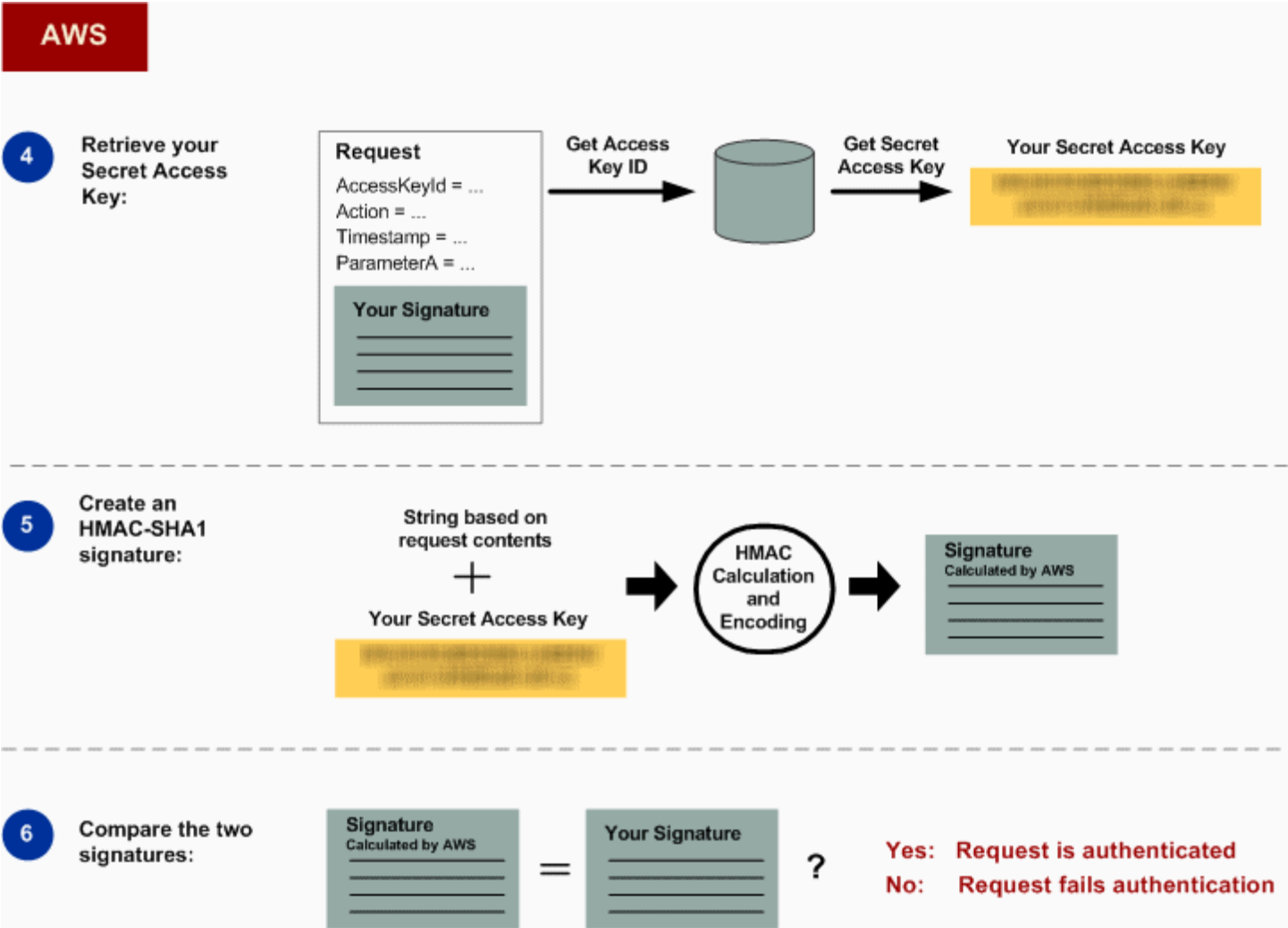


- 1

Construct a request to Amazon.
- 2

Calculate the signature using your secret access key.
- 3

Send the request to Amazon S3. Include your access key ID and the signature in your request. Amazon S3 performs the next three steps.



Detailed authentication information

For detailed information about REST authentication, see [Signing and authenticating REST requests \(Amazon signature version 2\)](#).

Signing and authenticating REST requests (Amazon signature version 2)

Topics

- [Using temporary security credentials](#)
- [The authentication header](#)
- [Request canonicalization for signing](#)
- [Constructing the CanonicalizedResource element](#)
- [Constructing the CanonicalizedAmzHeaders element](#)
- [Positional versus named HTTP header StringToSign elements](#)
- [Time stamp requirement](#)
- [Authentication examples](#)
- [REST request signing problems](#)
- [Query string request authentication alternative](#)

Note

This topic explains authenticating requests using Signature Version 2. Amazon S3 now supports the latest Signature Version 4. This latest signature version is supported in all regions and any new regions after January 30, 2014 will support only Signature Version 4. For more information, go to [Authenticating Requests \(Amazon Signature Version 4\)](#) in the *Amazon Simple Storage Service API Reference*.

Authentication is the process of proving your identity to the system. Identity is an important factor in Amazon S3 access control decisions. Requests are allowed or denied in part based on the identity of the requester. For example, the right to create buckets is reserved for registered developers and (by default) the right to create objects in a bucket is reserved for the owner of the bucket in question. As a developer, you'll be making requests that invoke these privileges, so you'll need to prove your identity to the system by authenticating your requests. This section shows you how.

Note

The content in this section does not apply to HTTP POST. For more information, see [Browser-based uploads using POST \(Amazon signature version 2\)](#).

The Amazon S3 REST API uses a custom HTTP scheme based on a keyed-HMAC (Hash Message Authentication Code) for authentication. To authenticate a request, you first concatenate selected elements of the request to form a string. You then use your Amazon secret access key to calculate the HMAC of that string. Informally, we call this process "signing the request," and we call the output of the HMAC algorithm the signature, because it simulates the security properties of a real signature. Finally, you add this signature as a parameter of the request by using the syntax described in this section.

When the system receives an authenticated request, it fetches the Amazon secret access key that you claim to have and uses it in the same way to compute a signature for the message it received. It then compares the signature it calculated against the signature presented by the requester. If the two signatures match, the system concludes that the requester must have access to the Amazon secret access key and therefore acts with the authority of the principal to whom the key was issued. If the two signatures do not match, the request is dropped and the system responds with an error message.

Example Authenticated Amazon S3 REST request

```
GET /photos/puppy.jpg HTTP/1.1
Host: awsexamplebucket1.us-west-1.s3.amazonaws.com
Date: Tue, 27 Mar 2007 19:36:42 +0000
```

```
Authorization: AWS AKIAIOSFODNN7EXAMPLE:
qgk2+6Sv9/oM7G3qLEjTH1a1l1g=
```

Using temporary security credentials

If you are signing your request using temporary security credentials (see [Making requests](#)), you must include the corresponding security token in your request by adding the `x-amz-security-token` header.

When you obtain temporary security credentials using the Amazon Security Token Service API, the response includes temporary security credentials and a session token. You provide the session

token value in the `x-amz-security-token` header when you send requests to Amazon S3. For information about the Amazon Security Token Service API provided by IAM, go to [Action](#) in the *Amazon Security Token Service API Reference Guide*.

The authentication header

The Amazon S3 REST API uses the standard HTTP Authorization header to pass authentication information. (The name of the standard header is unfortunate because it carries authentication information, not authorization.) Under the Amazon S3 authentication scheme, the Authorization header has the following form:

```
Authorization: AWS AWSAccessKeyId:Signature
```

Developers are issued an Amazon access key ID and Amazon secret access key when they register. For request authentication, the `AWSAccessKeyId` element identifies the access key ID that was used to compute the signature and, indirectly, the developer making the request.

The `Signature` element is the RFC 2104 HMAC-SHA1 of selected elements from the request, and so the `Signature` part of the Authorization header will vary from request to request. If the request signature calculated by the system matches the `Signature` included with the request, the requester will have demonstrated possession of the Amazon secret access key. The request will then be processed under the identity, and with the authority, of the developer to whom the key was issued.

Following is pseudogrammar that illustrates the construction of the Authorization request header. (In the example, `\n` means the Unicode code point U+000A, commonly called newline).

```
Authorization = "AWS" + " " + AWSAccessKeyId + ":" + Signature;

Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of(YourSecretAccessKey), UTF-8-Encoding-Of( StringToSign ) ) );

StringToSign = HTTP-Verb + "\n" +
  Content-MD5 + "\n" +
  Content-Type + "\n" +
  Date + "\n" +
  CanonicalizedAmzHeaders +
  CanonicalizedResource;
```

```
CanonicalizedResource = [ "/" + Bucket ] +  
    <HTTP-Request-URI, from the protocol name up to the query string> +  
    [ subresource, if present. For example "?acl", "?location", or "?logging" ];  
  
CanonicalizedAmzHeaders = <described below>
```

HMAC-SHA1 is an algorithm defined by [RFC 2104 - Keyed-Hashing for Message Authentication](#). The algorithm takes as input two byte-strings, a key and a message. For Amazon S3 request authentication, use your Amazon secret access key (YourSecretAccessKey) as the key, and the UTF-8 encoding of the StringToSign as the message. The output of HMAC-SHA1 is also a byte string, called the digest. The Signature request parameter is constructed by Base64 encoding this digest.

Request canonicalization for signing

Recall that when the system receives an authenticated request, it compares the computed request signature with the signature provided in the request in StringToSign. For that reason, you must compute the signature by using the same method used by Amazon S3. We call the process of putting a request in an agreed-upon form for signing *canonicalization*.

Constructing the CanonicalizedResource element

CanonicalizedResource represents the Amazon S3 resource targeted by the request. Construct it for a REST request as follows:

1. Start with an empty string ("").
2. If the request specifies a bucket using the HTTP Host header (virtual hosted-style), append the bucket name preceded by a "/" (e.g., "/bucketname"). For path-style requests and requests that don't address a bucket, do nothing. For more information about virtual hosted-style requests, see [Virtual hosting of buckets](#).

For a virtual hosted-style request "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg", the CanonicalizedResource is "/awsexamplebucket1".

For the path-style request, "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg", the CanonicalizedResource is "".

3. Append the path part of the un-decoded HTTP Request-URI, up-to but not including the query string.

For a virtual hosted-style request "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg", the CanonicalizedResource is "/awsexamplebucket1/photos/puppy.jpg".

For a path-style request, "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg", the CanonicalizedResource is "/awsexamplebucket1/photos/puppy.jpg". At this point, the CanonicalizedResource is the same for both the virtual hosted-style and path-style request.

For a request that does not address a bucket, such as [GET Service](#), append "/".

4. If the request addresses a subresource, such as ?versioning, ?location, ?acl, ?lifecycle, or ?versionid, append the subresource, its value if it has one, and the question mark. Note that in case of multiple subresources, subresources must be lexicographically sorted by subresource name and separated by '&', e.g., ?acl&versionId=*value*.

The subresources that must be included when constructing the CanonicalizedResource Element are acl, lifecycle, location, logging, notification, partNumber, policy, requestPayment, uploadId, uploads, versionId, versioning, versions, and website.

If the request specifies query string parameters overriding the response header values (see [Get Object](#)), append the query string parameters and their values. When signing, you do not encode these values; however, when making the request, you must encode these parameter values. The query string parameters in a GET request include response-content-type, response-content-language, response-expires, response-cache-control, response-content-disposition, and response-content-encoding.

The delete query string parameter must be included when you create the CanonicalizedResource for a multi-object Delete request.

Elements of the CanonicalizedResource that come from the HTTP Request-URI should be signed literally as they appear in the HTTP request, including URL-Encoding meta characters.

The CanonicalizedResource might be different than the HTTP Request-URI. In particular, if your request uses the HTTP Host header to specify a bucket, the bucket does not appear in the HTTP Request-URI. However, the CanonicalizedResource continues to include the bucket. Query string parameters might also appear in the Request-URI but are not included in CanonicalizedResource. For more information, see [Virtual hosting of buckets](#).

Constructing the CanonicalizedAmzHeaders element

To construct the CanonicalizedAmzHeaders part of `StringToSign`, select all HTTP request headers that start with 'x-amz-' (using a case-insensitive comparison), and use the following process.

1. Convert each HTTP header name to lowercase. For example, 'X-Amz-Date' becomes 'x-amz-date'.
2. Sort the collection of headers lexicographically by header name.
3. Combine header fields with the same name into one "header-name:comma-separated-value-list" pair as prescribed by RFC 2616, section 4.2, without any spaces between values. For example, the two metadata headers 'x-amz-meta-username: fred' and 'x-amz-meta-username: barney' would be combined into the single header 'x-amz-meta-username: fred,barney'.
4. "Unfold" long headers that span multiple lines (as allowed by RFC 2616, section 4.2) by replacing the folding spaces (including new-line) by a single space.
5. Trim any spaces around the colon in the header. For example, the header 'x-amz-meta-username: fred, barney' would become 'x-amz-meta-username:fred,barney'.
6. Finally, append a newline character (U+000A) to each canonicalized header in the resulting list. Construct the CanonicalizedResource element by concatenating all headers in this list into a single string.

Positional versus named HTTP header StringToSign elements

The first few header elements of `StringToSign` (Content-Type, Date, and Content-MD5) are positional in nature. `StringToSign` does not include the names of these headers, only their values from the request. In contrast, the 'x-amz-' elements are named. Both the header names and the header values appear in `StringToSign`.

If a positional header called for in the definition of `StringToSign` is not present in your request (for example, Content-Type or Content-MD5 are optional for PUT requests and meaningless for GET requests), substitute the empty string ("") for that position.

Time stamp requirement

A valid time stamp (using either the HTTP Date header or an x-amz-date alternative) is mandatory for authenticated requests. Furthermore, the client timestamp included with an authenticated request must be within 15 minutes of the Amazon S3 system time when the

request is received. If not, the request will fail with the `RequestTimeTooSkewed` error code. The intention of these restrictions is to limit the possibility that intercepted requests could be replayed by an adversary. For stronger protection against eavesdropping, use the HTTPS transport for authenticated requests.

Note

The validation constraint on request date applies only to authenticated requests that do not use query string authentication. For more information, see [Query string request authentication alternative](#).

Some HTTP client libraries do not expose the ability to set the `Date` header for a request. If you have trouble including the value of the `'Date'` header in the canonicalized headers, you can set the timestamp for the request by using an `'x-amz-date'` header instead. The value of the `x-amz-date` header must be in one of the RFC 2616 formats (<http://www.ietf.org/rfc/rfc2616.txt>). When an `x-amz-date` header is present in a request, the system will ignore any `Date` header when computing the request signature. Therefore, if you include the `x-amz-date` header, use the empty string for the `Date` when constructing the `StringToSign`. See the next section for an example.

Authentication examples

The examples in this section use the (non-working) credentials in the following table.

Parameter	Value
<code>AWSSecretAccessKey</code>	<code>AKIAIOSFODNN7EXAMPLE</code>
<code>AWSSecretAccessKey</code>	<code>wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY</code>

In the example `StringToSign`, formatting is not significant, and `\n` means the Unicode code point U+000A, commonly called newline. Also, the examples use `" +0000"` to designate the time zone. You can use `"GMT"` to designate timezone instead, but the signatures shown in the examples will be different.

Object GET

This example gets an object from the `awsexamplebucket1` bucket.

Request	StringToSign
<pre>GET /photos/puppy.jpg HTTP/1.1 Host: awsexamplebucket1.us- west-1.s3.amazonaws.com Date: Tue, 27 Mar 2007 19:36:42 +0000 Authorization: AWS AKIAIOSF0 DNN7EXAMPLE: qgk2+6Sv9/oM7G3qLEjTH1a11lg=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:36:42 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Note that the CanonicalizedResource includes the bucket name, but the HTTP Request-URI does not. (The bucket is specified by the Host header.)

Note

The following Python script calculates the preceding signature, using the provided parameters. You can use this script to construct your own signatures, replacing the keys and StringToSign as appropriate.

```
import base64
import hmac
from hashlib import sha1

access_key = 'AKIAIOSFODNN7EXAMPLE'.encode("UTF-8")
secret_key = 'wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY'.encode("UTF-8")

string_to_sign = 'GET\n\n\nTue, 27 Mar 2007 19:36:42 +0000\n/awsexamplebucket1/
photos/puppy.jpg'.encode("UTF-8")
signature = base64.b64encode(
    hmac.new(
        secret_key, string_to_sign, sha1
    ).digest()
).strip()

print(f"AWS {access_key.decode()}:{signature.decode()}")
```

Object PUT

This example puts an object into the `awsexamplebucket1` bucket.

Request	StringToSign
<pre>PUT /photos/puppy.jpg HTTP/1.1 Content-Type: image/jpeg Content-Length: 94328 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:15:45 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: iqRzw+ileNPu1fhspnRs8n0jjIA=</pre>	<pre>PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Note the Content-Type header in the request and in the StringToSign. Also note that the Content-MD5 is left blank in the StringToSign, because it is not present in the request.

List

This example lists the content of the `awsexamplebucket1` bucket.

Request	StringToSign
<pre>GET /?prefix=photos&max-keys=50&marker=puppy HTTP/1.1 User-Agent: Mozilla/5.0 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 19:42:41 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: m0WP8eCtspQl5Ahe6L1SozdX9YA=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:42:41 +0000\n /awsexamplebucket1/</pre>

Note the trailing slash on the CanonicalizedResource and the absence of query string parameters.

Fetch

This example fetches the access control policy subresource for the 'awsexamplebucket1' bucket.

Request	StringToSign
<pre>GET /?acl HTTP/1.1 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 19:44:46 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: 82ZHiFIjc+WbcwFKGUVeqspPn+0=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:44:46 +0000\n /awsexamplebucket1/?acl</pre>

Notice how the subresource query string parameter is included in the CanonicalizedResource.

Delete

This example deletes an object from the 'awsexamplebucket1' bucket using the path-style and Date alternative.

Request	StringToSign
<pre>DELETE /awsexamplebucket1/photos/puppy.jpg HTTP/1.1 User-Agent: dotnet Host: s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:20:27 +0000 x-amz-date: Tue, 27 Mar 2007 21:20:26 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: XbyTlbQdu9Xw5o8P4iMwPktxQd8=</pre>	<pre>DELETE\n \n \n Tue, 27 Mar 2007 21:20:26 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Note how we used the alternate 'x-amz-date' method of specifying the date (because our client library prevented us from setting the date, say). In this case, the x-amz-date takes precedence over the Date header. Therefore, date entry in the signature must contain the value of the x-amz-date header.

Upload

This example uploads an object to a CNAME style virtual hosted bucket with metadata.

Request	StringToSign
<pre>PUT /db-backup.dat.gz HTTP/1.1 User-Agent: curl/7.15.5 Host: static.example.com:8080 Date: Tue, 27 Mar 2007 21:06:08 +0000 x-amz-acl: public-read content-type: application/x-download Content-MD5: 4gJE4saaMU4BqNR0kLY+lw== X-Amz-Meta-ReviewedBy: joe@example.com X-Amz-Meta-ReviewedBy: jane@exam ple.com X-Amz-Meta-FileChecksum: 0x02661779 X-Amz-Meta-ChecksumAlgorithm: crc32 Content-Disposition: attachment; filename=database.dat Content-Encoding: gzip Content-Length: 5913339 Authorization: AWS AKIAIOSFODNN7EXAMP LE: jtBQa0Aq+DkULFI8qrpwIjGEx0E=</pre>	<pre>PUT\n 4gJE4saaMU4BqNR0kLY+lw==\n application/x-download\n Tue, 27 Mar 2007 21:06:08 +0000\n x-amz-acl:public-read\n x-amz-meta-checksumalgorithm:c rc32\n x-amz-meta-filechecksum:0x026 61779\n x-amz-meta-reviewedby: joe@example.com,jane@example.com \n /static.example.com/db-backup.dat .gz</pre>

Notice how the 'x-amz-' headers are sorted, trimmed of extra spaces, and converted to lowercase. Note also that multiple headers with the same name have been joined using commas to separate values.

Note how only the Content-Type and Content-MD5 HTTP entity headers appear in the StringToSign. The other Content-* entity headers do not.

Again, note that the CanonicalizedResource includes the bucket name, but the HTTP Request-URI does not. (The bucket is specified by the Host header.)

List all my buckets

Request	StringToSign
<pre>GET / HTTP/1.1 Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:29:59 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdE RIC03wnaRNKh60qZehG9s=</pre>	<pre>GET\n \n \n Wed, 28 Mar 2007 01:29:59 +0000\n /</pre>

Unicode keys

Request	StringToSign
<pre>GET /dictionary/fran%C3%A7ais/pr %c3%a9f%c3%a8re HTTP/1.1 Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:49:49 +0000 Authorization: AWS AKIAIOSFODNN7EXAMP LE:DNEZGsoieTZ92F3bUfSPQcbGmLM=</pre>	<pre>GET\n \n \n Wed, 28 Mar 2007 01:49:49 +0000\n /dictionary/fran%C3%A7ais/pr %c3%a9f%c3%a8re</pre>

Note

The elements in `StringToSign` that were derived from the Request-URI are taken literally, including URL-Encoding and capitalization.

REST request signing problems

When REST request authentication fails, the system responds to the request with an XML error document. The information contained in this error document is meant to help developers diagnose the problem. In particular, the `StringToSign` element of the `SignatureDoesNotMatch` error document tells you exactly what request canonicalization the system is using.

Some toolkits silently insert headers that you do not know about beforehand, such as adding the header Content-Type during a PUT. In most of these cases, the value of the inserted header remains constant, allowing you to discover the missing headers by using tools such as `Ethereal` or `tcpmon`.

Query string request authentication alternative

You can authenticate certain types of requests by passing the required information as query-string parameters instead of using the `Authorization` HTTP header. This is useful for enabling direct third-party browser access to your private Amazon S3 data without proxying the request. The idea is to construct a "presigned" request and encode it as a URL that an end-user's browser can retrieve. Additionally, you can limit a presigned request by specifying an expiration time.

For more information on using query parameters to authenticate requests , see [Authenticating Requests: Using Query Parameters \(Amazon Signature Version 4\)](#) in the *Amazon Simple Storage Service API Reference*. For examples of using the Amazon SDKs to generating presigned URLs, see [Sharing objects with presigned URLs](#) .

Creating a signature

Following is an example query string authenticated Amazon S3 REST request.

```
GET /photos/puppy.jpg
?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv4%3D HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
Date: Mon, 26 Mar 2007 19:37:58 +0000
```

The query string request authentication method doesn't require any special HTTP headers. Instead, the required authentication elements are specified as query string parameters:

Query string parameter name	Example value	Description
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE	Your Amazon access key ID. Specifies the Amazon secret access key used to sign the request and, indirectly,

Query string parameter name	Example value	Description
		the identity of the developer making the request.
Expires	1141889120	The time when the signature expires, specified as the number of seconds since the epoch (00:00:00 UTC on January 1, 1970). A request received after this time (according to the server) will be rejected.
Signature	vjbyPxybdZaNmGa%2B yT272YEAiv4%3D	The URL encoding of the Base64 encoding of the HMAC-SHA1 of StringToSign.

The query string request authentication method differs slightly from the ordinary method but only in the format of the Signature request parameter and the StringToSign element. Following is pseudo-grammar that illustrates the query string request authentication method.

```
Signature = URL-Encode( Base64( HMAC-SHA1( YourSecretAccessKey, UTF-8-Encoding-Of( StringToSign ) ) ) );
```

```
StringToSign = HTTP-VERB + "\n" +  
    Content-MD5 + "\n" +  
    Content-Type + "\n" +  
    Expires + "\n" +  
    CanonicalizedAmzHeaders +  
    CanonicalizedResource;
```

YourSecretAccessKey is the Amazon secret access key ID that Amazon assigns to you when you sign up to be an Amazon Web Service developer. Notice how the Signature is URL-Encoded to make it suitable for placement in the query string. Note also that in StringToSign, the HTTP Date positional element has been replaced with Expires. The CanonicalizedAmzHeaders and CanonicalizedResource are the same.

Note

In the query string authentication method, you do not use the Date or the x-amz-date request header when calculating the string to sign.

Query string request authentication

Request	StringToSign
<pre>GET /photos/puppy.jpg?AWSAccess KeyId=AKIAIOSFODNN7EXAMPLE& Signature=NpgCjnDziM%2BWFzo ENXmpNDUsSn8%3D& Expires=1175139620 HTTP/1.1 Host: awsexamplebucket1.s3.us-wes t-1.amazonaws.com</pre>	<pre>GET\n \n \n 1175139620\n /awsexamplebucket1/photos/puppy.jpg</pre>

We assume that when a browser makes the GET request, it won't provide a Content-MD5 or a Content-Type header, nor will it set any x-amz- headers, so those parts of the StringToSign are left blank.

Using Base64 encoding

HMAC request signatures must be Base64 encoded. Base64 encoding converts the signature into a simple ASCII string that can be attached to the request. Characters that could appear in the signature string like plus (+), forward slash (/), and equals (=) must be encoded if used in a URI. For example, if the authentication code includes a plus (+) sign, encode it as %2B in the request. Encode a forward slash as %2F and equals as %3D.

For examples of Base64 encoding, refer to the Amazon S3 [Authentication examples](#).

Browser-based uploads using POST (Amazon signature version 2)

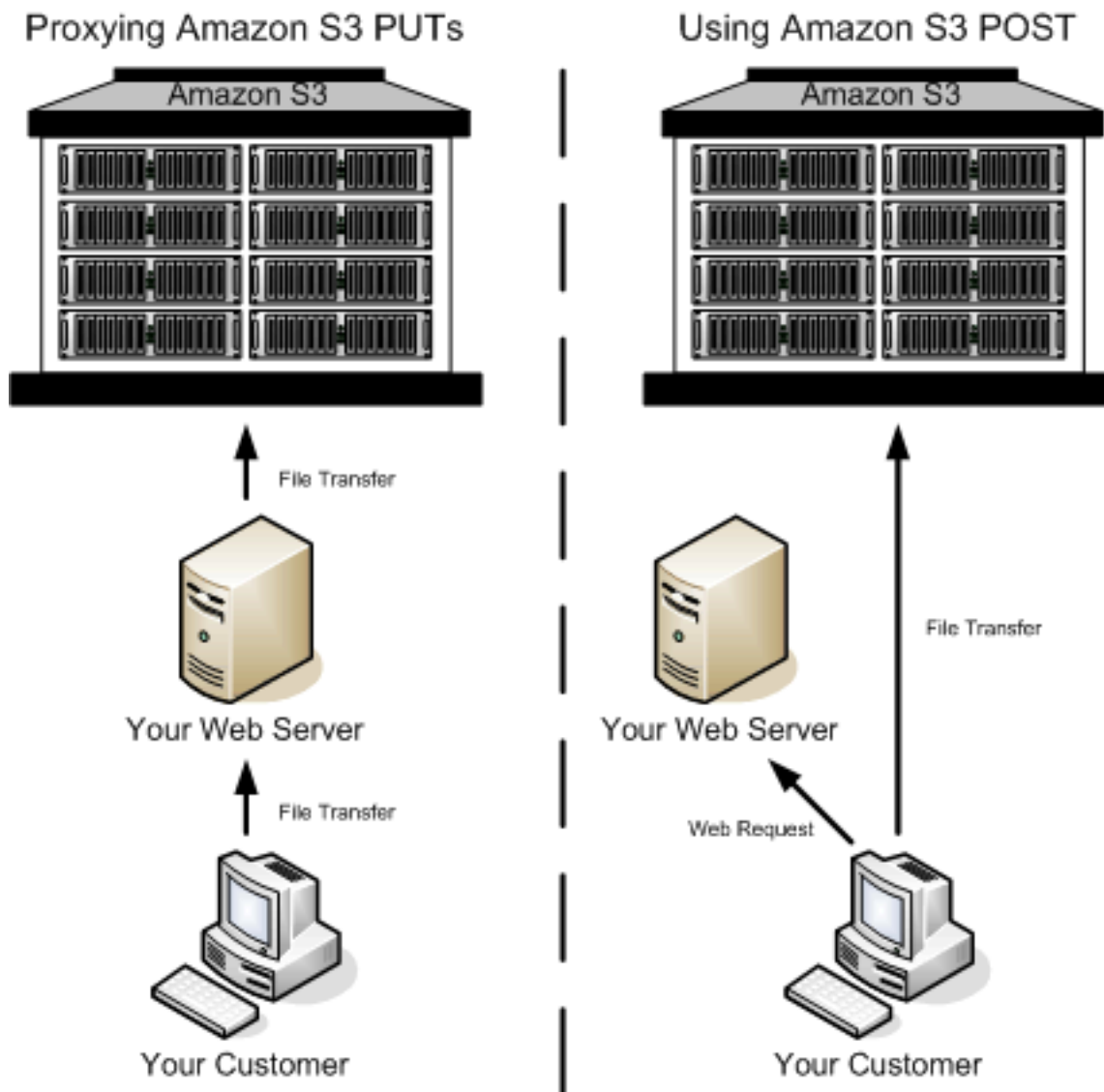
Amazon S3 supports POST, which allows your users to upload content directly to Amazon S3. POST is designed to simplify uploads, reduce upload latency, and save you money on applications where users upload data to store in Amazon S3.

Note

The request authentication discussed in this section is based on Amazon Signature Version 2, a protocol for authenticating inbound API requests to Amazon services.

Amazon S3 now supports Signature Version 4, a protocol for authenticating inbound API requests to Amazon services, in all Amazon Web Services Regions. At this time, Amazon Web Services Regions created before January 30, 2014 will continue to support the previous protocol, Signature Version 2. Any new regions after January 30, 2014 will support only Signature Version 4 and therefore all requests to those regions must be made with Signature Version 4. For more information, see [Authenticating Requests in Browser-Based Uploads Using POST \(Amazon Signature Version 4\)](#) in the *Amazon Simple Storage Service API Reference*.

The following figure shows an upload using Amazon S3 POST.



Uploading using POST

- 1 The user opens a web browser and accesses your web page.
- 2 Your web page contains an HTTP form that contains all the information necessary for the user to upload content to Amazon S3.
- 3 The user uploads content directly to Amazon S3.

Note

Query string authentication is not supported for POST.

HTML forms (Amazon signature version 2)

Topics

- [HTML form encoding](#)
- [HTML form declaration](#)
- [HTML form fields](#)
- [Policy construction](#)
- [Constructing a signature](#)
- [Redirection](#)

When you communicate with Amazon S3, you normally use the REST or SOAP API to perform put, get, delete, and other operations. With POST, users upload data directly to Amazon S3 through their browsers, which cannot process the SOAP API or create a REST PUT request.

Note

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the Amazon SDKs.

To allow users to upload content to Amazon S3 by using their browsers, you use HTML forms. HTML forms consist of a form declaration and form fields. The form declaration contains high-level information about the request. The form fields contain detailed information about the request, as well as the policy that is used to authenticate it and ensure that it meets the conditions that you specify.

Note

The form data and boundaries (excluding the contents of the file) cannot exceed 20 KB.

This section explains how to use HTML forms.

HTML form encoding

The form and policy must be UTF-8 encoded. You can apply UTF-8 encoding to the form by specifying it in the HTML heading or as a request header.

Note

The HTML form declaration does not accept query string authentication parameters.

The following is an example of UTF-8 encoding in the HTML heading:

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

The following is an example of UTF-8 encoding in a request header:

```
Content-Type: text/html; charset=UTF-8
```

HTML form declaration

The form declaration has three components: the action, the method, and the enclosure type. If any of these values is improperly set, the request fails.

The action specifies the URL that processes the request, which must be set to the URL of the bucket. For example, if the name of your bucket is `awsexamplebucket1` and the Region is US West (N. California), the URL is `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/`.

Note

The key name is specified in a form field.

The method must be POST.

The enclosure type (enctype) must be specified and must be set to multipart/form-data for both file uploads and text area uploads. For more information, go to [RFC 1867](#).

Example

The following example is a form declaration for the bucket "awsexamplebucket1".

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

HTML form fields

The following table describes fields that can be used within an HTML form.

 **Note**

The variable `${filename}` is automatically replaced with the name of the file provided by the user and is recognized by all form fields. If the browser or client provides a full or partial path to the file, only the text following the last slash (/) or backslash (\) will be used. For example, "C:\Program Files\directory1\file.txt" will be interpreted as "file.txt". If no file or file name is provided, the variable is replaced with an empty string.

Field name	Description	Required
AWSAccessKeyId	The Amazon Access Key ID of the owner of the bucket who grants an anonymous user access for a request that satisfies the set of constraints in the policy. This field is required if the request includes a policy document.	Conditional
acl	An Amazon S3 access control list (ACL). If an invalid access control list is specified, an error is generated.	No

Field name	Description	Required
	Type: String Default: private Valid Values: private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	REST-specific headers. For more information, see PUT Object .	No
key	The name of the uploaded key. To use the filename provided by the user, use the <code>\${filename}</code> variable. For example, if user Betty uploads the file lolcatz.jpg and you specify <code>/user/betty/\${filename}</code>, the file is stored as <code>/user/betty/lolcatz.jpg</code>. For more information, see Working with object metadata.	Yes
policy	Security policy describing what is permitted in the request. Requests without a security policy are considered anonymous and will succeed only on publicly writable buckets.	No

Field name	Description	Required
success_action_redirect, redirect	The URL to which the client is redirected upon successful upload. Amazon S3 appends the bucket, key, and etag values as query string parameters to the URL. If success_action_redirect is not specified, Amazon S3 returns the empty document type specified in the success_action_status field. If Amazon S3 cannot interpret the URL, it ignores the field. If the upload fails, Amazon S3 displays an error and does not redirect the user to a URL. For more information, see Redirection.  Note The redirect field name is deprecated and support for the redirect field name will be removed in the future.	No

Field name	Description	Required
success_action_status	The status code returned to the client upon successful upload if success_action_redirect is not specified. Valid values are 200, 201, or 204 (default). If the value is set to 200 or 204, Amazon S3 returns an empty document with a 200 or 204 status code. If the value is set to 201, Amazon S3 returns an XML document with a 201 status code. For information about the content of the XML document, see POST Object. If the value is not set or if it is set to an invalid value, Amazon S3 returns an empty document with a 204 status code.  Note Some versions of the Adobe Flash player do not properly handle HTTP responses with an empty body. To support uploads through Adobe Flash, we recommend setting success_action_status to 201.	No

Field name	Description	Required
signature	The HMAC signature constructed by using the secret access key that corresponds to the provided <code>AWSAccessKeyId</code>. This field is required if a policy document is included with the request. For more information, see Identity and Access Management for Amazon S3.	Conditional
x-amz-security-token	A security token used by session credentials If the request is using Amazon DevPay then it requires two <code>x-amz-security-token</code> form fields: one for the product token and one for the user token. If the request is using session credentials, then it requires one <code>x-amz-security-token</code> form. For more information, see Temporary Security Credentials in the <i>IAM User Guide</i>.	No
Other field names prefixed with x-amz-meta-	User-specified metadata. Amazon S3 does not validate or use this data. For more information, see PUT Object.	No

Field name	Description	Required
file	File or text content. The file or content must be the last field in the form. Any fields below it are ignored. You cannot upload more than one file at a time.	Yes

Policy construction

Topics

- [Expiration](#)
- [Conditions](#)
- [Condition matching](#)
- [Character escaping](#)

The policy is a UTF-8 and Base64-encoded JSON document that specifies conditions that the request must meet and is used to authenticate the content. Depending on how you design your policy documents, you can use them per upload, per user, for all uploads, or according to other designs that meet your needs.

Note

Although the policy document is optional, we highly recommend it over making a bucket publicly writable.

The following is an example of a policy document:

```
{ "expiration": "2007-12-01T12:00:00.000Z",  
  
  "conditions": [  
  
    {"acl": "public-read" },
```



```
{ "bucket": "awsexamplebucket1" },  
  
  [ "starts-with", "$key", "user/eric/" ],  
  
]  
  
}
```

The policy document contains the expiration and conditions.

Expiration

The expiration element specifies the expiration date of the policy in ISO 8601 UTC date format. For example, "2007-12-01T12:00:00.000Z" specifies that the policy is not valid after midnight UTC on 2007-12-01. Expiration is required in a policy.

Conditions

The conditions in the policy document validate the contents of the uploaded object. Each form field that you specify in the form (except `AWSSignature`, `signature`, `file`, `policy`, and field names that have an `x-ignore-` prefix) must be included in the list of conditions.

Note

If you have multiple fields with the same name, the values must be separated by commas. For example, if you have two fields named "x-amz-meta-tag" and the first one has a value of "Ninja" and second has a value of "Stallman", you would set the policy document to `Ninja,Stallman`.

All variables within the form are expanded before the policy is validated. Therefore, all condition matching should be performed against the expanded fields. For example, if you set the key field to `user/betty/${filename}`, your policy might be `[ "starts-with", "$key", "user/betty/" ]`. Do not enter `[ "starts-with", "$key", "user/betty/${filename}" ]`. For more information, see [Condition matching](#).

The following table describes policy document conditions.

Element name	Description
acl	Specifies conditions that the ACL must meet. Supports exact matching and starts-with .
content-length-range	Specifies the minimum and maximum allowable size for the uploaded content. Supports range matching.
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	REST-specific headers. Supports exact matching and starts-with .
key	The name of the uploaded key. Supports exact matching and starts-with .
success_action_redirect, redirect	The URL to which the client is redirected upon successful upload. Supports exact matching and starts-with .
success_action_status	The status code returned to the client upon successful upload if success_action_redirect is not specified. Supports exact matching.
x-amz-security-token	Amazon DevPay security token. Each request that uses Amazon DevPay requires two x-amz-security-token form fields: one for the product token and one for the user token. As a result, the values must be separated by commas. For example, if the user token is ew91dHVIZQ== and the product

Element name	Description
	token is b0hnNVNKWVJIQTA= , you set the policy entry to: { "x-amz-security-token": "ew91dHViZQ==,b0hnNVNKWVJIQTA=" } .
Other field names prefixed with x-amz-meta-	User-specified metadata. Supports exact matching and starts-with .

 **Note**

If your toolkit adds additional fields (e.g., Flash adds filename), you must add them to the policy document. If you can control this functionality, prefix x-ignore- to the field so Amazon S3 ignores the feature and it won't affect future versions of this feature.

Condition matching

The following table describes condition matching types. Although you must specify one condition for each form field that you specify in the form, you can create more complex matching criteria by specifying multiple conditions for a form field.

Condition	Description
Exact Matches	Exact matches verify that fields match specific values. This example indicates that the ACL must be set to public-read: <pre>{"acl": "public-read" }</pre> This example is an alternate way to indicate that the ACL must be set to public-read: <pre>["eq", "\$acl", "public-read"]</pre>

Condition	Description
Starts With	If the value must start with a certain value, use <code>starts-with</code>. This example indicates that the key must start with <code>user/betty</code>: ["starts-with", "\$key", "user/betty/"]
Matching Any Content	To configure the policy to allow any content within a field, use <code>starts-with</code> with an empty value. This example allows any <code>success_action_redirect</code>: ["starts-with", "\$success_action_redirect", ""]
Specifying Ranges	For fields that accept ranges, separate the upper and lower ranges with a comma. This example allows a file size from 1 to 10 megabytes: ["content-length-range", 1048579, 10485760]

Character escaping

The following table describes characters that must be escaped within a policy document.

Escape sequence	Description
\\	Backslash
\\$	Dollar sign
\b	Backspace
\f	Form feed

Escape sequence	Description
<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\t</code>	Horizontal tab
<code>\v</code>	Vertical tab
<code>\uXXXX</code>	All Unicode characters

Constructing a signature

Step	Description
1	Encode the policy by using UTF-8.
2	Encode those UTF-8 bytes by using Base64.
3	Sign the policy with your secret access key by using HMAC SHA-1.
4	Encode the SHA-1 signature by using Base64.

For general information about authentication, see [Identity and Access Management for Amazon S3](#).

Redirection

This section describes how to handle redirects.

General redirection

On completion of the POST request, the user is redirected to the location that you specified in the `success_action_redirect` field. If Amazon S3 cannot interpret the URL, it ignores the `success_action_redirect` field.

If `success_action_redirect` is not specified, Amazon S3 returns the empty document type specified in the `success_action_status` field.

If the POST request fails, Amazon S3 displays an error and does not provide a redirect.

Pre-upload redirection

If your bucket was created using `<CreateBucketConfiguration>`, your end users might require a redirect. If this occurs, some browsers might handle the redirect incorrectly. This is relatively rare but is most likely to occur right after a bucket is created.

Upload examples (Amazon signature version 2)

Topics

- [File upload](#)
- [Text area upload](#)

Note

The request authentication discussed in this section is based on Amazon Signature Version 2, a protocol for authenticating inbound API requests to Amazon services.

Amazon S3 now supports Signature Version 4, a protocol for authenticating inbound API requests to Amazon services, in all Amazon Web Services Regions. At this time, Amazon Web Services Regions created before January 30, 2014 will continue to support the previous protocol, Signature Version 2. Any new regions after January 30, 2014 will support only Signature Version 4 and therefore all requests to those regions must be made with Signature Version 4. For more information, see [Examples: Browser-Based Upload using HTTP POST \(Using Amazon Signature Version 4\)](#) in the *Amazon Simple Storage Service API Reference*.

File upload

This example shows the complete process for constructing a policy and form that can be used to upload a file attachment.

Policy and form construction

The following policy supports uploads to Amazon S3 for the `awsexamplebucket1` bucket.

```
{  "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

This policy requires the following:

- The upload must occur before 12:00 UTC on December 1, 2007.
- The content must be uploaded to the `awsexamplebucket1` bucket.
- The key must start with `"user/eric/"`.
- The ACL is set to `public-read`.
- The `success_action_redirect` is set to `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html`.
- The object is an image file.
- The `x-amz-meta-uuid` tag must be set to `14365123651274`.
- The `x-amz-meta-tag` can contain any value.

The following is a Base64-encoded version of this policy.

```
eyJhZiZxhwaXJhdGlvbiI6IClYMDA3LTEyLTExVDEyMDAwMDAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgZidWN
```

Using your credentials create a signature, for example `0RavWzkygo6QX9caELEqKi9kDbU=` is the signature for the preceding policy document.

The following form supports a POST request to the `amzn-s3-demo-bucket` bucket that uses this policy.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
    ...
    <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
    enctype="multipart/form-data">
      Key to upload: <input type="input" name="key" value="user/eric/" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html" />
      Content-Type: <input type="input" name="Content-Type" value="image/jpeg" /><br />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
      <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
      <input type="hidden" name="Policy" value="POLICY" />
      <input type="hidden" name="Signature" value="SIGNATURE" />
      File: <input type="file" name="file" /> <br />
      <!-- The elements after this will be ignored -->
      <input type="submit" name="submit" value="Upload to Amazon S3" />
    </form>
    ...
  </html>
```

Sample request

This request assumes that the image uploaded is 117,108 bytes; the image data is not included.

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
```



```
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: 118698

--9431149156168
Content-Disposition: form-data; name="key"

user/eric/MyPicture.jpg
--9431149156168
Content-Disposition: form-data; name="acl"

public-read
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html
--9431149156168
Content-Disposition: form-data; name="Content-Type"

image/jpeg
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

Some,Tag,For,Picture
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--9431149156168
Content-Disposition: form-data; name="Policy"

eyJhZXBwaXJhdGlvbiI6IClYMDA3LTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN0ZXQoPGEvbm9keSBkZWU9
```

```
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

...file content...
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

Sample response

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
successful_upload.html?bucket=awsexamplebucket1&key=user/eric/
MyPicture.jpg&etag=&quot;39d459dfbc0faabbb5e179358dfb94c3&quot;
Server: AmazonS3
```

Text area upload

Topics

- [Policy and form construction](#)
- [Sample request](#)
- [Sample response](#)

The following example shows the complete process for constructing a policy and form to upload a text area. Uploading a text area is useful for submitting user-created content, such as blog postings.

Policy and form construction

The following policy supports text area uploads to Amazon S3 for the awsexamplebucket1 bucket.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
```

```

{"bucket": "awsexamplebucket1"},
["starts-with", "$key", "user/eric/"],
{"acl": "public-read"},
{"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
new_post.html"},
["eq", "$Content-Type", "text/html"],
{"x-amz-meta-uuid": "14365123651274"},
["starts-with", "$x-amz-meta-tag", ""]
]
}

```

This policy requires the following:

- The upload must occur before 12:00 GMT on 2007-12-01.
- The content must be uploaded to the `awsexamplebucket1` bucket.
- The key must start with `"user/eric/"`.
- The ACL is set to `public-read`.
- The `success_action_redirect` is set to `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html`.
- The object is HTML text.
- The `x-amz-meta-uuid` tag must be set to `14365123651274`.
- The `x-amz-meta-tag` can contain any value.

Following is a Base64-encoded version of this policy.

eyAiZXhwaXJhdGlvbiI6IClYMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obnNtaXR0In0sCiAgICBbInN0YXJ0cy13aXR0IiwgIiRrZXkiLCaIdXNlciLAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3Q0IiAiaHR0cDovL2pC5zM55hbWV6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCaIjENvbnRlbnQtVHlwZSIsICJ0ZXh0L2h0CAGIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXR0IiwgIiR4LWFtei1IsIClIiXQogIF0KfQo=

Using your credentials, create a signature. For example, qA7FWXKq6VvU68lI9KdveT1cWgF= is the signature for the preceding policy document.

The following form supports a POST request to the `amzn-s3-demo-bucket` bucket that uses this policy.

```
<html>
```

```

<head>
  ...
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  ...
</head>
<body>
  ...
  <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
  enctype="multipart/form-data">
    Key to upload: <input type="input" name="key" value="user/eric/" /><br />
    <input type="hidden" name="acl" value="public-read" />
    <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html" />
    <input type="hidden" name="Content-Type" value="text/html" />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
    <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
    <input type="hidden" name="Policy" value="POLICY" />
    <input type="hidden" name="Signature" value="SIGNATURE" />
    Entry: <textarea name="file" cols="60" rows="10">

Your blog post goes here.

    </textarea><br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>
  ...
</html>

```

Sample request

This request assumes that the image uploaded is 117,108 bytes; the image data is not included.

```

POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300

```

```
Connection: keep-alive
Content-Type: multipart/form-data; boundary=178521717625888
Content-Length: 118635

-178521717625888
Content-Disposition: form-data; name="key"

ser/eric/NewEntry.html
--178521717625888
Content-Disposition: form-data; name="acl"

public-read
--178521717625888
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html
--178521717625888
Content-Disposition: form-data; name="Content-Type"

text/html
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-tag"

Interesting Post
--178521717625888
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--178521717625888
Content-Disposition: form-data; name="Policy"
eyJhZlZxhwaXJhdGlvbiI6IClYMDA3LTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgZjIdWm
--178521717625888
Content-Disposition: form-data; name="Signature"

qA7FWXKq6VvU68lI9KdveT1cWgF=
--178521717625888
Content-Disposition: form-data; name="file"

...content goes here...
```

```
--178521717625888
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--178521717625888--
```

Sample response

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html?
bucket=awsexamplebucket1&key=user/eric/
NewEntry.html&etag=40c3271af26b7f1672e41b8a274d28d4
Server: AmazonS3
```

POST with adobe flash (Amazon signature version 2)

This section describes how to use POST with Adobe Flash.

Adobe flash player security

By default, the Adobe Flash Player security model prohibits Adobe Flash Players from making network connections to servers outside the domain that serves the SWF file.

To override the default, you must upload a publicly readable `crossdomain.xml` file to the bucket that will accept POST uploads. The following is a sample `crossdomain.xml` file.

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" secure="false" />
</cross-domain-policy>
```

Note

For more information about the Adobe Flash security model, go to the Adobe website.

Adding the `crossdomain.xml` file to your bucket allows any Adobe Flash Player to connect to the `crossdomain.xml` file within your bucket; however, it does not grant access to the actual Amazon S3 bucket.

Adobe flash considerations

The FileReference API in Adobe Flash adds the `Filename` form field to the POST request. When you build Adobe Flash applications that upload to Amazon S3 by using the FileReference API action, include the following condition in your policy:

```
['starts-with', '$Filename', '']
```

Some versions of the Adobe Flash Player do not properly handle HTTP responses that have an empty body. To configure POST to return a response that does not have an empty body, set `success_action_status` to 201. Amazon S3 will then return an XML document with a 201 status code. For information about the content of the XML document, see [POST Object](#). For information about form fields, see [HTML form fields](#).

Appendix: Lifecycle Configuration APIs (Deprecated)

Bucket lifecycle configuration is updated to support filters based on object tags. That is, you can now specify a rule that specifies key name prefix, one or more object tags, or both to select a subset of objects to which the rule applies. The APIs have been updated accordingly. The following topics describe the prior version of the PUT and GET bucket lifecycle operations for backward compatibility.

Topics

- [PUT Bucket lifecycle \(Deprecated\)](#)
- [GET Bucket lifecycle \(Deprecated\)](#)

PUT Bucket lifecycle (Deprecated)

Description

Important

For an updated version of this API, see [PutBucketLifecycleConfiguration](#). This version has been deprecated. Existing lifecycle configurations will work. For new lifecycle configurations, use the updated API.

Creates a new lifecycle configuration for the bucket or replaces an existing lifecycle configuration. For information about lifecycle configuration, see [Object Lifecycle Management](#) in the *Amazon Simple Storage Service User Guide*.

Permissions

By default, all Amazon S3 resources, including buckets, objects, and related subresources (for example, lifecycle configuration and website configuration) are private. Only the resource owner, the Amazon Web Services account that created the resource, can access it. The resource owner can optionally grant access permissions to others by writing an access policy. For this operation, users must get the `s3:PutLifecycleConfiguration` permission.

You can also explicitly deny permissions. Explicit denial also supersedes any other permissions. If you want to prevent users or accounts from removing or deleting objects from your bucket, you must deny them permissions for the following actions:

- `s3:DeleteObject`
- `s3:DeleteObjectVersion`
- `s3:PutLifecycleConfiguration`

For more information about permissions, see [Managing Access Permissions to Your Amazon S3 Resources](#) in the *Amazon Simple Storage Service User Guide*.

Requests

Syntax

```
PUT /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Content-Length: length
Date: date
Authorization: authorization string
Content-MD5: MD5
```

Lifecycle configuration in the request body

For details about authorization strings, see [Authenticating Requests \(Amazon Signature Version 4\)](#).

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

Name	Description	Required
Content-MD5	The base64-encoded 128-bit MD5 digest of the data. You must use this header as a message integrity check to verify that the request body was not corrupted in transit. For more information, see RFC 1864. Type: String Default: None	Yes

Request Body

In the request, you specify the lifecycle configuration in the request body. The lifecycle configuration is specified as XML. The following is an example of a basic lifecycle configuration. It specifies one rule. The Prefix in the rule identifies objects to which the rule applies. The rule also specifies two actions (Transition and Expiration). Each action specifies a timeline when

Amazon S3 should perform the action. The Status indicates whether the rule is enabled or disabled.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>key-prefix</Prefix>
    <Status>rule-status</Status>
    <Transition>
      <Date>value</Date>
      <StorageClass>storage class</StorageClass>
    </Transition>
    <Expiration>
      <Days>value</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

If the state of your bucket is versioning-enabled or versioning-suspended, you can have many versions of the same object: one current version and zero or more noncurrent versions. The following lifecycle configuration specifies the actions (NoncurrentVersionTransition, NoncurrentVersionExpiration) that are specific to noncurrent object versions.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>key-prefix</Prefix>
    <Status>rule-status</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>value</NoncurrentDays>
      <StorageClass>storage class</StorageClass>
    </NoncurrentVersionTransition>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>value</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

You can use the multipart upload API to upload large objects in parts. For more information about multipart uploads, see [Multipart Upload Overview](#) in the *Amazon Simple Storage Service User Guide*. With lifecycle configuration, you can tell Amazon S3 to cancel incomplete multipart

uploads, which are identified by the key name prefix specified in the rule, if they don't complete within a specified number of days. When Amazon S3 cancels a multipart upload, it deletes all parts associated with the upload. This ensures that you don't have incomplete multipart uploads that have left parts stored in Amazon S3, so you don't have to pay storage costs for them. The following is an example lifecycle configuration that specifies a rule with the `AbortIncompleteMultipartUpload` action. This action tells Amazon S3 to cancel incomplete multipart uploads seven days after initiation.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>SomeKeyPrefix</Prefix>
    <Status>rule-status</Status>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>7</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

The following table describes the XML elements in the lifecycle configuration.

Name	Description	Required
AbortIncompleteMultipartUpload	Container for specifying when an incomplete multipart upload becomes eligible for an abort operation. Child: DaysAfterInitiation Type: Container Ancestor: Rule	Yes, if no other action is specified for the rule
Date	Date when you want Amazon S3 to take the action. For more information, see Lifecycle Rules: Based on a Specific Date in the <i>Amazon Simple Storage Service User Guide</i>.	Yes, if Days and ExpiredObjectDelete

Name	Description	Required
	The date value must conform to ISO 8601 format. The time is always midnight UTC. Type: String Ancestor: <code>Expiration</code> or <code>Transition</code>	<code>eMarker</code> are absent
Days	Specifies the number of days after object creation when the specific rule action takes effect. Type: Nonnegative Integer when used with <code>Transition</code> , Positive Integer when used with <code>Expiration</code> Ancestor: <code>Expiration</code> , <code>Transition</code>	Yes, if <code>Date</code> and <code>ExpiredObjectDeleteMarker</code> are absent
DaysAfterInitiation	Specifies the number of days after initiating a multipart upload when the multipart upload must be completed. If it does not complete by the specified number of days, it becomes eligible for an abort operation and Amazon S3 cancels the incomplete multipart upload. Type: Positive Integer Ancestor: <code>AbortIncompleteMultipartUpload</code>	Yes, if a parent tag is specified

Name	Description	Required
Expiration	This action specifies a period in an object's lifetime when Amazon S3 should take the appropriate expiration action. The action Amazon S3 takes depends on whether the bucket is versioning-enabled. • If versioning has never been enabled on the bucket, Amazon S3 deletes the only copy of the object permanently. • If the bucket is versioning-enabled (or versioning is suspended), the action applies only to the current version of the object. A versioning-enabled bucket can have many versions of the same object: one current version and zero or more noncurrent versions. Instead of deleting the current version, Amazon S3 makes it a noncurrent version by adding a delete marker as the new current version.  Important If a bucket's state is versioning-suspended, Amazon S3 creates a delete marker with version ID null. If you have a version with version ID null, Amazon S3 overwrites that version.	Yes, if no other action is present in the Rule.

Note

To set the expiration for noncurrent objects, use the `NoncurrentVersionExpiration` action.

Name	Description	Required
ExpiredObjectDeleteMarker	On a versioned bucket (a versioning-enabled or versioning-suspended bucket), you can add this element in the lifecycle configuration to tell Amazon S3 to delete expired object delete markers. For an example, see Example 8: Removing Expired Object Delete Markers in the <i>Amazon Simple Storage Service User Guide</i>. Don't add it to a non-versioned bucket, because that type of bucket cannot include delete markers. Type: String Valid values: true false (the value false is allowed, but it is no-op, which means that Amazon S3 will not take action) Ancestor: Expiration	Yes, if Date and Days are absent
NoncurrentDays	Specifies the number of days an object is noncurrent before Amazon S3 can perform the associated action. For information about the noncurrent days calculations, see How Amazon S3 Calculates When an Object Became Noncurrent in the <i>Amazon Simple Storage Service User Guide</i>. Type: Nonnegative Integer when used with NoncurrentVersionTransition , Positive Integer when used with NoncurrentVersionExpiration Ancestor: NoncurrentVersionExpiration or NoncurrentVersionTransition	Yes

Name	Description	Required
NoncurrentVersionExpiration	Specifies when noncurrent object versions expire. Upon expiration, Amazon S3 permanently deletes the noncurrent object versions. Set this lifecycle configuration action on a bucket that has versioning enabled (or suspended) to tell Amazon S3 to delete noncurrent object versions at a specific period in the object's lifetime. Type: Container Children: NoncurrentDays Ancestor: Rule	Yes, if no other action is present in the Rule
NoncurrentVersionTransition	Container for the transition rule that describes when noncurrent objects transition to the STANDARD_IA , ONEZONE_IA , or storage class. If your bucket is versioning-enabled (or if versioning is suspended), you can set this action to tell Amazon S3 to transition noncurrent object versions at a specific period in the object's lifetime. Type: Container Children: NoncurrentDays and StorageClass Ancestor: Rule	Yes, if no other action is present in the Rule

Name	Description	Required
Prefix	Object key prefix that identifies one or more objects to which the rule applies. Type: String Ancestor: Rule	Yes
Rule	Container for a lifecycle rule. A lifecycle configuration can contain as many as 1000 rules. Type: Container Ancestor: LifecycleConfiguration	Yes
Status	If enabled, Amazon S3 executes the rule as scheduled. If it is disabled, Amazon S3 ignores the rule. Type: String Ancestor: Rule Valid values: Enabled, Disabled	Yes
StorageClass	Specifies the Amazon S3 storage class to which you want the object to transition. Type: String Ancestor: Transition and NoncurrentVersionTransition Valid values: STANDARD_IA ONEZONE_IA GLACIER	Yes This element is required only if you specify one or both its ancestors.

Name	Description	Required
Transition	This action specifies a period in the objects' lifetime when Amazon S3 should transition them to the <code>STANDARD_IA</code> , <code>ONEZONE_IA</code> , or storage class. When this action is in effect, what Amazon S3 does depends on whether the bucket is versioning-enabled. • If versioning has never been enabled on the bucket, Amazon S3 transitions the only copy of the object to the specified storage class. • If your bucket is versioning-enabled (or versioning is suspended), Amazon S3 transitions only the current versions of objects identified in the rule.  Note A versioning-enabled bucket can have many versions of an object. This action has no effect on noncurrent object versions. To transition noncurrent objects, you must use the <code>NoncurrentVersionTransition</code> action. Type: Container Children: Days or Date, and StorageClass	Yes, if no other action is present in the Rule

Name	Description	Required
	Ancestor: Rule	

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see [Common Response Headers](#).

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see [Error Responses](#).

Examples

Example 1: Add Lifecycle Configuration to a Bucket That Is Not Versioning-enabled

The following lifecycle configuration specifies two rules, each with one action.

- The Transition action tells Amazon S3 to transition objects with the "documents/" prefix to the storage class 30 days after creation.
- The Expiration action tells Amazon S3 to delete objects with the "logs/" prefix 365 days after creation.

```
<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
```

```
</Rule>
<Rule>
  <ID>id2</ID>
  <Prefix>logs/</Prefix>
  <Status>Enabled</Status>
  <Expiration>
    <Days>365</Days>
  </Expiration>
</Rule>
</LifecycleConfiguration>
```

The following is a sample PUT `/?lifecycle` request that adds the preceding lifecycle configuration to the `examplebucket` bucket.

```
PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Wed, 14 May 2014 02:11:21 GMT
Content-MD5: q6yJD1IkBaGGfb3QLY69A==
Authorization: authorization string
Content-Length: 415

<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>id2</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

The following is a sample response.

```
HTTP/1.1 200 OK
x-amz-id-2: r+qR7+nhXtJDDIJ0JJYcd+1j5nM/rUFiiiZ/fNbD0sd3JUE8NWMLNHXmvPfwMpdC
x-amz-request-id: 9E26D08072A8EF9E
Date: Wed, 14 May 2014 02:11:22 GMT
Content-Length: 0
Server: AmazonS3
```

Example 2: Add Lifecycle Configuration to a Versioning-enabled Bucket

The following lifecycle configuration specifies two rules, each with one action for Amazon S3 to perform. You specify these actions when your bucket is versioning-enabled or versioning is suspended:

- The `NoncurrentVersionExpiration` action tells Amazon S3 to expire noncurrent versions of objects with the "logs/" prefix 100 days after the objects become noncurrent.
- The `NoncurrentVersionTransition` action tells Amazon S3 to transition noncurrent versions of objects with the "documents/" prefix to the storage class 30 days after they become noncurrent.

```
<LifecycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>100</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>TransitionAfterBecomingNonCurrent</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>30</NoncurrentDays>
      <StorageClass>GLACIER</StorageClass>
    </NoncurrentVersionTransition>
  </Rule>
</LifecycleConfiguration>
```

The following is a sample PUT `/?lifecycle` request that adds the preceding lifecycle configuration to the `examplebucket` bucket.

```
PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Wed, 14 May 2014 02:21:48 GMT
Content-MD5: 96rxH9mDqVNKkaZDddgnw==
Authorization: authorization string
Content-Length: 598

<LifecycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>1</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>TransitionSoonAfterBecomingNonCurrent</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>0</NoncurrentDays>
      <StorageClass>GLACIER</StorageClass>
    </NoncurrentVersionTransition>
  </Rule>
</LifecycleConfiguration>
```

The following is a sample response.

```
HTTP/1.1 200 OK
x-amz-id-2: aXQ+KbIrmMmo0//3bMdDTw/CnjArwje+J49Hf+j44yRb/VmbIkgI05A+PT98Cp/6k07hf
+LD2mY=
x-amz-request-id: 02D7EC4C10381EB1
Date: Wed, 14 May 2014 02:21:50 GMT
Content-Length: 0
Server: AmazonS3
```

Additional Examples

For more examples of transitioning objects to storage classes such as STANDARD_IA or ONEZONE_IA, see [Examples of Lifecycle Configuration](#).

Related Resources

- [GetBucketLifecycleConfiguration](#)
- [POST Object restore](#)
- By default, a resource owner—in this case, a bucket owner, which is the Amazon Web Services account that created the bucket—can perform any of the operations. A resource owner can also grant others permission to perform the operation. For more information, see the following topics in the *Amazon Simple Storage Service User Guide*:
 - [Specifying Permissions in a Policy](#)
 - [Managing Access Permissions to Your Amazon S3 Resources](#)

GET Bucket lifecycle (Deprecated)

Description

Important

For an updated version of this API, see [GetBucketLifecycleConfiguration](#). If you configured a bucket lifecycle using the <filter> element, you should see an updated version of this topic. This topic is provided for backward compatibility.

Returns the lifecycle configuration information set on the bucket. For information about lifecycle configuration, go to [Object Lifecycle Management](#) in the *Amazon Simple Storage Service User Guide*.

To use this operation, you must have permission to perform the `s3:GetLifecycleConfiguration` action. The bucket owner has this permission by default. The bucket owner can grant this permission to others. For more information about permissions, see [Managing Access Permissions to Your Amazon S3 Resources](#) in the *Amazon Simple Storage Service User Guide*.

Requests

Syntax

```
GET /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests \(Amazon Signature Version 4\))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see [Common Request Headers](#).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see [Common Response Headers](#).

Response Elements

This implementation of GET returns the following response elements.

Name	Description	Required
AbortIncompleteMultipartUpload	Container for specifying when an incomplete multipart upload becomes eligible for an abort operation. Child: DaysAfterInitiation Type: Container Ancestor: Rule	Yes, if no other action is specified for the rule
Date	Date when you want Amazon S3 to take the action. For more information, see Lifecycle Rules: Based on a Specific Date in the <i>Amazon Simple Storage Service User Guide</i> . The date value must conform to the ISO 8601 format. The time is always midnight UTC. Type: String Ancestor: Expiration or Transition	Yes, if Days and ExpiredObjectDeleteMarker are absent

Name	Description	Required
Days	Specifies the number of days after object creation when the specific rule action takes effect. The object's eligibility time is calculated as creation time + the number of days with the resulting time rounded to midnight UTC of the next day. Type: Non-negative Integer when used with <code>Transition</code> , Positive Integer when used with <code>Expiration</code> . Ancestor: <code>Transition</code> or <code>Expiration</code>	Yes, if <code>Date and ExpiredObjectDeleteMarker</code> are absent
DaysAfterInitiation	Specifies the number of days after initiating a multipart upload when the multipart upload must be completed. If it does not complete by the specified number of days, it becomes eligible for an abort operation and Amazon S3 cancels the incomplete multipart upload. Type: Positive Integer Ancestor: <code>AbortIncompleteMultipartUpload</code>	Yes, if <code>Date</code> is absent

Name	Description	Required
Expiration	This action specifies a period in the object's lifetime when Amazon S3 should take the appropriate expiration action. The expiration action occurs only on objects that are eligible according to the period specified in the child Date or Days element. The action Amazon S3 takes depends on whether the bucket is versioning enabled. • If versioning has never been enabled on the bucket, Amazon S3 deletes the only copy of the object permanently. • Otherwise, if your bucket is versioning-enabled (or versioning is suspended), the action applies only to the current version of the object. Buckets that are versioning-enabled or versioning-suspended can have many versions of the same object: one current version, and zero or more noncurrent versions. Instead of deleting the current version, Amazon S3 makes it a noncurrent version by adding a delete marker as the new current version.  Important If the state of a bucket is versioning-suspended, Amazon S3 creates a delete marker with version ID null. If you have a version with	Yes, if the parent tag is specified

Name	Description	Required
	version ID null, then Amazon S3 overwrites that version. Note To set the expiration for noncurrent objects, you must use the NoncurrentVersionExpiration action. Type: Container Children: Days or Date Ancestor: Rule	
ID	Unique identifier for the rule. The value cannot be longer than 255 characters. Type: String Ancestor: Rule	No
LifecycleConfiguration	Container for lifecycle rules. You can add as many as 1000 rules. Type: Container Children: Rule Ancestor: None	Yes

Name	Description	Required
ExpiredObjectDeleteMarker	On a versioned bucket (versioning-enabled or versioning-suspended bucket), this element indicates whether Amazon S3 will delete any expired object delete markers in the bucket. For an example, go to Example 8: Specify Expiration Action to Remove Expired Object Delete Markers in the <i>Amazon Simple Storage Service User Guide</i>. Type: String Valid values: true false (the value false is allowed but it is no-op, Amazon S3 doesn't take action if the value is false) Ancestor: Expiration	Yes, if Date and Days are absent
NoncurrentDays	Specifies the number of days that an object is noncurrent before Amazon S3 can perform the associated action. For information about calculating noncurrent days, see Lifecycle Rules Based on the Number of Days in the <i>Amazon Simple Storage Service User Guide</i>. Type: Nonnegative Integer when used with NoncurrentVersionTransition , Positive Integer when used with NoncurrentVersionExpiration Ancestor: NoncurrentVersionExpiration or NoncurrentVersionTransition	Yes, only if the ancestor is present

Name	Description	Required
NoncurrentVersionExpiration	Specifies when noncurrent object versions expire. Upon expiration, Amazon S3 permanently deletes the noncurrent object versions. Set this lifecycle configuration action on a bucket that has versioning enabled (or suspended) to request that Amazon S3 delete noncurrent object versions at a specific period in the object's lifetime. Type: Container Children: NoncurrentDays Ancestor: Rule	Yes, if no other action is present in the Rule
NoncurrentVersionTransition	Container for the transition rule that describes when noncurrent objects transition to the STANDARD_IA , ONEZONE_IA , or the storage class. If your bucket is versioning-enabled (or versioning is suspended), you can set this action to request Amazon S3 to transition noncurrent object versions to the storage class at a specific period in the object's lifetime. Type: Container Children: NoncurrentDays and StorageClass Ancestor: Rule	Yes, if no other action is present in the Rule

Name	Description	Required
Prefix	Object key prefix identifying one or more objects to which the rule applies. Type: String Ancestor: Rule	Yes
Rule	Container for a lifecycle rule. Type: Container Ancestor: LifecycleConfiguration	Yes
Status	If Enabled, Amazon S3 executes the rule as scheduled. If Disabled, Amazon S3 ignores the rule. Type: String Ancestor: Rule Valid values: Enabled or Disabled	Yes
StorageClass	Specifies the Amazon S3 storage class to which you want to transition the object. Type: String Ancestor: Transition and NoncurrentVersionTransition Valid values: STANDARD_IA ONEZONE_IA	Yes

Name	Description	Required
Transition	This action specifies a period in the objects' lifetime when Amazon S3 should transition them to the <code>STANDARD_IA</code> , <code>ONEZONE_IA</code> , or storage class. When this action is in effect, what Amazon S3 does depends on whether the bucket is versioning-enabled. • If versioning has never been enabled on the bucket, Amazon S3 transitions the only copy of the object to the specified storage class. • When your bucket is versioning-enabled (or versioning is suspended), Amazon S3 transitions only the current versions of the objects identified in the rule.  Note A versioning-enabled or versioning-suspended bucket can contain many versions of an object. This action has no effect on the noncurrent object versions. To transition noncurrent objects, you must use the <code>NoncurrentVersionTransition</code> action. Type: Container	Yes, if no other action is present in the <code>Rule</code>

Name	Description	Required
	Children: Days or Date, and StorageClass	
	Ancestor: Rule	

Special Errors

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
NoSuchLifecycleConfiguration	The lifecycle configuration does not exist.	404 Not Found	Client

For general information about Amazon S3 errors and a list of error codes, see [Error responses](#).

Examples

Example 1: Retrieve a Lifecycle Subresource

This example is a GET request to retrieve the `lifecycle` subresource from the specified bucket, and an example response with the returned lifecycle configuration.

Sample Request

```
GET /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Thu, 15 Nov 2012 00:17:21 GMT
Authorization: signatureValue
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: ITnGT1y4RyTmXa3rPi4hk1TXouTf0hccUjo0iCPjz6FnfIutBj3M7fPGlW02SEWp
x-amz-request-id: 51991C342C575321
Date: Thu, 15 Nov 2012 00:17:23 GMT
Server: AmazonS3
Content-Length: 358
```

```
<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Rule>
    <ID>Archive and then delete rule</ID>
    <Prefix>projectdocs</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Related Resources

- [PutBucketLifecycleConfiguration](#)
- [DeleteBucketLifecycle](#)

Document history