亚马逊云科技

Developer Guide

Amazon CloudFront

Amazon CloudFront: Developer Guide

Table of Contents

What is Amazon CloudFront? 1
How you set up CloudFront to deliver content 2
Pricing 4
Use cases 4
Accelerate static website content delivery 4
Serve video on demand or live streaming video 5
Encrypt specific fields throughout system processing 5
Customize at the edge 5
Serve private content by using Lambda@Edge customizations 6
How CloudFront delivers content 6
How CloudFront delivers content to your users 7
How CloudFront works with regional edge caches 8
CloudFront edge servers 10
Use the CloudFront managed prefix list 11
Accessing CloudFront 11
Working with Amazon SDKs 12
Getting started 13
Setting up 13
Sign up for an Amazon Web Services account 13
Secure IAM users 14
Set up the Amazon Command Line Interface or Amazon Tools for Windows PowerShell 14
Download an Amazon SDK 14
Getting started with a basic distribution 15
Prerequisites 15
Step 1: Create bucket 16
Step 2: Upload content 16
Step 3: Create distribution 17
Step 4: Access the content 18
Step 5: Clean up 19
Tips 19
Getting started with a secure static website 19
Solution overview 20
Deploying the solution 21
Working with distributions 27

Overview of distributions ... 27
Actions you can use with distributions ... 28
Required API fields for creating and updating distributions ... 29
Creating, updating, and deleting distributions ... 32
Steps for creating a distribution ... 33
Creating a distribution ... 35
Values that you specify ... 36
Values that are displayed ... 71
Testing a distribution ... 72
Updating a distribution ... 73
Tagging a distribution ... 75
Deleting a distribution ... 77
Using continuous deployment to safely test changes ... 79
Workflow for using CloudFront continuous deployment ... 81
Working with a staging distribution and continuous deployment policy ... 82
Monitoring a staging distribution ... 91
Understanding how continuous deployment works ... 91
Quotas and other considerations for continuous deployment ... 94
Using various origins ... 94
Using an Amazon S3 bucket ... 95
Using a MediaStore container or a MediaPackage channel ... 106
Using an Application Load Balancer ... 107
Using a Lambda function URL ... 107
Using Amazon EC2 (or another custom origin) ... 108
Using CloudFront origin groups ... 109
Using custom URLs ... 110
Adding an alternate domain name ... 110
Moving an alternate domain name to a different distribution ... 114
Removing an alternate domain name ... 120
Using wildcards in alternate domain names ... 121
Requirements for using alternate domain names ... 122
Restrictions on using alternate domain names ... 123
Using WebSockets ... 125
How the WebSocket protocol works ... 125
WebSocket requirements ... 126
Recommended settings ... 126

Working with policies 128
Controlling the cache key 129
Creating cache policies 129
Understanding cache policies 133
Using the managed cache policies 140
Understanding the cache key 145
Controlling origin requests 148
Creating origin request policies 149
Understanding origin request policies 153
Using the managed origin request policies 156
Adding CloudFront request headers 162
Understanding how origin request policies and cache policies work together 166
Adding or removing response headers 169
Creating response headers policies 170
Using the managed response headers policies 176
Understanding response headers policies 182
Adding, removing, or replacing content 197
Adding and accessing content 197
Updating existing content 198
Updating existing files using versioned file names 198
Updating existing content using the same file names 199
Removing content so CloudFront won't distribute it 199
Customizing file URLs 200
Using your own domain name (example.com) 200
Using a trailing slash (/) in URLs 201
Creating signed URLs for restricted content 201
Specifying a default root object 201
How to specify a default root object 202
How default root object works 203
How CloudFront works if you don't define a root object 205
Invalidating files 205
Choosing between invalidating files and using versioned file names 206
Determining which files to invalidate 207
Specifying the files to invalidate 207
Invalidating files using the console 211
Invalidating files using the CloudFront API 214

Concurrent invalidation request maximum 214
Paying for file invalidation 214
Serving compressed files 215
Configuring CloudFront to compress objects 216
How CloudFront compression works 216
Notes about CloudFront compression 217
File types that CloudFront compresses 219
ETag header conversion 221
Generating custom error responses 222
Configuring error response behavior 222
Creating a custom error page for specific HTTP status codes 224
Storing objects and custom error pages in different locations 226
Changing response codes returned by CloudFront 226
Controlling how long CloudFront caches errors 227
Using Amazon WAF protections 229
Enabling Amazon WAF for new distributions 230
Using an existing web ACL 231
Enabling Amazon WAF for existing distributions 231
Using an existing web ACL 232
Disabling Amazon WAF security protections 232
Setting up rate limiting 233
Using CloudFront security dashboards 234
Enabling Amazon WAF 235
Understanding trend data 236
Enabling bot control 236
Understanding logs 238
Managing CloudFront geographic restrictions 239
Security dashboard pricing 240
Configuring secure access and restricting access to content 241
Using HTTPS with CloudFront 241
Requiring HTTPS between viewers and CloudFront 243
Requiring HTTPS to a custom origin 245
Requiring HTTPS to an Amazon S3 origin 248
Supported protocols and ciphers between viewers and CloudFront 250
Supported protocols and ciphers between CloudFront and the origin 255
Charges for HTTPS connections 257

Using alternate domain names and HTTPS ... 258
Choosing how CloudFront serves HTTPS requests ... 258
Requirements for using SSL/TLS certificates with CloudFront ... 261
Quotas on using SSL/TLS certificates with CloudFront (HTTPS between viewers and CloudFront only) ... 266
Configuring alternate domain names and HTTPS ... 268
Determining the size of the public key in an SSL/TLS RSA certificate ... 272
Increasing the quotas for SSL/TLS certificates ... 272
Rotating SSL/TLS certificates ... 274
Reverting from a custom SSL/TLS certificate to the default CloudFront certificate ... 275
Switching from a custom SSL/TLS certificate with dedicated IP addresses to SNI ... 276
Restricting content with signed URLs and signed cookies ... 277
Overview of serving private content ... 278
Task list for serving private content ... 280
Specifying signers ... 281
Choosing between signed URLs and signed cookies ... 290
Using signed URLs ... 291
Using signed cookies ... 314
Using Linux commands and OpenSSL for base64 encoding and encryption ... 336
Code examples for signed URLs ... 337
Restricting access to an Amazon origin ... 365
Restricting access to an AWS Elemental MediaPackage v2 origin ... 366
Restricting access to an AWS Elemental MediaStore origin ... 372
Restricting access to an Amazon Lambda function URL origin ... 380
Restricting access to an Amazon Simple Storage Service origin ... 386
Restricting access to Application Load Balancers ... 400
Configuring CloudFront to add a custom HTTP header to requests ... 401
Configuring an Application Load Balancer to only forward requests that contain a specific header ... 403
(Optional) Improve the security of this solution ... 408
(Optional) Limit access to origin by using the Amazon-managed prefix list for CloudFront ... 409
Geographically restricting content ... 409
Using CloudFront geographic restrictions ... 410
Using a third-party geolocation service ... 412
Using field-level encryption to help protect sensitive data ... 413

Overview of field-level encryption ... 415
Setting up field-level encryption ... 416
Decrypting data fields at your origin ... 422
Optimizing caching and availability ... 425
Caching with edge locations ... 425
Improving your cache hit ratio ... 426
Specifying how long CloudFront caches your objects ... 426
Using Origin Shield ... 426
Caching based on query string parameters ... 427
Caching based on cookie values ... 427
Caching based on request headers ... 428
Remove `Accept-Encoding` header when compression is not needed ... 429
Serving media content by using HTTP ... 430
Using Origin Shield ... 430
Use cases for Origin Shield ... 431
Choosing the Amazon Region for Origin Shield ... 435
Enabling Origin Shield ... 437
Estimating Origin Shield costs ... 439
Origin Shield high availability ... 440
How Origin Shield interacts with other CloudFront features ... 440
Increasing availability with origin failover ... 441
Creating an origin group ... 443
Controlling origin timeouts and attempts ... 444
Use origin failover with Lambda@Edge functions ... 445
Use custom error pages with origin failover ... 446
Managing cache expiration ... 447
Using headers to control cache duration for individual objects ... 448
Serving stale (expired) content ... 449
Specifying the amount of time that CloudFront caches objects ... 451
Adding headers to your objects using the Amazon S3 console ... 457
Caching and query string parameters ... 457
Console and API settings for query string forwarding and caching ... 459
Optimizing caching ... 460
Query string parameters and CloudFront standard logs (access logs) ... 461
Caching content based on cookies ... 462
Caching content based on request headers ... 465

Headers and distributions – overview ... 465
Selecting the headers to base caching on ... 466
Configuring CloudFront to respect CORS settings ... 468
Configuring caching based on the device type ... 468
Configuring caching based on the language of the viewer ... 469
Configuring caching based on the location of the viewer ... 469
Configuring caching based on the protocol of the request ... 469
Configuring caching for compressed files ... 469
How caching based on headers affects performance ... 469
How the case of headers and header values affects caching ... 470
Headers that CloudFront returns to the viewer ... 470
Troubleshooting ... 471
Troubleshooting distribution issues ... 471
CloudFront returns an InvalidViewerCertificate error when I try to add an alternate domain name ... 471
I can't view the files in my distribution ... 473
Error message: Certificate: <certificate-id> is being used by CloudFront ... 474
Troubleshooting error responses from your origin ... 475
HTTP 400 status code (Bad Request) ... 475
HTTP 502 status code (Bad Gateway) ... 476
HTTP 502 status code (Lambda validation error) ... 479
HTTP 502 status code (DNS error) ... 479
HTTP 503 status code (function execution error) ... 480
HTTP 503 status code (Lambda limit exceeded) ... 481
HTTP 503 status code (Service Unavailable) ... 481
HTTP 504 status code (Gateway Timeout) ... 482
Load testing CloudFront ... 486
Request and response behavior ... 488
Request and response behavior for Amazon S3 origins ... 488
How CloudFront processes HTTP and HTTPS requests ... 488
How CloudFront processes and forwards requests to your Amazon S3 origin ... 489
How CloudFront processes responses from your Amazon S3 origin ... 495
Request and response behavior for custom origins ... 498
How CloudFront processes and forwards requests to your custom origin ... 498
How CloudFront processes responses from your custom origin ... 515
Request and response behavior for origin groups ... 519

Adding custom headers to origin requests ... 520
Use cases for origin custom headers ... 521
Configuring CloudFront to add custom headers to origin requests ... 522
Custom headers that CloudFront can't add to origin requests ... 522
Configuring CloudFront to forward the `Authorization` header ... 523
How range GETs are processed ... 524
Use range requests to cache large objects ... 525
How CloudFront processes HTTP 3xx status codes from your origin ... 525
How CloudFront processes and caches HTTP 4xx and 5xx status codes from your origin ... 526
How CloudFront processes errors when you have configured custom error pages ... 527
How CloudFront processes errors when you have not configured custom error pages ... 529
HTTP 4xx and 5xx status codes that CloudFront caches ... 531
Video on demand (VOD) and live streaming video ... 533
About streaming video: video on demand and live streaming ... 533
Delivering video on demand (VOD) ... 534
Configuring video on demand for Microsoft Smooth Streaming ... 535
Delivering live streaming video ... 537
Serving video using AWS Elemental MediaStore as the origin ... 538
Serving live video formatted with AWS Elemental MediaPackage ... 539
Functions at the edge ... 546
Which functions type to use ... 546
CloudFront Functions ... 549
Tutorial: A simple function ... 550
Tutorial: A function with key values ... 552
Writing function code ... 555
Managing functions ... 636
Using CloudFront KeyValueStore ... 652
Customizing with Lambda@Edge ... 665
Getting started ... 667
Setting IAM permissions and roles ... 676
Writing and creating functions ... 682
Adding triggers ... 688
Testing and debugging ... 695
Deleting functions and replicas ... 703
Event structure ... 704
Working with requests and responses ... 720

Example functions ... 726
Restrictions on edge functions ... 765
Restrictions on all edge functions ... 765
Restrictions on CloudFront Functions ... 771
Restrictions on Lambda@Edge ... 772
Reports, metrics, and logs ... 777
Amazon billing and usage reports for CloudFront ... 777
Amazon billing report for CloudFront ... 778
Amazon usage report for CloudFront ... 779
Interpreting your Amazon bill and the Amazon usage report for CloudFront ... 780
CloudFront console reports ... 785
CloudFront cache statistics reports ... 788
CloudFront popular objects report ... 793
CloudFront top referrers report ... 799
CloudFront usage reports ... 802
CloudFront viewers reports ... 809
Monitoring CloudFront metrics with Amazon CloudWatch ... 820
Viewing CloudFront and edge function metrics ... 821
Creating alarms ... 829
Downloading metrics data ... 830
Getting metrics using the API ... 833
CloudFront and edge function logging ... 839
Logging requests ... 839
Logging edge functions ... 839
Logging service activity ... 840
Using standard logs (access logs) ... 840
Real-time logs ... 860
Edge function logs ... 880
CloudTrail logs ... 883
Tracking configuration changes with Amazon Config ... 896
Set up Amazon Config with CloudFront ... 896
View CloudFront configuration history ... 897
Security ... 899
Data protection ... 899
Encryption in transit ... 901
Encryption at rest ... 902

Restrict access to content ... 902
Identity and Access Management ... 903
Audience ... 903
Authenticating with identities ... 904
Managing access using policies ... 907
How Amazon CloudFront works with IAM ... 909
Identity-based policy examples ... 916
Amazon managed policies ... 926
Troubleshooting ... 932
Logging and monitoring ... 934
Compliance validation ... 935
CloudFront compliance best practices ... 936
Resilience ... 937
CloudFront origin failover ... 937
Infrastructure security ... 938
Quotas ... 939
General quotas ... 939
General quotas on distributions ... 940
General quotas on policies ... 942
Quotas on CloudFront Functions ... 944
Quotas on key value stores ... 944
Quotas on Lambda@Edge ... 945
Quotas on SSL certificates ... 947
Quotas on invalidations ... 947
Quotas on key groups ... 947
Quotas on WebSocket connections ... 948
Quotas on field-level encryption ... 948
Quotas on cookies (legacy cache settings) ... 949
Quotas on query strings (legacy cache settings) ... 950
Quotas on headers ... 950
Code examples ... 952
Actions ... 953
`CreateDistribution` ... 953
`CreateFunction` ... 964
`CreateInvalidation` ... 967
`CreateKeyGroup` ... 969

CreatePublicKey ... 971
DeleteDistribution ... 973
GetCloudFrontOriginAccessIdentity ... 977
GetCloudFrontOriginAccessIdentityConfig ... 978
GetDistribution ... 980
GetDistributionConfig ... 983
ListCloudFrontOriginAccessIdentities ... 988
ListDistributions ... 989
UpdateDistribution ... 999
Scenarios ... 1011
Delete signing resources ... 1012
Sign URLs and cookies ... 1014
Related information ... 1018
Additional Amazon CloudFront documentation ... 1018
Getting support ... 1018
CloudFront developer tools and SDKs ... 1019
Tips from the Amazon Web Services blog ... 1019
Document history ... 1020
Updates before 2022 ... 1026
Amazon Glossary ... 1034

What is Amazon CloudFront?

Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.

- If the content is already in the edge location with the lowest latency, CloudFront delivers it immediately.
- If the content is not in that edge location, CloudFront retrieves it from an origin that you've defined—such as an Amazon S3 bucket, a MediaPackage channel, or an HTTP server (for example, a web server) that you have identified as the source for the definitive version of your content.

As an example, suppose that you're serving an image from a traditional web server, not from CloudFront. For example, you might serve an image, sunsetphoto.png, using the URL `https://example.com/sunsetphoto.png`.

Your users can easily navigate to this URL and see the image. But they probably don't know that their request is routed from one network to another—through the complex collection of interconnected networks that comprise the internet—until the image is found.

CloudFront speeds up the distribution of your content by routing each user request through the Amazon backbone network to the edge location that can best serve your content. Typically, this is a CloudFront edge server that provides the fastest delivery to the viewer. Using the Amazon network dramatically reduces the number of networks that your users' requests must pass through, which improves performance. Users get lower latency—the time it takes to load the first byte of the file—and higher data transfer rates.

You also get increased reliability and availability because copies of your files (also known as *objects)* are now held (or cached) in multiple edge locations around the world.

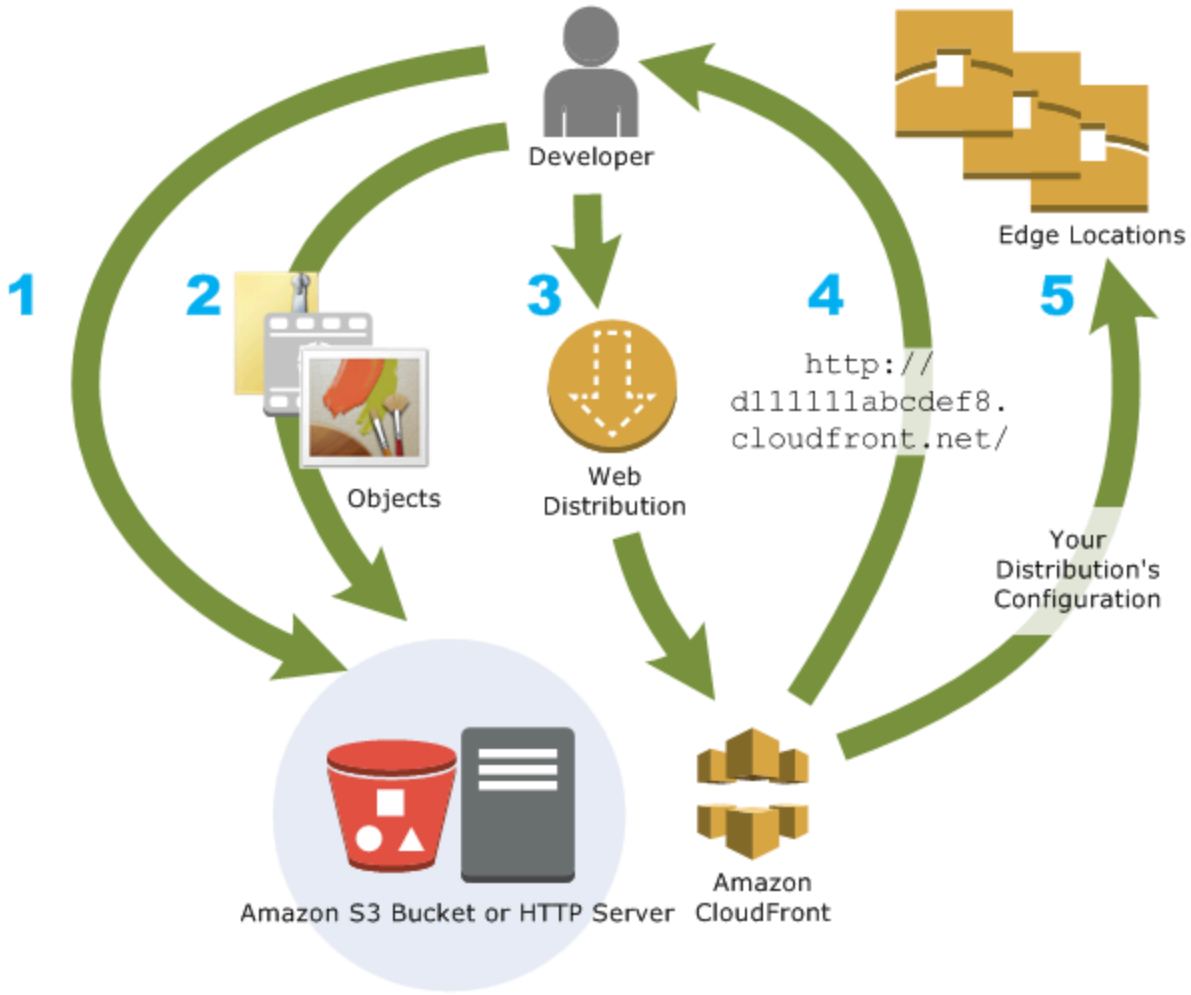
Topics

- How you set up CloudFront to deliver content
- Pricing

- CloudFront use cases
- How CloudFront delivers content
- Locations and IP address ranges of CloudFront edge servers
- Accessing CloudFront
- Using CloudFront with an Amazon SDK

How you set up CloudFront to deliver content

You create a CloudFront distribution to tell CloudFront where you want content to be delivered from, and the details about how to track and manage content delivery. Then CloudFront uses computers—edge servers—that are close to your viewers to deliver that content quickly when someone wants to see it or use it.



How you configure CloudFront to deliver your content

1. You specify *origin servers*, like an Amazon S3 bucket or your own HTTP server, from which CloudFront gets your files which will then be distributed from CloudFront edge locations all over the world.

   An origin server stores the original, definitive version of your objects. If you're serving content over HTTP, your origin server is either an Amazon S3 bucket or an HTTP server, such as a web server. Your HTTP server can run on an Amazon Elastic Compute Cloud (Amazon EC2) instance or on a server that you manage; these servers are also known as *custom origins.*

2. You upload your files to your origin servers. Your files, also known as *objects*, typically include web pages, images, and media files, but can be anything that can be served over HTTP.

   If you're using an Amazon S3 bucket as an origin server, you can make the objects in your bucket publicly readable, so that anyone who knows the CloudFront URLs for your objects can access them. You also have the option of keeping objects private and controlling who accesses them. See Serving private content with signed URLs and signed cookies.

3. You create a CloudFront *distribution*, which tells CloudFront which origin servers to get your files from when users request the files through your web site or application. At the same time, you specify details such as whether you want CloudFront to log all requests and whether you want the distribution to be enabled as soon as it's created.

4. CloudFront assigns a domain name to your new distribution that you can see in the CloudFront console, or that is returned in the response to a programmatic request, for example, an API request. If you like, you can add an alternate domain name to use instead.

5. CloudFront sends your distribution's configuration (but not your content) to all of its *edge locations* or *points of presence* (POPs)— collections of servers in geographically-dispersed data centers where CloudFront caches copies of your files.

As you develop your website or application, you use the domain name that CloudFront provides for your URLs. For example, if CloudFront returns `d111111abcdef8.cloudfront.net` as the domain name for your distribution, the URL for logo.jpg in your Amazon S3 bucket (or in the root directory on an HTTP server) is `https://d111111abcdef8.cloudfront.net/logo.jpg`.

Or you can set up CloudFront to use your own domain name with your distribution. In that case, the URL might be `https://www.example.com/logo.jpg`.

Optionally, you can configure your origin server to add headers to the files, to indicate how long you want the files to stay in the cache in CloudFront edge locations. By default, each file stays in an edge location for 24 hours before it expires. The minimum expiration time is 0 seconds; there isn't a maximum expiration time. For more information, see Managing how long content stays in the cache (expiration).

Pricing

CloudFront charges for data transfers out from its edge locations, along with HTTP or HTTPS requests. Pricing varies by usage type, geographical region, and feature selection.

The data transfer from your origin to CloudFront is always free when using Amazon origins like Amazon Simple Storage Service (Amazon S3), Elastic Load Balancing, or Amazon API Gateway. You are only billed for the outbound data transfer from CloudFront to the viewer when using Amazon origins.

For more information, see CloudFront pricing and the Billing and Savings Bundle FAQs.

CloudFront use cases

Using CloudFront can help you accomplish a variety of goals. This section lists just a few, together with links to more information, to give you an idea of the possibilities.

Topics

- Accelerate static website content delivery
- Serve video on demand or live streaming video
- Encrypt specific fields throughout system processing
- Customize at the edge
- Serve private content by using Lambda@Edge customizations

Accelerate static website content delivery

CloudFront can speed up the delivery of your static content (for example, images, style sheets, JavaScript, and so on) to viewers across the globe. By using CloudFront, you can take advantage of the Amazon backbone network and CloudFront edge servers to give your viewers a fast, safe, and reliable experience when they visit your website.

A simple approach for storing and delivering static content is to use an Amazon S3 bucket. Using S3 together with CloudFront has a number of advantages, including the option to use origin access control to easily restrict access to your S3 content.

For more information about using S3 together with CloudFront, including a Amazon CloudFormation template to help you get started quickly, see Amazon S3 + Amazon CloudFront: A Match Made in the Cloud.

Serve video on demand or live streaming video

CloudFront offers several options for streaming your media to global viewers—both pre-recorded files and live events.

- For video on demand (VOD) streaming, you can use CloudFront to stream in common formats such as MPEG DASH, Apple HLS, Microsoft Smooth Streaming, and CMAF, to any device.
- For broadcasting a live stream, you can cache media fragments at the edge, so that multiple requests for the manifest file that delivers the fragments in the right order can be combined, to reduce the load on your origin server.

For more information about how to deliver streaming content with CloudFront, see Video on demand and live streaming video with CloudFront.

Encrypt specific fields throughout system processing

When you configure HTTPS with CloudFront, you already have secure end-to-end connections to origin servers. When you add field-level encryption, you can protect specific data throughout system processing in addition to HTTPS security, so that only certain applications at your origin can see the data.

To set up field-level encryption, you add a public key to CloudFront, and then specify the set of fields that you want to be encrypted with the key. For more information, see Using field-level encryption to help protect sensitive data.

Customize at the edge

Running serverless code at the edge opens up a number of possibilities for customizing the content and experience for viewers, at reduced latency. For example, you can return a custom error message when your origin server is down for maintenance, so viewers don't get a generic

HTTP error message. Or you can use a function to help authorize users and control access to your content, before CloudFront forwards a request to your origin.

Using Lambda@Edge with CloudFront enables a variety of ways to customize the content that CloudFront delivers. To learn more about Lambda@Edge and how to create and deploy functions with CloudFront, see Customizing at the edge with Lambda@Edge. To see a number of code samples that you can customize for your own solutions, see Lambda@Edge example functions.

Serve private content by using Lambda@Edge customizations

Using Lambda@Edge can help you configure your CloudFront distribution to serve private content from your own custom origin, in addition to using signed URLs or signed cookies.

To serve private content using CloudFront, you do the following:

- Require that your users (viewers) access content using signed URLs or signed cookies.
- Restrict access to your origin so that it's only available from CloudFront's origin-facing servers. To do this, you can do one of the following:
  - For an Amazon S3 origin, you can use an origin access control (OAC).
  - For a custom origin, you can do the following:
    - If the custom origin is protected by an Amazon VPC security group or Amazon Firewall Manager, you can use the CloudFront managed prefix list to allow inbound traffic to your origin from only CloudFront's origin-facing IP addresses.
    - Use a custom HTTP header to restrict access to only requests from CloudFront. For more information, see the section called " Restricting access to files on custom origins" and the section called "Adding custom headers to origin requests". For an example that uses a custom header to restrict access to an Application Load Balancer origin, see the section called "Restricting access to Application Load Balancers".
    - If the custom origin requires custom access control logic, you can use Lambda@Edge to implement that logic, as described in this blog post: Serving Private Content Using Amazon CloudFront & Lambda@Edge.

How CloudFront delivers content

After some initial setup, CloudFront works together with your website or application and speeds up delivery of your content. This section explains how CloudFront serves your content when viewers request it.

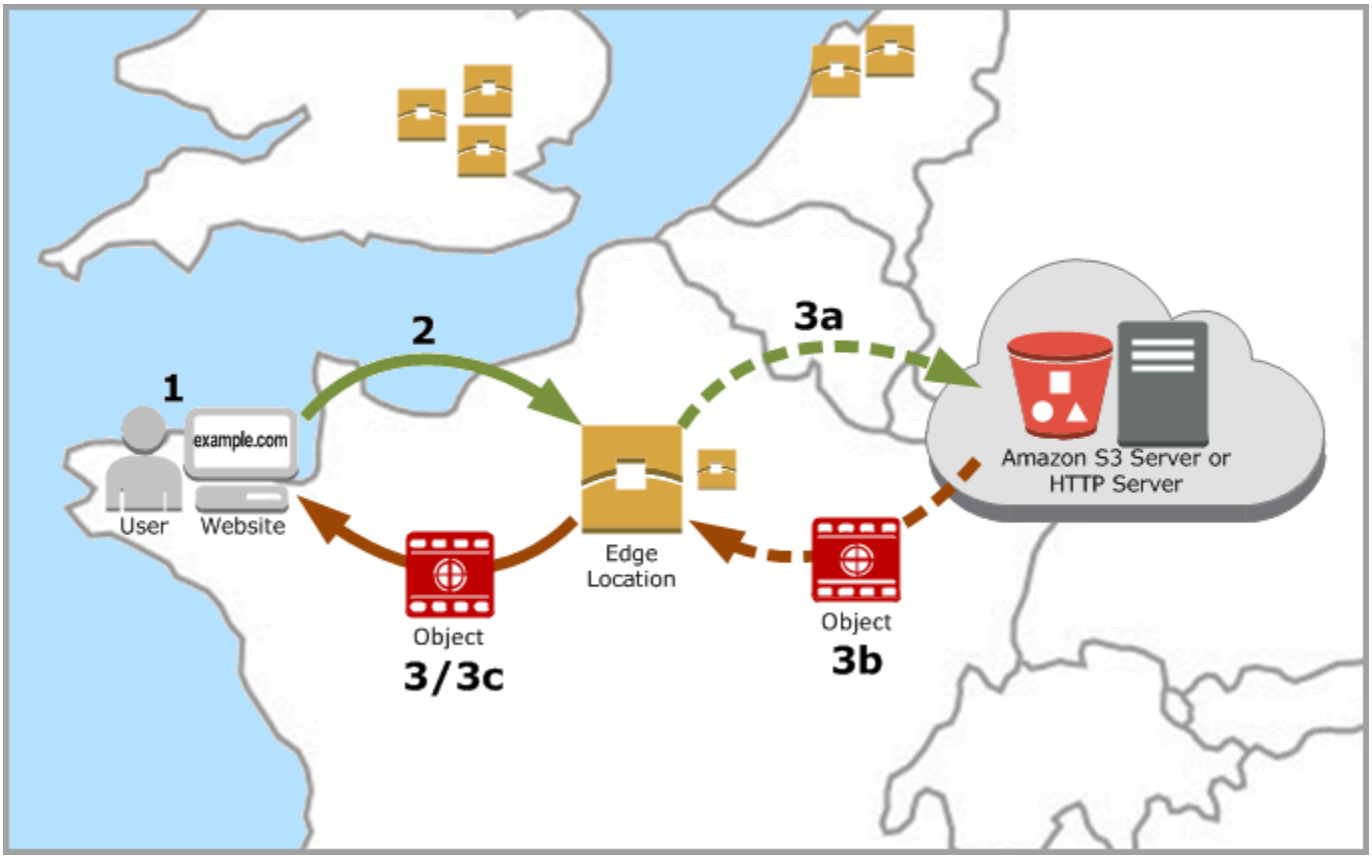
Topics

- How CloudFront delivers content to your users
- How CloudFront works with regional edge caches

How CloudFront delivers content to your users

After you configure CloudFront to deliver your content, here's what happens when users request your objects:

1. A user accesses your website or application and sends a request for an object, such as an image file or an HTML file.
2. DNS routes the request to the CloudFront POP (edge location) that can best serve the request, typically the nearest CloudFront POP in terms of latency.
3. CloudFront checks its cache for the requested object. If the object is in the cache, CloudFront returns it to the user. If the object is *not* in the cache, CloudFront does the following:
   a. CloudFront compares the request with the specifications in your distribution and forwards the request to your origin server for the corresponding object—for example, to your Amazon S3 bucket or your HTTP server.
   b. The origin server sends the object back to the edge location.
   c. As soon as the first byte arrives from the origin, CloudFront begins to forward the object to the user. CloudFront also adds the object to the cache for the next time someone requests it.



How CloudFront works with regional edge caches

CloudFront points of presence (also known as *POPs* or *edge locations*) make sure that popular content can be served quickly to your viewers. CloudFront also has *regional edge caches* that bring more of your content closer to your viewers, even when the content is not popular enough to stay at a POP, to help improve performance for that content.

Regional edge caches help with all types of content, particularly content that tends to become less popular over time. Examples include user-generated content, such as video, photos, or artwork; e-commerce assets such as product photos and videos; and news and event-related content that might suddenly find new popularity.

How regional caches work

Regional edge caches are CloudFront locations that are deployed globally, close to your viewers. They're located between your origin server and the POPs—global edge locations that serve content directly to viewers. As objects become less popular, individual POPs might remove those objects to make room for more popular content. Regional edge caches have a larger cache than an individual POP, so objects remain in the cache longer at the nearest regional edge cache location.

This helps keep more of your content closer to your viewers, reducing the need for CloudFront to go back to your origin server, and improving overall performance for viewers.

When a viewer makes a request on your website or through your application, DNS routes the request to the POP that can best serve the user's request. This location is typically the nearest CloudFront edge location in terms of latency. In the POP, CloudFront checks its cache for the requested object. If the object is in the cache, CloudFront returns it to the user. If the object is not in the cache, the POP typically goes to the nearest regional edge cache to fetch it. For more information about when the POP skips the regional edge cache and goes directly to the origin, see the following note.

In the regional edge cache location, CloudFront again checks its cache for the requested object. If the object is in the cache, CloudFront forwards it to the POP that requested it. As soon as the first byte arrives from regional edge cache location, CloudFront begins to forward the object to the user. CloudFront also adds the object to the cache in the POP for the next time someone requests it.

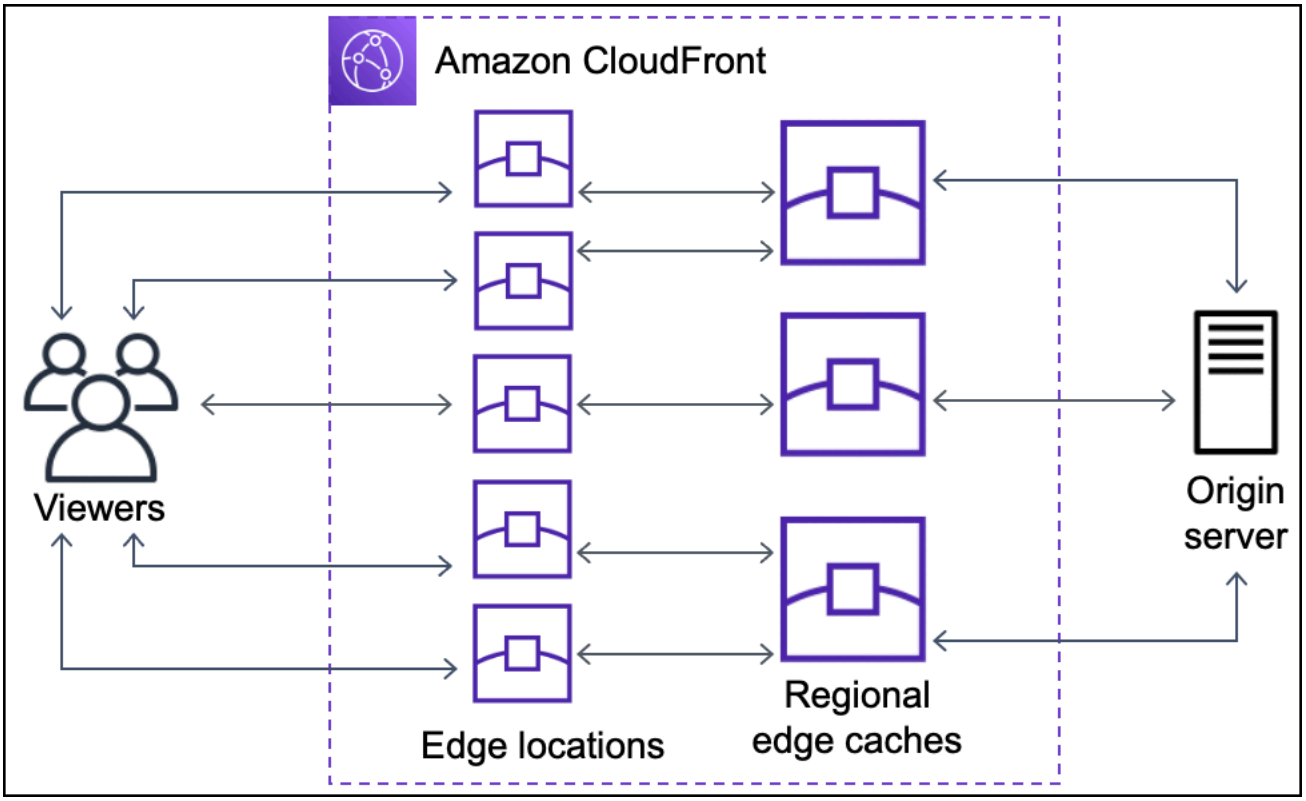
For objects not cached at either the POP or the regional edge cache location, CloudFront compares the request with the specifications in your distributions and forwards the request to the origin server. After your origin server sends the object back to the regional edge cache location, it is forwarded to the POP, and then CloudFront forwards it to the user. In this case, CloudFront also adds the object to the cache in the regional edge cache location in addition to the POP for the next time a viewer requests it. This makes sure that all of the POPs in a region share a local cache, eliminating multiple requests to origin servers. CloudFront also keeps persistent connections with origin servers so objects are fetched from the origins as quickly as possible.

Note

- Regional edge caches have feature parity with POPs. For example, a cache invalidation request removes an object from both POP caches and regional edge caches before it expires. The next time a viewer requests the object, CloudFront returns to the origin to fetch the latest version of the object.
- Proxy HTTP methods (`PUT`, `POST`, `PATCH`, `OPTIONS`, and `DELETE`) go directly to the origin from the POPs and do not proxy through the regional edge caches.
- Dynamic requests, as determined at request time, do not flow through regional edge caches, but go directly to the origin.

- When the origin is an Amazon S3 bucket and the request's optimal regional edge cache is in the same Amazon Web Services Region as the S3 bucket, the POP skips the regional edge cache and goes directly to the S3 bucket.

The following diagram illustrates how requests and responses flow through CloudFront edge locations and regional edge caches.



Locations and IP address ranges of CloudFront edge servers

For a list of the locations of CloudFront edge servers, see the Amazon CloudFront Global Edge Network page.

Amazon Web Services (Amazon) publishes its current IP address ranges in JSON format. To view the current ranges, download ip-ranges.json. For more information, see Amazon IP address ranges in the *Amazon Web Services General Reference*.

To find the IP address ranges that are associated with CloudFront edge servers, search ip-ranges.json for the following string:

```
"region": "GLOBAL",
```

```
"service": "CLOUDFRONT"
```

Alternatively, you can view only the CloudFront IP ranges at https://d7uri8nf7uskq.cloudfront.net/tools/list-cloudfront-ips.

Use the CloudFront managed prefix list

The CloudFront managed prefix list contains the IP address ranges of all of CloudFront's globally distributed origin-facing servers. If your origin is hosted on Amazon and protected by an Amazon VPC security group, you can use the CloudFront managed prefix list to allow inbound traffic to your origin only from CloudFront's origin-facing servers, preventing any non-CloudFront traffic from reaching your origin. CloudFront maintains the managed prefix list so it's always up to date with the IP addresses of all of CloudFront's global origin-facing servers. With the CloudFront managed prefix list, you don't need to read or maintain a list of IP address ranges yourself.

For example, imagine that your origin is an Amazon EC2 instance in the Europe (London) Region (`eu-west-2`). If the instance is in a VPC, you can create a security group rule that allows inbound HTTPS access from the CloudFront managed prefix list. This allows all of CloudFront's global origin-facing servers to reach the instance. If you remove all other inbound rules from the security group, you prevent any non-CloudFront traffic from reaching the instance.

The CloudFront managed prefix list is named **com.amazonaws.global.cloudfront.origin-facing**. The CloudFront managed prefix list is not available in the China Regions. For more information, see Use an Amazon-managed prefix list in the *Amazon VPC User Guide*.

> **Important**
>
> The CloudFront managed prefix list is unique in how it applies to Amazon VPC quotas. For more information, see Amazon-managed prefix list weight in the *Amazon VPC User Guide*.

Accessing CloudFront

You can access Amazon CloudFront in the following ways:

- **Amazon Web Services Management Console** – The procedures throughout this guide explain how to use the Amazon Web Services Management Console to perform tasks.
- **Amazon SDKs** – If you're using a programming language that Amazon provides an SDK for, you can use an SDK to access CloudFront. SDKs simplify authentication, integrate easily with your

development environment, and provide access to CloudFront commands. For more information, see Tools for Amazon Web Services.

- **CloudFront API** – If you're using a programming language that an SDK isn't available for, see the Amazon CloudFront API Reference for information about API actions and about how to make API requests.
- **Amazon Command Line Interface** – For more information, see Getting Set Up with the Amazon Command Line Interface in the *Amazon Command Line Interface User Guide*.
- **Amazon Tools for Windows PowerShell** – For more information, see Setting up the Amazon Tools for Windows PowerShell in the *Amazon Tools for Windows PowerShell User Guide*.

Using CloudFront with an Amazon SDK

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.

SDK documentation
Amazon SDK for Java
Amazon SDK for JavaScript
Amazon SDK for .NET
Amazon SDK for PHP
Amazon SDK for Python (Boto3)
Amazon SDK for Ruby
Amazon SDK for SAP ABAP

Getting started with Amazon CloudFront

Get started with the basic steps to deliver your content with CloudFront by creating a basic distribution or a secure static website.

Topics

- Setting up
- Getting started with a basic CloudFront distribution
- Getting started with a secure static website

Setting up

This topic describes preliminary steps, such as creating an Amazon Web Services account, to prepare you to use Amazon CloudFront.

Topics

- Sign up for an Amazon Web Services account
- Secure IAM users
- Set up the Amazon Command Line Interface or Amazon Tools for Windows PowerShell
- Download an Amazon SDK

Sign up for an Amazon Web Services account

If you do not have an Amazon Web Services account, use the following procedure to create one.

To sign up for Amazon Web Services

1. Open http://www.amazonaws.cn/ and choose **Sign Up**.
2. Follow the on-screen instructions.

Amazon sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to http://www.amazonaws.cn/ and choosing **My Account**.

Secure IAM users

After you sign up for an Amazon Web Services account, safeguard your administrative user by turning on multi-factor authentication (MFA). For instructions, see Enable a virtual MFA device for an IAM user (console) in the *IAM User Guide*.

To give other users access to your Amazon Web Services account resources, create IAM users. To secure your IAM users, turn on MFA and only give the IAM users the permissions needed to perform their tasks.

For more information about creating and securing IAM users, see the following topics in the *IAM User Guide*:

- Creating an IAM user in your Amazon Web Services account
- Access management for Amazon resources
- Example IAM identity-based policies

Set up the Amazon Command Line Interface or Amazon Tools for Windows PowerShell

The Amazon Command Line Interface (Amazon CLI) is a unified tool for managing Amazon services. For information about how to install and configure the Amazon CLI, see Getting Set Up with the Amazon Command Line Interface in the *Amazon Command Line Interface User Guide*.

If you have experience with Windows PowerShell, you might prefer to use Amazon Tools for Windows PowerShell. For more information, see Setting up the Amazon Tools for Windows PowerShell in the *Amazon Tools for Windows PowerShell User Guide*.

Download an Amazon SDK

If you're using a programming language that Amazon provides an SDK for, we recommend that you use an SDK instead of the Amazon CloudFront API. The SDKs make authentication simpler, integrate easily with your development environment, and provide easy access to CloudFront commands. For more information, see Tools to Build on Amazon.

Getting started with a basic CloudFront distribution

The procedures in this section show you how to use CloudFront to set up a basic configuration that does the following:

- Creates a bucket to use as your distribution origin.
- Stores the original versions of your objects in an Amazon Simple Storage Service (Amazon S3) bucket.
- Uses origin access control (OAC) to send authenticated requests to your Amazon S3 origin. OAC sends requests through CloudFront to prevent viewers from accessing your S3 bucket directly. For more information about OAC, see Restricting access to an Amazon Simple Storage Service origin.
- Uses the CloudFront domain name in URLs for your objects (for example, `https://d111111abcdef8.cloudfront.net/index.html`).
- Keeps your objects in CloudFront edge locations for the default duration of 24 hours (the minimum duration is 0 seconds).

Most of these options are customizable. For information about how to customize your CloudFront distribution options, see Steps for creating a distribution (overview).

Topics

- Prerequisites
- Step 1: Create an Amazon S3 bucket
- Step 2: Upload the content to the bucket
- Step 3: Create a CloudFront distribution that uses an Amazon S3 origin with OAC
- Step 4: Access your content through CloudFront
- Step 5: Clean up
- Tips

Prerequisites

Before you begin, make sure that you've completed the steps in Setting up.

Step 1: Create an Amazon S3 bucket

An Amazon S3 bucket is a container for files (objects) or folders. CloudFront can distribute almost any type of file for you when an S3 bucket is the source. For example, CloudFront can distribute text, images, and videos. There is no maximum for the amount of data that you can store on Amazon S3.

For this tutorial, you create an S3 bucket with the provided sample `hello world` files that you will use to create a basic webpage.

To create a bucket

1. Sign in to the Amazon Web Services Management Console and open the Amazon S3 console at https://console.amazonaws.cn/s3/.
2. We recommend that you use our Hello World sample for this Getting started. Download the *hello world* webpage: hello-world-html.zip. Unzip it and save the `css` folder and `index` file in a convenient location, such as the desktop where you are running your browser.
3. Choose **Create bucket**.
4. Enter a unique **Bucket name** that conforms to the General purpose buckets naming rules in the *Amazon Simple Storage Service User Guide*.
5. For **Region**, we recommend choosing an Amazon Web Services Region that is geographically close to you. (This reduces latency and costs.)

   - Choosing a different Region works, too. You might do this to address regulatory requirements, for example.
6. Leave all other settings at their defaults, and then choose **Create bucket**.

Step 2: Upload the content to the bucket

After you create your Amazon S3 bucket, upload the contents of the unzipped `hello world` file to it. (You downloaded and unzipped this file in Step 1: Create an Amazon S3 bucket.)

To upload the content to Amazon S3

1. In the **General purpose buckets** section, choose the name of your new bucket.
2. Choose **Upload**.
3. On the **Upload** page, drag the `css` folder and `index` file into the drop area.

4. Leave all other settings at their defaults, and then choose **Upload**.

Step 3: Create a CloudFront distribution that uses an Amazon S3 origin with OAC

For this tutorial, you will create a CloudFront distribution that uses an Amazon S3 origin with origin access control (OAC). OAC helps you securely send authenticated requests to your Amazon S3 origin. For more information about OAC, see Restricting access to an Amazon Simple Storage Service origin.

To create a CloudFront distribution with an Amazon S3 origin that uses OAC

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose **Create distribution**.
3. For **Origin**, **Origin domain**, choose the S3 bucket that you created for this tutorial.
4. For **Origin**, **Origin access**, select **Origin access control settings (recommended)**.
5. For **Origin access control**, choose **Create new OAC**.
6. In the **Create new OAC** pane, keep the default settings and choose **Create**.
7. For **Web Application Firewall (WAF)**, select one of the options.
8. For all other sections and settings, accept the default values. For more information about these options, see Distribution settings.
9. Choose **Create distribution**.
10. In **The S3 bucket policy needs to be updated** banner, read the message and choose **Copy policy**.
11. In the same banner, choose the link to **Go to S3 bucket permissions to update policy**. (This takes you to your bucket detail page in the Amazon S3 console.)
12. For **Bucket policy**, choose **Edit**.
13. In the **Edit statement** field, paste the policy that you copied in step 10.
14. Choose **Save changes**.
15. Return to the CloudFront console and review the **Details** section for your new distribution. When your distribution is done deploying, the **Last modified** field changes from **Deploying** to a date and time.
16. Record the domain name that CloudFront assigns to your distribution. It looks similar to the following: `d111111abcdef8.cloudfront.net`.

Before using the distribution and S3 bucket from this tutorial in a production environment, make sure to configure it to meet your specific needs. For information about configuring access in a production environment, see *Configuring secure access and restricting access to content*.

Step 4: Access your content through CloudFront

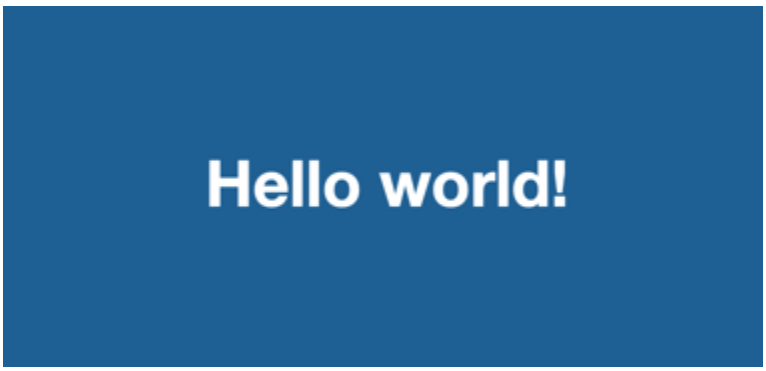
To access your content through CloudFront, combine the domain name for your CloudFront distribution with the main page for your content. (You recorded your distribution domain name in Step 3: Create a CloudFront distribution that uses an Amazon S3 origin with OAC.)

- Your distribution domain name might look like this: `d111111abcdef8.cloudfront.net`.
- The path to the main page of a website is typically `/index.html`.

Therefore, the URL to access your content through CloudFront might look like this:

`https://d111111abcdef8.cloudfront.net/index.html`.

If you followed the previous steps and used the *hello world* webpage, you should see the following content:



When you upload more content to this S3 bucket, you can access the content through CloudFront by combining the CloudFront distribution domain name with the path to the object in the S3 bucket. For example, if you upload a new file named `new-page.html` to the root of your S3 bucket, the URL looks like this:

`https://d111111abcdef8.cloudfront.net/new-page.html`.

Step 5: Clean up

If you created your distribution and S3 bucket only as a learning exercise, delete them so that you no longer accrue charges. Delete the distribution first. For more information, see the following links:

- Deleting a distribution
- Deleting a bucket

Tips

This Getting started tutorial provides a minimal framework for creating a distribution. We recommend that you explore the following enhancements:

- By default, the files (objects) in the Amazon S3 bucket are set up as private. Only the Amazon Web Services account that created the bucket has permission to read or write the files. If you want to allow anyone to access the files in your Amazon S3 bucket using CloudFront URLs, you must grant public read permissions to the objects.
- You can use the CloudFront private content feature to restrict access to the content in the Amazon S3 buckets. For more information about distributing private content, see Serving private content with signed URLs and signed cookies.
- You can configure your CloudFront distribution to use a custom domain name (for example, `www.example.com` instead of `d111111abcdef8.cloudfront.net`). For more information, see Using custom URLs.
- This tutorial uses an Amazon S3 origin with origin access control (OAC). However, you can't use OAC if your origin is an S3 bucket configured as a website endpoint. If that's the case, you must set up your bucket with CloudFront as a custom origin. For more information, see Using an Amazon S3 bucket that's configured as a website endpoint. For more information about OAC, see Restricting access to an Amazon Simple Storage Service origin.

Getting started with a secure static website

You can get started with Amazon CloudFront by using the solution described in this topic to create a secure static website for your domain name. A *static website* uses only static files—like HTML, CSS, JavaScript, images, and videos—and doesn't need servers or server-side processing. With this solution, your website gets the following benefits:

- **Uses the durable storage of Amazon Simple Storage Service (Amazon S3)** – This solution creates an Amazon S3 bucket to host your static website's content. To update your website, just upload your new files to the S3 bucket.
- **Is sped up by the Amazon CloudFront content delivery network** – This solution creates a CloudFront distribution to serve your website to viewers with low latency. The distribution is configured with an origin access identity to make sure that the website is accessible only through CloudFront, not directly from S3.
- **Is secured by HTTPS and additional security headers** – This solution creates an SSL/TLS certificate in Amazon Certificate Manager (ACM), and attaches it to the CloudFront distribution. This certificate enables the distribution to serve your domain's website securely with HTTPS.

  This solution also uses Lambda@Edge to add security headers to every server response. Security headers are a group of headers in the web server response that tell web browsers to take extra security precautions. For more information, refer to this blog post: Adding HTTP Security Headers Using Lambda@Edge and Amazon CloudFront.
- **Is configured and deployed with Amazon CloudFormation** – This solution uses an Amazon CloudFormation template to set up all the components, so you can focus more on your website's content and less on configuring components.

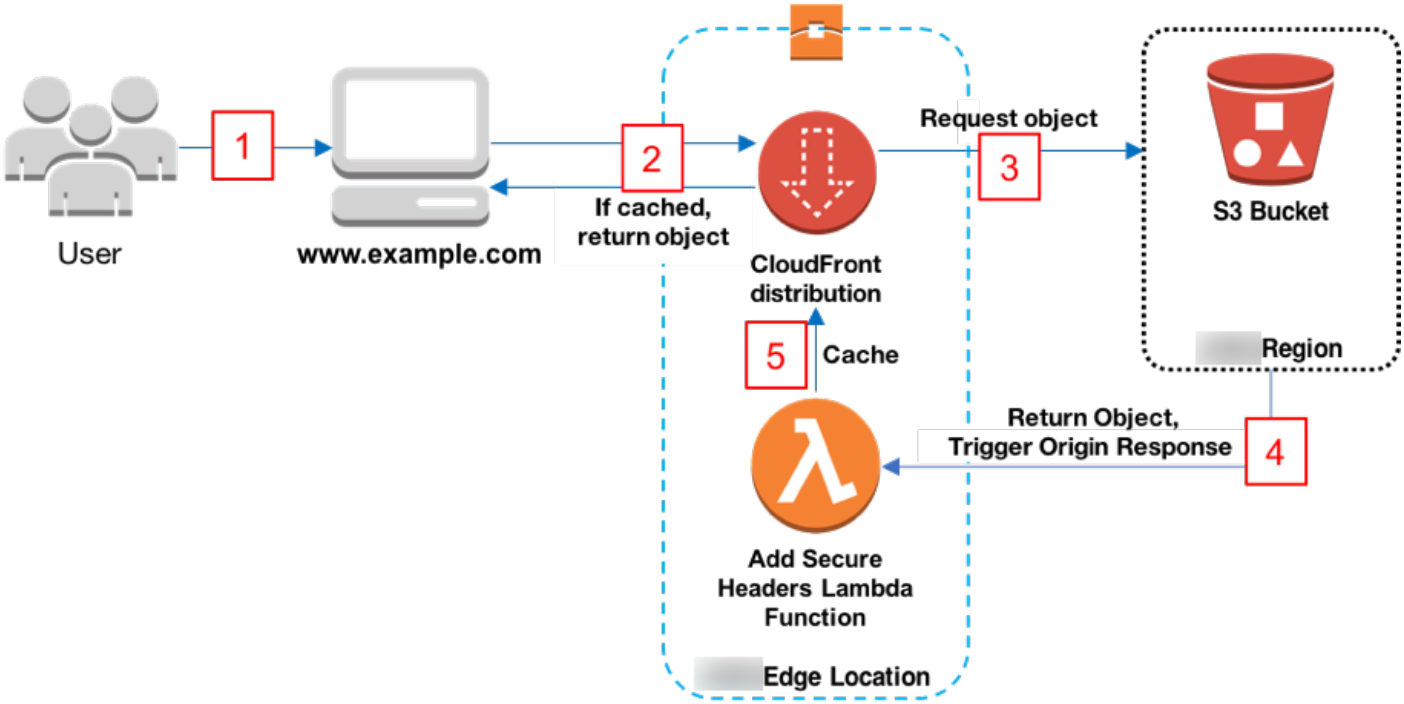
This solution is open source on GitHub. To view the code, submit a pull request, or open an issue, go to https://github.com/aws-samples/amazon-cloudfront-secure-static-site.

Topics

- Solution overview
- Deploying the solution

Solution overview

The following diagram shows an overview of how this static website solution works:



1. The viewer requests the website at www.example.com.
2. If the requested object is cached, CloudFront returns the object from its cache to the viewer.
3. If the object is not in CloudFront's cache, CloudFront requests the object from the origin (an S3 bucket).
4. S3 returns the object to CloudFront, which triggers the Lambda@Edge origin response event.
5. The object, including the security headers added by the Lambda@Edge function, is added to CloudFront's cache.
6. (Not shown) The objects is returned to the viewer. Subsequent requests for the object that come to the same CloudFront edge location are served from the CloudFront cache.

Deploying the solution

To deploy this secure static website solution, you can choose from either of the following options:

- Use the Amazon CloudFormation console to deploy the solution with default content, then upload your website content to Amazon S3.
- Clone the solution to your computer to add your website content. Then, deploy the solution with the Amazon Command Line Interface (Amazon CLI).

> **Note**
>
> You must use the US East (N. Virginia) Region to deploy the CloudFormation template.

Topics

- Prerequisites
- Using the Amazon CloudFormation console
- Cloning the solution locally
- Finding access logs

Prerequisites

To use this solution, you must have the following prerequisites:

- A registered domain name, such as example.com, that's pointed to an Amazon Route 53 hosted zone. The hosted zone must be in the same Amazon Web Services account where you deploy this solution. If you don't have a registered domain name, you can register one with Route 53. If you have a registered domain name but it's not pointed to a Route 53 hosted zone, configure Route 53 as your DNS service.
- Amazon Identity and Access Management (IAM) permissions to launch CloudFormation templates that create IAM roles, and permissions to create all the Amazon resources in the solution.

You are responsible for the costs incurred while using this solution. For more information about costs, see the pricing pages for each Amazon Web Service.

Using the Amazon CloudFormation console

To deploy using the CloudFormation console

1. Choose **Launch** to open this solution in the Amazon CloudFormation console. If necessary, sign in to your Amazon Web Services account.

2. The **Create stack** wizard opens in the CloudFormation console, with prepopulated fields that specify this solution's CloudFormation template.

   At the bottom of the page, choose **Next**.

3. On the **Specify stack details** page, enter values for the following fields:

   - **SubDomain** – Enter the subdomain to use for your website. For example, if the subdomain is *www*, your website is available at *www*.example.com. (Replace example.com with your domain name, as explained in the following bullet.)
   - **DomainName** – Enter your domain name, such as *example.com*. This domain must be pointed to a Route 53 hosted zone.
   - **HostedZoneId** – The Route 53 hosted zone ID of your domain name.

   When finished, choose **Next**.

4. (Optional) On the **Configure stack options** page, add tags and other stack options.

   When finished, choose **Next**.

5. On the **Review** page, scroll to the bottom of the page, then select the two boxes in the **Capabilities** section. These capabilities allow Amazon CloudFormation to create an IAM role that allows access to the stack's resources, and to name the resources dynamically.

6. Choose **Create stack**.

7. Wait for the stack to finish creating. The stack creates some nested stacks, and can take several minutes to finish. When it's finished, the **Status** changes to **CREATE_COMPLETE**.

   When the status is **CREATE_COMPLETE**, go to https://*www.example.com* to view your website (replace www.example.com with the subdomain and domain name that you specified in step 3). You should see the website's default content:

To replace the website's default content with your own

1. Open the Amazon S3 console at https://console.amazonaws.cn/s3/.

2. Choose the bucket whose name begins with **amazon-cloudfront-secure-static-site-s3bucketroot-**.

> **Note**
>
> Make sure to choose the bucket with **s3bucketroot** in its name, not **s3bucketlogs**. The bucket with **s3bucketroot** in its name contains the website content. The one with **s3bucketlogs** contains only log files.

3. Delete the website's default content, then upload your own.

> **Note**
>
> If you viewed your website with this solution's default content, then it's likely that some of the default content is cached in a CloudFront edge location. To make sure that viewers see your updated website content, *invalidate* the files to remove the cached copies from CloudFront edge locations. For more information, see Invalidating files.

Cloning the solution locally

Prerequisites

To add your website content before deploying this solution, you must package the solution's artifacts locally, which requires Node.js and npm. For more information, see https://www.npmjs.com/get-npm.

To add your website content and deploy the solution

1. Clone or download the solution from https://github.com/aws-samples/amazon-cloudfront-secure-static-site. After you clone or download it, open a command prompt or terminal and navigate to the `amazon-cloudfront-secure-static-site` folder.
2. Run the following command to install and package the solution's artifacts:

```
make package-static
```

3. Copy your website's content into the `www` folder, overwriting the default website content.
4. Run the following Amazon CLI command to create an Amazon S3 bucket to store the solution's artifacts. Replace *`example-bucket-for-artifacts`* with your own bucket name.

```
aws s3 mb s3://example-bucket-for-artifacts --region us-east-1
```

5. Run the following Amazon CLI command to package the solution's artifacts as an Amazon CloudFormation template. Replace *example-bucket-for-artifacts* with the name of the bucket that you created in the previous step.

```
aws cloudformation package \
    --region us-east-1 \
    --template-file templates/main.yaml \
    --s3-bucket example-bucket-for-artifacts \
    --output-template-file packaged.template
```

6. Run the following command to deploy the solution with Amazon CloudFormation, replacing the following values:

   - *your-CloudFormation-stack-name* – Replace with a name for the Amazon CloudFormation stack.
   - *example.com* – Replace with your domain name. This domain must be pointed to a Route 53 hosted zone in the same Amazon account.
   - *www* – Replace with the subdomain to use for your website. For example, if the subdomain is *www*, your website is available at www.example.com.

```
aws cloudformation deploy \
    --region us-east-1 \
    --stack-name your-CloudFormation-stack-name \
    --template-file packaged.template \
    --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND \
    --parameter-overrides DomainName=example.com SubDomain=www
```

7. Wait for the Amazon CloudFormation stack to finish creating. The stack creates some nested stacks, and can take several minutes to finish. When it's finished, the **Status** changes to **CREATE_COMPLETE**.

   When the status changes to **CREATE_COMPLETE**, go to https://www.example.com to view your website (replace www.example.com with the subdomain and domain name that you specified in the previous step). You should see your website's content.

Finding access logs

This solution enables access logs for the CloudFront distribution. Complete the following steps to locate the distribution's access logs.

To locate the distribution's access logs

1. Open the Amazon S3 console at https://console.amazonaws.cn/s3/.
2. Choose the bucket whose name begins with **amazon-cloudfront-secure-static-site-s3bucketlogs-**.

> **Note**
>
> Make sure to choose the bucket with **s3bucketlogs** in its name, not **s3bucketroot**. The bucket with **s3bucketlogs** in its name contains log files. The one with **s3bucketroot** contains the website content.

3. The folder named **cdn** contains the CloudFront access logs.

Working with distributions

You create an Amazon CloudFront distribution to tell CloudFront from where you want content to be delivered, and the details about how to track and manage content delivery. The following topics explain some basics about CloudFront distributions and provide detailed information about the settings you can choose to configure your distributions to meet your business needs.

Topics

- Overview of distributions
- Creating, updating, and deleting distributions
- Using CloudFront continuous deployment to safely test CDN configuration changes
- Using various origins with CloudFront distributions
- Using custom URLs by adding alternate domain names (CNAMEs)
- Using WebSockets with CloudFront distributions

Overview of distributions

When using CloudFront to distribute your content, you create a distribution and choose from the following configuration settings:

- **Your content origin**—The Amazon S3 bucket, AWS Elemental MediaPackage channel, AWS Elemental MediaStore container, Elastic Load Balancing load balancer, or HTTP server from which CloudFront gets the files to distribute. You can specify any combination of up to 25 origins for a single distribution.
- **Access**—Whether you want access to the files to be available to everyone or restricted to some users.
- **Security**—Whether you want to enable Amazon WAF protection and require users to use HTTPS to access your content.
- **Cache key**—Which values, if any, you want to include in the *cache key*. The cache key uniquely identifies each file in the cache for a given distribution.
- **Origin request settings**—Whether you want CloudFront to include HTTP headers, cookies, or query strings in requests that it sends to your origin.
- **Geographic restrictions**—Whether you want CloudFront to prevent users in selected countries from accessing your content.

- **Logs**—Whether you want CloudFront to create standard logs or real-time logs that show viewer activity.

For the current maximum number of distributions that you can create for each Amazon account, see General quotas on distributions. There is no maximum number of files that you can serve per distribution.

You can use distributions to serve the following content over HTTP or HTTPS:

- Static and dynamic download content, such as HTML, CSS, JavaScript, and image files, using HTTP or HTTPS.
- Video on demand in different formats, such as Apple HTTP Live Streaming (HLS) and Microsoft Smooth Streaming. For more information, see Delivering video on demand (VOD) with CloudFront.
- A live event, such as a meeting, conference, or concert, in real time. For live streaming, you can create the distribution automatically by using an Amazon CloudFormation stack. For more information, see Delivering live streaming video with CloudFront and Amazon Media Services.

For information about creating a distribution, see Steps for creating a distribution (overview).

Actions you can use with distributions

The following table lists the CloudFront actions that you can take to work with distributions. The table provides links to the corresponding documentation on how to do the actions with the CloudFront console and the CloudFront APIs.

Action	Using the CloudFront console	Using the CloudFront API
Create a distribution	See Steps for creating a distribution (overview)	Go to CreateDistribution
List your distributions	See Updating a distribution	Go to ListDistributions
	See Updating a distribution	Go to GetDistribution

Action	Using the CloudFront console	Using the CloudFront API
Get all information about a distribution		
Get the distribution configuration	See Updating a distribution	Go to GetDistributionConfig
Update a distribution	See Updating a distribution	Go to UpdateDistribution
Delete a distribution	See Deleting a distribution	Go to DeleteDistribution

Required API fields for creating and updating distributions

When you update a distribution by using the UpdateDistribution CloudFront API action, there are more required fields than when you create a distribution by using CreateDistribution. To update a distribution, complete the following steps:

1. Use GetDistribution to get the current configuration of the distribution that you want to update.
2. Modify the fields in the distribution configuration that you want to update. Also, rename the `ETag` field to `IfMatch`, but don't change the field's value.
3. Use UpdateDistribution to update the distribution, providing the entire distribution configuration, including the fields that you modified and those that you didn't.

The following tables summarizes the fields that are required for creating and for updating a distribution.

DistributionConfig

Members	Required in CreateDistribution API call	Required in UpdateDistribution API call
CallerReference	Yes	Yes

Members	Required in CreateDistribution API call	Required in UpdateDistribution API call
Aliases	-	Yes (this field is required, but a quantity of `0` with no items is valid)
DefaultRootObject	-	Yes (this field is required, but an empty string is a valid value)
Origins	Yes	Yes
OriginGroups	-	-
DefaultCacheBehavior	Yes	Yes
CacheBehaviors	-	Yes (this field is required, but a quantity of `0` with no items is valid)
CustomErrorResponses	-	Yes (this field is required, but a quantity of `0` with no items is valid)
Comment	Yes (this field is required, but an empty string is a valid value)	Yes (this field is required, but an empty string is a valid value)
Logging	-	Yes
PriceClass	-	Yes
Enabled	Yes	Yes
ViewerCertificate	-	Yes

Members	Required in CreateDistribution API call	Required in UpdateDistribution API call
Restrictions	-	Yes (this field is required, but a `RestrictionsType` of `none` and a quantity of `0` with no items is valid)
WebACLId	-	Yes (this field is required, but an empty string is a valid value)
HttpVersion	-	Yes
IsIPV6Enabled	-	-

CacheBehavior (including DefaultCacheBehavior)

Members	Required in CreateDistribution API call	Required in UpdateDistribution API call
PathPattern (this field does not apply to DefaultCacheBehavior)	Yes	Yes
TargetOriginId	Yes	Yes
TrustedSigners	-	-
TrustedKeyGroups	-	-
ViewerProtocolPolicy	Yes	Yes
AllowedMethods	-	Yes
SmoothStreaming	-	Yes

Members	Required in CreateDistribution API call	Required in UpdateDistribution API call
Compress	-	Yes
LambdaFunctionAssociations	-	Yes (this field is required, but a quantity of `0` with no items is valid)
FunctionAssociations	-	-
FieldLevelEncryptionId	-	Yes (this field is required, but an empty string is a valid value)
RealtimeLogConfigArn	-	-
CachePolicyId	Yes (`CachePolicyId` is not required when you use the following deprecated fields, which is not recommended: `ForwardedValues`, `MinTTL`, `DefaultTTL`, and `MaxTTL`)	Yes (`CachePolicyId` is not required when you use the following deprecated fields, which is not recommended: `ForwardedValues`, `MinTTL`, `DefaultTTL`, and `MaxTTL`)
OriginRequestPolicyId	-	-
ResponseHeadersPolicyId	-	-

Creating, updating, and deleting distributions

The following topics explain how to create, update, or delete an Amazon CloudFront distribution.

Topics

- Steps for creating a distribution (overview)
- Creating a distribution
- Values that you specify when you create or update a distribution
- Values that CloudFront displays in the console
- Testing a distribution
- Updating a distribution
- Tagging Amazon CloudFront distributions
- Deleting a distribution

Steps for creating a distribution (overview)

The following task list summarizes the process for creating a distribution.

To create a distribution

1. Create one or more Amazon S3 buckets, or configure HTTP servers as your origin servers. An *origin* is the location where you store the original version of your content. When CloudFront gets a request for your files, it goes to the origin to get the files that it distributes at edge locations. You can use any combination of Amazon S3 buckets and HTTP servers as your origin servers.

   If you use Amazon S3, the name of your bucket must be all lowercase and cannot contain spaces.

   If you use an Amazon EC2 server or another custom origin, review Using Amazon EC2 (or another custom origin).

   For the current maximum number of origins that you can create for a distribution, or to request a higher quota, see General quotas on distributions.
2. Upload your content to your origin servers. You make your objects publicly readable, or you can use CloudFront signed URLs to restrict access to your content.

> **Important**
>
> You are responsible for ensuring the security of your origin server. You must ensure that CloudFront has permission to access the server and that the security settings safeguard your content.

3. Create your CloudFront distribution:

   - For more information about using the CloudFront console to create a distribution, see Creating a distribution.
   - For information about creating a distribution using the CloudFront APIs, go to CreateDistribution in the *Amazon CloudFront API Reference*.

4. (Optional) If you use the CloudFront console to create your distribution, create more cache behaviors or origins for the distribution. For more information about behaviors and origins, see To update a CloudFront distribution.

5. Test your distribution. For more information about testing, see Testing a distribution.

6. Develop your website or application to access your content using the domain name that CloudFront returned after you created your distribution in Step 3. For example, if CloudFront returns d111111abcdef8.cloudfront.net as the domain name for your distribution, the URL for the file `image.jpg` in an Amazon S3 bucket or in the root directory on an HTTP server is `https://d111111abcdef8.cloudfront.net/image.jpg`.

   If you specified one or more alternate domain names (CNAMEs) when you created your distribution, you can use your own domain name. In that case, the URL for `image.jpg` might be `https://www.example.com/image.jpg`.

   Note the following:

   - If you want to use signed URLs to restrict access to your content, see Serving private content with signed URLs and signed cookies.
   - If you want to serve compressed content, see Serving compressed files.
   - For information about CloudFront request and response behavior for Amazon S3 and custom origins, see Request and response behavior.

Creating a distribution

This topic explains how to use the CloudFront console to create a distribution.

For information about using the CloudFront APIs to create a distribution, see Create Distribution in the *Amazon CloudFront API Reference*.

For information about updating distributions, see Updating a distribution, later in this section.

To see the current maximum number of distributions that you can create for each Amazon account, or to request a higher quota (formerly known as limit), see General quotas on distributions.

To create a distribution (console)

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Distributions**, then choose **Create distribution**.
3. Specify settings for the distribution. For more information, see Values that you specify when you create or update a distribution.
4. Save your changes.
5. After CloudFront creates your distribution, the value of the **Status** column for your distribution will change from **Deploying** to the date and time that the distribution is deployed. If you chose to enable the distribution, it will be ready to process requests at this time.

   The domain name that CloudFront assigns to your distribution appears in the list of distributions. (It also appears on the **General** tab for a selected distribution.)

   > **Tip**
   >
   > You can use an alternate domain name, instead of the name assigned to you by CloudFront; by following the steps in Using custom URLs by adding alternate domain names (CNAMEs).

6. When your distribution is deployed, confirm that you can access your content using your new CloudFront URL or CNAME. For more information, see Testing a distribution.

To update a distribution (for example, to add or change cache behaviors), see Updating a distribution.

Values that you specify when you create or update a distribution

When you use the CloudFront console to create a new distribution or update an existing distribution, you specify the following values:

the section called “Origin settings”

- the section called “Origin domain”
- the section called “Protocol (custom origins only)”
- the section called “Origin path”
- the section called “Name”
- the section called “Origin access (Amazon S3 origins only)”
- the section called “Add custom header”
- the section called “Enable Origin Shield”
- the section called “Connection attempts”
- the section called “Connection timeout”
- the section called “Response timeout (custom origins only)”
- the section called “Keep-alive timeout (custom origins only)”

Cache behavior settings

The following values apply to the **Default Cache Behavior Settings** when you create a distribution. They also apply to other cache behaviors that you create later.

- Path pattern
- Origin or origin group (Applies only when you create or update a cache behavior for an existing distribution)
- Viewer protocol policy
- Allowed HTTP methods
- Field-level encryption config
- Cached HTTP methods
- Cache based on selected request headers
- Allowlist headers (Applies only when you choose **Allowlist** for **Cache Based on Selected Request Headers**)

- Object caching
- Minimum TTL
- Maximum TTL
- Default TTL
- Forward cookies
- Allowlist cookies (Applies only when you choose **Allowlist** for **Forward Cookies**)
- Query string forwarding and caching
- Query string allowlist (Applies only when you choose **Forward all, cache based on allowlist** for **Query String Forwarding and Caching**)
- Smooth Streaming
- Restrict viewer access (use signed URLs or signed cookies)
- Trusted signers (Applies only when you choose **Yes** for **Restrict Viewer Access (Use Signed URLs or Signed Cookies**)
- Amazon Web Services account numbers (Applies only when you choose **Specify Accounts** for **Trusted Signers**)
- Compress objects automatically

The following values apply to **Lambda Function Associations**.

- CloudFront event
- Lambda function ARN
- Include body

Distribution settings

- Price class
- Amazon WAF web ACL
- Alternate domain names (CNAMEs)
- SSL certificate
- Custom SSL client support (Applies only when you choose **Custom SSL Certificate (example.com)** for **SSL Certificate**)
- Security policy (Minimum SSL/TLS version)

- Supported HTTP versions
- Default root object
- Logging
- Bucket for logs
- Log prefix
- Cookie logging
- Enable IPv6
- Comment
- Distribution state

Custom error pages and error caching

- HTTP error code
- Response page path
- HTTP response code
- Error caching minimum TTL (seconds)

Geographic restrictions

For more information about creating or updating a distribution by using the CloudFront console, see the section called "Creating a distribution" or the section called "Updating a distribution".

Origin settings

When you use the CloudFront console to create or update a distribution, you provide information about one or more locations, known as *origins*, where you store the original versions of your web content. CloudFront gets your web content from your origins and serves it to viewers via a worldwide network of edge servers.

For the current maximum number of origins that you can create for a distribution, or to request a higher quota, see the section called "General quotas on distributions".

If you want to delete an origin, you must first edit or delete the cache behaviors that are associated with that origin.

> **Important**
>
> If you delete an origin, confirm that files that were previously served by that origin are available in another origin and that your cache behaviors are now routing requests for those files to the new origin.

When you create or update a distribution, you specify the following values for each origin.

Origin domain

The origin domain is the DNS domain name of the Amazon S3 bucket or HTTP server from which you want CloudFront to get objects for this origin, for example:

- **Amazon S3 bucket** – *DOC-EXAMPLE-BUCKET*.s3.*us-west-2*.amazonaws.com

  > **Note**
  >
  > If you recently created the S3 bucket, the CloudFront distribution might return HTTP `307 Temporary Redirect` responses for up to 24 hours. It can take up to 24 hours for the S3 bucket name to propagate to all Amazon Regions. When the propagation is complete, the distribution automatically stops sending these redirect responses; you don't need to take any action. For more information, see Why am I getting an HTTP 307 Temporary Redirect response from Amazon S3? and Temporary Request Redirection.

- **Amazon S3 bucket configured as a website** – *DOC-EXAMPLE-BUCKET*.s3-website.*us-west-2*.amazonaws.com
- **MediaStore container** – *examplemediastore*.data.mediastore.*us-west-1*.amazonaws.com
- **MediaPackage endpoint** – *examplemediapackage*.mediapackage.*us-west-1*.amazonaws.com
- **Amazon EC2 instance** – *ec2-203-0-113-25*.compute-1.amazonaws.com
- **Elastic Load Balancing load balancer** – *example-load-balancer-1234567890*.*us-west-2*.elb.amazonaws.com
- **Your own web server** – `https://www.example.com`

Choose the domain name in the **Origin domain** field, or type the name. The domain name is not case-sensitive.

If your origin is an Amazon S3 bucket, note the following:

- If the bucket is configured as a website, enter the Amazon S3 static website hosting endpoint for your bucket; don't select the bucket name from the list in the **Origin domain** field. The static website hosting endpoint appears in the Amazon S3 console, on the **Properties** page under **Static website hosting**. For more information, see the section called "Using an Amazon S3 bucket that's configured as a website endpoint".
- If you configured Amazon S3 Transfer Acceleration for your bucket, do not specify the `s3-accelerate` endpoint for **Origin domain**.
- If you're using a bucket from a different Amazon account and if the bucket is not configured as a website, enter the name, using the following format:

  bucket-name`.s3.`*region*`.amazonaws.com`

  If your bucket reside in a US Region, and you want Amazon S3 to route requests to a facility in northern Virginia, use the following format:

  bucket-name`.s3.us-east-1.amazonaws.com`
- The files must be publicly readable unless you secure your content in Amazon S3 by using a CloudFront origin access control. For more information about access control, see the section called "Restricting access to an Amazon Simple Storage Service origin".

> **Important**
>
> If the origin is an Amazon S3 bucket, the bucket name must conform to DNS naming requirements. For more information, go to Bucket restrictions and limitations in the *Amazon Simple Storage Service User Guide*.

When you change the value of **Origin domain** for an origin, CloudFront immediately begins replicating the change to CloudFront edge locations. Until the distribution configuration is updated in a given edge location, CloudFront continues to forward requests to the previous origin. As soon as the distribution configuration is updated in that edge location, CloudFront begins to forward requests to the new origin.

Changing the origin does not require CloudFront to repopulate edge caches with objects from the new origin. As long as the viewer requests in your application have not changed, CloudFront continues to serve objects that are already in an edge cache until the TTL on each object expires or until seldom-requested objects are evicted.

Origin path

If you want CloudFront to request your content from a directory in your origin, enter the directory path, beginning with a slash (/). CloudFront appends the directory path to the value of **Origin domain**, for example, **`cf-origin.example.com/production/images`**. Do not add a slash (/) at the end of the path.

For example, suppose you've specified the following values for your distribution:

- **Origin domain** – An Amazon S3 bucket named **`DOC-EXAMPLE-BUCKET`**
- **Origin path** – **`/production`**
- **Alternate domain names (CNAME)** – **`example.com`**

When a user enters `example.com/index.html` in a browser, CloudFront sends a request to Amazon S3 for `DOC-EXAMPLE-BUCKET/production/index.html`.

When a user enters `example.com/acme/index.html` in a browser, CloudFront sends a request to Amazon S3 for `DOC-EXAMPLE-BUCKET/production/acme/index.html`.

Name

A name is a string that uniquely identifies this origin in this distribution. If you create cache behaviors in addition to the default cache behavior, you use the name that you specify here to identify the origin that you want CloudFront to route a request to when the request matches the path pattern for that cache behavior.

Add custom header

If you want CloudFront to add custom headers whenever it sends a request to your origin, specify the header name and its value. For more information, see the section called "Adding custom headers to origin requests".

For the current maximum number of custom headers that you can add, the maximum length of a custom header name and value, and the maximum total length of all header names and values, see *Quotas*.

Enable Origin Shield

Choose **Yes** to enable CloudFront Origin Shield. For more information about Origin Shield, see the section called "Using Origin Shield".

Connection attempts

You can set the number of times that CloudFront attempts to connect to the origin. You can specify 1, 2, or 3 as the number of attempts. The default number (if you don't specify otherwise) is 3.

Use this setting together with **Connection timeout** to specify how long CloudFront waits before attempting to connect to the secondary origin or returning an error response to the viewer. By default, CloudFront waits as long as 30 seconds (3 attempts of 10 seconds each) before attempting to connect to the secondary origin or returning an error response. You can reduce this time by specifying fewer attempts, a shorter connection timeout, or both.

If the specified number of connection attempts fail, CloudFront does one of the following:

- If the origin is part of an origin group, CloudFront attempts to connect to the secondary origin. If the specified number of connection attempts to the secondary origin fail, then CloudFront returns an error response to the viewer.
- If the origin is not part of an origin group, CloudFront returns an error response to the viewer.

For a custom origin (including an Amazon S3 bucket that's configured with static website hosting), this setting also specifies the number of times that CloudFront attempts to get a response from the origin. For more information, see the section called "Response timeout (custom origins only)".

Connection timeout

The connection timeout is the number of seconds that CloudFront waits when trying to establish a connection to the origin. You can specify a number of seconds between 1 and 10 (inclusive). The default timeout (if you don't specify otherwise) is 10 seconds.

Use this setting together with **Connection attempts** to specify how long CloudFront waits before attempting to connect to the secondary origin or before returning an error response to the viewer. By default, CloudFront waits as long as 30 seconds (3 attempts of 10 seconds each) before attempting to connect to the secondary origin or returning an error response. You can reduce this time by specifying fewer attempts, a shorter connection timeout, or both.

If CloudFront doesn't establish a connection to the origin within the specified number of seconds, CloudFront does one of the following:

- If the specified number of **Connection attempts** is more than 1, CloudFront tries again to establish a connection. CloudFront tries up to 3 times, as determined by the value of **Connection attempts**.
- If all the connection attempts fail and the origin is part of an origin group, CloudFront attempts to connect to the secondary origin. If the specified number of connection attempts to the secondary origin fail, then CloudFront returns an error response to the viewer.
- If all the connection attempts fail and the origin is not part of an origin group, CloudFront returns an error response to the viewer.

Response timeout (custom origins only)

> **Note**
>
> This applies only to custom origins.

The origin response timeout, also known as the *origin read timeout* or *origin request timeout*, applies to both of the following values:

- How long (in seconds) CloudFront waits for a response after forwarding a request to the origin.
- How long (in seconds) CloudFront waits after receiving a packet of a response from the origin and before receiving the next packet.

The default timeout is 30 seconds. You can change the value to be from 1 to 60 seconds. If you need a timeout value outside that range, create a case in the Amazon Support Center Console.

> **Tip**
>
> If you want to increase the timeout value because viewers are experiencing HTTP 504 status code errors, consider exploring other ways to eliminate those errors before changing the timeout value. See the troubleshooting suggestions in the section called "HTTP 504 status code (Gateway Timeout)".

CloudFront behavior depends on the HTTP method in the viewer request:

- `GET` and `HEAD` requests – If the origin doesn't respond or stops responding within the duration of the response timeout, CloudFront drops the connection. CloudFront tries again to connect according to the value of the section called "Connection attempts".
- `DELETE`, `OPTIONS`, `PATCH`, `PUT`, and `POST` requests – If the origin doesn't respond for the duration of the read timeout, CloudFront drops the connection and doesn't try again to contact the origin. The client can resubmit the request if necessary.

Keep-alive timeout (custom origins only)

> **Note**
>
> This applies only to custom origins.

The keep-alive timeout is how long (in seconds) CloudFront tries to maintain a connection to your custom origin after it gets the last packet of a response. Maintaining a persistent connection saves the time that is required to re-establish the TCP connection and perform another TLS handshake for subsequent requests. Increasing the keep-alive timeout helps improve the request-per-connection metric for distributions.

> **Note**
>
> For the **Keep-alive timeout** value to have an effect, your origin must be configured to allow persistent connections.

The default timeout is 5 seconds. You can change the value to a number from 1 to 60 seconds. If you need a keep-alive timeout longer than 60 seconds, create a case in the Amazon Support Center Console.

Origin access (Amazon S3 origins only)

> **Note**
>
> This applies only to Amazon S3 bucket origins (those that are *not* using the S3 static website endpoint).

Choose **Origin access control settings (recommended)** if you want to make it possible to restrict access to an Amazon S3 bucket origin to only specific CloudFront distributions.

Choose **Public** if the Amazon S3 bucket origin is publicly accessible.

For more information, see the section called "Restricting access to an Amazon Simple Storage Service origin".

For information about how to require users to access objects on a custom origin by using only CloudFront URLs, see the section called " Restricting access to files on custom origins".

Protocol (custom origins only)

> **Note**
>
> This applies only to custom origins.

The protocol policy that you want CloudFront to use when fetching objects from your origin.

Choose one of the following values:

- **HTTP only:** CloudFront uses only HTTP to access the origin.

  > **Important**
  >
  > **HTTP only** is the default setting when the origin is an Amazon S3 static website hosting endpoint, because Amazon S3 doesn't support HTTPS connections for static website hosting endpoints. The CloudFront console does not support changing this setting for Amazon S3 static website hosting endpoints.

- **HTTPS only:** CloudFront uses only HTTPS to access the origin.
- **Match viewer:** CloudFront communicates with your origin using HTTP or HTTPS, depending on the protocol of the viewer request. CloudFront caches the object only once even if viewers make requests using both HTTP and HTTPS protocols.

  > **Important**
  >
  > For HTTPS viewer requests that CloudFront forwards to this origin, one of the domain names in the SSL/TLS certificate on your origin server must match the domain name

that you specify for **Origin domain**. Otherwise, CloudFront responds to the viewer requests with an HTTP status code 502 (Bad Gateway) instead of returning the requested object. For more information, see the section called "Requirements for using SSL/TLS certificates with CloudFront".

HTTP port

Note

This applies only to custom origins.

(Optional) You can specify the HTTP port on which the custom origin listens. Valid values include ports 80, 443, and 1024 to 65535. The default value is port 80.

Important

Port 80 is the default setting when the origin is an Amazon S3 static website hosting endpoint, because Amazon S3 only supports port 80 for static website hosting endpoints. The CloudFront console does not support changing this setting for Amazon S3 static website hosting endpoints.

HTTPS port

Note

This applies only to custom origins.

(Optional) You can specify the HTTPS port on which the custom origin listens. Valid values include ports 80, 443, and 1024 to 65535. The default value is port 443. When **Protocol** is set to **HTTP only**, you cannot specify a value for **HTTPS port**.

Minimum origin SSL protocol

> **Note**
>
> This applies only to custom origins.

Choose the minimum TLS/SSL protocol that CloudFront can use when it establishes an HTTPS connection to your origin. Lower TLS protocols are less secure, so we recommend that you choose the latest TLS protocol that your origin supports. When **Protocol** is set to **HTTP only**, you cannot specify a value for **Minimum origin SSL protocol**.

If you use the CloudFront API to set the TLS/SSL protocol for CloudFront to use, you cannot set a minimum protocol. Instead, you specify all of the TLS/SSL protocols that CloudFront can use with your origin. For more information, see OriginSslProtocols in the *Amazon CloudFront API Reference*.

Cache behavior settings

By setting the cache behavior, you can configure a variety of CloudFront functionality for a given URL path pattern for files on your website. For example, one cache behavior might apply to all `.jpg` files in the `images` directory on a web server that you're using as an origin server for CloudFront. The functionality that you can configure for each cache behavior includes:

- The path pattern
- If you have configured multiple origins for your CloudFront distribution, the origin to which you want CloudFront to forward your requests
- Whether to forward query strings to your origin
- Whether accessing the specified files requires signed URLs
- Whether to require users to use HTTPS to access those files
- The minimum amount of time that those files stay in the CloudFront cache regardless of the value of any `Cache-Control` headers that your origin adds to the files

When you create a new distribution, you specify settings for the default cache behavior, which automatically forwards all requests to the origin that you specify when you create the distribution. After you create a distribution, you can create additional cache behaviors that define how CloudFront responds when it receives a request for objects that match a path pattern, for example,

`*.jpg`. If you create additional cache behaviors, the default cache behavior is always the last to be processed. Other cache behaviors are processed in the order in which they're listed in the CloudFront console or, if you're using the CloudFront API, the order in which they're listed in the `DistributionConfig` element for the distribution. For more information, see Path pattern.

When you create a cache behavior, you specify the one origin from which you want CloudFront to get objects. As a result, if you want CloudFront to distribute objects from all of your origins, you must have at least as many cache behaviors (including the default cache behavior) as you have origins. For example, if you have two origins and only the default cache behavior, the default cache behavior causes CloudFront to get objects from one of the origins, but the other origin is never used.

For the current maximum number of cache behaviors that you can add to a distribution, or to request a higher quota (formerly known as limit), see General quotas on distributions.

Path pattern

A path pattern (for example, `images/*.jpg`) specifies to which requests you want this cache behavior to apply. When CloudFront receives an end-user request, the requested path is compared with path patterns in the order in which cache behaviors are listed in the distribution. The first match determines which cache behavior is applied to that request. For example, suppose you have three cache behaviors with the following three path patterns, in this order:

- `images/*.jpg`
- `images/*`
- `*.gif`

> **Note**
>
> You can optionally include a slash (/) at the beginning of the path pattern, for example, `/images/*.jpg`. CloudFront behavior is the same with or without the leading /. If you don't specify the / at the beginning of the path, this character is automatically implied; CloudFront treats the path the same with or without the leading /. For example, CloudFront treats `/*product.jpg` the same as `*product.jpg`

A request for the file `images/sample.gif` doesn't satisfy the first path pattern, so the associated cache behaviors are not applied to the request. The file does satisfy the second path pattern, so the

cache behaviors associated with the second path pattern are applied even though the request also matches the third path pattern.

> **Note**
>
> When you create a new distribution, the value of **Path Pattern** for the default cache behavior is set to * (all files) and cannot be changed. This value causes CloudFront to forward all requests for your objects to the origin that you specified in the Origin domain field. If the request for an object does not match the path pattern for any of the other cache behaviors, CloudFront applies the behavior that you specify in the default cache behavior.

> **Important**
>
> Define path patterns and their sequence carefully or you may give users undesired access to your content. For example, suppose a request matches the path pattern for two cache behaviors. The first cache behavior does not require signed URLs and the second cache behavior does require signed URLs. Users are able to access the objects without using a signed URL because CloudFront processes the cache behavior associated with the first match.

If you're working with a MediaPackage channel, you must include specific path patterns for the cache behavior that you define for the endpoint type for your origin. For example, for a DASH endpoint, you type `*.mpd` for **Path Pattern**. For more information and specific instructions, see Serving live video formatted with AWS Elemental MediaPackage.

The path you specify applies to requests for all files in the specified directory and in subdirectories below the specified directory. CloudFront does not consider query strings or cookies when evaluating the path pattern. For example, if an `images` directory contains `product1` and `product2` subdirectories, the path pattern `images/*.jpg` applies to requests for any .jpg file in the `images`, `images/product1`, and `images/product2` directories. If you want to apply a different cache behavior to the files in the `images/product1` directory than the files in the `images` and `images/product2` directories, create a separate cache behavior for `images/product1` and move that cache behavior to a position above (before) the cache behavior for the `images` directory.

You can use the following wildcard characters in your path pattern:

- * matches 0 or more characters.
- ? matches exactly 1 character.

The following examples show how the wildcard characters work:

Path pattern	Files that match the path pattern
`*.jpg`	All .jpg files.
`images/*.jpg`	All .jpg files in the `images` directory and in subdirectories under the `images` directory.
`a*.jpg`	• All .jpg files for which the file name begins with a, for example, `apple.jpg` and `appalachian_trail_2012_05_21.jpg` . • All .jpg files for which the file path begins with a, for example, `abra/cadabra/magic.jpg` .
`a??.jpg`	All .jpg files for which the file name begins with a and is followed by exactly two other characters, for example, `ant.jpg` and `abe.jpg`.
`*.doc*`	All files for which the file name extension begins with `.doc`, for example, `.doc`, `.docx`, and `.docm` files. You can't use the path pattern `*.doc?` in this case, because that path pattern wouldn't apply to requests for `.doc` files; the ? wildcard character replaces exactly one character.

The maximum length of a path pattern is 255 characters. The value can contain any of the following characters:

- A-Z, a-z

Path patterns are case-sensitive, so the path pattern `*.jpg` doesn't apply to the file `LOGO.JPG`

- 0-9
- _ - . * $ / ~ " ' @ : +
- &, passed and returned as `&`

Path normalization

CloudFront normalizes URI paths consistent with RFC 3986 and then matches the path with the correct cache behavior. Once the cache behavior is matched, CloudFront sends the raw URI path to the origin. If they don't match, requests are instead matched to your default cache behavior.

Some characters are normalized and removed from the path, such as multiple slashes (`//`) or periods (`..`). This can alter the URL that CloudFront uses to match the intended cache behavior.

Example Example

You specify the `/a/b*` and `/a*` paths for your cache behavior.

- A viewer sending the `/a/b?c=1` path will match the `/a/b*` cache behavior.
- A viewer sending the `/a/b/..?c=1` path will match the `/a*` cache behavior.

To work around the paths being normalized, you can update your request paths or the path pattern for the cache behavior.

Origin or origin group

Enter the value of an existing origin or origin group. This identifies the origin or origin group to which you want CloudFront to route requests when a request (such as https://example.com/logo.jpg) matches the path pattern for a cache behavior (such as *.jpg) or for the default cache behavior (*).

Viewer protocol policy

Choose the protocol policy that you want viewers to use to access your content in CloudFront edge locations:

- **HTTP and HTTPS**: Viewers can use both protocols.

- **Redirect HTTP to HTTPS**: Viewers can use both protocols, but HTTP requests are automatically redirected to HTTPS requests.
- **HTTPS Only**: Viewers can only access your content if they're using HTTPS.

For more information, see Requiring HTTPS for communication between viewers and CloudFront.

Allowed HTTP methods

Specify the HTTP methods that you want CloudFront to process and forward to your origin:

- **GET, HEAD:** You can use CloudFront only to get objects from your origin or to get object headers.
- **GET, HEAD, OPTIONS:** You can use CloudFront only to get objects from your origin, get object headers, or retrieve a list of the options that your origin server supports.
- **GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE:** You can use CloudFront to get, add, update, and delete objects, and to get object headers. In addition, you can perform other POST operations such as submitting data from a web form.

> **Note**
>
> CloudFront caches responses to `GET` and `HEAD` requests and, optionally, `OPTIONS` requests. Responses to `OPTIONS` requests are cached separately from responses to `GET` and `HEAD` requests (the `OPTIONS` method is included in the cache key for `OPTIONS` requests). CloudFront does not cache responses to requests that use other methods.

> **Important**
>
> If you choose **GET, HEAD, OPTIONS** or **GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE**, you might need to restrict access to your Amazon S3 bucket or to your custom origin to prevent users from performing operations that you don't want them to perform. The following examples explain how to restrict access:
>
> - **If you're using Amazon S3 as an origin for your distribution:** Create a CloudFront origin access control to restrict access to your Amazon S3 content, and give permissions to the origin access control. For example, if you configure CloudFront to accept and forward these methods *only* because you want to use PUT, you must still configure Amazon S3

bucket policies to handle DELETE requests appropriately. For more information, see Restricting access to an Amazon Simple Storage Service origin.

- **If you're using a custom origin:** Configure your origin server to handle all methods. For example, if you configure CloudFront to accept and forward these methods *only* because you want to use POST, you must still configure your origin server to handle DELETE requests appropriately.

Field-level encryption config

If you want to enforce field-level encryption on specific data fields, in the dropdown list, choose a field-level encryption configuration.

For more information, see Using field-level encryption to help protect sensitive data.

Cached HTTP methods

Specify whether you want CloudFront to cache the response from your origin when a viewer submits an OPTIONS request. CloudFront always caches the response to GET and HEAD requests.

Cache based on selected request headers

Specify whether you want CloudFront to cache objects based on the values of specified headers:

- **None (improves caching)** – CloudFront doesn't cache your objects based on header values.
- **Allowlist** – CloudFront caches your objects based only on the values of the specified headers. Use **Allowlist Headers** to choose the headers that you want CloudFront to base caching on.
- **All** – CloudFront doesn't cache the objects that are associated with this cache behavior. Instead, CloudFront sends every request to the origin. (Not recommended for Amazon S3 origins.)

Regardless of the option that you choose, CloudFront forwards certain headers to your origin and takes specific actions based on the headers that you forward. For more information about how CloudFront handles header forwarding, see HTTP request headers and CloudFront behavior (custom and Amazon S3 origins).

For more information about how to configure caching in CloudFront by using request headers, see Caching content based on request headers.

Allowlist headers

Specify the headers that you want CloudFront to consider when caching your objects. Select headers from the list of available headers and choose **Add**. To forward a custom header, enter the name of the header in the field, and choose **Add Custom**.

For the current maximum number of headers that you can allowlist for each cache behavior, or to request a higher quota (formerly known as limit), see Quotas on headers.

Object caching

If your origin server is adding a `Cache-Control` header to your objects to control how long the objects stay in the CloudFront cache and if you don't want to change the `Cache-Control` value, choose **Use Origin Cache Headers**.

To specify a minimum and maximum time that your objects stay in the CloudFront cache regardless of `Cache-Control` headers, and a default time that your objects stay in the CloudFront cache when the `Cache-Control` header is missing from an object, choose **Customize**. Then specify values in the **Minimum TTL**, **Default TTL**, and **Maximum TTL** fields.

For more information, see Managing how long content stays in the cache (expiration).

Minimum TTL

Specify the minimum amount of time, in seconds, that you want objects to stay in the CloudFront cache before CloudFront sends another request to the origin to determine whether the object has been updated.

For more information, see Managing how long content stays in the cache (expiration).

Maximum TTL

Specify the maximum amount of time, in seconds, that you want objects to stay in CloudFront caches before CloudFront queries your origin to see whether the object has been updated. The value that you specify for **Maximum TTL** applies only when your origin adds HTTP headers such as `Cache-Control max-age`, `Cache-Control s-maxage`, or `Expires` to objects. For more information, see Managing how long content stays in the cache (expiration).

To specify a value for **Maximum TTL**, you must choose the **Customize** option for the **Object Caching** setting.

The default value for **Maximum TTL** is 31536000 seconds (one year). If you change the value of **Minimum TTL** or **Default TTL** to more than 31536000 seconds, then the default value of **Maximum TTL** changes to the value of **Default TTL**.

Default TTL

Specify the default amount of time, in seconds, that you want objects to stay in CloudFront caches before CloudFront forwards another request to your origin to determine whether the object has been updated. The value that you specify for **Default TTL** applies only when your origin does *not* add HTTP headers such as `Cache-Control max-age`, `Cache-Control s-maxage`, or `Expires` to objects. For more information, see Managing how long content stays in the cache (expiration).

To specify a value for **Default TTL**, you must choose the **Customize** option for the **Object Caching** setting.

The default value for **Default TTL** is 86400 seconds (one day). If you change the value of **Minimum TTL** to more than 86400 seconds, then the default value of **Default TTL** changes to the value of **Minimum TTL**.

Forward cookies

> **Note**
>
> For Amazon S3 origins, this option applies to only buckets that are configured as a website endpoint.

Specify whether you want CloudFront to forward cookies to your origin server and, if so, which ones. If you choose to forward only selected cookies (an allowlist of cookies), enter the cookie names in the **Allowlist Cookies** field. If you choose **All**, CloudFront forwards all cookies regardless of how many your application uses.

Amazon S3 doesn't process cookies, and forwarding cookies to the origin reduces cache ability. For cache behaviors that are forwarding requests to an Amazon S3 origin, choose **None** for **Forward Cookies**.

For more information about forwarding cookies to the origin, go to Caching content based on cookies.

Allowlist cookies

> **Note**
>
> For Amazon S3 origins, this option applies to only buckets that are configured as a website endpoint.

If you chose **Allowlist** in the **Forward Cookies** list, then in the **Allowlist Cookies** field, enter the names of cookies that you want CloudFront to forward to your origin server for this cache behavior. Enter each cookie name on a new line.

You can specify the following wildcards to specify cookie names:

- ***** matches 0 or more characters in the cookie name
- **?** matches exactly one character in the cookie name

For example, suppose viewer requests for an object include a cookie named:

`userid_`*`member-number`*

Where each of your users has a unique value for *`member-number`*. You want CloudFront to cache a separate version of the object for each member. You could accomplish this by forwarding all cookies to your origin, but viewer requests include some cookies that you don't want CloudFront to cache. Alternatively, you could specify the following value as a cookie name, which causes CloudFront to forward to the origin all of the cookies that begin with `userid_`:

```
userid_*
```

For the current maximum number of cookie names that you can allowlist for each cache behavior, or to request a higher quota (formerly known as limit), see Quotas on cookies (legacy cache settings).

Query string forwarding and caching

CloudFront can cache different versions of your content based on the values of query string parameters. Choose one of the following options:

None (Improves Caching)

Choose this option if your origin returns the same version of an object regardless of the values of query string parameters. This increases the likelihood that CloudFront can serve a request from the cache, which improves performance and reduces the load on your origin.

Forward all, cache based on allowlist

Choose this option if your origin server returns different versions of your objects based on one or more query string parameters. Then specify the parameters that you want CloudFront to use as a basis for caching in the Query string allowlist field.

Forward all, cache based on all

Choose this option if your origin server returns different versions of your objects for all query string parameters.

For more information about caching based on query string parameters, including how to improve performance, see Caching content based on query string parameters.

Query string allowlist

If you chose **Forward all, cache based on allowlist** for Query string forwarding and caching, specify the query string parameters that you want CloudFront to use as a basis for caching.

Smooth Streaming

Choose **Yes** if you want to distribute media files in the Microsoft Smooth Streaming format and you do not have an IIS server.

Choose **No** if you have a Microsoft IIS server that you want to use as an origin to distribute media files in the Microsoft Smooth Streaming format, or if you are not distributing Smooth Streaming media files.

> **Note**
>
> If you specify **Yes**, you can still distribute other content using this cache behavior if that content matches the value of **Path Pattern**.

For more information, see Configuring video on demand for Microsoft Smooth Streaming.

Restrict viewer access (use signed URLs or signed cookies)

If you want requests for objects that match the `PathPattern` for this cache behavior to use public URLs, choose **No**.

If you want requests for objects that match the `PathPattern` for this cache behavior to use signed URLs, choose **Yes**. Then specify the Amazon accounts that you want to use to create signed URLs; these accounts are known as trusted signers.

For more information about trusted signers, see Specifying the signers that can create signed URLs and signed cookies.

Trusted signers

Choose which Amazon accounts you want to use as trusted signers for this cache behavior:

- **Self:** Use the account with which you're currently signed into the Amazon Web Services Management Console as a trusted signer. If you're currently signed in as an IAM user, the associated Amazon account is added as a trusted signer.
- **Specify Accounts:** Enter account numbers for trusted signers in the **Amazon Account Numbers** field.

To create signed URLs, an Amazon account must have at least one active CloudFront key pair.

> **Important**
>
> If you're updating a distribution that you're already using to distribute content, add trusted signers only when you're ready to start generating signed URLs for your objects. After you add trusted signers to a distribution, users must use signed URLs to access the objects that match the `PathPattern` for this cache behavior.

Amazon Web Services account numbers

If you want to create signed URLs using Amazon Web Services accounts in addition to or instead of the current account, enter one Amazon Web Services account number per line in this field. Note the following:

- The accounts that you specify must have at least one active CloudFront key pair. For more information, see Creating key pairs for your signers.

- You can't create CloudFront key pairs for IAM users, so you can't use IAM users as trusted signers.
- For information about how to get the Amazon Web Services account number for an account, see Your Amazon Web Services account identifiers in the *Amazon Web Services General Reference*.
- If you enter the account number for the current account, CloudFront automatically checks the **Self** check box and removes the account number from the **Amazon Account Numbers** list.

Compress objects automatically

If you want CloudFront to automatically compress files of certain types when viewers support compressed content, choose **Yes**. When CloudFront compresses your content, downloads are faster because the files are smaller, and your web pages render faster for your users. For more information, see Serving compressed files.

CloudFront event

You can choose to run a Lambda function when one or more of the following CloudFront events occur:

- When CloudFront receives a request from a viewer (viewer request)
- Before CloudFront forwards a request to the origin (origin request)
- When CloudFront receives a response from the origin (origin response)
- Before CloudFront returns the response to the viewer (viewer response)

For more information, see How to decide which CloudFront event to use to trigger a Lambda@Edge function.

Lambda function ARN

Specify the Amazon Resource Name (ARN) of the Lambda function that you want to add a trigger for. To learn how to get the ARN for a function, see step 1 of the procedure Adding Triggers by Using the CloudFront Console.

Distribution settings

The following values apply to the entire distribution.

Price class

Choose the price class that corresponds with the maximum price that you want to pay for CloudFront service. By default, CloudFront serves your objects from edge locations in all CloudFront Regions.

For more information about price classes and about how your choice of price class affects CloudFront performance for your distribution, see CloudFront pricing.

Amazon WAF web ACL

You can protect your CloudFront distribution with Amazon WAF, a web application firewall that allows you to secure your web applications and APIs to block requests before they reach your servers. You can Enabling Amazon WAF for new distributions when creating or editing a CloudFront distribution.

Optionally, you can later configure additional security protections for other threats specific to your application in the Amazon WAF console at https://console.amazonaws.cn/wafv2/.

For more information about Amazon WAF, see the Amazon WAF Developer Guide.

Alternate domain names (CNAMEs)

Optional. Specify one or more domain names that you want to use for URLs for your objects instead of the domain name that CloudFront assigns when you create your distribution. You must own the domain name, or have authorization to use it, which you verify by adding an SSL/TLS certificate.

For example, if you want the URL for the object:

`/images/image.jpg`

To look like this:

`https://www.example.com/images/image.jpg`

Instead of like this:

`https://d111111abcdef8.cloudfront.net/images/image.jpg`

Add a CNAME for `www.example.com`.

Important

If you add a CNAME for `www.example.com` to your distribution, you also must do the following:

- Create (or update) a CNAME record with your DNS service to route queries for `www.example.com` to `d111111abcdef8.cloudfront.net`.
- Add a certificate to CloudFront from a trusted certificate authority (CA) that covers the domain name (CNAME) that you add to your distribution, to validate your authorization to use the domain name.

You must have permission to create a CNAME record with the DNS service provider for the domain. Typically, this means that you own the domain, or that you're developing an application for the domain owner.

For the current maximum number of alternate domain names that you can add to a distribution, or to request a higher quota (formerly known as limit), see General quotas on distributions.

For more information about alternate domain names, see Using custom URLs by adding alternate domain names (CNAMEs). For more information about CloudFront URLs, see Customizing the URL format for files in CloudFront.

SSL certificate

If you specified an alternate domain name to use with your distribution, choose **Custom SSL Certificate**, and then, to validate your authorization to use the alternate domain name, choose a certificate that covers it. If you want viewers to use HTTPS to access your objects, choose the settings that support that.

Note

Before you can specify a custom SSL certificate, you must specify a valid alternate domain name. For more information, see Requirements for using alternate domain names and Using alternate domain names and HTTPS.

- **Default CloudFront Certificate (*.cloudfront.net)** – Choose this option if you want to use the CloudFront domain name in the URLs for your objects, such as `https://d111111abcdef8.cloudfront.net/image1.jpg`.
- **Custom SSL Certificate** – Choose this option if you want to use your own domain name in the URLs for your objects as an alternate domain name, such as `https://example.com/image1.jpg`. Then choose a certificate to use that covers the alternate domain name. The list of certificates can include any of the following:
  - Certificates provided by Amazon Certificate Manager
  - Certificates that you purchased from a third-party certificate authority and uploaded to ACM
  - Certificates that you purchased from a third-party certificate authority and uploaded to the IAM certificate store

  If you choose this setting, we recommend that you use only an alternate domain name in your object URLs (https://example.com/logo.jpg). If you use your CloudFront distribution domain name (https://d111111abcdef8.cloudfront.net/logo.jpg) and a client uses an older viewer that doesn't support SNI, how the viewer responds depends on the value that you choose for **Clients Supported**:
  - **All Clients**: The viewer displays a warning because the CloudFront domain name doesn't match the domain name in your SSL/TLS certificate.
  - **Only Clients that Support Server Name Indication (SNI)**: CloudFront drops the connection with the viewer without returning the object.

Custom SSL client support

If you specified one or more alternate domain names and a custom SSL certificate for the distribution, choose how you want CloudFront to serve HTTPS requests:

- **Clients that Support Server Name Indication (SNI) - (Recommended)** – With this setting, virtually all modern web browsers and clients can connect to the distribution, because they support SNI. However, some viewers might use older web browsers or clients that don’t support SNI, which means they can’t connect to the distribution.

  To apply this setting using the CloudFront API, specify `sni-only` in the `SSLSupportMethod` field. In Amazon CloudFormation, the field is named `SslSupportMethod` (note the different capitalization).

- **Legacy Clients Support** – With this setting, older web browsers and clients that don't support SNI can connect to the distribution. However, this setting incurs additional monthly charges. For the exact price, go to the Amazon CloudFront Pricing page, and search the page for **Dedicated IP custom SSL**.

  To apply this setting using the CloudFront API, specify `vip` in the `SSLSupportMethod` field. In Amazon CloudFormation, the field is named `SslSupportMethod` (note the different capitalization).

For more information, see Choosing how CloudFront serves HTTPS requests.

Security policy

Specify the security policy that you want CloudFront to use for HTTPS connections with viewers (clients). A security policy determines two settings:

- The minimum SSL/TLS protocol that CloudFront uses to communicate with viewers.
- The ciphers that CloudFront can use to encrypt the content that it returns to viewers.

For more information about the security policies, including the protocols and ciphers that each one includes, see Supported protocols and ciphers between viewers and CloudFront.

The security policies that are available depend on the values that you specify for **SSL Certificate** and **Custom SSL Client Support** (known as `CloudFrontDefaultCertificate` and `SSLSupportMethod` in the CloudFront API):

- When **SSL Certificate** is **Default CloudFront Certificate (*.cloudfront.net)** (when `CloudFrontDefaultCertificate` is `true` in the API), CloudFront automatically sets the security policy to TLSv1.
- When **SSL Certificate** is **Custom SSL Certificate (example.com)** *and* **Custom SSL Client Support** is **Clients that Support Server Name Indication (SNI) - (Recommended)** (when `CloudFrontDefaultCertificate` is `false` *and* `SSLSupportMethod` is `sni-only` in the API), you can choose from the following security policies:
  - TLSv1.2_2021
  - TLSv1.2_2019
  - TLSv1.2_2018
  - TLSv1.1_2016

- TLSv1_2016
- TLSv1

- When **SSL Certificate** is **Custom SSL Certificate (example.com)** *and* **Custom SSL Client Support** is **Legacy Clients Support** (when `CloudFrontDefaultCertificate` is `false` *and* `SSLSupportMethod` is `vip` in the API), you can choose from the following security policies:
  - TLSv1
  - SSLv3

  In this configuration, the TLSv1.2_2021, TLSv1.2_2019, TLSv1.2_2018, TLSv1.1_2016, and TLSv1_2016 security policies aren't available in the CloudFront console or API. If you want to use one of these security policies, you have the following options:

  - Evaluate whether your distribution needs Legacy Clients Support with dedicated IP addresses. If your viewers support server name indication (SNI), we recommend that you update your distribution's **Custom SSL Client Support** setting to **Clients that Support Server Name Indication (SNI)** (set `SSLSupportMethod` to `sni-only` in the API). This enables you to use any of the available TLS security policies, and it can also reduce your CloudFront charges.
  - If you must keep Legacy Clients Support with dedicated IP addresses, you can request one of the other TLS security policies (TLSv1.2_2021, TLSv1.2_2019, TLSv1.2_2018, TLSv1.1_2016, or TLSv1_2016) by creating a case in the Amazon Support Center.

    Note

    Before you contact Amazon Support to request this change, consider the following:

    - When you add one of these security policies (TLSv1.2_2021, TLSv1.2_2019, TLSv1.2_2018, TLSv1.1_2016, or TLSv1_2016) to a Legacy Clients Support distribution, the security policy is applied to *all* non-SNI viewer requests for *all* Legacy Clients Support distributions in your Amazon account. However, when viewers send SNI requests to a distribution with Legacy Clients Support, the security policy of that distribution applies. To make sure that your desired security policy is applied to *all* viewer requests sent to *all* Legacy Clients Support distributions in your Amazon account, add the desired security policy to each distribution individually.
    - By definition, the new security policy doesn't support the same ciphers and protocols as the old one. For example, if you chose to upgrade a distribution's security policy from TLSv1 to TLSv1.1_2016, that distribution will no longer support the DES-CBC3-SHA cipher. For more information about the ciphers and protocols that each

security policy supports, see Supported protocols and ciphers between viewers and CloudFront.

Supported HTTP versions

Choose the HTTP versions that you want your distribution to support when viewers communicate with CloudFront.

For viewers and CloudFront to use HTTP/2, viewers must support TLSv1.2 or later, and Server Name Indication (SNI). CloudFront does not offer native support for gRPC over HTTP/2.

For viewers and CloudFront to use HTTP/3, viewers must support TLSv1.3 and Server Name Indication (SNI). CloudFront supports HTTP/3 connection migration to allow the viewer to switch networks without losing connection. For more information about connection migration, see Connection Migration at RFC 9000.

Note

For more information about supported TLSv1.3 ciphers, see Supported protocols and ciphers between viewers and CloudFront.

Default root object

Optional. The object that you want CloudFront to request from your origin (for example, `index.html`) when a viewer requests the root URL of your distribution (`https://www.example.com/`) instead of an object in your distribution (`https://www.example.com/product-description.html`). Specifying a default root object avoids exposing the contents of your distribution.

The maximum length of the name is 255 characters. The name can contain any of the following characters:

- A-Z, a-z
- 0-9
- _ - . * $ / ~ " '
- &, passed and returned as `&`

When you specify the default root object, enter only the object name, for example, `index.html`. Do not add a `/` before the object name.

For more information, see Specifying a default root object.

Logging

Whether you want CloudFront to log information about each request for an object and store the log files in an Amazon S3 bucket. You can enable or disable logging at any time. There is no extra charge if you enable logging, but you accrue the usual Amazon S3 charges for storing and accessing the files in an Amazon S3 bucket. You can delete the logs at any time. For more information about CloudFront access logs, see Configuring and using standard logs (access logs).

Bucket for logs

If you chose **On** for **Logging**, the Amazon S3 bucket that you want CloudFront to store access logs in, for example, *myLogs-DOC-EXAMPLE-BUCKET*`.s3.amazonaws.com`.

Important

Don't choose an Amazon S3 bucket in any of the following Regions, because CloudFront doesn't deliver standard logs to buckets in these Regions:

- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asia Pacific (Hyderabad)
- Asia Pacific (Jakarta)
- Asia Pacific (Melbourne)
- Canada West (Calgary)
- Europe (Milan)
- Europe (Spain)
- Europe (Zurich)
- Israel (Tel Aviv)
- Middle East (Bahrain)
- Middle East (UAE)

If you enable logging, CloudFront records information about each end-user request for an object and stores the files in the specified Amazon S3 bucket. You can enable or disable logging at any time. For more information about CloudFront access logs, see Configuring and using standard logs (access logs).

> **Note**
>
> You must have the permissions required to get and update Amazon S3 bucket ACLs, and the S3 ACL for the bucket must grant you `FULL_CONTROL`. This allows CloudFront to give the `awslogsdelivery` account permission to save log files in the bucket. For more information, see Permissions required to configure standard logging and to access your log files.

Log prefix

Optional. If you chose **On** for **Logging**, specify the string, if any, that you want CloudFront to prefix to the access log file names for this distribution, for example, `exampleprefix/`. The trailing slash (/) is optional but recommended to simplify browsing your log files. For more information about CloudFront access logs, see Configuring and using standard logs (access logs).

Cookie logging

If you want CloudFront to include cookies in access logs, choose **On**. If you choose to include cookies in logs, CloudFront logs all cookies regardless of how you configure the cache behaviors for this distribution: forward all cookies, forward no cookies, or forward a specified list of cookies to the origin.

Amazon S3 doesn't process cookies, so unless your distribution also includes an Amazon EC2 or other custom origin, we recommend that you choose **Off** for the value of **Cookie Logging**.

For more information about cookies, go to Caching content based on cookies.

Enable IPv6

IPv6 is a new version of the IP protocol. It's the eventual replacement for IPv4 and uses a larger address space. CloudFront always responds to IPv4 requests. If you want CloudFront to respond to requests from IPv4 IP addresses (such as 192.0.2.44) and requests from IPv6 addresses (such as 2001:0db8:85a3::8a2e:0370:7334), select **Enable IPv6**.

In general, you should enable IPv6 if you have users on IPv6 networks who want to access your content. However, if you're using signed URLs or signed cookies to restrict access to your content, and if you're using a custom policy that includes the `IpAddress` parameter to restrict the IP addresses that can access your content, do not enable IPv6. If you want to restrict access to some content by IP address and not restrict access to other content (or restrict access but not by IP address), you can create two distributions. For information about creating signed URLs by using a custom policy, see Creating a signed URL using a custom policy. For information about creating signed cookies by using a custom policy, see Setting signed cookies using a custom policy.

If you're using a Route 53 alias resource record set to route traffic to your CloudFront distribution, you need to create a second alias resource record set when both of the following are true:

- You enable IPv6 for the distribution
- You're using alternate domain names in the URLs for your objects

For more information, see Routing traffic to an Amazon CloudFront distribution by using your domain name in the *Amazon Route 53 Developer Guide*.

If you created a CNAME resource record set, either with Route 53 or with another DNS service, you don't need to make any changes. A CNAME record routes traffic to your distribution regardless of the IP address format of the viewer request.

If you enable IPv6 and CloudFront access logs, the `c-ip` column includes values in IPv4 and IPv6 format. For more information, see Configuring and using standard logs (access logs).

Note

To maintain high customer availability, CloudFront responds to viewer requests by using IPv4 if our data suggests that IPv4 will provide a better user experience. To find out what percentage of requests CloudFront is serving over IPv6, enable CloudFront logging for your distribution and parse the `c-ip` column, which contains the IP address of the viewer that made the request. This percentage should grow over time, but it will remain a minority of traffic as IPv6 is not yet supported by all viewer networks globally. Some viewer networks have excellent IPv6 support, but others don't support IPv6 at all. (A viewer network is analogous to your home internet or wireless carrier.)
For more information about our support for IPv6, see the CloudFront FAQ. For information about enabling access logs, see the fields Logging, Bucket for logs, and Log prefix.

Comment

Optional. When you create a distribution, you can include a comment of up to 128 characters. You can update the comment at any time.

Distribution state

Indicates whether you want the distribution to be enabled or disabled once it's deployed:

- *Enabled* means that as soon as the distribution is fully deployed you can deploy links that use the distribution's domain name and users can retrieve content. Whenever a distribution is enabled, CloudFront accepts and handles any end-user requests for content that use the domain name associated with that distribution.

  When you create, modify, or delete a CloudFront distribution, it takes time for your changes to propagate to the CloudFront database. An immediate request for information about a distribution might not show the change. Propagation usually completes within minutes, but a high system load or network partition might increase this time.
- *Disabled* means that even though the distribution might be deployed and ready to use, users can't use it. Whenever a distribution is disabled, CloudFront doesn't accept any end-user requests that use the domain name associated with that distribution. Until you switch the distribution from disabled to enabled (by updating the distribution's configuration), no one can use it.

You can toggle a distribution between disabled and enabled as often as you want. Follow the process for updating a distribution's configuration. For more information, see Updating a distribution.

Custom error pages and error caching

You can have CloudFront return an object to the viewer (for example, an HTML file) when your Amazon S3 or custom origin returns an HTTP 4xx or 5xx status code to CloudFront. You can also specify how long an error response from your origin or a custom error page is cached in CloudFront edge caches. For more information, see Creating a custom error page for specific HTTP status codes.

> **Note**
>
> The following values aren't included in the Create Distribution wizard, so you can configure custom error pages only when you update a distribution.

HTTP error code

The HTTP status code for which you want CloudFront to return a custom error page. You can configure CloudFront to return custom error pages for none, some, or all of the HTTP status codes that CloudFront caches.

Error caching minimum TTL (seconds)

The minimum amount of time that you want CloudFront to cache error responses from your origin server.

Response page path

The path to the custom error page (for example, `/4xx-errors/403-forbidden.html`) that you want CloudFront to return to a viewer when your origin returns the HTTP status code that you specified for **Error Code** (for example, 403). If you want to store your objects and your custom error pages in different locations, your distribution must include a cache behavior for which the following is true:

- The value of **Path Pattern** matches the path to your custom error messages. For example, suppose you saved custom error pages for 4xx errors in an Amazon S3 bucket in a directory named `/4xx-errors`. Your distribution must include a cache behavior for which the path pattern routes requests for your custom error pages to that location, for example, **/4xx-errors/***.
- The value of **Origin** specifies the value of **Origin ID** for the origin that contains your custom error pages.

HTTP response code

The HTTP status code that you want CloudFront to return to the viewer along with the custom error page.

Geographic restrictions

If you need to prevent users in selected countries from accessing your content, you can configure your CloudFront distribution with an **Allowlist** or a **Block list**. There is no additional charge for configuring geographic restrictions. For more information, see Restricting the geographic distribution of your content.

Values that CloudFront displays in the console

When you create a new distribution or update an existing distribution, CloudFront displays the following information in the CloudFront console.

> **Note**
>
> Active trusted signers, the Amazon accounts that have an active CloudFront key pair and can be used to create valid signed URLs, are currently not visible in the CloudFront console.

Distribution ID

When you perform an action on a distribution using the CloudFront API, you use the distribution ID to specify which distribution to use, for example, `EDFDVBD6EXAMPLE`. You can't change a distribution's distribution ID.

Deploying and status

When you deploy a distribution, you see the **Deploying** status under the **Last modified** column. Wait for the distribution to finish deploying and make sure the **Status** column shows **Enabled**. For more information, see Distribution state.

Last modified

The date and time that the distribution was last modified, using ISO 8601 format, for example, 2012-05-19T19:37:58Z. For more information, see https://www.w3.org/TR/NOTE-datetime.

Domain name

You use the distribution's domain name in the links to your objects. For example, if your distribution's domain name is `d111111abcdef8.cloudfront.net`, the link to `/images/image.jpg` would be `https://d111111abcdef8.cloudfront.net/images/image.jpg`. You can't change the CloudFront domain name for your distribution. For more information about CloudFront URLs for links to your objects, see Customizing the URL format for files in CloudFront.

If you specified one or more alternate domain names (CNAMEs), you can use your own domain names for links to your objects instead of using the CloudFront domain name. For more information about CNAMEs, see Alternate domain names (CNAMEs).

> **Note**
>
> CloudFront domain names are unique. Your distribution's domain name was never used for a previous distribution and will never be reused for another distribution in the future.

CNAME status

The ICP recordal status for CNAMEs associated with the distributions. Amazon services in China customers must file for an Internet Content Provider (ICP) recordal if they want to serve content publicly on an alternate domain name, or CNAME, that they've added to CloudFront.

The possible status values for CNAME status are listed in the following table.

Value	Description
APPROVED	All alternate domains names, or CNAMEs, that are associated with the distribution have valid ICP recordal numbers. Multiple CNAMEs can be associated with a distribution, and CNAMEs can correspond to different ICP recordals.
SUSPENDED	*All* alternate domain names that are associated with the distribution do not have valid ICP recordal numbers.
PENDING	*At least one* alternate domain name that is associated with the distribution does not have a valid ICP recordal number.

For more information about ICP recordals, see Signup, Accounts, and Credentials in *Getting Started with Amazon services in China*.

Testing a distribution

After you've created your distribution, CloudFront knows where your origin server is, and you know the domain name associated with the distribution. You can create links to your objects using the CloudFront domain name, and CloudFront will serve the objects to your webpage or application.

> **Note**
>
> You must wait until the status of the distribution changes to **Deployed** before you can test your links.

To create links to objects in a web distribution

1. Copy the following HTML code into a new file, replace *domain-name* with your distribution's domain name, and replace *object-name* with the name of your object.

   ```
   <html>
   <head>My CloudFront Test</head>
   <body>
   <p>My text content goes here.</p>
   <p><img src="https://domain-name/object-name" alt="my test image"
   </body>
   </html>
   ```

   For example, if your domain name were `d111111abcdef8.cloudfront.net` and your object were `image.jpg`, the URL for the link would be:

   `https://d111111abcdef8.cloudfront.net/image.jpg`.

   If your object is in a folder on your origin server, then the folder must also be included in the URL. For example, if image.jpg were located in the images folder on your origin server, then the URL would be:

   `https://d111111abcdef8.cloudfront.net/images/image.jpg`
2. Save the HTML code in a file that has an .html file name extension.
3. Open your webpage in a browser to ensure that you can see your object.

The browser returns your page with the embedded image file, served from the edge location that CloudFront determined was appropriate to serve the object.

Updating a distribution

In the CloudFront console, you can see the CloudFront distributions that are associated with your Amazon account, view the settings for a distribution, and update most settings. Be aware that

settings changes that you make won't take effect until the distribution has propagated to the Amazon edge locations.

To update a CloudFront distribution

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Select the ID of a distribution. The list includes all of the distributions associated with the Amazon account that you used to sign in to the CloudFront console.
3. To edit settings for a distribution, choose the **Distribution Settings** tab.
4. To update general settings, choose **Edit**. Otherwise, choose the tab for the settings that you want to update: **Origins** or **Behaviors**.
5. Make the updates, and then, to save your changes, choose **Yes, Edit**. For information about the fields, see the following topics:
   - **General settings:** Distribution settings
   - **Origin settings:** Origin settings
   - **Cache behavior settings:** Cache behavior settings
6. If you want to delete an origin in your distribution, do the following:
   a. Choose **Behaviors**, and then make sure you have moved any default cache behaviors associated with the origin to another origin.
   b. Choose **Origins**, and then select an origin.
   c. Choose **Delete**.

You can also update a distribution by using the CloudFront API:

- To update a distribution, see UpdateDistribution in the *Amazon CloudFront API Reference*.

Important

When you update your distribution, be aware that a number of additional fields are required that are not required to create a distribution. For a summary of the fields required for when you create or update a distribution, see Required API fields for creating and updating distributions. To help make sure that all of the required fields are included

when you use the CloudFront API to update a distribution, follow the steps described in UpdateDistribution in the *Amazon CloudFront API Reference*.

When you save changes to your distribution configuration, CloudFront starts to propagate the changes to all edge locations. Successive configuration changes propagate in their respective order. Until your configuration is updated in an edge location, CloudFront continues to serve your content from that location based on the previous configuration. After your configuration is updated in an edge location, CloudFront immediately starts to serve your content from that location based on the new configuration.

Your changes don't propagate to every edge location instantaneously. When propagation is complete, the status of your distribution changes from **InProgress** to **Deployed**. While CloudFront is propagating your changes, we can't determine whether a given edge location is serving your content based on the previous configuration or the new configuration.

Tagging Amazon CloudFront distributions

Tags are words or phrases that you can use to identify and organize your Amazon resources. You can add multiple tags to each resource, and each tag includes a key and a value that you define. For example, the key might be "domain" and the value might be "example.com". You can search and filter your resources based on the tags you add.

The following are two examples of how it can be useful to work with tags in CloudFront:

- Use tags to track billing information in different categories. When you apply tags to CloudFront distributions or other Amazon resources (such as Amazon EC2 instances or Amazon S3 buckets) and activate the tags, Amazon generates a cost allocation report as a comma-separated value (CSV file) with your usage and costs aggregated by your active tags. You can apply tags that represent business categories (such as cost centers, application names, or owners) to organize your costs across multiple services. For more information about using tags for cost allocation, see Using Cost Allocation Tags in the *Amazon Billing User Guide*.
- Use tags to enforce tag-based permissions on CloudFront distributions. For more information, see ABAC with CloudFront.

Note the following:

- You can tag distributions, but you can't tag origin access identities or invalidations.

- Tag Editor and Resource groups are currently not supported for CloudFront.

For the current maximum number of tags that you can add to a distribution, see Quotas. To request a higher quota (formerly known as limit), create a case with the Amazon Support Center.

You can also apply tags to resources by using the CloudFront API, Amazon CLI, SDKs, and Amazon Tools for Windows PowerShell. For more information, see the following documentation:

- CloudFront API – See the following operations in the *Amazon CloudFront API Reference*:
  - ListTagsForResource
  - TagResource
  - UntagResource
- Amazon CLI – See cloudfront in the *Amazon CLI Command Reference*
- SDKs – See the applicable SDK documentation on the Amazon Documentation page
- Tools for Windows PowerShell – See Amazon CloudFront in the Amazon Tools for PowerShell Cmdlet Reference

Topics

- Tag restrictions
- Adding, editing, and deleting tags for distributions

Tag restrictions

The following basic restrictions apply to tags:

- Maximum number of tags per resource – 50
- Maximum key length – 128 Unicode characters
- Maximum value length – 256 Unicode characters
- Valid values for key and value – a-z, A-Z, 0-9, space, and the following characters: _ . : / = + - and @
- Tag keys and values are case sensitive
- Don't use `aws:` as a prefix for keys; it's reserved for Amazon use

Adding, editing, and deleting tags for distributions

The following procedure explains how to add, edit, and delete tags for your distributions in the CloudFront console.

To add tags, edit, or delete tags for a distribution

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose the ID for the distribution that you want to update.
3. Choose the **Tags** tab.
4. Choose **Manage tags**.
5. On the **Manage tags** page, you can do the following:
   - To add a tag, type a key and, optionally, a value for the tag. Choose the **Add new tag** button to add more tags.
   - To edit a tag, change the tag's key or its value, or both. You can delete the value for a tag, but the key is required.
   - To delete a tag, choose the **Remove** button next to the tag.
6. Choose **Save changes**.

Deleting a distribution

If you no longer want to use a distribution, you can delete it by using the CloudFront console or by using the CloudFront API.

Be aware that before you can delete a distribution, you must disable it, which requires permission to update the distribution.

If you need to delete a distribution with an OAC attached to an S3 bucket, see Deleting a distribution with an OAC attached to an S3 bucket for important details.

> **Note**
>
> If you disable a distribution that has an alternate domain name associated with it, CloudFront stops accepting traffic for that domain name (such as www.example.com), even

if another distribution has an alternate domain name with a wildcard (*) that matches the same domain (such as *.example.com).

To delete a CloudFront distribution

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the right pane of the CloudFront console, find the distribution that you want to delete.
   - If the **Status** column shows **Disabled**, skip to Step 6.
   - If the **Status** shows **Enabled** but the distribution still shows **Deploying** in the **Last modified** column, wait for deployment to finish before continuing to step 3.
3. In the right pane of the CloudFront console, select the check box for the distribution that you want to delete.
4. Choose **Disable** to disable the distribution, and choose **Yes, Disable** to confirm. Then choose **Close**.

   Note

   Because CloudFront must propagate this change to all edge locations, it might take a few minutes before the update is complete and you can delete your distribution.
5. The value of the **Status** column immediately changes to **Disabled**. Wait until the new timestamp appears under the **Last modified** column.
6. Select the check box for the distribution that you want to delete.
7. Choose **Delete**, **Delete**.

   Note

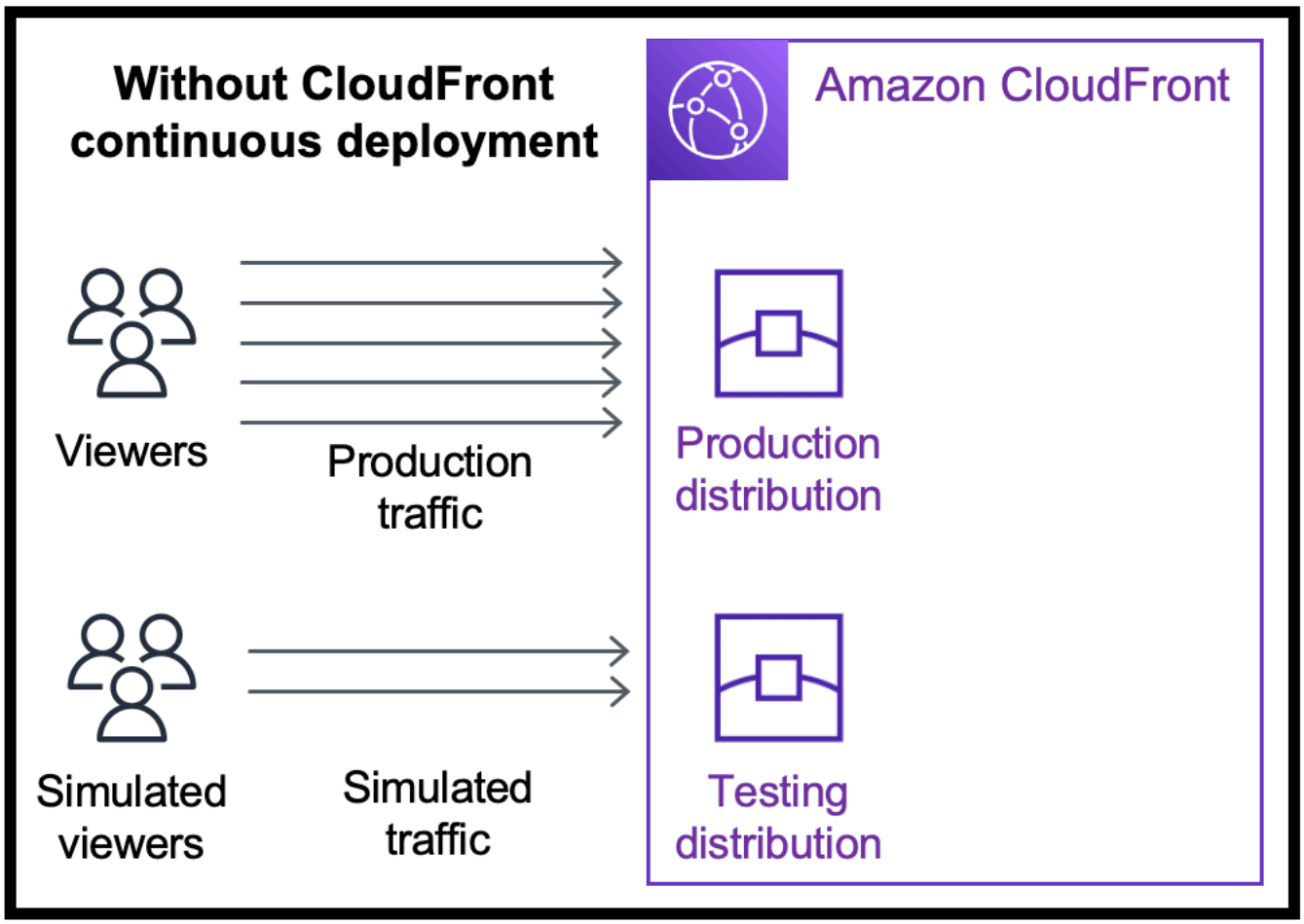
   If you have just marked your distribution as disabled, CloudFront might still need a few more minutes to propagate that change to the edge locations. Until propagation is complete, the **Delete** option isn't available.

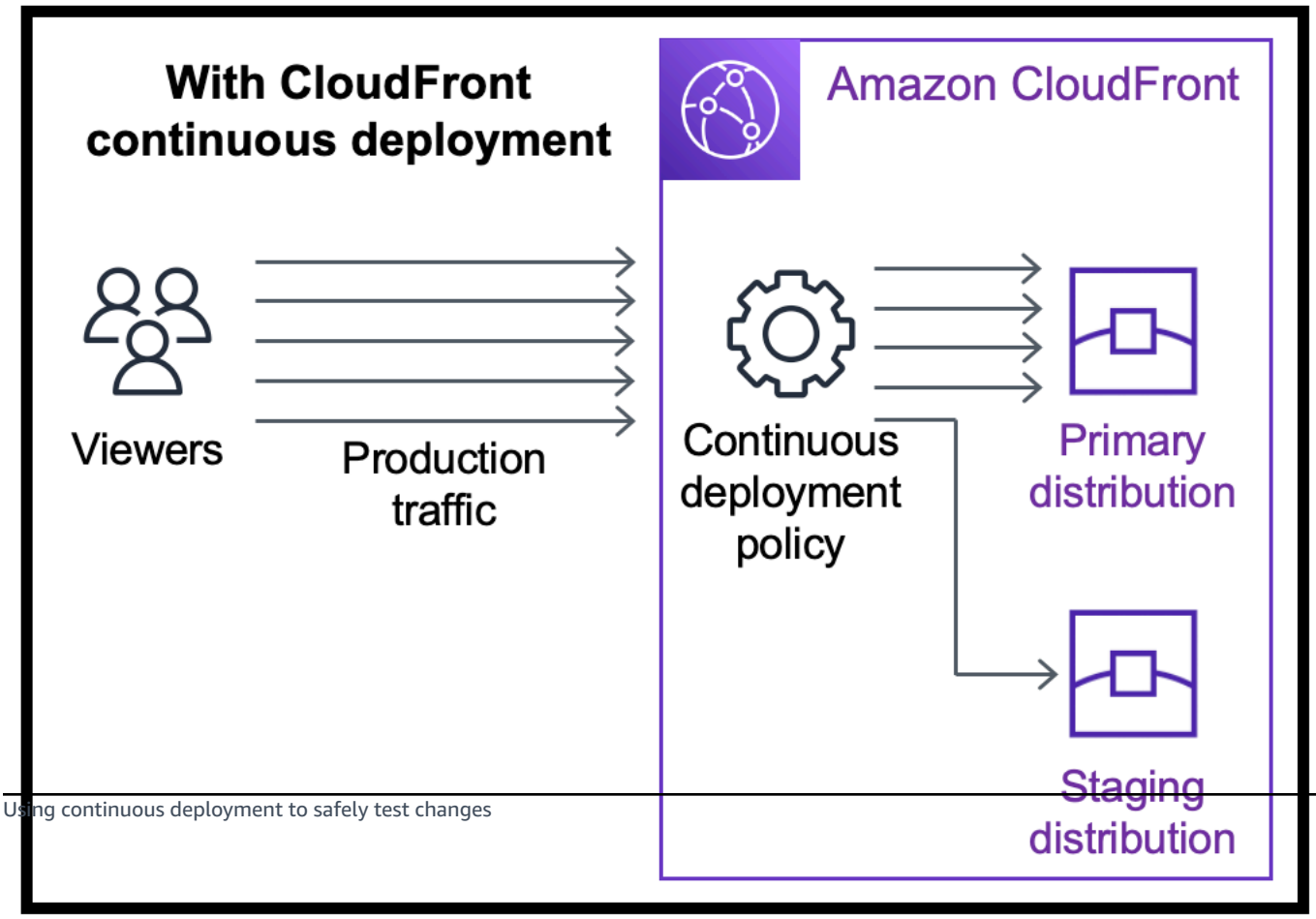
You can also delete a distribution using the CloudFront API. For more information, see DeleteDistribution in the *Amazon CloudFront API Reference*.

Using CloudFront continuous deployment to safely test CDN configuration changes

With Amazon CloudFront *continuous deployment* you can safely deploy changes to your CDN configuration by testing first with a subset of production traffic. You can use a *staging distribution* and a *continuous deployment policy* to send some traffic from real (production) viewers to the new CDN configuration and validate that it works as expected. You can monitor the performance of the new configuration in real time, and promote the new configuration to serve all traffic via the *primary distribution* when you're ready.

The following diagram shows the benefit of using CloudFront continuous deployment. Without it, you would have to test CDN configuration changes with simulated traffic. With continuous deployment you can test the changes with a subset of production traffic, then promote the changes to the primary distribution when you're ready.


Without CloudFront continuous deployment
Amazon CloudFront
Viewers
Production traffic
Production distribution
Simulated viewers
Simulated traffic
Testing distribution


With CloudFront continuous deployment
Amazon CloudFront
Viewers
Production traffic
Continuous deployment policy
Primary distribution
Staging distribution

Topics

- Workflow for using CloudFront continuous deployment
- Working with a staging distribution and continuous deployment policy
- Monitoring a staging distribution
- Understanding how continuous deployment works
- Quotas and other considerations for continuous deployment

Workflow for using CloudFront continuous deployment

The following high-level workflow explains how to safely test and deploy configuration changes with CloudFront continuous deployment.

1. Choose the distribution that you want to use as the *primary distribution*. The primary distribution is one that currently serves production traffic.
2. From the primary distribution, create a *staging distribution*. A staging distribution starts as a copy of the primary distribution.
3. Create a *traffic configuration* inside a *continuous deployment policy*, and attach it to the primary distribution. This determines how CloudFront routes traffic to the staging distribution. For more information about routing requests to a staging distribution, see the section called “Routing requests to the staging distribution”.
4. Update the configuration of the staging distribution. For more information about the settings that you can update, see the section called “Updating primary and staging distributions”.
5. Monitor the staging distribution to determine whether the configuration changes perform as expected. For more information about monitoring a staging distribution, see the section called “Monitoring a staging distribution”.

   As you monitor the staging distribution, you can:

   - Update the configuration of the staging distribution again, to continue testing configuration changes.
   - Update the continuous deployment policy (traffic configuration) to send more or less traffic to the staging distribution.
6. When you're satisfied with the performance of the staging distribution, *promote* the staging distribution's configuration to the primary distribution, which copies the staging distribution's configuration to the primary distribution. This also disables the continuous deployment policy which means that CloudFront routes all traffic to the primary distribution.

You can build automation that monitors the performance of the staging distribution (step 5) and promotes the configuration automatically (step 6) when certain criteria are met.

After you promote a configuration, you can reuse the same staging distribution the next time you want to test a configuration change.

For more information about working with staging distributions and continuous deployment policies in the CloudFront console, the Amazon CLI, or the CloudFront API, see the following section.

Working with a staging distribution and continuous deployment policy

You can create, update, and modify staging distributions and continuous deployment policies in the CloudFront console, with the Amazon Command Line Interface (Amazon CLI), or with the CloudFront API.

Console

To work with a staging distribution and a continuous deployment policy with the Amazon Web Services Management Console, use the following procedures.

To create a staging distribution and continuous deployment policy (console)

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Distributions**.
3. Choose the distribution that you want to use as the *primary distribution*. The primary distribution is one that currently serves production traffic, the one from which you will create the staging distribution.
4. In the **Continuous deployment** section, choose **Create staging distribution**. This opens the **Create staging distribution** wizard.
5. In the **Create staging distribution** wizard, do the following:
   a. (Optional) Type a description for the staging distribution.
   b. Choose **Next**.
   c. Modify the configuration of the staging distribution. For more information about the settings that you can update, see the section called “Updating primary and staging distributions”.

When you are finished modifying the staging distribution's configuration, choose **Next**.

d. Use the console to specify the **Traffic configuration**. This determines how CloudFront routes traffic to the staging distribution. (CloudFront stores the traffic configuration in a *continuous deployment policy*.)

   For more information about the options in a **Traffic configuration**, see the section called “Routing requests to the staging distribution”.

   When you are finished with the **Traffic configuration**, choose **Next**.

e. Review the configuration for the staging distribution, including the traffic configuration, then choose **Create staging distribution**.

When you finish the **Create staging distribution** wizard in the CloudFront console, CloudFront does the following:

- Creates a staging distribution with the settings that you specified (in step 5c)
- Creates a continuous deployment policy with the traffic configuration that you specified (in step 5d)
- Attaches the continuous deployment policy to the primary distribution that you created the staging distribution from

When the primary distribution's configuration, with the attached continuous deployment policy, deploys to edge locations, CloudFront begins sending the specified portion of traffic to the staging distribution based on the traffic configuration.

To update a staging distribution (console)

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Distributions**.
3. Choose the primary distribution. This is the distribution that currently serves production traffic, the one from which you created the staging distribution.
4. Choose **View staging distribution**.
5. Use the console to modify the configuration of the staging distribution. For more information about the settings that you can update, see the section called “Updating primary and staging distributions”.

As soon as the staging distribution's configuration deploys to edge locations it takes effect for incoming traffic that's routed to the staging distribution.

To update a continuous deployment policy (console)

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Distributions**.
3. Choose the primary distribution. This is the distribution that currently serves production traffic, the one from which you created the staging distribution.
4. In the **Continuous deployment** section, choose **Edit policy**.
5. Modify the traffic configuration in the continuous deployment policy. When you are finished, choose **Save changes**.

When the primary distribution's configuration with the updated continuous deployment policy deploys to edge locations, CloudFront begins sending traffic to the staging distribution based on the updated traffic configuration.

To promote a staging distribution's configuration (console)

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Distributions**.
3. Choose the primary distribution. This is the distribution that currently serves production traffic, the one from which you created the staging distribution.
4. In the **Continuous deployment** section, choose **Promote**.
5. Type **`confirm`** and then choose **Promote**.

When you *promote* a staging distribution, CloudFront copies the configuration from the staging distribution to the primary distribution. CloudFront also disables the continuous deployment policy and routes all traffic to the primary distribution.

After you promote a configuration, you can reuse the same staging distribution the next time you want to test a configuration change.

CLI

To work with a staging distribution and a continuous deployment policy with the Amazon CLI, use the following procedures.

To create a staging distribution (CLI)

1. Use the **aws cloudfront get-distribution** and **grep** commands together to get the ETag value of the distribution that you want to use as the *primary distribution*. The primary distribution is one that currently serves production traffic, from which you will create the staging distribution.

   The following command shows an example. In the following example, replace *primary_distribution_ID* with the ID of the primary distribution.

   ```
   aws cloudfront get-distribution --id primary_distribution_ID | grep 'ETag'
   ```

   Copy the ETag value because you need it for the following step.

2. Use the **aws cloudfront copy-distribution** command to create a staging distribution. The following example command uses escape characters (\) and line breaks for readability, but you should omit these from the command. In the following example command:

   - Replace *primary_distribution_ID* with the ID of the primary distribution.
   - Replace *primary_distribution_ETag* with the ETag value of the primary distribution (that you got in the previous step).
   - (Optional) Replace *CLI_example* with the desired caller reference ID.

   ```
   aws cloudfront copy-distribution --primary-distribution-
   id primary_distribution_ID \
                                    --if-match primary_distribution_ETag \
                                    --staging \
                                    --caller-reference 'CLI_example'
   ```

   The command's output shows information about the staging distribution and its configuration. Copy the staging distribution's CloudFront domain name because you need it for a following step.

To create a continuous deployment policy (CLI with input file)

1. Use the following command to create file named `continuous-deployment-policy.yaml` that contains all of the input parameters for the **create-continuous-deployment-policy** command. The following command uses escape characters (\) and line breaks for readability, but you should omit these from the command.

```
aws cloudfront create-continuous-deployment-policy --generate-cli-skeleton yaml-input \
                                                    > continuous-deployment-policy.yaml
```

2. Open the file named `continuous-deployment-policy.yaml` that you just created. Edit the file to specify the continuous deployment policy settings that you want, then save the file. When you edit the file:

   - In the `StagingDistributionDnsNames` section:
     - Change the value of `Quantity` to `1`.
     - For `Items`, paste the CloudFront domain name of the staging distribution (that you saved from a previous step).
   - In the `TrafficConfig` section:
     - Choose a `Type`, either `SingleWeight` or `SingleHeader`.
     - Remove the settings for the other type. For example, if you want a weight-based traffic configuration, set `Type` to `SingleWeight` and then remove the `SingleHeaderConfig` settings.
     - To use a weight-based traffic configuration, set the value of `Weight` to a decimal number between `.01` (one percent) and `.15` (fifteen percent).

   For more information about the options in `TrafficConfig`, see the section called "Routing requests to the staging distribution" and the section called "Session stickiness for weight-based configurations".

3. Use the following command to create the continuous deployment policy using input parameters from the `continuous-deployment-policy.yaml` file.

```
aws cloudfront create-continuous-deployment-policy --cli-input-yaml file://
continuous-deployment-policy.yaml
```

Copy the `Id` value in the command's output. This is the continuous deployment policy ID, and you need it in a following step.

To attach a continuous deployment policy to a primary distribution (CLI with input file)

1. Use the following command to save the primary distribution's configuration to a file named `primary-distribution.yaml`. Replace *`primary_distribution_ID`* with the primary distribution's ID.

   ```
   aws cloudfront get-distribution-config --id primary_distribution_ID --output
    yaml > primary-distribution.yaml
   ```

2. Open the file named `primary-distribution.yaml` that you just created. Edit the file, making the following changes:

   - Paste the continuous deployment policy ID (that you copied from a previous step) into the `ContinuousDeploymentPolicyId` field.
   - Rename the `ETag` field to `IfMatch`, but don't change the field's value.

   Save the file when finished.

3. Use the following command to update the primary distribution to use the continuous deployment policy. Replace *`primary_distribution_ID`* with the primary distribution's ID.

   ```
   aws cloudfront update-distribution --id primary_distribution_ID --cli-input-yaml
    file://primary-distribution.yaml
   ```

When the primary distribution's configuration, with the attached continuous deployment policy, deploys to edge locations, CloudFront begins sending the specified portion of traffic to the staging distribution based on the traffic configuration.

To update a staging distribution (CLI with input file)

1. Use the following command to save the staging distribution's configuration to a file named `staging-distribution.yaml`. Replace *`staging_distribution_ID`* with the staging distribution's ID.

```
aws cloudfront get-distribution-config --id staging_distribution_ID --output
 yaml > staging-distribution.yaml
```

2. Open the file named `staging-distribution.yaml` that you just created. Edit the file, making the following changes:

   - Modify the configuration of the staging distribution. For more information about the settings that you can update, see the section called “Updating primary and staging distributions”.
   - Rename the `ETag` field to `IfMatch`, but don't change the field's value.

   Save the file when finished.

3. Use the following command to update the staging distribution's configuration. Replace *`staging_distribution_ID`* with the staging distribution's ID.

```
aws cloudfront update-distribution --id staging_distribution_ID --cli-input-yaml
 file://staging-distribution.yaml
```

As soon as the staging distribution's configuration deploys to edge locations it takes effect for incoming traffic that's routed to the staging distribution.

To update a continuous deployment policy (CLI with input file)

1. Use the following command to save the continuous deployment policy's configuration to a file named `continuous-deployment-policy.yaml`. Replace *`continuous_deployment_policy_ID`* with the continuous deployment policy's ID. The following command uses escape characters (\) and line breaks for readability, but you should omit these from the command.

```
aws cloudfront get-continuous-deployment-policy-config --
id continuous_deployment_policy_ID \
                                                    --output yaml >
 continuous-deployment-policy.yaml
```

2. Open the file named `continuous-deployment-policy.yaml` that you just created. Edit the file, making the following changes:

   - Modify the configuration of the continuous deployment policy as desired. For example, you can change from using a header-based to a weight-based traffic configuration, or you can change the percentage of traffic (weight) for a weight-based configuration. For more information, see the section called “Routing requests to the staging distribution” and the section called “Session stickiness for weight-based configurations”.
   - Rename the `ETag` field to `IfMatch`, but don't change the field's value.

   Save the file when finished.

3. Use the following command to update the continuous deployment policy. Replace *continuous_deployment_policy_ID* with the continuous deployment policy's ID. The following command uses escape characters (\) and line breaks for readability, but you should omit these from the command.

```
aws cloudfront update-continuous-deployment-policy --
id continuous_deployment_policy_ID \
                                                --cli-input-yaml file://
continuous-deployment-policy.yaml
```

When the primary distribution's configuration with the updated continuous deployment policy deploys to edge locations, CloudFront begins sending traffic to the staging distribution based on the updated traffic configuration.

To promote a staging distribution's configuration (CLI)

- Use the **aws cloudfront update-distribution-with-staging-config** command to promote the staging distribution's configuration to the primary distribution. The following example command uses escape characters (\) and line breaks for readability, but you should omit these from the command. In the following example command:

- Replace *primary_distribution_ID* with the ID of the primary distribution.
- Replace *staging_distribution_ID* with the ID of the staging distribution.
- Replace *primary_distribution_ETag* and *staging_distribution_ETag* with the ETag values of the primary and staging distributions. Make sure the primary distribution's value is first, as shown in the example.

```
aws cloudfront update-distribution-with-staging-config --
id primary_distribution_ID \
                                                    --staging-distribution-
id staging_distribution_ID \
                                                    --if-match
 'primary_distribution_ETag,staging_distribution_ETag'
```

When you *promote* a staging distribution, CloudFront copies the configuration from the staging distribution to the primary distribution. CloudFront also disables the continuous deployment policy and routes all traffic to the primary distribution.

After you promote a configuration, you can reuse the same staging distribution the next time you want to test a configuration change.

API

To create a staging distribution and continuous deployment policy with the CloudFront API, use the following API operations:

- CopyDistribution
- CreateContinuousDeploymentPolicy

For more information about the fields that you specify in these API calls, see the following:

- the section called “Routing requests to the staging distribution”
- the section called “Session stickiness for weight-based configurations”
- The API reference documentation for your Amazon SDK or other API client

After you create a staging distribution and a continuous deployment policy, use UpdateDistribution (on the primary distribution) to attach the continuous deployment policy to the primary distribution.

To update the configuration of a staging distribution, use UpdateDistribution (on the staging distribution) to modify the configuration of the staging distribution. For more information about the settings that you can update, see the section called "Updating primary and staging distributions".

To promote a staging distribution's configuration to the primary distribution, use UpdateDistributionWithStagingConfig.

For more information about the fields that you specify in these API calls, see the API reference documentation for your Amazon SDK or other API client.

Monitoring a staging distribution

To monitor the performance of a staging distribution, you can use the same metrics, logs, and reports that CloudFront provides for all distributions. For example:

- You can view the default CloudFront distribution metrics (such as total requests and error rate) in the CloudFront console, and you can turn on additional metrics (such as cache hit rate and error rate by status code) for an additional cost. You can also create alarms based on these metrics.
- You can view standard logs and real-time logs to get detailed information about the requests that are received by the staging distribution. Standard logs contain the following two fields that help you identify the primary distribution that the request was originally sent to before CloudFront routed it to the staging distribution: `primary-distribution-id` and `primary-distribution-dns-name`.
- You can view and download reports in the CloudFront console, for example the cache statistics report.

Understanding how continuous deployment works

The following topics explain how CloudFront continuous deployment works.

Topics

- Routing requests to the staging distribution

- Session stickiness for weight-based configurations
- Updating primary and staging distributions
- Primary and staging distributions don't share a cache

Routing requests to the staging distribution

When you use CloudFront continuous deployment, you don't need to change anything about the viewer requests. Viewers cannot send requests directly to a staging distribution using a DNS name, IP address, or CNAME. Instead, viewers send requests to the primary (production) distribution, and CloudFront routes some of those requests to the staging distribution based on the traffic configuration settings in the continuous deployment policy. There are two types of traffic configurations:

Weight-based

A weight-based configuration routes the specified percentage of viewer requests to the staging distribution. When you use a weight-based configuration, you can also enable *session stickiness*, which helps make sure that CloudFront treats requests from the same viewer as part of a single session. For more information, see the section called “Session stickiness for weight-based configurations”.

Header-based

A header-based configuration routes requests to the staging distribution when the viewer request contains a specific HTTP header (you specify the header and the value). Requests that don't contain the specified header and value are routed to the primary distribution. This configuration is useful for local testing, or when you have control over the viewer requests.

> **Note**
>
> Headers routed to your staging distribution must contain the prefix `aws-cf-cd-`.

Session stickiness for weight-based configurations

When you use a weight-based configuration to route traffic to a staging distribution, you can also enable *session stickiness*, which helps make sure that CloudFront treats requests from the same viewer as a single session. When you enable session stickiness, CloudFront sets a cookie so that all

requests from the same viewer in a single session are served by one distribution, either the primary or staging.

When you enable session stickiness, you can also specify the *idle duration*. If the viewer is idle (sends no requests) for this amount of time, the session expires and CloudFront treats future requests from this viewer as a new session. You specify the idle duration as a number of seconds, from 300 (five minutes) to 3600 (one hour).

In the following cases, CloudFront resets all sessions (even active ones) and considers all requests to be a new session:

- You disable or enable the continuous deployment policy
- You disable or enable the session stickiness setting

Updating primary and staging distributions

When a primary distribution has an attached continuous deployment policy, the following configuration changes are available for both primary and staging distributions:

- All cache behavior settings, including the default cache behavior
- All origin settings (origins and origin groups)
- Custom error responses (error pages)
- Geographic restrictions
- Default root object
- Logging settings
- Description (comment)

You can also update external resources that are referenced in a distribution's configuration—such as a cache policy, a response headers policy, a CloudFront function, or a Lambda@Edge function.

Primary and staging distributions don't share a cache

The primary and staging distributions don't share a cache. When CloudFront sends the first request to a staging distribution, its cache is empty. As requests arrive at the staging distribution, it begins caching responses (if configured to do so).

Quotas and other considerations for continuous deployment

CloudFront continuous deployment is subject to the following quotas and other considerations.

Quotas

- Maximum number of staging distributions per Amazon Web Services account: 20
- Maximum number of continuous deployment policies per Amazon Web Services account: 20
- Maximum percentage of traffic you can send to a staging distribution in a weight-based configuration: 15%
- Minimum and maximum values for session stickiness idle duration: 300–3600 seconds

For more information, see *Quotas*.

> **Note**
>
> When using continuous deployment and your primary distribution is set with OAC for S3 bucket access, update your S3 bucket policy to allow access for the staging distribution. For example S3 bucket policies, see the section called "Giving the origin access control permission to access the S3 bucket".

HTTP/3

You cannot use continuous deployment with a distribution that supports HTTP/3.

Cases when CloudFront sends all requests to the primary distribution

In certain cases, such as periods of high resource utilization, CloudFront might send all requests to the primary distribution regardless of what's specified in the continuous deployment policy.

CloudFront sends all requests to the primary distribution during peak traffic hours, regardless of what's specified in the continuous deployment policy. Peak traffic refers to the traffic on the *CloudFront service*, and not the traffic on your distribution.

Using various origins with CloudFront distributions

When you create a distribution, you specify the *origin* where CloudFront sends requests for the files. You can use several different kinds of origins with CloudFront. For example, you can use an

Amazon S3 bucket, a MediaStore container, a MediaPackage channel, an Application Load Balancer, or an Amazon Lambda function URL.

Topics

- Using an Amazon S3 bucket
- Using a MediaStore container or a MediaPackage channel
- Using an Application Load Balancer
- Using a Lambda function URL
- Using Amazon EC2 (or another custom origin)
- Using CloudFront origin groups

Using an Amazon S3 bucket

The following topics describe the different ways that you can use an Amazon S3 bucket as the origin for a CloudFront distribution.

Topics

- Using a standard Amazon S3 bucket
- Using Amazon S3 Object Lambda
- Using Amazon S3 Access Point
- Using an Amazon S3 bucket that's configured as a website endpoint
- Adding CloudFront to an existing Amazon S3 bucket
- Moving an Amazon S3 bucket to a different Amazon Web Services Region

Using a standard Amazon S3 bucket

When you use Amazon S3 as an origin for your distribution, you place the objects that you want CloudFront to deliver in an Amazon S3 bucket. You can use any method that is supported by Amazon S3 to get your objects into Amazon S3. For example, you can use the Amazon S3 console or API, or a third-party tool. You can create a hierarchy in your bucket to store the objects, just as you would with any other standard Amazon S3 bucket.

Using an existing Amazon S3 bucket as your CloudFront origin server doesn't change the bucket in any way; you can still use it as you normally would to store and access Amazon S3 objects at the standard Amazon S3 price. You incur regular Amazon S3 charges for storing the objects in the

bucket. For more information about the charges to use CloudFront, see Amazon CloudFront Pricing. For more information about using CloudFront with an existing S3 bucket, see the section called "Adding CloudFront to an existing Amazon S3 bucket".

> **Important**
>
> For your bucket to work with CloudFront, the name must conform to DNS naming requirements. For more information, go to Bucket naming rules in the *Amazon Simple Storage Service User Guide*.

When you specify an Amazon S3 bucket as an origin for CloudFront, we recommend that you use the following format:

bucket-name.s3.*region*.amazonaws.com

When you specify the bucket name in this format, you can use the following CloudFront features:

- Configure CloudFront to communicate with your Amazon S3 bucket using SSL/TLS. For more information, see the section called "Using HTTPS with CloudFront".
- Use an origin access control to require that viewers access your content using CloudFront URLs, not by using Amazon S3 URLs. For more information, see the section called "Restricting access to an Amazon Simple Storage Service origin".
- Update the content of your bucket by submitting POST and PUT requests to CloudFront. For more information, see the section called "HTTP methods" in the topic the section called "How CloudFront processes and forwards requests to your Amazon S3 origin".

Don't specify the bucket using the following formats:

- The Amazon S3 path style: s3.amazonaws.com/*bucket-name*
- The Amazon S3 CNAME

Using Amazon S3 Object Lambda

When you create an Object Lambda Access Point, Amazon S3 automatically generates a unique alias for your Object Lambda Access Point. You can use this alias instead of an Amazon S3 bucket name as an origin for your CloudFront distribution.

When you use an Object Lambda Access Point alias as an origin for CloudFront, we recommend that you use the following format:

alias.s3.*region*.amazonaws.com

For more information about finding the *alias*, see How to use a bucket-style alias for your S3 bucket Object Lambda Access Point in the *Amazon S3 User Guide*.

> **Important**
>
> When you use an Object Lambda Access Point as an origin for CloudFront, you must use origin access control.

For an example use case, see Use Amazon S3 Object Lambda with Amazon CloudFront to Tailor Content for End Users.

CloudFront treats an Object Lambda Access Point origin the same as a standard Amazon S3 bucket origin.

If you're using Amazon S3 Object Lambda as an origin for your distribution, you must configure the following four permissions.

Object Lambda Access Point

To add permissions for the Object Lambda Access Point

1. Sign in to the Amazon Web Services Management Console and open the Amazon S3 console at https://console.amazonaws.cn/s3/.
2. In the navigation pane, choose **Object Lambda Access Points**.
3. Choose the Object Lambda Access Point that you want to use.
4. Choose the **Permissions** tab.
5. Choose **Edit** in the **Object Lambda Access Point policy** section.
6. Paste the following policy into the **Policy** field.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3-object-lambda:Get*",
            "Resource": "arn:aws:s3-object-lambda:region:Amazon-account-
ID:accesspoint/Object-Lambda-Access-Point-name",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:cloudfront::Amazon-account-
ID:distribution/CloudFront-distribution-ID"
                }
            }
        }
    ]
}
```

7. Choose **Save changes**.

Amazon S3 Access Point

To add permissions for the Amazon S3 Access Point

1. Sign in to the Amazon Web Services Management Console and open the Amazon S3 console at https://console.amazonaws.cn/s3/.
2. In the navigation pane, choose **Access Points**.
3. Choose the Amazon S3 Access Point that you want to use.
4. Choose the **Permissions** tab.
5. Choose **Edit** in the **Access Point policy** section.
6. Paste the following policy into the **Policy** field.

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "s3objlambda",
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:region:Amazon-account-ID:accesspoint/Access-Point-
name",
                "arn:aws:s3:region:Amazon-account-ID:accesspoint/Access-Point-
name/object/*"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "s3-object-lambda.amazonaws.com"
                }
            }
        }
    ]
}
```

7. Choose **Save**.

Amazon S3 bucket

To add permissions to the Amazon S3 bucket

1. Sign in to the Amazon Web Services Management Console and open the Amazon S3 console at https://console.amazonaws.cn/s3/.
2. In the navigation pane, choose **Buckets**.
3. Choose the Amazon S3 bucket that you want to use.
4. Choose the **Permissions** tab.
5. Choose **Edit** in the **Bucket policy** section.
6. Paste the following policy into the **Policy** field.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
```

```
            "Action": "*",
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:DataAccessPointAccount": "Amazon-account-ID"
                }
            }
        }
    ]
}
```

7. Choose **Save changes**.

Amazon Lambda function

To add permissions to the Lambda function

1. Sign in to the Amazon Web Services Management Console and open the Amazon Lambda console at https://console.amazonaws.cn/lambda/.
2. In the navigation pane, choose **Functions**.
3. Choose the Amazon Lambda function that you want to use.
4. Choose the **Configuration** tab, then choose **Permissions**.
5. Choose **Add permissions** in the **Resource-based policy statements** section.
6. Choose **Amazon Web Services account**.
7. Enter a name for **Statement ID**.
8. Enter `cloudfront.amazonaws.com` for **Principal**.
9. Choose `lambda:InvokeFunction` from the **Action** dropdown menu.
10. Choose **Save**.

Using Amazon S3 Access Point

When you use an S3 Access Point, Amazon S3 automatically generates a unique alias for you. You can use this alias instead of an Amazon S3 bucket name as an origin for your CloudFront distribution.

When you use an Amazon S3 Access Point alias as an origin for CloudFront, we recommend that you use the following format:

alias.s3.*region*.amazonaws.com

For more information about finding the *alias*, see Using a bucket-style alias for your S3 bucket access point in the *Amazon S3 User Guide*.

> **Important**
>
> When you use an Amazon S3 Access Point as an origin for CloudFront, you must use origin access control.

CloudFront treats an Amazon S3 Access Point origin the same as a standard Amazon S3 bucket origin.

If you're using Amazon S3 Object Lambda as an origin for your distribution, you must configure the following two permissions.

Amazon S3 Access Point

To add permissions for the Amazon S3 Access Point

1. Sign in to the Amazon Web Services Management Console and open the Amazon S3 console at https://console.amazonaws.cn/s3/.
2. In the navigation pane, choose **Access Points**.
3. Choose the Amazon S3 Access Point that you want to use.
4. Choose the **Permissions** tab.
5. Choose **Edit** in the **Access Point policy** section.
6. Paste the following policy into the **Policy** field.

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "s3objlambda",
            "Effect": "Allow",
```

```
            "Principal": {"Service": "cloudfront.amazonaws.com"},
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:region:Amazon-account-ID:accesspoint/Access-Point-
name",
                "arn:aws:s3:region:Amazon-account-ID:accesspoint/Access-Point-
name/object/*"
            ],
            "Condition": {
                "StringEquals": {"aws:SourceArn": "arn:aws:cloudfront::Amazon-
account-ID:distribution/CloudFront-distribution-ID"}
            }
        }
    ]
}
```

7. Choose **Save**.

Amazon S3 bucket

To add permissions to the Amazon S3 bucket

1. Sign in to the Amazon Web Services Management Console and open the Amazon S3 console at https://console.amazonaws.cn/s3/.
2. In the navigation pane, choose **Buckets**.
3. Choose the Amazon S3 bucket that you want to use.
4. Choose the **Permissions** tab.
5. Choose **Edit** in the **Bucket policy** section.
6. Paste the following policy into the **Policy** field.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "*",
            "Resource": [
```

```
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:DataAccessPointAccount": "Amazon-account-ID"
                }
            }
        }
    ]
}
```

7. Choose **Save changes**.

Using an Amazon S3 bucket that's configured as a website endpoint

You can use an Amazon S3 bucket that's configured as a website endpoint as a custom origin with CloudFront. When you configure your CloudFront distribution, for the origin, enter the Amazon S3 static website hosting endpoint for your bucket. This value appears in the Amazon S3 console, on the **Properties** tab, in the **Static website hosting** pane. For example:

`http://bucket-name.s3-website-region.amazonaws.com`

For more information about specifying Amazon S3 static website endpoints, see Website endpoints in the *Amazon Simple Storage Service User Guide*.

When you specify the bucket name in this format as your origin, you can use Amazon S3 redirects and Amazon S3 custom error documents. For more information, see Configuring a custom error document and Configuring a redirect in the *Amazon Simple Storage Service User Guide*. (CloudFront also provides custom error pages. For more information, see the section called "Creating a custom error page for specific HTTP status codes".)

Using an Amazon S3 bucket as your CloudFront origin server doesn't change the bucket in any way. You can still use it as you normally would and you incur regular Amazon S3 charges. For more information about the charges to use CloudFront, see Amazon CloudFront Pricing.

> **Note**
>
> If you use the CloudFront API to create your distribution with an Amazon S3 bucket that is configured as a website endpoint, you must configure it by using `CustomOriginConfig`, even though the website is hosted in an Amazon S3 bucket. For more information about

creating distributions by using the CloudFront API, see CreateDistribution in the *Amazon CloudFront API Reference*.

Adding CloudFront to an existing Amazon S3 bucket

If you store your objects in an Amazon S3 bucket, you can either have users get your objects directly from S3, or you can configure CloudFront to get your objects from S3 and then distribute them to your users. Using CloudFront can be more cost effective if your users access your objects frequently because, at higher usage, the price for CloudFront data transfer is lower than the price for Amazon S3 data transfer. In addition, downloads are faster with CloudFront than with Amazon S3 alone because your objects are stored closer to your users.

Note

If you want CloudFront to respect Amazon S3 cross-origin resource sharing settings, configure CloudFront to forward the `Origin` header to Amazon S3. For more information, see the section called "Caching content based on request headers".

If you currently distribute content directly from your Amazon S3 bucket using your own domain name (such as example.com) instead of the domain name of your Amazon S3 bucket (such as DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com), you can add CloudFront with no disruption by using the following procedure.

To add CloudFront when you're already distributing your content from Amazon S3

1. Create a CloudFront distribution. For more information, see the section called "Steps for creating a distribution".

   When you create the distribution, specify the name of your Amazon S3 bucket as the origin server.

   Important

   For your bucket to work with CloudFront, the name must conform to DNS naming requirements. For more information, go to Bucket naming rules in the *Amazon Simple Storage Service User Guide*.

If you're using a CNAME with Amazon S3, specify the CNAME for your distribution, too.

2. Create a test webpage that contains links to publicly readable objects in your Amazon S3 bucket, and test the links. For this initial test, use the CloudFront domain name of your distribution in the object URLs, for example, `https://d111111abcdef8.cloudfront.net/images/image.jpg`.

   For more information about the format of CloudFront URLs, see the section called "Customizing file URLs".

3. If you're using Amazon S3 CNAMEs, your application uses your domain name (for example, example.com) to reference the objects in your Amazon S3 bucket instead of using the name of your bucket (for example, DOC-EXAMPLE-BUCKET.s3.amazonaws.com). To continue using your domain name to reference objects instead of using the CloudFront domain name for your distribution (for example, d111111abcdef8.cloudfront.net), you need to update your settings with your DNS service provider.

   For Amazon S3 CNAMEs to work, your DNS service provider must have a CNAME resource record set for your domain that currently routes queries for the domain to your Amazon S3 bucket. For example, if a user requests this object:

   `https://example.com/images/image.jpg`

   The request is automatically rerouted, and the user sees this object:

   `https://DOC-EXAMPLE-BUCKET.s3.amazonaws.com/images/image.jpg`

   To route queries to your CloudFront distribution instead of your Amazon S3 bucket, you need to use the method provided by your DNS service provider to update the CNAME resource record set for your domain. This updated CNAME record redirects DNS queries from your domain to the CloudFront domain name for your distribution. For more information, see the documentation provided by your DNS service provider.

   Note

   If you're using Route 53 as your DNS service, you can use either a CNAME resource record set or an alias resource record set. For information about editing resource record sets, see Editing records. For information about alias resource record sets, see Choosing

between alias and non-alias records. Both topics are in the *Amazon Route 53 Developer Guide*.

For more information about using CNAMEs with CloudFront, see the section called "Using custom URLs".

After you update the CNAME resource record set, it can take up to 72 hours for the change to propagate throughout the DNS system, although it usually happens faster. During this time, some requests for your content will continue to be routed to your Amazon S3 bucket, and others will be routed to CloudFront.

Moving an Amazon S3 bucket to a different Amazon Web Services Region

If you're using Amazon S3 as the origin for a CloudFront distribution and you move the bucket to a different Amazon Web Services Region, CloudFront can take up to an hour to update its records to use the new Region when both of the following are true:

- You're using a CloudFront origin access identity (OAI) to restrict access to the bucket.
- You move the bucket to an Amazon S3 Region that requires Signature Version 4 for authentication.

When you're using OAIs, CloudFront uses the Region (among other values) to calculate the signature that it uses to request objects from your bucket. For more information about OAIs, see the section called "Using an origin access identity (legacy, not recommended)". For a list of Amazon Web Services Regions that support Signature Version 2, see Signature Version 2 signing process in the *Amazon Web Services General Reference*.

To force a faster update to CloudFront's records, you can update your CloudFront distribution, for example, by updating the **Description** field on the **General** tab in the CloudFront console. When you update a distribution, CloudFront immediately checks the Region that your bucket is in. Propagation of the change to all edge locations should take only a few minutes.

Using a MediaStore container or a MediaPackage channel

To stream video using CloudFront, you can set up an Amazon S3 bucket that is configured as a MediaStore container, or create a channel and endpoints with MediaPackage. Then you create and configure a distribution in CloudFront to stream the video.

For more information and step-by-step instructions, see the following topics:

- the section called “Serving video using AWS Elemental MediaStore as the origin”
- the section called “Serving live video formatted with AWS Elemental MediaPackage”

Using an Application Load Balancer

If your origin is one or more HTTP servers (web servers) hosted on one or more Amazon EC2 instances, you can use an Application Load Balancer to distribute traffic to the instances. For more information about using an Application Load Balancer as your origin for CloudFront, including how to make sure that viewers can only access your web servers through CloudFront and not by accessing the load balancer directly, see the section called “Restricting access to Application Load Balancers”.

Using a Lambda function URL

A Lambda function URL is a dedicated HTTPS endpoint for an Amazon Lambda function. You can use a Lambda function URL to build a serverless web application entirely within Amazon Lambda. You can invoke the Lambda web application directly through the function URL, with no need to integrate with API Gateway or an Application Load Balancer.

If you build a serverless web application using Lambda functions with function URLs, you can add CloudFront to get the following benefits:

- Speed up your application by caching content closer to viewers
- Use a custom domain name for your web application
- Route different URL paths to different Lambda functions using CloudFront cache behaviors
- Block specific requests using CloudFront geographic restrictions or Amazon WAF (or both)
- Use Amazon WAF with CloudFront to help protect your application from malicious bots, help prevent common application exploits, and enhance protection from DDoS attacks

To use a Lambda function URL as the origin for a CloudFront distribution, specify the full domain name of the Lambda function URL as the origin domain. A Lambda function URL domain name uses the following format:

function-URL-ID.lambda-url.*AWS-Region*.on.aws

When you use a Lambda function URL as the origin for a CloudFront distribution, you must make sure that the function URL is publicly accessible. To do this, you set the `AuthType` parameter of the function URL to `NONE` and allow the `lambda:InvokeFunctionUrl` permission in a resource-based policy. For more information, see Using the NONE AuthType in the *Amazon Lambda Developer Guide*. However, you can also add a custom origin header to the requests that CloudFront sends to the origin, and write function code to return an error response if the header is not present in the request. This helps to make sure that users can only access your web application through CloudFront, not directly using the Lambda function URL.

For more information about Lambda function URLs, see the following topics in the *Amazon Lambda Developer Guide*:

- Lambda function URLs – A general overview of the Lambda function URLs feature
- Invoking Lambda function URLs – Includes details about the request and response payloads to use for coding your serverless web application

Using Amazon EC2 (or another custom origin)

A custom origin is an HTTP server, for example, a web server. The HTTP server can be an Amazon EC2 instance or an HTTP server that you host somewhere else. An Amazon S3 origin configured as a website endpoint is also considered a custom origin.

When you use your own HTTP server as a custom origin, you specify the DNS name of the server, along with the HTTP and HTTPS ports and the protocol that you want CloudFront to use when fetching objects from your origin.

Most CloudFront features are supported when you use a custom origin with the exception of private content. Although you can use a signed URL to distribute content from a custom origin, for CloudFront to access the custom origin, the origin must remain publicly accessible. For more information, see the section called “Restricting content with signed URLs and signed cookies”.

Follow these guidelines for using Amazon EC2 instances and other custom origins with CloudFront.

- Host and serve the same content on all servers that are serving content for the same CloudFront origin. For more information, see the section called “Origin settings” in the the section called “Values that you specify” topic.
- Log the `X-Amz-Cf-Id` header entries on all servers in case you need Amazon Web Services Support or CloudFront to use this value for debugging.

- Restrict requests to the HTTP and HTTPS ports that your custom origin listens on.
- Synchronize the clocks of all servers in your implementation. Note that CloudFront uses Coordinated Universal Time (UTC) for signed URLs and signed cookies, for logs, and reports. In addition, if you monitor CloudFront activity using CloudWatch metrics, note that CloudWatch also uses UTC.
- Use redundant servers to handle failures.
- For information about using a custom origin to serve private content, see the section called " Restricting access to files on custom origins".
- For information about request and response behavior and about supported HTTP status codes, see *Request and response behavior*.

If you use Amazon EC2 for a custom origin, we recommend that you do the following:

- Use an Amazon Machine Image that automatically installs the software for a web server. For more information, see the Amazon EC2 documentation.
- Use an Elastic Load Balancing load balancer to handle traffic across multiple Amazon EC2 instances and to isolate your application from changes to Amazon EC2 instances. For example, if you use a load balancer, you can add and delete Amazon EC2 instances without changing your application. For more information, see the Elastic Load Balancing documentation.
- When you create your CloudFront distribution, specify the URL of the load balancer for the domain name of your origin server. For more information, see the section called "Creating a distribution".

Using CloudFront origin groups

You can specify an origin group for your CloudFront origin if, for example, you want to configure origin failover for scenarios when you need high availability. Use origin failover to designate a primary origin for CloudFront plus a second origin that CloudFront automatically switches to when the primary origin returns specific HTTP status code failure responses.

For more information, including the steps for setting up an origin group, see the section called "Increasing availability with origin failover".

Using custom URLs by adding alternate domain names (CNAMEs)

When you create a distribution, CloudFront provides a domain name for it, such as d111111abcdef8.cloudfront.net. Instead of using this provided domain name, you can use an alternate domain name (also known as a CNAME).

To use your own domain name, such as www.example.com, see the following sections:

Topics

- Adding an alternate domain name
- Moving an alternate domain name to a different distribution
- Removing an alternate domain name
- Using wildcards in alternate domain names
- Requirements for using alternate domain names
- Restrictions on using alternate domain names

Adding an alternate domain name

The following task list describes how to use the CloudFront console to add an alternate domain name to your distribution so that you can use your own domain name in your links instead of the CloudFront domain name. For information about updating your distribution using the CloudFront API, see Working with distributions.

> **Note**
>
> If you want viewers to use HTTPS with your alternate domain name, see Using alternate domain names and HTTPS.

Before you begin: Make sure that you do the following before you update your distribution to add an alternate domain name:

- Register the domain name with Route 53 or another domain registrar.

- Get an SSL/TLS certificate from an authorized certificate authority (CA) that covers the domain name. Add the certificate to your distribution to validate that you are authorized to use the domain. For more information, see Requirements for using alternate domain names.

Adding an alternate domain name

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose the ID for the distribution that you want to update.
3. On the **General** tab, choose **Edit**.
4. Update the following values:

   Alternate Domain Names (CNAMEs)

   Add your alternate domain names. Separate domain names with commas, or type each domain name on a new line.

   SSL Certificate

   Choose the following setting:

   - **Use HTTPS** – Choose **Custom SSL Certificate**, and then choose a certificate from the list. The list includes certificates provisioned by Amazon Certificate Manager (ACM), certificates that you purchased from another CA and uploaded to ACM, and certificates that you purchased from another CA and uploaded to the IAM certificate store.

     If you uploaded a certificate to the IAM certificate store but it doesn't appear in the list, review the procedure Importing an SSL/TLS certificate to confirm that you correctly uploaded the certificate.

     If you choose this setting, we recommend that you use only an alternate domain name in your object URLs (https://www.example.com/logo.jpg). If you use your CloudFront distribution domain name (https://d111111abcdef8.cloudfront.net.cloudfront.net/logo.jpg), a viewer might behave as follows, depending on the value that you choose for **Clients Supported**:

     - **All Clients**: If the viewer doesn't support SNI, it displays a warning because the CloudFront domain name doesn't match the domain name in your TLS/SSL certificate.
     - **Only Clients that Support Server Name Indication (SNI)**: CloudFront drops the connection with the viewer without returning the object.

Clients Supported

Choose an option:

- **All Clients**: CloudFront serves your HTTPS content using dedicated IP addresses. If you select this option, you incur additional charges when you associate your SSL/TLS certificate with a distribution that is enabled. For more information, see Amazon CloudFront Pricing.
- **Only Clients that Support Server Name Indication (SNI) (Recommended)**: Older browsers or other clients that don't support SNI must use another method to access your content.

For more information, see Choosing how CloudFront serves HTTPS requests.

5. Choose **Yes, Edit**.
6. On the **General** tab for the distribution, confirm that **Distribution Status** has changed to **Deployed**. If you try to use an alternate domain name before the updates to your distribution have been deployed, the links that you create in the following steps might not work.
7. Configure the DNS service for the alternate domain name (such as www.example.com) to route traffic to the CloudFront domain name for your distribution (such as d111111abcdef8.cloudfront.net). The method that you use depends on whether you're using Route 53 as the DNS service provider for the domain or another provider.

> **Note**
>
> If your DNS record already points to a distribution that is not the distribution that you are updating, then you only add the alternate domain name to your distribution after you update your DNS. For more information, see Restrictions on using alternate domain names.

Route 53

Create an alias resource record set. With an alias resource record set, you don't pay for Route 53 queries. In addition, you can create an alias resource record set for the root domain name (example.com), which DNS doesn't allow for CNAMEs. For more information, see Routing traffic to an Amazon CloudFront web distribution by using your domain name in the *Amazon Route 53 Developer Guide*.

Another DNS service provider

Use the method provided by your DNS service provider to add a CNAME record for your domain. This new CNAME record will redirect DNS queries from your alternate domain name (for example, www.example.com) to the CloudFront domain name for your distribution (for example, d111111abcdef8.cloudfront.net). For more information, see the documentation provided by your DNS service provider.

> **Important**
>
> If you already have an existing CNAME record for your alternate domain name, update that record or replace it with a new one that points to the CloudFront domain name for your distribution.

8. Using `dig` or a similar DNS tool, confirm that the DNS configuration that you created in the previous step points to the domain name for your distribution.

   The following example shows a `dig` request on the www.example.com domain, as well as the relevant part of the response.

```
PROMPT> dig www.example.com

; <<> DiG 9.3.3rc2 <<> www.example.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15917
;; flags: qr rd ra; QUERY: 1, ANSWER: 9, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.com.     IN    A

;; ANSWER SECTION:
www.example.com. 10800 IN CNAME d111111abcdef8.cloudfront.net.
...
```

   The answer section shows a CNAME record that routes queries for www.example.com to the CloudFront distribution domain name d111111abcdef8.cloudfront.net. If the name on the right side of CNAME is the domain name for your CloudFront distribution, the CNAME record is configured correctly. If it's any other value, for example, the domain name for your Amazon

S3 bucket, then the CNAME record is configured incorrectly. In that case, go back to step 7 and correct the CNAME record to point to the domain name for your distribution.

9. Test the alternate domain name by visiting URLs with your domain name instead of the CloudFront domain name for your distribution.
10. In your application, change the URLs for your objects to use your alternate domain name instead of the domain name of your CloudFront distribution.

Moving an alternate domain name to a different distribution

When you try to add an alternate domain name to a distribution but the alternate domain name is already in use on a different distribution, you get a `CNAMEAlreadyExists` error (One or more of the CNAMEs you provided are already associated with a different resource). For example, you get this error when you attempt to add www.example.com to a distribution, but www.example.com is already associated with a different distribution.

In that case, you might want to move the existing alternate domain name from one distribution (the *source distribution*) to another (the *target distribution*). The following steps are an overview of the process. For more information, follow the link at each step in the overview.

To move an alternate domain name

1. Set up the target distribution. This distribution must have an SSL/TLS certificate that covers the alternate domain name that you are moving. For more information, see Set up the target distribution.
2. Find the source distribution. You can use the Amazon Command Line Interface (Amazon CLI) to find the distribution that the alternate domain name is associated with. For more information, see Find the source distribution.
3. Move the alternate domain name. The way you do this depends on whether the source and target distributions are in the same Amazon account. For more information, see the section called “Move the alternate domain name”.

Set up the target distribution

Before you can move an alternate domain name, you must set up the target distribution (the distribution that you are moving the alternate domain name to).

To set up the target distribution

1. Get an SSL/TLS certificate that includes the alternate domain name that you are moving. If you don't have one, you can request one from Amazon Certificate Manager (ACM), or get one from another certificate authority (CA) and import it into ACM. Make sure that you request or import the certificate in the US East (N. Virginia) (`us-east-1`) Region.
2. If you haven't created the target distribution, create one now. As part of creating the target distribution, associate your certificate (from the previous step) with the distribution. For more information, see Creating a distribution.

   If you already have a target distribution, associate your certificate (from the previous step) with the target distribution. For more information, see Updating a distribution.
3. Create a DNS TXT record that associates the alternate domain name with the distribution domain name of the target distribution. Create your TXT record with an underscore (_) in front of the alternate domain name. The following shows an example TXT record in DNS:

   ```
   _www.example.com TXT d111111abcdef8.cloudfront.net
   ```

   CloudFront uses this TXT record to validate your ownership of the alternate domain name.

Find the source distribution

Before you move an alternate domain name from one distribution to another, you should find the *source distribution* (the distribution where the alternate domain name is currently in use). When you know the Amazon account ID of both the source and target distributions, you can determine how to move the alternate domain name.

To find the source distribution for the alternate domain name

1. Use the CloudFront list-conflicting-aliases command in the Amazon Command Line Interface (Amazon CLI) as shown in the following example. Replace *`www.example.com`* with the alternate domain name, and *`EDFDVBD6EXAMPLE`* with the ID of the target distribution that you set up previously. Run this command using credentials that are in the same Amazon account as the target distribution. To use this command, you must have `cloudfront:GetDistribution` and `cloudfront:ListConflictingAlias` permissions on the target distribution.

```
aws cloudfront list-conflicting-aliases --alias www.example.com --distribution-
id EDFDVBD6EXAMPLE
```

The command's output shows a list of all the alternate domain names that conflict or overlap with the provided one. For example:

- If you provide www.example.com to the command, the command's output includes www.example.com and the overlapping wildcard alternate domain name (*.example.com) if it exists.
- If you provide *.example.com to the command, the command's output includes *.example.com and any alternate domain names covered by that wildcard (for example, www.example.com, test.example.com, dev.example.com, and so on).

For each alternate domain name in the command's output, you can see the ID of the distribution that it's associated with, and the Amazon account ID that owns the distribution. The distribution and account IDs are partially hidden, which allows you to identify the distributions and accounts that you own, but helps to protect the information of ones that you don't own.

2. In the command's output, find the distribution for the alternate domain name that you are moving, and note the source distribution's Amazon account ID. Compare the source distribution's account ID with the account ID where you created the target distribution, and determine whether these two distribution are in the same Amazon account. This helps you determine how to move the alternate domain name.

   To move the alternate domain name, see the following topic.

Move the alternate domain name

Depending on your situation, choose from the following ways to move the alternate domain name:

If the source and target distributions are in the same Amazon account

Use the **associate-alias** command in the Amazon CLI to move the alternate domain name. This method works for all same-account moves, including when the alternate domain name is an apex domain (also called a *root domain*, like example.com). For more information, see the section called "Use `associate-alias` to move an alternate domain name".

If the source and target distributions are in different Amazon accounts

If you have access to the source distribution, the alternate domain name is *not* an apex domain (also called a root domain, like `example.com`), and you are not already using a wildcard that overlaps with that alternate domain name, use a wildcard to move the alternate domain name. For more information, see the section called "Use a wildcard to move an alternate domain name".

If you don't have access to the source distribution's Amazon account, you can try using the **associate-alias** command in the Amazon CLI to move the alternate domain name. If the source distribution is disabled, you can move the alternate domain name. For more information, see the section called "Use `associate-alias` to move an alternate domain name". If the **associate-alias** command doesn't work, contact Amazon Web Services Support. For more information, see the section called "Contact Amazon Web Services Support to move an alternate domain name".

Use `associate-alias` to move an alternate domain name

If the source distribution is in the same Amazon account as the target distribution, or if it's in a different account but disabled, you can use the CloudFront associate-alias command in the Amazon CLI to move the alternate domain name.

To use associate-alias to move an alternate domain name

1. Use the Amazon CLI to run the CloudFront **associate-alias** command, as shown in the following example. Replace *`www.example.com`* with the alternate domain name, and *`EDFDVBD6EXAMPLE`* with the target distribution ID. Run this command using credentials that are in the same Amazon account as the target distribution. Note the following restrictions for using this command:

   - You must have `cloudfront:AssociateAlias` and `cloudfront:UpdateDistribution` permissions on the target distribution.
   - If the source and target distributions are in the same Amazon account, you must have `cloudfront:UpdateDistribution` permission on the source distribution.
   - If the source and target distributions are in different Amazon accounts, the source distribution must be disabled.
   - The target distribution must be set up as described in the section called "Set up the target distribution".

```
aws cloudfront associate-alias --alias www.example.com --target-distribution-
id EDFDVBD6EXAMPLE
```

This command updates both distributions by removing the alternate domain name from the source distribution and adding it to the target distribution.

2. After the target distribution is fully deployed, update your DNS configuration to point the alternate domain name's DNS record to the distribution domain name of the target distribution.

Use a wildcard to move an alternate domain name

If the source distribution is in a different Amazon account than the target distribution, and the source distribution is enabled, you can use a wildcard to move the alternate domain name.

> **Note**
>
> You can't use a wildcard to move an apex domain (like example.com). To move an apex domain when the source and target distributions are in different Amazon accounts, contact Amazon Web Services Support. For more information, see the section called "Contact Amazon Web Services Support to move an alternate domain name".

To use a wildcard to move an alternate domain name

> **Note**
>
> This process involves multiple updates to your distributions. Wait for each distribution to fully deploy the latest change before proceeding to the next step.

1. Update the target distribution to add a wildcard alternate domain name that covers the alternate domain name that you are moving. For example, if the alternate domain name that you're moving is www.example.com, add the alternate domain name *.example.com to the target distribution. To do this, the SSL/TLS certificate on the target distribution must

include the wildcard domain name. For more information, see the section called “Updating a distribution”.

2. Update the DNS settings for the alternate domain name to point to the domain name of the target distribution. For example, if the alternate domain name that you’re moving is www.example.com, update the DNS record for www.example.com to route traffic to the domain name of the target distribution (for example d111111abcdef8.cloudfront.net).

   > **Note**
   >
   > Even after you update the DNS settings, the alternate domain name is still served by the source distribution because that’s where the alternate domain name is currently configured.

3. Update the source distribution to remove the alternate domain name. For more information, see Updating a distribution.
4. Update the target distribution to add the alternate domain name. For more information, see Updating a distribution.
5. Use **dig** (or a similar DNS query tool) to validate that the DNS record for the alternate domain name resolves to the domain name of the target distribution.
6. (Optional) Update the target distribution to remove the wildcard alternate domain name.

Contact Amazon Web Services Support to move an alternate domain name

If the source and target distributions are in different Amazon accounts, and you don’t have access to the source distribution’s Amazon account or can’t disable the source distribution, you can contact Amazon Web Services Support to move the alternate domain name.

To contact Amazon Web Services Support to move an alternate domain name

1. Set up a target distribution, including the DNS TXT record that points to the target distribution. For more information, see Set up the target distribution.
2. Contact Amazon Web Services Support to request that they verify that you own the domain, and move the domain to the new CloudFront distribution for you.
3. After the target distribution is fully deployed, update your DNS configuration to point the alternate domain name’s DNS record to the distribution domain name of the target distribution.

Removing an alternate domain name

If you want to stop routing traffic for a domain or subdomain to a CloudFront distribution, follow the steps in this section to update both the DNS configuration and the CloudFront distribution.

It's important that you remove the alternate domain names from the distribution as well as update your DNS configuration. This helps prevent issues later if you want to associate the domain name with another CloudFront distribution. If an alternate domain name is already associated with one distribution, it can't be set up with another.

> **Note**
>
> If you want to remove the alternate domain name from this distribution so you can add it to another one, follow the steps in Moving an alternate domain name to a different distribution. If you follow the steps here instead (to remove a domain) and then add the domain to another distribution, there will be a period of time during which the domain won't link to the new distribution because CloudFront is propagating to the updates to edge locations.

To remove an alternate domain name from a distribution

1. To start, route internet traffic for your domain to another resource that isn't your CloudFront distribution, such as an Elastic Load Balancing load balancer. Or you can delete the DNS record that's routing traffic to CloudFront.

   Do one of the following, depending on the DNS service for your domain:

   - **If you're using Route 53**, update or delete alias records or CNAME records. For more information, see Editing records or Deleting records.
   - **If you're using another DNS service provider**, use the method provided by the DNS service provider to update or delete the CNAME record that directs traffic to CloudFront. For more information, see the documentation provided by your DNS service provider.
2. After you update your domain's DNS records, wait until the changes have propagated and DNS resolvers are routing traffic to the new resource. You can check to see when this is complete by creating some test links that use your domain in the URL.

3. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home, and update your CloudFront distribution to remove the domain name by doing the following:

   a. Choose the ID for the distribution that you want to update.

   b. On the **General** tab, choose **Edit**.

   c. In **Alternate Domain Names (CNAMEs)**, remove the alternate domain name (or domain names) that you no longer want to use for your distribution.

   d. Choose **Yes, Edit**.

Using wildcards in alternate domain names

When you add alternate domain names, you can use the * wildcard at the beginning of a domain name instead of adding subdomains individually. For example, with an alternate domain name of *.example.com, you can use any domain name that ends with example.com in your URLs, such as www.example.com, product-name.example.com, marketing.product-name.example.com, and so on. The path to the object is the same regardless of the domain name, for example:

- www.example.com/images/image.jpg
- product-name.example.com/images/image.jpg
- marketing.product-name.example.com/images/image.jpg

Follow these requirements for alternate domain names that include wildcards:

- The alternate domain name must begin with an asterisk and a dot (*.).
- You *cannot* use a wildcard to replace part of a subdomain name, like this: *domain.example.com.
- You cannot replace a subdomain in the middle of a domain name, like this: subdomain.*.example.com.
- All alternate domain names, including alternate domain names that use wildcards, must be covered by the subject alternative name (SAN) on the certificate.

A wildcard alternate domain name, such as *.example.com, can include another alternate domain name that's in use, such as example.com.

Requirements for using alternate domain names

When you add an alternate domain name, such as www.example.com, to a CloudFront distribution, the following are requirements:

Alternate domain names must be lowercase

All alternate domain names (CNAMEs) must be lowercase.

Alternate domain names must be covered by a valid SSL/TLS certificate

To add an alternate domain name (CNAME) to a CloudFront distribution, you must attach to your distribution a trusted, valid SSL/TLS certificate that covers the alternate domain name. This ensures that only people with access to your domain's certificate can associate with CloudFront a CNAME related to your domain.

A trusted certificate is one that is issued by Amazon Certificate Manager (ACM) or by another valid certificate authority (CA). You can use a self-signed certificate to validate an existing CNAME, but *not* for a new CNAME. CloudFront supports the same certificate authorities as Mozilla. For the current list, see Mozilla Included CA Certificate List.

To verify an alternate domain name by using the certificate that you attach, including alternate domain names that include wildcards, CloudFront checks the subject alternative name (SAN) on the certificate. The alternate domain name that you're adding must be covered by the SAN.

> **Note**
>
> Only one certificate can be attached to a CloudFront distribution at a time.

You prove that you are authorized to add a specific alternate domain name to your distribution by doing one of the following:

- Attaching a certificate that includes the alternate domain name, like product-name.example.com.
- Attaching a certificate that includes a * wildcard at the beginning of a domain name, to cover multiple subdomains with one certificate. When you specify a wildcard, you can add multiple subdomains as alternate domain names in CloudFront.

The following examples illustrate how using wildcards in domain names in a certificate work to authorize you to add specific alternate domain names in CloudFront.

- You want to add marketing.example.com as an alternate domain name. You list in your certificate the following domain name: *.example.com. When you attach this certificate to CloudFront, you can add any alternate domain name for your distribution that replaces the wildcard at that level, including marketing.example.com. You can also, for example, add the following alternate domain names:
  - product.example.com
  - api.example.com

  However, you can't add alternate domain names that are at levels higher or lower than the wildcard. For example, you can't add the alternate domain names example.com or marketing.product.example.com.
- You want to add example.com as an alternate domain name. To do this, you must list the domain name example.com itself on the certificate that you attach to your distribution.
- You want to add marketing.product.example.com as an alternate domain name. To do this, you can list *.product.example.com on the certificate, or you can list marketing.product.example.com itself on the certificate.

Permission to change DNS configuration

When you add alternate domain names, you must create CNAME records to route DNS queries for the alternate domain names to your CloudFront distribution. To do this, you must have permission to create CNAME records with the DNS service provider for the alternate domain names that you're using. Typically, this means that you own the domains, but you might be developing an application for the domain owner.

Alternate domain names and HTTPS

If you want viewers to use HTTPS with an alternate domain name, you must complete some additional configuration. For more information, see Using alternate domain names and HTTPS.

Restrictions on using alternate domain names

Note the following restrictions on using alternate domain names:

Maximum number of alternate domain names

For the current maximum number of alternate domain names that you can add to a distribution, or to request a higher quota (formerly known as limit), see General quotas on distributions.

Duplicate and overlapping alternate domain names

You cannot add an alternate domain name to a CloudFront distribution if the same alternate domain name already exists in another CloudFront distribution, even if your Amazon account owns the other distribution.

However, you can add a wildcard alternate domain name, such as *.example.com, that includes (that overlaps with) a non-wildcard alternate domain name, such as www.example.com. If you have overlapping alternate domain names in two distributions, CloudFront sends the request to the distribution with the more specific name match, regardless of the distribution that the DNS record points to. For example, marketing.domain.com is more specific than *.domain.com.

Domain fronting

CloudFront includes protection against domain fronting occurring across different Amazon accounts. Domain fronting is a scenario in which a non-standard client creates a TLS/SSL connection to a domain name in one Amazon account, but then makes an HTTPS request for an unrelated name in another Amazon account. For example, the TLS connection might connect to www.example.com, and then send an HTTP request for www.example.org.

To prevent cases where domain fronting crosses different Amazon accounts, CloudFront makes sure that the Amazon account that owns the certificate that it serves for a specific connection always matches the Amazon account that owns the request that it handles on that same connection.

If the two Amazon account numbers do not match, CloudFront responds with an HTTP 421 Misdirected Request response to give the client a chance to connect using the correct domain.

Adding an alternate domain name at the top node (zone apex) for a domain

When you add an alternate domain name to a distribution, you typically create a CNAME record in your DNS configuration to route DNS queries for the domain name to your CloudFront distribution. However, you can’t create a CNAME record for the top node of a DNS namespace, also known as the zone apex; the DNS protocol doesn’t allow it. For example, if you register the DNS name example.com, the zone apex is example.com. You can’t create a CNAME record for example.com, but you can create CNAME records for www.example.com, newproduct.example.com, and so on.

If you’re using Route 53 as your DNS service, you can create an alias resource record set, which has two advantages over CNAME records. You can create an alias resource record set for a

domain name at the top node (example.com). In addition, when you use an alias resource record set, you don't pay for Route 53 queries.

> **Note**
>
> If you enable IPv6, you must create two alias resource record sets: one to route IPv4 traffic (an A record) and one to route IPv6 traffic (an AAAA record). For more information, see Enable IPv6 in the topic Values that you specify when you create or update a distribution.

For more information, see Routing traffic to an Amazon CloudFront web distribution by using your domain name in the *Amazon Route 53 Developer Guide*.

Using WebSockets with CloudFront distributions

Amazon CloudFront supports using WebSocket, a TCP-based protocol that is useful when you need long-lived bidirectional connections between clients and servers. A persistent connection is often a requirement with real-time applications. The scenarios in which you might use WebSockets include social chat platforms, online collaboration workspaces, multi-player gaming, and services that provide real-time data feeds like financial trading platforms. Data over a WebSocket connection can flow in both directions for full-duplex communication.

WebSocket functionality is automatically enabled to work with any distribution. To use WebSockets, configure one of the following in the cache behavior that's attached to your distribution:

- Forward all viewer request headers to your origin. (You can use the AllViewer managed origin request policy.)
- Specifically forward the `Sec-WebSocket-Key` and `Sec-WebSocket-Version` request headers in your origin request policy.

How the WebSocket protocol works

The WebSocket protocol is an independent, TCP-based protocol that allows you to avoid some of the overhead—and potentially increased latency—of HTTP.

To establish a WebSocket connection, the client sends a regular HTTP request that uses HTTP's upgrade semantics to change the protocol. The server can then complete the handshake. The WebSocket connection remains open and either the client or server can send data frames to each other without having to establish new connections each time.

By default, the WebSocket protocol uses port 80 for regular WebSocket connections and port 443 for WebSocket connections over TLS/SSL. The options that you choose for your CloudFront Viewer protocol policy and Protocol (custom origins only) apply to WebSocket connections as well as to HTTP traffic.

WebSocket requirements

WebSocket requests must comply with RFC 6455 in the following standard formats.

Sample client request:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: https://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

Sample server response:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

If the WebSocket connection is disconnected by the client or server, or by a network disruption, client applications are expected to re-initiate the connection with the server.

Recommended settings

In order to avoid unexpected compression-related issues when using WebSockets, we recommend that you include the following headers in an origin request policy:

- `Sec-WebSocket-Key`
- `Sec-WebSocket-Version`
- `Sec-WebSocket-Protocol`
- `Sec-WebSocket-Accept`
- `Sec-WebSocket-Extensions`

Working with policies

Amazon CloudFront offers three different kinds of *policies* that you can use to customize CloudFront in the following ways:

Specify cache and compression settings

With a CloudFront *cache policy*, you can specify the HTTP headers, cookies, and query strings that CloudFront includes in the *cache key*. The cache key determines whether a viewer's HTTP request results in a *cache hit* (the object is served to the viewer from the CloudFront cache). Including fewer values in the cache key increases the likelihood of a cache hit.

You can also use the cache policy to specify time to live (TTL) settings for objects in the CloudFront cache, and enable CloudFront to request and cache compressed objects.

Specify values to include in origin requests (but not in the cache key)

With a CloudFront *origin request policy*, you can specify the HTTP headers, cookies, and query strings that CloudFront includes in *origin requests*. These are the requests that CloudFront sends to the origin when there's a cache miss.

All of the values in the cache policy are automatically included in origin requests, but with an origin request policy you can include additional values in origin requests without including them in the cache key.

Specify HTTP headers to remove or add in viewer responses

With a CloudFront *response headers policy*, you can control the HTTP headers that CloudFront includes in HTTP responses that it sends to viewers (web browsers or other clients). You can remove headers from the origin's HTTP response, or add HTTP headers to the responses that CloudFront sends to viewers, without making any changes to your origin or writing any code.

For more information, see the following topics.

Topics

- the section called “Controlling the cache key”
- the section called “Controlling origin requests”
- Adding or removing response headers

Controlling the cache key

With Amazon CloudFront, you can control the *cache key* for objects that are cached at CloudFront edge locations. The cache key is the unique identifier for every object in the cache, and it determines whether a viewer request results in a *cache hit*. A cache hit occurs when a viewer request generates the same cache key as a prior request, and the object for that cache key is in the edge location's cache and valid. When there's a cache hit, the object is served to the viewer from a CloudFront edge location, which has the following benefits:

- Reduced load on your origin server
- Reduced latency for the viewer

You can get better performance from your website or application when you have a higher *cache hit ratio* (a higher proportion of viewer requests result in a cache hit). One way to improve your cache hit ratio is to include only the minimum necessary values in the cache key. For more information, see Understanding the cache key.

To control the cache key, you use a CloudFront *cache policy*. You attach a cache policy to one or more cache behaviors in a CloudFront distribution.

Topics

- Creating cache policies
- Understanding cache policies
- Using the managed cache policies
- Understanding the cache key

Creating cache policies

You can use a cache policy to improve your cache hit ratio by controlling the values (URL query strings, HTTP headers, and cookies) that are included in the cache key. You can create a cache policy in the CloudFront console, with the Amazon Command Line Interface (Amazon CLI), or with the CloudFront API.

After you create a cache policy, you attach it to one or more cache behaviors in a CloudFront distribution.

Console

To create a cache policy (console)

1. Sign in to the Amazon Web Services Management Console and open the **Policies** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home?#/policies.
2. Choose **Create cache policy**.
3. Choose the desired setting for this cache policy. For more information, see Understanding cache policies.
4. When finished, choose **Create**.

After you create a cache policy, you can attach it to a cache behavior.

To attach a cache policy to an existing distribution (console)

1. Open the **Distributions** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#/distributions.
2. Choose the distribution to update, then choose the **Behaviors** tab.
3. Choose the cache behavior to update, then choose **Edit**.

   Or, to create a new cache behavior, choose **Create behavior**.
4. In the **Cache key and origin requests** section, make sure that **Cache policy and origin request policy** is chosen.
5. For **Cache policy**, choose the cache policy to attach to this cache behavior.
6. At the bottom of the page, choose **Save changes**.

To attach a cache policy to a new distribution (console)

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose **Create distribution**.
3. In the **Cache key and origin requests** section, make sure that **Cache policy and origin request policy** is chosen.
4. For **Cache policy**, choose the cache policy to attach to this distribution's default cache behavior.

5. Choose the desired settings for the origin, default cache behavior, and other distribution settings. For more information, see Values that you specify when you create or update a distribution.
6. When finished, choose **Create distribution**.

CLI

To create a cache policy with the Amazon Command Line Interface (Amazon CLI), use the **aws cloudfront create-cache-policy** command. You can use an input file to provide the command's input parameters, rather than specifying each individual parameter as command line input.

To create a cache policy (CLI with input file)

1. Use the following command to create a file named `cache-policy.yaml` that contains all of the input parameters for the **create-cache-policy** command.

```
aws cloudfront create-cache-policy --generate-cli-skeleton yaml-input > cache-policy.yaml
```

2. Open the file named `cache-policy.yaml` that you just created. Edit the file to specify the cache policy settings that you want, then save the file. You can remove optional fields from the file, but don't remove the required fields.

   For more information about the cache policy settings, see Understanding cache policies.
3. Use the following command to create the cache policy using input parameters from the `cache-policy.yaml` file.

```
aws cloudfront create-cache-policy --cli-input-yaml file://cache-policy.yaml
```

   Make note of the `Id` value in the command's output. This is the cache policy ID, and you need it to attach the cache policy to a CloudFront distribution's cache behavior.

To attach a cache policy to an existing distribution (CLI with input file)

1. Use the following command to save the distribution configuration for the CloudFront distribution that you want to update. Replace *`distribution_ID`* with the distribution's ID.

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml >
 dist-config.yaml
```

2. Open the file named `dist-config.yaml` that you just created. Edit the file, making the following changes to each cache behavior that you are updating to use a cache policy.

   - In the cache behavior, add a field named `CachePolicyId`. For the field's value, use the cache policy ID that you noted after creating the policy.
   - Remove the `MinTTL`, `MaxTTL`, `DefaultTTL`, and `ForwardedValues` fields from the cache behavior. These settings are specified in the cache policy, so you can't include these fields and a cache policy in the same cache behavior.
   - Rename the `ETag` field to `IfMatch`, but don't change the field's value.

   Save the file when finished.

3. Use the following command to update the distribution to use the cache policy. Replace *distribution_ID* with the distribution's ID.

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

To attach a cache policy to a new distribution (CLI with input file)

1. Use the following command to create a file named `distribution.yaml` that contains all of the input parameters for the **create-distribution** command.

```
aws cloudfront create-distribution --generate-cli-skeleton yaml-input >
 distribution.yaml
```

2. Open the file named `distribution.yaml` that you just created. In the default cache behavior, in the `CachePolicyId` field, enter the cache policy ID that you noted after creating the policy. Continue editing the file to specify the distribution settings that you want, then save the file when finished.

For more information about the distribution settings, see Values that you specify when you create or update a distribution.

3. Use the following command to create the distribution using input parameters from the `distribution.yaml` file.

```
aws cloudfront create-distribution --cli-input-yaml file://distribution.yaml
```

API

To create a cache policy with the CloudFront API, use CreateCachePolicy. For more information about the fields that you specify in this API call, see Understanding cache policies and the API reference documentation for your Amazon SDK or other API client.

After you create a cache policy, you can attach it to a cache behavior, using one of the following API calls:

- To attach it to a cache behavior in an existing distribution, use UpdateDistribution.
- To attach it to a cache behavior in a new distribution, use CreateDistribution.

For both of these API calls, provide the cache policy's ID in the `CachePolicyId` field, inside a cache behavior. For more information about the other fields that you specify in these API calls, see Values that you specify when you create or update a distribution and the API reference documentation for your Amazon SDK or other API client.

Understanding cache policies

You can use a cache policy to improve your cache hit ratio by controlling the values (URL query strings, HTTP headers, and cookies) that are included in the cache key. CloudFront provides some predefined cache policies, known as *managed policies*, for common use cases. You can use these managed policies, or you can create your own cache policy that's specific to your needs. For more information about the managed policies, see Using the managed cache policies.

A cache policy contains the following settings, which are categorized into *policy information*, *time to live (TTL) settings*, and *cache key settings*.

Policy information

Name

A name to identify the cache policy. In the console, you use the name to attach the cache policy to a cache behavior.

Description

A comment to describe the cache policy. This is optional, but it can help you identify the purpose of the cache policy.

Time to live (TTL) settings

The time to live (TTL) settings work together with the `Cache-Control` and `Expires` HTTP headers (if they're in the origin response) to determine how long objects in the CloudFront cache remain valid.

Minimum TTL

The minimum amount of time, in seconds, that you want objects to stay in the CloudFront cache before CloudFront checks with the origin to see if the object has been updated. For more information, see Managing how long content stays in the cache (expiration).

Maximum TTL

The maximum amount of time, in seconds, that objects stay in the CloudFront cache before CloudFront checks with the origin to see if the object has been updated. CloudFront uses this setting only when the origin sends `Cache-Control` or `Expires` headers with the object. For more information, see Managing how long content stays in the cache (expiration).

Default TTL

The default amount of time, in seconds, that you want objects to stay in the CloudFront cache before CloudFront checks with the origin to see if the object has been updated. CloudFront uses this setting's value as the object's TTL only when the origin does *not* send `Cache-Control` or `Expires` headers with the object. For more information, see Managing how long content stays in the cache (expiration).

> **Note**
>
> If the **Minimum TTL**, **Maximum TTL**, and **Default TTL** settings are all set to 0, this disables CloudFront caching.

Cache key settings

Cache key settings specify the values in viewer requests that CloudFront includes in the cache key. The values can include URL query strings, HTTP headers, and cookies. The values that you include in the cache key are automatically included in requests that CloudFront sends to the origin, known as *origin requests*. For information about controlling origin requests without affecting the cache key, see Controlling origin requests.

Cache key settings include:

- Headers
- Cookies
- Query strings
- Compression support

Headers

The HTTP headers in viewer requests that CloudFront includes in the cache key and in origin requests. For headers, you can choose one of the following settings:

- **None** – The HTTP headers in viewer requests are *not* included in the cache key and are *not* automatically included in origin requests.
- **Include the following headers** – You specify which of the HTTP headers in viewer requests are included in the cache key and automatically included in origin requests.

When you use the **Include the following headers** setting, you specify HTTP headers by their name, not their value. For example, consider the following HTTP header:

```
Accept-Language: en-US,en;q=0.5
```

In this case, you specify the header as `Accept-Language`, not as `Accept-Language: en-US,en;q=0.5`. However, CloudFront includes the full header, including its value, in the cache key and in origin requests.

You can also include certain headers generated by CloudFront in the cache key. For more information, see the section called “Adding CloudFront request headers”.

Cookies

The cookies in viewer requests that CloudFront includes in the cache key and in origin requests. For cookies, you can choose one of the following settings:

- **None** – The cookies in viewer requests are *not* included in the cache key and are *not* automatically included in origin requests.
- **All** – All cookies in viewer requests are included in the cache key and are automatically included in origin requests.
- **Include specified cookies** – You specify which of the cookies in viewer requests are included in the cache key and automatically included in origin requests.
- **Include all cookies except** – You specify which of the cookies in viewer requests are *not* included in the cache key and are *not* automatically included in origin requests. All other cookies, except for the ones you specify, *are* included in the cache key and automatically included in origin requests.

When you use the **Include specified cookies** or **Include all cookies except** setting, you specify cookies by their name, not their value. For example, consider the following `Cookie` header:

```
Cookie: session_ID=abcd1234
```

In this case, you specify the cookie as `session_ID`, not as `session_ID=abcd1234`. However, CloudFront includes the full cookie, including its value, in the cache key and in origin requests.

Query strings

The URL query strings in viewer requests that CloudFront includes in the cache key and in origin requests. For query strings, you can choose one of the following settings:

- **None** – The query strings in viewer requests are *not* included in the cache key and are *not* automatically included in origin requests.
- **All** – All query strings in viewer requests are included in the cache key and are also automatically included in origin requests.

- **Include specified query strings** – You specify which of the query strings in viewer requests are included in the cache key and automatically included in origin requests.
- **Include all query strings except** – You specify which of the query strings in viewer requests are *not* included in the cache key and are *not* automatically included in origin requests. All other query strings, except for the ones you specify, *are* included in the cache key and automatically included in origin requests.

When you use the **Include specified query strings** or **Include all query strings except** setting, you specify query strings by their name, not their value. For example, consider the following URL path:

```
/content/stories/example-story.html?split-pages=false
```

In this case, you specify the query string as `split-pages`, not as `split-pages=false`. However, CloudFront includes the full query string, including its value, in the cache key and in origin requests.

Compression support

These settings enable CloudFront to request and cache objects that are compressed in the Gzip or Brotli compression formats, when the viewer supports it. These settings also allow CloudFront compression to work. Viewers indicate their support for these compression formats with the `Accept-Encoding` HTTP header.

> **Note**
>
> The Chrome and Firefox web browsers support Brotli compression only when the request is sent using HTTPS. These browsers do not support Brotli with HTTP requests.

Enable these settings when any of the following are true:

- Your origin returns Gzip compressed objects when viewers support them (requests contain the `Accept-Encoding` HTTP header with `gzip` as a value). In this case, use the **Gzip enabled** setting (set `EnableAcceptEncodingGzip` to `true` in the CloudFront API, Amazon SDKs, Amazon CLI, or Amazon CloudFormation).
- Your origin returns Brotli compressed objects when viewers support them (requests contain the `Accept-Encoding` HTTP header with `br` as a value). In this case, use the **Brotli enabled**

setting (set `EnableAcceptEncodingBrotli` to `true` in the CloudFront API, Amazon SDKs, Amazon CLI, or Amazon CloudFormation).

- The cache behavior that this cache policy is attached to is configured with CloudFront compression. In this case, you can enable caching for either Gzip or Brotli, or both. When CloudFront compression is enabled, enabling caching for both formats can help to reduce your costs for data transfer out to the internet.

> **Note**
>
> If you enable caching for one or both of these compression formats, do not include the `Accept-Encoding` header in an origin request policy that's associated with the same cache behavior. CloudFront always includes this header in origin requests when caching is enabled for either of these formats, so including `Accept-Encoding` in an origin request policy has no effect.

If your origin server does not return Gzip or Brotli compressed objects, or the cache behavior is not configured with CloudFront compression, don't enable caching for compressed objects. If you do, it might cause a decrease in your cache hit ratio.

The following explains how these settings affect a CloudFront distribution. All of the following scenarios assume that the viewer request includes the `Accept-Encoding` header. When the viewer request does not include the `Accept-Encoding` header, CloudFront doesn't include this header in the cache key and doesn't include it in the corresponding origin request.

When caching compressed objects is enabled for both compression formats

If the viewer supports both Gzip and Brotli—that is, if the `gzip` and `br` values are both in the `Accept-Encoding` header in the viewer request—CloudFront does the following:

- Normalizes the header to `Accept-Encoding: br,gzip` and includes the normalized header in the cache key. The cache key doesn't include other values that were in the `Accept-Encoding` header sent by the viewer.
- If the edge location has a Brotli or Gzip compressed object in the cache that matches the request and is not expired, the edge location returns the object to the viewer.
- If the edge location doesn't have a Brotli or Gzip compressed object in the cache that matches the request and is not expired, CloudFront includes the normalized header (`Accept-Encoding: br,gzip`) in the corresponding origin request. The origin request

doesn't include other values that were in the `Accept-Encoding` header sent by the viewer.

If the viewer supports one compression format but not the other—for example, if `gzip` is a value in the `Accept-Encoding` header in the viewer request but `br` is not—CloudFront does the following:

- Normalizes the header to `Accept-Encoding: gzip` and includes the normalized header in the cache key. The cache key doesn't include other values that were in the `Accept-Encoding` header sent by the viewer.
- If the edge location has a Gzip compressed object in the cache that matches the request and is not expired, the edge location returns the object to the viewer.
- If the edge location doesn't have a Gzip compressed object in the cache that matches the request and is not expired, CloudFront includes the normalized header (`Accept-Encoding: gzip`) in the corresponding origin request. The origin request doesn't include other values that were in the `Accept-Encoding` header sent by the viewer.

To understand what CloudFront does if the viewer supports Brotli but not Gzip, replace the two compression formats with each other in the preceding example.

If the viewer does not support Brotli or Gzip—that is, the `Accept-Encoding` header in the viewer request does not contain `br` or `gzip` as values—CloudFront:

- Doesn't include the `Accept-Encoding` header in the cache key.
- Includes `Accept-Encoding: identity` in the corresponding origin request. The origin request doesn't include other values that were in the `Accept-Encoding` header sent by the viewer.

When caching compressed objects is enabled for one compression format, but not the other

If the viewer supports the format for which caching is enabled—for example, if caching compressed objects is enabled for Gzip and the viewer supports Gzip (`gzip` is one of the values in the `Accept-Encoding` header in the viewer request)—CloudFront does the following:

- Normalizes the header to `Accept-Encoding: gzip` and includes the normalized header in the cache key.
- If the edge location has a Gzip compressed object in the cache that matches the request and is not expired, the edge location returns the object to the viewer.

- If the edge location doesn't have a Gzip compressed object in the cache that matches the request and is not expired, CloudFront includes the normalized header (`Accept-Encoding: gzip`) in the corresponding origin request. The origin request doesn't include other values that were in the `Accept-Encoding` header sent by the viewer.

This behavior is the same when the viewer supports both Gzip and Brotli (the `Accept-Encoding` header in the viewer request includes both `gzip` *and* `br` as values), because in this scenario, caching compressed objects for Brotli is not enabled.

To understand what CloudFront does if caching compressed objects is enabled for Brotli but not Gzip, replace the two compression formats with each other in the preceding example.

If the viewer does not support the compression format for which caching is enabled (the `Accept-Encoding` header in the viewer request doesn't contain the value for that format), CloudFront:

- Doesn't include the `Accept-Encoding` header in the cache key.
- Includes `Accept-Encoding: identity` in the corresponding origin request. The origin request doesn't include other values that were in the `Accept-Encoding` header sent by the viewer.

When caching compressed objects is disabled for both compression formats

When caching compressed objects is disabled for both compression formats, CloudFront treats the `Accept-Encoding` header the same as any other HTTP header in the viewer request. By default, it's not included in the cache key and it's not included in origin requests. You can include it in the headers list in a cache policy or an origin request policy the same as any other HTTP header.

Using the managed cache policies

CloudFront provides a set of managed cache policies that you can attach to any of your distribution's cache behaviors. With a managed cache policy, you don't need to write or maintain your own cache policy. The managed policies use settings that are optimized for specific use cases.

Topics

- Attaching a managed cache policy
- Available managed cache policies

Attaching a managed cache policy

To use a managed cache policy, you attach it to a cache behavior in your distribution. The process is the same as when you create a cache policy, but instead of creating a new one, you just attach one of the managed cache policies. You attach the policy either by name (with the console) or by ID (with the Amazon CLI or SDKs). The names and IDs are listed in the following section.

For more information, see Creating cache policies.

Available managed cache policies

The following topics describe the managed cache policies that you can use.

Topics

- Amplify
- CachingDisabled
- CachingOptimized
- CachingOptimizedForUncompressedObjects
- Elemental-MediaPackage

Amplify

View this policy in the CloudFront console

This policy is designed for use with an origin that is an Amazon Amplify web app.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

```
2e54312d-136d-493c-8eb9-b001f22f67d2
```

This policy has the following settings:

- **Minimum TTL:** 2 seconds
- **Maximum TTL:** 600 seconds (10 minutes)
- **Default TTL:** 2 seconds
- **Headers included in cache key:**

- `Authorization`
- `CloudFront-Viewer-Country`
- `Host`

The normalized `Accept-Encoding` header is also included because the cache compressed objects setting is enabled. For more information, see Compression support.

- **Cookies included in cache key:** All cookies are included.
- **Query strings included in cache key:** All query strings are included.
- **Cache compressed objects setting:** Enabled. For more information, see Compression support.

CachingDisabled

View this policy in the CloudFront console

This policy disables caching. This policy is useful for dynamic content and for requests that are not cacheable.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`4135ea2d-6df8-44a3-9df3-4b5a84be39ad`

This policy has the following settings:

- **Minimum TTL:** 0 seconds
- **Maximum TTL:** 0 seconds
- **Default TTL:** 0 seconds
- **Headers included in the cache key:** None
- **Cookies included in the cache key:** None
- **Query strings included in the cache key:** None
- **Cache compressed objects setting:** Disabled

CachingOptimized

View this policy in the CloudFront console

This policy is designed to optimize cache efficiency by minimizing the values that CloudFront includes in the cache key. CloudFront doesn't include any query strings or cookies in the cache key, and only includes the normalized `Accept-Encoding` header. This enables CloudFront to separately cache objects in the Gzip and Brotli compressions formats when the origin returns them or when CloudFront edge compression is enabled.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`658327ea-f89d-4fab-a63d-7e88639e58f6`

This policy has the following settings:

- **Minimum TTL:** 1 second.
- **Maximum TTL:** 31,536,000 seconds (365 days).
- **Default TTL:** 86,400 seconds (24 hours).
- **Headers included in the cache key:** None are explicitly included. The normalized `Accept-Encoding` header is included because the cache compressed objects setting is enabled. For more information, see Compression support.
- **Cookies included in the cache key:** None.
- **Query strings included in the cache key:** None.
- **Cache compressed objects setting:** Enabled. For more information, see Compression support.

CachingOptimizedForUncompressedObjects

View this policy in the CloudFront console

This policy is designed to optimize cache efficiency by minimizing the values included in the cache key. No query strings, headers, or cookies are included. This policy is identical to the previous one, but it disables the cache compressed objects setting.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`b2884449-e4de-46a7-ac36-70bc7f1ddd6d`

This policy has the following settings:

- **Minimum TTL:** 1 second
- **Maximum TTL:** 31,536,000 seconds (365 days)
- **Default TTL:** 86,400 seconds (24 hours)
- **Headers included in the cache key:** None
- **Cookies included in the cache key:** None
- **Query strings included in the cache key:** None
- **Cache compressed objects setting:** Disabled

Elemental-MediaPackage

View this policy in the CloudFront console

This policy is designed for use with an origin that is an AWS Elemental MediaPackage endpoint.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`08627262-05a9-4f76-9ded-b50ca2e3a84f`

This policy has the following settings:

- **Minimum TTL:** 0 seconds
- **Maximum TTL:** 31,536,000 seconds (365 days)
- **Default TTL:** 86,400 seconds (24 hours)
- **Headers included in the cache key:**
  - `Origin`

  The normalized `Accept-Encoding` header is also included because the cache compressed objects setting is enabled for Gzip. For more information, see Compression support.
- **Cookies included in the cache key:** None
- **Query strings included in the cache key:**
  - `aws.manifestfilter`
  - `start`
  - `end`
  - `m`

- **Cache compressed objects setting:** Enabled for Gzip. For more information, see Compression support.

Understanding the cache key

The *cache key* determines whether a viewer request to a CloudFront edge location results in a *cache hit*. The cache key is the unique identifier for an object in the cache. Each object in the cache has a unique cache key.

A cache hit occurs when a viewer request generates the same cache key as a prior request, and the object for that cache key is in the edge location's cache and valid. When there's a cache hit, the requested object is served to the viewer from a CloudFront edge location, which has the following benefits:

- Reduced load on your origin server
- Reduced latency for the viewer

You can get better performance from your website or application when you have a higher *cache hit ratio* (a higher proportion of viewer requests that result in a cache hit). One way to improve your cache hit ratio is to include only the minimum necessary values in the cache key. For more information, see the following sections.

You can modify the values (URL query strings, HTTP headers, and cookies) in the cache key by using a cache policy. (You can also modify the cache key using a Lambda@Edge function.) Before modifying the cache key, it's important to understand how your application is designed and when and how it might serve different responses based on characteristics of the viewer request. When a value in the viewer request determines the response that your origin returns, you should include that value in the cache key. But if you include a value in the cache key that doesn't affect the response that your origin returns, you might end up caching duplicate objects.

The default cache key

By default, the cache key for a CloudFront distribution includes the following information:

- The domain name of the CloudFront distribution (for example, d111111abcdef8.cloudfront.net)
- The URL path of the requested object (for example, `/content/stories/example-story.html`)

> **Note**
>
> The OPTIONS method is included in the cache key for OPTIONS requests. This means that responses to OPTIONS requests are cached separately from responses to GET and HEAD requests.

Other values from the viewer request are not included in the cache key, by default. Consider the following HTTP request from a web browser.

```
            GET /content/stories/example-story.html?ref=0123abc&split-pages=false
HTTP/1.1
            Host: d111111abcdef8.cloudfront.net
            User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/68.0
            Accept: text/html,*/*
            Accept-Language: en-US,en
            Cookie: session_id=01234abcd
            Referer: https://news.example.com/
```

When a viewer request like this example comes in to a CloudFront edge location, CloudFront uses the cache key to determine if there's a cache hit. By default, only the following components of the request are included in the cache key: `/content/stories/example-story.html` and `d111111abcdef8.cloudfront.net`. If the requested object is not in the cache (a cache miss), then CloudFront sends a request to the origin to get the object. After getting the object, CloudFront returns it to the viewer and stores it in the edge location's cache.

When CloudFront receives another request for the same object, as determined by the cache key, CloudFront serves the cached object to the viewer immediately, without sending a request to the origin. For example, consider the following HTTP request that comes in after the previous request.

```
            GET /content/stories/example-story.html?ref=xyz987&split-pages=true
HTTP/1.1
            Host: d111111abcdef8.cloudfront.net
            User-Agent: Mozilla/5.0 AppleWebKit/537.36 Chrome/83.0.4103.116
            Accept: text/html,*/*
            Accept-Language: en-US,en
            Cookie: session_id=wxyz9876
```

```
Referer: https://rss.news.example.net/
```

This request is for the same object as the previous request, but is different from the previous request. It has a different URL query string, different `User-Agent` and `Referer` headers, and a different `session_id` cookie. However, none of these values are part of the cache key by default, so this second request results in a cache hit.

Customizing the cache key

In some cases, you might want to include more information in the cache key, even though doing so might result in fewer cache hits. You specify what to include in the cache key by using a cache policy.

For example, if your origin server uses the `Accept-Language` HTTP header in viewer requests to return different content based on the viewer's language, you might want to include this header in the cache key. When you do that, CloudFront uses this header to determine cache hits, and includes the header in *origin requests* (requests that CloudFront sends to the origin when there's a cache miss).

One potential consequence of including additional values in the cache key is that CloudFront might end up caching duplicate objects because of the variation that can occur in viewer requests. For example, viewers might send any of the following values for the `Accept-Language` header:

- `en-US,en`
- `en,en-US`
- `en-US, en`
- `en-US`

All of these different values indicate that the viewer's language is English, but the variation can cause CloudFront to cache the same object multiple times. This can reduce cache hits and increase the number of origin requests. You could avoid this duplication by not including the `Accept-Language` header in the cache key, and instead configuring your website or application to use different URLs for content in different languages (for example, `/en-US/content/stories/example-story.html`).

For any given value that you intend to include in the cache key, you should make sure that you understand how many different variations of that value might appear in viewer requests. For

certain request values, it rarely makes sense to include them in the cache key. For example, the `User-Agent` header can have thousands of unique variations, so it's generally not a good candidate for including in the cache key. Cookies that have user-specific or session-specific values and are unique across thousands (or even millions) of requests are also not good candidates for cache key inclusion. If you do include these values in the cache key, each unique variation results in another copy of the object in the cache. If these copies of the object are not unique, or if you end up with such a large number of slightly different objects that each object only gets a small number of cache hits, you might want to consider a different approach. You can exclude these highly variable values from the cache key, or you can mark objects as non-cacheable.

Use caution when customizing the cache key. Sometimes it's desirable, but it can have unintended consequences such as caching duplicate objects, lowering your cache hit ratio, and increasing the number of origin requests. If your origin website or application needs to receive certain values from viewer requests for analytics, telemetry, or other uses, but these values don't change the object that the origin returns, use an origin request policy to include these values in origin requests but *not* include them in the cache key.

Controlling origin requests

When a viewer request to CloudFront results in a *cache miss* (the requested object is not cached at the edge location), CloudFront sends a request to the origin to retrieve the object. This is called an *origin request*. The origin request always includes the following information from the viewer request:

- The URL path (the path only, without URL query strings or the domain name)
- The request body (if there is one)
- The HTTP headers that CloudFront automatically includes in every origin request, including `Host`, `User-Agent`, and `X-Amz-Cf-Id`

Other information from the viewer request, such as URL query strings, HTTP headers, and cookies, is not included in the origin request by default. (Exception: With legacy cache settings, CloudFront forwards the headers to your origin by default.) However, you might want to receive some of this other information at the origin, for example to collect data for analytics or telemetry. You can use an *origin request policy* to control the information that's included in an origin request.

Origin request policies are separate from cache policies, which control the cache key. This separation enables you to receive additional information at the origin and also maintain a good

cache hit ratio (the proportion of viewer requests that result in a cache hit). You do this by separately controlling which information is included in origin requests (using the origin request policy) and which is included in the cache key (using the cache policy).

Although the two kinds of policies are separate, they are related. All URL query strings, HTTP headers, and cookies that you include in the cache key (using a cache policy) are automatically included in origin requests. Use the origin request policy to specify the information that you want to include in origin requests, but *not* include in the cache key. Just like a cache policy, you attach an origin request policy to one or more cache behaviors in a CloudFront distribution.

You can also use an origin request policy to add additional HTTP headers to an origin request that were not included in the viewer request. These additional headers are added by CloudFront before sending the origin request, with header values that are determined automatically based on the viewer request. For more information, see the section called “Adding CloudFront request headers”.

Topics

- Creating origin request policies
- Understanding origin request policies
- Using the managed origin request policies
- Adding CloudFront request headers
- Understanding how origin request policies and cache policies work together

Creating origin request policies

You can use an origin request policy to control the values (URL query strings, HTTP headers, and cookies) that are included in requests that CloudFront sends to your origin. You can create an origin request policy in the CloudFront console, with the Amazon Command Line Interface (Amazon CLI), or with the CloudFront API.

After you create an origin request policy, you attach it to one or more cache behaviors in a CloudFront distribution.

Origin request policies are not required. When a cache behavior does not have an origin request policy attached, the origin request includes all the values that are specified in the cache policy, but nothing more.

> **Note**
>
> To use an origin request policy, the cache behavior must also use a cache policy. You cannot use an origin request policy in a cache behavior without a cache policy.

Console

To create an origin request policy (console)

1. Sign in to the Amazon Web Services Management Console and open the **Policies** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home?#/policies.
2. Choose **Origin request**, then choose **Create origin request policy**.
3. Choose the desired setting for this origin request policy. For more information, see Understanding origin request policies.
4. When finished, choose **Create**.

After you create an origin request policy, you can attach it to a cache behavior.

To attach an origin request policy to an existing distribution (console)

1. Open the **Distributions** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#/distributions.
2. Choose the distribution to update, then choose the **Behaviors** tab.
3. Choose the cache behavior to update, then choose **Edit**.

   Or, to create a new cache behavior, choose **Create behavior**.
4. In the **Cache key and origin requests** section, make sure that **Cache policy and origin request policy** is chosen.
5. For **Origin request policy**, choose the origin request policy to attach to this cache behavior.
6. At the bottom of the page, choose **Save changes**.

To attach an origin request policy to a new distribution (console)

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose **Create distribution**.

3. In the **Cache key and origin requests** section, make sure that **Cache policy and origin request policy** is chosen.
4. For **Origin request policy**, choose the origin request policy to attach to this distribution's default cache behavior.
5. Choose the desired settings for the origin, default cache behavior, and other distribution settings. For more information, see Values that you specify when you create or update a distribution.
6. When finished, choose **Create distribution**.

CLI

To create an origin request policy with the Amazon Command Line Interface (Amazon CLI), use the **aws cloudfront create-origin-request-policy** command. You can use an input file to provide the command's input parameters, rather than specifying each individual parameter as command line input.

To create an origin request policy (CLI with input file)

1. Use the following command to create a file named `origin-request-policy.yaml` that contains all of the input parameters for the **create-origin-request-policy** command.

```
aws cloudfront create-origin-request-policy --generate-cli-skeleton yaml-input >
 origin-request-policy.yaml
```

2. Open the file named `origin-request-policy.yaml` that you just created. Edit the file to specify the origin request policy settings that you want, then save the file. You can remove optional fields from the file, but don't remove the required fields.

   For more information about the origin request policy settings, see Understanding origin request policies.

3. Use the following command to create the origin request policy using input parameters from the `origin-request-policy.yaml` file.

```
aws cloudfront create-origin-request-policy --cli-input-yaml file://origin-
request-policy.yaml
```

Make note of the `Id` value in the command's output. This is the origin request policy ID, and you need it to attach the origin request policy to a CloudFront distribution's cache behavior.

To attach an origin request policy to an existing distribution (CLI with input file)

1. Use the following command to save the distribution configuration for the CloudFront distribution that you want to update. Replace *`distribution_ID`* with the distribution's ID.

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml >
 dist-config.yaml
```

2. Open the file named `dist-config.yaml` that you just created. Edit the file, making the following changes to each cache behavior that you are updating to use an origin request policy.

   - In the cache behavior, add a field named `OriginRequestPolicyId`. For the field's value, use the origin request policy ID that you noted after creating the policy.
   - Rename the `ETag` field to `IfMatch`, but don't change the field's value.

   Save the file when finished.

3. Use the following command to update the distribution to use the origin request policy. Replace *`distribution_ID`* with the distribution's ID.

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

To attach an origin request policy to a new distribution (CLI with input file)

1. Use the following command to create a file named `distribution.yaml` that contains all of the input parameters for the **create-distribution** command.

```
aws cloudfront create-distribution --generate-cli-skeleton yaml-input >
 distribution.yaml
```

2. Open the file named `distribution.yaml` that you just created. In the default cache behavior, in the `OriginRequestPolicyId` field, enter the origin request policy ID that you noted after creating the policy. Continue editing the file to specify the distribution settings that you want, then save the file when finished.

   For more information about the distribution settings, see Values that you specify when you create or update a distribution.

3. Use the following command to create the distribution using input parameters from the `distribution.yaml` file.

   ```
   aws cloudfront create-distribution --cli-input-yaml file://distribution.yaml
   ```

API

To create an origin request policy with the CloudFront API, use CreateOriginRequestPolicy. For more information about the fields that you specify in this API call, see Understanding origin request policies and the API reference documentation for your Amazon SDK or other API client.

After you create an origin request policy, you can attach it to a cache behavior, using one of the following API calls:

- To attach it to a cache behavior in an existing distribution, use UpdateDistribution.
- To attach it to a cache behavior in a new distribution, use CreateDistribution.

For both of these API calls, provide the origin request policy's ID in the `OriginRequestPolicyId` field, inside a cache behavior. For more information about the other fields that you specify in these API calls, see Values that you specify when you create or update a distribution and the API reference documentation for your Amazon SDK or other API client.

Understanding origin request policies

CloudFront provides some predefined origin request policies, known as *managed policies*, for common use cases. You can use these managed policies, or you can create your own origin request policy that's specific to your needs. For more information about the managed policies, see Using the managed origin request policies.

An origin request policy contains the following settings, which are categorized into *policy information* and *origin request settings*.

Policy information

Name

A name to identify the origin request policy. In the console, you use the name to attach the origin request policy to a cache behavior.

Description

A comment to describe the origin request policy. This is optional.

Origin request settings

Origin request settings specify the values in viewer requests that are included in requests that CloudFront sends to the origin (known as origin requests). The values can include URL query strings, HTTP headers, and cookies. The values that you specify are included in origin requests, but are not included in the cache key. For information about controlling the cache key, see Controlling the cache key.

Headers

The HTTP headers in viewer requests that CloudFront includes in origin requests. For headers, you can choose one of the following settings:

- **None** – The HTTP headers in viewer requests are *not* included in origin requests.
- **All viewer headers** – All HTTP headers in viewer requests are included in origin requests.
- **All viewer headers and the following CloudFront headers** – All HTTP headers in viewer requests are included in origin requests. Additionally, you specify which of the CloudFront headers you want to add to origin requests. For more information about the CloudFront headers, see the section called “Adding CloudFront request headers”.
- **Include the following headers** – You specify which HTTP headers are included in origin requests.

> **Note**
>
> Do not specify a header that is already included in your **Origin Custom Headers** settings. For more information, see Configuring CloudFront to add custom headers to origin requests.

- **All viewer headers except** – You specify which HTTP headers are ***not*** included in origin requests. All other HTTP headers in viewer requests, except for the ones specified, are included.

When you use the **All viewer headers and the following CloudFront headers**, **Include the following headers**, or **All viewer headers except** setting, you specify HTTP headers by the header name only. CloudFront includes the full header, including its value, in origin requests.

> **Note**
>
> When you use the **All viewer headers except** setting to remove the viewer's `Host` header, CloudFront adds a new `Host` header with the origin's domain name to the origin request.

Cookies

The cookies in viewer requests that CloudFront includes in origin requests. For cookies, you can choose one of the following settings:

- **None** – The cookies in viewer requests are *not* included in origin requests.
- **All** – All cookies in viewer requests are included in origin requests.
- **Include the following cookies** – You specify which cookies in viewer requests are included in origin requests.
- **All cookies except** – You specify which cookies in viewer requests are ***not*** included in origin requests. All other cookies in viewer requests are included.

When you use the **Include the following cookies** or **All cookies except** setting, you specify cookies by their name only. CloudFront includes the full cookie, including its value, in origin requests.

Query strings

The URL query strings in viewer requests that CloudFront includes in origin requests. For query strings, you can choose one of the following settings:

- **None** – The query strings in viewer requests are *not* included in origin requests.
- **All** – All query strings in viewer requests are included in origin requests.
- **Include the following query strings** – You specify which query strings in viewer requests are included in origin requests.
- **All query strings except** – You specify which query strings in viewer requests are ***not*** included in origin requests. All other query strings are included.

When you use the **Include the following query strings** or **All query strings except** setting, you specify query strings by their name only. CloudFront includes the full query string, including its value, in origin requests.

Using the managed origin request policies

CloudFront provides a set of managed origin request policies that you can attach to any of your distribution's cache behaviors. With a managed origin request policy, you don't need to write or maintain your own origin request policy. The managed policies use settings that are optimized for specific use cases.

Topics

- Attaching a managed origin request policy
- Available managed origin request policies

Attaching a managed origin request policy

To use a managed origin request policy, you attach it to a cache behavior in your distribution. The process is the same as when you create an origin request policy, but instead of creating a new one, you just attach one of the managed origin request policies. You attach the policy either by name (with the console) or by ID (with the Amazon CLI or SDKs). The names and IDs are listed in the following section.

For more information, see Creating origin request policies.

Available managed origin request policies

The following topics describe the managed origin request policies that you can use.

Topics

- AllViewer
- AllViewerAndCloudFrontHeaders-2022-06
- AllViewerExceptHostHeader
- CORS-CustomOrigin
- CORS-S3Origin
- Elemental-MediaTailor-PersonalizedManifests
- UserAgentRefererHeaders

AllViewer

View this policy in the CloudFront console

This policy includes all values (headers, cookies, and query strings) from the viewer request.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

```
216adef6-5c7f-47e4-b989-5492eafa07d3
```

This policy has the following settings:

- **Headers included in origin requests:** All headers in the viewer request
- **Cookies included in origin requests:** All
- **Query strings included in origin requests:** All

AllViewerAndCloudFrontHeaders-2022-06

View this policy in the CloudFront console

This policy includes all values (headers, cookies, and query strings) from the viewer request, and all CloudFront headers that were released through June 2022 (CloudFront headers released after June 2022 are not included).

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`33f36d7e-f396-46d9-90e0-52428a34d9dc`

This policy has the following settings:

- **Headers included in origin requests:** All headers in the viewer request, and the following CloudFront headers:
  - `CloudFront-Forwarded-Proto`
  - `CloudFront-Is-Android-Viewer`
  - `CloudFront-Is-Desktop-Viewer`
  - `CloudFront-Is-IOS-Viewer`
  - `CloudFront-Is-Mobile-Viewer`
  - `CloudFront-Is-SmartTV-Viewer`
  - `CloudFront-Is-Tablet-Viewer`
  - `CloudFront-Viewer-Address`
  - `CloudFront-Viewer-ASN`
  - `CloudFront-Viewer-City`
  - `CloudFront-Viewer-Country`
  - `CloudFront-Viewer-Country-Name`
  - `CloudFront-Viewer-Country-Region`
  - `CloudFront-Viewer-Country-Region-Name`
  - `CloudFront-Viewer-Http-Version`
  - `CloudFront-Viewer-Latitude`
  - `CloudFront-Viewer-Longitude`
  - `CloudFront-Viewer-Metro-Code`
  - `CloudFront-Viewer-Postal-Code`
  - `CloudFront-Viewer-Time-Zone`
  - `CloudFront-Viewer-TLS`
- **Cookies included in origin requests:** All
- **Query strings included in origin requests:** All

AllViewerExceptHostHeader

View this policy in the CloudFront console

This policy does ***not*** include the `Host` header from the viewer request, but does include all others values (headers, cookies, and query strings) from the viewer request.

This policy also includes additional CloudFront request headers for HTTP protocol, HTTP version, TLS version, and all device type and viewer location headers.

This policy is intended for use with Amazon API Gateway and Amazon Lambda function URL origins. These origins expect the `Host` header to contain the origin domain name, not the domain name of the CloudFront distribution. Forwarding the `Host` header from the viewer request to these origins can prevent them from working.

> **Note**
>
> When you use this managed origin request policy to remove the viewer's `Host` header, CloudFront adds a new `Host` header with the origin's domain name to the origin request.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`b689b0a8-53d0-40ab-baf2-68738e2966ac`

This policy has the following settings:

- **Headers included in origin requests:** All headers in the viewer request ***except*** for the `Host` header
- **Cookies included in origin requests:** All
- **Query strings included in origin requests:** All

CORS-CustomOrigin

View this policy in the CloudFront console

This policy includes the header that enables cross-origin resource sharing (CORS) requests when the origin is a custom origin.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`59781a5b-3903-41f3-afcb-af62929ccde1`

This policy has the following settings:

- **Headers included in origin requests:**
  - `Origin`
- **Cookies included in origin requests:** None
- **Query strings included in origin requests:** None

CORS-S3Origin

View this policy in the CloudFront console

This policy includes the headers that enable cross-origin resource sharing (CORS) requests when the origin is an Amazon S3 bucket.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`88a5eaf4-2fd4-4709-b370-b4c650ea3fcf`

This policy has the following settings:

- **Headers included in origin requests:**
  - `Origin`
  - `Access-Control-Request-Headers`
  - `Access-Control-Request-Method`
- **Cookies included in origin requests:** None
- **Query strings included in origin requests:** None

Elemental-MediaTailor-PersonalizedManifests

View this policy in the CloudFront console

This policy is intended for use with an origin that is an AWS Elemental MediaTailor endpoint.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`775133bc-15f2-49f9-abea-afb2e0bf67d2`

This policy has the following settings:

- **Headers included in origin requests:**
  - `Origin`
  - `Access-Control-Request-Headers`
  - `Access-Control-Request-Method`
  - `User-Agent`
  - `X-Forwarded-For`
- **Cookies included in origin requests:** None
- **Query strings included in origin requests:** All

UserAgentRefererHeaders

View this policy in the CloudFront console

This policy includes only the `User-Agent` and `Referer` headers. It doesn't include any query strings or cookies.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`acba4595-bd28-49b8-b9fe-13317c0390fa`

This policy has the following settings:

- **Headers included in origin requests:**
  - `User-Agent`
  - `Referer`
- **Cookies included in origin requests:** None
- **Query strings included in origin requests:** None

Adding CloudFront request headers

You can configure CloudFront to add specific HTTP headers to the requests that CloudFront receives from viewers and forwards on to your origin or edge function. The values of these HTTP headers are based on characteristics of the viewer or the viewer request. The headers provide information about the viewer's device type, IP address, geographic location, request protocol (HTTP or HTTPS), HTTP version, TLS connection details, and JA3 fingerprint.

With these headers, your origin or your edge function can receive information about the viewer without the need for you to write your own code to determine this information. If your origin returns different responses based on the information in these headers, you can include them in the *cache key* so that CloudFront caches the responses separately. For example, your origin might respond with content in a specific language based on the country that the viewer is in, or with content tailored to a specific device type. Your origin might also write these headers to log files, which you can use to determine information about where your viewers are, which device types they're on, and more.

To include these headers in the cache key, use a *cache policy*. For more information, see the section called “Controlling the cache key” and the section called “Understanding the cache key”.

To receive these headers at your origin but not include them in the cache key, use an *origin request policy*. For more information, see the section called “Controlling origin requests”.

Topics

- Headers for determining the viewer's device type
- Headers for determining the viewer's location
- Headers for determining the viewer's header structure
- Other CloudFront headers

Headers for determining the viewer's device type

You can add the following headers to determine the viewer's device type. Based on the value of the `User-Agent` header, CloudFront sets the value of these headers to `true` or `false`. If a device falls into more than one category, more than one value can be `true`. For example, for some tablet devices, CloudFront sets both `CloudFront-Is-Mobile-Viewer` and `CloudFront-Is-Tablet-Viewer` to `true`.

- `CloudFront-Is-Android-Viewer` – Set to `true` when CloudFront determines that the viewer is a device with the Android operating system.
- `CloudFront-Is-Desktop-Viewer` – Set to `true` when CloudFront determines that the viewer is a desktop device.
- `CloudFront-Is-IOS-Viewer` – Set to `true` when CloudFront determines that the viewer is a device with an Apple mobile operating system, such as iPhone, iPod touch, and some iPad devices.
- `CloudFront-Is-Mobile-Viewer` – Set to `true` when CloudFront determines that the viewer is a mobile device.
- `CloudFront-Is-SmartTV-Viewer` – Set to `true` when CloudFront determines that the viewer is a smart TV.
- `CloudFront-Is-Tablet-Viewer` – Set to `true` when CloudFront determines that the viewer is a tablet.

Headers for determining the viewer's location

You can add the following headers to determine the viewer's location. CloudFront determines the values for these headers based on the viewer's IP address. For non-ASCII characters in these headers' values, CloudFront percent-encodes the character according to section 1.2 of RFC 3986.

- `CloudFront-Viewer-Address` – Contains the IP address of the viewer and the source port of the request. For example, a header value of `198.51.100.10:46532` means the viewer's IP address is 198.51.100.10 and the request source port is 46532.
- `CloudFront-Viewer-ASN` – Contains the autonomous system number (ASN) of the viewer.

> **Note**
>
> `CloudFront-Viewer-Address` and `CloudFront-Viewer-ASN` can be added in an origin request policy, but not in a cache policy.

- `CloudFront-Viewer-Country` – Contains the two-letter country code for the viewer's country. For a list of country codes, see ISO 3166-1 alpha-2.

When you add the following headers, CloudFront applies them to all requests *except* those that originate from the Amazon network:

- `CloudFront-Viewer-City` – Contains the name of the viewer's city.
- `CloudFront-Viewer-Country-Name` – Contains the name of the viewer's country.
- `CloudFront-Viewer-Country-Region` – Contains a code (up to three characters) that represent the viewer's region. The region is the first-level subdivision (the broadest or least specific) of the ISO 3166-2 code.
- `CloudFront-Viewer-Country-Region-Name` – Contains the name of the viewer's region. The region is the first-level subdivision (the broadest or least specific) of the ISO 3166-2 code.
- `CloudFront-Viewer-Latitude` – Contains the viewer's approximate latitude.
- `CloudFront-Viewer-Longitude` – Contains the viewer's approximate longitude.
- `CloudFront-Viewer-Metro-Code` – Contains the viewer's metro code. This is present only when the viewer is in the United States.
- `CloudFront-Viewer-Postal-Code` – Contains the viewer's postal code.
- `CloudFront-Viewer-Time-Zone` Contains the viewer's time zone, in IANA time zone database format (for example, `America/Los_Angeles`).

Headers for determining the viewer's header structure

You can add the following headers to help identify the viewer based on the headers that it sends. For example, different browsers may send HTTP headers in a certain order. If the browser specified in the `User-Agent` header doesn't match that browser's expected header order, you can deny the request. Additionally, if the `CloudFront-Viewer-Header-Count` value does not match the number of headers in `CloudFront-Viewer-Header-Order`, you can deny the request.

- `CloudFront-Viewer-Header-Order` – Contains the viewer's header names in the order requested, separated by a colon. For example: `CloudFront-Viewer-Header-Order: Host:User-Agent:Accept:Accept-Encoding`. Headers beyond the character limit of 7,680 are truncated.
- `CloudFront-Viewer-Header-Count` – Contains the total number of the viewer's headers.

Other CloudFront headers

You can add the following headers to determine the viewer's protocol, version, JA3 fingerprint, and TLS connection details:

- `CloudFront-Forwarded-Proto` – Contains the protocol of the viewer's request (HTTP or HTTPS).
- `CloudFront-Viewer-Http-Version` – Contains the HTTP version of the viewer's request.
- `CloudFront-Viewer-JA3-Fingerprint` – Contains the JA3 fingerprint of the viewer. The JA3 fingerprint can help you determine whether the request comes from a known client, whether that's malware or a malicious bot, or an expected (allow-listed) application. This header relies on the viewer's SSL/TLS `Client Hello` packet and is only present for HTTPS requests.

  > **Note**
  >
  > You can add `CloudFront-Viewer-JA3-Fingerprint` in an origin request policy, but not in a cache policy.

- `CloudFront-Viewer-TLS` – Contains the SSL/TLS version, the cipher, and information about the SSL/TLS handshake that was used for the connection between the viewer and CloudFront. The header value is in the following format:

  ```
  SSL/TLS_version:cipher:handshake_information
  ```

  For *`handshake_information`*, the header can contain the following values:

  - `fullHandshake` – A full handshake was performed for the SSL/TLS session.
  - `sessionResumed` – A previous SSL/TLS session was resumed.
  - `connectionReused` – A previous SSL/TLS connection was reused.

  The following are some example values for this header:

  ```
  TLSv1.3:TLS_AES_128_GCM_SHA256:sessionResumed
  ```

  ```
  TLSv1.2:ECDHE-ECDSA-AES128-GCM-SHA256:connectionReused
  ```

  ```
  TLSv1.1:ECDHE-RSA-AES128-SHA256:fullHandshake
  ```

  ```
  TLSv1:ECDHE-RSA-AES256-SHA:fullHandshake
  ```

For the full list of possible SSL/TLS versions and ciphers that can be in this header value, see the section called "Supported protocols and ciphers between viewers and CloudFront".

> **Note**
>
> You can add `CloudFront-Viewer-TLS` in an origin request policy, but not in a cache policy.

Understanding how origin request policies and cache policies work together

You can use a CloudFront origin request policy to control the requests that CloudFront sends to the origin, which are called *origin requests*. To use an origin request policy, you must attach a cache policy to the same cache behavior. You cannot use an origin request policy in a cache behavior without a cache policy. For more information, see the section called "Controlling origin requests".

Origin request policies and cache policies work together to determine the values that CloudFront includes in origin requests. All URL query strings, HTTP headers, and cookies that you specify in the cache key (using a cache policy) are automatically included in origin requests. Any additional query strings, headers, and cookies that you specify in an origin request policy are also included in origin requests (but not in the cache key).

Origin request policies and cache policies have settings that might appear to conflict with each other. For example, one policy might allow certain values while another policy blocks them. The following table explains which values CloudFront includes in origin requests when you use the settings of an origin request policy and a cache policy together. These settings generally apply to all types of values (query strings, headers, and cookies), with the exception that you cannot specify all headers or use a header block list in a cache policy.

	Origin request policy			
	None	**All**	**Allow list**	**Block list**
Cache policy				
None	No values from the viewer	All values from the viewer	Only the values specified in the	All values from the viewer

	Origin request policy			
	None	**All**	**Allow list**	**Block list**
	request are included in the origin request, except for the defaults that are included in every origin request. For more informati on, see the section called "Controlling origin requests".	request are included in the origin request.	origin request policy are included in the origin request.	request ***except*** those specified in the origin request policy are included in the origin request.
All **Note:** You cannot specify all headers in a cache policy.	All query strings and cookies from the viewer request are included in the origin request.	All values from the viewer request are included in the origin request.	All query strings and cookies from the viewer request, and any headers specified in the origin request policy, are included in the origin request.	All query strings and cookies from the viewer request are included in the origin request, even those specified in the origin request policy block list. The cache policy setting overrides the origin request policy block list.

	Origin request policy			
	None	**All**	**Allow list**	**Block list**
Allow list	Only the specified values from the viewer request are included in the origin request.	All values from the viewer request are included in the origin request.	All values specified in the cache policy or the origin request policy are included in the origin request.	The values specified in the cache policy are included in the origin request, even if those same values are specified in the origin request policy block list. The cache policy allow list overrides the origin request policy block list.
Block list **Note:** You cannot specify headers in a cache policy block list.	All query strings and cookies from the viewer request ***except*** those specified are included in the origin request.	All values from the viewer request are included in the origin request.	The values specified in the origin request policy are included in the origin request, even if those same values are specified in the cache policy block list. The origin request policy allow list overrides the cache policy block list.	All values from the viewer request ***except*** those specified in the cache policy or the origin request policy are included in the origin request.

Adding or removing HTTP headers in CloudFront responses

You can configure CloudFront to modify the HTTP headers in the responses that it sends to viewers. CloudFront can remove headers that it received from the origin, or add headers to the response, before sending the response to viewers. Making these changes doesn't require writing code or changing the origin.

For example, you can remove headers such as `X-Powered-By` and `Vary` so that CloudFront doesn't include these headers in the responses that it sends to viewers. Or, you can add HTTP headers such as the following:

- A `Cache-Control` header to control browser caching.
- An `Access-Control-Allow-Origin` header to enable cross-origin resource sharing (CORS). You can also add other CORS headers.
- A set of common security headers, such as `Strict-Transport-Security`, `Content-Security-Policy`, and `X-Frame-Options`.
- A `Server-Timing` header to see information that's related to the performance and routing of both the request and response through CloudFront.

To specify the headers that CloudFront adds or removes in HTTP responses, you use a *response headers policy*. You attach a response headers policy to one more *cache behaviors*, and CloudFront modifies the headers in the responses that it sends to requests that match the cache behavior. CloudFront modifies the headers in the responses that it serves from the cache and the ones that it forwards from the origin. If the origin response includes one or more of the headers that are added in a response headers policy, the policy can specify if CloudFront uses the header it received from the origin or overwrites that header with the one in the response headers policy.

CloudFront provides predefined response headers policies, known as *managed policies*, for common use cases. You can use these managed policies or create your own policies. You can attach a single response headers policy to multiple cache behaviors in multiple distributions in your Amazon Web Services account.

For more information, see the following topics.

Topics

- Creating response headers policies
- Using the managed response headers policies

- Understanding response headers policies

Creating response headers policies

You can use a response headers policy to specify the HTTP headers that Amazon CloudFront adds or removes in HTTP responses. For more information about response headers policies and reasons to use them, see the section called "Adding or removing response headers".

You can create a response headers policy in the CloudFront console. Or you can create one by using Amazon CloudFormation, the Amazon Command Line Interface (Amazon CLI), or the CloudFront API. After you create a response headers policy, you attach it to one or more cache behaviors in a CloudFront distribution.

Before you create a custom response headers policy, check if one of the managed response headers policies fits your use case. If one does, you can attach it to your cache behavior. That way, you don't need to create or manage your own response headers policy.

Console

To create a response headers policy (console)

1. Sign in to the Amazon Web Services Management Console, then go to the **Response headers** tab on the **Policies** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#/policies/responseHeaders.
2. Choose **Create response headers policy**.
3. In the **Create response headers policy** form, do the following:

   a. In the **Details** panel, enter a **Name** for the response headers policy and (optionally) a **Description** that explains what the policy is for.

   b. In the **Cross-origin resource sharing (CORS)** panel, choose the **Configure CORS** toggle and configure any CORS headers that you want to add to the policy. If you want the configured headers to override the headers that CloudFront receives from the origin, select the **Origin override** check box.

      For more information about the CORS headers settings, see the section called "CORS headers".

   c. In the **Security headers** panel, choose the toggle and configure each of the security headers that you want to add to the policy.

For more information about the security headers settings, see the section called "Security headers".

d. In the **Custom headers** panel, add any custom headers that you want to include in the policy.

   For more information about the custom headers settings, see the section called "Custom headers".

e. In the **Remove headers** panel, add the names of any headers that you want CloudFront to remove from the origin's response and not include in the response that CloudFront sends to viewers.

   For more information about the remove headers settings, see the section called "Remove headers".

f. In the **Server-Timing header** panel, choose the **Enable** toggle and enter a sampling rate (a number between 0 and 100, inclusive).

   For more information about the `Server-Timing` header, see the section called "Server-Timing header".

4. Choose **Create** to create the policy.

After you create a response headers policy, you can attach it to a cache behavior in a CloudFront distribution.

To attach a response headers policy to an existing distribution (console)

1. Open the **Distributions** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#/distributions.
2. Choose the distribution to update, then choose the **Behaviors** tab.
3. Select the cache behavior to update, then choose **Edit**.

   Or, to create a new cache behavior, choose **Create behavior**.
4. For **Response headers policy**, choose the policy to add to the cache behavior.
5. Choose **Save changes** to update the cache behavior. If you're creating a new cache behavior, choose **Create behavior**.

To attach a response headers policy to a new distribution (console)

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose **Create distribution**.
3. For **Response headers policy**, choose the policy to add to the cache behavior.
4. Choose the other settings for your distribution. For more information, see the section called "Values that you specify".
5. Choose **Create distribution** to create the distribution.

Amazon CloudFormation

To create a response headers policy with Amazon CloudFormation, use the `AWS::CloudFront::ResponseHeadersPolicy` resource type. The following example shows the Amazon CloudFormation template syntax, in YAML format, for creating a response headers policy.

```
Type: AWS::CloudFront::ResponseHeadersPolicy
Properties:
  ResponseHeadersPolicyConfig:
    Name: EXAMPLE-Response-Headers-Policy
    Comment: Example response headers policy for the documentation
    CorsConfig:
      AccessControlAllowCredentials: false
      AccessControlAllowHeaders:
        Items:
          - '*'
      AccessControlAllowMethods:
        Items:
          - GET
          - OPTIONS
      AccessControlAllowOrigins:
        Items:
          - https://example.com
          - https://docs.example.com
      AccessControlExposeHeaders:
        Items:
          - '*'
      AccessControlMaxAgeSec: 600
      OriginOverride: false
    CustomHeadersConfig:
```

```
      Items:
        - Header: Example-Custom-Header-1
          Value: value-1
          Override: true
        - Header: Example-Custom-Header-2
          Value: value-2
          Override: true
    SecurityHeadersConfig:
      ContentSecurityPolicy:
        ContentSecurityPolicy: default-src 'none'; img-src 'self'; script-src
 'self'; style-src 'self'; object-src 'none'; frame-ancestors 'none'
        Override: false
      ContentTypeOptions: # You don't need to specify a value for 'X-Content-Type-
Options'.
                          # Simply including it in the template sets its value to
 'nosniff'.
        Override: false
      FrameOptions:
        FrameOption: DENY
        Override: false
      ReferrerPolicy:
        ReferrerPolicy: same-origin
        Override: false
      StrictTransportSecurity:
        AccessControlMaxAgeSec: 63072000
        IncludeSubdomains: true
        Preload: true
        Override: false
      XSSProtection:
        ModeBlock: true # You can set ModeBlock to 'true' OR set a value for
 ReportUri, but not both
        Protection: true
        Override: false
    ServerTimingHeadersConfig:
      Enabled: true
      SamplingRate: 50
    RemoveHeadersConfig:
      Items:
        - Header: Vary
        - Header: X-Powered-By
```

For more information, see AWS::CloudFront::ResponseHeadersPolicy in the *Amazon CloudFormation User Guide*.

CLI

To create a response headers policy with the Amazon Command Line Interface (Amazon CLI), use the **aws cloudfront create-response-headers-policy** command. You can use an input file to provide the input parameters for the command, rather than specifying each individual parameter as command line input.

To create a response headers policy (CLI with input file)

1. Use the following command to create a file that's named `response-headers-policy.yaml`. This file contains all of the input parameters for the **create-response-headers-policy** command.

```
aws cloudfront create-response-headers-policy --generate-cli-skeleton yaml-input
 > response-headers-policy.yaml
```

2. Open the `response-headers-policy.yaml` file that you just created. Edit the file to specify a policy name and the desired response headers policy configuration, then save the file.

   For more information about the response headers policy settings, see the section called "Understanding response headers policies".

3. Use the following command to create the response headers policy. The policy that you create uses the input parameters from the `response-headers-policy.yaml` file.

```
aws cloudfront create-response-headers-policy --cli-input-yaml file://response-
headers-policy.yaml
```

   Make note of the `Id` value in the command output. This is the response headers policy ID. You need it to attach the policy to the cache behavior of a CloudFront distribution.

To attach a response headers policy to an existing distribution (CLI with input file)

1. Use the following command to save the distribution configuration for the CloudFront distribution that you want to update. Replace *`distribution_ID`* with the distribution ID.

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml >
 dist-config.yaml
```

2. Open the file that's named `dist-config.yaml` that you just created. Edit the file, making the following changes to the cache behavior to make it use the response headers policy.

   - In the cache behavior, add a field that's named `ResponseHeadersPolicyId`. For the field's value, use the response headers policy ID that you noted after creating the policy.
   - Rename the `ETag` field to `IfMatch`, but don't change the field's value.

   Save the file when finished.

3. Use the following command to update the distribution to use the response headers policy. Replace *`distribution_ID`* with the distribution ID.

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

To attach a response headers policy to a new distribution (CLI with input file)

1. Use the following command to create a file that's named `distribution.yaml`. This file contains all of the input parameters for the **create-distribution** command.

```
aws cloudfront create-distribution --generate-cli-skeleton yaml-input >
 distribution.yaml
```

2. Open the `distribution.yaml` file that you just created. In the default cache behavior, in the `ResponseHeadersPolicyId` field, enter the response headers policy ID that you noted after creating the policy. Continue editing the file to specify the distribution settings that you want, then save the file when finished.

   For more information about the distribution settings, see Values that you specify when you create or update a distribution.

3. Use the following command to create the distribution using input parameters from the `distribution.yaml` file.

```
aws cloudfront create-distribution --cli-input-yaml file://distribution.yaml
```

API

To create a response headers policy with the CloudFront API, use CreateResponseHeadersPolicy. For more information about the fields that you specify in this API call, see the section called "Understanding response headers policies" and the API reference documentation for your Amazon SDK or other API client.

After you create a response headers policy, you can attach it to a cache behavior, using one of the following API calls:

- To attach it to a cache behavior in an existing distribution, use UpdateDistribution.
- To attach it to a cache behavior in a new distribution, use CreateDistribution.

For both of these API calls, provide the response headers policy ID in the `ResponseHeadersPolicyId` field, inside a cache behavior. For more information about the other fields that you specify in these API calls, see Values that you specify when you create or update a distribution and the API reference documentation for your Amazon SDK or other API client.

Using the managed response headers policies

With a CloudFront response headers policy, you can specify the HTTP headers that Amazon CloudFront removes or adds in responses that it sends to viewers. For more information about response headers policies and reasons to use them, see the section called "Adding or removing response headers".

CloudFront provides managed response headers policies that you can attach to cache behaviors in your CloudFront distributions. With a managed response headers policy, you don't need to write or maintain your own policy. The managed policies contain sets of HTTP response headers for common use cases.

Topics

- Attaching a managed response headers policy
- Available managed response headers policies

Attaching a managed response headers policy

To use a managed response headers policy, you attach it to a cache behavior in your distribution. The process is the same as when you create a custom response headers policy. However, instead of creating a new policy, you attach one of the managed policies. You attach the policy either by name (with the console) or by ID (with Amazon CloudFormation, the Amazon CLI, or the Amazon SDKs). The names and IDs are listed in the following section.

For more information, see the section called “Creating response headers policies”.

Available managed response headers policies

The following topics describe the managed response headers policies that you can use.

Topics

- CORS-and-SecurityHeadersPolicy
- CORS-With-Preflight
- CORS-with-preflight-and-SecurityHeadersPolicy
- SecurityHeadersPolicy
- SimpleCORS

CORS-and-SecurityHeadersPolicy

View this policy in the CloudFront console

Use this managed policy to allow simple CORS requests from any origin. This policy also adds a set of security headers to all responses that CloudFront sends to viewers. This policy combines the the section called “SimpleCORS” and the section called “SecurityHeadersPolicy” policies into one.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

```
e61eb60c-9c35-4d20-a928-2b84e02af89c
```

Policy settings

	Header name	Header value	Override origin?
CORS headers:	`Access-Control-Allow-Origin`	`*`	No
Security headers:	`Referrer-Policy`	`strict-origin-when-cross-origin`	No
	`Strict-Transport-Security`	`max-age=31536000`	No
	`X-Content-Type-Options`	`nosniff`	Yes
	`X-Frame-Options`	`SAMEORIGIN`	No
	`X-XSS-Protection`	`1; mode=block`	No

CORS-With-Preflight

View this policy in the CloudFront console

Use this managed policy to allow CORS requests from any origin, including preflight requests. For preflight requests (using the HTTP `OPTIONS` method), CloudFront adds all three of the following headers to the response. For simple CORS requests, CloudFront adds only the `Access-Control-Allow-Origin` header.

If the response that CloudFront receives from the origin includes any of these headers, CloudFront uses the received header (and its value) in its response to the viewer. CloudFront doesn't use the header in this policy.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`5cc3b908-e619-4b99-88e5-2cf7f45965bd`

Policy settings

	Header name	Header value	Override origin?
CORS headers:	`Access-Control-Allow-Methods`	`DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT`	No
	`Access-Control-Allow-Origin`	`*`	
	`Access-Control-Expose-Headers`	`*`	

CORS-with-preflight-and-SecurityHeadersPolicy

View this policy in the CloudFront console

Use this managed policy to allow CORS requests from any origin. This includes preflight requests. This policy also adds a set of security headers to all responses that CloudFront sends to viewers. This policy combines the the section called "CORS-With-Preflight" and the section called "SecurityHeadersPolicy" policies into one.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`eaab4381-ed33-4a86-88ca-d9558dc6cd63`

Policy settings

	Header name	Header value	Override origin?
CORS headers:	`Access-Control-Allow-Methods`	`DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT`	No
	`Access-Control-Allow-Origin`	`*`	

	Header name	Header value	Override origin?
	`Access-Control-Expose-Headers`	`*`	
Security headers:	`Referrer-Policy`	`strict-origin-when-cross-origin`	No
	`Strict-Transport-Security`	`max-age=31536000`	No
	`X-Content-Type-Options`	`nosniff`	Yes
	`X-Frame-Options`	`SAMEORIGIN`	No
	`X-XSS-Protection`	`1; mode=block`	No

SecurityHeadersPolicy

View this policy in the CloudFront console

Use this managed policy to add a set of security headers to all responses that CloudFront sends to viewers. For more information about these security headers, see Mozilla's web security guidelines.

With this response headers policy, CloudFront adds `X-Content-Type-Options: nosniff` to all responses. This is the case when the response that CloudFront received from the origin included this header and when it didn't. For all other headers in this policy, if the response that CloudFront receives from the origin includes the header, CloudFront uses the received header (and its value) in its response to the viewer. It doesn't use the header in this policy.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`67f7725c-6f97-4210-82d7-5512b31e9d03`

Policy settings

	Header name	Header value	Override origin?
Security headers:	`Referrer-Policy`	`strict-origin-when-cross-origin`	No
	`Strict-Transport-Security`	`max-age=31536000`	No
	`X-Content-Type-Options`	`nosniff`	Yes
	`X-Frame-Options`	`SAMEORIGIN`	No
	`X-XSS-Protection`	`1; mode=block`	No

SimpleCORS

View this policy in the CloudFront console

Use this managed policy to allow simple CORS requests from any origin. With this policy, CloudFront adds the header `Access-Control-Allow-Origin: *` to all responses for simple CORS requests.

If the response that CloudFront receives from the origin includes the `Access-Control-Allow-Origin` header, CloudFront uses that header (and its value) in its response to the viewer. CloudFront doesn't use the header in this policy.

When using Amazon CloudFormation, the Amazon CLI, or the CloudFront API, the ID for this policy is:

`60669652-455b-4ae9-85a4-c4c02393f86c`

Policy settings

	Header name	Header value	Override origin?
CORS headers:	`Access-Control-Allow-Origin`	*	No

Understanding response headers policies

You can use a response headers policy to specify the HTTP headers that Amazon CloudFront removes or adds in responses that it sends to viewers. For more information about response headers policies and reasons to use them, see the section called “Adding or removing response headers”.

The following topics explain the settings in a response headers policy. The settings are grouped into categories, which are represented in the following topics.

Topics

- Policy details (metadata)
- CORS headers
- Security headers
- Custom headers
- Remove headers
- Server-Timing header

Policy details (metadata)

The policy details settings contain metadata about a response headers policy.

- **Name** – A name to identify the response headers policy. In the console, you use the name to attach the policy to a cache behavior.
- **Description** (optional) – A comment to describe the response headers policy. This is optional, but it can help you identify the purpose of the policy.

CORS headers

The cross-origin resource sharing (CORS) settings allow you to add and configure CORS headers in a response headers policy.

This list focuses on how to specify settings and valid values in a response headers policy. For more information about each of these headers and how they're used for real-world CORS requests and responses, see cross-origin resource sharing in the MDN Web Docs and the CORS protocol spec.

Access-Control-Allow-Credentials

This is a Boolean setting (`true` or `false`) that determines if CloudFront adds the `Access-Control-Allow-Credentials` header in responses to CORS requests. When this setting is set to `true`, CloudFront adds the `Access-Control-Allow-Credentials: true` header in responses to CORS requests. Otherwise, CloudFront doesn't add this header to responses.

Access-Control-Allow-Headers

Specifies the header names that CloudFront uses as values for the `Access-Control-Allow-Headers` header in responses to CORS preflight requests. Valid values for this setting include HTTP header names or the wildcard character (*), which indicates that all headers are allowed. Note that the `Authorization` header can't be wildcarded and always needs to be listed explicitly.

Examples of valid use of the wildcard character are shown in this table:

Example	Will match	Will not match
`x-amz-*`	`x-amz-test` `x-amz-`	`x-amz`
`x-*-amz`	`x-test-amz` `x--amz`	
*	All headers except `Authorization`	`Authorization`

Access-Control-Allow-Methods

Specifies the HTTP methods that CloudFront uses as values for the `Access-Control-Allow-Methods` header in responses to CORS preflight requests. Valid values are `GET`, `DELETE`, `HEAD`, `OPTIONS`, `PATCH`, `POST`, `PUT`, and `ALL`. `ALL` is a special value that includes all of the listed HTTP methods.

Access-Control-Allow-Origin

Specifies the values that CloudFront can use in the `Access-Control-Allow-Origin` response header. Valid values for this setting include a specific origin (such as `http://www.example.com`) or the wildcard character (*), which indicates that all origins are allowed. See the following table for examples:

> **Note**
>
> The wildcard character (*) is allowed as the leftmost part of the domain (`*.example.org`).
>
> The wildcard character (*) is *not* allowed in the following positions:
>
> - Top-level domains (`example.*`)
> - To the right of sub-domains (`test.*.example.org`)
> - Inside of terms (`exa*mple.org`)

Examples of valid use of the wildcard character are shown in this table:

Example	Will match	Will not match

`http://*.example.org`	`http://www.example.org`
	
	`http://test.example.org`
	
	`http://test.example.org:123`
		`https``://test.example.org`
		
		`https``://test.example.org:123`

`*.example.org`	`test.example.org`	

Example	Will match	Will not match
	test.test.example.org .example.org http://test.example.org https://test.example.org http://test.example.org:123 https://test.example.org:123	
example.org	http://example.org https://example.org	
http://example.org		**https**://example.org http://example.org:123
http://example.org:*	http://example.org:123 http://example.org	
http://example.org:1*3	http://example.org:123 http://example.org:1893 http://example.org:13	

Example	Will match	Will not match
`*.example.org:1*`	`test.example.org:123`	

Access-Control-Expose-Headers

Specifies the header names that CloudFront uses as values for the `Access-Control-Expose-Headers` header in responses to CORS requests. Valid values for this setting include HTTP header names or the wildcard character (*).

Access-Control-Max-Age

A number of seconds, which CloudFront uses as the value for the `Access-Control-Max-Age` header in responses to CORS preflight requests.

Origin override

This is a Boolean setting (`true` or `false`) that determines how CloudFront behaves when the response from the origin contains one of the CORS headers that's also in the policy.

When this setting is set to `true` and the origin response contains a CORS header that's also in the policy, CloudFront adds the CORS header in the policy to the response that it sends to the viewer. It ignores the header that it received from the origin.

When this setting is `false` and the origin response contains a CORS header that's also in the policy, CloudFront includes the CORS header it received from the origin in the response it sends to the viewer.

When the origin response doesn't contain a CORS header that's in the policy, CloudFront adds the CORS header in the policy to the response it sends to the viewer. CloudFront does this when this setting is set to `true` or `false`.

Security headers

You can use the security headers settings to add and configure several security-related HTTP response headers in a response headers policy.

This list describes how you can specify settings and valid values in a response headers policy. For more information about each of these headers and how they're used in real-world HTTP responses, see the links to the MDN Web Docs.

Content-Security-Policy

Specifies the content security policy directives that CloudFront uses as values for the `Content-Security-Policy` response header.

For more information about this header and valid policy directives, see Content-Security-Policy in the MDN Web Docs.

> **Note**
>
> The `Content-Security-Policy` header value is limited to 1783 characters.

Referrer-Policy

Specifies the referrer policy directive that CloudFront uses as the value for the `Referrer-Policy` response header. Valid values for this setting are `no-referrer`, `no-referrer-when-downgrade`, `origin`, `origin-when-cross-origin`, `same-origin`, `strict-origin`, `strict-origin-when-cross-origin`, and `unsafe-url`.

For more information about this header and these directives, see Referrer-Policy in the MDN Web Docs.

Strict-Transport-Security

Specifies the directives and settings that CloudFront uses as the value for the `Strict-Transport-Security` response header. For this setting, you separately specify:

- A number of seconds, which CloudFront uses as the value for the `max-age` directive of this header
- A Boolean setting (`true` or `false`) for `preload`, which determines whether CloudFront includes the `preload` directive in the value of this header
- A Boolean setting (`true` or `false`) for `includeSubDomains`, which determines whether CloudFront includes the `includeSubDomains` directive in the value of this header

For more information about this header and these directives, see Strict-Transport-Security in the MDN Web Docs.

X-Content-Type-Options

This is a Boolean setting (`true` or `false`) that determines if CloudFront adds the `X-Content-Type-Options` header to responses. When this setting is `true`, CloudFront adds the `X-`

`Content-Type-Options: nosniff` header to responses. Otherwise CloudFront doesn't add this header.

For more information about this header, see X-Content-Type-Options in the MDN Web Docs.

X-Frame-Options

Specifies the directive that CloudFront uses as the value for the `X-Frame-Options` response header. Valid values for this setting are `DENY` or `SAMEORIGIN`.

For more information about this header and these directives, see X-Frame-Options in the MDN Web Docs.

X-XSS-Protection

Specifies the directives and settings that CloudFront uses as the value for the `X-XSS-Protection` response header. For this setting, you separately specify:

- An `X-XSS-Protection` setting of `0` (disables XSS filtering) or `1` (enables XSS filtering)
- A Boolean setting (`true` or `false`) for `block`, which determines whether CloudFront includes the `mode=block` directive in the value for this header
- A reporting URI, which determines whether CloudFront includes the `report=`*`reporting URI`* directive in the value for this header

You can specify `true` for `block`, or you can specify a reporting URI, but you can't specify both together. For more information about this header and these directives, see X-XSS-Protection in the MDN Web Docs.

Origin override

Each of these security headers settings contains a Boolean setting (`true` or `false`) that determines how CloudFront behaves when the response from the origin contains that header.

When this setting is set to `true` and the origin response contains the header, CloudFront adds the header in the policy to the response that it sends to the viewer. It ignores the header that it received from the origin.

When this setting is set to `false` and the origin response contains the header, CloudFront includes the header that it received from the origin in the response that it sends to the viewer.

When the origin response doesn't contain the header, CloudFront adds the header in the policy to the response that it sends to the viewer. CloudFront does this when this setting is set to `true` or `false`.

Custom headers

You can use custom headers settings to add and configure custom HTTP headers in a response headers policy. CloudFront adds these headers to every response that it returns to viewers. For each custom header, you also specify the value for the header, though specifying a value is optional. This is because CloudFront can add a response header with no value.

Each custom header also has its own **Origin override** setting:

- When this setting is set to `true` and the origin response contains the custom header that's in the policy, CloudFront adds the custom header in the policy to the response that it sends to the viewer. It ignores the header that it received from the origin.
- When this setting is `false` and the origin response contains the custom header that's in the policy, CloudFront includes the custom header that it received from the origin in the response that it sends to the viewer.
- When the origin response doesn't contain the custom header that's in the policy, CloudFront adds the custom header in the policy to the response that it sends to the viewer. CloudFront does this when this setting is set to `true` or `false`.

Remove headers

You can specify headers that you want CloudFront to remove from the responses it receives from the origin so the headers are not included in the responses that CloudFront sends to viewers. CloudFront removes the headers from every response it sends to viewers, whether the objects is served from CloudFront's cache or from the origin. For example, you can remove headers that are of no use to browsers, such as `X-Powered-By` or `Vary`, so that CloudFront removes these headers from the responses that it sends to viewers.

When you specify headers to remove using a response headers policy, CloudFront removes the headers first and then adds any headers that are specified in other sections of the response headers policy (CORS headers, security headers, custom headers, etc.). If you specify a header to remove but also add the same header in another section of the policy, CloudFront includes the header in the responses that it sends to viewers.

> **Note**
>
> You can use a response headers policy to remove the `Server` and `Date` headers that CloudFront received from the origin, so that these headers (as received from the origin) are

not included in the responses that CloudFront sends to viewers. However, if you do that, CloudFront adds its own version of these headers to responses that it sends to viewers. For the `Server` header that CloudFront adds, the header's value is `CloudFront`.

Headers that you can't remove

You cannot remove the following headers using a response headers policy. If you specify these headers in the **Remove headers** section of a response headers policy (`ResponseHeadersPolicyRemoveHeadersConfig` in the API), you receive an error.

- `Connection`
- `Content-Encoding`
- `Content-Length`
- `Expect`
- `Host`
- `Keep-Alive`
- `Proxy-Authenticate`
- `Proxy-Authorization`
- `Proxy-Connection`
- `Trailer`
- `Transfer-Encoding`
- `Upgrade`
- `Via`
- `Warning`
- `X-Accel-Buffering`
- `X-Accel-Charset`
- `X-Accel-Limit-Rate`
- `X-Accel-Redirect`
- `X-Amz-Cf-.*`
- `X-Amzn-Auth`

- `X-Amzn-Cf-Billing`
- `X-Amzn-Cf-Id`
- `X-Amzn-Cf-Xff`
- `X-Amzn-ErrorType`
- `X-Amzn-Fle-Profile`
- `X-Amzn-Header-Count`
- `X-Amzn-Header-Order`
- `X-Amzn-Lambda-Integration-Tag`
- `X-Amzn-RequestId`
- `X-Cache`
- `X-Edge-.*`
- `X-Forwarded-Proto`
- `X-Real-Ip`

Server-Timing header

Use the `Server-Timing` header setting to enable the `Server-Timing` header in HTTP responses sent from CloudFront. You can use this header to view metrics that can help you gain insights about the behavior and performance of CloudFront and your origin. For example, you can see which cache layer served a cache hit. Or, you can see the first byte latency from the origin if there's a cache miss. The metrics in the `Server-Timing` header can help you troubleshoot issues or test the efficiency of your CloudFront or origin configuration.

For more information about using the `Server-Timing` header with CloudFront, see the following topics.

To enable the `Server-Timing` header, create (or edit) a response headers policy.

Topics

- Sampling rate and Pragma request header
- Server-Timing header from the origin
- Server-Timing header metrics
- Server-Timing header examples

Sampling rate and Pragma request header

When you enable the `Server-Timing` header in a response headers policy, you also specify the *sampling rate*. The sampling rate is a number 0–100 (inclusive) that specifies the percentage of responses that you want CloudFront to add the `Server-Timing` header to. When you set the sampling rate to 100, CloudFront adds the `Server-Timing` header to the HTTP response for every request that matches the cache behavior that the response headers policy is attached to. When you set it to 50, CloudFront adds the header to 50% of the responses for requests that match the cache behavior. You can set the sampling rate to any number 0–100 with up to four decimal places.

When the sampling rate is set to a number lower than 100, you can't control which responses CloudFront adds the `Server-Timing` header to, only the percentage. However, you can add the `Pragma` header with a value set to `server-timing` in an HTTP request to receive the `Server-Timing` header in the response to that request. This works no matter what the sampling rate is set to. Even when the sampling rate is set to zero (0), CloudFront adds the `Server-Timing` header to the response if the request contains the `Pragma: server-timing` header.

Server-Timing header from the origin

When there is a cache miss and CloudFront forwards the request to the origin, the origin might include a `Server-Timing` header in its response to CloudFront. In this case, CloudFront adds its metrics to the `Server-Timing` header that it received from the origin. The response that CloudFront sends to the viewer contains a single `Server-Timing` header that includes the value that came from the origin and the metrics that CloudFront added. The header value from the origin might be at the end, or in between two sets of metrics that CloudFront adds to the header.

When there is a cache hit, the response that CloudFront sends to the viewer contains a single `Server-Timing` header that includes only the CloudFront metrics in the header value (the value from the origin is not included).

Server-Timing header metrics

When CloudFront adds the `Server-Timing` header to an HTTP response, the value of the header contains one or more metrics that can help you gain insights about the behavior and performance of CloudFront and your origin. The following list contains all the metrics and their potential values. A `Server-Timing` header contains only some of these metrics, depending on the nature of the request and response through CloudFront.

Some of these metrics are included in the `Server-Timing` header with a name only (no value). Others are a name and a value. When a metric has a value, the name and value are separated by a semicolon (`;`). When the header contains more than one metric, the metrics are separated by a comma (`,`).

cdn-cache-hit

CloudFront provided a response from the cache without making a request to the origin.

cdn-cache-refresh

CloudFront provided a response from the cache after sending a request to the origin to verify that the cached object is still valid. In this case, CloudFront didn't retrieve the full object from the origin.

cdn-cache-miss

CloudFront didn't provide the response from the cache. In this case, CloudFront requested the full object from the origin before returning the response.

cdn-pop

Contains a value that describes which CloudFront point of presence (POP) handled the request.

cdn-rid

Contains a value with the CloudFront unique identifier for the request. You can use this request identifier (RID) when troubleshooting issues with Amazon Web Services Support

cdn-hit-layer

This metric is present when CloudFront provides a response from the cache without making a request to the origin. It contains one of the following values:

- **EDGE** – CloudFront provided the cached response from a POP location.
- **REC** – CloudFront provided the cached response from a regional edge cache (REC) location.
- **Origin Shield** – CloudFront provided the cached response from the REC that's acting as Origin Shield.

cdn-upstream-layer

When CloudFront requests the full object from the origin, this metric is present and contains one of the following values:

- **EDGE** – A POP location sent the request directly to the origin.

- **REC** – A REC location sent the request directly to the origin.
- **Origin Shield** – The REC that's acting as Origin Shield sent the request directly to the origin.

cdn-upstream-dns

Contains a value with the number of milliseconds that were spent retrieving the DNS record for the origin. A value of zero (0) indicates that CloudFront used a cached DNS result or reused an existing connection.

cdn-upstream-connect

Contains a value with the number of milliseconds between when the origin DNS request completed and a TCP (and TLS, if applicable) connection to the origin completed. A value of zero (0) indicates that CloudFront reused an existing connection.

cdn-upstream-fbl

Contains a value with the number of milliseconds between when the origin HTTP request is completed and when the first byte is received in the response from the origin (first byte latency).

cdn-downstream-fbl

Contains a value with the number of milliseconds between when the edge location finished receiving the request and when it sent the first byte of the response to the viewer.

Server-Timing header examples

The following are examples of a `Server-Timing` header that a viewer might receive from CloudFront when the `Server-Timing` header setting is enabled.

Example – cache miss

The following example shows a `Server-Timing` header that a viewer might receive when the requested object is not in the CloudFront cache.

```
Server-Timing: cdn-upstream-layer;desc="EDGE",cdn-upstream-dns;dur=0,cdn-upstream-
connect;dur=114,cdn-upstream-fbl;dur=177,cdn-cache-miss,cdn-pop;desc="PHX50-C2",cdn-
rid;desc="yNPsyYn7skvTzwWkq3Wcc8Nj_foxUjQUe9H1ifslzWhb0w7aLbFvGg==",cdn-downstream-
fbl;dur=436
```

This `Server-Timing` header indicates the following:

- The origin request was sent from a CloudFront point of presence (POP) location (`cdn-upstream-layer;desc="EDGE"`).
- CloudFront used a cached DNS result for the origin (`cdn-upstream-dns;dur=0`).
- It took 114 milliseconds for CloudFront to complete the TCP (and TLS, if applicable) connection to the origin (`cdn-upstream-connect;dur=114`).
- It took 177 milliseconds for CloudFront to receive the first byte of the response from the origin, after completing the request (`cdn-upstream-fbl;dur=177`).
- The requested object wasn't in CloudFront's cache (`cdn-cache-miss`).
- The request was received at the edge location identified by the code `PHX50-C2` (`cdn-pop;desc="PHX50-C2"`).
- The CloudFront unique ID for this request was `yNPsyYn7skvTzwWkq3Wcc8Nj_foxUjQUe9H1ifslzWhb0w7aLbFvGg==` (`cdn-rid;desc="yNPsyYn7skvTzwWkq3Wcc8Nj_foxUjQUe9H1ifslzWhb0w7aLbFvGg=="`).
- It took 436 milliseconds for CloudFront to send the first byte of the response to the viewer, after receiving the viewer request (`cdn-downstream-fbl;dur=436`).

Example – cache hit

The following example shows a `Server-Timing` header that a viewer might receive when the requested object is in CloudFront's cache.

```
Server-Timing: cdn-cache-hit,cdn-pop;desc="SEA19-C1",cdn-
rid;desc="nQBz4aJU2kP9iC3KHEq7vFxfMozu-VYBwGzkW9diOpeVc7xsrLKj-g==",cdn-hit-
layer;desc="REC",cdn-downstream-fbl;dur=137
```

This `Server-Timing` header indicates the following:

- The requested object was in the cache (`cdn-cache-hit`).
- The request was received at the edge location identified by the code `SEA19-C1` (`cdn-pop;desc="SEA19-C1"`).
- The CloudFront unique ID for this request was `nQBz4aJU2kP9iC3KHEq7vFxfMozu-VYBwGzkW9diOpeVc7xsrLKj-g==` (`cdn-rid;desc="nQBz4aJU2kP9iC3KHEq7vFxfMozu-VYBwGzkW9diOpeVc7xsrLKj-g=="`).
- The requested object was cached in a regional edge cache (REC) location (`cdn-hit-layer;desc="REC"`).

- It took 137 milliseconds for CloudFront to send the first byte of the response to the viewer, after receiving the viewer request (`cdn-downstream-fbl;dur=137`).

Adding, removing, or replacing content that CloudFront distributes

This section explains how to make sure CloudFront can access the content that you want to be served to your viewers, how to specify the objects in your website or in your application, and how to remove or replace content.

Topics

- Adding and accessing content that CloudFront distributes
- Updating existing content with a CloudFront distribution
- Removing content so CloudFront won't distribute it
- Customizing the URL format for files in CloudFront
- Specifying a default root object
- Invalidating files
- Serving compressed files
- Generating custom error responses

Adding and accessing content that CloudFront distributes

When you want CloudFront to distribute content (objects), you add files to one of the origins that you specified for the distribution, and you expose a CloudFront link to the files. A CloudFront edge location doesn't fetch the new files from an origin until the edge location receives viewer requests for them. For more information, see How CloudFront delivers content.

When you add a file that you want CloudFront to distribute, make sure that you add it to one of the Amazon S3 buckets specified in your distribution or, for a custom origin, to a directory in the specified domain. In addition, confirm that the path pattern in the applicable cache behavior sends requests to the correct origin.

For example, suppose the path pattern for a cache behavior is `*.html`. If you don't have any other cache behaviors configured to forward requests to that origin, CloudFront will only forward `*.html` files. In this scenario, for example, CloudFront will never distribute .jpg files that you upload to the origin, because you haven't created a cache behavior that includes .jpg files.

CloudFront servers don't determine the MIME type for the objects that they serve. When you upload a file to your origin, we recommend that you set the `Content-Type` header field for it.

Updating existing content with a CloudFront distribution

There are two ways to update existing content that CloudFront is set up to distribute for you:

- Update files by using the same name
- Update by using a version identifier in the file name

We recommend that you use a version identifier in file names or in folder names, to help give you more control over managing the content that CloudFront serves.

Updating existing files using versioned file names

When you update existing files in a CloudFront distribution, we recommend that you include some sort of version identifier either in your file names or in your directory names to give yourself better control over your content. This identifier might be a date-time stamp, a sequential number, or some other method of distinguishing two versions of the same object.

For example, instead of naming a graphic file image.jpg, you might call it image_1.jpg. When you want to start serving a new version of the file, you'd name the new file image_2.jpg, and you'd update the links in your web application or website to point to image_2.jpg. Alternatively, you might put all graphics in an images_v1 directory and, when you want to start serving new versions of one or more graphics, you'd create a new images_v2 directory, and you'd update your links to point to that directory. With versioning, you don't have to wait for an object to expire before CloudFront begins to serve a new version of it, and you don't have to pay for object invalidation.

Even if you version your files, we still recommend that you set an expiration date. For more information, see Managing how long content stays in the cache (expiration).

> **Note**
>
> Specifying versioned file names or directory names is not related to Amazon S3 object versioning.

Updating existing content using the same file names

Although you can update existing files in a CloudFront distribution and use the same file names, we don't recommend it. CloudFront distributes files to edge locations only when the files are requested, not when you put new or updated files in your origin. If you update an existing file in your origin with a newer version that has the same name, an edge location won't get that new version from your origin until both of the following occur:

- The old version of the file in the cache expires. For more information, see Managing how long content stays in the cache (expiration).
- There's a user request for the file at that edge location.

If you use the same names when you replace files, you can't control when CloudFront starts to serve the new files. By default, CloudFront caches files in edge locations for 24 hours. (For more information, see Managing how long content stays in the cache (expiration).) For example, if you're replacing all of the files on an entire website:

- Files for the less popular pages may not be in any edge locations. The new versions of these files will start being served on the next request.
- Files for some pages may be in some edge locations and not in others, so your end users will see different versions depending on which edge location they're served from.
- New versions of the files for the most popular pages might not be served for up to 24 hours because CloudFront might have retrieved the files for those pages just before you replaced the files with new versions.

Removing content so CloudFront won’t distribute it

You can remove files from your origin that you no longer want to be included in your CloudFront distribution. However, CloudFront will continue to show viewers content from the edge cache until the files expire.

If you want to remove a file right away, you must do one of the following:

- **Invalidate the file.** For more information, see Invalidating files.
- **Use file versioning.** When you use versioning, different versions of a file have different names that you can use in your CloudFront distribution, to change which file is returned to viewers. For more information, see Updating existing files using versioned file names.

Customizing the URL format for files in CloudFront

After you set up your origin with the objects (content) that you want CloudFront to serve to your viewers, you must use the correct URLs to reference those objects in your website or application code so that CloudFront can serve it.

The domain name that you use in the URLs for objects on your web pages or in your web application can be either of the following:

- The domain name, such as `d111111abcdef8.cloudfront.net`, that CloudFront automatically assigns when you create a distribution
- Your own domain name, such as `example.com`

For example, you might use one of the following URLs to return the file `image.jpg`:

`https://d111111abcdef8.cloudfront.net/images/image.jpg`

`https://example.com/images/image.jpg`

You use the same URL format whether you store the content in Amazon S3 buckets or at a custom origin, like one of your own web servers.

> **Note**
>
> The URL format depends in part on the value that you specify for **Origin Path** in your distribution. This value gives CloudFront a top directory path for your objects. For more information about setting the origin path when you create a distribution, see Origin path.

For more information about URL formats, see the following sections.

Using your own domain name (example.com)

Instead of using the default domain name that CloudFront assigns for you when you create a distribution, you can add an alternate domain name that's easier to work with, like `example.com`. By setting up your own domain name with CloudFront, you can use a URL like this for objects in your distribution:

`https://example.com/images/image.jpg`

If you plan to use HTTPS between viewers and CloudFront, see Using alternate domain names and HTTPS.

Using a trailing slash (/) in URLs

When you specify URLs for directories in your CloudFront distribution, choose either to always use a trailing slash or to never use a trailing slash. For example, choose only one of the following formats for all of your URLs:

```
https://d111111abcdef8.cloudfront.net/images/
```

```
https://d111111abcdef8.cloudfront.net/images
```

Why does it matter?

Both formats work to link to CloudFront objects, but being consistent can help prevent issues when you want to invalidate a directory later. CloudFront stores URLs exactly as they are defined, including trailing slashes. So if your format is inconsistent, you'll need to invalidate directory URLs with and without the slash, to ensure that CloudFront removes the directory.

It's inconvenient to have to invalidate both URL formats, and it can lead to additional costs. That's because if you must double up invalidations to cover both types of URLs, you might exceed the maximum number of free invalidations allowed for the month. And if that happens, you'll have to pay for all the invalidations, even if only one format for each directory URL exists in CloudFront.

Creating signed URLs for restricted content

If you have content that you want to restrict access to, you can create signed URLs. For example, if you want to distribute your content only to users who have authenticated, you can create URLs that are valid only for a specified time period or that are available only from a specified IP address. For more information, see Serving private content with signed URLs and signed cookies.

Specifying a default root object

You can configure CloudFront to return a specific object (the default root object) when a user requests the root URL for your distribution instead of requesting an object in your distribution. Specifying a default root object lets you avoid exposing the contents of your distribution.

Topics

- How to specify a default root object

- How default root object works
- How CloudFront works if you don't define a root object

How to specify a default root object

To avoid exposing the contents of your distribution or returning an error, specify a default root object for your distribution by completing the following steps.

To specify a default root object for your distribution

1. Upload the default root object to the origin that your distribution points to.

   The file can be any type supported by CloudFront. For a list of constraints on the file name, see the description of the `DefaultRootObject` element in DistributionConfig.

   > **Note**
   >
   > If the file name of the default root object is too long or contains an invalid character, CloudFront returns the error `HTTP 400 Bad Request - InvalidDefaultRootObject`. In addition, CloudFront caches the code for 10 seconds (by default) and writes the results to the access logs.

2. Confirm that the permissions for the object grant CloudFront at least `read` access.

   For more information about Amazon S3 permissions, see Identity and access management in Amazon S3 in the *Amazon Simple Storage Service User Guide*.

3. Update your distribution to refer to the default root object using the CloudFront console or the CloudFront API.

   To specify a default root object using the CloudFront console:

   a. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.

   b. In the list of distributions in the top pane, select the distribution to update.

   c. In the **Settings** pane, on the **General** tab, choose **Edit**.

   d. In the **Edit settings** dialog box, in the **Default root object** field, enter the file name of the default root object.

Enter only the object name, for example, `index.html`. Do not add a `/` before the object name.

e. Choose **Save changes**.

To update your configuration using the CloudFront API, you specify a value for the `DefaultRootObject` element in your distribution. For information about using the CloudFront API to specify a default root object, see UpdateDistribution in the *Amazon CloudFront API Reference*.

4. Confirm that you have enabled the default root object by requesting your root URL. If your browser doesn't display the default root object, perform the following steps:

   a. Confirm that your distribution is fully deployed by viewing the status of your distribution in the CloudFront console.

   b. Repeat steps 2 and 3 to verify that you granted the correct permissions and that you correctly updated the configuration of your distribution to specify the default root object.

How default root object works

Suppose the following request points to the object `image.jpg`:

```
https://d111111abcdef8.cloudfront.net/image.jpg
```

In contrast, the following request points to the root URL of the same distribution instead of to a specific object, as in the first example:

```
https://d111111abcdef8.cloudfront.net/
```

When you define a default root object, an end-user request that calls the root of your distribution returns the default root object. For example, if you designate the file `index.html` as your default root object, a request for:

```
https://d111111abcdef8.cloudfront.net/
```

Returns:

```
https://d111111abcdef8.cloudfront.net/index.html
```

Note

CloudFront does not determine whether a URL with multiple trailing slashes (`https://d111111abcdef8.cloudfront.net///`) is equivalent to `https://d111111abcdef8.cloudfront.net/`. Your origin server makes that comparison.

If you define a default root object, an end-user request for a subdirectory of your distribution does not return the default root object. For example, suppose `index.html` is your default root object and that CloudFront receives an end-user request for the `install` directory under your CloudFront distribution:

`https://d111111abcdef8.cloudfront.net/install/`

CloudFront does not return the default root object even if a copy of `index.html` appears in the `install` directory.

If you configure your distribution to allow all of the HTTP methods that CloudFront supports, the default root object applies to all methods. For example, if your default root object is index.php and you write your application to submit a POST request to the root of your domain (https://example.com), CloudFront sends the request to https://example.com/index.php.

The behavior of CloudFront default root objects is different from the behavior of Amazon S3 index documents. When you configure an Amazon S3 bucket as a website and specify the index document, Amazon S3 returns the index document even if a user requests a subdirectory in the bucket. (A copy of the index document must appear in every subdirectory.) For more information about configuring Amazon S3 buckets as websites and about index documents, see the Hosting Websites on Amazon S3 chapter in the *Amazon Simple Storage Service User Guide*.

Important

Remember that a default root object applies only to your CloudFront distribution. You still need to manage security for your origin. For example, if you are using an Amazon S3 origin, you still need to set your Amazon S3 bucket ACLs appropriately to ensure the level of access you want on your bucket.

How CloudFront works if you don't define a root object

If you don't define a default root object, requests for the root of your distribution pass to your origin server. If you are using an Amazon S3 origin, any of the following might be returned:

- **A list of the contents of your Amazon S3 bucket** – Under any of the following conditions, the contents of your origin are visible to anyone who uses CloudFront to access your distribution:
  - Your bucket is not properly configured.
  - The Amazon S3 permissions on the bucket associated with your distribution and on the objects in the bucket grant access to *everyone*.
  - An end user accesses your origin using your origin root URL.
- **A list of the private contents of your origin** – If you configure your origin as a private distribution (only you and CloudFront have access), the contents of the Amazon S3 bucket associated with your distribution are visible to anyone who has the credentials to access your distribution through CloudFront. In this case, users are not able to access your content through your origin root URL. For more information about distributing private content, see the section called "Restricting content with signed URLs and signed cookies".
- `Error 403 Forbidden`—CloudFront returns this error if the permissions on the Amazon S3 bucket associated with your distribution or the permissions on the objects in that bucket deny access to CloudFront and to everyone.

Invalidating files

If you need to remove a file from CloudFront edge caches before it expires, you can do one of the following:

- Invalidate the file from edge caches. The next time a viewer requests the file, CloudFront returns to the origin to fetch the latest version of the file.
- Use file versioning to serve a different version of the file that has a different name. For more information, see Updating existing files using versioned file names.

To invalidate files, you can specify either the path for individual files or a path that ends with the * wildcard, which might apply to one file or to many, as shown in the following examples:

- `/images/image1.jpg`

- `/images/image*`
- `/images/*`

> **Note**
>
> If you use the Amazon Command Line Interface (Amazon CLI) for invalidating files and you specify a path that includes the * wildcard, you must use quotes (") around the path. For example: `aws cloudfront create-invalidation --distribution-id` *`distribution_ID`* `--paths "/*"`

You can submit a certain number of invalidation paths each month for free. If you submit more than the allotted number of invalidation paths in a month, you pay a fee for each invalidation path that you submit. For more information about the charges for invalidation, see Paying for file invalidation.

Topics

- Choosing between invalidating files and using versioned file names
- Determining which files to invalidate
- Specifying the files to invalidate
- Invalidating files using the console
- Invalidating files using the CloudFront API
- Concurrent invalidation request maximum
- Paying for file invalidation

Choosing between invalidating files and using versioned file names

To control the versions of files that are served from your distribution, you can either invalidate files or give them versioned file names. If you want to update your files frequently, we recommend that you primarily use file versioning for the following reasons:

- Versioning enables you to control which file a request returns even when the user has a version cached either locally or behind a corporate caching proxy. If you invalidate the file, the user might continue to see the old version until it expires from those caches.

- CloudFront access logs include the names of your files, so versioning makes it easier to analyze the results of file changes.
- Versioning provides a way to serve different versions of files to different users.
- Versioning simplifies rolling forward and back between file revisions.
- Versioning is less expensive. You still have to pay for CloudFront to transfer new versions of your files to edge locations, but you don't have to pay for invalidating files.

For more information about file versioning, see Updating existing files using versioned file names.

Determining which files to invalidate

If you want to invalidate multiple files such as all of the files in a directory or all files that begin with the same characters, you can include the * wildcard at the end of the invalidation path. For more information about using the * wildcard, see Invalidation paths.

If you want to invalidate selected files but your users don't necessarily access every file on your origin, you can determine which files viewers have requested from CloudFront and invalidate only those files. To determine which files viewers have requested, enable CloudFront access logging. For more information about access logs, see Configuring and using standard logs (access logs).

Specifying the files to invalidate

Note the following about specifying the files that you want to invalidate.

Case sensitivity

Invalidation paths are case sensitive, so `/images/image.jpg` and `/images/Image.jpg` specify two different files.

Changing the URI using a Lambda function

If your CloudFront distribution triggers a Lambda function on viewer request events, and if the function changes the URI of the requested file, we recommend that you invalidate both URIs to remove the file from CloudFront edge caches:

- The URI in the viewer request
- The URI after the function changed it

For example, suppose your Lambda function changes the URI for a file from this:

`https://d111111abcdef8.cloudfront.net/index.html`

to a URI that includes a language directory:

`https://d111111abcdef8.cloudfront.net/en/index.html`

To invalidate the file, you must specify the following paths:

- `/index.html`
- `/en/index.html`

For more information, see Invalidation paths.

Default root object

To invalidate the default root object (file), specify the path the same way that you specify the path for any other file. For more information, see How default root object works.

Forwarding cookies

If you configured CloudFront to forward cookies to your origin, CloudFront edge caches might contain several versions of the file. When you invalidate a file, CloudFront invalidates every cached version of the file regardless of its associated cookies. You can't selectively invalidate some versions and not others based on the associated cookies. For more information, see Caching content based on cookies.

Forwarding headers

If you configured CloudFront to forward a list of headers to your origin and to cache based on the values of the headers, CloudFront edge caches might contain several versions of the file. When you invalidate a file, CloudFront invalidates every cached version of the file regardless of the header values. You can't selectively invalidate some versions and not others based on header values. (If you configure CloudFront to forward all headers to your origin, CloudFront doesn't cache your files.) For more information, see Caching content based on request headers.

Forwarding query strings

If you configured CloudFront to forward query strings to your origin, you must include the query strings when invalidating files, as shown in the following examples:

- `/images/image.jpg?parameter1=a`
- `/images/image.jpg?parameter1=b`

If client requests include five different query strings for the same file, you can either invalidate the file five times, once for each query string, or you can use the * wildcard in the invalidation path, as shown in the following example:

```
/images/image.jpg*
```

For more information about using wildcards in the invalidation path, see Invalidation paths. For more information about query strings, see Caching content based on query string parameters. To determine which query strings are in use, you can enable CloudFront logging. For more information, see Configuring and using standard logs (access logs).

Maximum allowed

For information about the maximum number of invalidations allowed, see Concurrent invalidation request maximum.

Microsoft Smooth Streaming files

You cannot invalidate media files in the Microsoft Smooth Streaming format when you have enabled Smooth Streaming for the corresponding cache behavior.

Non-ASCII or unsafe characters in the path

If the path includes non-ASCII characters or unsafe characters as defined in RFC 1738, URL-encode those characters. Do not URL-encode any other characters in the path, or CloudFront will not invalidate the old version of the updated file.

Invalidation paths

The path is relative to the distribution. For example, to invalidate the file at `https://d111111abcdef8.cloudfront.net/images/image2.jpg`, you would specify the following:

```
/images/image2.jpg
```

> **Note**
>
> In the CloudFront console, you can omit the leading slash in the path, like this: `images/image2.jpg`. When you use the CloudFront API directly, invalidation paths must begin with a leading slash.

You can also invalidate multiple files simultaneously by using the * wildcard. The *, which replaces 0 or more characters, must be the last character in the invalidation path. Also, if you use the Amazon Command Line Interface (Amazon CLI) for invalidating files and you specify a path that includes the * wildcard, you must use quotes (") around the path (like this: `"/*"`).

The following are some examples:

- To invalidate all of the files in a directory:

  */directory-path/**

- To invalidate a directory, all of its subdirectories, and all of the files in the directory and subdirectories:

  */directory-path**

- To invalidate all files that have the same name but different file name extensions, such as logo.jpg, logo.png, and logo.gif:

  */directory-path/file-name.**

- To invalidate all of the files in a directory for which the file name starts with the same characters (such as all of the files for a video in HLS format), regardless of the file name extension:

  */directory-path/initial-characters-in-file-name**

- When you configure CloudFront to cache based on query string parameters and you want to invalidate every version of a file:

  */directory-path/file-name.file-name-extension**

- To invalidate all of the files in a distribution:

  /*

The maximum length of a path is 4,000 characters. You can't use a wildcard within the path; only at the end of the path.

For information about invalidating files if you use a Lambda function to change the URI, see Changing the URI Using a Lambda Function.

The charge to submit an invalidation path is the same regardless of the number of files you're invalidating: a single file (`/images/logo.jpg`) or all of the files that are associated with a distribution (`/*`). For more information, see Amazon CloudFront Pricing.

If the invalidation path is a directory and if you have not standardized on a method for specifying directories—with or without a trailing slash (/)—we recommend that you invalidate the directory both with and without a trailing slash, for example, `/images` and `/images/`.

Signed URLs

If you are using signed URLs, invalidate a file by including only the portion of the URL before the question mark (?).

Invalidating files using the console

You can use the CloudFront console to create and run an invalidation, display a list of the invalidations that you submitted previously, and display detailed information about an individual invalidation. You can also copy an existing invalidation, edit the list of file paths, and run the edited invalidation. You can't remove invalidations from the list.

- Invalidating files
- Copying, editing, and rerunning an existing invalidation
- Canceling invalidations
- Listing invalidations
- Displaying information about an invalidation

Invalidating files

To invalidate files using the CloudFront console, do the following.

To invalidate files

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Select the distribution for which you want to invalidate files.
3. Choose **Distribution Settings**.
4. Choose the **Invalidations** tab.
5. Choose **Create Invalidation**.
6. For the files that you want to invalidate, enter one invalidation path per line. For information about specifying invalidation paths, see Specifying the files to invalidate.

> **Important**
>
> Specify file paths carefully. You can’t cancel an invalidation request after you start it.

7. Choose **Invalidate**.

Copying, editing, and rerunning an existing invalidation

You can copy an invalidation that you created previously, update the list of invalidation paths, and run the updated invalidation. You cannot copy an existing invalidation, update the invalidation paths, and then save the updated invalidation without running it.

> **Important**
>
> If you copy an invalidation that is still in progress, update the list of invalidation paths, and then run the updated invalidation, CloudFront will not stop or delete the invalidation that you copied. If any invalidation paths appear in the original and in the copy, CloudFront will try to invalidate the files twice, and both invalidations will count against your maximum number of free invalidations for the month. If you've already reached the maximum number of free invalidations, you'll be charged for both invalidations of each file. For more information, see Concurrent invalidation request maximum.

To copy, edit, and rerun an existing invalidation

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Select the distribution that contains the invalidation that you want to copy.
3. Choose **Distribution Settings**.
4. Choose the **Invalidations** tab.
5. Choose the invalidation that you want to copy.

   If you aren't sure which invalidation you want to copy, you can choose an invalidation and choose **Details** to display detailed information about that invalidation.
6. Choose **Copy**.
7. Update the list of invalidation paths if applicable.
8. Choose **Invalidate**.

Canceling invalidations

When you submit an invalidation request to CloudFront, CloudFront forwards the request to all edge locations within a few seconds, and each edge location starts processing the invalidation immediately. As a result, you can't cancel an invalidation after you submit it.

Listing invalidations

You can display a list of the last 100 invalidations that you've created and run for a distribution by using the CloudFront console. If you want to get a list of more than 100 invalidations, use the `ListInvalidations` API action. For more information, see ListInvalidations in the *Amazon CloudFront API Reference*.

To list invalidations

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Select the distribution for which you want to display a list of invalidations.
3. Choose **Distribution Settings**.
4. Choose the **Invalidations** tab.

> **Note**
>
> You can't remove invalidations from the list.

Displaying information about an invalidation

You can display detailed information about an invalidation, including distribution ID, invalidation ID, the status of the invalidation, the date and time that the invalidation was created, and a complete list of the invalidation paths.

To display information about an invalidation

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Select the distribution that contains the invalidation that you want to display detailed information for.

3. Choose **Distribution Settings**.
4. Choose the **Invalidations** tab.
5. Choose the applicable invalidation.
6. Choose **Details**.

Invalidating files using the CloudFront API

For information about invalidating objects and about displaying information about invalidations using the CloudFront API, see the following topics in the *Amazon CloudFront API Reference*:

- Invalidating files: CreateInvalidation
- Getting a list of your invalidations: ListInvalidations
- Getting information about a specific invalidation: GetInvalidation

Concurrent invalidation request maximum

If you're invalidating files individually, you can have invalidation requests for up to 3,000 files per distribution in progress at one time. This can be one invalidation request for up to 3,000 files, up to 3,000 requests for one file each, or any other combination that doesn't exceed 3,000 files. For example, you can submit 30 invalidation requests that invalidate 100 files each. As long as all 30 invalidation requests are still in progress, you can't submit any more invalidation requests. If you exceed the maximum, CloudFront returns an error message.

If you're using the * wildcard, you can have requests for up to 15 invalidation paths in progress at one time. You can also have invalidation requests for up to 3,000 individual files per distribution in progress at the same time; the maximum on wildcard invalidation requests allowed is independent of the maximum on invalidating files individually.

Paying for file invalidation

The first 1,000 invalidation paths that you submit per month are free; you pay for each invalidation path over 1,000 in a month. An invalidation path can be for a single file (such as `/images/logo.jpg`) or for multiple files (such as `/images/*`). A path that includes the `*` wildcard counts as one path even if it causes CloudFront to invalidate thousands of files.

The maximum of 1,000 free invalidation paths per month applies to the total number of invalidation paths across all of the distributions that you create with one Amazon account. For

example, if you use the Amazon account `john@example.com` to create three distributions, and you submit 600 invalidation paths for each distribution in a given month (for a total of 1,800 invalidation paths), Amazon will charge you for 800 invalidation paths in that month.

Because you are charged per path in your invalidation request, even if you bundle multiple paths into a single request, each path is still counted individually for billing purposes. For specific information about invalidation pricing, see Amazon CloudFront Pricing. For more information about invalidation paths, see Invalidation paths.

Serving compressed files

You can use CloudFront to automatically compress certain types of objects (files) and serve the compressed objects when viewers (web browsers or other clients) support them. Viewers indicate their support for compressed objects with the `Accept-Encoding` HTTP header.

CloudFront can compress objects using the Gzip and Brotli compression formats. When the viewer supports both formats, and both are present in the cache server that's reached, then CloudFront prefers Brotli. If only one compression format is present in the cache server, CloudFront returns it.

> **Note**
>
> The Chrome and Firefox web browsers support Brotli compression only when the request is sent using HTTPS. These browsers do not support Brotli with HTTP requests.

When requested objects are compressed, downloads can be faster because the objects are smaller—in some cases, less than a quarter the size of the original. Especially for JavaScript and CSS files, faster downloads can result in faster rendering of webpages for your users. In addition, because the cost of CloudFront data transfer is based on the total amount of data served, serving compressed objects can be less expensive than serving them uncompressed.

Some custom origins can also compress objects. Your origin might be able to compress objects that CloudFront doesn't compress (see File types that CloudFront compresses). If your origin returns a compressed object to CloudFront, CloudFront detects that the object is compressed based on the presence of a `Content-Encoding` header and doesn't compress the object again.

Configuring CloudFront to compress objects

To configure CloudFront to compress objects, update the cache behavior that you want to serve the compressed objects by doing ***all*** of the following:

1. Make sure the **Compress objects automatically** setting is **Yes**. (In Amazon CloudFormation or the CloudFront API, set `Compress` to `true`.)
2. Use a cache policy to specify caching settings, and make sure the **Gzip** and **Brotli** settings are both enabled. (In Amazon CloudFormation or the CloudFront API, set `EnableAcceptEncodingGzip` and `EnableAcceptEncodingBrotli` to `true`.)
3. Make sure the TTL values in the cache policy are set to a value greater than zero. When you set the TTL values to zero, caching is disabled and CloudFront doesn't compress objects.

To update a cache behavior, you can use any of the following tools:

- The CloudFront console
- Amazon CloudFormation
- The Amazon SDKs and command line tools

How CloudFront compression works

When you configure CloudFront to compress objects (see the previous section), here's how it works:

1. A viewer requests an object. The viewer includes the `Accept-Encoding` HTTP header in the request, and the header value includes `gzip`, `br`, or both. This indicates that the viewer supports compressed objects. When the viewer supports both Gzip and Brotli, CloudFront prefers Brotli.

   > **Note**
   >
   > The Chrome and Firefox web browsers support Brotli compression only when the request is sent using HTTPS. These browsers do not support Brotli with HTTP requests.

2. At the edge location, CloudFront checks the cache for a compressed copy of the requested object.
3. If the compressed object is already in the cache, CloudFront sends it to the viewer and skips the remaining steps.

If the compressed object is not in the cache, CloudFront forwards the request to the origin.

> **Note**
>
> If an uncompressed copy of the object is already in the cache, CloudFront might send it to the viewer without forwarding the request to the origin. For example, this can happen when CloudFront previously skipped compression. When this happens, CloudFront caches the uncompressed object and continues to serve it until the object expires, is evicted, or is invalidated.

4. If the origin returns a compressed object, as indicated by the presence of a `Content-Encoding` header in the HTTP response, CloudFront sends the compressed object to the viewer, adds it to the cache, and skips the remaining step. CloudFront doesn't compress the object again.

   If the origin returns an uncompressed object to CloudFront (there's no `Content-Encoding` header in the HTTP response), CloudFront determines whether the object is compressible. For more information about how CloudFront determines whether an object is compressible, see the following section.

5. If the object is compressible, CloudFront compresses it, sends it to the viewer, and adds it to the cache. (In rare cases, CloudFront might skip compression and send the uncompressed object to the viewer.)

Notes about CloudFront compression

The following list provides more information about when CloudFront compresses objects.

Request uses HTTP 1.0

If a request to CloudFront uses HTTP 1.0, CloudFront removes the `Accept-Encoding` header and does not compress the object in the response.

`Accept-Encoding` request header

If the `Accept-Encoding` header is missing from the viewer request, or if it doesn't contain `gzip` or `br` as a value, CloudFront does not compress the object in the response. If the `Accept-Encoding` header includes additional values such as `deflate`, CloudFront removes them before forwarding the request to the origin.

When CloudFront is configured to compress objects, it includes the `Accept-Encoding` header in the cache key and in origin requests automatically.

Dynamic content

CloudFront does not always compress dynamic content. Sometimes responses for dynamic content are compressed, and sometimes they are not.

Content is already cached when you configure CloudFront to compress objects

CloudFront compresses objects when it gets them from the origin. When you configure CloudFront to compress objects, CloudFront doesn't compress objects that are already cached in edge locations. In addition, when a cached object expires in an edge location and CloudFront forwards another request for the object to your origin, CloudFront doesn't compress the object when your origin returns an HTTP status code 304, which means that the edge location already has the latest version of the object. If you want CloudFront to compress objects that are already cached in edge locations, you need to invalidate those objects. For more information, see Invalidating files.

Origin is already configured to compress objects

If you configure CloudFront to compress objects and the origin also compresses objects, the origin should include a `Content-Encoding` header, which indicates to CloudFront that the object is already compressed. When a response from an origin includes the `Content-Encoding` header, CloudFront does not compress the object, regardless of the header's value. CloudFront sends the response to the viewer and caches the object in the edge location.

File types that CloudFront compresses

For a complete list of the file types that CloudFront compresses, see File types that CloudFront compresses.

Size of objects that CloudFront compresses

CloudFront compresses objects that are between 1,000 bytes and 10,000,000 bytes in size.

`Content-Length` header

The origin must include a `Content-Length` header in the response, which CloudFront uses to determine whether the size of the object is in the range that CloudFront compresses. If the `Content-Length` header is missing, contains an invalid value, or contains a value outside the range of sizes that CloudFront compresses, CloudFront does not compress the object.

HTTP status code of the response

CloudFront compresses objects only when the HTTP status code of the response is `200`, `403`, or `404`.

Response has no body

When the HTTP response from the origin has no body, there is nothing for CloudFront to compress.

ETag header

CloudFront sometimes modifies the `ETag` header in the HTTP response when it compresses objects. For more information, see the section called “ETag header conversion”.

CloudFront skips compression

CloudFront compresses objects on a best-effort basis. In rare cases, CloudFront skips compression. CloudFront makes this decision based on a variety of factors, including host capacity. If CloudFront skips compression for an object, it caches the uncompressed object and continues to serve it to viewers until the object expires, is evicted, or is invalidated.

File types that CloudFront compresses

If you configure CloudFront to compress objects, CloudFront only compresses objects that have one of the following values in the `Content-Type` response header:

- `application/dash+xml`
- `application/eot`
- `application/font`
- `application/font-sfnt`
- `application/javascript`
- `application/json`
- `application/opentype`
- `application/otf`
- `application/pdf`
- `application/pkcs7-mime`
- `application/protobuf`

- application/rss+xml
- application/truetype
- application/ttf
- application/vnd.apple.mpegurl
- application/vnd.mapbox-vector-tile
- application/vnd.ms-fontobject
- application/wasm
- application/xhtml+xml
- application/xml
- application/x-font-opentype
- application/x-font-truetype
- application/x-font-ttf
- application/x-httpd-cgi
- application/x-javascript
- application/x-mpegurl
- application/x-opentype
- application/x-otf
- application/x-perl
- application/x-ttf
- font/eot
- font/opentype
- font/otf
- font/ttf
- image/svg+xml
- text/css
- text/csv
- text/html
- text/javascript
- text/js

- text/plain
- text/richtext
- text/tab-separated-values
- text/xml
- text/x-component
- text/x-java-source
- text/x-script
- vnd.apple.mpegurl

ETag header conversion

When the uncompressed object from the origin includes a valid, strong ETag HTTP header, and CloudFront compresses the object, CloudFront also converts the strong ETag header value to a weak ETag, and returns the weak ETag value to the viewer. Viewers can store the weak ETag value and use it to send conditional requests with the If-None-Match HTTP header. This allows viewers, CloudFront, and the origin to treat the compressed and uncompressed versions of an object as semantically equivalent, which reduces unnecessary data transfer.

A valid, strong ETag header value begins with a double quote character ("). To convert the strong ETag value to a weak one, CloudFront adds the characters W/ to the beginning of the strong ETag value.

When the object from the origin includes a weak ETag header value (a value that begins with the characters W/), CloudFront does not modify this value, and returns it to the viewer as received from the origin.

When the object from the origin includes an invalid ETag header value (the value does not begin with " or with W/), CloudFront removes the ETag header and returns the object to the viewer without the ETag response header.

For more information, see the following pages in the MDN web docs:

- Directives (ETag HTTP header)
- Weak validation (HTTP conditional requests)
- If-None-Match HTTP header

Generating custom error responses

If an object that you're serving through CloudFront is unavailable for some reason, your web server typically returns a relevant HTTP status code to CloudFront to indicate this. For example, if a viewer requests an invalid URL, your web server returns an HTTP 404 (Not Found) status code to CloudFront, and CloudFront returns that status code to the viewer.

You can configure CloudFront to return a custom error response to the viewer instead, if you like. You also have several options for managing how CloudFront responds when there's an error. To specify options for custom error messages, you update your CloudFront distribution to specify those values. For more information, see Configuring error response behavior.

If you configure CloudFront to return a custom error page for an HTTP status code but the custom error page isn't available, CloudFront returns to the viewer the status code that CloudFront received from the origin that contains the custom error pages. For example, suppose your custom origin returns a 500 status code and you have configured CloudFront to get a custom error page for a 500 status code from an Amazon S3 bucket. However, someone accidentally deleted the custom error page from your bucket. CloudFront returns an HTTP 404 status code (Not Found) to the viewer that requested the object.

When CloudFront returns a custom error page to a viewer, you pay the standard CloudFront charges for the custom error page, not the charges for the requested object. For more information about CloudFront charges, see Amazon CloudFront Pricing.

Topics

- Configuring error response behavior
- Creating a custom error page for specific HTTP status codes
- Storing objects and custom error pages in different locations
- Changing response codes returned by CloudFront
- Controlling how long CloudFront caches errors

Configuring error response behavior

To configure custom error responses, you can use the CloudFront console, the CloudFront API, or Amazon CloudFormation. Regardless of how you choose to update the configuration, consider the following tips and recommendations:

- Save your custom error pages in a location that is accessible to CloudFront. We recommend that you store them in an Amazon S3 bucket, and that you don't store them in the same place as the rest of your website or application's content. If you store the custom error pages on the same origin as your website or application, and the origin starts to return 5xx errors, CloudFront can't get the custom error pages because the origin server is unavailable. For more information, see Storing objects and custom error pages in different locations.
- Make sure that CloudFront has permission to get your custom error pages. If the custom error pages are stored in Amazon S3, the pages must be publicly accessible or you must configure a CloudFront origin access control (OAC). If the custom error pages are stored in a custom origin, the pages must be publicly accessible.
- (Optional) Configure your origin to add a `Cache-Control` or `Expires` header along with the custom error pages, if you want. You can also use the **Error Caching Minimum TTL** setting to control how long CloudFront caches the custom error pages. For more information, see Controlling how long CloudFront caches errors.

Configure custom error responses (CloudFront console)

To configure custom error responses in the CloudFront console, you must have a CloudFront distribution. In the console, the configuration settings for custom error responses are only available for existing distributions. To learn how to create a distribution, see Getting started with a basic CloudFront distribution.

To configure custom error responses (console)

1. Sign in to the Amazon Web Services Management Console and open the **Distributions** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#distributions.
2. In the list of distributions, choose the distribution to update.
3. Choose the **Error Pages** tab, then choose **Create Custom Error Response**.
4. Enter the applicable values. For more information, see Custom error pages and error caching.
5. After entering the desired values, choose **Create**.

Configure custom error responses (CloudFront API or Amazon CloudFormation)

To configure custom error responses with the CloudFront API or Amazon CloudFormation, use the `CustomErrorResponse` type in a distribution. For more information, see the following:

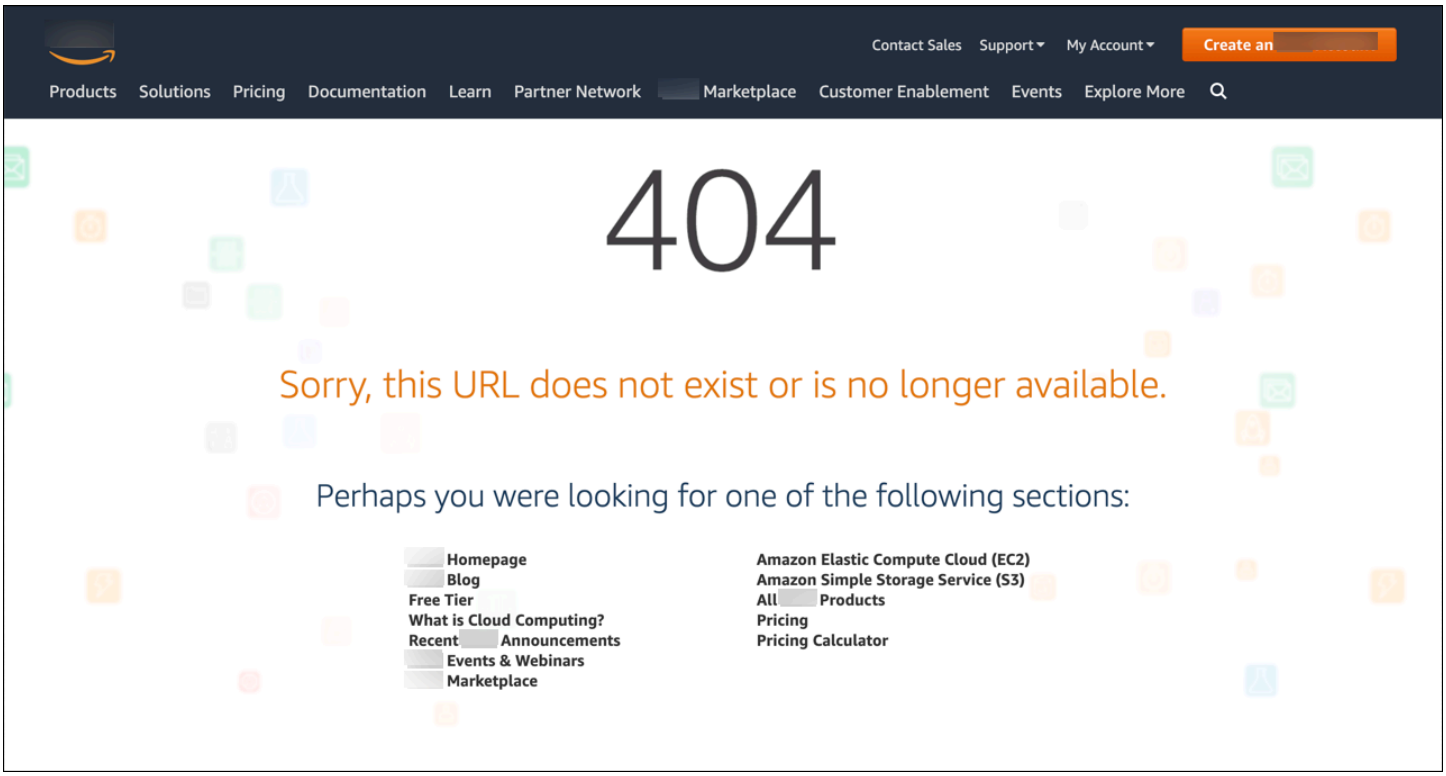
- AWS::CloudFront::Distribution CustomErrorResponse in the *Amazon CloudFormation User Guide*
- CustomErrorResponse in the *Amazon CloudFront API Reference*

Creating a custom error page for specific HTTP status codes

If you'd rather display a custom error message instead of the default message—for example, a page that uses the same formatting as the rest of your website—you can have CloudFront return to the viewer an object (such as an HTML file) that contains your custom error message.

To specify the file that you want to return and the errors for which the file should be returned, you update your CloudFront distribution to specify those values. For more information, see Configuring error response behavior.

For example, the following is a custom error page:



You can specify a different object for each supported HTTP status code, or you can use the same object for all of the supported status codes. You can choose to specify custom error pages for some status codes and not for others.

The objects that you're serving through CloudFront can be unavailable for a variety of reasons. These fall into two broad categories:

- *Client errors* indicate a problem with the request. For example, an object with the specified name isn't available, or the user doesn't have the permissions required to get an object in your Amazon S3 bucket. When a client error occurs, the origin returns an HTTP status code in the 4xx range to CloudFront.
- *Server errors* indicate a problem with the origin server. For example, the HTTP server is busy or unavailable. When a server error occurs, either your origin server returns an HTTP status code in the 5xx range to CloudFront, or CloudFront doesn't get a response from your origin server for a certain period of time and assumes a 504 status code (Gateway Timeout).

The HTTP status codes for which CloudFront can return a custom error page include the following:

- 400, 403, 404, 405, 414, 416

  Notes

  - If CloudFront detects that the request might be unsafe, CloudFront returns a 400 (Bad Request) error instead of a custom error page.
  - You can create a custom error page for HTTP status code 416 (Requested Range Not Satisfiable), and you can change the HTTP status code that CloudFront returns to viewers when your origin returns a status code 416 to CloudFront. (For more information, see Changing response codes returned by CloudFront.) However, CloudFront doesn't cache status code 416 responses, so even if you specify a value for **Error Caching Minimum TTL** for status code 416, CloudFront doesn't use it.

- 500, 501, 502, 503, 504

  Note

  In some cases, CloudFront doesn't return a custom error page for the HTTP 503 status code even if you configure CloudFront to do so. If the CloudFront error code is `Capacity Exceeded` or `Limit Exceeded`, CloudFront returns a 503 status code to the viewer without using your custom error page.

For a detailed explanation of how CloudFront handles error responses from your origin, see How CloudFront processes and caches HTTP 4xx and 5xx status codes from your origin.

Storing objects and custom error pages in different locations

If you want to store your objects and your custom error pages in different locations, your distribution must include a cache behavior for which the following is true:

- The value of **Path Pattern** matches the path to your custom error messages. For example, suppose you saved custom error pages for 4xx errors in an Amazon S3 bucket in a directory named `/4xx-errors`. Your distribution must include a cache behavior for which the path pattern routes requests for your custom error pages to that location, for example, `/4xx-errors/*`.
- The value of **Origin** specifies the value of **Origin ID** for the origin that contains your custom error pages.

For more information, see Cache behavior settings.

Changing response codes returned by CloudFront

You can configure CloudFront to return a different HTTP status code to the viewer than what CloudFront received from the origin. For example, if your origin returns a 500 status code to CloudFront, you might want CloudFront to return a custom error page and a 200 status code (OK) to the viewer. There are a variety of reasons that you might want CloudFront to return a status code to the viewer that is different from the one that your origin returned to CloudFront:

- Some internet devices (some firewalls and corporate proxies, for example) intercept HTTP 4xx and 5xx status codes and prevent the response from being returned to the viewer. In this scenario, if you substitute `200`, the response is not intercepted.
- If you don't care about distinguishing among different client errors or server errors, you can specify `400` or `500` as the value that CloudFront returns for all 4xx or 5xx status codes.
- You might want to return a `200` status code (OK) and a static website so your customers don't know that your website is down.

If you enable CloudFront standard logs and you configure CloudFront to change the HTTP status code in the response, the value of the `sc-status` column in the logs contains the status code that you specify. However, the value of the `x-edge-result-type` column is not affected. It contains the result type of the response from the origin. For example, suppose you configure CloudFront to return a status code of `200` to the viewer when the origin returns `404` (Not Found) to CloudFront.

When the origin responds to a request with a `404` status code, the value in the `sc-status` column in the log will be `200`, but the value in the `x-edge-result-type` column will be `Error`.

You can configure CloudFront to return any of the following HTTP status codes along with a custom error page:

- 200
- 400, 403, 404, 405, 414, 416
- 500, 501, 502, 503, 504

Controlling how long CloudFront caches errors

CloudFront caches error responses for a default duration of 10 seconds. CloudFront then submits the next request for the object to your origin to see if the problem that caused the error has been resolved and the requested object is available.

You can specify the error-caching duration—the **Error Caching Minimum TTL**—for each 4xx and 5xx status code that CloudFront caches. (For more information, see HTTP 4xx and 5xx status codes that CloudFront caches.) When you specify a duration, note the following:

- If you specify a short error-caching duration, CloudFront forwards more requests to your origin than if you specify a longer duration. For 5xx errors, this might aggravate the problem that originally caused your origin to return an error.
- When your origin returns an error for an object, CloudFront responds to requests for the object either with the error response or with your custom error page until the error-caching duration elapses. If you specify a long error-caching duration, CloudFront might continue to respond to requests with an error response or your custom error page for a long time after the object becomes available again.

> **Note**
>
> You can create a custom error page for HTTP status code 416 (Requested Range Not Satisfiable), and you can change the HTTP status code that CloudFront returns to viewers when your origin returns a status code 416 to CloudFront. (For more information, see Changing response codes returned by CloudFront.) However, CloudFront doesn't cache

status code 416 responses, so even if you specify a value for **Error Caching Minimum TTL** for status code 416, CloudFront doesn't use it.

If you want to control how long CloudFront caches errors for individual objects, you can configure your origin server to add the applicable header to the error response for that object.

If the origin adds a `Cache-Control: max-age` or `Cache-Control: s-maxage` directive, or an `Expires` header, CloudFront caches error responses for the greater of the value in the header or the **Error Caching Minimum TTL**.

Note

The `Cache-Control: max-age` and `Cache-Control: s-maxage` values cannot be greater than the **Maximum TTL** value set for the cache behavior for which the error page is being fetched.

If the origin adds other `Cache-Control` directives or adds no headers, CloudFront caches error responses for the value of **Error Caching Minimum TTL**.

If the expiration time for a 4xx or 5xx status code for an object is longer than you want, and the object is available again, you can invalidate cached error code by using the URL of the requested object. If your origin is returning an error response for multiple objects, you need to invalidate each object separately. For more information about invalidating objects, see Invalidating files.

Using Amazon WAF protections

You can use Amazon WAF to protect your CloudFront distributions and origin servers. Amazon WAF is a web application firewall that helps secure your web applications and APIs by blocking requests before they reach your servers. For more details, see Accelerate and protect your websites using CloudFront and Amazon WAF.

To enable Amazon WAF protections, you can:

- Use one-click protection in the CloudFront console. One-click protection creates an Amazon WAF web access control list (web ACL), configures rules to protect your servers from common web threats, and attaches the web ACL to the CloudFront distribution for you. The topics in this section assume the use of one-click protections.
- Use a preconfigured web ACL (access control list) that you create in the Amazon WAF console, or by using the Amazon WAF APIs. For more information, see Web access control lists (ACLs) in the *Amazon WAF Developer Guide* and AssociateWebACL in the *Amazon WAF API Reference*

You can enable Amazon WAF when you:

- Create a distribution
- Use the **Security** dashboard to edit the security settings of an existing distribution

When you use one-click protection, CloudFront applies an Amazon-recommended set of protections that:

- Block IP addresses from potential threats based on Amazon internal threat intelligence.
- Protect against the most common vulnerabilities found in web applications as described in the OWASP Top 10.
- Defend against malicious actors discovering application vulnerabilities.

Important

You must enable Amazon WAF if you want to view security metrics in the CloudFront **Security** dashboard. Without Amazon WAF, enabled, you can only use the **Security** dashboard to enable Amazon WAF or configure CloudFront geographic restrictions. For

more information about the dashboard, see Using CloudFront security dashboards, later in this section.

Topics

- Enabling Amazon WAF for new distributions
- Enabling Amazon WAF for existing distributions
- Disabling Amazon WAF security protections
- Setting up rate limiting
- Using CloudFront security dashboards

Enabling Amazon WAF for new distributions

The following steps explain how to enable Amazon WAF when you create a distribution, and how to use an existing ACL with the new distribution.

To enable Amazon WAF for new distributions

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Distributions**, then choose **Create distribution**.
3. As needed, follow the steps in Creating a distribution.
4. In the **Web Application Firewall** section, select **Edit**, then **Enable security protections**. More fields appear.
5. Complete the following fields:

   - **Use monitor mode**: You enable monitor mode when you want to first collect data to test how protection will work. When you enable monitor mode, requests aren't blocked if the protections were active. Instead, monitor mode collects data about requests that would be blocked if the protections were active. When you are ready to begin blocking, you can enable blocking on the **Security** page.
   - You might see an **Additional protections** section. Choose any options that you want to enable. If you enable rate limiting, see the section called “Setting up rate limiting” for more information.
   - A **Price estimate** section. You can open the section to display a field where you enter a different number of requests/month and see a new estimate.

6. Review the remaining distribution settings, then select **Create distribution** or **Save Settings** (if editing an existing distribution).

Using an existing web ACL

If you have a web ACL, you can use it instead of the protection offered by one-click WAF.

To use an existing Amazon WAF configuration

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Do one of the following:
   a. Choose **Create distribution** and follow the steps in Creating a distribution, then return to this topic.
   b. Choose an existing configuration, then choose the **Security** tab.
3. In the **Web Application Firewall (WAF)** section, select **Edit**, then **Enable security protections**.
4. Choose **Use existing WAF configuration**. This option appears only if you have web ACLs configured.
5. Choose your existing web ACL from the **Choose a web ACL** table.
6. Review the remaining distribution settings, then choose **Create distribution**, or **Save Settings** (if editing an existing distribution).

Enabling Amazon WAF for existing distributions

CloudFront creates a **Security** dashboard when you create a distribution. You use the dashboard to enable Amazon WAF after you create a distribution. The charts and graphs in the dashboard remain blank until you enable Amazon WAF.

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Distributions**, then choose the distribution that you want to change.
3. Choose the **Security** tab.
4. Under **Web Application Firewall**, select **Edit**, then **Enable security protections**.
5. (Optional) Choose **Use monitor mode**.
6. Choose **Save changes**.

For more information about monitor mode, see the previous section, Enabling Amazon WAF for new distributions.

Using an existing web ACL

If you web ACLs configured, you can use them instead of the protection offered by one-click WAF.

To use an existing Amazon WAF configuration

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Do one of the following:
   a. Choose **Create distribution** and follow the steps in Creating a distribution, then return to this topic.
   b. Choose the **Security** to configure an existing distribution.
3. In the **Web Application Firewall (WAF)** section, select **Enable security protections**.
4. Choose **Use existing WAF configuration**. This option appears only if you have web ACLs configured.
5. Choose your existing web ACL from the **Choose a web ACL** table.
6. Review the remaining distribution settings, then choose **Create distribution**, or **Save Settings** (if editing an existing distribution).

Disabling Amazon WAF security protections

If your distribution doesn't need Amazon WAF security protections, you can disable this feature by using the CloudFront console.

If you previously enabled Amazon WAF protection and didn't choose an existing WAF configuration (also known as one-click protection), CloudFront automatically created a web ACL for you. For web ACLs created this way, the CloudFront console will disassociate the resource and delete the web ACL.

Disassociating a web ACL is different from deleting it. Disassociating removes the web ACL from your distribution, but it's not deleted from your Amazon Web Services account. For more information, see Associating or disassociating a web ACL with an Amazon resource in the *Amazon WAF, Amazon Firewall Manager, and Amazon Shield Advanced Developer Guide*.

See the following procedure to disable Amazon WAF protections and disassociate the web ACL from your distribution.

To disable Amazon WAF security protections in CloudFront

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Distributions**, and then choose the distribution that you want to change.
3. Choose the **Security** tab and then choose **Edit**.
4. In the **Web Application Firewall (WAF)** section, choose **Disable Amazon WAF protection**.
5. Choose **Save changes**.

Notes

- If you disabled Amazon WAF security protection and you still want to delete the web ACL from your Amazon Web Services account, you can delete it manually. Follow the procedure to delete a web ACL. In the Amazon WAF & Shield console, for the **Web ACLs** page, you *must* choose the **Global (CloudFront)** list to find the web ACLs.
- When you delete a distribution from the CloudFront console, CloudFront will attempt to also delete the web ACL if you chose one-click protection. This is best effort and isn't always guaranteed. For more information, see Deleting a distribution.

Setting up rate limiting

Rate limiting is one of the recommendations you may receive when configuring security protections.

CloudFront always enables rate limiting in monitor mode. When monitor mode is enabled, CloudFront captures metrics that tell you if the rate you configured in the **Rate limiting** field has been exceeded, how often, and by how much.

After you save the distribution, CloudFront starts to collect data based on the number in the **Rate limiting** field.

You can manage the rate limiting settings in the **Security - Web Application Firewall (WAF)** section on the **Security** tab of any CloudFront distribution. Select the **Monitor mode** message to

display a dialog with details about the collected data. On that dialog, you can optionally change the rate limit. When you have fine-tuned the rate, you can choose **Enable blocking** (on the dialog) to deactivate monitor mode. CloudFront will start to block requests that exceed the specified rate limit.

Using CloudFront security dashboards

CloudFront creates a security dashboard for each of your distributions. You use the dashboards in the CloudFront console. With the dashboards, you can use CloudFront and Amazon WAF together in a single location to monitor and manage common security protections for your web applications. The dashboards provide the following tasks and data:

- **Security configuration**: You can enable and disable Amazon WAF protections, and see any app-specific protections such as WordPress protections.
- **Security trends**: These include allowed and blocked requests, challenge and CAPTCHA requests, and top attack types.
- **Bot requests**: You can see how much traffic comes from bots, which types of bots (verified vs non-verified), and how the percentage allocations of bot types (verified vs non-verified) change over time.
- **Request logs**: Log data can help answer questions about security trends or bot requests. You can search your logs without writing queries, and view aggregate charts to help determine if a filtered set of logs is primarily being driven by a subset of HTTP methods, IP addresses, URI paths, or countries. You can hover over values in the charts and block IP addresses and countries.
- **Geographic restriction management**

> **Note**
>
> You must enable Amazon WAF if you want to view security metrics in the CloudFront **Security** dashboard. Without Amazon WAF enabled, you can only use the **Security** dashboard to enable Amazon WAF or configure CloudFront geographic restrictions.

The following sections explain how to use the dashboards.

Topics

- Enabling Amazon WAF

- Understanding trend data
- Enabling bot control
- Understanding logs
- Managing CloudFront geographic restrictions
- Security dashboard pricing

Enabling Amazon WAF

You use the top section of the **Security** dashboard to enable or disable Amazon WAF protections. CloudFront also displays security recommendations specific to your distribution. depending on your configuration. For example, if you configure a cache behavior with a WordPress path pattern, you see WordPress protections and rate limiting.

> **Note**
>
> You must enable Amazon WAF if you want to view security metrics in the CloudFront **Security** dashboard. Without Amazon WAF enabled, you can only use the **Security** dashboard to enable Amazon WAF or configure CloudFront geographic restrictions.

To enable Amazon WAF

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Distributions**, then choose the distribution that you want to change.
3. Choose the **Security** tab.
4. In the **Web Application Firewall** section, select **Enable security protections**.
5. Select **Save changes**.

To disable Amazon WAF protections

- Repeat the steps listed above, but select **Disable security protections**.

Understanding trend data

The **Security trends for the specified time range** section of the dashboard displays summary metrics of your traffic for a given period of time. The dashboard displays data for Allowed, Blocked, Challenge, and CAPTCHA requests. You can see traffic ratios and how they change over time. For example, if all requests increase by 3% but allowed requests increase by 14%, that means you allowed a larger portion of your traffic through in the current period.

The section provides three bar charts, **Requests**, **Top attack types**, and **Top countries**. You can use the **Top countries** chart to block a country.

To use the charts

- Use the **Date range**, **Rule actions**, and **Granularity** controls above the chart to set a time range and filter your data.
- Hover over any bar to see the request, attack, or country data for the specified time period.
- To block a country, hover over that bar and move the **Block** *country name* slider to the on position.

> **Note**
>
> The **Block** option may not be available if you previously created a custom Amazon WAF rule outside of the CloudFront console to block countries.

Enabling bot control

The **Bot requests for a given time range** section displays bot request data. You can also enable or disable the Amazon WAF bot control. You incur charges when you enable bot control, and the dashboard provides a cost estimate.

If you enable bot control, the dashboard charts display the amount of traffic coming from each type and category of bot. If you disable bot controls, the charts display the amount of traffic based on request sampling.

The section provides two bar charts, **Requests by bot type** and **Requests by bot category**. The charts vary, depending on whether you have bot control enabled.

To enable bot control

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Distributions**, then choose the distribution that you want to change.
3. Choose the **Security** tab.
4. Scroll down to the **Bot requests for a given time range** section and choose **Enable Bot Control**.
5. In the **Bot Control** dialog box, under **Configuration**, select the **Enable Bot Control for common bots** check box.
6. Choose **Save changes**.

When you enable bot protection, you have the option of configuring how each unverified bot is handled per bot category. For example, you can set an HTTP library bot to **Monitor mode** and assign a **Challenge** to a Link Checker.

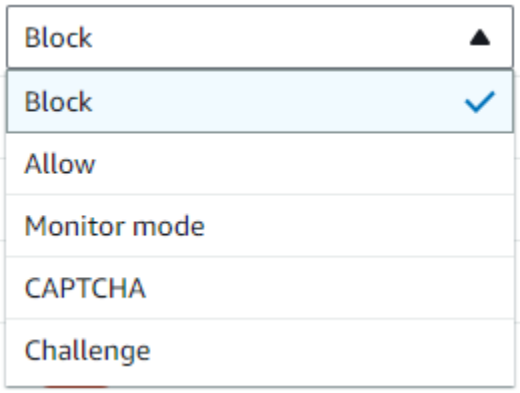
Bots that are known by Amazon to be common and verifiable, such as known search engine crawlers, aren't subject to the actions you set here. Bot control confirms that validated bots come from the source that they claim before marking them as verified.

To configure protection for a category

1. Repeat steps 1 and 2 in the previous steps to start the **Security** dashboard.
2. In the **Requests by bot category** chart, point to any of the items in the **Unverified bot action** column and choose the edit icon.

3. Open the resulting list and choose one of the following:

   - **Block**
   - **Allow**
   - **Monitor mode**
   - **CAPTCHA**
   - **Challenge**



4. Select the check mark next to the list to commit your change.

To use the charts

- In the **Security trends for the specified time range** section, use the **Date range**, **Rule actions**, and **Granularity** controls set a time range and filter your bot data.
- In the **Requests by bot type** chart, hover over any bar to see the number of requests by bot type.
- In the **Requests by bot category** chart, hover over any bar to see the number of requests by bot category.

Understanding logs

Log data can help you isolate specific traffic patterns. For example, logs can show you where certain traffic comes from or what it does.

To enable logs

1. Enter your expected request volume in the **Number of requests/month** box to estimate the costs of enabling logs.
2. Select the **Enable Amazon WAF logs** check box.
3. Choose **Enable**.

CloudFront creates a CloudWatch logs group and updates your Amazon WAF configuration to begin logging to CloudWatch. On first use, log data can take several minutes to appear. The **Requests**

section of the chart lists each request. Below the individual requests, the bar charts aggregate data by HTTP method, top URI paths, top IP addresses, and top countries. The charts can help you find patterns. For example, you may see a disproportionate volume of requests from a single IP address, or data from a country that you haven't previously seen in your logs. You can filter requests based on **Country**, **Host Header**, and other attributes to help find unwanted traffic. Once you identify that traffic, hover over an individual request or chart item and block an IP address or country.

To use the charts

- Use the **Date range**, **Rule actions**, and **Granularity** controls in the **Security trends for the specified time range** section to set a time range and filter your data.
- Hover over any bar to see the URI path, IP address, or country data for the specified time period.
- To block an IP address or country, hover over that bar and move the **Block** *item name* slider to the on position.

> **Note**
>
> The **Block** option may not be available if you previously created a custom Amazon WAF rule outside of the CloudFront console to block countries or IP addresses.

> **Note**
>
> Displayed metrics are based on web access control list (ACL). Therefore, if you associate the same web ACL to multiple distributions, you will see all metrics for your web ACL, not only the Amazon WAF requests that are processed for that distribution.

Managing CloudFront geographic restrictions

You can manage geographic restrictions at any time.

To manage geo restrictions

1. Scroll down to the **Geographic restrictions** section.
2. Choose **Edit**.

3. Select **Allow list** to add a country to your list of allowed countries, or **Block list** to add a country to your list of blocked countries.
4. Add the desired country or countries to the list, then choose **Save changes**.

CloudFront and Amazon WAF provide geographic restriction features. CloudFront provides geographic restrictions for free, but your dashboard will not display metrics for the blocked countries. In contrast, when you hover over a country bar in the **Security** dashboard and block a country, you use Amazon WAF geographic restrictions. They also block countries, but your dashboard displays the request metrics for blocked requests.

Security dashboard pricing

If you enable Amazon WAF logging to Amazon CloudWatch, the CloudFront **Security** dashboard queries, aggregates, and displays insights from the CloudWatch logs. We don't charge to use the **Security** dashboard, but Amazon CloudWatch pricing applies to logs queried through the dashboard. For more information, see Amazon CloudWatch Pricing.

Configuring secure access and restricting access to content

CloudFront provides several options for securing content that it delivers. The following are some ways you can use CloudFront to secure and restrict access to content:

- Configure HTTPS connections
- Prevent users in specific geographic locations from accessing content
- Require users to access content using CloudFront signed URLs or signed cookies
- Set up field-level encryption for specific content fields
- Use Amazon WAF to control access to your content

Topics

- Using HTTPS with CloudFront
- Using alternate domain names and HTTPS
- Serving private content with signed URLs and signed cookies
- Restricting access to an Amazon origin
- Restricting access to Application Load Balancers
- Restricting the geographic distribution of your content
- Using field-level encryption to help protect sensitive data

Using HTTPS with CloudFront

You can configure CloudFront to require that viewers use HTTPS so that connections are encrypted when CloudFront communicates with viewers. You also can configure CloudFront to use HTTPS with your origin so that connections are encrypted when CloudFront communicates with your origin.

If you configure CloudFront to require HTTPS both to communicate with viewers and to communicate with your origin, here's what happens when CloudFront receives a request:

1. A viewer submits an HTTPS request to CloudFront. There's some SSL/TLS negotiation here between the viewer and CloudFront. In the end, the viewer submits the request in an encrypted format.
2. If the CloudFront edge location contains a cached response, CloudFront encrypts the response and returns it to the viewer, and the viewer decrypts it.
3. If the CloudFront edge location doesn't contain a cached response, CloudFront performs SSL/TLS negotiation with your origin and, when the negotiation is complete, forwards the request to your origin in an encrypted format.
4. Your origin decrypts the request, processes it (generates a response), encrypts the response, and returns the response to CloudFront.
5. CloudFront decrypts the response, re-encrypts it, and forwards it to the viewer. CloudFront also caches the response in the edge location so that it's available the next time it's requested.
6. The viewer decrypts the response.

The process works basically the same way whether your origin is an Amazon S3 bucket, MediaStore, or a custom origin such as an HTTP/S server.

> **Note**
>
> To help thwart SSL renegotiation-type attacks, CloudFront does not support renegotiation for viewer and origin requests.

For information about how to require HTTPS between viewers and CloudFront, and between CloudFront and your origin, see the following topics.

Topics

- Requiring HTTPS for communication between viewers and CloudFront
- Requiring HTTPS for communication between CloudFront and your custom origin
- Requiring HTTPS for communication between CloudFront and your Amazon S3 origin
- Supported protocols and ciphers between viewers and CloudFront
- Supported protocols and ciphers between CloudFront and the origin
- Charges for HTTPS connections

Requiring HTTPS for communication between viewers and CloudFront

You can configure one or more cache behaviors in your CloudFront distribution to require HTTPS for communication between viewers and CloudFront. You also can configure one or more cache behaviors to allow both HTTP and HTTPS, so that CloudFront requires HTTPS for some objects but not for others. The configuration steps depend on which domain name you're using in object URLs:

- If you're using the domain name that CloudFront assigned to your distribution, such as d111111abcdef8.cloudfront.net, you change the **Viewer Protocol Policy** setting for one or more cache behaviors to require HTTPS communication. In that configuration, CloudFront provides the SSL/TLS certificate.

  To change the value of **Viewer Protocol Policy** by using the CloudFront console, see the procedure later in this section.

  For information about how to use the CloudFront API to change the value of the `ViewerProtocolPolicy` element, see UpdateDistribution in the *Amazon CloudFront API Reference*.
- If you're using your own domain name, such as example.com, you need to change several CloudFront settings. You also need to use an SSL/TLS certificate provided by Amazon Certificate Manager (ACM), or import a certificate from a third-party certificate authority into ACM or the IAM certificate store. For more information, see Using alternate domain names and HTTPS.

> **Note**
>
> If you want to ensure that the objects that viewers get from CloudFront were encrypted when CloudFront got them from your origin, always use HTTPS between CloudFront and your origin. If you recently changed from HTTP to HTTPS between CloudFront and your origin, we recommend that you invalidate objects in CloudFront edge locations. CloudFront will return an object to a viewer regardless of whether the protocol used by the viewer (HTTP or HTTPS) matches the protocol that CloudFront used to get the object. For more information about removing or replacing objects in a distribution, see Adding, removing, or replacing content that CloudFront distributes.

To require HTTPS between viewers and CloudFront for one or more cache behaviors, perform the following procedure.

To configure CloudFront to require HTTPS between viewers and CloudFront

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the top pane of the CloudFront console, choose the ID for the distribution that you want to update.
3. On the **Behaviors** tab, select the cache behavior that you want to update, and then choose **Edit**.
4. Specify one of the following values for **Viewer protocol policy**:

   Redirect HTTP to HTTPS

   Viewers can use both protocols. HTTP `GET` and `HEAD` requests are automatically redirected to HTTPS requests. CloudFront returns HTTP status code 301 (Moved Permanently) along with the new HTTPS URL. The viewer then resubmits the request to CloudFront using the HTTPS URL.

   > **Important**
   >
   > If you send `POST`, `PUT`, `DELETE`, `OPTIONS`, or `PATCH` over HTTP with an HTTP to HTTPS cache behavior and a request protocol version of HTTP 1.1 or above, CloudFront redirects the request to a HTTPS location with a HTTP status code 307 (Temporary Redirect). This guarantees that the request is sent again to the new location using the same method and body payload.
   > If you send `POST`, `PUT`, `DELETE`, `OPTIONS`, or `PATCH` requests over HTTP to HTTPS cache behavior with a request protocol version below HTTP 1.1, CloudFront returns a HTTP status code 403 (Forbidden).

   When a viewer makes an HTTP request that is redirected to an HTTPS request, CloudFront charges for both requests. For the HTTP request, the charge is only for the request and for the headers that CloudFront returns to the viewer. For the HTTPS request, the charge is for the request, and for the headers and the object that are returned by your origin.

HTTPS only

Viewers can access your content only if they're using HTTPS. If a viewer sends an HTTP request instead of an HTTPS request, CloudFront returns HTTP status code 403 (Forbidden) and does not return the object.

5. Choose **Save changes**.
6. Repeat steps 3 through 5 for each additional cache behavior that you want to require HTTPS for between viewers and CloudFront.
7. Confirm the following before you use the updated configuration in a production environment:

   - The path pattern in each cache behavior applies only to the requests that you want viewers to use HTTPS for.
   - The cache behaviors are listed in the order that you want CloudFront to evaluate them in. For more information, see Path pattern.
   - The cache behaviors are routing requests to the correct origins.

Requiring HTTPS for communication between CloudFront and your custom origin

You can require HTTPS for communication between CloudFront and your origin.

> **Note**
>
> If your origin is an Amazon S3 bucket that's configured as a website endpoint, you can't configure CloudFront to use HTTPS with your origin because Amazon S3 doesn't support HTTPS for website endpoints.

To require HTTPS between CloudFront and your origin, follow the procedures in this topic to do the following:

1. In your distribution, change the **Origin Protocol Policy** setting for the origin.
2. Install an SSL/TLS certificate on your origin server (this isn't required when you use an Amazon S3 origin or certain other Amazon origins).

Topics

- Changing CloudFront settings
- Installing an SSL/TLS certificate on your custom origin

Changing CloudFront settings

The following procedure explains how to configure CloudFront to use HTTPS to communicate with an Elastic Load Balancing load balancer, an Amazon EC2 instance, or another custom origin. For information about using the CloudFront API to update a distribution, see UpdateDistribution in the *Amazon CloudFront API Reference*.

To configure CloudFront to require HTTPS between CloudFront and your custom origin

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the top pane of the CloudFront console, choose the ID for the distribution that you want to update.
3. On the **Behaviors** tab, select the origin that you want to update, and then choose **Edit**.
4. Update the following settings:

   Origin Protocol Policy

   Change the **Origin Protocol Policy** for the applicable origins in your distribution:

   - **HTTPS Only** – CloudFront uses only HTTPS to communicate with your custom origin.
   - **Match Viewer** – CloudFront communicates with your custom origin using HTTP or HTTPS, depending on the protocol of the viewer request. For example, if you choose **Match Viewer** for **Origin Protocol Policy** and the viewer uses HTTPS to request an object from CloudFront, CloudFront also uses HTTPS to forward the request to your origin.

     Choose **Match Viewer** only if you specify **Redirect HTTP to HTTPS** or **HTTPS Only** for **Viewer Protocol Policy**.

     CloudFront caches the object only once even if viewers make requests using both HTTP and HTTPS protocols.

   Origin SSL Protocols

   Choose the **Origin SSL Protocols** for the applicable origins in your distribution. The SSLv3 protocol is less secure, so we recommend that you choose SSLv3 only if your origin doesn't

support TLSv1 or later. The TLSv1 handshake is both backwards and forwards compatible with SSLv3, but TLSv1.1 and TLSv1.2 are not. When you choose SSLv3, CloudFront *only* sends SSLv3 handshake requests.

5. Choose **Save changes**.
6. Repeat steps 3 through 5 for each additional origin that you want to require HTTPS for between CloudFront and your custom origin.
7. Confirm the following before you use the updated configuration in a production environment:

   - The path pattern in each cache behavior applies only to the requests that you want viewers to use HTTPS for.
   - The cache behaviors are listed in the order that you want CloudFront to evaluate them in. For more information, see Path pattern.
   - The cache behaviors are routing requests to the origins that you changed the **Origin Protocol Policy** for.

Installing an SSL/TLS certificate on your custom origin

You can use an SSL/TLS certificate from the following sources on your custom origin:

- If your origin is an Elastic Load Balancing load balancer, you can use a certificate provided by Amazon Certificate Manager (ACM). You also can use a certificate that is signed by a trusted third-party certificate authority and imported into ACM.
- For origins other than Elastic Load Balancing load balancers, you must use a certificate that is signed by a trusted third-party certificate authority (CA), for example, Comodo, DigiCert, or Symantec.

The certificate returned from the origin must include one of the following domain names:

- The domain name in the origin's **Origin domain** field (the `DomainName` field in the CloudFront API).
- The domain name in the `Host` header, if the cache behavior is configured to forward the `Host` header to the origin.

When CloudFront uses HTTPS to communicate with your origin, CloudFront verifies that the certificate was issued by a trusted certificate authority. CloudFront supports the same certificate

authorities that Mozilla does. For the current list, see Mozilla Included CA Certificate List. You can't use a self-signed certificate for HTTPS communication between CloudFront and your origin.

> **Important**
>
> If the origin server returns an expired certificate, an invalid certificate, or a self-signed certificate, or if the origin server returns the certificate chain in the wrong order, CloudFront drops the TCP connection, returns HTTP status code 502 (Bad Gateway) to the viewer, and sets the `X-Cache` header to `Error from cloudfront`. Also, if the full chain of certificates, including the intermediate certificate, is not present, CloudFront drops the TCP connection.

Requiring HTTPS for communication between CloudFront and your Amazon S3 origin

When your origin is an Amazon S3 bucket, your options for using HTTPS for communications with CloudFront depend on how you're using the bucket. If your Amazon S3 bucket is configured as a website endpoint, you can't configure CloudFront to use HTTPS to communicate with your origin because Amazon S3 doesn't support HTTPS connections in that configuration.

When your origin is an Amazon S3 bucket that supports HTTPS communication, CloudFront always forwards requests to S3 by using the protocol that viewers used to submit the requests. The default setting for the Protocol (custom origins only) setting is **Match Viewer** and can't be changed.

If you want to require HTTPS for communication between CloudFront and Amazon S3, you must change the value of **Viewer Protocol Policy** to **Redirect HTTP to HTTPS** or **HTTPS Only**. The procedure later in this section explains how to use the CloudFront console to change **Viewer Protocol Policy**. For information about using the CloudFront API to update the `ViewerProtocolPolicy` element for a distribution, see UpdateDistribution in the *Amazon CloudFront API Reference*.

When you use HTTPS with an Amazon S3 bucket that supports HTTPS communication, Amazon S3 provides the SSL/TLS certificate, so you don't have to.

To configure CloudFront to require HTTPS to your Amazon S3 origin

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the top pane of the CloudFront console, choose the ID for the distribution that you want to update.
3. On the **Behaviors** tab, choose the cache behavior that you want to update, and then choose **Edit**.
4. Specify one of the following values for **Viewer Protocol Policy**:

   Redirect HTTP to HTTPS

   Viewers can use both protocols, but HTTP requests are automatically redirected to HTTPS requests. CloudFront returns HTTP status code 301 (Moved Permanently) along with the new HTTPS URL. The viewer then resubmits the request to CloudFront using the HTTPS URL.

   > **Important**
   >
   > CloudFront doesn't redirect `DELETE`, `OPTIONS`, `PATCH`, `POST`, or `PUT` requests from HTTP to HTTPS. If you configure a cache behavior to redirect to HTTPS, CloudFront responds to HTTP `DELETE`, `OPTIONS`, `PATCH`, `POST`, or `PUT` requests for that cache behavior with HTTP status code 403 (Forbidden).

   When a viewer makes an HTTP request that is redirected to an HTTPS request, CloudFront charges for both requests. For the HTTP request, the charge is only for the request and for the headers that CloudFront returns to the viewer. For the HTTPS request, the charge is for the request, and for the headers and the object returned by your origin.

   HTTPS Only

   Viewers can access your content only if they're using HTTPS. If a viewer sends an HTTP request instead of an HTTPS request, CloudFront returns HTTP status code 403 (Forbidden) and does not return the object.
5. Choose **Yes, Edit**.
6. Repeat steps 3 through 5 for each additional cache behavior that you want to require HTTPS for between viewers and CloudFront, and between CloudFront and S3.

7. Confirm the following before you use the updated configuration in a production environment:

   - The path pattern in each cache behavior applies only to the requests that you want viewers to use HTTPS for.
   - The cache behaviors are listed in the order that you want CloudFront to evaluate them in. For more information, see Path pattern.
   - The cache behaviors are routing requests to the correct origins.

Supported protocols and ciphers between viewers and CloudFront

When you require HTTPS between viewers and your CloudFront distribution, you must choose a security policy, which determines the following settings:

- The minimum SSL/TLS protocol that CloudFront uses to communicate with viewers.
- The ciphers that CloudFront can use to encrypt the communication with viewers.

To choose a security policy, specify the applicable value for Security policy. The following table lists the protocols and ciphers that CloudFront can use for each security policy.

A viewer must support at least one of the supported ciphers to establish an HTTPS connection with CloudFront. CloudFront chooses a cipher in the listed order from among the ciphers that the viewer supports. See also OpenSSL, s2n, and RFC cipher names.

	Security policy						
	SSLv3	**TLSv1**	**TLSv1_2016**	**TLSv1.1_2016**	**TLSv1.2_2018**	**TLSv1.2_2019**	**TLSv1.2_2021**
Supported SSL/TLS protocols							
TLSv1.3	♦	♦	♦	♦	♦	♦	♦
TLSv1.2	♦	♦	♦	♦	♦	♦	♦
TLSv1.1	♦	♦	♦	♦			
TLSv1	♦	♦	♦				

	Security policy						
	SSLv3	**TLSv1**	**TLSv1_26**	**TLSv1.1_016**	**TLSv1.2_018**	**TLSv1.2_019**	**TLSv1.2_2021**
SSLv3	♦						
Supported TLSv1.3 ciphers							
TLS_AES_128_GCM_SHA256	♦	♦	♦	♦	♦	♦	♦
TLS_AES_256_GCM_SHA384	♦	♦	♦	♦	♦	♦	♦
TLS_CHACHA20_POLY1305_SHA256	♦	♦	♦	♦	♦	♦	♦
Supported ECDSA ciphers							
ECDHE-ECDSA-AES128-GCM-SHA256	♦	♦	♦	♦	♦	♦	♦
ECDHE-ECDSA-AES128-SHA256	♦	♦	♦	♦	♦	♦	
ECDHE-ECDSA-AES128-SHA	♦	♦	♦	♦			
ECDHE-ECDSA-AES256-GCM-SHA384	♦	♦	♦	♦	♦	♦	♦
ECDHE-ECDSA-CHACHA20-POLY1305	♦	♦	♦	♦	♦	♦	♦
ECDHE-ECDSA-AES256-SHA384	♦	♦	♦	♦	♦	♦	
ECDHE-ECDSA-AES256-SHA	♦	♦	♦	♦			

	Security policy						
	SSLv3	TLSv1	TLSv1_26	TLSv1.1_016	TLSv1.2_018	TLSv1.2_019	TLSv1.2_2021
Supported RSA ciphers							
ECDHE-RSA-AES128-GCM-SHA256	♦	♦	♦	♦	♦	♦	♦
ECDHE-RSA-AES128-SHA256	♦	♦	♦	♦	♦	♦	
ECDHE-RSA-AES128-SHA	♦	♦	♦	♦			
ECDHE-RSA-AES256-GCM-SHA384	♦	♦	♦	♦	♦	♦	♦
ECDHE-RSA-CHACHA20-POLY1305	♦	♦	♦	♦	♦	♦	♦
ECDHE-RSA-AES256-SHA384	♦	♦	♦	♦	♦	♦	
ECDHE-RSA-AES256-SHA	♦	♦	♦	♦			
AES128-GCM-SHA256	♦	♦	♦	♦	♦		
AES256-GCM-SHA384	♦	♦	♦	♦	♦		
AES128-SHA256	♦	♦	♦	♦	♦		
AES256-SHA	♦	♦	♦	♦			
AES128-SHA	♦	♦	♦	♦			
DES-CBC3-SHA	♦	♦					
RC4-MD5	♦						

OpenSSL, s2n, and RFC cipher names

OpenSSL and s2n use different names for ciphers than the TLS standards use (RFC 2246, RFC 4346, RFC 5246, and RFC 8446). The following table maps the OpenSSL and s2n names to the RFC name for each cipher.

For ciphers with elliptic curve key exchange algorithms, CloudFront supports the following elliptic curvers:

- prime256v1
- secp384r1
- X25519

OpenSSL and s2n cipher name	RFC cipher name
Supported TLSv1.3 ciphers	
TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256
Supported ECDSA ciphers	
ECDHE-ECDSA-AES128-GCM-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
ECDHE-ECDSA-AES128-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
ECDHE-ECDSA-AES128-SHA	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
ECDHE-ECDSA-AES256-GCM-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
ECDHE-ECDSA-CHACHA20-POLY1305	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256

OpenSSL and s2n cipher name	RFC cipher name
ECDHE-ECDSA-AES256-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
ECDHE-ECDSA-AES256-SHA	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
Supported RSA ciphers	
ECDHE-RSA-AES128-GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
ECDHE-RSA-AES256-GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE-RSA-CHACHA20-POLY1305	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
AES256-GCM-SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384
AES128-SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA
RC4-MD5	TLS_RSA_WITH_RC4_128_MD5

Supported signature schemes between viewers and CloudFront

CloudFront supports the following signature schemes for connections between viewers and CloudFront.

- TLS_SIGNATURE_SCHEME_RSA_PSS_PSS_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PSS_PSS_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PSS_PSS_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PSS_RSAE_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PSS_RSAE_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PSS_RSAE_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA224
- TLS_SIGNATURE_SCHEME_ECDSA_SHA256
- TLS_SIGNATURE_SCHEME_ECDSA_SHA384
- TLS_SIGNATURE_SCHEME_ECDSA_SHA512
- TLS_SIGNATURE_SCHEME_ECDSA_SHA224
- TLS_SIGNATURE_SCHEME_ECDSA_SECP256R1_SHA256
- TLS_SIGNATURE_SCHEME_ECDSA_SECP384R1_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA1
- TLS_SIGNATURE_SCHEME_ECDSA_SHA1

Supported protocols and ciphers between CloudFront and the origin

If you choose to require HTTPS between CloudFront and your origin, you can decide which SSL/TLS protocol to allow for the secure connection, and CloudFront can connect to the origin using any of the ECDSA or RSA ciphers listed in the following table. Your origin must support at least one of these ciphers for CloudFront to establish an HTTPS connection to your origin.

OpenSSL and s2n use different names for ciphers than the TLS standards use (RFC 2246, RFC 4346, RFC 5246, and RFC 8446). The following table includes the OpenSSL and s2n name, and the RFC name, for each cipher.

For ciphers with elliptic curve key exchange algorithms, CloudFront supports the following elliptic curvers:

- prime256v1
- secp384r1
- X25519

OpenSSL and s2n cipher name	RFC cipher name
Supported ECDSA ciphers	
ECDHE-ECDSA-AES256-GCM-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
ECDHE-ECDSA-AES256-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
ECDHE-ECDSA-AES256-SHA	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
ECDHE-ECDSA-AES128-GCM-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
ECDHE-ECDSA-AES128-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
ECDHE-ECDSA-AES128-SHA	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
Supported RSA ciphers	
ECDHE-RSA-AES256-GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
ECDHE-RSA-AES128-GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

OpenSSL and s2n cipher name	RFC cipher name
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA
RC4-MD5	TLS_RSA_WITH_RC4_128_MD5

Supported signature schemes between CloudFront and the origin

CloudFront supports the following signature schemes for connections between CloudFront and the origin.

- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA224
- TLS_SIGNATURE_SCHEME_ECDSA_SHA256
- TLS_SIGNATURE_SCHEME_ECDSA_SHA384
- TLS_SIGNATURE_SCHEME_ECDSA_SHA512
- TLS_SIGNATURE_SCHEME_ECDSA_SHA224
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA1
- TLS_SIGNATURE_SCHEME_ECDSA_SHA1

Charges for HTTPS connections

You always incur a surcharge for HTTPS requests. For more information, see Amazon CloudFront Pricing.

Using alternate domain names and HTTPS

If you want to use your own domain name in the URLs for your files (for example, `https://www.example.com/image.jpg`) and you want your viewers to use HTTPS, you must complete the steps in the following topics. (If you use the default CloudFront distribution domain name in your URLs, for example, `https://d111111abcdef8.cloudfront.net/image.jpg`, follow the guidance in the following topic instead: Requiring HTTPS for communication between viewers and CloudFront.)

> **Important**
>
> When you add a certificate to your distribution, CloudFront immediately propagates the certificate to all of its edge locations. As new edge locations become available, CloudFront propagates the certificate to those locations, too. You can't restrict the edge locations that CloudFront propagates the certificates to.

Topics

- Choosing how CloudFront serves HTTPS requests
- Requirements for using SSL/TLS certificates with CloudFront
- Quotas on using SSL/TLS certificates with CloudFront (HTTPS between viewers and CloudFront only)
- Configuring alternate domain names and HTTPS
- Determining the size of the public key in an SSL/TLS RSA certificate
- Increasing the quotas for SSL/TLS certificates
- Rotating SSL/TLS certificates
- Reverting from a custom SSL/TLS certificate to the default CloudFront certificate
- Switching from a custom SSL/TLS certificate with dedicated IP addresses to SNI

Choosing how CloudFront serves HTTPS requests

If you want your viewers to use HTTPS and to use alternate domain names for your files, choose one of the following options for how CloudFront serves HTTPS requests:

- Use Server Name Indication (SNI) – Recommended

- Use a dedicated IP address in each edge location

This section explains how each option works.

Using SNI to serve HTTPS requests (works for most clients)

Server Name Indication (SNI) is an extension to the TLS protocol that is supported by browsers and clients released after 2010. If you configure CloudFront to serve HTTPS requests using SNI, CloudFront associates your alternate domain name with an IP address for each edge location. When a viewer submits an HTTPS request for your content, DNS routes the request to the IP address for the correct edge location. The IP address to your domain name is determined during the SSL/TLS handshake negotiation; the IP address isn't dedicated to your distribution.

The SSL/TLS negotiation occurs early in the process of establishing an HTTPS connection. If CloudFront can't immediately determine which domain the request is for, it drops the connection. When a viewer that supports SNI submits an HTTPS request for your content, here's what happens:

1. The viewer automatically gets the domain name from the request URL and adds it to a field in the request header.
2. When CloudFront receives the request, it finds the domain name in the request header and responds to the request with the applicable SSL/TLS certificate.
3. The viewer and CloudFront perform SSL/TLS negotiation.
4. CloudFront returns the requested content to the viewer.

For a current list of the browsers that support SNI, see the Wikipedia entry Server Name Indication.

If you want to use SNI but some of your users' browsers don't support SNI, you have several options:

- Configure CloudFront to serve HTTPS requests by using dedicated IP addresses instead of SNI. For more information, see Using a dedicated IP address to serve HTTPS requests (works for all clients).
- Use the CloudFront SSL/TLS certificate instead of a custom certificate. This requires that you use the CloudFront domain name for your distribution in the URLs for your files, for example, `https://d111111abcdef8.cloudfront.net/logo.png`.

  If you use the default CloudFront certificate, viewers must support the SSL protocol TLSv1 or later. CloudFront doesn't support SSLv3 with the default CloudFront certificate.

You also must change the SSL/TLS certificate that CloudFront is using from a custom certificate to the default CloudFront certificate:

- If you haven't used your distribution to distribute your content, you can just change the configuration. For more information, see Updating a distribution.
- If you have used your distribution to distribute your content, you must create a new CloudFront distribution and change the URLs for your files to reduce or eliminate the amount of time that your content is unavailable. For more information, see Reverting from a custom SSL/TLS certificate to the default CloudFront certificate.

- If you can control which browser your users use, have them upgrade their browser to one that supports SNI.
- Use HTTP instead of HTTPS.

Using a dedicated IP address to serve HTTPS requests (works for all clients)

Server Name Indication (SNI) is one way to associate a request with a domain. Another way is to use a dedicated IP address. If you have users who can't upgrade to a browser or client released after 2010, you can use a dedicated IP address to serve HTTPS requests. For a current list of the browsers that support SNI, see the Wikipedia entry Server Name Indication.

> **Important**
>
> If you configure CloudFront to serve HTTPS requests using dedicated IP addresses, you incur an additional monthly charge. The charge begins when you associate your SSL/TLS certificate with a distribution and you enable the distribution. For more information about CloudFront pricing, see Amazon CloudFront Pricing. In addition, see Using the Same Certificate for Multiple CloudFront Distributions.

When you configure CloudFront to serve HTTPS requests by using dedicated IP addresses, CloudFront associates your certificate with a dedicated IP address in each CloudFront edge location. When a viewer submits an HTTPS request for your content, here's what happens:

1. DNS routes the request to the IP address for your distribution in the applicable edge location.
2. If a client request provides the SNI extension in the `ClientHello` message, CloudFront searches for a distribution that is associated with that SNI.

- If there's a match, CloudFront responds to the request with the SSL/TLS certificate.
- If there's no match, CloudFront uses the IP address instead to identify your distribution and to determine which SSL/TLS certificate to return to the viewer.

3. The viewer and CloudFront perform SSL/TLS negotiation using your SSL/TLS certificate.
4. CloudFront returns the requested content to the viewer.

This method works for every HTTPS request, regardless of the browser or other viewer that the user is using.

Requesting permission to use three or more dedicated IP SSL/TLS certificates

If you need permission to permanently associate three or more SSL/TLS dedicated IP certificates with CloudFront, perform the following procedure. For more details about HTTPS requests, see Choosing how CloudFront serves HTTPS requests.

> **Note**
>
> This procedure is for using three or more dedicated IP certificates across your CloudFront distributions. The default value is 2. Keep in mind you cannot bind more than one SSL certificate to a distribution.
> You can only associate a single SSL/TLS certificate to a CloudFront distribution at a time. This number is for the total number of dedicated IP SSL certificates you can use across all of your CloudFront distributions.

To request permission to use three or more certificates with a CloudFront distribution

1. Go to the Support Center and create a case.
2. Indicate how many certificates you need permission to use, and describe the circumstances in your request. We'll update your account as soon as possible.
3. Continue with the next procedure.

Requirements for using SSL/TLS certificates with CloudFront

The requirements for SSL/TLS certificates are described in this topic. They apply to both of the following, except as noted:

- Certificates for using HTTPS between viewers and CloudFront
- Certificates for using HTTPS between CloudFront and your origin

Topics

- Certificate issuer
- Amazon Web Services Region for Amazon Certificate Manager
- Certificate format
- Intermediate certificates
- Key type
- Private key
- Permissions
- Size of the certificate key
- Supported types of certificates
- Certificate expiration date and renewal
- Domain names in the CloudFront distribution and in the certificate
- Minimum SSL/TLS protocol version
- Supported HTTP versions

Certificate issuer

We recommend that you use a certificate issued by Amazon Certificate Manager (ACM). For information about getting a certificate from ACM, see the *Amazon Certificate Manager User Guide*. To use an ACM certificate with CloudFront, make sure you request (or import) the certificate in the US East (N. Virginia) Region (`us-east-1`).

CloudFront supports the same certificate authorities (CAs) as Mozilla, so if you don't use ACM, use a certificate issued by a CA on the Mozilla Included CA Certificate List. For more information about getting and installing a certificate, refer to the documentation for your HTTP server software and to the documentation for the CA.

Amazon Web Services Region for Amazon Certificate Manager

To use a certificate in Amazon Certificate Manager (ACM) to require HTTPS between viewers and CloudFront, make sure you request (or import) the certificate in the US East (N. Virginia) Region (`us-east-1`).

If you want to require HTTPS between CloudFront and your origin, and you're using a load balancer in Elastic Load Balancing as your origin, you can request or import the certificate in any Amazon Web Services Region.

Certificate format

The certificate must be in X.509 PEM format. This is the default format if you're using Amazon Certificate Manager.

Intermediate certificates

If you're using a third-party certificate authority (CA), list all of the intermediate certificates in the certificate chain that's in the `.pem` file, beginning with one for the CA that signed the certificate for your domain. Typically, you'll find a file on the CA website that lists intermediate and root certificates in the proper chained order.

> **Important**
>
> Do not include the following: the root certificate, intermediate certificates that are not in the trust path, or your CA's public key certificate.

Here's an example:

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 2
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Intermediate certificate 1
-----END CERTIFICATE-----
```

Key type

CloudFront supports RSA and ECDSA public–private key pairs.

CloudFront supports HTTPS connections to both viewers and origins using RSA and ECDSA certificates. With Amazon Certificate Manager (ACM), you can request and import RSA or ECDSA certificates and then associate them with your CloudFront distribution.

For lists of the RSA and ECDSA ciphers supported by CloudFront that you can negotiate in HTTPS connections, see the section called "Supported protocols and ciphers between viewers and

CloudFront" and the section called "Supported protocols and ciphers between CloudFront and the origin".

Private key

If you're using a certificate from a third-party certificate authority (CA), note the following:

- The private key must match the public key that is in the certificate.
- The private key must be in PEM format.
- The private key cannot be encrypted with a password.

If Amazon Certificate Manager (ACM) provided the certificate, ACM doesn't release the private key. The private key is stored in ACM for use by Amazon services that are integrated with ACM.

Permissions

You must have permission to use and import the SSL/TLS certificate. If you're using Amazon Certificate Manager (ACM), we recommend that you use Amazon Identity and Access Management permissions to restrict access to the certificates. For more information, see Identity and access management in the *Amazon Certificate Manager User Guide*.

Size of the certificate key

The certificate key size that CloudFront supports depends on the type of key and certificate.

For RSA certificates:

CloudFront supports 1024-bit, 2048-bit, and 3072-bit, and 4096-bit RSA keys. The maximum key length for an RSA certificate that you use with CloudFront is 4096 bits.

Note that ACM issues RSA certificates with up to 2048-bit keys. To use a 3072-bit or 4096-bit RSA certificate, you need to obtain the certificate externally and import it into ACM, after which it will be available for you to use with CloudFront.

For information about how to determine the size of an RSA key, see Determining the size of the public key in an SSL/TLS RSA certificate.

For ECDSA certificates:

CloudFront supports 256-bit keys. To use an ECDSA certificate in ACM to require HTTPS between viewers and CloudFront, use the prime256v1 elliptic curve.

Supported types of certificates

CloudFront supports all types of certificates issued by a trusted certificate authority.

Certificate expiration date and renewal

If you're using certificates that you get from a third-party certificate authority (CA), you must monitor certificate expiration dates and renew the certificates that you import into Amazon Certificate Manager (ACM) or upload to the Amazon Identity and Access Management certificate store before they expire.

If you're using ACM-provided certificates, ACM manages certificate renewals for you. For more information, see Managed renewal in the *Amazon Certificate Manager User Guide*.

Domain names in the CloudFront distribution and in the certificate

When you're using a custom origin, the SSL/TLS certificate on your origin includes a domain name in the **Common Name** field, and possibly several more in the **Subject Alternative Names** field. (CloudFront supports wildcard characters in certificate domain names.)

One of the domain names in the certificate must match the domain name that you specify for Origin Domain Name. If no domain name matches, CloudFront returns HTTP status code `502 (Bad Gateway)` to the viewer.

> **Important**
>
> When you add an alternate domain name to a distribution, CloudFront checks that the alternate domain name is covered by the certificate that you've attached. The certificate must cover the alternate domain name in the subject alternate name (SAN) field of the certificate. This means the SAN field must contain an exact match for the alternate domain name, or contain a wildcard at the same level of the alternate domain name that you're adding.
>
> For more information, see Requirements for using alternate domain names.

Minimum SSL/TLS protocol version

If you're using dedicated IP addresses, set the minimum SSL/TLS protocol version for the connection between viewers and CloudFront by choosing a security policy.

For more information, see Security policy in the topic Values that you specify when you create or update a distribution.

Supported HTTP versions

If you associate one certificate with more than one CloudFront distribution, all the distributions associated with the certificate must use the same option for Supported HTTP versions. You specify this option when you create or update a CloudFront distribution.

Quotas on using SSL/TLS certificates with CloudFront (HTTPS between viewers and CloudFront only)

Note the following quotas (formerly known as limits) on using SSL/TLS certificates with CloudFront. These quotas apply only to the SSL/TLS certificates that you provision by using Amazon Certificate Manager (ACM), that you import into ACM, or upload to the IAM certificate store for HTTPS communication between viewers and CloudFront.

Maximum number of certificates per CloudFront distribution

You can associate a maximum of one SSL/TLS certificate with each CloudFront distribution.

Maximum number of certificates that you can import into ACM or upload to the IAM certificate store

If you obtained your SSL/TLS certificates from a third-party CA, you must store the certificates in one of the following locations:

- **Amazon Certificate Manager** – For the current quota on the number of ACM certificates, see Quotas in the *Amazon Certificate Manager User Guide*. The listed quota is a total that includes certificates that you provision by using ACM and certificates that you import into ACM.
- **IAM certificate store** – For the current quota (formerly known as limit) on the number of certificates that you can upload to the IAM certificate store for an Amazon account, see IAM and STS Limits in the *IAM User Guide*. You can request a higher quota in the Amazon Web Services Management Console.

Maximum number of certificates per Amazon account (dedicated IP addresses only)

If you want to serve HTTPS requests by using dedicated IP addresses, note the following:

- By default, CloudFront gives you permission to use two certificates with your Amazon account, one for everyday use and one for when you need to rotate certificates for multiple distributions.

- If you need more than two custom SSL/TLS certificates for your Amazon account, go to the Support Center and create a case. Indicate how many certificates that you need permission to use, and describe the circumstances in your request. We'll update your account as soon as possible.

Using the same certificate for CloudFront distributions that were created by using different Amazon accounts

If you're using a third-party CA and you want to use the same certificate with multiple CloudFront distributions that were created by using different Amazon accounts, you must import the certificate into ACM or upload it to the IAM certificate store once for each Amazon account.

If you're using certificates provided by ACM, you can't configure CloudFront to use certificates that were created by a different Amazon account.

Using the same certificate for CloudFront and for other Amazon services

If you bought a certificate from a trusted certificate authority such as Comodo, DigiCert, or Symantec, you can use the same certificate for CloudFront and for other Amazon services. If you're importing the certificate into ACM, you need to import it only once to use it for multiple Amazon services.

If you're using certificates provided by ACM, the certificates are stored in ACM.

Using the same certificate for multiple CloudFront distributions

You can use the same certificate for any or all of the CloudFront distributions that you're using to serve HTTPS requests. Note the following:

- You can use the same certificate both for serving requests using dedicated IP addresses and for serving requests using SNI.
- You can associate only one certificate with each distribution.
- Each distribution must include one or more alternate domain names that also appear in the **Common Name** field or the **Subject Alternative Names** field in the certificate.
- If you're serving HTTPS requests using dedicated IP addresses and you created all of your distributions by using the same Amazon account, you can significantly reduce your cost by using the same certificate for all distributions. CloudFront charges for each certificate, not for each distribution.

For example, suppose you create three distributions by using the same Amazon account, and you use the same certificate for all three distributions. You would be charged only one fee for using dedicated IP addresses.

However, if you're serving HTTPS requests using dedicated IP addresses and using the same certificate to create CloudFront distributions in different Amazon accounts, each account is charged the fee for using dedicated IP addresses. For example, if you create three distributions by using three different Amazon accounts and you use the same certificate for all three distributions, each account is charged the full fee for using dedicated IP addresses.

Configuring alternate domain names and HTTPS

To use alternate domain names in the URLs for your files and to use HTTPS between viewers and CloudFront, perform the applicable procedures.

Topics

- Getting an SSL/TLS certificate
- Importing an SSL/TLS certificate
- Updating your CloudFront distribution

Getting an SSL/TLS certificate

Get an SSL/TLS certificate if you don’t already have one. For more information, see the applicable documentation:

- To use a certificate provided by Amazon Certificate Manager (ACM), see the Amazon Certificate Manager User Guide. Then skip to Updating your CloudFront distribution.

  > **Note**
  >
  > We recommend that you use ACM to provision, manage, and deploy SSL/TLS certificates on Amazon managed resources. You must request an ACM certificate in the US East (N. Virginia) Region.

- To get a certificate from a third-party certificate authority (CA), see the documentation provided by the certificate authority. When you have the certificate, continue with the next procedure.

Importing an SSL/TLS certificate

If you got your certificate from a third-party CA, import the certificate into ACM or upload it to the IAM certificate store:

ACM (recommended)

ACM lets you import third-party certificates from the ACM console, as well as programmatically. For information about importing a certificate to ACM, see Importing Certificates into Amazon Certificate Manager in the *Amazon Certificate Manager User Guide*. You must import the certificate in the US East (N. Virginia) Region.

IAM certificate store

(Not recommended) Use the following Amazon CLI command to upload your third-party certificate to the IAM certificate store.

```
aws iam upload-server-certificate \
        --server-certificate-name CertificateName \
        --certificate-body file://public_key_certificate_file \
        --private-key file://privatekey.pem \
        --certificate-chain file://certificate_chain_file \
        --path /cloudfront/path/
```

Note the following:

- **Amazon account** – You must upload the certificate to the IAM certificate store using the same Amazon account that you used to create your CloudFront distribution.
- **--path parameter** – When you upload the certificate to IAM, the value of the `--path` parameter (certificate path) must start with `/cloudfront/`, for example, `/cloudfront/production/` or `/cloudfront/test/`. The path must end with a /.
- **Existing certificates** – You must specify values for the `--server-certificate-name` and `--path` parameters that are different from the values that are associated with existing certificates.
- **Using the CloudFront console** – The value that you specify for the `--server-certificate-name` parameter in the Amazon CLI, for example, `myServerCertificate`, appears in the **SSL Certificate** list in the CloudFront console.
- **Using the CloudFront API** – Make note of the alphanumeric string that the Amazon CLI returns, for example, `AS1A2M3P4L5E67SIIXR3J`. This is the value that you will specify in the `IAMCertificateId` element. You don't need the IAM ARN, which is also returned by the CLI.

For more information about the Amazon CLI, see the Amazon Command Line Interface User Guide and the Amazon CLI Command Reference.

Updating your CloudFront distribution

To update settings for your distribution, perform the following procedure:

To configure your CloudFront distribution for alternate domain names

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose the ID for the distribution that you want to update.
3. On the **General** tab, choose **Edit**.
4. Update the following values:

   Alternate domain name (CNAME)

   Choose **Add item** to add the applicable alternate domain names. Separate domain names with commas, or type each domain name on a new line.

   Custom SSL certificate

   Select a certificate from the dropdown list.

   Up to 100 certificates are listed here. If you have more than 100 certificates and you don't see the certificate that you want to add, you can type a certificate ARN in the field to choose it.

   If you uploaded a certificate to the IAM certificate store but it's not listed, and you can't choose it by typing the name in the field, review the procedure Importing an SSL/TLS certificate to confirm that you correctly uploaded the certificate.

   > **Important**
   >
   > After you associate your SSL/TLS certificate with your CloudFront distribution, do not delete the certificate from ACM or the IAM certificate store until you remove the certificate from all distributions and all the distributions are deployed.

5. Choose **Save changes**.

6. Configure CloudFront to require HTTPS between viewers and CloudFront:

   a. On the **Behaviors** tab, choose the cache behavior that you want to update, and choose **Edit**.

   b. Specify one of the following values for **Viewer Protocol Policy**:

      Redirect HTTP to HTTPS

      Viewers can use both protocols, but HTTP requests are automatically redirected to HTTPS requests. CloudFront returns HTTP status code `301 (Moved Permanently)` along with the new HTTPS URL. The viewer then resubmits the request to CloudFront using the HTTPS URL.

      > **Important**
      >
      > CloudFront doesn't redirect `DELETE`, `OPTIONS`, `PATCH`, `POST`, or `PUT` requests from HTTP to HTTPS. If you configure a cache behavior to redirect to HTTPS, CloudFront responds to HTTP `DELETE`, `OPTIONS`, `PATCH`, `POST`, or `PUT` requests for that cache behavior with HTTP status code `403 (Forbidden)`.

      When a viewer makes an HTTP request that is redirected to an HTTPS request, CloudFront charges for both requests. For the HTTP request, the charge is only for the request and for the headers that CloudFront returns to the viewer. For the HTTPS request, the charge is for the request, and for the headers and the file returned by your origin.

      HTTPS Only

      Viewers can access your content only if they're using HTTPS. If a viewer sends an HTTP request instead of an HTTPS request, CloudFront returns HTTP status code `403 (Forbidden)` and does not return the file.

   c. Choose **Yes, Edit**.

   d. Repeat steps a through c for each additional cache behavior that you want to require HTTPS for between viewers and CloudFront.

7. Confirm the following before you use the updated configuration in a production environment:

- The path pattern in each cache behavior applies only to the requests that you want viewers to use HTTPS for.
- The cache behaviors are listed in the order that you want CloudFront to evaluate them in. For more information, see Path pattern.
- The cache behaviors are routing requests to the correct origins.

Determining the size of the public key in an SSL/TLS RSA certificate

When you're using CloudFront alternate domain names and HTTPS, the maximum size of the public key in an SSL/TLS RSA certificate is 4096 bits. (This is the key size, not the number of characters in the public key.) If you use Amazon Certificate Manager for your certificates, although ACM supports larger RSA keys, you cannot use the larger keys with CloudFront.

You can determine the size of the RSA public key by running the following OpenSSL command:

```
openssl x509 -in path and filename of SSL/TLS certificate -text -noout
```

Where:

- `-in` specifies the path and file name of your SSL/TLS RSA certificate.
- `-text` causes OpenSSL to display the length of the RSA public key in bits.
- `-noout` prevents OpenSSL from displaying the public key.

Example output:

```
Public-Key: (2048 bit)
```

Increasing the quotas for SSL/TLS certificates

There are quotas (formerly known as limits) on the number of SSL/TLS certificates that you can import into Amazon Certificate Manager or upload to Amazon Identity and Access Management. There also is a quota on the number of SSL/TLS certificates that you can use with an Amazon account when you configure CloudFront to serve HTTPS requests by using dedicated IP addresses. However, you can request higher quotas.

Topics

- Certificates that you can import into ACM
- Certificates that you can upload to IAM
- Certificates that you can use with dedicated IP addresses

Certificates that you can import into ACM

For the quota on the number of certificates that you can import into ACM, see Quotas in the *Amazon Certificate Manager User Guide*.

To request a higher quota, create a case in the Support Center Console. Specify the following values:

- Accept the default value of **Service limit increase**.
- For **Limit type**, choose **Certificate Manager**.
- For **Region**, choose the Amazon Region where you want to import certificates.
- For **Limit**, choose **Number of ACM certificates**.

Then fill out the rest of the form and submit it.

Certificates that you can upload to IAM

For the quota (formerly known as limit) on the number of certificates that you can upload to IAM, see IAM and STS Limits in the *IAM User Guide*.

To request a higher quota, create a case in the Support Center Console. Specify the following values:

- Accept the default value of **Service limit increase**.
- For **Limit type**, choose **Certificate Manager**.
- For **Region**, choose the Amazon Region where you want to import certificates.
- For **Limit**, choose **Server Certificate Limit (IAM)**.

Then fill out the rest of the form and submit it.

Certificates that you can use with dedicated IP addresses

For the quota (formerly known as limit) on the number of SSL certificates that you can use for each Amazon account when serving HTTPS requests using dedicated IP addresses, see Quotas on SSL certificates.

To request a higher quota, create a case in the Support Center Console. Specify the following values:

- Accept the default value of **Service limit increase**.
- For **Limit Type**, choose **CloudFront Distributions**.
- For **Limit**, choose **Dedicated IP SSL Certificate Limit per Account**.

Then fill out the rest of the form and submit it.

Rotating SSL/TLS certificates

If you're using certificates provided by Amazon Certificate Manager (ACM), you don't need to rotate SSL/TLS certificates. ACM manages certificate renewals for you. For more information, see Managed Renewal in the *Amazon Certificate Manager User Guide*.

> **Note**
>
> ACM does not manage certificate renewals for certificates that you acquire from third-party certificate authorities and import into ACM.

If you're using a third-party certificate authority and you imported certificates into ACM (recommended) or uploaded them to the IAM certificate store, you must occasionally replace one certificate with another. For example, you must replace a certificate when the expiration date on the certificate approaches.

> **Important**
>
> If you configured CloudFront to serve HTTPS requests by using dedicated IP addresses, you might incur an additional, pro-rated charge for using one or more additional certificates while you're rotating certificates. We recommend that you update your distributions promptly to minimize the additional charge.

To rotate certificates, perform the following procedure. Viewers can continue to access your content while you rotate certificates as well as after the process is complete.

To rotate SSL/TLS certificates

1. Increasing the quotas for SSL/TLS certificates to determine whether you need permission to use more SSL certificates. If so, request permission and wait until permission is granted before you continue with step 2.
2. Import the new certificate into ACM or upload it to IAM. For more information, see Importing an SSL/TLS Certificate in the *Amazon CloudFront Developer Guide*.
3. Update your distributions one at a time to use the new certificate. For more information, see Listing, Viewing, and Updating CloudFront Distributions in the *Amazon CloudFront Developer Guide*.
4. (Optional) After you have updated all of your CloudFront distributions, you can delete the old certificate from ACM or from IAM.

> **Important**
>
> Do not delete an SSL/TLS certificate until you remove it from all distributions and until the status of the distributions that you have updated has changed to `Deployed`.

Reverting from a custom SSL/TLS certificate to the default CloudFront certificate

If you configured CloudFront to use HTTPS between viewers and CloudFront, and you configured CloudFront to use a custom SSL/TLS certificate, you can change your configuration to use the default CloudFront SSL/TLS certificate. The process depends on whether you've used your distribution to distribute your content:

- If you have not used your distribution to distribute your content, you can just change the configuration. For more information, see Updating a distribution.
- If you have used your distribution to distribute your content, you must create a new CloudFront distribution and change the URLs for your files to reduce or eliminate the amount of time that your content is unavailable. To do that, perform the following procedure.

To revert to the default CloudFront certificate

1. Create a new CloudFront distribution with the desired configuration. For **SSL Certificate**, choose **Default CloudFront Certificate (*.cloudfront.net)**.

   For more information, see Steps for creating a distribution (overview).

2. For files that you're distributing using CloudFront, update the URLs in your application to use the domain name that CloudFront assigned to the new distribution. For example, change `https://www.example.com/images/logo.png` to `https://d111111abcdef8.cloudfront.net/images/logo.png`.

3. Either delete the distribution that is associated with a custom SSL/TLS certificate, or update the distribution to change the value of **SSL Certificate** to **Default CloudFront Certificate (*.cloudfront.net)**. For more information, see Updating a distribution.

   > **Important**
   >
   > Until you complete this step, Amazon continues to charge you for using a custom SSL/TLS certificate.

4. (Optional) Delete your custom SSL/TLS certificate.

   a. Run the Amazon CLI command `list-server-certificates` to get the certificate ID of the certificate that you want to delete. For more information, see list-server-certificates in the *Amazon CLI Command Reference*.

   b. Run the Amazon CLI command `delete-server-certificate` to delete the certificate. For more information, see delete-server-certificate in the *Amazon CLI Command Reference*.

Switching from a custom SSL/TLS certificate with dedicated IP addresses to SNI

If you configured CloudFront to use a custom SSL/TLS certificate with dedicated IP addresses, you can switch to using a custom SSL/TLS certificate with SNI instead and eliminate the charge that is associated with dedicated IP addresses. The following procedure shows you how.

> **Important**
>
> This update to your CloudFront configuration has no effect on viewers that support SNI. Viewers can access your content before and after the change, as well as while the change is propagating to CloudFront edge locations. Viewers that don't support SNI cannot access your content after the change. For more information, see Choosing how CloudFront serves HTTPS requests.

To switch from a custom SSL/TLS certificate with dedicated IP addresses to SNI

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose the ID of the distribution that you want to view or update.
3. Choose **Distribution Settings**.
4. On the **General** tab, choose **Edit**.
5. Change the setting of **Custom SSL Client Support** to **Only Clients that Support Server Name Indication (SNI)**.
6. Choose **Yes, Edit**.

Serving private content with signed URLs and signed cookies

Many companies that distribute content over the internet want to restrict access to documents, business data, media streams, or content that is intended for selected users, for example, users who have paid a fee. To securely serve this private content by using CloudFront, you can do the following:

- Require that your users access your private content by using special CloudFront signed URLs or signed cookies.
- Require that your users access your content by using CloudFront URLs, not URLs that access content directly on the origin server (for example, Amazon S3 or a private HTTP server). Requiring CloudFront URLs isn't necessary, but we recommend it to prevent users from bypassing the restrictions that you specify in signed URLs or signed cookies.

Topics

- Overview of serving private content
- Task list for serving private content
- Specifying the signers that can create signed URLs and signed cookies
- Choosing between signed URLs and signed cookies
- Using signed URLs
- Using signed cookies
- Using Linux commands and OpenSSL for base64 encoding and encryption
- Code examples for creating a signature for a signed URL

Overview of serving private content

You can control user access to your private content in two ways:

- Restrict access to files in CloudFront caches.
- Restrict access to files in your origin by doing one of the following:
  - Set up an origin access control (OAC) for your Amazon S3 bucket.
  - Configure custom headers for a private HTTP server (a custom origin).

Restricting access to files in CloudFront caches

You can configure CloudFront to require that users access your files using either *signed URLs* or *signed cookies*. You then develop your application either to create and distribute signed URLs to authenticated users or to send `Set-Cookie` headers that set signed cookies for authenticated users. (To give a few users long-term access to a small number of files, you can also create signed URLs manually.)

When you create signed URLs or signed cookies to control access to your files, you can specify the following restrictions:

- An ending date and time, after which the URL is no longer valid.
- (Optional) The date and time that the URL becomes valid.
- (Optional) The IP address or range of addresses of the computers that can be used to access your content.

One part of a signed URL or a signed cookie is hashed and signed using the private key from a public–private key pair. When someone uses a signed URL or signed cookie to access a file, CloudFront compares the signed and unsigned portions of the URL or cookie. If they don't match, CloudFront doesn't serve the file.

You must use RSA-SHA1 for signing URLs or cookies. CloudFront doesn't accept other algorithms.

Restricting access to files in Amazon S3 buckets

You can optionally secure the content in your Amazon S3 bucket so that users can access it through the specified CloudFront distribution but cannot access it directly by using Amazon S3 URLs. This prevents someone from bypassing CloudFront and using the Amazon S3 URL to get content that you want to restrict access to. This step isn't required to use signed URLs, but we recommend it.

To require that users access your content through CloudFront URLs, you do the following tasks:

- Give a CloudFront *origin access control* permission to read the files in the S3 bucket.
- Create the origin access control and associate it with your CloudFront distribution.
- Remove permission for anyone else to use Amazon S3 URLs to read the files.

For more information, see the section called “Restricting access to an Amazon Simple Storage Service origin”.

Restricting access to files on custom origins

If you use a custom origin, you can optionally set up custom headers to restrict access. For CloudFront to get your files from a custom origin, the files must be accessible by CloudFront using a standard HTTP (or HTTPS) request. But by using custom headers, you can further restrict access to your content so that users can access it only through CloudFront, not directly. This step isn't required to use signed URLs, but we recommend it.

To require that users access content through CloudFront, change the following settings in your CloudFront distributions:

Origin Custom Headers

Configure CloudFront to forward custom headers to your origin. See Configuring CloudFront to add custom headers to origin requests.

Viewer Protocol Policy

Configure your distribution to require viewers to use HTTPS to access CloudFront. See Viewer protocol policy.

Origin Protocol Policy

Configure your distribution to require CloudFront to use the same protocol as viewers to forward requests to the origin. See Protocol (custom origins only).

After you've made these changes, update your application on your custom origin to only accept requests that include the custom headers that you've configured CloudFront to send.

The combination of **Viewer Protocol Policy** and **Origin Protocol Policy** ensure that the custom headers are encrypted in transit. However, we recommend that you periodically do the following to rotate the custom headers that CloudFront forwards to your origin:

1. Update your CloudFront distribution to begin forwarding a new header to your custom origin.
2. Update your application to accept the new header as confirmation that the request is coming from CloudFront.
3. When requests no longer include the header that you're replacing, update your application to no longer accept the old header as confirmation that the request is coming from CloudFront.

Task list for serving private content

To configure CloudFront to serve private content, do the following tasks:

1. (Optional but recommended) Require your users to access your content only through CloudFront. The method that you use depends on whether you're using Amazon S3 or custom origins:

   - **Amazon S3** – See the section called "Restricting access to an Amazon Simple Storage Service origin".
   - **Custom origin** – See Restricting access to files on custom origins.

   Custom origins include Amazon EC2, Amazon S3 buckets configured as website endpoints, Elastic Load Balancing, and your own HTTP web servers.

2. Specify the *trusted key groups* or *trusted signers* that you want to use to create signed URLs or signed cookies. We recommend that you use trusted key groups. For more information, see Specifying the signers that can create signed URLs and signed cookies.
3. Write your application to respond to requests from authorized users either with signed URLs or with `Set-Cookie` headers that set signed cookies. Follow the steps in one of the following topics:

   - Using signed URLs
   - Using signed cookies

   If you're not sure which method to use, see Choosing between signed URLs and signed cookies.

Specifying the signers that can create signed URLs and signed cookies

Topics

- Choosing between trusted key groups (recommended) and Amazon accounts
- Creating key pairs for your signers
- Reformatting the private key (.NET and Java only)
- Adding a signer to a distribution
- Rotating key pairs

To create signed URLs or signed cookies, you need a *signer*. A signer is either a trusted key group that you create in CloudFront, or an Amazon account that contains a CloudFront key pair. We recommend that you use trusted key groups with signed URLs and signed cookies. For more information, see Choosing between trusted key groups (recommended) and Amazon accounts.

The signer has two purposes:

- As soon as you add the signer to your distribution, CloudFront starts to require that viewers use signed URLs or signed cookies to access your files.
- When you create signed URLs or signed cookies, you use the private key from the signer's key pair to sign a portion of the URL or the cookie. When someone requests a restricted file, CloudFront compares the signature in the URL or cookie with the unsigned URL or cookie, to verify that it hasn't been tampered with. CloudFront also verifies that the URL or cookie is valid, meaning, for example, that the expiration date and time hasn't passed.

When you specify a signer, you also indirectly specify the files that require signed URLs or signed cookies by adding the signer to a cache behavior. If your distribution has only one cache behavior, viewers must use signed URLs or signed cookies to access any file in the distribution. If you create multiple cache behaviors and add signers to some cache behaviors and not to others, you can require that viewers use signed URLs or signed cookies to access some files and not others.

To specify the signers (the private keys) that are allowed to create signed URLs or signed cookies, and to add the signers to your CloudFront distribution, do the following tasks:

1. Decide whether to use a trusted key group or an Amazon account as the signer. We recommend using a trusted key group. For more information, see Choosing between trusted key groups (recommended) and Amazon accounts.
2. For the signer that you chose in step 1, create a public–private key pair. For more information, see Creating key pairs for your signers.
3. If you're using .NET or Java to create signed URLs or signed cookies, reformat the private key. For more information, see Reformatting the private key (.NET and Java only).
4. In the distribution for which you're creating signed URLs or signed cookies, specify the signer. For more information, see Adding a signer to a distribution.

Choosing between trusted key groups (recommended) and Amazon accounts

To use signed URLs or signed cookies, you need a *signer*. A signer is either a trusted key group that you create in CloudFront, or an Amazon account that contains a CloudFront key pair. We recommend that you use trusted key groups, for the following reasons:

- With CloudFront key groups, you don't need to use the Amazon account root user to manage the public keys for CloudFront signed URLs and signed cookies. Amazon best practices recommend that you don't use the root user when you don't have to.
- With CloudFront key groups, you can manage public keys, key groups, and trusted signers using the CloudFront API. You can use the API to automate key creation and key rotation. When you use the Amazon root user, you have to use the Amazon Web Services Management Console to manage CloudFront key pairs, so you can't automate the process.
- Because you can manage key groups with the CloudFront API, you can also use Amazon Identity and Access Management (IAM) permissions policies to limit what different users are allowed to do. For example, you can allow users to upload public keys, but not delete them. Or you can allow users to delete public keys, but only when certain conditions are met, such as using multi-

factor authentication, sending the request from a particular network, or sending the request within a particular date and time range.

- With CloudFront key groups, you can associate a higher number of public keys with your CloudFront distribution, giving you more flexibility in how you use and manage the public keys. By default, you can associate up to four key groups with a single distribution, and you can have up to five public keys in a key group.

  When you use the Amazon account root user to manage CloudFront key pairs, you can only have up to two active CloudFront key pairs per Amazon account.

Creating key pairs for your signers

Each signer that you use to create CloudFront signed URLs or signed cookies must have a public–private key pair. The signer uses its private key to sign the URL or cookies, and CloudFront uses the public key to verify the signature.

The way that you create a key pair depends on whether you use a trusted key group as the signer (recommended), or a CloudFront key pair. For more information, see the following sections. The key pair that you create must meet the following requirements:

- It must be an SSH-2 RSA key pair.
- It must be in base64-encoded PEM format.
- It must be a 2048-bit key pair.

To help secure your applications, we recommend that you rotate key pairs periodically. For more information, see Rotating key pairs.

Create a key pair for a trusted key group (recommended)

To create a key pair for a trusted key group, perform the following steps:

1. Create the public–private key pair.
2. Upload the public key to CloudFront.
3. Add the public key to a CloudFront key group.

For more information, see the following procedures.

To create a key pair

> **Note**
>
> The following steps use OpenSSL as an example of one way to create a key pair. There are many other ways to create an RSA key pair.

1. The following example command uses OpenSSL to generate an RSA key pair with a length of 2048 bits and save to the file named `private_key.pem`.

   ```
   openssl genrsa -out private_key.pem 2048
   ```

2. The resulting file contains both the public and the private key. The following example command extracts the public key from the file named `private_key.pem`.

   ```
   openssl rsa -pubout -in private_key.pem -out public_key.pem
   ```

   You upload the public key (in the `public_key.pem` file) later, in the following procedure.

To upload the public key to CloudFront

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation menu, choose **Public keys**.
3. Choose **Create public key**.
4. In the **Create public key** window, do the following:

   a. For **Key name**, type a name to identify the public key.

   b. For **Key value**, paste the public key. If you followed the steps in the preceding procedure, the public key is in the file named `public_key.pem`. To copy and paste the contents of the public key, you can:

   - Use the **cat** command on the macOS or Linux command line, like this:

```
cat public_key.pem
```

Copy the output of that command, then paste it into the **Key value** field.

- Open the `public_key.pem` file with a plaintext editor like Notepad (on Windows) or TextEdit (on macOS). Copy the contents of the file, then paste it into the **Key value** field.

c. (Optional) For **Comment**, add a comment to describe the public key.

When finished, choose **Add**.

5. Record the public key ID. You use it later when you create signed URLs or signed cookies, as the value of the `Key-Pair-Id` field.

To add the public key to a key group

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation menu, choose **Key groups**.
3. Choose **Add key group**.
4. On the **Create key group** page, do the following:

   a. For **Key group name**, type a name to identify the key group.

   b. (Optional) For **Comment**, type a comment to describe the key group.

   c. For **Public keys**, select the public key to add to the key group, then choose **Add**. Repeat this step for each public key that you want to add to the key group.

5. Choose **Create key group**.
6. Record the key group name. You use it later to associate the key group with a cache behavior in a CloudFront distribution. (In the CloudFront API, you use the key group ID to associate the key group with a cache behavior.)

Create a CloudFront key pair (not recommended, requires the Amazon account root user)

> **Important**
>
> We recommend that you create a public key for a trusted key group instead of following these steps. For the recommended way to create public keys for signed URLs and signed cookies, see Create a key pair for a trusted key group (recommended).

You can create a CloudFront key pair in the following ways:

- Create a key pair in the Amazon Web Services Management Console and download the private key. See the following procedure.
- Create an RSA key pair by using an application such as OpenSSL, and then upload the public key to the Amazon Web Services Management Console. For more information about creating an RSA key pair, see Create a key pair for a trusted key group (recommended).

To create CloudFront key pairs in the Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console using the credentials of the Amazon account root user.

   > **Important**
   >
   > IAM users can't create CloudFront key pairs. You must sign in using root user credentials to create key pairs.

2. Choose your account name, then choose **My Security Credentials**.
3. Choose **CloudFront key pairs**.
4. Confirm that you have no more than one active key pair. You can't create a key pair if you already have two active key pairs.
5. Choose **Create New Key Pair**.

   > **Note**
   >
   > You can also choose to create your own key pair and upload the public key. CloudFront key pairs support 1024, 2048, or 4096-bit keys.

6. In the **Create Key Pair** dialog box, choose **Download Private Key File**, and then save the file on your computer.

   > **Important**
   >
   > Save the private key for your CloudFront key pair in a secure location, and set permissions on the file so that only the desired administrators can read it. If someone gets your private key, they can generate valid signed URLs and signed cookies and download your content. You cannot get the private key again, so if you lose or delete it, you must create a new CloudFront key pair.

7. Record the key pair ID for your key pair. (In the Amazon Web Services Management Console, this is called the **Access Key ID**.) You'll use it when you create signed URLs or signed cookies.

Reformatting the private key (.NET and Java only)

If you're using .NET or Java to create signed URLs or signed cookies, you cannot use the private key from your key pair in the default PEM format to create the signature. Instead, do the following:

- **.NET framework** – Convert the private key to the XML format that the .NET framework uses. Several tools are available.
- **Java** – Convert the private key to DER format. One way to do this is with the following OpenSSL command. In the following command, `private_key.pem` is the name of the file that contains the PEM-formatted private key, and `private_key.der` is the name of the file that contains the DER-formatted private key after you run the command.

  ```
  openssl pkcs8 -topk8 -nocrypt -in private_key.pem -inform PEM -out private_key.der -
  outform DER
  ```

  To ensure that the encoder works correctly, add the JAR for the Bouncy Castle Java cryptography APIs to your project and then add the Bouncy Castle provider.

Adding a signer to a distribution

A signer is the trusted key group (recommended) or CloudFront key pair that can create signed URLs and signed cookies for a distribution. To use signed URLs or signed cookies with a CloudFront distribution, you must specify a signer.

Signers are associated with cache behaviors. This allows you to require signed URLs or signed cookies for some files and not for others in the same distribution. A distribution requires signed URLs or cookies only for files that are associated with the corresponding cache behaviors.

Similarly, a signer can only sign URLs or cookies for files that are associated with the corresponding cache behaviors. For example, if you have one signer for one cache behavior and a different signer for a different cache behavior, neither signer can create signed URLs or cookies for files that are associated with the other cache behavior.

Important

Before you add a signer to your distribution, do the following:

- Define the path patterns in cache behaviors and the sequence of cache behaviors carefully so you don't give users unintended access to your content or prevent them from accessing content that you want to be available to everyone.

  For example, suppose a request matches the path pattern for two cache behaviors. The first cache behavior does not require signed URLs or signed cookies and the second cache behavior does. Users will be able to access the files without using signed URLs or signed cookies because CloudFront processes the cache behavior that is associated with the first match.

  For more information about path patterns, see Path pattern.
- For a distribution that you're already using to distribute content, make sure you're ready to start generating signed URLs and signed cookies before you add a signer. When you add a signer, CloudFront rejects requests that don't include a valid signed URL or signed cookie.

You can add signers to your distribution using either the CloudFront console or the CloudFront API.

Topics

- Adding a signer to a distribution using the CloudFront console
- Adding a signer to a distribution using the CloudFront API

Adding a signer to a distribution using the CloudFront console

The following steps show how to add a trusted key group as a signer. You can also add an Amazon account as a trusted signer, but it's not recommended.

To add a signer to a distribution using the console

1. Record the key group ID of the key group that you want to use as a trusted signer. For more information, see Create a key pair for a trusted key group (recommended).
2. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
3. Choose the distribution whose files you want to protect with signed URLs or signed cookies.

   > **Note**
   >
   > To add a signer to a new distribution, you specify the same settings that are described in step 6 when you create the distribution.

4. Choose the **Behaviors** tab.
5. Select the cache behavior whose path pattern matches the files that you want to protect with signed URLs or signed cookies, and then choose **Edit**.
6. On the **Edit Behavior** page, do the following:
   a. For **Restrict Viewer Access (Use Signed URLs or Signed Cookies)**, choose **Yes**.
   b. For **Trusted Key Groups or Trusted Signer**, choose **Trusted Key Groups**.
   c. For **Trusted Key Groups**, choose the key group to add, and then choose **Add**. Repeat if you want to add more than one key group.
7. Choose **Yes, Edit** to update the cache behavior.

Adding a signer to a distribution using the CloudFront API

You can use the CloudFront API to add a trusted key group as a signer. You can add a signer to an existing distribution or to a new distribution. In either case, specify the values in the `TrustedKeyGroups` element.

You can also add an Amazon account as a trusted signer, but it's not recommended.

See the following topics in the *Amazon CloudFront API Reference*:

- **Update an existing distribution** – UpdateDistribution
- **Create a new distribution** – CreateDistribution

Rotating key pairs

We recommend that you periodically rotate (change) your key pairs for signed URLs and signed cookies. To rotate key pairs that you're using to create signed URLs or signed cookies without invalidating URLs or cookies that haven't expired yet, do the following tasks:

1. Create a new key pair, and add the public key to a key group. For more information, see Create a key pair for a trusted key group (recommended).
2. If you created a new key group in the previous step, add the key group to the distribution as a signer.

   > **Important**
   >
   > Don't remove any existing public keys from the key group, or any key groups from the distribution yet. Only add the new ones.

3. Update your application to create signatures using the private key from the new key pair. Confirm that the signed URLs or cookies that are signed with the new private keys are working.
4. Wait until the expiration date has passed in URLs or cookies that were signed using the previous private key. Then remove the old public key from the key group. If you created a new key group in step 2, remove the old key group from your distribution.

Choosing between signed URLs and signed cookies

CloudFront signed URLs and signed cookies provide the same basic functionality: they allow you to control who can access your content. If you want to serve private content through CloudFront and you're trying to decide whether to use signed URLs or signed cookies, consider the following.

Use signed URLs in the following cases:

- You want to restrict access to individual files, for example, an installation download for your application.
- Your users are using a client (for example, a custom HTTP client) that doesn't support cookies.

Use signed cookies in the following cases:

- You want to provide access to multiple restricted files, for example, all of the files for a video in HLS format or all of the files in the subscribers' area of website.
- You don't want to change your current URLs.

If you are not currently using signed URLs, and if your (unsigned) URLs contain any of the following query string parameters, you cannot use either signed URLs or signed cookies:

- `Expires`
- `Policy`
- `Signature`
- `Key-Pair-Id`

CloudFront assumes that URLs that contain any of those query string parameters are signed URLs, and therefore won't look at signed cookies.

Using both signed URLs and signed cookies

Signed URLs take precedence over signed cookies. If you use both signed URLs and signed cookies to control access to the same files and a viewer uses a signed URL to request a file, CloudFront determines whether to return the file to the viewer based only on the signed URL.

Using signed URLs

Topics

- Choosing between canned and custom policies for signed URLs
- How signed URLs work
- Choosing how long signed URLs are valid
- When does CloudFront check the expiration date and time in a signed URL?
- Example code and third-party tools

- Creating a signed URL using a canned policy
- Creating a signed URL using a custom policy

A signed URL includes additional information, for example, an expiration date and time, that gives you more control over access to your content. This additional information appears in a policy statement, which is based on either a canned policy or a custom policy. The differences between canned and custom policies are explained in the next two sections.

> **Note**
>
> You can create some signed URLs using canned policies and create some signed URLs using custom policies for the same distribution.

Choosing between canned and custom policies for signed URLs

When you create a signed URL, you write a policy statement in JSON format that specifies the restrictions on the signed URL, for example, how long the URL is valid. You can use either a canned policy or a custom policy. Here's how canned and custom policies compare:

Description	Canned policy	Custom policy
You can reuse the policy statement for multiple files. To reuse the policy statement, you must use wildcard characters in the `Resource` object. For more information, see Values that you specify in the policy statement for a signed URL that uses a custom policy.)	No	Yes
You can specify the date and time that users can begin to access your content.	No	Yes (optional)
You can specify the date and time that users can no longer access your content.	Yes	Yes
You can specify the IP address or range of IP addresses of the users who can access your content.	No	Yes (optional)

Description	Canned policy	Custom policy
The signed URL includes a base64-encoded version of the policy, which results in a longer URL.	No	Yes

For information about creating signed URLs using a *canned* policy, see Creating a signed URL using a canned policy.

For information about creating signed URLs using a *custom* policy, see Creating a signed URL using a custom policy.

How signed URLs work

Here's an overview of how you configure CloudFront and Amazon S3 for signed URLs and how CloudFront responds when a user uses a signed URL to request a file.

1. In your CloudFront distribution, specify one or more trusted key groups, which contain the public keys that CloudFront can use to verify the URL signature. You use the corresponding private keys to sign the URLs.

   For more information, see Specifying the signers that can create signed URLs and signed cookies.
2. Develop your application to determine whether a user should have access to your content and to create signed URLs for the files or parts of your application that you want to restrict access to. For more information, see the following topics:

   - Creating a signed URL using a canned policy
   - Creating a signed URL using a custom policy
3. A user requests a file for which you want to require signed URLs.
4. Your application verifies that the user is entitled to access the file: they've signed in, they've paid for access to the content, or they've met some other requirement for access.
5. Your application creates and returns a signed URL to the user.
6. The signed URL allows the user to download or stream the content.

   This step is automatic; the user usually doesn't have to do anything additional to access the content. For example, if a user is accessing your content in a web browser, your application

returns the signed URL to the browser. The browser immediately uses the signed URL to access the file in the CloudFront edge cache without any intervention from the user.

7. CloudFront uses the public key to validate the signature and confirm that the URL hasn't been tampered with. If the signature is invalid, the request is rejected.

   If the signature is valid, CloudFront looks at the policy statement in the URL (or constructs one if you're using a canned policy) to confirm that the request is still valid. For example, if you specified a beginning and ending date and time for the URL, CloudFront confirms that the user is trying to access your content during the time period that you want to allow access.

   If the request meets the requirements in the policy statement, CloudFront does the standard operations: determines whether the file is already in the edge cache, forwards the request to the origin if necessary, and returns the file to the user.

> **Note**
>
> If an unsigned URL contains query string parameters, make sure you include them in the portion of the URL that you sign. If you add a query string to a signed URL after signing it, the URL returns an HTTP 403 status.

Choosing how long signed URLs are valid

You can distribute private content using a signed URL that is valid for only a short time—possibly for as little as a few minutes. Signed URLs that are valid for such a short period are good for distributing content on-the-fly to a user for a specific purpose, such as distributing movie rentals or music downloads to customers on demand. If your signed URLs will be valid for just a short period, you'll probably want to generate them automatically using an application that you develop. When the user starts to download a file or starts to play a media file, CloudFront compares the expiration time in the URL with the current time to determine whether the URL is still valid.

You can also distribute private content using a signed URL that is valid for a longer time, possibly for years. Signed URLs that are valid for a longer period are useful for distributing private content to known users, such as distributing a business plan to investors or distributing training materials to employees. You can develop an application to generate these longer-term signed URLs for you.

When does CloudFront check the expiration date and time in a signed URL?

CloudFront checks the expiration date and time in a signed URL at the time of the HTTP request. If a client begins to download a large file immediately before the expiration time, the download should complete even if the expiration time passes during the download. If the TCP connection drops and the client tries to restart the download after the expiration time passes, the download will fail.

If a client uses Range GETs to get a file in smaller pieces, any GET request that occurs after the expiration time passes will fail. For more information about Range GETs, see How CloudFront processes partial requests for an object (range GETs).

Example code and third-party tools

For example code that creates the hashed and signed part of signed URLs, see the following topics:

- Create a URL signature using Perl
- Create a URL signature using PHP
- Create a URL signature using C# and the .NET Framework
- Create a URL signature using Java

Creating a signed URL using a canned policy

To create a signed URL using a canned policy, complete the following steps.

To create a signed URL using a canned policy

1. If you're using .NET or Java to create signed URLs, and if you haven't reformatted the private key for your key pair from the default .pem format to a format compatible with .NET or with Java, do so now. For more information, see Reformatting the private key (.NET and Java only).
2. Concatenate the following values in the specified order, and remove the white space (including tabs and newline characters) between the parts. You might have to include escape characters in the string in application code. All values have a type of String. Each part is keyed by number ( )
   to the two examples that follow.

1

Base URL for the file

The base URL is the CloudFront URL that you would use to access the file if you were not using signed URLs, including your own query string parameters, if any. For more information about the format of URLs for distributions, see Customizing the URL format for files in CloudFront.

- The following CloudFront URL is for an image file in a distribution (using the CloudFront domain name). Note that `image.jpg` is in an `images` directory. The path to the file in the URL must match the path to the file on your HTTP server or in your Amazon S3 bucket.

  ```
  https://d111111abcdef8.cloudfront.net/images/image.jpg
  ```

- The following CloudFront URL includes a query string:

  ```
  https://d111111abcdef8.cloudfront.net/images/image.jpg?size=large
  ```

- The following CloudFront URLs are for image files in a distribution. Both use an alternate domain name; the second one includes a query string:

  ```
  https://www.example.com/images/image.jpg
  ```

  ```
  https://www.example.com/images/image.jpg?color=red
  ```

- The following CloudFront URL is for an image file in a distribution that uses an alternate domain name and the HTTPS protocol:

  ```
  https://www.example.com/images/image.jpg
  ```

2

?

The **?** indicates that query string parameters follow the base URL. Include the **?** even if you don't have any query string parameters of your own.

3

***Your query string parameters, if any*&**

This value is optional. If you want to add your own query string parameters, for example:

```
color=red&size=medium
```

then add the parameters after the **?** (see

2)

and before the `Expires` parameter. In certain rare circumstances, you might need to put your query string parameters after `Key-Pair-Id`.

> **Important**
>
> Your parameters cannot be named `Expires`, `Signature`, or `Key-Pair-Id`.

If you add your own parameters, append an **&** after each one, including the last one.

4

Expires=*date and time in Unix time format (in seconds) and Coordinated Universal Time (UTC)*

The date and time that you want the URL to stop allowing access to the file.

Specify the expiration date and time in Unix time format (in seconds) and Coordinated Universal Time (UTC). For example, January 1, 2013 10:00 am UTC converts to 1357034400 in Unix time format. To use epoch time, use a 32-bit integer for a date that's no later than 2147483647 (January 19th, 2038 at 03:14:07 UTC). For information about UTC, see *RFC 3339, Date and Time on the Internet: Timestamps*, https://tools.ietf.org/html/rfc3339.

5

&Signature=*hashed and signed version of the policy statement*

A hashed, signed, and base64-encoded version of the JSON policy statement. For more information, see Creating a signature for a signed URL that uses a canned policy.

6

&Key-Pair-Id=*public key ID for the CloudFront public key whose corresponding private key you're using to generate the signature*

The ID for a CloudFront public key, for example, `K2JCJMDEHXQW5F`. The public key ID tells CloudFront which public key to use to validate the signed URL. CloudFront compares the information in the signature with the information in the policy statement to verify that the URL has not been tampered with.

This public key must belong to a key group that is a trusted signer in the distribution. For more information, see Specifying the signers that can create signed URLs and signed cookies.

Example signed URL:

```
1 https
d111111abcdef8.cloudfront.net/image.jpg
2
?
3
color=red&size=medium&
4
Expires=1357034400
5
&Signature=nitfHRCrtziwO2HwPfWw~yYDhUF5EwRunQA-j19DzZrvDh6hQ73lDx~-
ar3UocvvRQVw6EkC~GdpGQyyOSKQim-
TxAnW7d8F5Kkai9HVx0FIu-5jcQb0UEmatEXAMPLE3ReXySpLSMj0yCd3ZAB4UcBCAqEijkytL6f3fVY
6
&Key-Pair-Id=K2JCJMDEHXQW5F
```

Creating a signature for a signed URL that uses a canned policy

To create the signature for a signed URL that uses a canned policy, you do the following procedures:

1. Create a policy statement. See Creating a policy statement for a signed URL that uses a canned policy.
2. Sign the policy statement to create a signature. See Creating a signature for a signed URL that uses a canned policy.

Creating a policy statement for a signed URL that uses a canned policy

When you create a signed URL using a canned policy, the `Signature` parameter is a hashed and signed version of a policy statement. For signed URLs that use a canned policy, you don't include

the policy statement in the URL, as you do for signed URLs that use a custom policy. To create the policy statement, do the following procedure.

To create the policy statement for a signed URL that uses a canned policy

1. Construct the policy statement using the following JSON format and using UTF-8 character encoding. Include all punctuation and other literal values exactly as specified. For information about the `Resource` and `DateLessThan` parameters, see Values that you specify in the policy statement for a signed URL that uses a canned policy.

```
{
    "Statement": [
        {
            "Resource": "base URL or stream name",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime": ending date and time in Unix time format and
 UTC
                }
            }
        }
    ]
}
```

2. Remove all white space (including tabs and newline characters) from the policy statement. You might have to include escape characters in the string in application code.

Values that you specify in the policy statement for a signed URL that uses a canned policy

When you create a policy statement for a canned policy, you specify the following values.

Resource

> **Note**
>
> You can specify only one value for `Resource`.

The base URL including your query strings, if any, but excluding the CloudFront `Expires`, `Signature`, and `Key-Pair-Id` parameters, for example:

`https://d111111abcdef8.cloudfront.net/images/horizon.jpg?size=large&license=yes`

Note the following:

- **Protocol** – The value must begin with `http://` or `https://`.
- **Query string parameters** – If you have no query string parameters, omit the question mark.
- **Alternate domain names** – If you specify an alternate domain name (CNAME) in the URL, you must specify the alternate domain name when referencing the file in your webpage or application. Do not specify the Amazon S3 URL for the object.

DateLessThan

The expiration date and time for the URL in Unix time format (in seconds) and Coordinated Universal Time (UTC). For example, January 1, 2013 10:00 am UTC converts to 1357034400 in Unix time format.

This value must match the value of the `Expires` query string parameter in the signed URL. Do not enclose the value in quotation marks.

For more information, see When does CloudFront check the expiration date and time in a signed URL?.

Example policy statement for a signed URL that uses a canned policy

When you use the following example policy statement in a signed URL, a user can access the file `https://d111111abcdef8.cloudfront.net/horizon.jpg` until January 1, 2013 10:00 am UTC:

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/horizon.jpg?
size=large&license=yes",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime": 1357034400
                }
            }
        }
    ]
```

```
}
```

Creating a signature for a signed URL that uses a canned policy

To create the value for the `Signature` parameter in a signed URL, you hash and sign the policy statement that you created in Creating a policy statement for a signed URL that uses a canned policy.

For additional information and examples of how to hash, sign, and encode the policy statement, see:

- Using Linux commands and OpenSSL for base64 encoding and encryption
- Code examples for creating a signature for a signed URL

Option 1: To create a signature by using a canned policy

1. Use the SHA-1 hash function and RSA to hash and sign the policy statement that you created in the procedure To create the policy statement for a signed URL that uses a canned policy. Use the version of the policy statement that no longer includes white space.

   For the private key that is required by the hash function, use a private key whose public key is in an active trusted key group for the distribution.

   > **Note**
   >
   > The method that you use to hash and sign the policy statement depends on your programming language and platform. For sample code, see Code examples for creating a signature for a signed URL.

2. Remove white space (including tabs and newline characters) from the hashed and signed string.
3. Base64-encode the string using MIME base64 encoding. For more information, see Section 6.8, Base64 Content-Transfer-Encoding in *RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies*.
4. Replace characters that are invalid in a URL query string with characters that are valid. The following table lists invalid and valid characters.

Replace these invalid characters	With these valid characters
+	- (hyphen)
=	_ (underscore)
/	~ (tilde)

5. Append the resulting value to your signed URL after `&Signature=`, and return to To create a signed URL using a canned policy to finish concatenating the parts of your signed URL.

Creating a signed URL using a custom policy

Topics

- Creating a policy statement for a signed URL that uses a custom policy
- Example policy statements for a signed URL that uses a custom policy
- Creating a signature for a signed URL that uses a custom policy

To create a signed URL using a custom policy, do the following procedure.

To create a signed URL using a custom policy

1. If you're using .NET or Java to create signed URLs, and if you haven't reformatted the private key for your key pair from the default .pem format to a format compatible with .NET or with Java, do so now. For more information, see Reformatting the private key (.NET and Java only).
2. Concatenate the following values in the specified order, and remove the white space (including tabs and newline characters) between the parts. You might have to include escape characters in the string in application code. All values have a type of String. Each part is keyed by number ( ) to the two examples that follow.

   Base URL for the file

   The base URL is the CloudFront URL that you would use to access the file if you were not using signed URLs, including your own query string parameters, if any. For more

information about the format of URLs for distributions, see Customizing the URL format for files in CloudFront.

The following examples show values that you specify for distributions.

- The following CloudFront URL is for an image file in a distribution (using the CloudFront domain name). Note that `image.jpg` is in an `images` directory. The path to the file in the URL must match the path to the file on your HTTP server or in your Amazon S3 bucket.

  ```
  https://d111111abcdef8.cloudfront.net/images/image.jpg
  ```

- The following CloudFront URL includes a query string:

  ```
  https://d111111abcdef8.cloudfront.net/images/image.jpg?size=large
  ```

- The following CloudFront URLs are for image files in a distribution. Both use an alternate domain name; the second one includes a query string:

  ```
  https://www.example.com/images/image.jpg
  ```

  ```
  https://www.example.com/images/image.jpg?color=red
  ```

- The following CloudFront URL is for an image file in a distribution that uses an alternate domain name and the HTTPS protocol:

  ```
  https://www.example.com/images/image.jpg
  ```

2

?

The **?** indicates that query string parameters follow the base URL. Include the **?** even if you don't have any query string parameters of your own.

3

***Your query string parameters, if any*&**

This value is optional. If you want to add your own query string parameters, for example:

```
color=red&size=medium
```

then add them after the **?** (see

2

)

and before the `Policy` parameter. In certain rare circumstances, you might need to put your query string parameters after `Key-Pair-Id`.

> **Important**
>
> Your parameters cannot be named `Policy`, `Signature`, or `Key-Pair-Id`.

If you add your own parameters, append an **&** after each one, including the last one.

4

`Policy=`*`base64 encoded version of policy statement`*

Your policy statement in JSON format, with white space removed, then base64 encoded. For more information, see Creating a policy statement for a signed URL that uses a custom policy.

The policy statement controls the access that a signed URL grants to a user. It includes the URL of the file, an expiration date and time, an optional date and time that the URL becomes valid, and an optional IP address or range of IP addresses that are allowed to access the file.

5

`&Signature=`*`hashed and signed version of the policy statement`*

A hashed, signed, and base64-encoded version of the JSON policy statement. For more information, see Creating a signature for a signed URL that uses a custom policy.

6

`&Key-Pair-Id=`*`public key ID for the CloudFront public key whose corresponding private key you're using to generate the signature`*

The ID for a CloudFront public key, for example, `K2JCJMDEHXQW5F`. The public key ID tells CloudFront which public key to use to validate the signed URL. CloudFront compares the information in the signature with the information in the policy statement to verify that the URL has not been tampered with.

This public key must belong to a key group that is a trusted signer in the distribution. For more information, see Specifying the signers that can create signed URLs and signed cookies.

Example signed URL:

```
(1) https
d111111abcdef8.cloudfront.net/image.jpg
(2) ?
(3) color
(4) Polic
6Ly9kemJlc3FtN3VuMW0wLmNsb3VkZnJvbnQubmV0L2RlbW8ucGhwIiwgDQogICAgICAiQ
29uZGl0aW9uIjp7IA0KICAgICAgICAgIklwQWRkcmVzcyI6eyJBV1M6U291cmNlSXAiOiI
yMDcuMTcxLjE4MC4xMDEvMzIifSwNCiAgICAgICAgICJEYXRlR3JlYXRlclRoYW4iOnsiQ
VdTOkVwb2NoVGltZSI6MTI5Njg2MDE3Nn0sDQogICAgICAgICAiRGF0ZUxlc3NUaGFuIjp
7IkFXUzpFcG9jaFRpbWUiOjEyOTY4NjAyMjZ9DQogICAgICB9IA0KICAgfV0gDQp9DQo
(5) &Sign
j19DzZrvDh6hQ73lDx~ -ar3UocvvRQVw6EkC~GdpGQyyOSKQim-
TxAnW7d8F5Kkai9HVx0FIu-5jcQb0UEmat
EXAMPLE3ReXySpLSMj0yCd3ZAB4UcBCAqEijkytL6f3fVYNGQI6
(6) &Key-
Pair-Id=K2JCJMDEHXQW5F
```

Creating a policy statement for a signed URL that uses a custom policy

Complete the following steps to create a policy statement for a signed URL that uses a custom policy.

For example policy statements that control access to files in a variety of ways, see the section called “Example policy statements for a signed URL that uses a custom policy”.

To create the policy statement for a signed URL that uses a custom policy

1. Construct the policy statement using the following JSON format. Replace the less than (<) and greater than (>) symbols, and the descriptions within them, with your own values. For more information, see the section called “Values that you specify in the policy statement for a signed URL that uses a custom policy”.

   ```
   {
       "Statement": [
           {
               "Resource": "<Optional but recommended: URL of the file>",
               "Condition": {
   ```

```
                "DateLessThan": {
                 "AWS:EpochTime": <Required: ending date and time in Unix time
 format and UTC>
                },
                "DateGreaterThan": {
                 "AWS:EpochTime": <Optional: beginning date and time in Unix time
 format and UTC>
                },
                "IpAddress": {
                 "AWS:SourceIp": "<Optional: IP address>"
                }
            }
        }
    ]
}
```

Note the following:

- You can include only one statement in the policy.
- Use UTF-8 character encoding.
- Include all punctuation and parameter names exactly as specified. Abbreviations for parameter names are not accepted.
- The order of the parameters in the `Condition` section doesn't matter.
- For information about the values for `Resource`, `DateLessThan`, `DateGreaterThan`, and `IpAddress`, see the section called "Values that you specify in the policy statement for a signed URL that uses a custom policy".

2. Remove all white space (including tabs and newline characters) from the policy statement. You might have to include escape characters in the string in application code.
3. Base64-encode the policy statement using MIME base64 encoding. For more information, see Section 6.8, Base64 Content-Transfer-Encoding in *RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies*.
4. Replace characters that are invalid in a URL query string with characters that are valid. The following table lists invalid and valid characters.

Replace these invalid characters	With these valid characters
+	- (hyphen)

Replace these invalid characters	With these valid characters
=	_ (underscore)
/	~ (tilde)

5. Append the resulting value to your signed URL after `Policy=`.
6. Create a signature for the signed URL by hashing, signing, and base64-encoding the policy statement. For more information, see the section called "Creating a signature for a signed URL that uses a custom policy".

Values that you specify in the policy statement for a signed URL that uses a custom policy

When you create a policy statement for a custom policy, you specify the following values.

Resource

The URL, including any query strings, but excluding the CloudFront `Policy`, `Signature`, and `Key-Pair-Id` parameters. For example:

```
https://d111111abcdef8.cloudfront.net/images/horizon.jpg\?
size=large&license=yes
```

You can specify only one URL value for `Resource`.

> **Important**
>
> You can omit the `Resource` parameter in a policy, but doing so means that anyone with the signed URL can access *all* of the files in *any* distribution that is associated with the key pair that you use to create the signed URL.

Note the following:

- **Protocol** – The value must begin with `http://`, `https://`, or `*://`.
- **Query string parameters** – If the URL has query string parameters, use a backslash character (\) to escape the question mark character (?) that begins the query string. For example:

  ```
  https://d111111abcdef8.cloudfront.net/images/horizon.jpg\?
  size=large&license=yes
  ```

- **Wildcard characters** – You can use wildcard characters in the URL in the policy. The following wildcard characters are supported:
  - asterisk (*), which matches zero or more characters
  - question mark (?), which matches exactly one character

  When CloudFront matches the URL in the policy to the URL in the HTTP request, the URL in the policy is divided into four sections—protocol, domain, path, and query string—as follows:

  ```
  [protocol]://[domain]/[path]\?[query string]
  ```

  When you use a wildcard character in the URL in the policy, the wildcard matching applies only within the boundaries of the section that contains the wildcard. For example, consider this URL in a policy:

  ```
  https://www.example.com/hello*world
  ```

  In this example, the asterisk wildcard (*) only applies within the path section, so it matches the URLs `https://www.example.com/helloworld` and `https://www.example.com/hello-world`, but it does not match the URL `https://www.example.net/hello?world`.

  The following exceptions apply to the section boundaries for wildcard matching:

  - A trailing asterisk in the path section implies an asterisk in the query string section. For example, `http://example.com/hello*` is equivalent to `http://example.com/hello*\?*`.
  - A trailing asterisk in the domain section implies an asterisk in both the path and query string sections. For example, `http://example.com*` is equivalent to `http://example.com*/*\?*`.
  - A URL in the policy can omit the protocol section and start with an asterisk in the domain section. In that case, the protocol section is implicitly set to an asterisk. For example, the URL `*example.com` in a policy is equivalent to `*://*example.com/`.
  - An asterisk by itself (`"Resource": "*"`) matches any URL.

  For example, the value: `https://d111111abcdef8.cloudfront.net/*game_download.zip*` in a policy matches all of the following URLs:

  - `https://d111111abcdef8.cloudfront.net/game_download.zip`
  - `https://d111111abcdef8.cloudfront.net/example_game_download.zip?license=yes`

- `https://d111111abcdef8.cloudfront.net/test_game_download.zip?license=temp`

- **Alternate domain names** – If you specify an alternate domain name (CNAME) in the URL in the policy, the HTTP request must use the alternate domain name in your webpage or application. Do not specify the Amazon S3 URL for the file in a policy.

DateLessThan

The expiration date and time for the URL in Unix time format (in seconds) and Coordinated Universal Time (UTC). In the policy, do not enclose the value in quotation marks. For information about UTC, see Date and Time on the Internet: Timestamps.

For example, January 31, 2023 10:00 AM UTC converts to 1675159200 in Unix time format.

This is the only required parameter in the `Condition` section. CloudFront requires this value to prevent users from having permanent access to your private content.

For more information, see the section called "When does CloudFront check the expiration date and time in a signed URL?"

DateGreaterThan (Optional)

An optional start date and time for the URL in Unix time format (in seconds) and Coordinated Universal Time (UTC). Users are not allowed to access the file on or before the specified date and time. Do not enclose the value in quotation marks.

IpAddress (Optional)

The IP address of the client making the HTTP request. Note the following:

- To allow any IP address to access the file, omit the `IpAddress` parameter.
- You can specify either one IP address or one IP address range. You can't use the policy to allow access if the client's IP address is in one of two separate ranges.
- To allow access from a single IP address, you specify:

  `"IPv4 IP address/32"`

- You must specify IP address ranges in standard IPv4 CIDR format (for example, `192.0.2.0/24`). For more information, see Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan.

> **Important**
>
> IP addresses in IPv6 format, such as 2001:0db8:85a3::8a2e:0370:7334, are not supported.

If you're using a custom policy that includes `IpAddress`, do not enable IPv6 for the distribution. If you want to restrict access to some content by IP address and support IPv6 requests for other content, you can create two distributions. For more information, see the section called "Enable IPv6" in the topic the section called "Values that you specify".

Example policy statements for a signed URL that uses a custom policy

The following example policy statements show how to control access to a specific file, all of the files in a directory, or all of the files associated with a key pair ID. The examples also show how to control access from an individual IP address or a range of IP addresses, and how to prevent users from using the signed URL after a specified date and time.

If you copy and paste any of these examples, remove any white space (including tabs and newline characters), replace the values with your own values, and include a newline character after the closing brace (}).

For more information, see the section called "Values that you specify in the policy statement for a signed URL that uses a custom policy".

Topics

- Example policy statement: accessing one file from a range of IP addresses
- Example policy statement: accessing all files in a directory from a range of IP addresses
- Example policy statement: accessing all files associated with a key pair ID from one IP address

Example policy statement: accessing one file from a range of IP addresses

The following example custom policy in a signed URL specifies that a user can access the file `https://d111111abcdef8.cloudfront.net/game_download.zip` from IP addresses in the range `192.0.2.0/24` until January 31, 2023 10:00 AM UTC:

```
{
```

```
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/game_download.zip",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.0/24"
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1675159200
                }
            }
        }
    ]
}
```

Example policy statement: accessing all files in a directory from a range of IP addresses

The following example custom policy allows you to create signed URLs for any file in the `training` directory, as indicated by the asterisk wildcard character (*) in the `Resource` parameter. Users can access the file from an IP address in the range `192.0.2.0/24` until January 31, 2023 10:00 AM UTC:

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/training/*",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.0/24"
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1675159200
                }
            }
        }
    ]
}
```

Each signed URL with which you use this policy has a URL that identifies a specific file, for example:

`https://d111111abcdef8.cloudfront.net/training/orientation.pdf`

Example policy statement: accessing all files associated with a key pair ID from one IP address

The following example custom policy allows you to create signed URLs for any file associated with any distribution, as indicated by the asterisk wildcard character (*) in the `Resource` parameter. The signed URL must use the `https://` protocol, not `http://`. The user must use the IP address `192.0.2.10/32`. (The value `192.0.2.10/32` in CIDR notation refers to a single IP address, `192.0.2.10`.) The files are available only from January 31, 2023 10:00 AM UTC until February 2, 2023 10:00 AM UTC:

```
{
    "Statement": [
       {
            "Resource": "https://*",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.10/32"
                },
                "DateGreaterThan": {
                    "AWS:EpochTime": 1675159200
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1675332000
                }
            }
        }
    ]
}
```

Each signed URL with which you use this policy has a URL that identifies a specific file in a specific CloudFront distribution, for example:

`https://d111111abcdef8.cloudfront.net/training/orientation.pdf`

The signed URL also includes a key pair ID, which must be associated with a trusted key group in the distribution (d111111abcdef8.cloudfront.net) that you specify in the URL.

Creating a signature for a signed URL that uses a custom policy

The signature for a signed URL that uses a custom policy is a hashed, signed, and base64-encoded version of the policy statement. To create a signature for a custom policy, complete the following steps.

For additional information and examples of how to hash, sign, and encode the policy statement, see:

- Using Linux commands and OpenSSL for base64 encoding and encryption
- Code examples for creating a signature for a signed URL

Option 1: To create a signature by using a custom policy

1. Use the SHA-1 hash function and RSA to hash and sign the JSON policy statement that you created in the procedure To create the policy statement for a signed URL that uses a custom policy. Use the version of the policy statement that no longer includes white space but that has not yet been base64-encoded.

   For the private key that is required by the hash function, use a private key whose public key is in an active trusted key group for the distribution.

   > **Note**
   >
   > The method that you use to hash and sign the policy statement depends on your programming language and platform. For sample code, see Code examples for creating a signature for a signed URL.

2. Remove white space (including tabs and newline characters) from the hashed and signed string.
3. Base64-encode the string using MIME base64 encoding. For more information, see Section 6.8, Base64 Content-Transfer-Encoding in *RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies*.
4. Replace characters that are invalid in a URL query string with characters that are valid. The following table lists invalid and valid characters.

Replace these invalid characters	With these valid characters
+	- (hyphen)
=	_ (underscore)
/	~ (tilde)

5. Append the resulting value to your signed URL after `&Signature=`, and return to To create a signed URL using a custom policy to finish concatenating the parts of your signed URL.

Using signed cookies

CloudFront signed cookies allow you to control who can access your content when you don't want to change your current URLs or when you want to provide access to multiple restricted files, for example, all of the files in the subscribers' area of a website. This topic explains the considerations when using signed cookies and describes how to set signed cookies using canned and custom policies.

Topics

- Choosing between canned and custom policies for signed cookies
- How signed cookies work
- Preventing misuse of signed cookies
- When does CloudFront check the expiration date and time in a signed cookie?
- Sample code and third-party tools
- Setting signed cookies using a canned policy
- Setting signed cookies using a custom policy

Choosing between canned and custom policies for signed cookies

When you create a signed cookie, you write a policy statement in JSON format that specifies the restrictions on the signed cookie, for example, how long the cookie is valid. You can use canned policies or custom policies. The following table compares canned and custom policies:

Description	Canned policy	Custom policy
You can reuse the policy statement for multiple files. To reuse the policy statement, you must use wildcard characters in the `Resource` object. For more information, see Values that you specify in the policy statement for a custom policy for signed cookies.)	No	Yes

Description	Canned policy	Custom policy
You can specify the date and time that users can begin to access your content	No	Yes (optional)
You can specify the date and time that users can no longer access your content	Yes	Yes
You can specify the IP address or range of IP addresses of the users who can access your content	No	Yes (optional)

For information about creating signed cookies using a canned policy, see Setting signed cookies using a canned policy.

For information about creating signed cookies using a custom policy, see Setting signed cookies using a custom policy.

How signed cookies work

Here's an overview of how you configure CloudFront for signed cookies and how CloudFront responds when a user submits a request that contains a signed cookie.

1. In your CloudFront distribution, specify one or more trusted key groups, which contain the public keys that CloudFront can use to verify the URL signature. You use the corresponding private keys to sign the URLs.

   For more information, see Specifying the signers that can create signed URLs and signed cookies.
2. You develop your application to determine whether a user should have access to your content and, if so, to send three `Set-Cookie` headers to the viewer. (Each `Set-Cookie` header can contain only one name-value pair, and a CloudFront signed cookie requires three name-value pairs.) You must send the `Set-Cookie` headers to the viewer before the viewer requests your private content. If you set a short expiration time on the cookie, you might also want to send three more `Set-Cookie` headers in response to subsequent requests, so that the user continues to have access.

Typically, your CloudFront distribution will have at least two cache behaviors, one that doesn't require authentication and one that does. The error page for the secure portion of the site includes a redirector or a link to a login page.

If you configure your distribution to cache files based on cookies, CloudFront doesn't cache separate files based on the attributes in signed cookies.

3. A user signs in to your website and either pays for content or meets some other requirement for access.
4. Your application returns the `Set-Cookie` headers in the response, and the viewer stores the name-value pairs.
5. The user requests a file.

   The user's browser or other viewer gets the name-value pairs from step 4 and adds them to the request in a `Cookie` header. This is the signed cookie.
6. CloudFront uses the public key to validate the signature in the signed cookie and to confirm that the cookie hasn't been tampered with. If the signature is invalid, the request is rejected.

   If the signature in the cookie is valid, CloudFront looks at the policy statement in the cookie (or constructs one if you're using a canned policy) to confirm that the request is still valid. For example, if you specified a beginning and ending date and time for the cookie, CloudFront confirms that the user is trying to access your content during the time period that you want to allow access.

   If the request meets the requirements in the policy statement, CloudFront serves your content as it does for content that isn't restricted: it determines whether the file is already in the edge cache, forwards the request to the origin if necessary, and returns the file to the user.

Preventing misuse of signed cookies

If you specify the `Domain` parameter in a `Set-Cookie` header, specify the most precise value possible to reduce the potential for access by someone with the same root domain name. For example, app.example.com is preferable to example.com, especially when you don't control example.com. This helps prevent someone from accessing your content from www.example.com.

To help prevent this type of attack, do the following:

- Exclude the `Expires` and `Max-Age` cookie attributes, so that the `Set-Cookie` header creates a session cookie. Session cookies are automatically deleted when the user closes the browser, which reduces the possibility of someone getting unauthorized access to your content.
- Include the `Secure` attribute, so that the cookie is encrypted when a viewer includes it in a request.
- When possible, use a custom policy and include the IP address of the viewer.
- In the `CloudFront-Expires` attribute, specify the shortest reasonable expiration time based on how long you want users to have access to your content.

When does CloudFront check the expiration date and time in a signed cookie?

To determine whether a signed cookie is still valid, CloudFront checks the expiration date and time in the cookie at the time of the HTTP request. If a client begins to download a large file immediately before the expiration time, the download should complete even if the expiration time passes during the download. If the TCP connection drops and the client tries to restart the download after the expiration time passes, the download will fail.

If a client uses Range GETs to get a file in smaller pieces, any GET request that occurs after the expiration time passes will fail. For more information about Range GETs, see How CloudFront processes partial requests for an object (range GETs).

Sample code and third-party tools

The sample code for private content shows only how to create the signature for signed URLs. However, the process for creating a signature for a signed cookie is very similar, so much of the sample code is still relevant. For more information, see the following topics:

- Create a URL signature using Perl
- Create a URL signature using PHP
- Create a URL signature using C# and the .NET Framework
- Create a URL signature using Java

Setting signed cookies using a canned policy

To set a signed cookie by using a canned policy, complete the following steps. To create the signature, see Creating a signature for a signed cookie that uses a canned policy.

To set a signed cookie using a canned policy

1. If you're using .NET or Java to create signed cookies, and if you haven't reformatted the private key for your key pair from the default .pem format to a format compatible with .NET or with Java, do so now. For more information, see Reformatting the private key (.NET and Java only).
2. Program your application to send three `Set-Cookie` headers to approved viewers. You need three `Set-Cookie` headers because each `Set-Cookie` header can contain only one name-value pair, and a CloudFront signed cookie requires three name-value pairs. The name-value pairs are: `CloudFront-Expires`, `CloudFront-Signature`, and `CloudFront-Key-Pair-Id`. The values must be present on the viewer before a user makes the first request for a file that you want to control access to.

   > **Note**
   >
   > In general, we recommend that you exclude `Expires` and `Max-Age` attributes. Excluding the attributes causes the browser to delete the cookie when the user closes the browser, which reduces the possibility of someone getting unauthorized access to your content. For more information, see Preventing misuse of signed cookies.

   The names of cookie attributes are case-sensitive.

   Line breaks are included only to make the attributes more readable.

```
Set-Cookie:
CloudFront-Expires=date and time in Unix time format (in seconds) and Coordinated
 Universal Time (UTC);
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly

Set-Cookie:
CloudFront-Signature=hashed and signed version of the policy statement;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly

Set-Cookie:
```

```
CloudFront-Key-Pair-Id=public key ID for the CloudFront public key whose
 corresponding private key you're using to generate the signature;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly
```

(Optional) `Domain`

The domain name for the requested file. If you don't specify a `Domain` attribute, the default value is the domain name in the URL, and it applies only to the specified domain name, not to subdomains. If you specify a `Domain` attribute, it also applies to subdomains. A leading dot in the domain name (for example, `Domain=.example.com`) is optional. In addition, if you specify a `Domain` attribute, the domain name in the URL and the value of the `Domain` attribute must match.

You can specify the domain name that CloudFront assigned to your distribution, for example, d111111abcdef8.cloudfront.net, but you can't specify *.cloudfront.net for the domain name.

If you want to use an alternate domain name such as example.com in URLs, you must add the alternate domain name to your distribution regardless of whether you specify the `Domain` attribute. For more information, see Alternate domain names (CNAMEs) in the topic Values that you specify when you create or update a distribution.

(Optional) `Path`

The path for the requested file. If you don't specify a `Path` attribute, the default value is the path in the URL.

`Secure`

Requires that the viewer encrypt cookies before sending a request. We recommend that you send the `Set-Cookie` header over an HTTPS connection to ensure that the cookie attributes are protected from man-in-the-middle attacks.

`HttpOnly`

Requires that the viewer send the cookie only in HTTP or HTTPS requests.

`CloudFront-Expires`

Specify the expiration date and time in Unix time format (in seconds) and Coordinated Universal Time (UTC). For example, January 1, 2013 10:00 am UTC converts to 1357034400 in Unix time format. To use epoch time, use a 32-bit integer for a date that's no later than 2147483647 (January 19th, 2038 at 03:14:07 UTC). For information about UTC, see *RFC 3339, Date and Time on the Internet: Timestamps*, https://tools.ietf.org/html/rfc3339.

`CloudFront-Signature`

A hashed, signed, and base64-encoded version of a JSON policy statement. For more information, see Creating a signature for a signed cookie that uses a canned policy.

`CloudFront-Key-Pair-Id`

The ID for a CloudFront public key, for example, `K2JCJMDEHXQW5F`. The public key ID tells CloudFront which public key to use to validate the signed URL. CloudFront compares the information in the signature with the information in the policy statement to verify that the URL has not been tampered with.

This public key must belong to a key group that is a trusted signer in the distribution. For more information, see Specifying the signers that can create signed URLs and signed cookies.

The following example shows `Set-Cookie` headers for one signed cookie when you're using the domain name that is associated with your distribution in the URLs for your files:

```
Set-Cookie: CloudFront-Expires=1426500000; Domain=d111111abcdef8.cloudfront.net; Path=/
images/*; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=yXrSIgyQoeE4FBI4eMKF6ho~CA8_;
 Domain=d111111abcdef8.cloudfront.net; Path=/images/*; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F;
 Domain=d111111abcdef8.cloudfront.net; Path=/images/*; Secure; HttpOnly
```

The following example shows `Set-Cookie` headers for one signed cookie when you're using the alternate domain name example.org in the URLs for your files:

```
Set-Cookie: CloudFront-Expires=1426500000; Domain=example.org; Path=/images/*; Secure;
 HttpOnly
Set-Cookie: CloudFront-Signature=yXrSIgyQoeE4FBI4eMKF6ho~CA8_; Domain=example.org;
 Path=/images/*; Secure; HttpOnly
```

```
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F; Domain=example.org; Path=/images/*;
 Secure; HttpOnly
```

If you want to use an alternate domain name such as example.com in URLs, you must add the alternate domain name to your distribution regardless of whether you specify the `Domain` attribute. For more information, see Alternate domain names (CNAMEs) in the topic Values that you specify when you create or update a distribution.

Creating a signature for a signed cookie that uses a canned policy

To create the signature for a signed cookie that uses a canned policy, do the following:

1. Create a policy statement. See Creating a policy statement for a signed cookie that uses a canned policy.
2. Sign the policy statement to create a signature. See Signing the policy statement to create a signature for a signed cookie that uses a canned policy.

Creating a policy statement for a signed cookie that uses a canned policy

When you set a signed cookie that uses a canned policy, the `CloudFront-Signature` attribute is a hashed and signed version of a policy statement. For signed cookies that use a canned policy, you don't include the policy statement in the `Set-Cookie` header, as you do for signed cookies that use a custom policy. To create the policy statement, complete the following steps.

To create a policy statement for a signed cookie that uses a canned policy

1. Construct the policy statement using the following JSON format and using UTF-8 character encoding. Include all punctuation and other literal values exactly as specified. For information about the `Resource` and `DateLessThan` parameters, see Values that you specify in the policy statement for a canned policy for signed cookies.

```
{
    "Statement": [
        {
            "Resource": "base URL or stream name",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime": ending date and time in Unix time format and
 UTC
                }
```

```
            }
        }
    ]
}
```

2. Remove all white space (including tabs and newline characters) from the policy statement. You might have to include escape characters in the string in application code.

Values that you specify in the policy statement for a canned policy for signed cookies

When you create a policy statement for a canned policy, you specify the following values:

Resource

The base URL including your query strings, if any, for example:

`https://d111111abcdef8.cloudfront.net/images/horizon.jpg?size=large&license=yes`

You can specify only one value for `Resource`.

Note the following:

- **Protocol** – The value must begin with `http://` or `https://`.
- **Query string parameters** – If you have no query string parameters, omit the question mark.
- **Alternate domain names** – If you specify an alternate domain name (CNAME) in the URL, you must specify the alternate domain name when referencing the file in your webpage or application. Do not specify the Amazon S3 URL for the file.

DateLessThan

The expiration date and time for the URL in Unix time format (in seconds) and Coordinated Universal Time (UTC). Do not enclose the value in quotation marks.

For example, March 16, 2015 10:00 am UTC converts to 1426500000 in Unix time format.

This value must match the value of the `CloudFront-Expires` attribute in the `Set-Cookie` header. Do not enclose the value in quotation marks.

For more information, see When does CloudFront check the expiration date and time in a signed cookie?.

Example policy statement for a canned policy

When you use the following example policy statement in a signed cookie, a user can access the file `https://d111111abcdef8.cloudfront.net/horizon.jpg` until March 16, 2015 10:00 am UTC:

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/horizon.jpg?
size=large&license=yes",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime": 1426500000
                }
            }
        }
    ]
}
```

Signing the policy statement to create a signature for a signed cookie that uses a canned policy

To create the value for the `CloudFront-Signature` attribute in a `Set-Cookie` header, you hash and sign the policy statement that you created in To create a policy statement for a signed cookie that uses a canned policy.

For additional information and examples of how to hash, sign, and encode the policy statement, see the following topics:

- Using Linux commands and OpenSSL for base64 encoding and encryption
- Code examples for creating a signature for a signed URL

To create a signature for a signed cookie using a canned policy

1. Use the SHA-1 hash function and RSA to hash and sign the policy statement that you created in the procedure To create a policy statement for a signed cookie that uses a canned policy. Use the version of the policy statement that no longer includes white space.

   For the private key that is required by the hash function, use a private key whose public key is in an active trusted key group for the distribution.

> **Note**
>
> The method that you use to hash and sign the policy statement depends on your programming language and platform. For sample code, see Code examples for creating a signature for a signed URL.

2. Remove white space (including tabs and newline characters) from the hashed and signed string.
3. Base64-encode the string using MIME base64 encoding. For more information, see Section 6.8, Base64 Content-Transfer-Encoding in *RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies*.
4. Replace characters that are invalid in a URL query string with characters that are valid. The following table lists invalid and valid characters.

Replace these invalid characters	With these valid characters
+	- (hyphen)
=	_ (underscore)
/	~ (tilde)

5. Include the resulting value in the `Set-Cookie` header for the `CloudFront-Signature` name-value pair. Then return to To set a signed cookie using a canned policy add the `Set-Cookie` header for `CloudFront-Key-Pair-Id`.

Setting signed cookies using a custom policy

Topics

- Example Set-Cookie headers for custom policies
- Creating a policy statement for a signed cookie that uses a custom policy
- Example policy statements for a signed cookie that uses a custom policy
- Creating a signature for a signed cookie that uses a custom policy

To set a signed cookie that uses a custom policy, complete the following steps.

To set a signed cookie using a custom policy

1. If you're using .NET or Java to create signed URLs, and if you haven't reformatted the private key for your key pair from the default .pem format to a format compatible with .NET or with Java, do so now. For more information, see Reformatting the private key (.NET and Java only).
2. Program your application to send three `Set-Cookie` headers to approved viewers. You need three `Set-Cookie` headers because each `Set-Cookie` header can contain only one name-value pair, and a CloudFront signed cookie requires three name-value pairs. The name-value pairs are: `CloudFront-Policy`, `CloudFront-Signature`, and `CloudFront-Key-Pair-Id`. The values must be present on the viewer before a user makes the first request for a file that you want to control access to.

   > **Note**
   >
   > In general, we recommend that you exclude `Expires` and `Max-Age` attributes. This causes the browser to delete the cookie when the user closes the browser, which reduces the possibility of someone getting unauthorized access to your content. For more information, see Preventing misuse of signed cookies.

   The names of cookie attributes are case-sensitive.

   Line breaks are included only to make the attributes more readable.

```
Set-Cookie:
CloudFront-Policy=base64 encoded version of the policy statement;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly


Set-Cookie:
CloudFront-Signature=hashed and signed version of the policy statement;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly

Set-Cookie:
```

```
CloudFront-Key-Pair-Id=public key ID for the CloudFront public key whose
 corresponding private key you're using to generate the signature;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly
```

(Optional) `Domain`

The domain name for the requested file. If you don't specify a `Domain` attribute, the default value is the domain name in the URL, and it applies only to the specified domain name, not to subdomains. If you specify a `Domain` attribute, it also applies to subdomains. A leading dot in the domain name (for example, `Domain=.example.com`) is optional. In addition, if you specify a `Domain` attribute, the domain name in the URL and the value of the `Domain` attribute must match.

You can specify the domain name that CloudFront assigned to your distribution, for example, d111111abcdef8.cloudfront.net, but you can't specify *.cloudfront.net for the domain name.

If you want to use an alternate domain name such as example.com in URLs, you must add the alternate domain name to your distribution regardless of whether you specify the `Domain` attribute. For more information, see Alternate domain names (CNAMEs) in the topic Values that you specify when you create or update a distribution.

(Optional) `Path`

The path for the requested file. If you don't specify a `Path` attribute, the default value is the path in the URL.

`Secure`

Requires that the viewer encrypt cookies before sending a request. We recommend that you send the `Set-Cookie` header over an HTTPS connection to ensure that the cookie attributes are protected from man-in-the-middle attacks.

`HttpOnly`

Requires that the viewer send the cookie only in HTTP or HTTPS requests.

`CloudFront-Policy`

Your policy statement in JSON format, with white space removed, then base64 encoded. For more information, see Creating a signature for a signed cookie that uses a custom policy.

The policy statement controls the access that a signed cookie grants to a user. It includes the files that the user can access, an expiration date and time, an optional date and time that the URL becomes valid, and an optional IP address or range of IP addresses that are allowed to access the file.

`CloudFront-Signature`

A hashed, signed, and base64-encoded version of the JSON policy statement. For more information, see Creating a signature for a signed cookie that uses a custom policy.

`CloudFront-Key-Pair-Id`

The ID for a CloudFront public key, for example, `K2JCJMDEHXQW5F`. The public key ID tells CloudFront which public key to use to validate the signed URL. CloudFront compares the information in the signature with the information in the policy statement to verify that the URL has not been tampered with.

This public key must belong to a key group that is a trusted signer in the distribution. For more information, see Specifying the signers that can create signed URLs and signed cookies.

Example `Set-Cookie` headers for custom policies

See the following examples of `Set-Cookie` header pairs.

If you want to use an alternate domain name such as example.org in URLs, you must add the alternate domain name to your distribution regardless of whether you specify the `Domain` attribute. For more information, see Alternate domain names (CNAMEs) in the topic Values that you specify when you create or update a distribution.

Example Example 1

You can use the `Set-Cookie` headers for one signed cookie when you're using the domain name that is associated with your distribution in the URLs for your files.

```
Set-Cookie: CloudFront-
Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2QxMTExMTFhYmNkZWY4LmNsb3VkZnJvbnQubmV0L2dh
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_;
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F;
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
```

Example Example 2

You can use the `Set-Cookie` headers for one signed cookie when you're using an alternate domain name (example.org) in the URLs for your files.

```
Set-Cookie: CloudFront-
Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2QxMTExMTFhYmNkZWY4LmNsb3VkZnJvbnQubmV0L2dh
 Domain=example.org; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_; Domain=example.org;
 Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F; Domain=example.org; Path=/; Secure;
 HttpOnly
```

Example Example 3

You can use the `Set-Cookie` header pairs for a signed request when you're using the domain name that is associated with your distribution in the URLs for your files.

```
Set-Cookie: CloudFront-
Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2QxMTExMTFhYmNkZWY4LmNsb3VkZnJvbnQubmV0L2dh
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_;
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F;
 Domain=dd111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
```

Example Example 4

You can use the `Set-Cookie` header pairs for one signed request when you're using an alternate domain name (example.org) that is associated with your distribution in the URLs for your files.

```
Set-Cookie: CloudFront-
Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2QxMTExMTFhYmNkZWY4LmNsb3VkZnJvbnQubmV0L2dh
 Domain=example.org; Path=/; Secure; HttpOnly
```

```
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_; Domain=example.org;
 Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F; Domain=example.org; Path=/; Secure;
 HttpOnly
```

Creating a policy statement for a signed cookie that uses a custom policy

To create a policy statement for a custom policy, complete the following steps. For several example policy statements that control access to files in a variety of ways, see Example policy statements for a signed cookie that uses a custom policy.

To create the policy statement for a signed cookie that uses a custom policy

1. Construct the policy statement using the following JSON format.

```
{
    "Statement": [
        {
            "Resource": "URL of the file",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime":required ending date and time in Unix time
 format and UTC
                },
                "DateGreaterThan": {
                    "AWS:EpochTime":optional beginning date and time in Unix time
 format and UTC
                },
                "IpAddress": {
                    "AWS:SourceIp": "optional IP address"
                }
            }
        }
    ]
}
```

Note the following:

- You can include only one statement.
- Use UTF-8 character encoding.

- Include all punctuation and parameter names exactly as specified. Abbreviations for parameter names are not accepted.
- The order of the parameters in the `Condition` section doesn't matter.
- For information about the values for `Resource`, `DateLessThan`, `DateGreaterThan`, and `IpAddress`, see Values that you specify in the policy statement for a custom policy for signed cookies.

2. Remove all white space (including tabs and newline characters) from the policy statement. You might have to include escape characters in the string in application code.
3. Base64-encode the policy statement using MIME base64 encoding. For more information, see Section 6.8, Base64 Content-Transfer-Encoding in *RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies*.
4. Replace characters that are invalid in a URL query string with characters that are valid. The following table lists invalid and valid characters.

Replace these invalid characters	With these valid characters
+	- (hyphen)
=	_ (underscore)
/	~ (tilde)

5. Include the resulting value in your `Set-Cookie` header after `CloudFront-Policy=`.
6. Create a signature for the `Set-Cookie` header for `CloudFront-Signature` by hashing, signing, and base64-encoding the policy statement. For more information, see Creating a signature for a signed cookie that uses a custom policy.

Values that you specify in the policy statement for a custom policy for signed cookies

When you create a policy statement for a custom policy, you specify the following values.

Resource

The base URL including your query strings, if any:

```
https://d111111abcdef8.cloudfront.net/images/horizon.jpg?
size=large&license=yes
```

> **Important**
>
> If you omit the `Resource` parameter, users can access all of the files associated with any distribution that is associated with the key pair that you use to create the signed URL.

You can specify only one value for `Resource`.

Note the following:

- **Protocol** – The value must begin with `http://` or `https://`.
- **Query string parameters** – If you have no query string parameters, omit the question mark.
- **Wildcards** – You can use the wildcard character that matches zero or more characters (*) or the wild-card character that matches exactly one character (?) anywhere in the string. For example, the value:

  `https://d111111abcdef8.cloudfront.net/*game_download.zip*`

  would include (for example) the following files:

  - `https://d111111abcdef8.cloudfront.net/game_download.zip`
  - `https://d111111abcdef8.cloudfront.net/example_game_download.zip?license=yes`
  - `https://d111111abcdef8.cloudfront.net/test_game_download.zip?license=temp`
- **Alternate domain names** – If you specify an alternate domain name (CNAME) in the URL, you must specify the alternate domain name when referencing the file in your webpage or application. Do not specify the Amazon S3 URL for the file.

DateLessThan

The expiration date and time for the URL in Unix time format (in seconds) and Coordinated Universal Time (UTC). Do not enclose the value in quotation marks.

For example, March 16, 2015 10:00 am UTC converts to 1426500000 in Unix time format.

For more information, see When does CloudFront check the expiration date and time in a signed cookie?.

DateGreaterThan (Optional)

An optional start date and time for the URL in Unix time format (in seconds) and Coordinated Universal Time (UTC). Users are not allowed to access the file on or before the specified date and time. Do not enclose the value in quotation marks.

IpAddress (Optional)

The IP address of the client making the GET request. Note the following:

- To allow any IP address to access the file, omit the `IpAddress` parameter.
- You can specify either one IP address or one IP address range. For example, you can't set the policy to allow access if the client's IP address is in one of two separate ranges.
- To allow access from a single IP address, you specify:

  `"IPv4 IP address/32"`

- You must specify IP address ranges in standard IPv4 CIDR format (for example, `192.0.2.0/24`). For more information, go to *RFC 4632, Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*, https://tools.ietf.org/html/rfc4632.

> **Important**
>
> IP addresses in IPv6 format, such as 2001:0db8:85a3::8a2e:0370:7334, are not supported.

If you're using a custom policy that includes `IpAddress`, do not enable IPv6 for the distribution. If you want to restrict access to some content by IP address and support IPv6 requests for other content, you can create two distributions. For more information, see Enable IPv6 in the topic Values that you specify when you create or update a distribution.

Example policy statements for a signed cookie that uses a custom policy

The following example policy statements show how to control access to a specific file, all of the files in a directory, or all of the files associated with a key pair ID. The examples also show how to

control access from an individual IP address or a range of IP addresses, and how to prevent users from using the signed cookie after a specified date and time.

If you copy and paste any of these examples, remove any white space (including tabs and newline characters), replace the values with your own values, and include a newline character after the closing brace (}).

For more information, see Values that you specify in the policy statement for a custom policy for signed cookies.

Topics

- Example policy statement: accessing one file from a range of IP addresses
- Example policy statement: accessing all files in a directory from a range of IP addresses
- Example policy statement: accessing all files associated with a key pair ID from one IP address

Example policy statement: accessing one file from a range of IP addresses

The following example custom policy in a signed cookie specifies that a user can access the file `https://d111111abcdef8.cloudfront.net/game_download.zip` from IP addresses in the range `192.0.2.0/24` until January 1, 2023 10:00 am UTC:

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/game_download.zip",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.0/24"
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1357034400
                }
            }
        }
    ]
}
```

Example policy statement: accessing all files in a directory from a range of IP addresses

The following example custom policy allows you to create signed cookies for any file in the `training` directory, as indicated by the * wildcard character in the `Resource` parameter. Users can access the file from an IP address in the range `192.0.2.0/24` until January 1, 2013 10:00 am UTC:

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/training/*",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.0/24"
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1357034400
                }
            }
        }
    ]
}
```

Each signed cookie in which you use this policy includes a base URL that identifies a specific file, for example:

`https://d111111abcdef8.cloudfront.net/training/orientation.pdf`

Example policy statement: accessing all files associated with a key pair ID from one IP address

The following sample custom policy allows you to set signed cookies for any file associated with any distribution, as indicated by the * wildcard character in the `Resource` parameter. The user must use the IP address `192.0.2.10/32`. (The value `192.0.2.10/32` in CIDR notation refers to a single IP address, `192.0.2.10`.) The files are available only from January 1, 2013 10:00 am UTC until January 2, 2013 10:00 am UTC:

```
{
    "Statement": [
        {
            "Resource": "https://*",
            "Condition": {
                "IpAddress": {
```

```
                    "AWS:SourceIp": "192.0.2.10/32"
                },
                "DateGreaterThan": {
                    "AWS:EpochTime": 1357034400
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1357120800
                }
            }
        }
    ]
}
```

Each signed cookie in which you use this policy includes a base URL that identifies a specific file in a specific CloudFront distribution, for example:

`https://d111111abcdef8.cloudfront.net/training/orientation.pdf`

The signed cookie also includes a key pair ID, which must be associated with a trusted key group in the distribution (d111111abcdef8.cloudfront.net) that you specify in the base URL.

Creating a signature for a signed cookie that uses a custom policy

The signature for a signed cookie that uses a custom policy is a hashed, signed, and base64-encoded version of the policy statement.

For additional information and examples of how to hash, sign, and encode the policy statement, see:

- Using Linux commands and OpenSSL for base64 encoding and encryption
- Code examples for creating a signature for a signed URL

To create a signature for a signed cookie by using a custom policy

1. Use the SHA-1 hash function and RSA to hash and sign the JSON policy statement that you created in the procedure To create the policy statement for a signed URL that uses a custom policy. Use the version of the policy statement that no longer includes white space but that has not yet been base64-encoded.

   For the private key that is required by the hash function, use a private key whose public key is in an active trusted key group for the distribution.

> **Note**
>
> The method that you use to hash and sign the policy statement depends on your programming language and platform. For sample code, see Code examples for creating a signature for a signed URL.

2. Remove white space (including tabs and newline characters) from the hashed and signed string.
3. Base64-encode the string using MIME base64 encoding. For more information, see Section 6.8, Base64 Content-Transfer-Encoding in *RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies*.
4. Replace characters that are invalid in a URL query string with characters that are valid. The following table lists invalid and valid characters.

Replace these invalid characters	With these valid characters
+	- (hyphen)
=	_ (underscore)
/	~ (tilde)

5. Include the resulting value in the `Set-Cookie` header for the `CloudFront-Signature=` name-value pair, and return to To set a signed cookie using a custom policy to add the `Set-Cookie` header for `CloudFront-Key-Pair-Id`.

Using Linux commands and OpenSSL for base64 encoding and encryption

You can use the following Linux command-line command and OpenSSL to hash and sign the policy statement, base64-encode the signature, and replace characters that are not valid in URL query string parameters with characters that are valid.

For information about OpenSSL, go to https://www.openssl.org.

```
cat policy |
tr -d "\n" | tr -d " \t\n\r" |
openssl sha1 -sign private_key.pem |
openssl base64 -A |
tr -- '+=/' '-_~'
```

where:

`cat` reads the `policy` file.

`tr -d "\n" | tr -d " \t\n\r"` removes the white spaces and newline character that were added by `cat`.

OpenSSL hashes the file using SHA-1 and signs it using RSA and the private key file `private_key.pem`.

OpenSSL base64-encodes the hashed and signed policy statement.

`tr` replaces characters that are not valid in URL query string parameters with characters that are valid.

For code examples that demonstrate creating a signature in several programming languages see Code examples for creating a signature for a signed URL.

Code examples for creating a signature for a signed URL

This section includes downloadable application examples that demonstrate how to create signatures for signed URLs. Examples are available in Perl, PHP, C#, and Java. You can use any of

the examples to create signed URLs. The Perl script runs on Linux and macOS platforms. The PHP example will work on any server that runs PHP. The C# example uses the .NET Framework.

For example code in JavaScript (Node.js), see Creating Amazon CloudFront Signed URLs in Node.js on the Amazon Developer Blog.

For example code in Python, see Generate a signed URL for Amazon CloudFront in the *Amazon SDK for Python (Boto3) API Reference* and this example code in the Boto3 GitHub repository.

Topics

- Create a URL signature using Perl
- Create a URL signature using PHP
- Create a URL signature using C# and the .NET Framework
- Create a URL signature using Java

Create a URL signature using Perl

This section includes a Perl script for Linux/Mac platforms that you can use to create the signature for private content. To create the signature, run the script with command line arguments that specify the CloudFront URL, the path to the private key of the signer, the key ID, and an expiration date for the URL. The tool can also decode signed URLs.

> **Note**
>
> Creating a URL signature is just one part of the process of serving private content using a signed URL. For more information about the end-to-end process, see Using signed URLs.

Topics

- Source for the Perl script to create a signed URL

Source for the Perl script to create a signed URL

The following Perl source code can be used to create a signed URL for CloudFront. Comments in the code include information about the command line switches and the features of the tool.

```
#!/usr/bin/perl -w
```

```
# Copyright 2008 Amazon Technologies, Inc.  Licensed under the Apache License, Version
 2.0 (the "License");
# you may not use this file except in compliance with the License. You may obtain a
 copy of the License at:
#
# https://aws.amazon.com/apache2.0
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 KIND, either express or implied.
# See the License for the specific language governing permissions and limitations under
 the License.

=head1 cfsign.pl

cfsign.pl - A tool to generate and verify Amazon CloudFront signed URLs

=head1 SYNOPSIS

This script uses an existing RSA key pair to sign and verify Amazon CloudFront signed
 URLs

View the script source for details as to which CPAN packages are required beforehand.

For help, try:

cfsign.pl --help

URL signing examples:

cfsign.pl --action encode --url https://images.my-website.com/gallery1.zip --policy
 sample_policy.json --private-key privkey.pem --key-pair-id mykey

cfsign.pl --action encode --url https://images.my-website.com/gallery1.zip --expires
 1257439868 --private-key privkey.pem --key-pair-id mykey

URL decode example:

cfsign.pl --action decode --url "http//mydist.cloudfront.net/?Signature=AGO-
PgxkYo99MkJFHvjfGXjG1QDEXeaDb4Qtzmy85wqyJjK7eKojQWa4BCRcow__&Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc29
Pair-Id=mykey"


To generate an RSA key pair, you can use openssl and the following commands:
```

```
# Generate a 2048 bit key pair
openssl genrsa -out private-key.pem 2048
openssl rsa -in private-key.pem -pubout -out public-key.pem


=head1 OPTIONS

=over 8

=item B<--help>

Print a help message and exits.

=item B<--action> [action]

The action to execute.  action can be one of:

  encode - Generate a signed URL (using a canned policy or a user policy)
  decode - Decode a signed URL

=item B<--url>

The URL to en/decode

=item B<--stream>

The stream to en/decode

=item B<--private-key>

The path to your private key.

=item B<--key-pair-id>

The key pair identifier.

=item B<--policy>

The CloudFront policy document.

=item B<--expires>

The Unix epoch time when the URL is to expire. If both this option and
the --policy option are specified, --policy will be used. Otherwise, this
```

```
option alone will use a canned policy.

=back

=cut

use strict;
use warnings;

# you might need to use CPAN to get these modules.
# run perl -MCPAN -e "install <module>" to get them.
# The openssl command line will also need to be in your $PATH.
use File::Temp qw/tempfile/;
use File::Slurp;
use Getopt::Long;
use IPC::Open2;
use MIME::Base64 qw(encode_base64 decode_base64);
use Pod::Usage;
use URI;

my $CANNED_POLICY
    = '{"Statement":[{"Resource":"<RESOURCE>","Condition":{"DateLessThan":
{"AWS:EpochTime":<EXPIRES>}}}]}';

my $POLICY_PARAM      = "Policy";
my $EXPIRES_PARAM     = "Expires";
my $SIGNATURE_PARAM   = "Signature";
my $KEY_PAIR_ID_PARAM = "Key-Pair-Id";

my $verbose = 0;
my $policy_filename = "";
my $expires_epoch = 0;
my $action = "";
my $help = 0;
my $key_pair_id = "";
my $url = "";
my $stream = "";
my $private_key_filename = "";

my $result = GetOptions("action=s"      => \$action,
                        "policy=s"      => \$policy_filename,
                        "expires=i"     => \$expires_epoch,
                        "private-key=s" => \$private_key_filename,
                        "key-pair-id=s" => \$key_pair_id,
```

```
                        "verbose"       => \$verbose,
                        "help"          => \$help,
                        "url=s"         => \$url,
                        "stream=s"      => \$stream,
                    );

if ($help or !$result) {
    pod2usage(1);
    exit;
}

if ($url eq "" and $stream eq "") {
    print STDERR "Must include a stream or a URL to encode or decode with the --stream
 or --url option\n";
    exit;
}

if ($url ne "" and $stream ne "") {
    print STDERR "Only one of --url and --stream may be specified\n";
    exit;
}

if ($url ne "" and !is_url_valid($url)) {
    exit;
}

if ($stream ne "") {
    exit unless is_stream_valid($stream);

    # The signing mechanism is identical, so from here on just pretend we're
    # dealing with a URL
    $url = $stream;
}

if ($action eq "encode") {
    # The encode action will generate a private content URL given a base URL,
    # a policy file (or an expires timestamp) and a key pair id parameter
    my $private_key;
    my $public_key;
    my $public_key_file;

    my $policy;
    if ($policy_filename eq "") {
        if ($expires_epoch == 0) {
```

```
            print STDERR "Must include policy filename with --policy argument or an
 expires" .
                         "time using --expires\n";
        }

        $policy = $CANNED_POLICY;
        $policy =~ s/<EXPIRES>/$expires_epoch/g;
        $policy =~ s/<RESOURCE>/$url/g;
    } else {
        if (! -e $policy_filename) {
            print STDERR "Policy file $policy_filename does not exist\n";
            exit;
        }
        $expires_epoch = 0; # ignore if set
        $policy = read_file($policy_filename);
    }

    if ($private_key_filename eq "") {
        print STDERR "You must specific the path to your private key file with --
private-key\n";
        exit;
    }

    if (! -e $private_key_filename) {
        print STDERR "Private key file $private_key_filename does not exist\n";
        exit;
    }

    if ($key_pair_id eq "") {
        print STDERR "You must specify a key pair id with --key-pair-id\n";
        exit;
    }

    my $encoded_policy = url_safe_base64_encode($policy);
    my $signature = rsa_sha1_sign($policy, $private_key_filename);
    my $encoded_signature = url_safe_base64_encode($signature);

    my $generated_url = create_url($url, $encoded_policy, $encoded_signature,
 $key_pair_id, $expires_epoch);


    if ($stream ne "") {
        print "Encoded stream (for use within a swf):\n" . $generated_url . "\n";
```

```
        print "Encoded and escaped stream (for use on a webpage):\n" .
 escape_url_for_webpage($generated_url) . "\n";
    } else {
        print "Encoded URL:\n" . $generated_url . "\n";
    }
} elsif ($action eq "decode") {
    my $decoded = decode_url($url);
    if (!$decoded) {
        print STDERR "Improperly formed URL\n";
        exit;
    }

    print_decoded_url($decoded);
} else {
    # No action specified, print help.  But only if this is run as a program (caller
 will be empty)
    pod2usage(1) unless caller();
}

# Decode a private content URL into its component parts
sub decode_url {
    my $url = shift;

    if ($url =~ /(.*)\?(.*)/) {
        my $base_url = $1;
        my $params = $2;

        my @unparsed_params = split(/&/, $params);
        my %params = ();
        foreach my $param (@unparsed_params) {
            my ($key, $val) = split(/=/, $param);
            $params{$key} = $val;
        }

        my $encoded_signature = "";
        if (exists $params{$SIGNATURE_PARAM}) {
            $encoded_signature = $params{"Signature"};
        } else {
            print STDERR "Missing Signature URL parameter\n";
            return 0;
        }

        my $encoded_policy = "";
        if (exists $params{$POLICY_PARAM}) {
```

```
            $encoded_policy = $params{$POLICY_PARAM};
        } else {
            if (!exists $params{$EXPIRES_PARAM}) {
                print STDERR "Either the Policy or Expires URL parameter needs to be
specified\n";
                return 0;
            }

            my $expires = $params{$EXPIRES_PARAM};

            my $policy = $CANNED_POLICY;
            $policy =~ s/<EXPIRES>/$expires/g;

            my $url_without_cf_params = $url;
            $url_without_cf_params =~ s/$SIGNATURE_PARAM=[^&]*&?//g;
            $url_without_cf_params =~ s/$POLICY_PARAM=[^&]*&?//g;
            $url_without_cf_params =~ s/$EXPIRES_PARAM=[^&]*&?//g;
            $url_without_cf_params =~ s/$KEY_PAIR_ID_PARAM=[^&]*&?//g;

            if ($url_without_cf_params =~ /(.*)\?$/) {
                $url_without_cf_params = $1;
            }

            $policy =~ s/<RESOURCE>/$url_without_cf_params/g;

            $encoded_policy = url_safe_base64_encode($policy);
        }

        my $key = "";
        if (exists $params{$KEY_PAIR_ID_PARAM}) {
            $key = $params{$KEY_PAIR_ID_PARAM};
        } else {
            print STDERR "Missing $KEY_PAIR_ID_PARAM parameter\n";
            return 0;
        }

        my $policy = url_safe_base64_decode($encoded_policy);

        my %ret = ();
        $ret{"base_url"} = $base_url;
        $ret{"policy"} = $policy;
        $ret{"key"} = $key;

        return \%ret;
```

```
    } else {
        return 0;
    }
}

# Print a decoded URL out
sub print_decoded_url {
    my $decoded = shift;

    print "Base URL: \n" . $decoded->{"base_url"} . "\n";
    print "Policy: \n" . $decoded->{"policy"} . "\n";
    print "Key: \n" . $decoded->{"key"} . "\n";
}

# Encode a string with base 64 encoding and replace some invalid URL characters
sub url_safe_base64_encode {
    my ($value) = @_;

    my $result = encode_base64($value);
    $result =~ tr|+=/|-_~|;

    return $result;
}

# Decode a string with base 64 encoding.  URL-decode the string first
# followed by reversing any special character ("+=/") translation.
sub url_safe_base64_decode {
    my ($value) = @_;

    $value =~ s/%([0-9A-Fa-f]{2})/chr(hex($1))/eg;
    $value =~ tr|-_~|+=/|;

    my $result = decode_base64($value);

    return $result;
}

# Create a private content URL
sub create_url {
    my ($path, $policy, $signature, $key_pair_id, $expires) = @_;

    my $result;
    my $separator = $path =~ /\?/ ? '&' : '?';
    if ($expires) {
```

```
        $result = "$path$separator$EXPIRES_PARAM=$expires&$SIGNATURE_PARAM=$signature&
$KEY_PAIR_ID_PARAM=$key_pair_id";
    } else {
        $result = "$path$separator$POLICY_PARAM=$policy&$SIGNATURE_PARAM=$signature&
$KEY_PAIR_ID_PARAM=$key_pair_id";
    }
    $result =~ s/\n//g;

    return $result;
}

# Sign a document with given private key file.
# The first argument is the document to sign
# The second argument is the name of the private key file
sub rsa_sha1_sign {
    my ($to_sign, $pvkFile) = @_;
    print "openssl sha1 -sign $pvkFile $to_sign\n";

    return write_to_program($pvkFile, $to_sign);
}

# Helper function to write data to a program
sub write_to_program {
my ($keyfile, $data) = @_;
unlink "temp_policy.dat" if (-e "temp_policy.dat");
unlink "temp_sign.dat" if (-e "temp_sign.dat");

write_file("temp_policy.dat", $data);

system("openssl dgst -sha1 -sign \"$keyfile\" -out temp_sign.dat temp_policy.dat");

my $output = read_file("temp_sign.dat");

    return $output;
}

# Read a file into a string and return the string
sub read_file {
    my ($file) = @_;

    open(INFILE, "<$file") or die("Failed to open $file: $!");
    my $str = join('', <INFILE>);
    close INFILE;
```

```
    return $str;
}

sub is_url_valid {
    my ($url) = @_;

    # HTTP distributions start with http[s]:// and are the correct thing to sign
    if ($url =~ /^https?:\/\//) {
        return 1;
    } else {
        print STDERR "CloudFront requires absolute URLs for HTTP distributions\n";
        return 0;
    }
}

sub is_stream_valid {
    my ($stream) = @_;

    if ($stream =~ /^rtmp:\/\// or $stream =~ /^\/?cfx\/st/) {
        print STDERR "Streaming distributions require that only the stream name is
 signed.\n";
        print STDERR "The stream name is everything after, but not including, cfx/st/
\n";
        return 0;
    } else {
        return 1;
    }
}

# flash requires that the query parameters in the stream name are url
# encoded when passed in through javascript, etc.  This sub handles the minimal
# required url encoding.
sub escape_url_for_webpage {
    my ($url) = @_;

    $url =~ s/\?/%3F/g;
    $url =~ s/=/%3D/g;
    $url =~ s/&/%26/g;

    return $url;
}

1;
```

Create a URL signature using PHP

Any web server that runs PHP can use this PHP example code to create policy statements and signatures for private CloudFront distributions. The full example creates a functioning webpage with signed URL links that play a video stream using CloudFront streaming. You can download the full example at https://docs.amazonaws.cn/AmazonCloudFront/latest/DeveloperGuide/samples/demo-php.zip.

You can also create signed URLs by using the `UrlSigner` class in the Amazon SDK for PHP. For more information, see Class UrlSigner in the *Amazon SDK for PHP API Reference*.

> **Note**
>
> Creating a URL signature is just one part of the process of serving private content using a signed URL. For more information about the entire process, see Using signed URLs.

Topics

- Sample: RSA SHA-1 signature
- Example: create a canned policy
- Example: create a custom policy
- Full code example

Sample: RSA SHA-1 signature

In the following code example, the function `rsa_sha1_sign` hashes and signs the policy statement. The arguments required are a policy statement and the private key that corresponds with a public key that's in a trusted key group for your distribution. Next, the `url_safe_base64_encode` function creates a URL-safe version of the signature.

```
function rsa_sha1_sign($policy, $private_key_filename) {
    $signature = "";

    // load the private key
    $fp = fopen($private_key_filename, "r");
    $priv_key = fread($fp, 8192);
    fclose($fp);
    $pkeyid = openssl_get_privatekey($priv_key);
```

```
    // compute signature
    openssl_sign($policy, $signature, $pkeyid);

    // free the key from memory
    openssl_free_key($pkeyid);

    return $signature;
}

function url_safe_base64_encode($value) {
    $encoded = base64_encode($value);
    // replace unsafe characters +, = and / with
    // the safe characters -, _ and ~
    return str_replace(
        array('+', '=', '/'),
        array('-', '_', '~'),
        $encoded);
}
```

Example: create a canned policy

The following example code constructs a *canned* policy statement for the signature. For more information about canned policies, see Creating a signed URL using a canned policy.

> **Note**
>
> The `$expires` variable is a date/time stamp that must be an integer, not a string.

```
function get_canned_policy_stream_name($video_path, $private_key_filename,
 $key_pair_id, $expires) {
    // this policy is well known by CloudFront, but you still need to sign it,
    // since it contains your parameters
    $canned_policy = '{"Statement":[{"Resource":"' . $video_path . '","Condition":
{"DateLessThan":{"AWS:EpochTime":'. $expires . '}}}]}';

    // sign the canned policy
    $signature = rsa_sha1_sign($canned_policy, $private_key_filename);
    // make the signature safe to be included in a url
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
```

```
    $stream_name = create_stream_name($video_path, null, $encoded_signature,
 $key_pair_id, $expires);
    // url-encode the query string characters to work around a flash player bug
    return encode_query_params($stream_name);
}
```

Example: create a custom policy

The following example code constructs a *custom* policy statement for the signature. For more information about custom policies, see Creating a signed URL using a custom policy.

```
function get_custom_policy_stream_name($video_path, $private_key_filename,
 $key_pair_id, $policy) {
    // sign the policy
    $signature = rsa_sha1_sign($policy, $private_key_filename);
    // make the signature safe to be included in a url
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, $encoded_policy, $encoded_signature,
 $key_pair_id, null);
    // url-encode the query string characters to work around a flash player bug
    return encode_query_params($stream_name);
}
```

Full code example

The following example code provides a complete demonstration of creating CloudFront signed URLs with PHP. You can download this full example at https://docs.amazonaws.cn/AmazonCloudFront/latest/DeveloperGuide/samples/demo-php.zip.

In the following example, you can modify the `$policy` `Condition` element to allow both IPv4 and IPv6 address ranges. For an example, see Using IPv6 addresses in IAM policies in the *Amazon Simple Storage Service User Guide*.

```
<?php

function rsa_sha1_sign($policy, $private_key_filename) {
    $signature = "";

    // load the private key
    $fp = fopen($private_key_filename, "r");
```

```
    $priv_key = fread($fp, 8192);
    fclose($fp);
    $pkeyid = openssl_get_privatekey($priv_key);

    // compute signature
    openssl_sign($policy, $signature, $pkeyid);

    // free the key from memory
    openssl_free_key($pkeyid);

    return $signature;
}

function url_safe_base64_encode($value) {
    $encoded = base64_encode($value);
    // replace unsafe characters +, = and / with the safe characters -, _ and ~
    return str_replace(
        array('+', '=', '/'),
        array('-', '_', '~'),
        $encoded);
}

function create_stream_name($stream, $policy, $signature, $key_pair_id, $expires) {
    $result = $stream;
    // if the stream already contains query parameters, attach the new query parameters
 to the end
    // otherwise, add the query parameters
    $separator = strpos($stream, '?') == FALSE ? '?' : '&';
    // the presence of an expires time means we're using a canned policy
    if($expires) {
        $result .= $path . $separator . "Expires=" . $expires . "&Signature=" .
 $signature . "&Key-Pair-Id=" . $key_pair_id;
    }
    // not using a canned policy, include the policy itself in the stream name
    else {
        $result .= $path . $separator . "Policy=" . $policy . "&Signature=" .
 $signature . "&Key-Pair-Id=" . $key_pair_id;
    }

    // new lines would break us, so remove them
    return str_replace('\n', '', $result);
}

function encode_query_params($stream_name) {
```

```
    // Adobe Flash Player has trouble with query parameters being passed into it,
    // so replace the bad characters with their URL-encoded forms
    return str_replace(
        array('?', '=', '&'),
        array('%3F', '%3D', '%26'),
        $stream_name);
}

function get_canned_policy_stream_name($video_path, $private_key_filename,
 $key_pair_id, $expires) {
    // this policy is well known by CloudFront, but you still need to sign it, since it
 contains your parameters
    $canned_policy = '{"Statement":[{"Resource":"' . $video_path . '","Condition":
{"DateLessThan":{"AWS:EpochTime":'. $expires . '}}}]}';
    // the policy contains characters that cannot be part of a URL, so we base64 encode
 it
    $encoded_policy = url_safe_base64_encode($canned_policy);
    // sign the original policy, not the encoded version
    $signature = rsa_sha1_sign($canned_policy, $private_key_filename);
    // make the signature safe to be included in a URL
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, null, $encoded_signature,
 $key_pair_id, $expires);
    // URL-encode the query string characters to support Flash Player
    return encode_query_params($stream_name);
}

function get_custom_policy_stream_name($video_path, $private_key_filename,
 $key_pair_id, $policy) {
    // the policy contains characters that cannot be part of a URL, so we base64 encode
 it
    $encoded_policy = url_safe_base64_encode($policy);
    // sign the original policy, not the encoded version
    $signature = rsa_sha1_sign($policy, $private_key_filename);
    // make the signature safe to be included in a URL
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, $encoded_policy, $encoded_signature,
 $key_pair_id, null);
    // URL-encode the query string characters to support Flash Player
    return encode_query_params($stream_name);
```

```
}


// Path to your private key.  Be very careful that this file is not accessible
// from the web!

$private_key_filename = '/home/test/secure/example-priv-key.pem';
$key_pair_id = 'K2JCJMDEHXQW5F';

$video_path = 'example.mp4';

$expires = time() + 300; // 5 min from now
$canned_policy_stream_name = get_canned_policy_stream_name($video_path,
 $private_key_filename, $key_pair_id, $expires);

$client_ip = $_SERVER['REMOTE_ADDR'];
$policy =
'{'.
    '"Statement":['.
        '{'.
            '"Resource":"'. $video_path . '",'.
            '"Condition":{'.
                '"IpAddress":{"AWS:SourceIp":"' . $client_ip . '/32"},'.
                '"DateLessThan":{"AWS:EpochTime":' . $expires . '}'.
            '}'.
        '}'.
    ']' .
    '}';
$custom_policy_stream_name = get_custom_policy_stream_name($video_path,
 $private_key_filename, $key_pair_id, $policy);

?>

<html>

<head>
    <title>CloudFront</title>
<script type='text/javascript' src='https://example.cloudfront.net/player/
swfobject.js'></script>
</head>

<body>
    <h1>Amazon CloudFront</h1>
    <h2>Canned Policy</h2>
```

```
    <h3>Expires at <?= gmdate('Y-m-d H:i:s T', $expires) ?></h3>
    <br />

    <div id='canned'>The canned policy video will be here</div>

    <h2>Custom Policy</h2>
    <h3>Expires at <?= gmdate('Y-m-d H:i:s T', $expires) ?> only viewable by IP <?=
 $client_ip ?></h3>
    <div id='custom'>The custom policy video will be here</div>

    <!-- ************** Have to update the player.swf path to a real JWPlayer instance.
    The fake one means that external people cannot watch the video right now -->
    <script type='text/javascript'>
  var so_canned = new SWFObject('https://files.example.com/
player.swf','mpl','640','360','9');
  so_canned.addParam('allowfullscreen','true');
  so_canned.addParam('allowscriptaccess','always');
  so_canned.addParam('wmode','opaque');
  so_canned.addVariable('file','<?= $canned_policy_stream_name ?>');
  so_canned.addVariable('streamer','rtmp://example.cloudfront.net/cfx/st');
        so_canned.write('canned');

  var so_custom = new SWFObject('https://files.example.com/
player.swf','mpl','640','360','9');
  so_custom.addParam('allowfullscreen','true');
  so_custom.addParam('allowscriptaccess','always');
  so_custom.addParam('wmode','opaque');
  so_custom.addVariable('file','<?= $custom_policy_stream_name ?>');
  so_custom.addVariable('streamer','rtmp://example.cloudfront.net/cfx/st');
  so_custom.write('custom');
    </script>
</body>

</html>
```

See also:

- Create a URL signature using Perl
- Create a URL signature using C# and the .NET Framework
- Create a URL signature using Java

Create a URL signature using C# and the .NET Framework

The C# examples in this section implement an example application that demonstrates how to create the signatures for CloudFront private distributions using canned and custom policy statements. The examples include utility functions based on the Amazon SDK for .NET that can be useful in .NET applications.

You can also create signed URLs and signed cookies by using the Amazon SDK for .NET. In the Amazon SDK for .NET API Reference, see the following topics:

- **Signed URLs** – Amazon.CloudFront > AmazonCloudFrontUrlSigner
- **Signed cookies** – Amazon.CloudFront > AmazonCloudFrontCookieSigner

> **Note**
>
> Creating a URL signature is just one part of the process of serving private content using a signed URL. For more information about the entire process, see Using signed URLs.

To download the code, go to Signature Code in C#.

To use an RSA key in the .NET Framework, you must convert the Amazon supplied .pem file to the XML format that the .NET Framework uses.

After conversion, the RSA private key file is in the following format:

Example RSA private key in the XML .NET Framework format

```
<RSAKeyValue>
  <Modulus>
    wO5IvYCP5UcoCKDo1dcspoMehWBZcyfs9QEzGi6Oe5y+ewGr1oW+vB2GPB
    ANBiVPcUHTFWhwaIBd3oglmF0lGQljP/jOfmXHUK2kUUnLnJp+oOBL2NiuFtqcW6h/L5lIpD8Yq+NRHg
    Ty4zDsyr2880MvXv88yEFURCkqEXAMPLE=
  </Modulus>
  <Exponent>AQAB</Exponent>
  <P>
    5bmKDaTz
    npENGVqz4Cea8XPH+sxt+2VaAwYnsarVUoSBeVt8WLloVuZGG9IZYmH5KteXEu7fZveYd9UEXAMPLE==
  </P>
  <Q>
    1v9l/WN1a1N3rOK4VGoCokx7kR2SyTMSbZgF9IWJNOugR/WZw7HTnjipO3c9dy1Ms9pUKwUF4
```

```
    6d7049EXAMPLE==
  </Q>
  <DP>
    RgrSKuLWXMyBH+/l1Dx/I4tXuAJIrlPyo+VmiOc7b5NzHptkSHEPfR9s1
    OK0VqjknclqCJ3Ig86OMEtEXAMPLE==
  </DP>
  <DQ>
    pjPjvSFw+RoaTu0pgCA/jwW/FGyfN6iim1RFbkT4
    z49DZb2IM885f3vf35eLTaEYRYUHQgZtChNEV0TEXAMPLE==
  </DQ>
  <InverseQ>
    nkvOJTg5QtGNgWb9i
    cVtzrL/1pFEOHbJXwEJdU99N+7sMK+1066DL/HSBUCD63qD4USpnf0myc24in0EXAMPLE==</InverseQ>
  <D>
      Bc7mp7XYHynuPZxChjWNJZIq+A73gm0ASDv6At7F8Vi9r0xUlQe/v0AQS3ycN8QlyR4XMbzMLYk
      3yjxFDXo4ZKQtOGzLGteCU2srANiLv26/imXA8FVidZftTAtLviWQZBVPTeYIA69ATUYPEq0a5u5wjGy
      UOij9OWyuEXAMPLE=
   </D>
</RSAKeyValue>
```

The following C# code creates a signed URL that uses a canned policy by doing the following:

- Creates a policy statement.
- Hashes the policy statement using SHA1, and signs the result using RSA and the private key whose corresponding public key is in a trusted key group.
- Base64-encodes the hashed and signed policy statement and replaces special characters to make the string safe to use as a URL request parameter.
- Concatenates the values.

For the complete implementation, see the example at Signature Code in C#.

Example Canned policy signing method in C#

```
public static string ToUrlSafeBase64String(byte[] bytes)
{
    return System.Convert.ToBase64String(bytes)
        .Replace('+', '-')
        .Replace('=', '_')
        .Replace('/', '~');
}
```

```
public static string CreateCannedPrivateURL(string urlString,
    string durationUnits, string durationNumber, string pathToPolicyStmnt,
    string pathToPrivateKey, string privateKeyId)
{
    // args[] 0-thisMethod, 1-resourceUrl, 2-seconds-minutes-hours-days
    // to expiration, 3-numberOfPreviousUnits, 4-pathToPolicyStmnt,
    // 5-pathToPrivateKey, 6-PrivateKeyId

    TimeSpan timeSpanInterval = GetDuration(durationUnits, durationNumber);

    // Create the policy statement.
    string strPolicy = CreatePolicyStatement(pathToPolicyStmnt,
        urlString,
        DateTime.Now,
        DateTime.Now.Add(timeSpanInterval),
        "0.0.0.0/0");
    if ("Error!" == strPolicy) return "Invalid time frame." +
        "Start time cannot be greater than end time.";

    // Copy the expiration time defined by policy statement.
    string strExpiration = CopyExpirationTimeFromPolicy(strPolicy);

    // Read the policy into a byte buffer.
    byte[] bufferPolicy = Encoding.ASCII.GetBytes(strPolicy);

    // Initialize the SHA1CryptoServiceProvider object and hash the policy data.
    using (SHA1CryptoServiceProvider
        cryptoSHA1 = new SHA1CryptoServiceProvider())
    {
        bufferPolicy = cryptoSHA1.ComputeHash(bufferPolicy);

        // Initialize the RSACryptoServiceProvider object.
        RSACryptoServiceProvider providerRSA = new RSACryptoServiceProvider();
        XmlDocument xmlPrivateKey = new XmlDocument();

        // Load your private key, which you created by converting your
        // .pem file to the XML format that the .NET framework uses.
        // Several tools are available.
        xmlPrivateKey.Load(pathToPrivateKey);

        // Format the RSACryptoServiceProvider providerRSA and
        // create the signature.
        providerRSA.FromXmlString(xmlPrivateKey.InnerXml);
        RSAPKCS1SignatureFormatter rsaFormatter =
```

```
            new RSAPKCS1SignatureFormatter(providerRSA);
        rsaFormatter.SetHashAlgorithm("SHA1");
        byte[] signedPolicyHash = rsaFormatter.CreateSignature(bufferPolicy);

        // Convert the signed policy to URL-safe base64 encoding and
        // replace unsafe characters + = / with the safe characters - _ ~
        string strSignedPolicy = ToUrlSafeBase64String(signedPolicyHash);

        // Concatenate the URL, the timestamp, the signature,
        // and the key pair ID to form the signed URL.
        return urlString +
            "?Expires=" +
            strExpiration +
            "&Signature=" +
            strSignedPolicy +
            "&Key-Pair-Id=" +
            privateKeyId;
    }
}
```

The following C# code creates a signed URL that uses a custom policy by doing the following:

1. Creates a policy statement.
2. Base64-encodes the policy statement and replaces special characters to make the string safe to use as a URL request parameter.
3. Hashes the policy statement using SHA1, and encrypts the result using RSA and the private key whose corresponding public key is in a trusted key group.
4. Base64-encodes the hashed policy statement and replacing special characters to make the string safe to use as a URL request parameter.
5. Concatenates the values.

For the complete implementation, see the example at Signature Code in C#.

Example Custom policy signing method in C#

```
public static string ToUrlSafeBase64String(byte[] bytes)
{
    return System.Convert.ToBase64String(bytes)
        .Replace('+', '-')
        .Replace('=', '_')
```

```
        .Replace('/', '~');
}

public static string CreateCustomPrivateURL(string urlString,
    string durationUnits, string durationNumber, string startIntervalFromNow,
    string ipaddress, string pathToPolicyStmnt, string pathToPrivateKey,
    string PrivateKeyId)
{
    // args[] 0-thisMethod, 1-resourceUrl, 2-seconds-minutes-hours-days
    // to expiration, 3-numberOfPreviousUnits, 4-starttimeFromNow,
    // 5-ip_address, 6-pathToPolicyStmt, 7-pathToPrivateKey, 8-privateKeyId

    TimeSpan timeSpanInterval = GetDuration(durationUnits, durationNumber);
    TimeSpan timeSpanToStart = GetDurationByUnits(durationUnits,
        startIntervalFromNow);
    if (null == timeSpanToStart)
        return "Invalid duration units." +
            "Valid options: seconds, minutes, hours, or days";

    string strPolicy = CreatePolicyStatement(
        pathToPolicyStmnt, urlString, DateTime.Now.Add(timeSpanToStart),
        DateTime.Now.Add(timeSpanInterval), ipaddress);

    // Read the policy into a byte buffer.
    byte[] bufferPolicy = Encoding.ASCII.GetBytes(strPolicy);

    // Convert the policy statement to URL-safe base64 encoding and
    // replace unsafe characters + = / with the safe characters - _ ~

    string urlSafePolicy = ToUrlSafeBase64String(bufferPolicy);

    // Initialize the SHA1CryptoServiceProvider object and hash the policy data.
    byte[] bufferPolicyHash;
    using (SHA1CryptoServiceProvider cryptoSHA1 =
        new SHA1CryptoServiceProvider())
    {
        bufferPolicyHash = cryptoSHA1.ComputeHash(bufferPolicy);

        // Initialize the RSACryptoServiceProvider object.
        RSACryptoServiceProvider providerRSA = new RSACryptoServiceProvider();
        XmlDocument xmlPrivateKey = new XmlDocument();

        // Load your private key, which you created by converting your
        // .pem file to the XML format that the .NET framework uses.
```

```
        // Several tools are available.
        xmlPrivateKey.Load(pathToPrivateKey);

        // Format the RSACryptoServiceProvider providerRSA
        // and create the signature.
        providerRSA.FromXmlString(xmlPrivateKey.InnerXml);
        RSAPKCS1SignatureFormatter RSAFormatter =
            new RSAPKCS1SignatureFormatter(providerRSA);
        RSAFormatter.SetHashAlgorithm("SHA1");
        byte[] signedHash = RSAFormatter.CreateSignature(bufferPolicyHash);

        // Convert the signed policy to URL-safe base64 encoding and
        // replace unsafe characters + = / with the safe characters - _ ~
        string strSignedPolicy = ToUrlSafeBase64String(signedHash);

        return urlString +
            "?Policy=" +
            urlSafePolicy +
            "&Signature=" +
            strSignedPolicy +
            "&Key-Pair-Id=" +
            PrivateKeyId;
    }
}
```

Example Utility methods for signature generation

The following methods get the policy statement from a file and parse time intervals for signature generation.

```
public static string CreatePolicyStatement(string policyStmnt,
   string resourceUrl,
   DateTime startTime,
   DateTime endTime,
   string ipAddress)

{
   // Create the policy statement.
   FileStream streamPolicy = new FileStream(policyStmnt, FileMode.Open,
 FileAccess.Read);
   using (StreamReader reader = new StreamReader(streamPolicy))
   {
      string strPolicy = reader.ReadToEnd();
```

```
      TimeSpan startTimeSpanFromNow = (startTime - DateTime.Now);
      TimeSpan endTimeSpanFromNow = (endTime - DateTime.Now);
      TimeSpan intervalStart =
         (DateTime.UtcNow.Add(startTimeSpanFromNow)) -
         new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);
      TimeSpan intervalEnd =
         (DateTime.UtcNow.Add(endTimeSpanFromNow)) -
         new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);

      int startTimestamp = (int)intervalStart.TotalSeconds; // START_TIME
      int endTimestamp = (int)intervalEnd.TotalSeconds;  // END_TIME

      if (startTimestamp > endTimestamp)
         return "Error!";

      // Replace variables in the policy statement.
      strPolicy = strPolicy.Replace("RESOURCE", resourceUrl);
      strPolicy = strPolicy.Replace("START_TIME", startTimestamp.ToString());
      strPolicy = strPolicy.Replace("END_TIME", endTimestamp.ToString());
      strPolicy = strPolicy.Replace("IP_ADDRESS", ipAddress);
      strPolicy = strPolicy.Replace("EXPIRES", endTimestamp.ToString());
      return strPolicy;
   }
}

public static TimeSpan GetDuration(string units, string numUnits)
{
   TimeSpan timeSpanInterval = new TimeSpan();
   switch (units)
   {
      case "seconds":
         timeSpanInterval = new TimeSpan(0, 0, 0, int.Parse(numUnits));
         break;
      case "minutes":
         timeSpanInterval = new TimeSpan(0, 0, int.Parse(numUnits), 0);
         break;
      case "hours":
         timeSpanInterval = new TimeSpan(0, int.Parse(numUnits), 0 ,0);
         break;
      case "days":
         timeSpanInterval = new TimeSpan(int.Parse(numUnits),0 ,0 ,0);
         break;
      default:
```

```
         Console.WriteLine("Invalid time units;" +
            "use seconds, minutes, hours, or days");
         break;
   }
   return timeSpanInterval;
}

private static TimeSpan GetDurationByUnits(string durationUnits,
   string startIntervalFromNow)
{
   switch (durationUnits)
   {
      case "seconds":
         return new TimeSpan(0, 0, int.Parse(startIntervalFromNow));
      case "minutes":
         return new TimeSpan(0, int.Parse(startIntervalFromNow), 0);
      case "hours":
         return new TimeSpan(int.Parse(startIntervalFromNow), 0, 0);
      case "days":
         return new TimeSpan(int.Parse(startIntervalFromNow), 0, 0, 0);
      default:
         return new TimeSpan(0, 0, 0, 0);
   }
}

public static string CopyExpirationTimeFromPolicy(string policyStatement)
{
   int startExpiration = policyStatement.IndexOf("EpochTime");
   string strExpirationRough = policyStatement.Substring(startExpiration +
      "EpochTime".Length);
   char[] digits = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' };

   List<char> listDigits = new List<char>(digits);
   StringBuilder buildExpiration = new StringBuilder(20);

   foreach (char c in strExpirationRough)
   {
      if (listDigits.Contains(c))
         buildExpiration.Append(c);
   }
   return buildExpiration.ToString();
}
```

See also

- Create a URL signature using Perl
- Create a URL signature using PHP
- Create a URL signature using Java

Create a URL signature using Java

In addition to the following code example, you can use the `CloudFrontUrlSigner` utility class in the Amazon SDK for Java (version 1) to create CloudFront signed URLs.

For more examples, see Create signed URLs and cookies using an Amazon SDK in the *Amazon SDK Code Examples Code Library*.

> **Note**
>
> Creating a signed URL is just one part of the process of serving private content with CloudFront. For more information about the entire process, see Using signed URLs.

The following example shows how to create a CloudFront signed URL.

Example Java policy and signature encryption methods

```
package org.example;

import java.time.Instant;
import java.time.temporal.ChronoUnit;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class Main {

    public static void main(String[] args) throws Exception {
        CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilities.create();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        String resourceUrl = "https://a1b2c3d4e5f6g7.cloudfront.net";
        String keyPairId = "K1UA3WV15I7JSD";
        CannedSignerRequest cannedRequest = CannedSignerRequest.builder()
                .resourceUrl(resourceUrl)
```

```
                .privateKey(new java.io.File("/path/to/private_key.pem").toPath())
                .keyPairId(keyPairId)
                .expirationDate(expirationDate)
                .build();
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedRequest);
        String url = signedUrl.url();
        System.out.println(url);

    }
}
```

See also:

- Create a URL signature using Perl
- Create a URL signature using PHP
- Create a URL signature using C# and the .NET Framework

Restricting access to an Amazon origin

You can configure CloudFront and some Amazon origins in a way that provides the following benefits:

- Restricts access to the Amazon origin so that it's not publicly accessible
- Makes sure that viewers (users) can access the content in the Amazon origin only through the specified CloudFront distribution—preventing them from accessing the content directly from the bucket, or through an unintended CloudFront distribution

To do this, configure CloudFront to send authenticated requests to your Amazon origin, and configure the Amazon origin to only allow access to authenticated requests from CloudFront. For more information, see following topics for compatible types of Amazon origins.

Topics

- Restricting access to an AWS Elemental MediaPackage v2 origin
- Restricting access to an AWS Elemental MediaStore origin
- Restricting access to an Amazon Lambda function URL origin
- Restricting access to an Amazon Simple Storage Service origin

Restricting access to an AWS Elemental MediaPackage v2 origin

CloudFront provides *origin access control* (OAC) for restricting access to a MediaPackage v2 origin.

> **Note**
>
> CloudFront OAC only supports MediaPackage v2. MediaPackage v1 isn't supported.

Topics

- Creating a new OAC
- Advanced settings for origin access control

Creating a new OAC

Complete the steps described in the following topics to set up a new OAC in CloudFront.

Topics

- Prerequisites
- Giving the OAC permission to access the MediaPackage v2 origin
- Creating the OAC

Prerequisites

Before you create and set up OAC, you must have a CloudFront distribution with a MediaPackage v2 origin. For more information, see Using a MediaStore container or a MediaPackage channel.

Giving the OAC permission to access the MediaPackage v2 origin

Before you create an OAC or set it up in a CloudFront distribution, make sure the OAC has permission to access the MediaPackage v2 origin. Do this after you create a CloudFront distribution, but before you add the OAC to the MediaPackage v2 origin in the distribution configuration.

To give the OAC permission to access the MediaPackage v2 origin, use a IAM policy to allow the CloudFront service principal (`cloudfront.amazonaws.com`) to access the origin. The `Condition` element in the policy allows CloudFront to access the MediaPackage v2 origin *only* when the request is on behalf of the CloudFront distribution that contains the MediaPackage v2 origin.

Example : IAM policy that allows read-only access to a CloudFront distribution

The following policy allows the CloudFront distribution (*E1PDK09ESKHJWT*) access to the MediaPackage v2 origin. The origin is the ARN specified for the `Resource` element.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipal",
            "Effect": "Allow",
            "Principal": {"Service": "cloudfront.amazonaws.com"},
            "Action": "mediapackagev2:GetObject",
            "Resource": "arn:aws:mediapackagev2:us-
east-1:123456789012:channelGroup/channel-group-name/channel/channel-name/
originEndpoint/origin_endpoint_name",
            "Condition": {
                "StringEquals": {"AWS:SourceArn":
 "arn:aws:cloudfront::123456789012:distribution/E1PDK09ESKHJWT"}
            }
        }
    ]
}
```

> **Note**
>
> If you create a distribution that doesn't have permission to your MediaPackage v2 origin, you can choose **Copy policy** from the CloudFront console and then choose **Update endpoint permissions**. You can then attach the copied permission to the endpoint. For more information, see Endpoint policy fields in the *AWS Elemental MediaPackage User Guide*.

Creating the OAC

To create an OAC, you can use the Amazon Web Services Management Console, Amazon CloudFormation, the Amazon CLI, or the CloudFront API.

Console

To create an OAC

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Origin access**.
3. Choose **Create control setting**.
4. On the **Create new OAC** form, do the following:
   a. Enter a **Name** and (optionally) a **Description** for the OAC.
   b. For **Signing behavior**, we recommend that you leave the default setting (**Sign requests (recommended)**). For more information, see the section called "Advanced settings for origin access control".
5. For **Origin type**, choose **MediaPackage V2**.
6. Choose **Create**.

> **Tip**
>
> After you create the OAC, make note of the **Name**. You need this in the following procedure.

To add an OAC to a MediaPackage v2 origin in a distribution

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose a distribution with a MediaPackage V2 origin that you want to add the OAC to, then choose the **Origins** tab.
3. Select the MediaPackage v2 origin that you want to add the OAC to, then choose **Edit**.
4. Select **HTTPS only** for your origin's **Protocol**.
5. From the **Origin access control** dropdown, choose the OAC name that you want to use.
6. Choose **Save changes**.

The distribution starts deploying to all of the CloudFront edge locations. When an edge location receives the new configuration, it signs all requests that it sends to the MediaPackage v2 origin.

CloudFormation

To create an OAC with Amazon CloudFormation, use the `AWS::CloudFront::OriginAccessControl` resource type. The following example shows the Amazon CloudFormation template syntax, in YAML format, for creating an OAC.

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
      Description: An optional description for the origin access control
      Name: ExampleOAC
      OriginAccessControlOriginType: mediapackagev2
      SigningBehavior: always
      SigningProtocol: sigv4
```

For more information, see AWS::CloudFront::OriginAccessControl in the *Amazon CloudFormation User Guide*.

CLI

To create an origin access control with the Amazon Command Line Interface (Amazon CLI), use the **aws cloudfront create-origin-access-control** command. You can use an input file to provide the input parameters for the command, rather than specifying each individual parameter as command line input.

To create an origin access control (CLI with input file)

1. Use the following command to create a file that's named `origin-access-control.yaml`. This file contains all of the input parameters for the **create-origin-access-control** command.

   ```
   aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
    origin-access-control.yaml
   ```

2. Open the `origin-access-control.yaml` file that you just created. Edit the file to add a name for the OAC, a description (optional), and change the `SigningBehavior` to `always`. Then save the file.

   For information about other OAC settings, see the section called “Advanced settings for origin access control”.

3. Use the following command to create the origin access control using the input parameters from the `origin-access-control.yaml` file.

```
aws cloudfront create-origin-access-control --cli-input-yaml file://origin-access-control.yaml
```

Make note of the `Id` value in the command output. You need it to add the OAC to a MediaPackage v2 origin in a CloudFront distribution.

To attach an OAC to a MediaPackage v2 origin in an existing distribution (CLI with input file)

1. Use the following command to save the distribution configuration for the CloudFront distribution that you want to add the OAC to. The distribution must have a MediaPackage v2 origin.

```
aws cloudfront get-distribution-config --id <CloudFront distribution ID> --output yaml > dist-config.yaml
```

2. Open the file that's named `dist-config.yaml` that you just created. Edit the file, making the following changes:

   - In the `Origins` object, add the OAC's ID to the field that's named `OriginAccessControlId`.
   - Remove the value from the field that's named `OriginAccessIdentity`, if one exists.
   - Rename the `ETag` field to `IfMatch`, but don't change the field's value.

   Save the file when finished.

3. Use the following command to update the distribution to use the origin access control.

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-input-yaml file://dist-config.yaml
```

The distribution starts deploying to all of the CloudFront edge locations. When an edge location receives the new configuration, it signs all requests that it sends to the MediaPackage v2 origin.

API

To create an OAC with the CloudFront API, use CreateOriginAccessControl. For more information about the fields that you specify in this API call, see the API reference documentation for your Amazon SDK or other API client.

After you create an OAC you can attach it to a MediaPackage v2 origin in a distribution, using one of the following API calls:

- To attach it to an existing distribution, use UpdateDistribution.
- To attach it to a new distribution, use CreateDistribution.

For both of these API calls, provide the OAC ID in the `OriginAccessControlId` field, inside an origin. For more information about the other fields that you specify in these API calls, see Values that you specify when you create or update a distribution and the API reference documentation for your Amazon SDK or other API client.

Advanced settings for origin access control

The CloudFront OAC feature includes advanced settings that are intended only for specific use cases. Use the recommended settings unless you have a specific need for the advanced settings.

OAC contains a setting named **Signing behavior** (in the console), or `SigningBehavior` (in the API, CLI, and Amazon CloudFormation). This setting provides the following options:

Always sign origin requests (recommended setting)

We recommend using this setting, named **Sign requests (recommended)** in the console, or `always` in the API, CLI, and Amazon CloudFormation. With this setting, CloudFront always signs all requests that it sends to the MediaPackage v2 origin.

Never sign origin requests

This setting is named **Do not sign requests** in the console, or `never` in the API, CLI, and Amazon CloudFormation. Use this setting to turn off OAC for all origins in all distributions that use this OAC. This can save time and effort compared to removing an OAC from all origins and distributions that use it, one by one. With this setting, CloudFront doesn't sign any requests that it sends to the MediaPackage v2 origin.

> **Warning**
>
> To use this setting, the MediaPackage v2 origin must be publicly accessible. If you use this setting with a MediaPackage v2 origin that's not publicly accessible, CloudFront can't access the origin. The MediaPackage v2 origin returns errors to CloudFront and CloudFront passes those errors on to viewers. For more information, see the example MediaPackage v2 policy for Policies and Permissions in MediaPackage in the *AWS Elemental MediaPackage User Guide*.

Don't override the viewer (client) `Authorization` header

This setting is named **Do not override authorization header** in the console, or `no-override` in the API, CLI, and Amazon CloudFormation. Use this setting when you want CloudFront to sign origin requests only when the corresponding viewer request does not include an `Authorization` header. With this setting, CloudFront passes on the `Authorization` header from the viewer request when one is present, but signs the origin request (adding its own `Authorization` header) when the viewer request doesn't include an `Authorization` header.

> **Warning**
>
> To pass along the `Authorization` header from the viewer request, you *must* add the `Authorization` header to a cache policy for all cache behaviors that use MediaPackage v2 origins associated with this origin access control.

Restricting access to an AWS Elemental MediaStore origin

CloudFront provides *origin access control* (OAC) for restricting access to an AWS Elemental MediaStore origin.

Topics

- Creating a new origin access control
- Advanced settings for origin access control

Creating a new origin access control

Complete the steps described in the following topics to set up a new origin access control in CloudFront.

Topics

- Prerequisites
- Giving the origin access control permission to access the MediaStore origin
- Creating the origin access control

Prerequisites

Before you create and set up origin access control, you must have a CloudFront distribution with a MediaStore origin.

Giving the origin access control permission to access the MediaStore origin

Before you create an origin access control or set it up in a CloudFront distribution, make sure the OAC has permission to access the MediaStore origin. Do this after creating a CloudFront distribution, but before adding the OAC to the MediaStore origin in the distribution configuration.

To give the OAC permission to access the MediaStore origin, use a MediaStore container policy to allow the CloudFront service principal (`cloudfront.amazonaws.com`) to access the origin. Use a `Condition` element in the policy to allow CloudFront to access the MediaStore container only when the request is on behalf of the CloudFront distribution that contains the MediaStore origin.

The following are examples of MediaStore container policies that allow a CloudFront OAC to access a MediaStore origin.

Example MediaStore container policy that allows read-only access to a CloudFront OAC

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "AllowCloudFrontServicePrincipalReadOnly",
                "Effect": "Allow",
                "Principal": {
                  "Service": "cloudfront.amazonaws.com"
                },
```

```
                "Action": [
                  "mediastore:GetObject"
                ],
                "Resource":
 "arn:aws:mediastore:<region>:111122223333:container/<container name>/*",
                "Condition": {
                    "StringEquals": {
                      "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
                    },
                    "Bool": {
                      "aws:SecureTransport": "true"
                    }
                }
            }
        ]
}
```

Example MediaStore container policy that allows read and write access to a CloudFront OAC

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "AllowCloudFrontServicePrincipalReadWrite",
                "Effect": "Allow",
                "Principal": {
                  "Service": "cloudfront.amazonaws.com"
                },
                "Action": [
                  "mediastore:GetObject",
                  "mediastore:PutObject"
                ],
                "Resource":
 "arn:aws:mediastore:<region>:111122223333:container/<container name>/*",
                "Condition": {
                    "StringEquals": {
                      "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
                    },
                    "Bool": {
                      "aws:SecureTransport": "true"
                    }
```

```
                }
            }
        ]
}
```

> **Note**
>
> To allow write access, you must configure **Allowed HTTP methods** to include PUT in your CloudFront distribution's behavior settings.

Creating the origin access control

To create an OAC, you can use the Amazon Web Services Management Console, Amazon CloudFormation, the Amazon CLI, or the CloudFront API.

Console

To create an origin access control

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Origin access**.
3. Choose **Create control setting**.
4. On the **Create control setting** form, do the following:
   a. In the **Details** pane, enter a **Name** and (optionally) a **Description** for the origin access control.
   b. In the **Settings** pane, we recommend that you leave the default setting (**Sign requests (recommended)**). For more information, see the section called “Advanced settings for origin access control”.
5. Choose MediaStore from the **Origin type** dropdown.
6. Choose **Create**.

   After the OAC is created, make note of the **Name**. You need this in the following procedure.

To add an origin access control to a MediaStore origin in a distribution

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose a distribution with a MediaStore origin that you want to add the OAC to, then choose the **Origins** tab.
3. Select the MediaStore origin that you want to add the OAC to, then choose **Edit**.
4. Select **HTTPS only** for your origin's **Protocol**.
5. From the **Origin access control** dropdown menu, choose the OAC that you want to use.
6. Choose **Save changes**.

The distribution starts deploying to all of the CloudFront edge locations. When an edge location receives the new configuration, it signs all requests that it sends to the MediaStore bucket origin.

CloudFormation

To create an origin access control (OAC) with Amazon CloudFormation, use the `AWS::CloudFront::OriginAccessControl` resource type. The following example shows the Amazon CloudFormation template syntax, in YAML format, for creating an origin access control.

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
      Description: An optional description for the origin access control
      Name: ExampleOAC
      OriginAccessControlOriginType: mediastore
      SigningBehavior: always
      SigningProtocol: sigv4
```

For more information, see AWS::CloudFront::OriginAccessControl in the *Amazon CloudFormation User Guide*.

CLI

To create an origin access control with the Amazon Command Line Interface (Amazon CLI), use the **aws cloudfront create-origin-access-control** command. You can use an input file to provide the input parameters for the command, rather than specifying each individual parameter as command line input.

To create an origin access control (CLI with input file)

1. Use the following command to create a file that's named `origin-access-control.yaml`. This file contains all of the input parameters for the **create-origin-access-control** command.

```
aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
 origin-access-control.yaml
```

2. Open the `origin-access-control.yaml` file that you just created. Edit the file to add a name for the OAC, a description (optional), and change the `SigningBehavior` to `always`. Then save the file.

   For information about other OAC settings, see the section called “Advanced settings for origin access control”.

3. Use the following command to create the origin access control using the input parameters from the `origin-access-control.yaml` file.

```
aws cloudfront create-origin-access-control --cli-input-yaml file://origin-
access-control.yaml
```

   Make note of the `Id` value in the command output. You need it to add the OAC to a MediaStore origin in a CloudFront distribution.

To attach an OAC to a MediaStore origin in an existing distribution (CLI with input file)

1. Use the following command to save the distribution configuration for the CloudFront distribution that you want to add the OAC to. The distribution must have a MediaStore origin.

```
aws cloudfront get-distribution-config --id <CloudFront distribution ID> --
output yaml > dist-config.yaml
```

2. Open the file that's named `dist-config.yaml` that you just created. Edit the file, making the following changes:

- In the `Origins` object, add the OAC's ID to the field that's named `OriginAccessControlId`.
- Remove the value from the field that's named `OriginAccessIdentity`, if one exists.
- Rename the `ETag` field to `IfMatch`, but don't change the field's value.

Save the file when finished.

3. Use the following command to update the distribution to use the origin access control.

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-
input-yaml file://dist-config.yaml
```

The distribution starts deploying to all of the CloudFront edge locations. When an edge location receives the new configuration, it signs all requests that it sends to the MediaStore origin.

API

To create an origin access control with the CloudFront API, use CreateOriginAccessControl. For more information about the fields that you specify in this API call, see the API reference documentation for your Amazon SDK or other API client.

After you create an origin access control you can attach it to a MediaStore origin in a distribution, using one of the following API calls:

- To attach it to an existing distribution, use UpdateDistribution.
- To attach it to a new distribution, use CreateDistribution.

For both of these API calls, provide the origin access control ID in the `OriginAccessControlId` field, inside an origin. For more information about the other fields that you specify in these API calls, see Values that you specify when you create or update a distribution and the API reference documentation for your Amazon SDK or other API client.

Advanced settings for origin access control

The CloudFront origin access control feature includes advanced settings that are intended only for specific use cases. Use the recommended settings unless you have a specific need for the advanced settings.

Origin access control contains a setting named **Signing behavior** (in the console), or `SigningBehavior` (in the API, CLI, and Amazon CloudFormation). This setting provides the following options:

Always sign origin requests (recommended setting)

We recommend using this setting, named **Sign requests (recommended)** in the console, or `always` in the API, CLI, and Amazon CloudFormation. With this setting, CloudFront always signs all requests that it sends to the MediaStore origin.

Never sign origin requests

This setting is named **Do not sign requests** in the console, or `never` in the API, CLI, and Amazon CloudFormation. Use this setting to turn off origin access control for all origins in all distributions that use this origin access control. This can save time and effort compared to removing an origin access control from all origins and distributions that use it, one by one. With this setting, CloudFront does not sign any requests that it sends to the MediaStore origin.

> **Warning**
>
> To use this setting, the MediaStore origin must be publicly accessible. If you use this setting with a MediaStore origin that's not publicly accessible, CloudFront cannot access the origin. The MediaStore origin returns errors to CloudFront and CloudFront passes those errors on to viewers. For more information, see the example MediaStore container policy for Public read access over HTTPS.

Don't override the viewer (client) `Authorization` header

This setting is named **Do not override authorization header** in the console, or `no-override` in the API, CLI, and Amazon CloudFormation. Use this setting when you want CloudFront to sign origin requests only when the corresponding viewer request does not include an `Authorization` header. With this setting, CloudFront passes on the `Authorization` header

from the viewer request when one is present, but signs the origin request (adding its own `Authorization` header) when the viewer request doesn't include an `Authorization` header.

> **Warning**
>
> To pass along the `Authorization` header from the viewer request, you *must* add the `Authorization` header to a cache policy for all cache behaviors that use MediaStore origins associated with this origin access control.

Restricting access to an Amazon Lambda function URL origin

CloudFront provides *origin access control* (OAC) for restricting access to a Lambda function URL origin.

Topics

- Creating a new OAC
- Advanced settings for origin access control

Creating a new OAC

Complete the steps described in the following topics to set up a new OAC in CloudFront.

> **Note**
>
> If you use PUT or POST methods with your Lambda function URL, your user must provide a signed payload to CloudFront. Lambda doesn't support unsigned payloads.

Topics

- Prerequisites
- Giving the OAC permission to access the Lambda function URL
- Creating the OAC

Prerequisites

Before you create and set up OAC, you must have a CloudFront distribution with a Lambda function URL as the origin. For more information, see Using a Lambda function URL.

Giving the OAC permission to access the Lambda function URL

Before you create an OAC or set it up in a CloudFront distribution, make sure the OAC has permission to access the Lambda function URL. Do this after you create a CloudFront distribution, but before you add the OAC to the Lambda function URL in the distribution configuration.

> **Note**
>
> To update the IAM policy for the Lambda function URL, you must use the Amazon Command Line Interface (Amazon CLI). Editing the IAM policy in the Lambda console isn't supported at this time.

The following Amazon CLI command grants the CloudFront service principal (`cloudfront.amazonaws.com`) access to your Lambda function URL. The `Condition` element in the policy allows CloudFront to access Lambda *only* when the request is on behalf of the CloudFront distribution that contains the Lambda function URL.

Example : Amazon CLI command to update a policy to allow read-only access to a CloudFront OAC

The following Amazon CLI command allows the CloudFront distribution (*E1PDK09ESKHJWT*) access your Lambda *FUNCTION_URL_NAME*.

```
aws lambda add-permission \
--statement-id "AllowCloudFrontServicePrincipal" \
--action "lambda:InvokeFunctionUrl" \
--principal "cloudfront.amazonaws.com" \
--source-arn "arn:aws:cloudfront::123456789012:distribution/E1PDK09ESKHJWT" \
--function-name FUNCTION_URL_NAME
```

> **Note**
>
> If you create a distribution and it doesn't have permission to your Lambda function URL, you can choose **Copy CLI command** from the CloudFront console, and then enter this

command from your command line terminal. For more information, see Granting function access to Amazon Web Services in the *Amazon Lambda Developer Guide*.

Creating the OAC

To create an OAC, you can use the Amazon Web Services Management Console, Amazon CloudFormation, the Amazon CLI, or the CloudFront API.

Console

To create an OAC

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Origin access**.
3. Choose **Create control setting**.
4. On the **Create new OAC** form, do the following:
   a. Enter a **Name** and (optionally) a **Description** for the OAC.
   b. For **Signing behavior**, we recommend that you leave the default setting (**Sign requests (recommended)**). For more information, see the section called "Advanced settings for origin access control".
5. For **Origin type**, choose **Lambda**.
6. Choose **Create**.

> **Tip**
>
> After you create the OAC, make note of the **Name**. You need this in the following procedure.

To add an origin access control to a Lambda function URL in a distribution

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose a distribution with a Lambda function URL that you want to add the OAC to, then choose the **Origins** tab.

3. Select the Lambda function URL that you want to add the OAC to, and then choose **Edit**.
4. Select **HTTPS only** for your origin's **Protocol**.
5. From the **Origin access control** dropdown, choose the OAC name that you want to use.
6. Choose **Save changes**.

The distribution starts deploying to all of the CloudFront edge locations. When an edge location receives the new configuration, it signs all requests that it sends to the Lambda function URL.

CloudFormation

To create an OAC with Amazon CloudFormation, use the `AWS::CloudFront::OriginAccessControl` resource type. The following example shows the Amazon CloudFormation template syntax, in YAML format, for creating an OAC.

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
      Description: An optional description for the origin access control
      Name: ExampleOAC
      OriginAccessControlOriginType: lambda
      SigningBehavior: always
      SigningProtocol: sigv4
```

For more information, see AWS::CloudFront::OriginAccessControl in the *Amazon CloudFormation User Guide*.

CLI

To create an origin access control with the Amazon Command Line Interface (Amazon CLI), use the **aws cloudfront create-origin-access-control** command. You can use an input file to provide the input parameters for the command, rather than specifying each individual parameter as command line input.

To create an origin access control (CLI with input file)

1. Use the following command to create a file that's named `origin-access-control.yaml`. This file contains all of the input parameters for the **create-origin-access-control** command.

```
aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
 origin-access-control.yaml
```

2. Open the `origin-access-control.yaml` file that you just created. Edit the file to add a name for the OAC, a description (optional), and change the `SigningBehavior` to `always`. Then save the file.

   For information about other OAC settings, see the section called "Advanced settings for origin access control".

3. Use the following command to create the origin access control using the input parameters from the `origin-access-control.yaml` file.

   ```
   aws cloudfront create-origin-access-control --cli-input-yaml file://origin-
   access-control.yaml
   ```

   Make note of the `Id` value in the command output. You need it to add the OAC to a Lambda function URL in a CloudFront distribution.

To attach an OAC to a Lambda function URL in an existing distribution (CLI with input file)

1. Use the following command to save the distribution configuration for the CloudFront distribution that you want to add the OAC to. The distribution must have a Lambda function URL as the origin.

   ```
   aws cloudfront get-distribution-config --id <CloudFront distribution ID> --
   output yaml > dist-config.yaml
   ```

2. Open the file that's named `dist-config.yaml` that you just created. Edit the file, making the following changes:

   - In the `Origins` object, add the OAC's ID to the field that's named `OriginAccessControlId`.
   - Remove the value from the field that's named `OriginAccessIdentity`, if one exists.
   - Rename the `ETag` field to `IfMatch`, but don't change the field's value.

   Save the file when finished.

3. Use the following command to update the distribution to use the origin access control.

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-
input-yaml file://dist-config.yaml
```

The distribution starts deploying to all of the CloudFront edge locations. When an edge location receives the new configuration, it signs all requests that it sends to the Lambda function URL.

API

To create an OAC with the CloudFront API, use CreateOriginAccessControl. For more information about the fields that you specify in this API call, see the API reference documentation for your Amazon SDK or other API client.

After you create an OAC you can attach it to a Lambda function URL in a distribution, using one of the following API calls:

- To attach it to an existing distribution, use UpdateDistribution.
- To attach it to a new distribution, use CreateDistribution.

For both of these API calls, provide the OAC ID in the `OriginAccessControlId` field, inside an origin. For more information about the other fields that you specify in these API calls, see and the API reference documentation for your Amazon SDK or other API client.

Advanced settings for origin access control

The CloudFront OAC feature includes advanced settings that are intended only for specific use cases. Use the recommended settings unless you have a specific need for the advanced settings.

OAC contains a setting named **Signing behavior** (in the console), or `SigningBehavior` (in the API, CLI, and Amazon CloudFormation). This setting provides the following options:

Always sign origin requests (recommended setting)

We recommend using this setting, named **Sign requests (recommended)** in the console, or `always` in the API, CLI, and Amazon CloudFormation. With this setting, CloudFront always signs all requests that it sends to the Lambda function URL.

Never sign origin requests

This setting is named **Do not sign requests** in the console, or `never` in the API, CLI, and Amazon CloudFormation. Use this setting to turn off OAC for all origins in all distributions that use this OAC. This can save time and effort compared to removing an OAC from all origins and distributions that use it, one by one. With this setting, CloudFront doesn't sign any requests that it sends to the Lambda function URL.

> **Warning**
>
> To use this setting, the Lambda function URL must be publicly accessible. If you use this setting with a Lambda function URL that's not publicly accessible, CloudFront can't access the origin. The Lambda function URL returns errors to CloudFront and CloudFront passes those errors on to viewers. For more information, see the example Lambda policy for Policies and Permissions in Lambda in the *Amazon Lambda User Guide*.

Don't override the viewer (client) `Authorization` header

This setting is named **Do not override authorization header** in the console, or `no-override` in the API, CLI, and Amazon CloudFormation. Use this setting when you want CloudFront to sign origin requests only when the corresponding viewer request does not include an `Authorization` header. With this setting, CloudFront passes on the `Authorization` header from the viewer request when one is present, but signs the origin request (adding its own `Authorization` header) when the viewer request doesn't include an `Authorization` header.

> **Warning**
>
> To pass along the `Authorization` header from the viewer request, you *must* add the `Authorization` header to a cache policy for all cache behaviors that use Lambda function URLs associated with this origin access control.

Restricting access to an Amazon Simple Storage Service origin

CloudFront provides two ways to send authenticated requests to an Amazon S3 origin: *origin access control* (OAC) and *origin access identity* (OAI). OAC helps you secure your origins, such as for Amazon S3. We recommend using OAC because it supports:

- All Amazon S3 buckets in all Amazon Web Services Regions, including opt-in Regions launched after December 2022
- Amazon S3 server-side encryption with Amazon KMS (SSE-KMS)
- Dynamic requests (PUT and DELETE) to Amazon S3

Origin access identity (OAI) doesn't work for the scenarios in the preceding list, or it requires extra workarounds in those scenarios. The following topics describe how to use origin access control (OAC) with an Amazon S3 origin. For information about how to migrate from origin access identity (OAI) to origin access control (OAC), see the section called "Migrating from origin access identity (OAI) to origin access control (OAC)".

Notes

- When you use CloudFront OAC with Amazon S3 bucket origins, you must set **Amazon S3 Object Ownership** to **Bucket owner enforced**, the default for new Amazon S3 buckets. If you require ACLs, use the **Bucket owner preferred** setting to maintain control over objects uploaded via CloudFront.
- If your origin is an Amazon S3 bucket configured as a website endpoint, you must set it up with CloudFront as a custom origin. That means you can't use OAC (or OAI). OAC doesn't support origin redirect by using Lambda@Edge.

Topics

- the section called "Creating a new origin access control"
- the section called "Deleting a distribution with an OAC attached to an S3 bucket"
- the section called "Migrating from origin access identity (OAI) to origin access control (OAC)"
- the section called "Advanced settings for origin access control"

Creating a new origin access control

Complete the steps described in the following topics to set up a new origin access control in CloudFront.

Topics

- Prerequisites

- Giving the origin access control permission to access the S3 bucket
- Creating the origin access control

Prerequisites

Before you create and set up origin access control (OAC), you must have a CloudFront distribution with an Amazon S3 bucket origin. This origin must be a regular S3 bucket, not a bucket configured as a website endpoint. For more information about setting up a CloudFront distribution with an S3 bucket origin, see the section called "Getting started with a basic distribution".

> **Note**
>
> When you use OAC to secure your S3 bucket origin, communication between CloudFront and Amazon S3 is *always* through HTTPS, regardless of your specific settings.

Giving the origin access control permission to access the S3 bucket

Before you create an origin access control (OAC) or set it up in a CloudFront distribution, make sure the OAC has permission to access the S3 bucket origin. Do this after creating a CloudFront distribution, but before adding the OAC to the S3 origin in the distribution configuration.

To give the OAC permission to access the S3 bucket, use an S3 bucket policy to allow the CloudFront service principal (`cloudfront.amazonaws.com`) to access the bucket. Use a `Condition` element in the policy to allow CloudFront to access the bucket only when the request is on behalf of the CloudFront distribution that contains the S3 origin.

For information about adding or modifying a bucket policy, see Adding a bucket policy using the Amazon S3 console in the *Amazon S3 User Guide*.

The following are examples of S3 bucket policies that allow a CloudFront OAC to access an S3 origin.

Example S3 bucket policy that allows read-only access to a CloudFront OAC

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "AllowCloudFrontServicePrincipalReadOnly",
        "Effect": "Allow",
```

```
        "Principal": {
            "Service": "cloudfront.amazonaws.com"
        },
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::<S3 bucket name>/*",
        "Condition": {
            "StringEquals": {
                "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
            }
        }
    }
}
```

Example S3 bucket policy that allows read and write access to a CloudFront OAC

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "AllowCloudFrontServicePrincipalReadWrite",
        "Effect": "Allow",
        "Principal": {
            "Service": "cloudfront.amazonaws.com"
        },
        "Action": [
            "s3:GetObject",
            "s3:PutObject"
        ],
        "Resource": "arn:aws:s3:::<S3 bucket name>/*",
        "Condition": {
            "StringEquals": {
                "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
            }
        }
    }
}
```

SSE-KMS

If the objects in the S3 bucket origin are encrypted using server-side encryption with Amazon Key Management Service (SSE-KMS), you must make sure that the OAC has permission to use the Amazon KMS key. To give the OAC permission to use the KMS key, add a statement to the KMS key

policy. For information about how to modify a key policy, see Changing a key policy in the *Amazon Key Management Service Developer Guide*.

The following example shows a KMS key policy statement that allows the OAC to use the KMS key.

Example KMS key policy statement that allows a CloudFront OAC to access a KMS key for SSE-KMS

```
{
    "Sid": "AllowCloudFrontServicePrincipalSSE-KMS",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "cloudfront.amazonaws.com"
        ]
     },
    "Action": [
        "kms:Decrypt",
        "kms:Encrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*",
    "Condition": {
            "StringEquals": {
                "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
            }
        }
}
```

Creating the origin access control

To create an origin access control (OAC), you can use the Amazon Web Services Management Console, Amazon CloudFormation, the Amazon CLI, or the CloudFront API.

Console

To create an origin access control

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Origin access**.

3. Choose **Create control setting**.
4. On the **Create control setting** form, do the following:

   a. In the **Details** pane, enter a **Name** and (optionally) a **Description** for the origin access control.

   b. In the **Settings** pane, we recommend that you leave the default setting (**Sign requests (recommended)**). For more information, see the section called "Advanced settings for origin access control".
5. Choose S3 from the **Origin type** dropdown.
6. Choose **Create**.

   After the OAC is created, make note of the **Name**. You need this in the following procedure.

To add an origin access control to an S3 origin in a distribution

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose a distribution with an S3 origin that you want to add the OAC to, then choose the **Origins** tab.
3. Select the S3 origin that you want to add the OAC to, then choose **Edit**.
4. For **Origin access**, choose **Origin access control settings (recommended)**.
5. From the **Origin access control** dropdown menu, choose the OAC that you want to use.
6. Choose **Save changes**.

The distribution starts deploying to all of the CloudFront edge locations. When an edge location receives the new configuration, it signs all requests that it sends to the S3 bucket origin.

CloudFormation

To create an origin access control (OAC) with Amazon CloudFormation, use the `AWS::CloudFront::OriginAccessControl` resource type. The following example shows the Amazon CloudFormation template syntax, in YAML format, for creating an origin access control.

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
      Description: An optional description for the origin access control
```

```
      Name: ExampleOAC
      OriginAccessControlOriginType: s3
      SigningBehavior: always
      SigningProtocol: sigv4
```

For more information, see AWS::CloudFront::OriginAccessControl in the *Amazon CloudFormation User Guide*.

CLI

To create an origin access control with the Amazon Command Line Interface (Amazon CLI), use the **aws cloudfront create-origin-access-control** command. You can use an input file to provide the input parameters for the command, rather than specifying each individual parameter as command line input.

To create an origin access control (CLI with input file)

1. Use the following command to create a file that's named `origin-access-control.yaml`. This file contains all of the input parameters for the **create-origin-access-control** command.

   ```
   aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
    origin-access-control.yaml
   ```

2. Open the `origin-access-control.yaml` file that you just created. Edit the file to add a name for the OAC, a description (optional), and change the `SigningBehavior` to `always`. Then save the file.

   For information about other OAC settings, see the section called “Advanced settings for origin access control”.

3. Use the following command to create the origin access control using the input parameters from the `origin-access-control.yaml` file.

   ```
   aws cloudfront create-origin-access-control --cli-input-yaml file://origin-
   access-control.yaml
   ```

   Make note of the `Id` value in the command output. You need it to add the OAC to an S3 bucket origin in a CloudFront distribution.

To attach an OAC to an S3 bucket origin in an existing distribution (CLI with input file)

1. Use the following command to save the distribution configuration for the CloudFront distribution that you want to add the OAC to. The distribution must have an S3 bucket origin.

```
aws cloudfront get-distribution-config --id <CloudFront distribution ID> --
output yaml > dist-config.yaml
```

2. Open the file that's named `dist-config.yaml` that you just created. Edit the file, making the following changes:

   - In the `Origins` object, add the OAC's ID to the field that's named `OriginAccessControlId`.
   - Remove the value from the field that's named `OriginAccessIdentity`, if one exists.
   - Rename the `ETag` field to `IfMatch`, but don't change the field's value.

   Save the file when finished.

3. Use the following command to update the distribution to use the origin access control.

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-
input-yaml file://dist-config.yaml
```

The distribution starts deploying to all of the CloudFront edge locations. When an edge location receives the new configuration, it signs all requests that it sends to the S3 bucket origin.

API

To create an origin access control with the CloudFront API, use CreateOriginAccessControl. For more information about the fields that you specify in this API call, see the API reference documentation for your Amazon SDK or other API client.

After you create an origin access control you can attach it to an S3 bucket origin in a distribution, using one of the following API calls:

- To attach it to an existing distribution, use UpdateDistribution.
- To attach it to a new distribution, use CreateDistribution.

For both of these API calls, provide the origin access control ID in the `OriginAccessControlId` field, inside an origin. For more information about the other fields that you specify in these API calls, see Values that you specify when you create or update a distribution and the API reference documentation for your Amazon SDK or other API client.

Deleting a distribution with an OAC attached to an S3 bucket

If you need to delete a distribution with an OAC attached to an S3 bucket, you should delete the distribution before you delete the S3 bucket origin. Alternatively, include the Region in the origin domain name. If this isn't possible, you can remove the OAC from the distribution by switching to public before deletion. For more information, see Deleting a distribution.

Migrating from origin access identity (OAI) to origin access control (OAC)

To migrate from a legacy origin access identity (OAI) to an origin access control (OAC), first update the S3 bucket origin to allow both the OAI and OAC to access the bucket's content. This makes sure that CloudFront never loses access to the bucket during the transition. To allow both OAI and OAC to access an S3 bucket, update the bucket policy to include two statements, one for each kind of principal.

The following example S3 bucket policy allows both an OAI and an OAC to access an S3 origin.

Example S3 bucket policy that allows read-only access to an OAI and an OAC

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipalReadOnly",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<S3 bucket name>/*",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
                }
            }
```

```
        },
        {
            "Sid": "AllowLegacyOAIReadOnly",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity <origin access identity ID>"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<S3 bucket name>/*"
        }
    ]
}
```

After you update the S3 origin's bucket policy to allow access to both OAI and OAC, you can update the distribution configuration to use OAC instead of OAI. For more information, see the section called “Creating a new origin access control”.

After the distribution is fully deployed, you can remove the statement in the bucket policy that allows access to the OAI. For more information, see the section called “Giving the origin access control permission to access the S3 bucket”.

Advanced settings for origin access control

The CloudFront origin access control feature includes advanced settings that are intended only for specific use cases. Use the recommended settings unless you have a specific need for the advanced settings.

Origin access control contains a setting named **Signing behavior** (in the console), or `SigningBehavior` (in the API, CLI, and Amazon CloudFormation). This setting provides the following options:

Always sign origin requests (recommended setting)

We recommend using this setting, named **Sign requests (recommended)** in the console, or `always` in the API, CLI, and Amazon CloudFormation. With this setting, CloudFront always signs all requests that it sends to the S3 bucket origin.

Never sign origin requests

This setting is named **Do not sign requests** in the console, or `never` in the API, CLI, and Amazon CloudFormation. Use this setting to turn off origin access control for all origins in all

distributions that use this origin access control. This can save time and effort compared to removing an origin access control from all origins and distributions that use it, one by one. With this setting, CloudFront does not sign any requests that it sends to the S3 bucket origin.

> **Warning**
>
> To use this setting, the S3 bucket origin must be publicly accessible. If you use this setting with an S3 bucket origin that's not publicly accessible, CloudFront cannot access the origin. The S3 bucket origin returns errors to CloudFront and CloudFront passes those errors on to viewers.

Don't override the viewer (client) `Authorization` header

This setting is named **Do not override authorization header** in the console, or `no-override` in the API, CLI, and Amazon CloudFormation. Use this setting when you want CloudFront to sign origin requests only when the corresponding viewer request does not include an `Authorization` header. With this setting, CloudFront passes on the `Authorization` header from the viewer request when one is present, but signs the origin request (adding its own `Authorization` header) when the viewer request doesn't include an `Authorization` header.

> **Warning**
>
> To pass along the `Authorization` header from the viewer request, you *must* add the `Authorization` header to a cache policy for all cache behaviors that use S3 bucket origins associated with this origin access control.

Using an origin access identity (legacy, not recommended)

Overview of origin access identity

CloudFront *origin access identity* (OAI) provides similar functionality as *origin access control* (OAC), but it doesn't work for all scenarios. This is why we recommend using OAC instead. Specifically, OAI doesn't support:

- Amazon S3 buckets in all Amazon Web Services Regions, including opt-in Regions
- Amazon S3 server-side encryption with Amazon KMS (SSE-KMS)
- Dynamic requests (`PUT`, `POST`, or `DELETE`) to Amazon S3

- New Amazon Web Services Regions launched after December 2022

For information about how to migrating from OAI to OAC, see the section called “Migrating from origin access identity (OAI) to origin access control (OAC)”.

Giving an origin access identity permission to read files in the Amazon S3 bucket

When you create an OAI or add one to a distribution with the CloudFront console, you can automatically update the Amazon S3 bucket policy to give the OAI permission to access your bucket. Alternatively, you can choose to manually create or update the bucket policy. Whichever method you use, you should still review the permissions to make sure that:

- Your CloudFront OAI can access files in the bucket on behalf of viewers who are requesting them through CloudFront.
- Viewers can't use Amazon S3 URLs to access your files outside of CloudFront.

> **Important**
>
> If you configure CloudFront to accept and forward all of the HTTP methods that CloudFront supports, make sure you give your CloudFront OAI the desired permissions. For example, if you configure CloudFront to accept and forward requests that use the `DELETE` method, configure your bucket policy to handle `DELETE` requests appropriately so viewers can delete only files that you want them to.

Using Amazon S3 bucket policies

You can give a CloudFront OAI access to files in an Amazon S3 bucket by creating or updating the bucket policy in the following ways:

- Using the Amazon S3 bucket's **Permissions** tab in the Amazon S3 console.
- Using PutBucketPolicy in the Amazon S3 API.
- Using the CloudFront console. When you add an OAI to your origin settings in the CloudFront console, you can choose **Yes, update the bucket policy** to tell CloudFront to update the bucket policy on your behalf.

If you update the bucket policy manually, make sure that you:

- Specify the correct OAI as the `Principal` in the policy.
- Give the OAI the permissions it needs to access objects on behalf of viewers.

For more information, see the following sections.

Specify an OAI as the `Principal` in a bucket policy

To specify an OAI as the `Principal` in an Amazon S3 bucket policy, use the OAI's Amazon Resource Name (ARN), which includes the OAI's ID. For example:

```
"Principal": {
    "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity <origin
 access identity ID>"
}
```

Find the OAI ID in the CloudFront console under **Security**, **Origin access**, **Identities (legacy)**. Alternatively, use ListCloudFrontOriginAccessIdentities in the CloudFront API.

Give permissions to an OAI

To give the OAI the permissions to access objects in your Amazon S3 bucket, use actions in the policy that relate to specific Amazon S3 API operations. For example, the `s3:GetObject` action allows the OAI to read objects in the bucket. For more information, see the examples in the following section, or see Amazon S3 actions in the *Amazon Simple Storage Service User Guide*.

Amazon S3 bucket policy examples

The following examples show Amazon S3 bucket policies that allow CloudFront OAI to access an S3 bucket.

Find the OAI ID in the CloudFront console under **Security**, **Origin access**, **Identities (legacy)**. Alternatively, use ListCloudFrontOriginAccessIdentities in the CloudFront API.

Example Amazon S3 bucket policy that gives the OAI read access

The following example allows the OAI to read objects in the specified bucket (`s3:GetObject`).

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity <origin access identity ID>"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<S3 bucket name>/*"
        }
    ]
}
```

Example Amazon S3 bucket policy that gives the OAI read and write access

The following example allows the OAI to read and write objects in the specified bucket (`s3:GetObject` and `s3:PutObject`). This allows viewers to upload files to your Amazon S3 bucket through CloudFront.

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity <origin access identity ID>"
            },
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::<S3 bucket name>/*"
        }
    ]
}
```

Using Amazon S3 object ACLs (not recommended)

> **Important**
>
> We recommend using Amazon S3 bucket policies to give an OAI access to an S3 bucket. You can use access control lists (ACLs) as described in this section, but we don't recommend it.

Amazon S3 recommends setting S3 Object Ownership to **bucket owner enforced**, which means that ACLs are disabled for the bucket and the objects in it. When you apply this setting for Object Ownership, you must use bucket policies to give access to the OAI (see the previous section).
This following section is only for legacy use cases that require ACLs.

You can give a CloudFront OAI access to files in an Amazon S3 bucket by creating or updating the file's ACL in the following ways:

- Using the Amazon S3 object's **Permissions** tab in the Amazon S3 console.
- Using PutObjectAcl in the Amazon S3 API.

When you grant access to an OAI using an ACL, you must specify the OAI using its Amazon S3 canonical user ID. In the CloudFront console, you can find this ID under **Security**, **Origin access**, **Identities (legacy)**. If you're using the CloudFront API, use the value of the `S3CanonicalUserId` element that was returned when you created the OAI, or call ListCloudFrontOriginAccessIdentities in the CloudFront API.

Using an origin access identity in Amazon S3 regions that support only signature version 4 authentication

Newer Amazon S3 Regions require that you use Signature Version 4 for authenticated requests. (For the signature versions supported in each Amazon S3 Region, see Amazon Simple Storage Service endpoints and quotas in the *Amazon Web Services General Reference*.) If you're using an origin access identity and if your bucket is in one of the Regions that requires Signature Version 4, note the following:

- `DELETE`, `GET`, `HEAD`, `OPTIONS`, and `PATCH` requests are supported without qualifications.
- `POST` requests are not supported.

Restricting access to Application Load Balancers

For a web application or other content that's served by an Application Load Balancer in Elastic Load Balancing, CloudFront can cache objects and serve them directly to users (viewers), reducing the load on your Application Load Balancer. CloudFront can also help to reduce latency and even absorb some distributed denial of service (DDoS) attacks. However, if users can bypass CloudFront

and access your Application Load Balancer directly, you don't get these benefits. But you can configure Amazon CloudFront and your Application Load Balancer to prevent users from directly accessing the Application Load Balancer. This allows users to access the Application Load Balancer only through CloudFront, ensuring that you get the benefits of using CloudFront.

To prevent users from directly accessing an Application Load Balancer and allow access only through CloudFront, complete these high-level steps:

1. Configure CloudFront to add a custom HTTP header to requests that it sends to the Application Load Balancer.
2. Configure the Application Load Balancer to only forward requests that contain the custom HTTP header.
3. (Optional) Require HTTPS to improve the security of this solution.

For more information, see the following topics. After you complete these steps, users can only access your Application Load Balancer through CloudFront.

Topics

- Configuring CloudFront to add a custom HTTP header to requests
- Configuring an Application Load Balancer to only forward requests that contain a specific header
- (Optional) Improve the security of this solution
- (Optional) Limit access to origin by using the Amazon-managed prefix list for CloudFront

Configuring CloudFront to add a custom HTTP header to requests

You can configure CloudFront to add a custom HTTP header to the requests that it sends to your origin (in this case, an Application Load Balancer).

> **Important**
>
> This use case relies on keeping the custom header name and value secret. If the header name and value are not secret, other HTTP clients could potentially include them in requests that they send directly to the Application Load Balancer. This can cause the Application Load Balancer to behave as though the requests came from CloudFront when they did not. To prevent this, keep the custom header name and value secret.

You can configure CloudFront to add a custom HTTP header to origin requests with the CloudFront console, Amazon CloudFormation, or the CloudFront API.

To add a custom HTTP header (CloudFront console)

In the CloudFront console, use the **Origin Custom Headers** setting in **Origin Settings**. Enter the **Header Name** and its **Value**, as shown in the following example.

> **Note**
>
> The header name and value in this example are just for demonstration. In production, use randomly generated values. Treat the header name and value as a secure credential, like a user name and password.

You can edit the **Origin Custom Headers** setting when you create or edit an origin for an existing CloudFront distribution, and when you create a new distribution. For more information, see Updating a distribution and Creating a distribution.

To add a custom HTTP header (Amazon CloudFormation)

In an Amazon CloudFormation template, use the `OriginCustomHeaders` property, as shown in the following example.

> **Note**
>
> The header name and value in this example are just for demonstration. In production, use randomly generated values. Treat the header name and value as a secure credential, like a user name and password.

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  TestDistribution:
    Type: 'AWS::CloudFront::Distribution'
    Properties:
```

```
      DistributionConfig:
        Origins:
          - DomainName: app-load-balancer.example.com
            Id: Example-ALB
            CustomOriginConfig:
              OriginProtocolPolicy: https-only
              OriginSSLProtocols:
                - TLSv1.2
            OriginCustomHeaders:
               - HeaderName: X-Custom-Header
                 HeaderValue: random-value-1234567890
        Enabled: 'true'
        DefaultCacheBehavior:
          TargetOriginId: Example-ALB
          ViewerProtocolPolicy: allow-all
          CachePolicyId: 658327ea-f89d-4fab-a63d-7e88639e58f6
        PriceClass: PriceClass_All
        ViewerCertificate:
          CloudFrontDefaultCertificate: 'true'
```

For more information, see the Origin and OriginCustomHeader properties in the *Amazon CloudFormation User Guide*.

To add a custom HTTP header (CloudFront API)

In the CloudFront API, use the `CustomHeaders` object inside `Origin`. For more information, see CreateDistribution and UpdateDistribution in the *Amazon CloudFront API Reference*, and the documentation for your SDK or other API client.

There are some header names that you can't specify as origin custom headers. For more information, see Custom headers that CloudFront can't add to origin requests.

Configuring an Application Load Balancer to only forward requests that contain a specific header

After you configure CloudFront to add a custom HTTP header to the requests that it sends to your Application Load Balancer (see the previous section), you can configure the load balancer to only forward requests that contain this custom header. You do this by adding a new rule and modifying the default rule in your load balancer's listener.

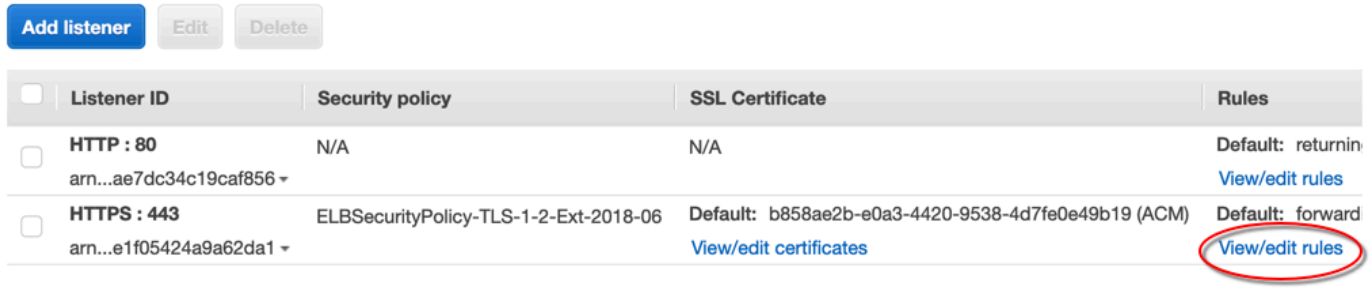
Prerequisites

To use the following procedures, you need an Application Load Balancer with at least one listener. If you haven't created one yet, see Create an Application Load Balancer in the *User Guide for Application Load Balancers*.

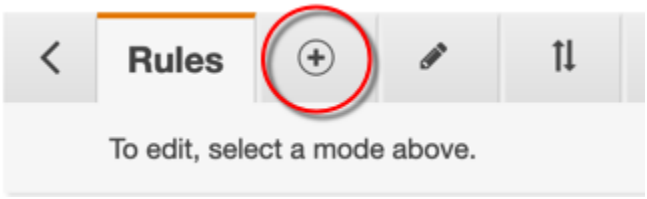
The following procedures modify an HTTPS listener. You can use the same process to modify an HTTP listener.

To update the rules in an Application Load Balancer listener

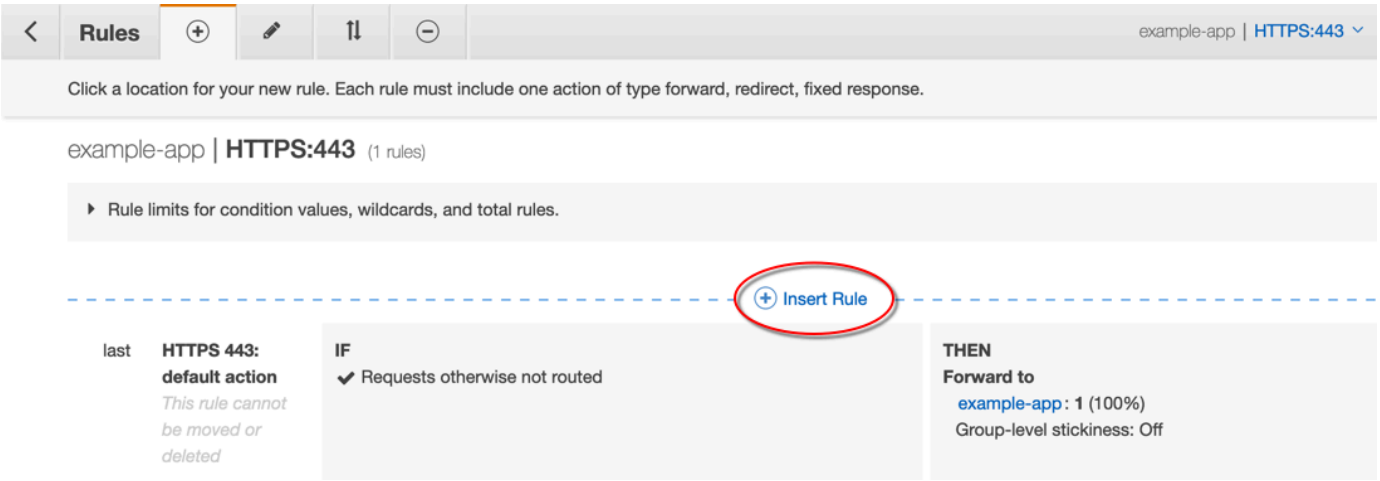
1. Open the **Load Balancers** page in the Amazon EC2 console.
2. Choose the load balancer that is the origin for your CloudFront distribution, then choose the **Listeners** tab.
3. For the listener that you are modifying, choose **View/edit rules**.



4. Choose the icon to add rules.



5. Choose **Insert Rule**.

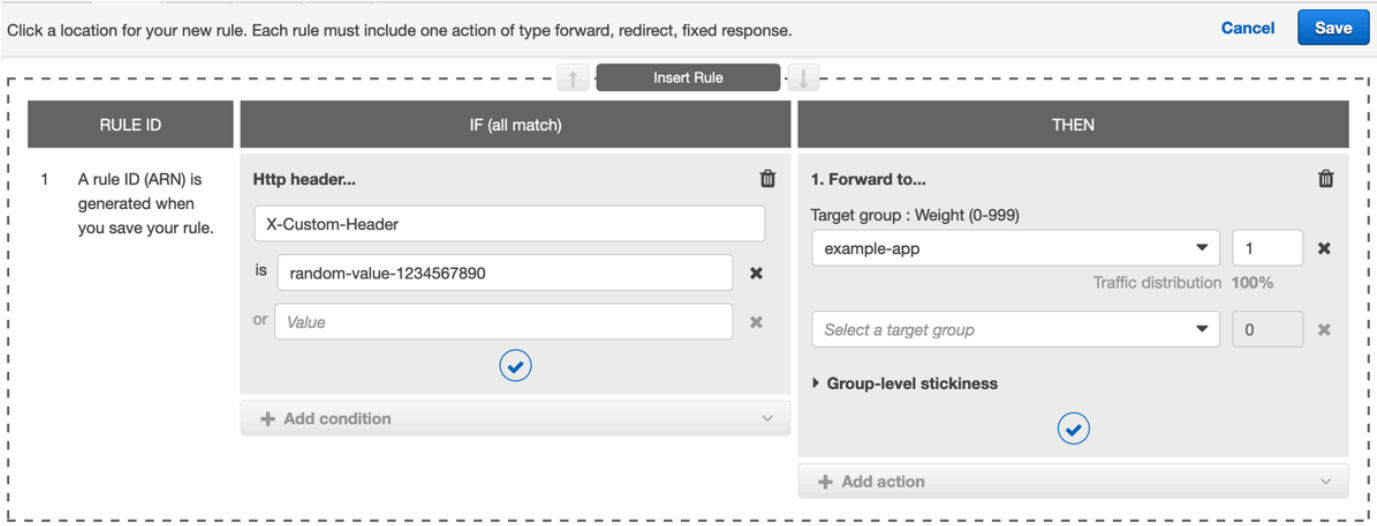


6. For the new rule, do the following:

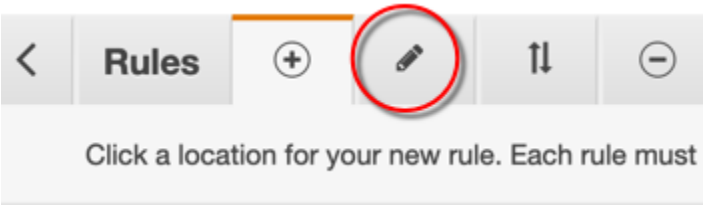
   a. Choose **Add condition** and then choose **Http header**. Specify the HTTP header name and value that you added as an origin custom header in CloudFront.

   b. Choose **Add action** and then choose **Forward to**. Choose the target group where you want to forward requests.

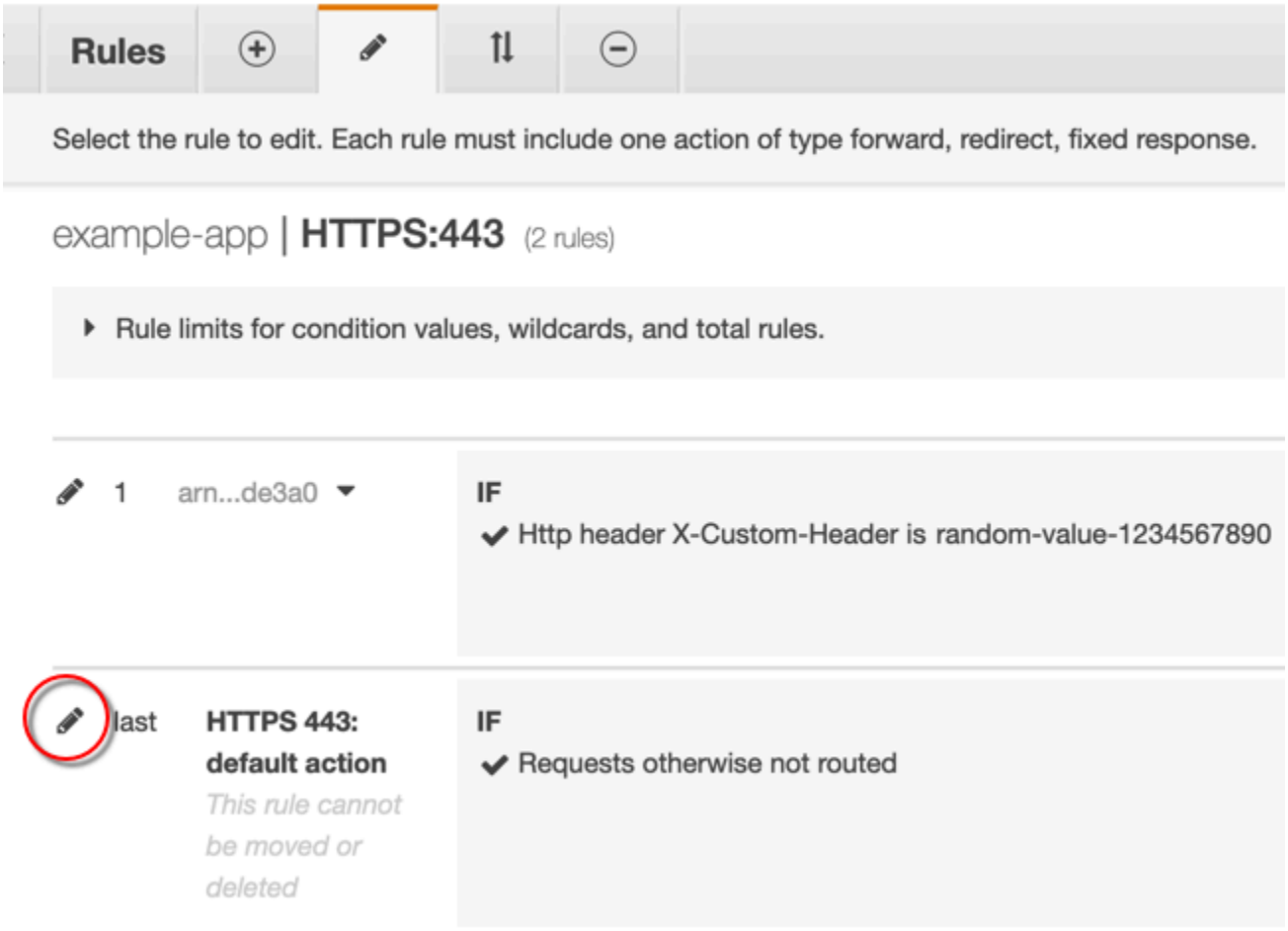
   c. Choose **Save** to create the new rule.



7. Choose the icon to edit rules.



8. Choose the edit icon for the default rule.



9. For the default rule, do the following:

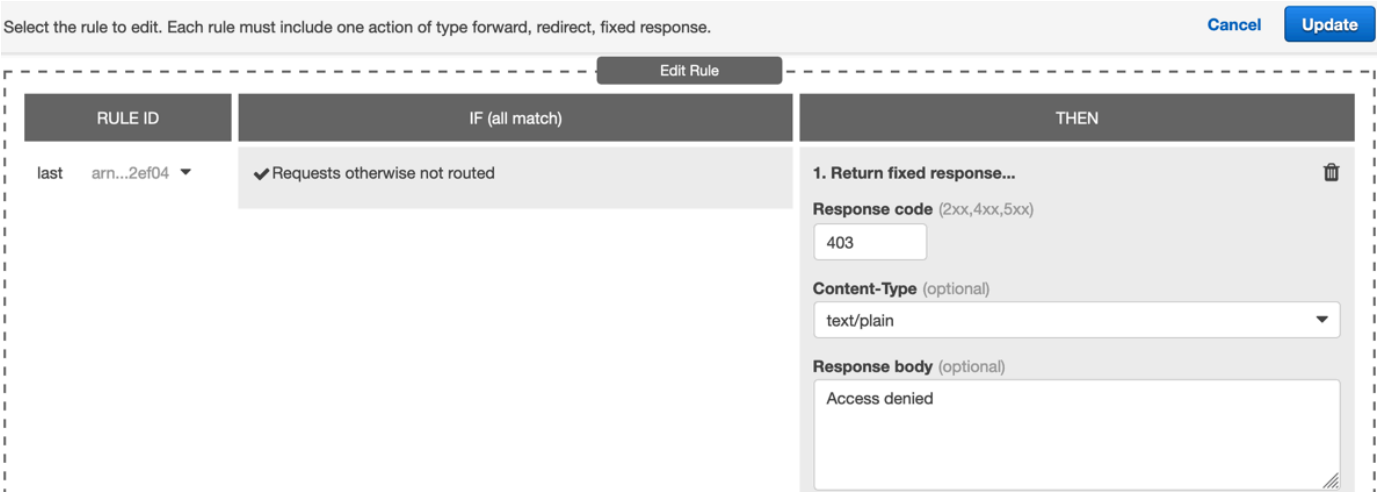
   a. Delete the default action.

   b. Choose **Add action** and then choose **Return fixed response**.

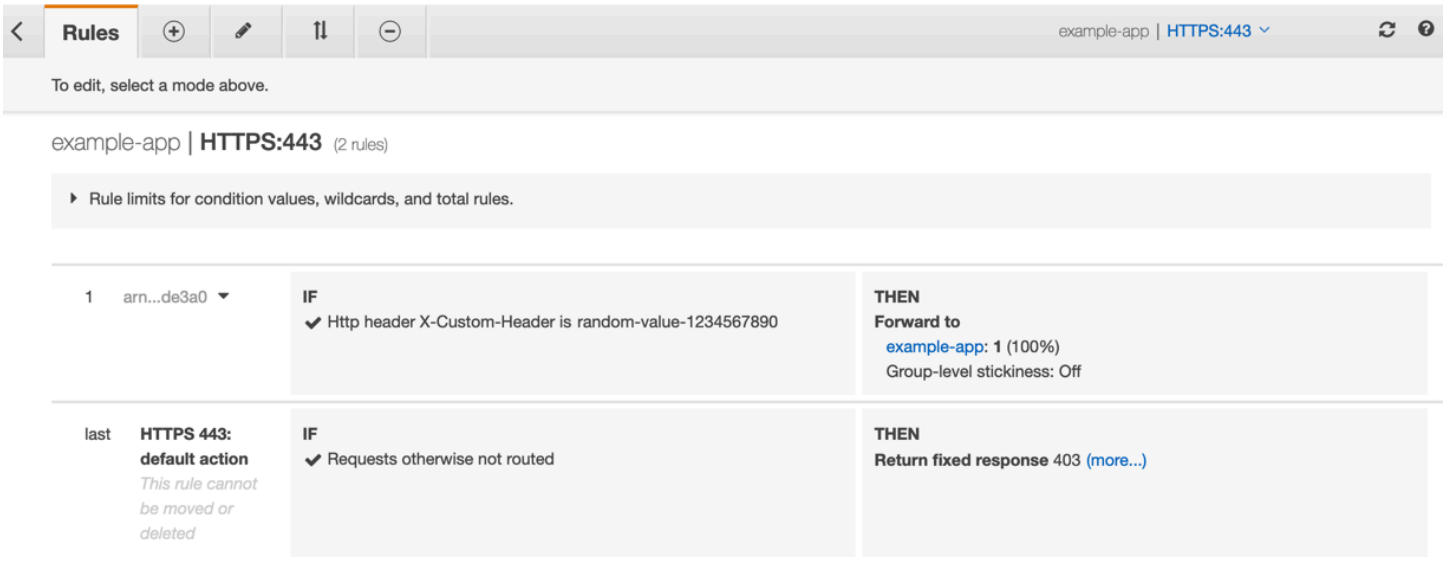
   c. For **Response code**, enter **`403`**.

   d. For **Response body**, enter **`Access denied`**.

   e. Choose **Update** to update the default rule.



After you complete these steps, your load balancer listener has two rules, as shown in the following image. The first rule forwards requests that contain the HTTP header (requests that come from CloudFront). The second rule sends a fixed response to all other requests (requests that don't come from CloudFront).



You can verify that the solution works by sending a request to your CloudFront distribution and one to your Application Load Balancer. The request to CloudFront returns your web application or content, and the one sent directly to your Application Load Balancer returns a `403` response with the plain text message `Access denied`.

(Optional) Improve the security of this solution

To improve the security of this solution, you can configure your CloudFront distribution to always use HTTPS when sending requests to your Application Load Balancer. Remember, this solution only works if you keep the custom header name and value secret. Using HTTPS can help prevent an eavesdropper from discovering the header name and value. We also recommend rotating the header name and value periodically.

Use HTTPS for origin requests

To configure CloudFront to use HTTPS for origin requests, set the **Origin Protocol Policy** setting to **HTTPS Only**. This setting is available in the CloudFront console, Amazon CloudFormation, and the CloudFront API. For more information, see Protocol (custom origins only).

The following also applies when you configure CloudFront to use HTTPS for origin requests:

- You must configure CloudFront to forward the `Host` header to the origin with the origin request policy. You can use the AllViewer managed origin request policy.
- Make sure that your Application Load Balancer has an HTTPS listener (as shown in the preceding section). For more information, see Create an HTTPS listener in the *User Guide for Application Load Balancers*. Using an HTTPS listener requires you to have an SSL/TLS certificate that matches the domain name that's routed to your Application Load Balancer.
- SSL/TLS certificates for CloudFront can only be requested (or imported) in the `us-east-1` Amazon Web Services Region in Amazon Certificate Manager (ACM). Because CloudFront is a global service, it automatically distributes the certificate from the `us-east-1` Region to all Regions associated with your CloudFront distribution.
  - For example, if you have an Application Load Balancer (ALB) in the `ap-southeast-2` Region, you must configure SSL/TLS certificates in both the `ap-southeast-2` Region (for using HTTPS between CloudFront and the ALB origin) and the `us-east-1` Region (for using HTTPS between viewers and CloudFront). Both certificates should match the domain name that is routed to your Application Load Balancer. For more information, see Amazon Web Services Region for Amazon Certificate Manager.
- If the end users (also known as *viewers*, or *clients*) of your web application can use HTTPS, you can also configure CloudFront to prefer (or even require) HTTPS connections from the end users. To do this, use the **Viewer Protocol Policy** setting. You can set it to redirect end users from HTTP to HTTPS, or to reject requests that use HTTP. This setting is available in the CloudFront console,

Amazon CloudFormation, and the CloudFront API. For more information, see Viewer protocol policy.

Rotate the header name and value

In addition to using HTTPS, we also recommend rotating the header name and value periodically. The high-level steps for doing this are as follows:

1. Configure CloudFront to add an additional custom HTTP header to requests that it sends to the Application Load Balancer.
2. Update the Application Load Balancer listener rule to forward requests that contain this additional custom HTTP header.
3. Configure CloudFront to stop adding the original custom HTTP header to requests that it sends to the Application Load Balancer.
4. Update the Application Load Balancer listener rule to stop forwarding requests that contain the original custom HTTP header.

For more information about accomplishing these steps, see the preceding sections.

(Optional) Limit access to origin by using the Amazon-managed prefix list for CloudFront

To further restrict access to your Application Load Balancer, you can configure the security group associated with the Application Load Balancer so that it only accept traffic from CloudFront when the service is using an Amazon-managed prefix list. This prevents traffic that doesn't originate from CloudFront from reaching your Application Load Balancer at the network layer (layer 3) or transport layer (layer 4).

For more information, see the Limit access to your origins using the Amazon-managed prefix list for Amazon CloudFront blog post.

Restricting the geographic distribution of your content

You can use *geographic restrictions*, sometimes known as *geo blocking*, to prevent users in specific geographic locations from accessing content that you're distributing through an Amazon CloudFront distribution. To use geographic restrictions, you have two options:

- Use the CloudFront geographic restrictions feature. Use this option to restrict access to all of the files that are associated with a distribution and to restrict access at the country level.
- Use a third-party geolocation service. Use this option to restrict access to a subset of the files that are associated with a distribution or to restrict access at a finer granularity than the country level.

Topics

- Using CloudFront geographic restrictions
- Using a third-party geolocation service

Using CloudFront geographic restrictions

When a user requests your content, CloudFront typically serves the requested content regardless of where the user is located. If you need to prevent users in specific countries from accessing your content, you can use the CloudFront geographic restrictions feature to do one of the following:

- Grant permission to your users to access your content only if they're in one of the approved countries on your allowlist.
- Prevent your users from accessing your content if they're in one of the banned countries on your denylist.

For example, if a request comes from a country where you are not authorized to distribute your content, you can use CloudFront geographic restrictions to block the request.

> **Note**
>
> CloudFront determines the location of your users by using a third-party database. The accuracy of the mapping between IP addresses and countries varies by Region. Based on recent tests, the overall accuracy is 99.8%. If CloudFront can't determine a user's location, CloudFront serves the content that the user has requested.

Here's how geographic restrictions work:

1. Suppose you have rights to distribute your content only in Liechtenstein. You update your CloudFront distribution to add an allowlist that contains only Liechtenstein. (Alternatively, you could add a denylist that contains every country except Liechtenstein.)
2. A user in Monaco requests your content, and DNS routes the request to a CloudFront edge location in Milan, Italy.
3. The edge location in Milan looks up your distribution and determines that the user in Monaco does not have permission to download your content.
4. CloudFront returns an HTTP status code `403 (Forbidden)` to the user.

You can optionally configure CloudFront to return a custom error message to the user, and you can specify how long you want CloudFront to cache the error response for the requested file. The default value is 10 seconds. For more information, see Creating a custom error page for specific HTTP status codes.

Geographic restrictions apply to an entire distribution. If you need to apply one restriction to part of your content and a different restriction (or no restriction) to another part of your content, you must create separate CloudFront distributions or use a third-party geolocation service.

If you enable CloudFront standard logs (access logs), you can identify the requests that CloudFront rejected by searching for the log entries in which the value of `sc-status` (the HTTP status code) is `403`. However, using only the standard logs, you can't distinguish a request that CloudFront rejected based on the location of the user from a request that CloudFront rejected because the user didn't have permission to access the file for another reason. If you have a third-party geolocation service such as Digital Element or MaxMind, you can identify the location of requests based on the IP address in the `c-ip` (client IP) column in the access logs. For more information about CloudFront standard logs, see Configuring and using standard logs (access logs).

The following procedure explains how to use the CloudFront console to add geographic restrictions to an existing distribution. For information about how to use the console to create a distribution, see Creating a distribution.

To add geographic restrictions to your CloudFront web distribution (console)

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Distributions**, then choose the distribution that you want to update.

3. Choose the **Security** tab, then choose **Geographic restrictions**.
4. Choose **Edit**.
5. Select **Allow list** to create a list of allowed countries, or **Block list** to create a list of blocked countries.
6. Add the desired countries to the list, then choose **Save changes**.

Using a third-party geolocation service

With the CloudFront geographic restrictions feature, you control distribution of your content at the country level for all files that you're distributing with a given web distribution. If you have a use case for geographic restrictions where the restrictions don't follow country boundaries, or if you want to restrict access to only some of the files that you're serving by a given distribution, you can combine CloudFront with a third-party geolocation service. This provides you with control over your content based not only on country but also based on city, ZIP, or postal code, or even latitude and longitude.

When you're using a third-party geolocation service, we recommend that you use CloudFront signed URLs, with which you can specify an expiration date and time after which the URL is no longer valid. In addition, we recommend that you use an Amazon S3 bucket as your origin because you can then use a CloudFront origin access control to prevent users from accessing your content directly from the origin. For more information about signed URLs and origin access control, see Serving private content with signed URLs and signed cookies.

The following steps explain how to control access to your files by using a third-party geolocation service.

To use a third-party geolocation service to restrict access to files in a CloudFront distribution

1. Get an account with a geolocation service.
2. Upload your content to an Amazon S3 bucket.
3. Configure Amazon CloudFront and Amazon S3 to serve private content. For more information, see Serving private content with signed URLs and signed cookies.
4. Write your web application to do the following:

   - Send the IP address for each user request to the geolocation service.
   - Evaluate the return value from the geolocation service to determine whether the user is in a location where you want CloudFront to distribute your content.

- If you want to distribute your content to the user's location, generate a signed URL for your CloudFront content. If you don't want to distribute content to that location, return HTTP status code `403 (Forbidden)` to the user. Alternatively, you can configure CloudFront to return a custom error message. For more information, see the section called "Creating a custom error page for specific HTTP status codes".

For more information, refer to the documentation for the geolocation service that you're using.

You can use a web server variable to get the IP addresses of the users who are visiting your website. Note the following caveats:

- If your web server is not connected to the internet through a load balancer, you can use a web server variable to get the remote IP address. However, this IP address isn't always the user's IP address. It can also be the IP address of a proxy server, depending on how the user is connected to the internet.
- If your web server is connected to the internet through a load balancer, a web server variable might contain the IP address of the load balancer, not the IP address of the user. In this configuration, we recommend that you use the last IP address in the `X-Forwarded-For` HTTP header. This header typically contains more than one IP address, most of which are for proxies or load balancers. The last IP address in the list is the one most likely to be associated with the user's geographic location.

If your web server is not connected to a load balancer, we recommend that you use web server variables instead of the `X-Forwarded-For` header to avoid IP address spoofing.

Using field-level encryption to help protect sensitive data

With Amazon CloudFront, you can enforce secure end-to-end connections to origin servers by using HTTPS. Field-level encryption adds an additional layer of security that lets you protect specific data throughout system processing so that only certain applications can see it.

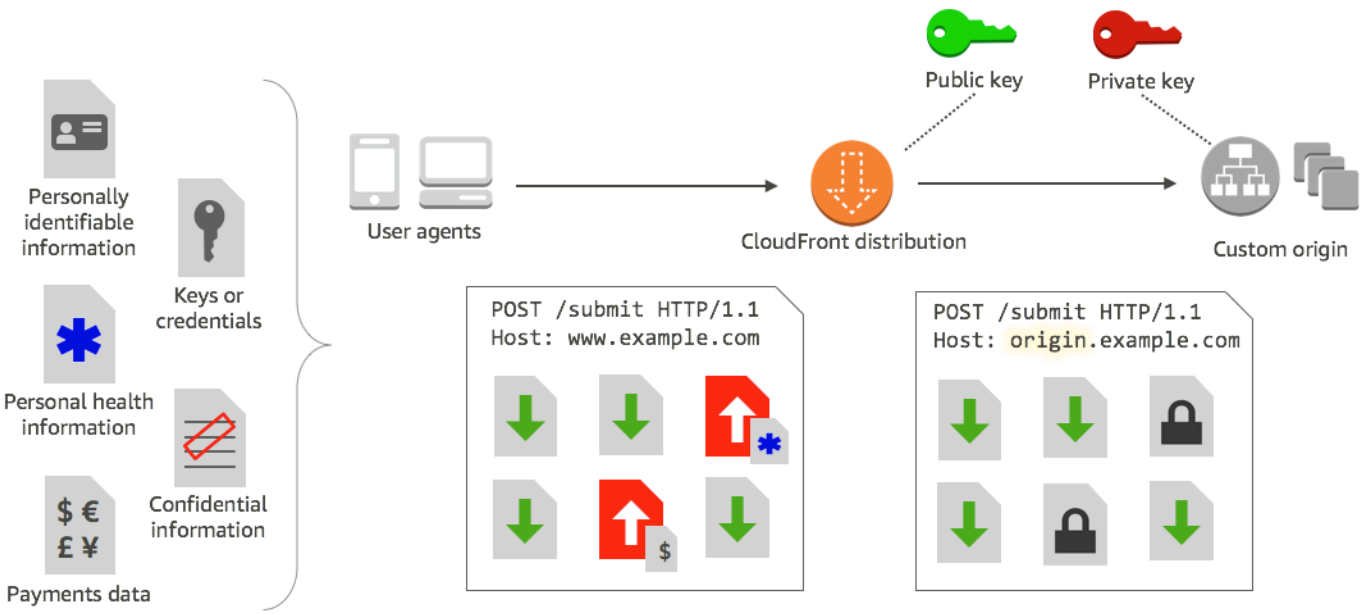
Field-level encryption allows you to enable your users to securely upload sensitive information to your web servers. The sensitive information provided by your users is encrypted at the edge, close to the user, and remains encrypted throughout your entire application stack. This encryption

ensures that only applications that need the data—and have the credentials to decrypt it—are able to do so.

To use field-level encryption, when you configure your CloudFront distribution, specify the set of fields in POST requests that you want to be encrypted, and the public key to use to encrypt them. You can encrypt up to 10 data fields in a request. (You can't encrypt all of the data in a request with field-level encryption; you must specify individual fields to encrypt.)

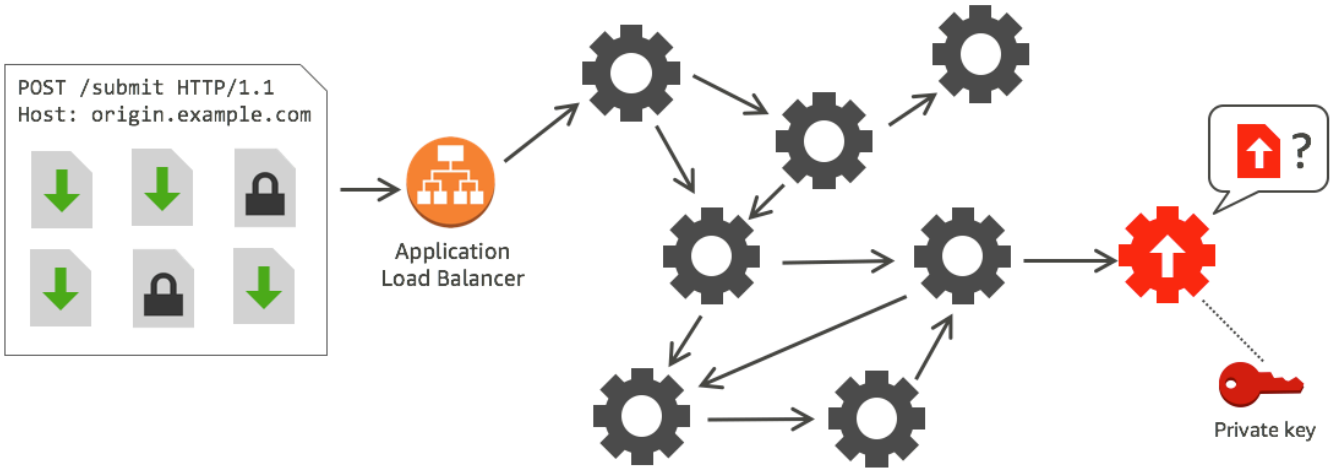
When the HTTPS request with field-level encryption is forwarded to the origin, and the request is routed throughout your origin application or subsystem, the sensitive data is still encrypted, reducing the risk of a data breach or accidental data loss of the sensitive data. Components that need access to the sensitive data for business reasons, such as a payment processing system needing access to a credit number, can use the appropriate private key to decrypt and access the data.

> **Note**
>
> To use field-level encryption, your origin must support chunked encoding.



CloudFront field-level encryption uses asymmetric encryption, also known as public key encryption. You provide a public key to CloudFront, and all sensitive data that you specify is

encrypted automatically. The key you provide to CloudFront cannot be used to decrypt the encrypted values; only your private key can do that.



Topics

- Overview of field-level encryption
- Setting up field-level encryption
- Decrypting data fields at your origin

Overview of field-level encryption

The following steps provide an overview of setting up field-level encryption. For specific steps, see Setting up field-level encryption.

1. **Get a public key-private key pair.** You must obtain and add the public key before you start setting up field-level encryption in CloudFront.
2. **Create a field-level encryption profile.** Field-level encryption profiles, which you create in CloudFront, define the fields that you want to be encrypted.
3. **Create a field-level encryption configuration.** A configuration specifies the profiles to use, based on the content type of the request or a query argument, for encrypting specific data fields. You can also choose the request-forwarding behavior options that you want for different scenarios. For example, you can set the behavior for when the profile name specified by the query argument in a request URL doesn't exist in CloudFront.

4. **Link to a cache behavior.** Link the configuration to a cache behavior for a distribution, to specify when CloudFront should encrypt data.

Setting up field-level encryption

Follow these steps to get started using field-level encryption. To learn about quotas (formerly known as limits) on field-level encryption, see Quotas.

- Step 1: Create an RSA key pair
- Step 2: Add your public key to CloudFront
- Step 3: Create a profile for field-level encryption
- Step 4: Create a configuration
- Step 5: Add a configuration to a cache behavior

Step 1: Create an RSA key pair

To get started, you must create an RSA key pair that includes a public key and a private key. The public key enables CloudFront to encrypt data, and the private key enables components at your origin to decrypt the fields that have been encrypted. You can use OpenSSL or another tool to create a key pair. The key size must be 2048 bits.

For example, if you're using OpenSSL, you can use the following command to generate a key pair with a length of 2048 bits and save it in the file `private_key.pem`:

```
openssl genrsa -out private_key.pem 2048
```

The resulting file contains both the public and the private key. To extract the public key from that file, run the following command:

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

The public key file (`public_key.pem`) contains the encoded key value that you paste in the following step.

Step 2: Add your public key to CloudFront

After you get your RSA key pair, add your public key to CloudFront.

To add your public key to CloudFront (console)

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Public key**.
3. Choose **Add public key**.
4. For **Key name**, type a unique name for the key. The name can't have spaces and can include only alphanumeric characters, underscores (_), and hyphens (-). The maximum number of characters is 128.
5. For **Key value**, paste the encoded key value for your public key, including the `-----BEGIN PUBLIC KEY-----` and `-----END PUBLIC KEY-----` lines.
6. For **Comment**, add an optional comment. For example, you could include the expiration date for the public key.
7. Choose **Add**.

You can add more keys to use with CloudFront by repeating the steps in the procedure.

Step 3: Create a profile for field-level encryption

After you add at least one public key to CloudFront, create a profile that tells CloudFront which fields to encrypt.

To create a profile for field-level encryption (console)

1. In the navigation pane, choose **Field-level encryption**.
2. Choose **Create profile**.
3. Fill in the following fields:

   Profile name

   Type a unique name for the profile. The name can't have spaces and can include only alphanumeric characters, underscores (_), and hyphens (-). The maximum number of characters is 128.

   Public key name

   In the drop-down list, choose the name of a public key that you added to CloudFront in step 2. CloudFront uses the key to encrypt the fields that you specify in this profile.

Provider name

Type a phrase to help identify the key, such as the provider where you got the key pair. This information, along with the private key, is needed when applications decrypt data fields. The provider name can't have spaces and can include only alphanumeric characters, colons (:), underscores (_), and hyphens (-). The maximum number of characters is 128.

Field name pattern to match

Type the names of the data fields, or patterns that identify data field names in the request, that you want CloudFront to encrypt. Choose the + option to add all the fields that you want to encrypt with this key.

For the field name pattern, you can type the entire name of the data field, like DateOfBirth, or just the first part of the name with a wildcard character (*), like CreditCard*. The field name pattern must include only alphanumeric characters, square brackets ([and]), periods (.), underscores (_), and hyphens (-), in addition to the optional wildcard character (*).

Make sure that you don’t use overlapping characters for different field name patterns. For example, if you have a field name pattern of ABC*, you can’t add another field name pattern that is AB*. In addition, field names are case-sensitive and the maximum number of characters that you can use is 128.

Comment

(Optional) Type a comment about this profile. The maximum number of characters that you can use is 128.

4. After you fill in the fields, choose **Create profile**.
5. If you want to add more profiles, choose **Add profile**.

Step 4: Create a configuration

After you create one or more field-level encryption profiles, create a configuration that specifies the content type of the request that includes the data to be encrypted, the profile to use for encryption, and other options that specify how you want CloudFront to handle encryption.

For example, when CloudFront can’t encrypt the data, you can specify whether CloudFront should block or forward a request to your origin in the following scenarios:

- **When a request's content type isn't in a configuration** – If you haven't added a content type to a configuration, you can specify whether CloudFront should forward the request with that content type to the origin without encrypting data fields, or block the request and return an error.

  > **Note**
  >
  > If you add a content type to a configuration but haven't specified a profile to use with that type, CloudFront always forwards requests with that content type to the origin.

- **When the profile name provided in a query argument is unknown** – When you specify the `fle-profile` query argument with a profile name that doesn't exist for your distribution, you can specify whether CloudFront should send the request to the origin without encrypting data fields, or block the request and return an error.

In a configuration, you can also specify whether providing a profile as a query argument in a URL overrides a profile that you've mapped to the content type for that query. By default, CloudFront uses the profile that you've mapped to a content type, if you specify one. This lets you have a profile that's used by default but decide for certain requests that you want to enforce a different profile.

So, for example, you might specify (in your configuration) **`SampleProfile`** as the query argument profile to use. Then you could use the URL `https://d1234.cloudfront.net?fle-profile=SampleProfile` instead of `https://d1234.cloudfront.net`, to have CloudFront use **`SampleProfile`** for this request, instead of the profile you'd set up for the content type of the request.

You can create up to 10 configurations for a single account, and then associate one of the configurations to the cache behavior of any distribution for the account.

To create a configuration for field-level encryption (console)

1. On the **Field-level encryption** page, choose **Create configuration**.

   Note: If you haven't created at least one profile, you won't see the option to create a configuration.

2. Fill in the following fields to specify the profile to use. (Some fields can't be changed.)

Content type (can't be changed)

The content type is set to `application/x-www-form-urlencoded` and can't be changed.

Default profile ID (optional)

In the drop-down list, choose the profile that you want to map to the content type in the **Content type** field.

Content format (can't be changed)

The content format is set to URLencoded and can't be changed.

3. If you want to change the CloudFront default behavior for the following options, select the appropriate check box.

   Forward request to origin when request's content type is not configured

   Select the check box if you want to allow the request to go to your origin *if you have not specified a profile to use for the content type of the request*.

   Override the profile for a content type with a provided query argument

   Select the check box if you want to allow a profile provided in a query argument to *override the profile that you've specified for a content type*.

4. If you select the check box to allow a query argument to override the default profile, you must complete the following additional fields for the configuration. You can create up to five of these query argument mappings to use with queries.

   Query argument

   Type the value that you want to include in URLs for the `fle-profile` query argument. This value tells CloudFront to use the profile ID (that you specify in the next field) associated with this query argument for field-level encryption for this query.

   The maximum number of characters that you can use is 128. The value can't include spaces, and must use only alphanumeric or the following characters: dash (-), period (.), underscore (_), asterisk (*), plus-sign (+), percent (%).

Profile ID

In the drop-down list, choose the profile that you want to associate with the value that you typed for **Query argument**.

Forward request to origin when the profile specified in a query argument does not exist

Select the check box if you want to allow the request to go to your origin *if the profile specified in a query argument isn't defined in CloudFront*.

Step 5: Add a configuration to a cache behavior

To use field-level encryption, link a configuration to a cache behavior for a distribution by adding the configuration ID as a value for your distribution.

> **Important**
>
> To link a field-level encryption configuration to a cache behavior, the distribution must be configured to always use HTTPS, and to accept HTTP `POST` and `PUT` requests from viewers. That is, the following must be true:
>
> - The cache behavior's **Viewer Protocol Policy** must be set to **Redirect HTTP to HTTPS** or **HTTPS Only**. (In Amazon CloudFormation or the CloudFront API, `ViewerProtocolPolicy` must be set to `redirect-to-https` or `https-only`.)
> - The cache behavior's **Allowed HTTP Methods** must be set to **GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE**. (In Amazon CloudFormation or the CloudFront API, `AllowedMethods` must be set to `GET`, `HEAD`, `OPTIONS`, `PUT`, `POST`, `PATCH`, `DELETE`. These can be specified in any order.)
> - The origin setting's **Origin Protocol Policy** must be set to **Match Viewer** or **HTTPS Only**. (In Amazon CloudFormation or the CloudFront API, `OriginProtocolPolicy` must be set to `match-viewer` or `https-only`.)

For more information, see Values that you specify when you create or update a distribution.

Decrypting data fields at your origin

CloudFront encrypts data fields by using the Amazon Encryption SDK. The data remains encrypted throughout your application stack and can be accessed only by applications that have the credentials to decrypt it.

After encryption, the ciphertext is base64 encoded. When your applications decrypt the text at the origin, they must first decode the ciphertext, and then use the Amazon Encryption SDK to decrypt the data.

The following code example illustrates how applications can decrypt data at your origin. Note the following:

- To simplify the example, this sample loads public and private keys (in DER format) from files in the working directory. In practice, you would store the private key in a secure offline location, such as an offline hardware security module, and distribute the public key to your development team.
- CloudFront uses specific information while encrypting the data, and the same set of parameters should be used at the origin to decrypt it. Parameters CloudFront uses while initializing the MasterKey include the following:
  - PROVIDER_NAME: You specified this value when you created a field-level encryption profile. Use the same value here.
  - KEY_NAME: You created a name for your public key when you uploaded it to CloudFront, and then specified the key name in the profile. Use the same value here.
  - ALGORITHM: CloudFront uses `RSA/ECB/OAEPWithSHA-256AndMGF1Padding` as the algorithm for encrypting, so you must use the same algorithm to decrypt the data.
- If you run the following sample program with ciphertext as input, the decrypted data is output to your console. For more information, see the Java Example Code in the Amazon Encryption SDK.

Sample code

```
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyFactory;
import java.security.PrivateKey;
```

```
import java.security.PublicKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;

import org.apache.commons.codec.binary.Base64;

import com.amazonaws.encryptionsdk.AwsCrypto;
import com.amazonaws.encryptionsdk.CryptoResult;
import com.amazonaws.encryptionsdk.jce.JceMasterKey;

/**
 * Sample example of decrypting data that has been encrypted by CloudFront field-level
 encryption.
 */
public class DecryptExample {

    private static final String PRIVATE_KEY_FILENAME = "private_key.der";
    private static final String PUBLIC_KEY_FILENAME = "public_key.der";
    private static PublicKey publicKey;
    private static PrivateKey privateKey;

    // CloudFront uses the following values to encrypt data, and your origin must use
 same values to decrypt it.
    // In your own code, for PROVIDER_NAME, use the provider name that you specified
 when you created your field-level
    // encryption profile. This sample uses 'DEMO' for the value.
    private static final String PROVIDER_NAME = "DEMO";
    // In your own code, use the key name that you specified when you added your public
 key to CloudFront. This sample
    // uses 'DEMOKEY' for the key name.
    private static final String KEY_NAME = "DEMOKEY";
    // CloudFront uses this algorithm when encrypting data.
    private static final String ALGORITHM = "RSA/ECB/OAEPWithSHA-256AndMGF1Padding";

    public static void main(final String[] args) throws Exception {

        final String dataToDecrypt = args[0];

        // This sample uses files to get public and private keys.
        // In practice, you should distribute the public key and save the private key
 in secure storage.
        populateKeyPair();

        System.out.println(decrypt(debase64(dataToDecrypt)));
```

```
    }

    private static String decrypt(final byte[] bytesToDecrypt) throws Exception {
        // You can decrypt the stream only by using the private key.

        // 1. Instantiate the SDK
        final AwsCrypto crypto = new AwsCrypto();

        // 2. Instantiate a JCE master key
        final JceMasterKey masterKey = JceMasterKey.getInstance(
                publicKey,
                privateKey,
                PROVIDER_NAME,
                KEY_NAME,
                ALGORITHM);

        // 3. Decrypt the data
        final CryptoResult <byte[], ? > result = crypto.decryptData(masterKey,
 bytesToDecrypt);
        return new String(result.getResult());
    }

    // Function to decode base64 cipher text.
    private static byte[] debase64(final String value) {
        return Base64.decodeBase64(value.getBytes());
    }

    private static void populateKeyPair() throws Exception {
        final byte[] PublicKeyBytes =
 Files.readAllBytes(Paths.get(PUBLIC_KEY_FILENAME));
        final byte[] privateKeyBytes =
 Files.readAllBytes(Paths.get(PRIVATE_KEY_FILENAME));
        publicKey = KeyFactory.getInstance("RSA").generatePublic(new
 X509EncodedKeySpec(PublicKeyBytes));
        privateKey = KeyFactory.getInstance("RSA").generatePrivate(new
 PKCS8EncodedKeySpec(privateKeyBytes));
    }
}
```

Optimizing caching and availability

This section describes how to set up and manage the caching of objects to improve performance and meet your business requirements.

To learn about adding and removing the content that you want CloudFront to serve, see Adding, removing, or replacing content that CloudFront distributes.

Topics

- How caching works with CloudFront edge locations
- Increasing the proportion of requests that are served directly from the CloudFront caches (cache hit ratio)
- Using Amazon CloudFront Origin Shield
- Optimizing high availability with CloudFront origin failover
- Managing how long content stays in the cache (expiration)
- Caching content based on query string parameters
- Caching content based on cookies
- Caching content based on request headers

How caching works with CloudFront edge locations

One of the purposes of using CloudFront is to reduce the number of requests that your origin server must respond to directly. With CloudFront caching, more objects are served from CloudFront edge locations, which are closer to your users. This reduces the load on your origin server and reduces latency.

The more requests that CloudFront can serve from edge caches, the fewer viewer requests that CloudFront must forward to your origin to get the latest version or a unique version of an object. To optimize CloudFront to make as few requests to your origin as possible, consider using a CloudFront Origin Shield. For more information, see Using Amazon CloudFront Origin Shield.

The proportion of requests that are served directly from the CloudFront cache compared to all requests is called the *cache hit ratio*. You can view the percentage of viewer requests that are hits, misses, and errors in the CloudFront console. For more information, see CloudFront cache statistics reports.

A number of factors affect the cache hit ratio. You can adjust your CloudFront distribution configuration to improve the cache hit ratio by following the guidance in Increasing the proportion of requests that are served directly from the CloudFront caches (cache hit ratio).

Increasing the proportion of requests that are served directly from the CloudFront caches (cache hit ratio)

You can improve performance by increasing the proportion of your viewer requests that are served directly from the CloudFront cache instead of going to your origin servers for content. This is known as improving the cache hit ratio.

The following sections explain how to improve your cache hit ratio.

Topics

- Specifying how long CloudFront caches your objects
- Using Origin Shield
- Caching based on query string parameters
- Caching based on cookie values
- Caching based on request headers
- Remove Accept-Encoding header when compression is not needed
- Serving media content by using HTTP

Specifying how long CloudFront caches your objects

To increase your cache hit ratio, you can configure your origin to add a Cache-Control max-age directive to your objects, and specify the longest practical value for `max-age`. The shorter the cache duration, the more frequently CloudFront sends requests to your origin to determine if an object has changed and to get the latest version. You can supplement `max-age` with the `stale-while-revalidate` and `stale-if-error` directives to further improve cache hit ratio under certain conditions. For more information, see Managing how long content stays in the cache (expiration).

Using Origin Shield

CloudFront Origin Shield can help improve the cache hit ratio of your CloudFront distribution, because it provides an additional layer of caching in front of your origin. When you use Origin

Shield, all requests from all of CloudFront's caching layers to your origin come from a single location. CloudFront can retrieve each object using a single origin request from Origin Shield, and all other layers of the CloudFront cache (edge locations and regional edge caches) can retrieve the object from Origin Shield.

For more information, see Using Amazon CloudFront Origin Shield.

Caching based on query string parameters

If you configure CloudFront to cache based on query string parameters, you can improve caching if you do the following:

- Configure CloudFront to forward only the query string parameters for which your origin will return unique objects.
- Use the same case (uppercase or lowercase) for all instances of the same parameter. For example, if one request contains `parameter1=A` and another contains `parameter1=a`, CloudFront forwards separate requests to your origin when a request contains `parameter1=A` and when a request contains `parameter1=a`. CloudFront then separately caches the corresponding objects returned by your origin separately even if the objects are identical. If you use just `A` or `a`, CloudFront forwards fewer requests to your origin.
- List parameters in the same order. As with differences in case, if one request for an object contains the query string `parameter1=a&parameter2=b` and another request for the same object contains `parameter2=b&parameter1=a`, CloudFront forwards both requests to your origin and separately caches the corresponding objects even if they're identical. If you always use the same order for parameters, CloudFront forwards fewer requests to your origin.

For more information, see Caching content based on query string parameters. If you want to review the query strings that CloudFront forwards to your origin, see the values in the `cs-uri-query` column of your CloudFront log files. For more information, see Configuring and using standard logs (access logs).

Caching based on cookie values

If you configure CloudFront to cache based on cookie values, you can improve caching if you do the following:

- Configure CloudFront to forward only specified cookies instead of forwarding all cookies. For the cookies that you configure CloudFront to forward to your origin, CloudFront forwards every

combination of cookie name and value. It then separately caches the objects that your origin returns, even if they're all identical.

For example, suppose that viewers include two cookies in every request, that each cookie has three possible values, and that all combinations of cookie values are possible. CloudFront forwards up to six different requests to your origin for each object. If your origin returns different versions of an object based on only one of the cookies, then CloudFront is forwarding more requests to your origin than necessary and is needlessly caching multiple identical versions of the object.

- Create separate cache behaviors for static and dynamic content, and configure CloudFront to forward cookies to your origin only for dynamic content.

  For example, suppose you have just one cache behavior for your distribution and that you're using the distribution both for dynamic content, such as `.js` files, and for `.css` files that rarely change. CloudFront caches separate versions of your `.css` files based on cookie values, so each CloudFront edge location forwards a request to your origin for every new cookie value or combination of cookie values.

  If you create a cache behavior for which the path pattern is `*.css` and for which CloudFront doesn't cache based on cookie values, then CloudFront forwards requests for `.css` files to your origin for only the first request that an edge location receives for a given `.css` file and for the first request after a `.css` file expires.
- If possible, create separate cache behaviors for dynamic content when cookie values are unique for each user (such as a user ID), and dynamic content that varies based on a smaller number of unique values.

For more information, see Caching content based on cookies. If you want to review the cookies that CloudFront forwards to your origin, see the values in the `cs(Cookie)` column of your CloudFront log files. For more information, see Configuring and using standard logs (access logs).

Caching based on request headers

If you configure CloudFront to cache based on request headers, you can improve caching if you do the following:

- Configure CloudFront to forward and cache based on only specified headers instead of forwarding and caching based on all headers. For the headers that you specify, CloudFront

forwards every combination of header name and value. It then separately caches the objects that your origin returns even if they're all identical.

> **Note**
>
> CloudFront always forwards to your origin the headers specified in the following topics:
>
> - How CloudFront Processes and Forwards Requests to Your Amazon S3 Origin Server > HTTP request headers that CloudFront removes or updates
> - How CloudFront Processes and Forwards Requests to Your Custom Origin Server > HTTP request headers and CloudFront behavior (custom and Amazon S3 origins)

When you configure CloudFront to cache based on request headers, you don't change the headers that CloudFront forwards, only whether CloudFront caches objects based on the header values.

- Try to avoid caching based on request headers that have large numbers of unique values.

  For example, if you want to serve different sizes of an image based on the user's device, then don't configure CloudFront to cache based on the `User-Agent` header, which has an enormous number of possible values. Instead, configure CloudFront to cache based on the CloudFront device-type headers `CloudFront-Is-Desktop-Viewer`, `CloudFront-Is-Mobile-Viewer`, `CloudFront-Is-SmartTV-Viewer`, and `CloudFront-Is-Tablet-Viewer`. In addition, if you're returning the same version of the image for tablets and desktops, then forward only the `CloudFront-Is-Tablet-Viewer` header, not the `CloudFront-Is-Desktop-Viewer` header.

For more information, see Caching content based on request headers.

Remove `Accept-Encoding` header when compression is not needed

If compression is not enabled—because the origin doesn't support it, CloudFront doesn't support it, or the content is not compressible—you can increase the cache hit ratio by associating a cache behavior in your distribution to an origin that sets the Custom Origin Header as follows:

- **Header name**: `Accept-Encoding`
- **Header value**: (Keep blank)

When you use this configuration, CloudFront removes the `Accept-Encoding` header from the cache key and doesn't include the header in origin requests. This configuration applies to all content that CloudFront serves with the distribution from that origin.

Serving media content by using HTTP

For information about optimizing video on demand (VOD) and streaming video content, see Video on demand and live streaming video with CloudFront.

Using Amazon CloudFront Origin Shield

CloudFront Origin Shield is an additional layer in the CloudFront caching infrastructure that helps to minimize your origin's load, improve its availability, and reduce its operating costs. With CloudFront Origin Shield, you get the following benefits:

Better cache hit ratio

Origin Shield can help improve the cache hit ratio of your CloudFront distribution because it provides an additional layer of caching in front of your origin. When you use Origin Shield, all requests from all of CloudFront's caching layers to your origin go through Origin Shield, increasing the likelihood of a cache hit. CloudFront can retrieve each object with a single origin request from Origin Shield to your origin, and all other layers of the CloudFront cache (edge locations and regional edge caches) can retrieve the object from Origin Shield.

Reduced origin load

Origin Shield can further reduce the number of simultaneous requests that are sent to your origin for the same object. Requests for content that is not in Origin Shield's cache are consolidated with other requests for the same object, resulting in as few as one request going to your origin. Handling fewer requests at your origin can preserve your origin's availability during peak loads or unexpected traffic spikes, and can reduce costs for things like just-in-time packaging, image transformations, and data transfer out (DTO).

Better network performance

When you enable Origin Shield in the Amazon Region that has the lowest latency to your origin, you can get better network performance. For origins in an Amazon Region, CloudFront network traffic remains on the high throughput CloudFront network all the way to your origin. For origins outside of Amazon, CloudFront network traffic remains on the CloudFront network all the way to Origin Shield, which has a low latency connection to your origin.

You incur additional charges for using Origin Shield. For more information, see CloudFront Pricing.

Topics

- Use cases for Origin Shield
- Choosing the Amazon Region for Origin Shield
- Enabling Origin Shield
- Estimating Origin Shield costs
- Origin Shield high availability
- How Origin Shield interacts with other CloudFront features

Use cases for Origin Shield

CloudFront Origin Shield can be beneficial for many use cases, including the following:

- Viewers that are spread across different geographical regions
- Origins that provide just-in-time packaging for live streaming or on-the-fly image processing
- On-premises origins with capacity or bandwidth constraints
- Workloads that use multiple content delivery networks (CDNs)

Origin Shield may not be a good fit in other cases, such as dynamic content that is proxied to the origin, content with low cacheability, or content that is infrequently requested.

The following sections explain the benefits of Origin Shield for the following use cases.

Use Cases

- Viewers in different geographical regions
- Multiple CDNs

Viewers in different geographical regions

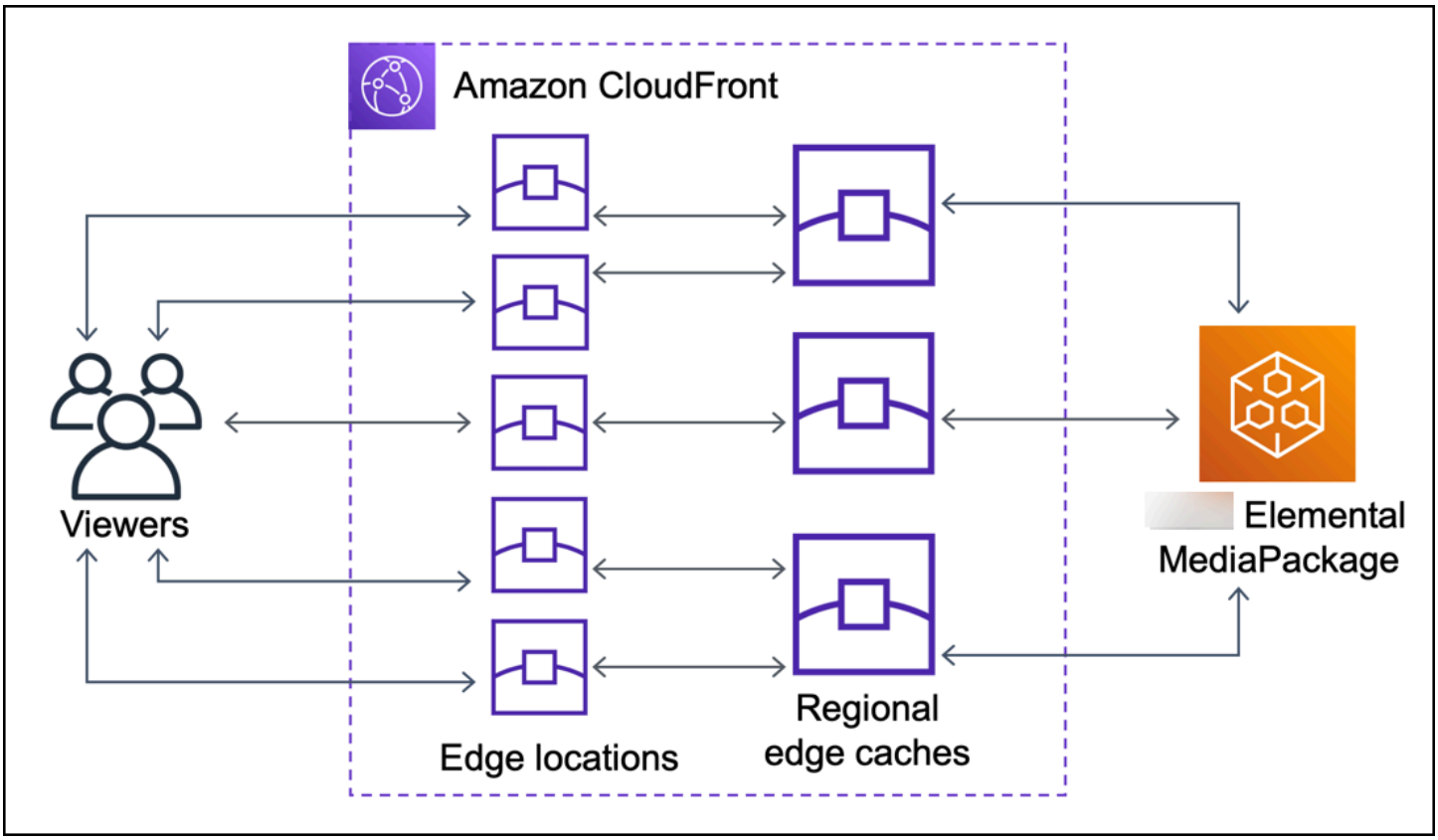
With Amazon CloudFront, you inherently get a reduced load on your origin because requests that CloudFront can serve from the cache don't go to your origin. In addition to CloudFront's global network of edge locations, regional edge caches serve as a mid-tier caching layer to provide cache hits and consolidate origin requests for viewers in nearby geographical regions. Viewer requests are

routed first to a nearby CloudFront edge location, and if the object isn't cached in that location, the request is sent on to a regional edge cache.

When viewers are in different geographical regions, requests can be routed through different regional edge caches, each of which can send a request to your origin for the same content. But with Origin Shield, you get an additional layer of caching between the regional edge caches and your origin. All requests from all regional edge caches go through Origin Shield, further reducing the load on your origin. The following diagrams illustrate this. In the following diagrams, the origin is AWS Elemental MediaPackage.

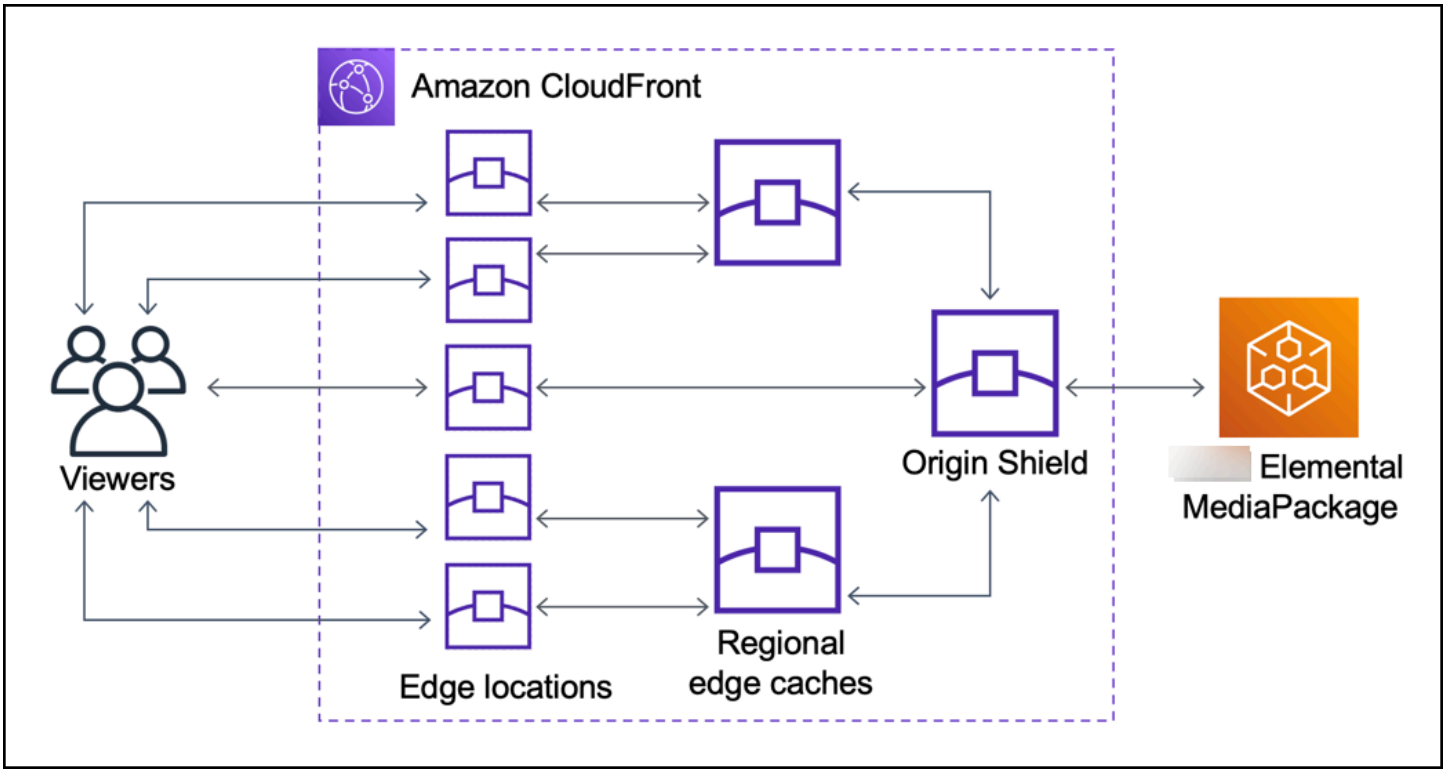
Without Origin Shield

Without Origin Shield, your origin might receive duplicate requests for the same content, as shown in the following diagram.



With Origin Shield

Using Origin Shield can help reduce the load on your origin, as shown in the following diagram.



Multiple CDNs

To serve live video events or popular on-demand content, you might use multiple content delivery networks (CDNs). Using multiple CDNs can offer certain advantages, but it also means that your origin might receive many duplicate requests for the same content, each coming from different CDNs or different locations within the same CDN. These redundant requests might adversely affect the availability of your origin or cause additional operating costs for processes like just-in-time packaging or data transfer out (DTO) to the internet.

When you combine Origin Shield with using your CloudFront distribution as the origin for other CDNs, you can get the following benefits:

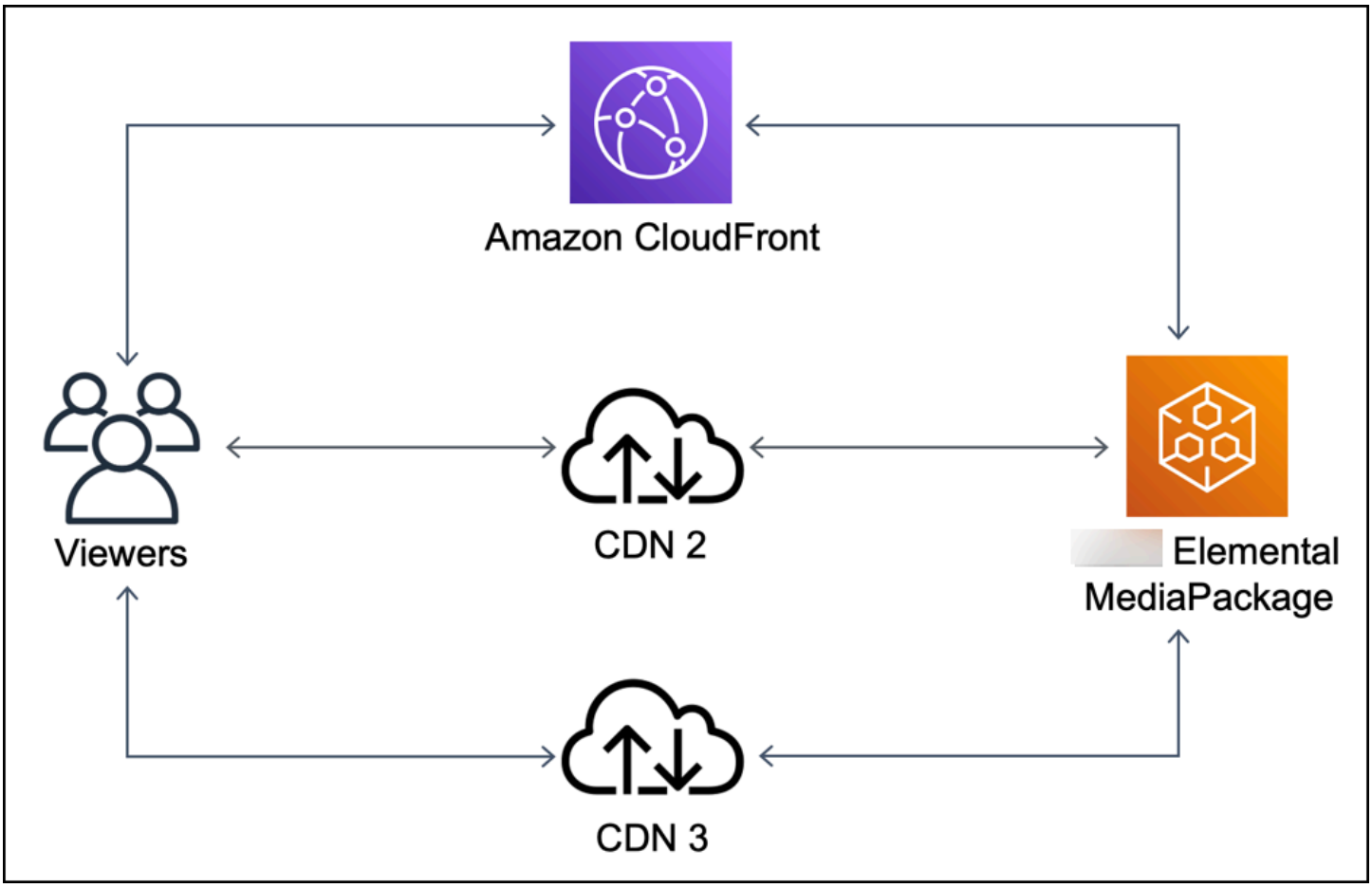
- Fewer redundant requests received at your origin, which helps to reduce the negative effects of using multiple CDNs.
- A common cache key across CDNs, and centralized management for origin-facing features.
- Improved network performance. Network traffic from other CDNs is terminated at a nearby CloudFront edge location, which might provide a hit from the local cache. If the requested object is not in the edge location cache, the request to the origin remains on the CloudFront network all the way to Origin Shield, which provides high throughput and low latency to the origin. If the requested object is in Origin Shield's cache, the request to your origin is avoided entirely.

> **Important**
>
> If you are interested in using Origin Shield in a multi-CDN architecture, and have discounted pricing, contact us or your Amazon sales representative for more information. Additional charges may apply.

The following diagrams show how this configuration can help minimize the load on your origin when you serve popular live video events with multiple CDNs. In the following diagrams, the origin is AWS Elemental MediaPackage.

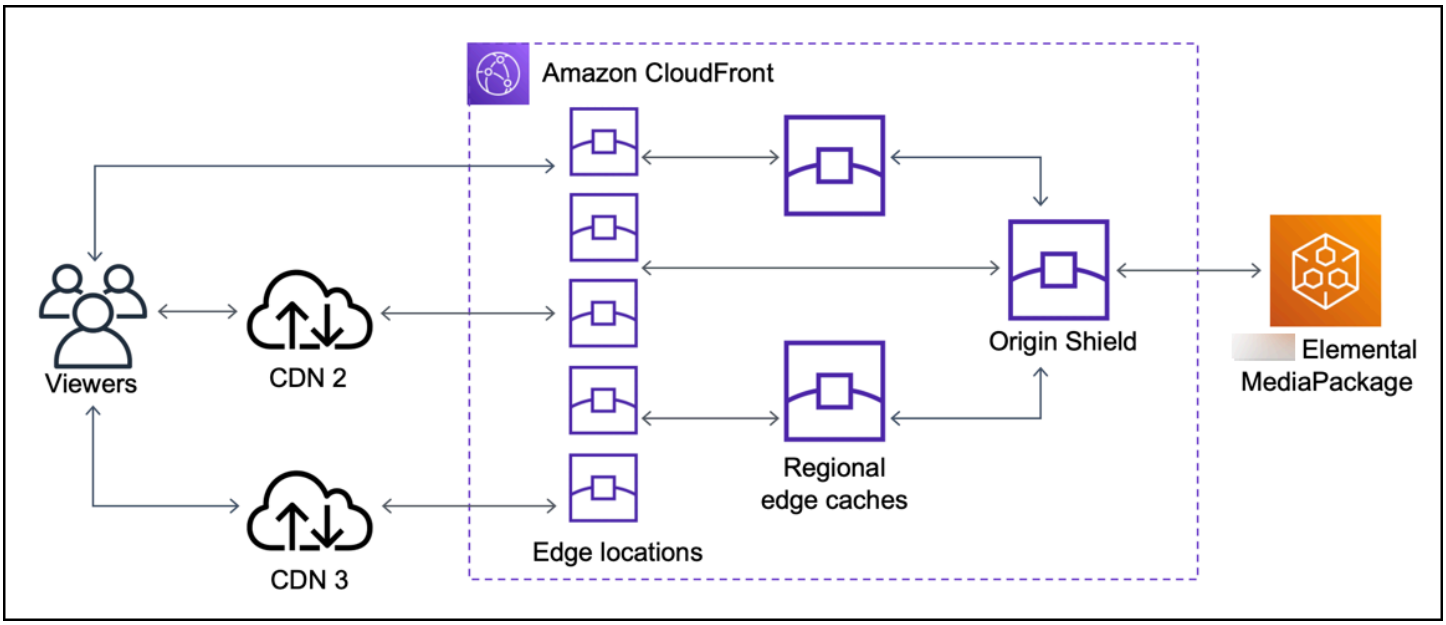
Without Origin Shield (multiple CDNs)

Without Origin Shield, your origin might receive many duplicate requests for the same content, each coming from a different CDN, as shown in the following diagram.



With Origin Shield (multiple CDNs)

Using Origin Shield, with CloudFront as the origin for your other CDNs, can help reduce the load on your origin, as shown in the following diagram.



Choosing the Amazon Region for Origin Shield

Amazon CloudFront offers Origin Shield in Amazon Regions where CloudFront has a regional edge cache. When you enable Origin Shield, you choose the Amazon Region for Origin Shield. You should choose the Amazon Region that has the lowest latency to your origin. You can use Origin Shield with origins that are in an Amazon Region, and with origins that are not in Amazon.

For origins in an Amazon Region

If your origin is in an Amazon Region, first determine whether your origin is in a Region in which CloudFront offers Origin Shield. CloudFront offers Origin Shield in the following Amazon Regions.

- US East (Ohio) – `us-east-2`
- US East (N. Virginia) – `us-east-1`
- US West (Oregon) – `us-west-2`
- Asia Pacific (Mumbai) – `ap-south-1`
- Asia Pacific (Seoul) – `ap-northeast-2`
- Asia Pacific (Singapore) – `ap-southeast-1`
- Asia Pacific (Sydney) – `ap-southeast-2`
- Asia Pacific (Tokyo) – `ap-northeast-1`

- Europe (Frankfurt) – `eu-central-1`
- Europe (Ireland) – `eu-west-1`
- Europe (London) – `eu-west-2`
- South America (São Paulo) – `sa-east-1`

If your origin is in an Amazon Region in which CloudFront offers Origin Shield

If your origin is in an Amazon Region in which CloudFront offers Origin Shield (see the preceding list), enable Origin Shield in the same Region as your origin.

If your origin is not in an Amazon Region in which CloudFront offers Origin Shield

If your origin is not in an Amazon Region in which CloudFront offers Origin Shield, see the following table to determine which Region to enable Origin Shield in.

If your origin is in ...	Enable Origin Shield in ...
US West (N. California) – `us-west-1`	US West (Oregon) – `us-west-2`
Africa (Cape Town) – `af-south-1`	Europe (Ireland) – `eu-west-1`
Asia Pacific (Hong Kong) – `ap-east-1`	Asia Pacific (Singapore) – `ap-southeast-1`
Canada (Central) – `ca-central-1`	US East (N. Virginia) – `us-east-1`
Europe (Milan) – `eu-south-1`	Europe (Frankfurt) – `eu-central-1`
Europe (Paris) – `eu-west-3`	Europe (London) – `eu-west-2`
Europe (Stockholm) – `eu-north-1`	Europe (London) – `eu-west-2`
Middle East (Bahrain) – `me-south-1`	Asia Pacific (Mumbai) – `ap-south-1`

For origins outside of Amazon

You can use Origin Shield with an origin that is on-premises or is not in an Amazon Region. In this case, enable Origin Shield in the Amazon Region that has the lowest latency to your origin. If you're not sure which Amazon Region has the lowest latency to your origin, you can use the following suggestions to help you make a determination.

- You can consult the preceding table for an approximation of which Amazon Region might have the lowest latency to your origin, based on your origin's geographic location.
- You can launch Amazon EC2 instances in a few different Amazon Regions that are geographically close to your origin, and run some tests using `ping` to measure the typical network latencies between those Regions and your origin.

Enabling Origin Shield

You can enable Origin Shield to improve your cache hit ratio, reduce the load on your origin, and help improve performance. To enable Origin Shield, change the origin settings in a CloudFront distribution. Origin Shield is a property of the origin. For each origin in your CloudFront distributions, you can separately enable Origin Shield in whichever Amazon Region provides the best performance for that origin.

You can enable Origin Shield in the CloudFront console, with Amazon CloudFormation, or with the CloudFront API.

Console

To enable Origin Shield for an existing origin (console)

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose the distribution that has the origin that you want to update.
3. Choose the **Origins and Origin Groups** tab.
4. Choose the origin to update, then choose **Edit**.
5. For **Enable Origin Shield**, choose **Yes**.
6. For **Origin Shield Region**, choose the Amazon Region where you want to enable Origin Shield. For help choosing a Region, see Choosing the Amazon Region for Origin Shield.
7. At the bottom of the page, choose **Yes, Edit**.

When your distribution status is **Deployed**, Origin Shield is ready. This takes a few minutes.

To enable Origin Shield for a new origin (console)

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.

2. To create the new origin in an existing distribution, do the following:

   1. Choose the distribution where you want to create the origin.
   2. Choose **Create Origin**, and then proceed to step 3.

   To create the new origin in a new distribution, do the following:

   1. Choose **Create Distribution**.
   2. In the **Web** section, choose **Get Started**. In the **Origin Settings** section, complete the following steps, starting with step 3.
3. For **Enable Origin Shield**, choose **Yes**.
4. For **Origin Shield Region**, choose the Amazon Region where you want to enable Origin Shield. For help choosing a Region, see Choosing the Amazon Region for Origin Shield.

   If you are creating a new distribution, continue configuring your distribution, using the other settings on the page. For more information, see Values that you specify when you create or update a distribution.
5. Make sure to save your changes by choosing **Create** (for a new origin in an existing distribution) or **Create Distribution** (for a new origin in a new distribution).

When your distribution status is **Deployed**, Origin Shield is ready. This takes a few minutes.

Amazon CloudFormation

To enable Origin Shield with Amazon CloudFormation, use the `OriginShield` property in the `Origin` property type in an `AWS::CloudFront::Distribution` resource. You can add the `OriginShield` property to an existing `Origin`, or include it when you create a new `Origin`.

The following example shows the syntax, in YAML format, for enabling `OriginShield` in the US West (Oregon) Region (`us-west-2`). For help choosing a Region, see the section called "Choosing the Amazon Region for Origin Shield". This example shows only the `Origin` property type, not the entire `AWS::CloudFront::Distribution` resource.

```
Origins:
- DomainName: 3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com
  Id: Example-EMP-3ae97e9482b0d011
  OriginShield:
    Enabled: true
```

```
    OriginShieldRegion: us-west-2
  CustomOriginConfig:
    OriginProtocolPolicy: match-viewer
    OriginSSLProtocols: TLSv1
```

For more information, see AWS::CloudFront::Distribution Origin in the resource and property reference section of the *Amazon CloudFormation User Guide*.

API

To enable Origin Shield with the CloudFront API using the Amazon SDKs or Amazon Command Line Interface (Amazon CLI), use the `OriginShield` type. You specify `OriginShield` in an `Origin`, in a `DistributionConfig`. For information about the `OriginShield` type, see the following information in the *Amazon CloudFront API Reference*.

- OriginShield (type)
- Origin (type)
- DistributionConfig (type)
- UpdateDistribution (operation)
- CreateDistribution (operation)

The specific syntax for using these types and operations varies based on the SDK, CLI, or API client. For more information, see the reference documentation for your SDK, CLI, or client.

Estimating Origin Shield costs

You accrue charges for Origin Shield based on the number of requests that go to Origin Shield as an incremental layer.

For dynamic (non-cacheable) requests that are proxied to the origin, Origin Shield is always an incremental layer. Dynamic requests use the HTTP methods `PUT`, `POST`, `PATCH`, and `DELETE`.

`GET` and `HEAD` requests that have a time to live (TTL) setting of less than 3600 seconds are considered dynamic requests. In addition, `GET` and `HEAD` requests that have disabled caching are also considered dynamic requests.

To estimate your charges for Origin Shield for dynamic requests, use the following formula:

Total number of dynamic requests **x** Origin Shield charge per 10,000 requests **/** 10,000

For non-dynamic requests with the HTTP methods `GET`, `HEAD`, and `OPTIONS`, Origin Shield is sometimes an incremental layer. When you enable Origin Shield, you choose the Amazon Web Services Region for Origin Shield. For requests that naturally go to the regional edge cache in the same Region as Origin Shield, Origin Shield is not an incremental layer. You don't accrue Origin Shield charges for these requests. For requests that go to a regional edge cache in a different Region from Origin Shield, and then go to Origin Shield, Origin Shield is an incremental layer. You do accrue Origin Shield charges for these requests.

To estimate your charges for Origin Shield for cacheable requests, use the following formula:

Total number of cacheable requests **x** (1 – cache hit rate) **x** percentage of requests that go to Origin Shield from a regional edge cache in a different region **x** Origin Shield charge per 10,000 requests **/** 10,000

For more information about the charge per 10,000 requests for Origin Shield, see CloudFront Pricing.

Origin Shield high availability

Origin Shield leverages the CloudFront regional edge caches feature. Each of these edge caches is built in an Amazon Region using at least three Availability Zones with fleets of auto-scaling Amazon EC2 instances. Connections from CloudFront locations to Origin Shield also use active error tracking for each request to automatically route the request to a secondary Origin Shield location if the primary Origin Shield location is unavailable.

How Origin Shield interacts with other CloudFront features

The following sections explain how Origin Shield interacts with other CloudFront features.

Origin Shield and CloudFront logging

To see when Origin Shield handled a request, you must enable one of the following:

- CloudFront standard logs (access logs). Standard logs are provided free of charge.
- CloudFront real-time logs. You incur additional charges for using real-time logs. See Amazon CloudFront Pricing.

Cache hits from Origin Shield appear as `OriginShieldHit` in the `x-edge-detailed-result-type` field in CloudFront logs. Origin Shield leverages Amazon CloudFront's regional edge caches.

If a request is routed from a CloudFront edge location to the regional edge cache that is acting as Origin Shield, it is reported as a `Hit` in the logs, not as an `OriginShieldHit`.

Origin Shield and origin groups

Origin Shield is compatible with CloudFront origin groups. Because Origin Shield is a property of the origin, requests always travel through Origin Shield for each origin even when the origin is part of an origin group. For a given request, CloudFront routes the request to the primary origin in the origin group through the primary origin's Origin Shield. If that request fails (according to the origin group failover criteria), CloudFront routes the request to the secondary origin through the secondary origin's Origin Shield.

Origin Shield and Lambda@Edge

Origin Shield does not impact the functionality of Lambda@Edge functions, but it can affect the Amazon Region where those functions run.

When you use Origin Shield with Lambda@Edge, origin-facing triggers (origin request and origin response) run in the Amazon Region where Origin Shield is enabled. If the primary Origin Shield location is unavailable and CloudFront routes requests to a secondary Origin Shield location, Lambda@Edge origin-facing triggers will also shift to use the secondary Origin Shield location.

Viewer-facing triggers are not affected.

Optimizing high availability with CloudFront origin failover

You can set up CloudFront with origin failover for scenarios that require high availability. To get started, you create an *origin group* with two origins: a primary and a secondary. If the primary origin is unavailable, or returns specific HTTP response status codes that indicate a failure, CloudFront automatically switches to the secondary origin.

To set up origin failover, you must have a distribution with at least two origins. Next, you create an origin group for your distribution that includes two origins, setting one as the primary. Finally, you create or update a cache behavior to use the origin group.

To see the steps for setting up origin groups and configuring specific origin failover options, see Creating an origin group.

After you configure origin failover for a cache behavior, CloudFront does the following for viewer requests:

- When there's a cache hit, CloudFront returns the requested object.
- When there's a cache miss, CloudFront routes the request to the primary origin in the origin group.
- When the primary origin returns a status code that is not configured for failover, such as an HTTP 2xx or 3xx status code, CloudFront serves the requested object to the viewer.
- When any of the following occur:
  - The primary origin returns an HTTP status code that you've configured for failover
  - CloudFront fails to connect to the primary origin
  - The response from the primary origin takes too long (times out)

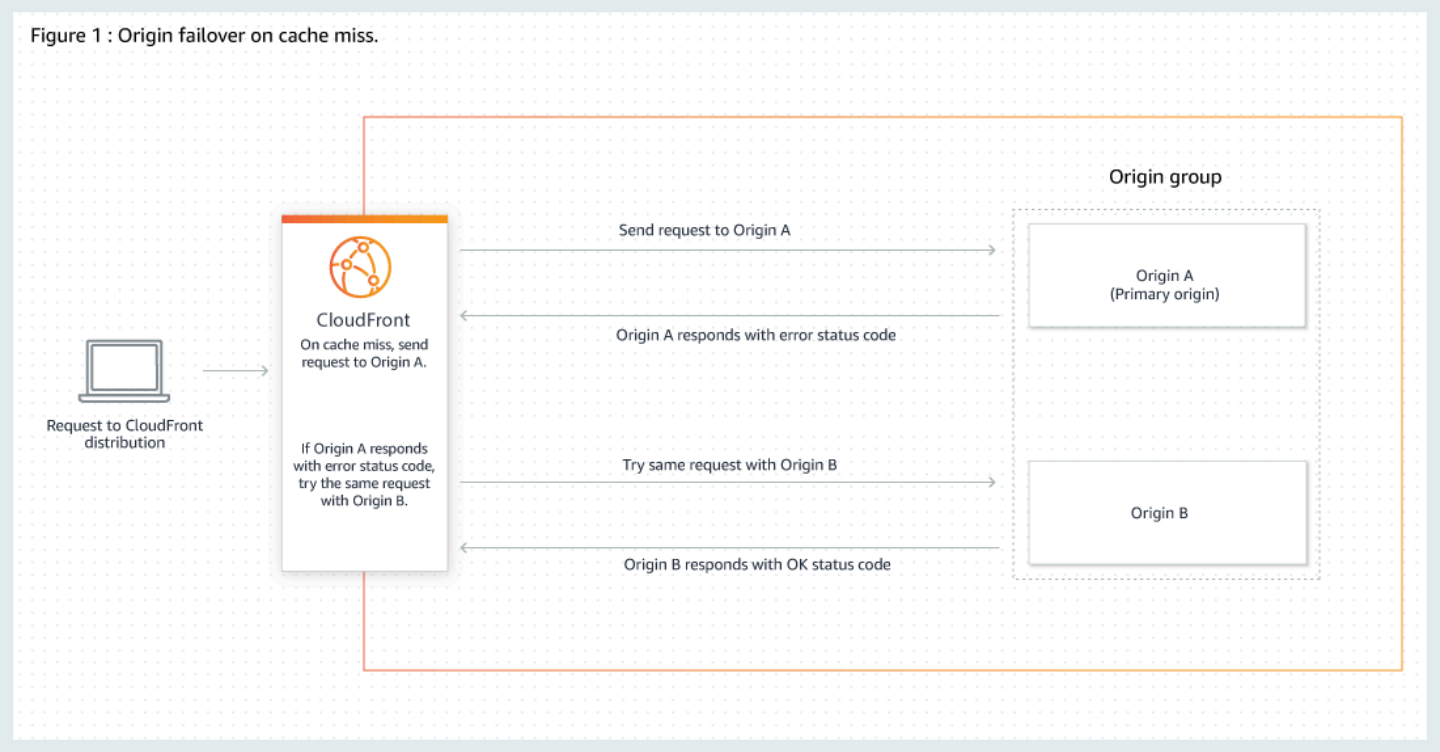
  Then CloudFront routes the request to the secondary origin in the origin group.

> **Note**
>
> For some use cases, like streaming video content, you might want CloudFront to fail over to the secondary origin quickly. To adjust how quickly CloudFront fails over to the secondary origin, see Controlling origin timeouts and attempts.

CloudFront routes all incoming requests to the primary origin, even when a previous request failed over to the secondary origin. CloudFront only sends requests to the secondary origin after a request to the primary origin fails.

CloudFront fails over to the secondary origin only when the HTTP method of the viewer request is `GET`, `HEAD`, or `OPTIONS`. CloudFront does not fail over when the viewer sends a different HTTP method (for example `POST`, `PUT`, and so on).

The following diagram illustrates how origin failover works.



Topics

- Creating an origin group
- Controlling origin timeouts and attempts
- Use origin failover with Lambda@Edge functions
- Use custom error pages with origin failover

Creating an origin group

To create an origin group

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose the distribution that you want to create the origin group for.
3. Choose the **Origins** tab.
4. Make sure the distribution has more than one origin. If it doesn't, add a second origin.
5. On the **Origins** tab, in the **Origin groups** pane, choose **Create origin group**.
6. Choose the origins for the origin group. After you add origins, use the arrows to set the priority —that is, which origin is primary and which is secondary.

7. Enter a name for the origin group.
8. Choose the HTTP status codes to use as failover criteria. You can choose any combination of the following status codes: 400, 403, 404, 416, 500, 502, 503, or 504. When CloudFront receives a response with one of the status codes that you specify, it fails over to the secondary origin.

   > **Note**
   >
   > CloudFront fails over to the secondary origin only when the HTTP method of the viewer request is `GET`, `HEAD`, or `OPTIONS`. CloudFront does not fail over when the viewer sends a different HTTP method (for example `POST`, `PUT`, and so on).

9. Choose **Create origin group**.

Make sure to assign your origin group as the origin for your distribution's cache behavior. For more information, see Name.

Controlling origin timeouts and attempts

By default, CloudFront tries to connect to the primary origin in an origin group for as long as 30 seconds (3 connection attempts of 10 seconds each) before failing over to the secondary origin. For some use cases, like streaming video content, you might want CloudFront to fail over to the secondary origin more quickly. You can adjust the following settings to affect how quickly CloudFront fails over to the secondary origin. If the origin is a secondary origin, or an origin that is not part of an origin group, these settings affect how quickly CloudFront returns an HTTP 504 response to the viewer.

To fail over more quickly, specify a shorter connection timeout, fewer connection attempts, or both. For custom origins (including Amazon S3 bucket origins that *are* configured with static website hosting), you can also adjust the origin response timeout.

Origin connection timeout

The origin connection timeout setting affects how long CloudFront waits when trying to establish a connection to the origin. By default, CloudFront waits 10 seconds to establish a connection, but you can specify 1–10 seconds (inclusive). For more information, see Connection timeout.

Origin connection attempts

The origin connection attempts setting affects the number of times that CloudFront attempts to connect to the origin. By default, CloudFront tries 3 times to connect, but you can specify 1–3 (inclusive). For more information, see Connection attempts.

For a custom origin (including an Amazon S3 bucket that's configured with static website hosting), this setting also affects the number of times that CloudFront attempts to get a response from the origin in the case of an origin response timeout.

Origin response timeout

> **Note**
>
> This applies only to custom origins.

The origin response timeout setting affects how long CloudFront waits to receive a response (or to receive the complete response) from the origin. By default, CloudFront waits for 30 seconds, but you can specify 1–60 seconds (inclusive). For more information, see Response timeout (custom origins only).

How to change these settings

To change these settings in the CloudFront console

- For a new origin or a new distribution, you specify these values when you create the resource.
- For an existing origin in an existing distribution, you specify these values when you edit the origin.

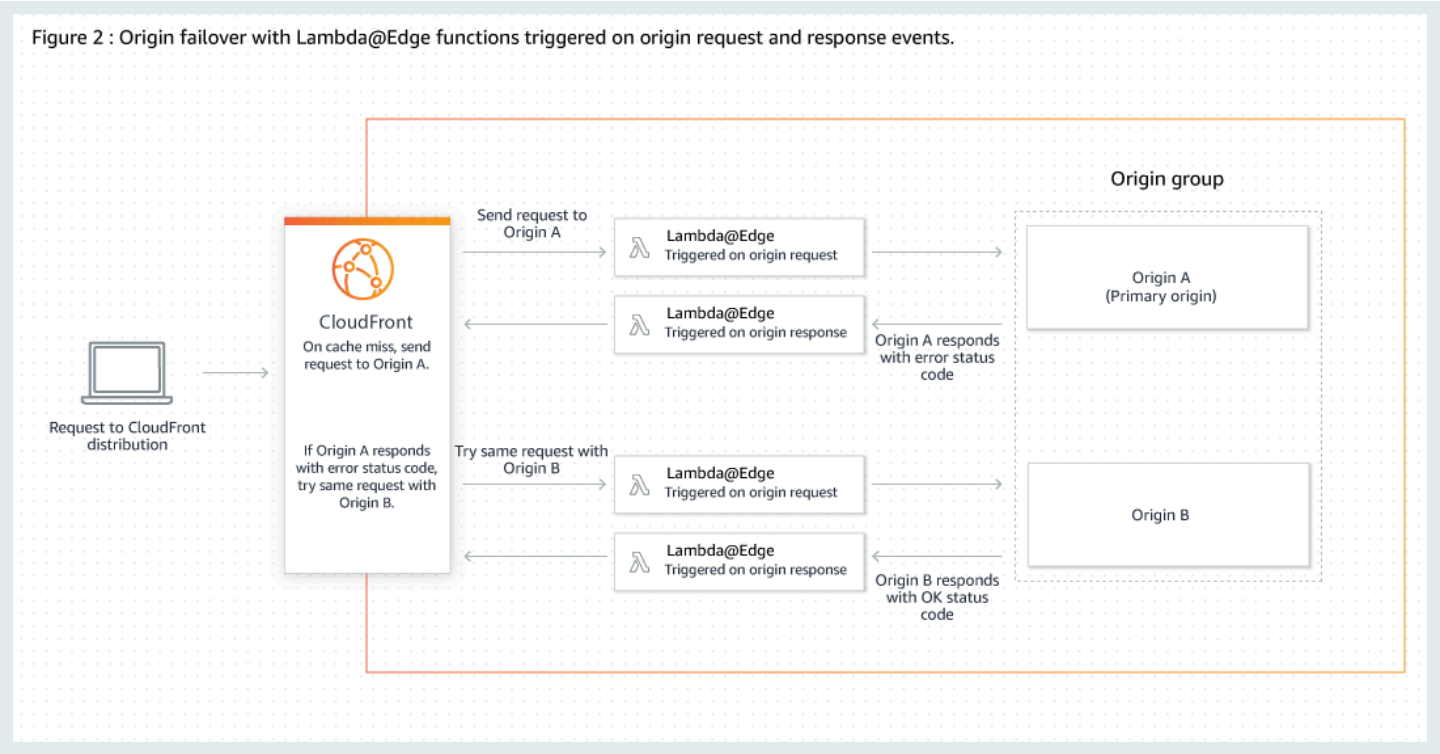
For more information, see Values that you specify when you create or update a distribution.

Use origin failover with Lambda@Edge functions

You can use Lambda@Edge functions with CloudFront distributions that you've set up with origin groups. To use a Lambda function, specify it in an origin request or origin response trigger for an origin group when you create the cache behavior. When you use a Lambda@Edge function with an origin group, the function can be triggered twice for a single viewer request. For example, consider this scenario:

1. You create a Lambda@Edge function with an origin request trigger.
2. The Lambda function is triggered once when CloudFront sends a request to the primary origin (on a cache miss).
3. The primary origin responds with an HTTP status code that's configured for failover.
4. The Lambda function is triggered again when CloudFront sends the same request to the secondary origin.

The following diagram illustrates how origin failover works when you include a Lambda@Edge function in an origin request or response trigger.



For more information about using Lambda@Edge triggers, see the section called "Adding triggers".

For more information about managing DNS failover, see Configuring DNS failover in the *Amazon Route 53 Developer Guide*.

Use custom error pages with origin failover

You can use custom error pages with origin groups similarly to how you use them with origins that are not set up for origin failover.

When you use origin failover, you can configure CloudFront to return a custom error page for the primary or secondary origin (or both):

- **Return a custom error page for the primary origin** – If the primary origin returns an HTTP status code that's not configured for failover, CloudFront returns the custom error page to viewers.
- **Return a custom error page for the secondary origin** – If CloudFront receives a failure status code from the secondary origin, CloudFront returns the custom error page.

For more information about using custom error pages with CloudFront, see Generating custom error responses.

Managing how long content stays in the cache (expiration)

You can control how long your files stay in a CloudFront cache before CloudFront forwards another request to your origin. Reducing the duration allows you to serve dynamic content. Increasing the duration means that your users get better performance because your files are more likely to be served directly from the edge cache. A longer duration also reduces the load on your origin.

Typically, CloudFront serves a file from an edge location until the cache duration that you specified passes—that is, until the file expires. After it expires, the next time the edge location gets a request for the file, CloudFront forwards the request to the origin to verify that the cache contains the latest version of the file. The response from the origin depends on whether the file has changed:

- If the CloudFront cache already has the latest version, the origin returns a status code `304 Not Modified`.
- If the CloudFront cache does not have the latest version, the origin returns a status code `200 OK` and the latest version of the file.

If a file in an edge location isn't frequently requested, CloudFront might evict the file—remove the file before its expiration date—to make room for files that have been requested more recently.

By default, each file automatically expires after 24 hours, but you can change the default behavior in two ways:

- To change the cache duration for all files that match the same path pattern, you can change the CloudFront settings for **Minimum TTL**, **Maximum TTL**, and **Default TTL** for a cache behavior. For

information about the individual settings, see Minimum TTL, Maximum TTL, and Default TTL in the section called "Values that you specify".

- To change the cache duration for an individual file, you can configure your origin to add a `Cache-Control` header with the `max-age` or `s-maxage` directive, or an `Expires` header to the file. For more information, see Using headers to control cache duration for individual objects.

For more information about how **Minimum TTL**, **Default TTL**, and **Maximum TTL** interact with the `max-age` and `s-maxage` directives and the `Expires` header field, see the section called "Specifying the amount of time that CloudFront caches objects".

You can also control how long errors (for example, `404 Not Found`) stay in a CloudFront cache before CloudFront tries again to get the requested object by forwarding another request to your origin. For more information, see the section called "How CloudFront processes and caches HTTP 4xx and 5xx status codes from your origin".

Topics

- Using headers to control cache duration for individual objects
- Serving stale (expired) content
- Specifying the amount of time that CloudFront caches objects
- Adding headers to your objects using the Amazon S3 console

Using headers to control cache duration for individual objects

You can use the `Cache-Control` and `Expires` headers to control how long objects stay in the cache. Settings for **Minimum TTL**, **Default TTL**, and **Maximum TTL** also affect cache duration, but here's an overview of how headers can affect cache duration:

- The `Cache-Control max-age` directive lets you specify how long (in seconds) that you want an object to remain in the cache before CloudFront gets the object again from the origin server. The minimum expiration time CloudFront supports is 0 seconds. The maximum value is 100 years. Specify the value in the following format:

  ```
  Cache-Control: max-age=seconds
  ```

  For example, the following directive tells CloudFront to keep the associated object in the cache for 3600 seconds (one hour):

```
Cache-Control: max-age=3600
```

If you want objects to stay in CloudFront edge caches for a different duration than they stay in browser caches, you can use the `Cache-Control max-age` and `Cache-Control s-maxage` directives together. For more information, see Specifying the amount of time that CloudFront caches objects.

- The `Expires` header field lets you specify an expiration date and time using the format specified in RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1 Section 3.3.1, Full Date, for example:

```
Sat, 27 Jun 2015 23:59:59 GMT
```

We recommend that you use the `Cache-Control max-age` directive instead of the `Expires` header field to control object caching. If you specify values both for `Cache-Control max-age` and for `Expires`, CloudFront uses only the value of `Cache-Control max-age`.

For more information, see Specifying the amount of time that CloudFront caches objects.

You cannot use the HTTP `Cache-Control` or `Pragma` header fields in a `GET` request from a viewer to force CloudFront to go back to the origin server for the object. CloudFront ignores those header fields in viewer requests.

For more information about the `Cache-Control` and `Expires` header fields, see the following sections in *RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1*:

- Section 14.9 Cache Control
- Section 14.21 Expires

Serving stale (expired) content

CloudFront supports the `Stale-While-Revalidate` and `Stale-If-Error` cache control directives.

- The `stale-while-revalidate` directive allows CloudFront to serve stale content from cache while it asynchronously fetches a fresh version from the origin. This improves latency as users receive responses immediately from CloudFront's edge locations without having to wait for the background fetch, and fresh content is loaded in the background for future requests.

In the following example, CloudFront caches the response for one hour (`max-age=3600`). If a request is made after this period, CloudFront serves the stale content while concurrently sending a request to the origin to revalidate and refresh the cached content. The stale content is served for up to 10 minutes (`stale-while-revalidate=600`) while the content is being revalidated.

```
Cache-Control: max-age=3600, stale-while-revalidate=600
```

- The `stale-if-error` directive allows CloudFront to serve stale content from cache if the origin is unreachable or returns an error code that is between 500 and 600. This ensures that viewers can access content even during an origin outage.

  In the following example, CloudFront caches the response for one hour (`max-age=3600`). If the origin is down or returns an error after this period, CloudFront continues to serve the stale content for up to 24 hours (`stale-if-error=86400`).

  ```
  Cache-Control: max-age=3600, stale-if-error=86400
  ```

  Note

  When both `stale-if-error` and custom error responses are configured, CloudFront first attempts to serve the stale content if an error is encountered within the specified `stale-if-error` duration. If stale content is unavailable, or the content is beyond the `stale-if-error` duration, CloudFront serves the custom error responses configured for the corresponding error status code.

Using both together

`stale-while-revalidate` and `stale-if-error` are independent cache control directives that can be used together to reduce latency and to add a buffer for your origin to respond or recover.

In the following example, CloudFront caches the response for one hour (`max-age=3600`). If a request is made after this period, CloudFront serves the stale content for up to 10 minutes (`stale-while-revalidate=600`) while the content is being revalidated. If the origin server returns an error while CloudFront attempts to revalidate the content, CloudFront continues to serve the stale content for up to 24 hours (`stale-if-error=86400`).

```
Cache-Control: max-age=3600, stale-while-revalidate=600, stale-if-error=86400
```

Tip

Caching is a balance between performance and freshness. Using directives like `stale-while-revalidate` and `stale-if-error` can enhance performance and user experience, but make sure the configurations align with how fresh you want your content to be. Stale content directives are best suited for use cases where content needs to be refreshed but having the latest version is non-essential. Additionally, if your content doesn't change or rarely changes, `stale-while-revalidate` could add unnecessary network requests. Instead, consider setting a long cache duration.

Specifying the amount of time that CloudFront caches objects

To control the amount of time that CloudFront keeps an object in the cache before sending another request to the origin, you can:

- Set the minimum, maximum, and default TTL values in a CloudFront distribution's cache behavior. You can set these values in a cache policy attached to the cache behavior (recommended), or in the legacy cache settings.
- Include the `Cache-Control` or `Expires` header in responses from the origin. These headers also help determine how long a browser keeps an object in the browser cache before sending another request to CloudFront.

The following table explains how the `Cache-Control` and `Expires` headers sent from the origin work together with the TTL settings in a cache behavior to affect caching.

Origin headers	Minimum TTL = 0	Minimum TTL > 0
The origin adds a `Cache-Control: max-age` directive to the object	**CloudFront caching** CloudFront caches the object for the lesser of the value of the `Cache-Control:`	**CloudFront caching** CloudFront caching depends on the values of the CloudFront minimum TTL

Origin headers	Minimum TTL = 0	Minimum TTL > 0
	`max-age` directive or the value of the CloudFront maximum TTL. **Browser caching** Browsers cache the object for the value of the `Cache-Control: max-age` directive.	and maximum TTL and the `Cache-Control max-age` directive: • If minimum TTL < `max-age` < maximum TTL, then CloudFront caches the object for the value of the `Cache-Control: max-age` directive. • If `max-age` < minimum TTL, then CloudFront caches the object for the value of the CloudFront minimum TTL. • If `max-age` > maximum TTL, then CloudFront caches the object for the value of the CloudFront maximum TTL. **Browser caching** Browsers cache the object for the value of the `Cache-Control: max-age` directive.

Origin headers	Minimum TTL = 0	Minimum TTL > 0
The origin does not add a `Cache-Control: max-age` directive to the object	**CloudFront caching** CloudFront caches the object for the value of the CloudFront default TTL. **Browser caching** Depends on the browser.	**CloudFront caching** CloudFront caches the object for the greater of the value of the CloudFront minimum TTL or default TTL. **Browser caching** Depends on the browser.

Origin headers	Minimum TTL = 0	Minimum TTL > 0
The origin adds `Cache-Control: max-age` and `Cache-Control: s-maxage` directives to the object	**CloudFront caching** CloudFront caches the object for the lesser of the value of the `Cache-Control: s-maxage` directive or the value of the CloudFront maximum TTL. **Browser caching** Browsers cache the object for the value of the `Cache-Control max-age` directive.	**CloudFront caching** CloudFront caching depends on the values of the CloudFront minimum TTL and maximum TTL and the `Cache-Control: s-maxage` directive: • If minimum TTL < `s-maxage` < maximum TTL, then CloudFront caches the object for the value of the `Cache-Control: s-maxage` directive. • If `s-maxage` < minimum TTL, then CloudFront caches the object for the value of the CloudFront minimum TTL. • If `s-maxage` > maximum TTL, then CloudFront caches the object for the value of the CloudFront maximum TTL. **Browser caching** Browsers cache the object for the value of the `Cache-Control: max-age` directive.

Origin headers	Minimum TTL = 0	Minimum TTL > 0
The origin adds an `Expires` header to the object	**CloudFront caching** CloudFront caches the object until the date in the `Expires` header or for the value of the CloudFront maximum TTL, whichever is sooner. **Browser caching** Browsers cache the object until the date in the `Expires` header.	**CloudFront caching** CloudFront caching depends on the values of the CloudFront minimum TTL and maximum TTL and the `Expires` header: • If minimum TTL < `Expires` < maximum TTL, then CloudFront caches the obj ect until the date and time in the `Expires` header. • If `Expires` < minimum TTL, then CloudFront caches the object for the value of the CloudFront minimum TTL. • If `Expires` > maximum TTL, then CloudFront caches the object for the value of the CloudFront maximum TTL. **Browser caching** Browsers cache the object until the date and time in the `Expires` header.

Origin headers	Minimum TTL = 0	Minimum TTL > 0
Origin adds `Cache-Control: no-cache , no-store`, and/or `private` directives to the object	CloudFront and browsers respect the headers.	**CloudFront caching** CloudFront caches the object for the value of the CloudFront minimum TTL. See the warning below this table. **Browser caching** Browsers respect the headers.

Warning

If your minimum TTL is greater than 0, CloudFront uses the cache policy's minimum TTL, even if the `Cache-Control: no-cache`, `no-store`, and/or `private` directives are present in the origin headers.
If the origin is reachable, CloudFront gets the object from the origin and returns it to the viewer.
If the origin is unreachable and the minimum *or* maximum TTL value is greater than 0, CloudFront will serve the object that it got from the origin previously.
To avoid this behavior, include the `Cache-Control: stale-if-error=0` directive with the object returned from the origin. This causes CloudFront to return an error in response to future requests if the origin is unreachable, rather than returning the object that it got from the origin previously.

For information about how to change settings for distributions using the CloudFront console, see Updating a distribution. For information about how to change settings for distributions using the CloudFront API, see UpdateDistribution.

Adding headers to your objects using the Amazon S3 console

To add a `Cache-Control` or `Expires` header field to Amazon S3 objects using the Amazon S3 console

1. Sign in to the Amazon Web Services Management Console and open the Amazon S3 console at https://console.amazonaws.cn/s3/.
2. In the list of buckets, choose the name of the bucket that contains the files that you are adding headers to.
3. Select the check box next to the name of the file or folder that you are adding headers to. When you add headers to a folder, it affects all the files inside that folder.
4. Choose **Actions**, then choose **Edit metadata**.
5. In the **Add metadata** panel, do the following:
   a. Choose **Add metadata**.
   b. For **Type**, choose **System defined**.
   c. For **Key**, choose the name of the header that you are adding (**Cache-Control** or **Expires**).
   d. For **Value**, enter a header value. For example, for a `Cache-Control` header, you could enter `max-age=86400`. For `Expires`, you could enter an expiration date and time such as `Wed, 30 Jun 2021 09:28:00 GMT`.
6. At the bottom of the page, choose **Edit metadata**.

Caching content based on query string parameters

Some web applications use query strings to send information to the origin. A query string is the part of a web request that appears after a ? character; the string can contain one or more parameters, separated by & characters. In the following example, the query string includes two parameters, *`color=red`* and *`size=large`*:

`https://d111111abcdef8.cloudfront.net/images/image.jpg?`*`color=red&size=large`*

For distributions, you can choose if you want CloudFront to forward query strings to your origin and whether to cache your content based on all parameters or on selected parameters. Why might this be useful? Consider the following example.

Suppose that your website is available in five languages. The directory structure and file names for all five versions of the website are identical. As a user views your website, requests that are forwarded to CloudFront include a language query string parameter based on the language that the user chose. You can configure CloudFront to forward query strings to the origin and to cache based on the language parameter. If you configure your web server to return the version of a given page that corresponds with the selected language, CloudFront caches each language version separately, based on the value of the language query string parameter.

In this example, if the main page for your website is `main.html`, the following five requests cause CloudFront to cache `main.html` five times, once for each value of the language query string parameter:

- `https://d111111abcdef8.cloudfront.net/main.html?language=de`
- `https://d111111abcdef8.cloudfront.net/main.html?language=en`
- `https://d111111abcdef8.cloudfront.net/main.html?language=es`
- `https://d111111abcdef8.cloudfront.net/main.html?language=fr`
- `https://d111111abcdef8.cloudfront.net/main.html?language=jp`

Note the following:

- Some HTTP servers don't process query string parameters and, therefore, don't return different versions of an object based on parameter values. For these origins, if you configure CloudFront to forward query string parameters to the origin, CloudFront still caches based on the parameter values even though the origin returns identical versions of the object to CloudFront for every parameter value.
- For query string parameters to work as described in the example above with the languages, you must use the & character as the delimiter between query string parameters. If you use a different delimiter, you may get unexpected results, depending on which parameters you specify for CloudFront to use as a basis for caching, and the order in which the parameters appear in the query string.

  The following examples show what happens if you use a different delimiter and you configure CloudFront to cache based only on the `color` parameter:

  - In the following request, CloudFront caches your content based on the value of the `color` parameter, but CloudFront interprets the value as *`red;size=large`*:

`https://d111111abcdef8.cloudfront.net/images/image.jpg?color=red;size=large`

- In the following request, CloudFront caches your content but doesn't base caching on the query string parameters. This is because you configured CloudFront to cache based on the `color` parameter, but CloudFront interprets the following string as containing only a `size` parameter that has a value of `large;color=red`:

  `https://d111111abcdef8.cloudfront.net/images/image.jpg?size=large;color=red`

You can configure CloudFront to do one of the following:

- Don't forward query strings to the origin at all. If you don't forward query strings, CloudFront doesn't cache based on query string parameters.
- Forward query strings to the origin, and cache based on all parameters in the query string.
- Forward query strings to the origin, and cache based on specified parameters in the query string.

For more information, see the section called “Optimizing caching”.

Topics

- Console and API settings for query string forwarding and caching
- Optimizing caching
- Query string parameters and CloudFront standard logs (access logs)

Console and API settings for query string forwarding and caching

To configure query string forwarding and caching in the CloudFront console, see the following settings in the section called “Values that you specify”:

- the section called “Query string forwarding and caching”
- the section called “Query string allowlist”

To configure query string forwarding and caching with the CloudFront API, see the following settings in DistributionConfig and in DistributionConfigWithTags in the *Amazon CloudFront API Reference*:

- `QueryString`
- `QueryStringCacheKeys`

Optimizing caching

When you configure CloudFront to cache based on query string parameters, you can take the following steps to reduce the number of requests that CloudFront forwards to your origin. When CloudFront edge locations serve objects, you reduce the load on your origin server and reduce latency because objects are served from locations that are closer to your users.

Cache based only on parameters for which your origin returns different versions of an object

For each query string parameter that your web application forwards to CloudFront, CloudFront forwards requests to your origin for every parameter value and caches a separate version of the object for every parameter value. This is true even if your origin always returns the same object regardless of the parameter value. For multiple parameters, the number of requests and the number of objects multiply.

We recommend that you configure CloudFront to cache based only on the query string parameters for which your origin returns different versions, and that you carefully consider the merits of caching based on each parameter. For example, suppose you have a retail website. You have pictures of a jacket in six different colors, and the jacket comes in 10 different sizes. The pictures that you have of the jacket show the different colors but not the different sizes. To optimize caching, you should configure CloudFront to cache based only on the color parameter, not on the size parameter. This increases the likelihood that CloudFront can serve a request from the cache, which improves performance and reduces the load on your origin.

Always list parameters in the same order

The order of parameters matters in query strings. In the following example, the query strings are identical except that the parameters are in a different order. This causes CloudFront to forward two separate requests for image.jpg to your origin and to cache two separate versions of the object:

- `https://d111111abcdef8.cloudfront.net/images/image.jpg?color=red&size=large`

- `https://d111111abcdef8.cloudfront.net/images/image.jpg?size=large&color=red`

We recommend that you always list parameter names in the same order, such as alphabetical order.

Always use the same case for parameter names and values

CloudFront considers the case of parameter names and values when caching based on query string parameters. In the following example, the query strings are identical except for the case of parameter names and values. This causes CloudFront to forward four separate requests for image.jpg to your origin and to cache four separate versions of the object:

- `https://d111111abcdef8.cloudfront.net/images/image.jpg?color=red`
- `https://d111111abcdef8.cloudfront.net/images/image.jpg?color=Red`
- `https://d111111abcdef8.cloudfront.net/images/image.jpg?Color=red`
- `https://d111111abcdef8.cloudfront.net/images/image.jpg?Color=Red`

We recommend that you use case consistently for parameter names and values, such as all lowercase.

Don't use parameter names that conflict with signed URLs

If you're using signed URLs to restrict access to your content (if you added trusted signers to your distribution), CloudFront removes the following query string parameters before forwarding the rest of the URL to your origin:

- `Expires`
- `Key-Pair-Id`
- `Policy`
- `Signature`

If you're using signed URLs and you want to configure CloudFront to forward query strings to your origin, your own query string parameters cannot be named `Expires`, `Key-Pair-Id`, `Policy`, or `Signature`.

Query string parameters and CloudFront standard logs (access logs)

If you enable logging, CloudFront logs the full URL, including query string parameters. This is true regardless of whether you have configured CloudFront to forward query strings to the origin. For

more information about CloudFront logging, see the section called “Using standard logs (access logs)”.

Caching content based on cookies

By default, CloudFront doesn’t consider cookies when processing requests and responses, or when caching your objects in edge locations. If CloudFront receives two requests that are identical except for what’s in the `Cookie` header, then, by default, CloudFront treats the requests as identical and returns the same object for both requests.

You can configure CloudFront to forward to your origin some or all of the cookies in viewer requests, and to cache separate versions of your objects based on the cookie values that it forwards. When you do this, CloudFront uses some or all of the cookies in viewer requests—whichever ones it’s configured to forward—to uniquely identify an object in the cache.

For example, suppose that requests for `locations.html` contain a `country` cookie that has a value of either `uk` or `fr`. When you configure CloudFront to cache your objects based on the value of the `country` cookie, CloudFront forwards requests for `locations.html` to the origin and includes the `country` cookie and its value. Your origin returns `locations.html`, and CloudFront caches the object once for requests in which the value of the `country` cookie is `uk` and once for requests in which the value is `fr`.

Important

Amazon S3 and some HTTP servers don’t process cookies. Don’t configure CloudFront to forward cookies to an origin that doesn’t process cookies or doesn’t vary its response based on cookies. That can cause CloudFront to forward more requests to the origin for the same object, which slows performance and increases the load on the origin. If, considering the previous example, your origin doesn’t process the `country` cookie or always returns the same version of `locations.html` to CloudFront regardless of the value of the `country` cookie, don’t configure CloudFront to forward that cookie.

Conversely, if your custom origin depends on a particular cookie or sends different responses based on a cookie, make sure you configure CloudFront to forward that cookie to the origin. Otherwise, CloudFront removes the cookie before forwarding the request to your origin.

To configure cookie forwarding, you update your distribution's cache behavior. For more information about cache behaviors, see Cache behavior settings, particularly the Forward cookies and Allowlist cookies sections.

You can configure each cache behavior to do one of the following:

- **Forward all cookies to your origin –** CloudFront includes all cookies sent by the viewer when it forwards requests to the origin. When your origin returns a response, CloudFront caches the response using the cookie names and values in the viewer request. If the origin response includes `Set-Cookie` headers, CloudFront returns them to the viewer with the requested object. CloudFront also caches the `Set-Cookie` headers with the object returned from the origin, and sends those `Set-Cookie` headers to viewers on all cache hits.
- **Forward a set of cookies that you specify –** CloudFront removes any cookies that the viewer sends that aren't on the allowlist before it forwards a request to the origin. CloudFront caches the response using the listed cookies names and values in the viewer request. If the origin response includes `Set-Cookie` headers, CloudFront returns them to the viewer with the requested object. CloudFront also caches the `Set-Cookie` headers with the object returned from the origin, and sends those `Set-Cookie` headers to viewers on all cache hits.

  For information about specifying wildcards in cookie names, see Allowlist cookies.

  For the current quota on the number of cookie names that you can forward for each cache behavior, or to request a higher quota, see Quotas on query strings (legacy cache settings).
- **Don't forward cookies to your origin –** CloudFront doesn't cache your objects based on cookie sent by the viewer. In addition, CloudFront removes cookies before forwarding requests to your origin, and removes `Set-Cookie` headers from responses before returning responses to your viewers. Because this isn't an optimal way to use your origin resources, when you select this cache behavior, you should make sure that your origin doesn't include cookies in origin responses by default.

Note the following about specifying the cookies that you want to forward:

Access logs

If you configure CloudFront to log requests and to log cookies, CloudFront logs all cookies and all cookie attributes, even if you configure CloudFront not to forward cookies to your origin or if you configure CloudFront to forward only specific cookies. For more information about CloudFront logging, see Configuring and using standard logs (access logs).

Case sensitivity

Cookie names and values are both case-sensitive. For example, if CloudFront is configured to forward all cookies, and two viewer requests for the same object have cookies that are identical except for case, CloudFront caches the object twice.

CloudFront sorts cookies

If CloudFront is configured to forward cookies (all or a subset), CloudFront sorts the cookies in natural order by cookie name before forwarding the request to your origin.

`If-Modified-Since` and `If-None-Match`

`If-Modified-Since` and `If-None-Match` conditional requests are not supported when CloudFront is configured to forward cookies (all or a subset).

Standard name–value pair format is required

CloudFront forwards a cookie header only if the value conforms to the standard name–value pair format, for example: `"Cookie: cookie1=value1; cookie2=value2"`

Disable caching of `Set-Cookie` headers

If CloudFront is configured to forward cookies to the origin (whether all or specific cookies), it also caches the `Set-Cookie` headers received in the origin response. CloudFront includes these `Set-Cookie` headers in its response to the original viewer, and also includes them in subsequent responses that are served from the CloudFront cache.

If you want to receive cookies at your origin but you don't want CloudFront to cache the `Set-Cookie` headers in your origin's responses, configure your origin to add a `Cache-Control` header with a `no-cache` directive that specifies `Set-Cookie` as a field name. For example: `Cache-Control: no-cache="Set-Cookie"`. For more information, see Response Cache-Control Directives in the *Hypertext Transfer Protocol (HTTP/1.1): Caching* standard.

Maximum length of cookie names

If you configure CloudFront to forward specific cookies to your origin, the total number of bytes in all of the cookie names that you configure CloudFront to forward can't exceed 512 minus the number of cookies that you're forwarding. For example, if you configure CloudFront to forward 10 cookies to your origin, the combined length of the names of the 10 cookies can't exceed 502 bytes (512 – 10).

If you configure CloudFront to forward all cookies to your origin, the length of cookie names doesn't matter.

For information about using the CloudFront console to update a distribution so CloudFront forwards cookies to the origin, see Updating a distribution. For information about using the CloudFront API to update a distribution, see UpdateDistribution in the *Amazon CloudFront API Reference*.

Caching content based on request headers

CloudFront lets you choose whether you want CloudFront to forward headers to your origin and to cache separate versions of a specified object based on the header values in viewer requests. This allows you to serve different versions of your content based on the device the user is using, the location of the viewer, the language the viewer is using, and a variety of other criteria.

Topics

- Headers and distributions – overview
- Selecting the headers to base caching on
- Configuring CloudFront to respect CORS settings
- Configuring caching based on the device type
- Configuring caching based on the language of the viewer
- Configuring caching based on the location of the viewer
- Configuring caching based on the protocol of the request
- Configuring caching for compressed files
- How caching based on headers affects performance
- How the case of headers and header values affects caching
- Headers that CloudFront returns to the viewer

Headers and distributions – overview

By default, CloudFront doesn't consider headers when caching your objects in edge locations. If your origin returns two objects and they differ only by the values in the request headers, CloudFront caches only one version of the object.

You can configure CloudFront to forward headers to the origin, which causes CloudFront to cache multiple versions of an object based on the values in one or more request headers. To configure

CloudFront to cache objects based on the values of specific headers, you specify cache behavior settings for your distribution. For more information, see Cache Based on Selected Request Headers.

For example, suppose viewer requests for `logo.jpg` contain a custom `Product` header that has a value of either `Acme` or `Apex`. When you configure CloudFront to cache your objects based on the value of the `Product` header, CloudFront forwards requests for `logo.jpg` to the origin and includes the `Product` header and header values. CloudFront caches `logo.jpg` once for requests in which the value of the `Product` header is `Acme` and once for requests in which the value is `Apex`.

You can configure each cache behavior in a distribution to do one of the following:

- Forward all headers to your origin

> **Note**
>
> **For legacy cache settings** – If you configure CloudFront to forward all headers to your origin, CloudFront doesn't cache the objects associated with this cache behavior. Instead, it sends every request to the origin.

- Forward a list of headers that you specify. CloudFront caches your objects based on the values in all of the specified headers. CloudFront also forwards the headers that it forwards by default, but it caches your objects based only on the headers that you specify.
- Forward only the default headers. In this configuration, CloudFront doesn't cache your objects based on the values in the request headers.

For the current quota on the number of headers that you can forward for each cache behavior or to request a higher quota, see Quotas on headers.

For information about using the CloudFront console to update a distribution so CloudFront forwards headers to the origin, see Updating a distribution. For information about using the CloudFront API to update an existing distribution, see Update Distribution in the *Amazon CloudFront API Reference*.

Selecting the headers to base caching on

The headers that you can forward to the origin and that CloudFront bases caching on depend on whether your origin is an Amazon S3 bucket or a custom origin.

- **Amazon S3 –** You can configure CloudFront to forward and to cache your objects based on a number of specific headers (see the following list of exceptions). However, we recommend that you avoid forwarding headers with an Amazon S3 origin unless you need to implement cross-origin resource sharing (CORS) or you want to personalize content by using Lambda@Edge in origin-facing events.
  - To configure CORS, you must forward headers that allow CloudFront to distribute content for websites that are enabled for cross-origin resource sharing (CORS). For more information, see Configuring CloudFront to respect CORS settings.
  - To personalize content by using headers that you forward to your Amazon S3 origin, you write and add Lambda@Edge functions and associate them with your CloudFront distribution to be triggered by an origin-facing event. For more information about working with headers to personalize content, see Personalize content by country or device type headers - examples.

    We recommend that you avoid forwarding headers that you aren't using to personalize content because forwarding extra headers can reduce your cache hit ratio. That is, CloudFront can't serve as many requests from edge caches, as a proportion of all requests.
- **Custom origin** – You can configure CloudFront to cache based on the value of any request header except the following:
  - `Connection`
  - `Cookie` – If you want to forward and cache based on cookies, you use a separate setting in your distribution. For more information, see Caching content based on cookies.
  - `Host (for Amazon S3 origins)`
  - `Proxy-Authorization`
  - `TE`
  - `Upgrade`

  You can configure CloudFront to cache objects based on values in the `Date` and `User-Agent` headers, but we don't recommend it. These headers have numerous possible values, and caching based on their values could cause CloudFront to forward significantly more requests to your origin.

For a full list of HTTP request headers and how CloudFront processes them, see HTTP request headers and CloudFront behavior (custom and Amazon S3 origins).

Configuring CloudFront to respect CORS settings

If you have enabled cross-origin resource sharing (CORS) on an Amazon S3 bucket or a custom origin, you must choose specific headers to forward, to respect the CORS settings. The headers that you must forward differ depending on the origin (Amazon S3 or custom) and whether you want to cache `OPTIONS` responses.

Amazon S3

- If you want `OPTIONS` responses to be cached, do the following:
  - Choose the options for default cache behavior settings that enable caching for `OPTIONS` responses.
  - Configure CloudFront to forward the following headers: `Origin`, `Access-Control-Request-Headers`, and `Access-Control-Request-Method`.
- If you don't want `OPTIONS` responses to be cached, configure CloudFront to forward the `Origin` header, together with any other headers required by your origin (for example, `Access-Control-Request-Headers`, `Access-Control-Request-Method`, or others).

Custom origins – Forward the `Origin` header along with any other headers required by your origin.

To configure CloudFront to cache responses based on CORS, you must configure CloudFront to forward headers by using a cache policy. For more information, see *Working with policies*.

For more information about CORS and Amazon S3, see Using cross-origin resource sharing (CORS) in the *Amazon Simple Storage Service User Guide*.

Configuring caching based on the device type

If you want CloudFront to cache different versions of your objects based on the device a user is using to view your content, configure CloudFront to forward the applicable headers to your custom origin:

- `CloudFront-Is-Desktop-Viewer`
- `CloudFront-Is-Mobile-Viewer`
- `CloudFront-Is-SmartTV-Viewer`
- `CloudFront-Is-Tablet-Viewer`

Based on the value of the `User-Agent` header, CloudFront sets the value of these headers to `true` or `false` before forwarding the request to your origin. If a device falls into more than one category, more than one value might be `true`. For example, for some tablet devices, CloudFront might set both `CloudFront-Is-Mobile-Viewer` and `CloudFront-Is-Tablet-Viewer` to `true`.

Configuring caching based on the language of the viewer

If you want CloudFront to cache different versions of your objects based on the language specified in the request, configure CloudFront to forward the `Accept-Language` header to your origin.

Configuring caching based on the location of the viewer

If you want CloudFront to cache different versions of your objects based on the country that the request came from, configure CloudFront to forward the `CloudFront-Viewer-Country` header to your origin. CloudFront automatically converts the IP address that the request came from into a two-letter country code. For an easy-to-use list of country codes, sortable by code and by country name, see the Wikipedia entry ISO 3166-1 alpha-2.

Configuring caching based on the protocol of the request

If you want CloudFront to cache different versions of your objects based on the protocol of the request, HTTP or HTTPS, configure CloudFront to forward the `CloudFront-Forwarded-Proto` header to your origin.

Configuring caching for compressed files

If your origin supports Brotli compression, you can cache based on the `Accept-Encoding` header. Configure caching based on `Accept-Encoding` only if your origin serves different content based on the header.

How caching based on headers affects performance

When you configure CloudFront to cache based on one or more headers and the headers have more than one possible value, CloudFront forwards more requests to your origin server for the same object. This slows performance and increases the load on your origin server. If your origin server returns the same object regardless of the value of a given header, we recommend that you don't configure CloudFront to cache based on that header.

If you configure CloudFront to forward more than one header, the order of the headers in viewer requests doesn't affect caching as long as the values are the same. For example, if one request contains the headers A:1,B:2 and another request contains B:2,A:1, CloudFront caches just one copy of the object.

How the case of headers and header values affects caching

When CloudFront caches based on header values, it doesn't consider the case of the header name, but it does consider the case of the header value:

- If viewer requests include both `Product:Acme` and `product:Acme`, CloudFront caches an object only once. The only difference between them is the case of the header name, which doesn't affect caching.
- If viewer requests include both `Product:Acme` and `Product:acme`, CloudFront caches an object twice, because the value is `Acme` in some requests and `acme` in others.

Headers that CloudFront returns to the viewer

Configuring CloudFront to forward and cache headers does not affect which headers CloudFront returns to the viewer. CloudFront returns all of the headers that it gets from the origin with a few exceptions. For more information, see the applicable topic:

- **Amazon S3 origins –** See HTTP response headers that CloudFront removes or updates.
- **Custom origins –** See HTTP response headers that CloudFront removes or replaces.

Troubleshooting

Troubleshoot common problems you might encounter when setting up Amazon CloudFront to distribute your content or when using Lambda@Edge, and find possible solutions.

Topics

- Troubleshooting distribution issues
- Troubleshooting error responses from your origin
- Load testing CloudFront

Troubleshooting distribution issues

Use the information here to help you diagnose and fix certificate errors, access-denied issues, or other common issues that you might encounter when setting up your website or application with Amazon CloudFront distributions.

Topics

- CloudFront returns an InvalidViewerCertificate error when I try to add an alternate domain name
- I can't view the files in my distribution
- Error message: Certificate: <certificate-id> is being used by CloudFront

CloudFront returns an InvalidViewerCertificate error when I try to add an alternate domain name

If CloudFront returns an `InvalidViewerCertificate` error when you try to add an alternate domain name (CNAME) to your distribution, review the following information to help troubleshoot the problem. This error can indicate that one of the following issues must be resolved before you can successfully add the alternate domain name.

The following errors are listed in the order in which CloudFront checks for authorization to add an alternate domain name. This can help you troubleshoot issues because based on the error that CloudFront returns, you can tell which verification checks have completed successfully.

There's no certificate attached to your distribution.

To add an alternate domain name (CNAME), you must attach a trusted, valid certificate to your distribution. Please review the requirements, obtain a valid certificate that meets them, attach it to your distribution, and then try again. For more information, see Requirements for using alternate domain names.

There are too many certificates in the certificate chain for the certificate that you've attached.

You can only have up to five certificates in a certificate chain. Reduce the number of certificates in the chain, and then try again.

The certificate chain includes one or more certificates that aren't valid for the current date.

The certificate chain for a certificate that you have added has one or more certificates that aren't valid, either because a certificate isn't valid yet or a certificate has expired. Check the **Not Valid Before** and **Not Valid After** fields in the certificates in your certificate chain to make sure that all of the certificates are valid based on the dates that you've listed.

The certificate that you've attached isn't signed by a trusted Certificate Authority (CA).

The certificate that you attach to CloudFront to verify an alternate domain name cannot be a self-signed certificate. It must be signed by a trusted CA. For more information, see Requirements for using alternate domain names.

The certificate that you've attached isn't formatted correctly

The domain name and IP address format that are included in the certificate, and the format of the certificate itself, must follow the standard for certificates.

There was a CloudFront internal error.

CloudFront was blocked by an internal issue and couldn't make validation checks for certificates. In this scenario, CloudFront returns an HTTP 500 status code and indicates that there is an internal CloudFront problem with attaching the certificate. Wait a few minutes, and then try again to add the alternate domain name with the certificate.

The certificate that you've attached doesn't cover the alternate domain name that you're trying to add.

For each alternate domain name that you add, CloudFront requires that you attach a valid SSL/TLS certificate from a trusted Certificate Authority (CA) that covers the domain name, to validate your authorization to use it. Please update your certificate to include a domain name that covers the CNAME that you're trying to add. For more information and examples of using domain names with wildcards, see Requirements for using alternate domain names.

I can't view the files in my distribution

If you can't view the files in your CloudFront distribution, see the following topics for some common solutions.

Did you sign up for both CloudFront and Amazon S3?

To use Amazon CloudFront with an Amazon S3 origin, you must sign up for both CloudFront and Amazon S3, separately. For more information about signing up for CloudFront and Amazon S3, see Setting up.

Are your Amazon S3 bucket and object permissions set correctly?

If you are using CloudFront with an Amazon S3 origin, the original versions of your content are stored in an S3 bucket. The easiest way to use CloudFront with Amazon S3 is to make all of your objects publicly readable in Amazon S3. To do this, you must explicitly enable public read privileges for each object that you upload to Amazon S3.

If your content is not publicly readable, you must create a CloudFront origin access control (OAC) so that CloudFront can access it. For more information about CloudFront origin access control, see the section called “Restricting access to an Amazon Simple Storage Service origin”.

Object properties and bucket properties are independent. You must explicitly grant privileges to each object in Amazon S3. Objects do not inherit properties from buckets, and object properties must be set independently of the bucket.

Is your alternate domain name (CNAME) correctly configured?

If you already have an existing CNAME record for your domain name, update that record or replace it with a new one that points to your distribution's domain name.

Also, make sure that your CNAME record points to your distribution's domain name, not your Amazon S3 bucket. You can confirm that the CNAME record in your DNS system points to your distribution's domain name. To do so, use a DNS tool like **dig**.

The following example shows a dig request for a domain name called `images.example.com` and the relevant part of the response. Under `ANSWER SECTION`, see the line that contains `CNAME`. The CNAME record for your domain name is set up correctly if the value on the right side of CNAME is your CloudFront distribution's domain name. If it's your Amazon S3 origin server bucket or some other domain name, then the CNAME record is set up incorrectly.

```
[prompt]> dig images.example.com

; <<> DiG 9.3.3rc2 <<> images.example.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15917
;; flags: qr rd ra; QUERY: 1, ANSWER: 9, AUTHORITY: 2, ADDITIONAL: 0
;; QUESTION SECTION:
;images.example.com.   IN  A
;; ANSWER SECTION:
images.example.com. 10800 IN CNAME d111111abcdef8.cloudfront.net.
...
...
```

For more information about CNAMEs, see Using custom URLs by adding alternate domain names (CNAMEs).

Are you referencing the correct URL for your CloudFront distribution?

Make sure that the URL that you're referencing uses the domain name (or CNAME) of your CloudFront distribution, not your Amazon S3 bucket or custom origin.

Do you need help troubleshooting a custom origin?

If you need Amazon to help you troubleshoot a custom origin, we probably will need to inspect the `X-Amz-Cf-Id` header entries from your requests. If you are not already logging these entries, you might want to consider it for the future. For more information, see the section called "Using Amazon EC2 (or another custom origin)". For further help, see the Amazon Support Center.

Error message: Certificate: <certificate-id> is being used by CloudFront

Problem: You're trying to delete an SSL/TLS certificate from the IAM certificate store, and you're getting the message "Certificate: <certificate-id> is being used by CloudFront."

Solution: Every CloudFront distribution must be associated either with the default CloudFront certificate or with a custom SSL/TLS certificate. Before you can delete an SSL/TLS certificate, you must either rotate the certificate (replace the current custom SSL/TLS certificate with another custom SSL/TLS certificate) or revert from using a custom SSL/TLS certificate to using the default CloudFront certificate. To fix that, complete the steps in one of the following procedures:

- Rotating SSL/TLS certificates

- Reverting from a custom SSL/TLS certificate to the default CloudFront certificate

Troubleshooting error responses from your origin

If CloudFront requests an object from your origin, and the origin returns an HTTP 4xx or 5xx status code, there's a problem with communication between CloudFront and your origin. The following topics describe common causes for some of these HTTP status codes, and some possible solutions.

Topics

- HTTP 400 status code (Bad Request)
- HTTP 502 status code (Bad Gateway)
- HTTP 502 status code (Lambda validation error)
- HTTP 502 status code (DNS error)
- HTTP 503 status code (function execution error)
- HTTP 503 status code (Lambda limit exceeded)
- HTTP 503 status code (Service Unavailable)
- HTTP 504 status code (Gateway Timeout)

HTTP 400 status code (Bad Request)

Your CloudFront distribution might send error responses with HTTP status code 400 Bad Request, and a message similar to the following:

The authorization header is malformed; the region '<Amazon Region>' is wrong; expecting '<Amazon Region>'

For example:

The authorization header is malformed; the region 'us-east-1' is wrong; expecting 'us-west-2'

This problem can occur in the following scenario:

1. Your CloudFront distribution's origin is an Amazon S3 bucket.
2. You moved the S3 bucket from one Amazon Region to another. That is, you deleted the S3 bucket, then later you created a new bucket with the same bucket name, but in a different Amazon Region than where the original S3 bucket was located.

To fix this error, update your CloudFront distribution so that it finds the S3 bucket in the bucket's current Amazon Region.

To update your CloudFront distribution

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose the distribution that produces this error.
3. Choose **Origins and Origin Groups**.
4. Find the origin for the S3 bucket that you moved. Select the check box next to this origin, then choose **Edit**.
5. Choose **Yes, Edit**. You do not need to change any settings before choosing **Yes, Edit**.

When you complete these steps, CloudFront redeploys your distribution. While the distribution is deploying, you see the **Deploying** status under the **Last modified** column. Some time after the deployment is complete, you should stop receiving the `AuthorizationHeaderMalformed` error responses.

HTTP 502 status code (Bad Gateway)

An HTTP 502 status code (Bad Gateway) indicates that CloudFront wasn't able to serve the requested object because it couldn't connect to the origin server.

Topics

- SSL/TLS negotiation failure between CloudFront and a custom origin server
- Origin is not responding with supported ciphers/protocols
- SSL/TLS certificate on the origin is expired, invalid, self-signed, or the certificate chain is in the wrong order
- Origin is not responding on specified ports in origin settings

SSL/TLS negotiation failure between CloudFront and a custom origin server

If you use a custom origin and you configured CloudFront to require HTTPS between CloudFront and your origin, the problem might be mismatched domain names. The SSL/TLS certificate that is installed on your origin includes a domain name in the **Common Name** field and possibly

several more in the **Subject Alternative Names** field. (CloudFront supports wildcard characters in certificate domain names.) One of the domain names in the certificate must match one or both of the following values:

- The value that you specified for **Origin Domain** for the applicable origin in your distribution.
- The value of the `Host` header if you configured CloudFront to forward the `Host` header to your origin. For more information about forwarding the `Host` header to your origin, see Caching content based on request headers.

If the domain names don't match, the SSL/TLS handshake fails, and CloudFront returns an HTTP status code 502 (Bad Gateway) and sets the `X-Cache` header to `Error from cloudfront`.

To determine whether domain names in the certificate match the **Origin Domain** in the distribution or the `Host` header, you can use an online SSL checker or OpenSSL. If the domain names don't match, you have two options:

- Get a new SSL/TLS certificate that includes the applicable domain names.

  If you use Amazon Certificate Manager (ACM), see Requesting a public certificate in the *Amazon Certificate Manager User Guide* to request a new certificate.
- Change the distribution configuration so CloudFront no longer tries to use SSL to connect with your origin.

Online SSL checker

To find an SSL test tool, search the internet for "online ssl checker." Typically, you specify the name of your domain, and the tool returns a variety of information about your SSL/TLS certificate. Confirm that the certificate contains your domain name in the **Common Name** or **Subject Alternative Names** fields.

OpenSSL

To help troubleshoot HTTP 502 errors from CloudFront, you can use OpenSSL to try to make an SSL/TLS connection to your origin server. If OpenSSL is not able to make a connection, that can indicate a problem with your origin server's SSL/TLS configuration. If OpenSSL is able to make a connection, it returns information about the origin server's certificate, including the certificate's common name (`Subject CN` field) and subject alternative name (`Subject Alternative Name` field).

Use the following OpenSSL command to test the connection to your origin server (replace *origin domain* with your origin server's domain name, such as example.com):

```
openssl s_client -connect origin domain name:443
```

If the following are true:

- Your origin server supports multiple domain names with multiple SSL/TLS certificates
- Your distribution is configured to forward the `Host` header to the origin

Then add the `-servername` option to the OpenSSL command, as in the following example (replace *CNAME* with the CNAME that's configured in your distribution):

```
openssl s_client -connect origin domain name:443 -servername CNAME
```

Origin is not responding with supported ciphers/protocols

CloudFront connects to origin servers using ciphers and protocols. For a list of the ciphers and protocols that CloudFront supports, see the section called "Supported protocols and ciphers between CloudFront and the origin". If your origin does not respond with one of these ciphers or protocols in the SSL/TLS exchange, CloudFront fails to connect. You can validate that your origin supports the ciphers and protocols by using an online tool such as SSL Labs. Type the domain name of your origin in the **Hostname** field, and then choose **Submit**. Review the **Common names** and **Alternative names** fields from the test to see if they match your origin's domain name. After the test is finished, find the **Protocols** and **Cipher Suites** sections in the test results to see which ciphers or protocols are supported by your origin. Compare them with the list of the section called "Supported protocols and ciphers between CloudFront and the origin".

SSL/TLS certificate on the origin is expired, invalid, self-signed, or the certificate chain is in the wrong order

If the origin server returns the following, CloudFront drops the TCP connection, returns HTTP status code 502 (Bad Gateway), and sets the `X-Cache` header to `Error from cloudfront`:

- An expired certificate
- Invalid certificate
- Self-signed certificate
- Certificate chain in the wrong order

> **Note**
>
> If the full chain of certificates, including the intermediate certificate, is not present, CloudFront drops the TCP connection.

For information about installing an SSL/TLS certificate on your custom origin server, see the section called "Requiring HTTPS to a custom origin".

Origin is not responding on specified ports in origin settings

When you create an origin on your CloudFront distribution, you can set the ports that CloudFront connects to the origin with for HTTP and HTTPS traffic. By default, these are TCP 80/443. You have the option to modify these ports. If your origin is rejecting traffic on these ports for any reason, or if your backend server isn't responding on the ports, CloudFront will fail to connect.

To troubleshoot these issues, check any firewalls running in your infrastructure and validate that they are not blocking the supported IP ranges. For more information, see Amazon IP address ranges in the *Amazon Web Services General Reference*. Additionally, verify whether your web server is running on the origin.

HTTP 502 status code (Lambda validation error)

If you're using Lambda@Edge, an HTTP 502 status code can indicate that your Lambda function response was incorrectly formed or included invalid content. For more information about troubleshooting Lambda@Edge errors, see Testing and debugging Lambda@Edge functions.

HTTP 502 status code (DNS error)

An HTTP 502 error with the `NonS3OriginDnsError` error code indicates that there's a DNS configuration problem that prevents CloudFront from connecting to the origin. If you get this error from CloudFront, make sure that the origin's DNS configuration is correct and working.

When CloudFront receives a request for an object that's expired or is not in its cache, it makes a request to the origin to get the object. To make a successful request to the origin, CloudFront performs a DNS resolution on the origin domain. If the DNS service for your domain is experiencing issues, CloudFront can't resolve the domain name to get the IP address, which results in an HTTP 502 error (`NonS3OriginDnsError`). To fix this problem, contact your DNS provider, or, if you are using Amazon Route 53, see Why can't I access my website that uses Route 53 DNS services?

To further troubleshoot this issue, ensure that the authoritative name servers of your origin's root domain or zone apex (such as `example.com`) are functioning correctly. You can use the following commands to find the name servers for your apex origin, with a tool such as dig or nslookup:

```
dig OriginAPEXDomainName NS +short
```

```
nslookup -query=NS OriginAPEXDomainName
```

When you have the names of your name servers, use the following commands to query the domain name of your origin against them to make sure that each responds with an answer:

```
dig OriginDomainName @NameServer
```

```
nslookup OriginDomainName NameServer
```

Important

Make sure that you perform this DNS troubleshooting using a computer that's connected to the public internet. CloudFront resolves the origin domain using public DNS on the internet, so it's important to troubleshoot in a similar context.

If your origin is a subdomain whose DNS authority is delegated to a different name server than the root domain, make sure that the name server (NS) and start of authority (SOA) records are configured correctly for the subdomain. You can check for these records using commands similar to the preceding examples.

For more information about DNS, see Domain Name System (DNS) concepts in the Amazon Route 53 documentation.

HTTP 503 status code (function execution error)

If you're using Lambda@Edge or CloudFront Functions, an HTTP 503 status code can indicate that your function returned an execution error.

For more information about troubleshooting Lambda@Edge errors, see Testing and debugging Lambda@Edge functions.

For more information about troubleshooting CloudFront Functions, see Testing functions.

HTTP 503 status code (Lambda limit exceeded)

If you're using Lambda@Edge, an HTTP 503 status code can indicate that the Lambda service returned an error. The error might be caused by one of the following:

- The number of function executions exceeded one of the quotas (formerly known as limits) that Lambda sets to throttle executions in an Amazon Region (concurrent executions or invocation frequency).
- The function exceeded the Lambda function timeout quota.

For more information about the Amazon Lambda quotas, see Lambda quotas in the *Amazon Lambda Developer Guide*. For more information about troubleshooting Lambda@Edge errors, see the section called "Testing and debugging".

HTTP 503 status code (Service Unavailable)

An HTTP 503 status code (Service Unavailable) typically indicates a performance issue on the origin server. In rare cases, it indicates that CloudFront temporarily can't satisfy a request because of resource constraints at an edge location.

Topics

- Origin server does not have enough capacity to support the request rate
- CloudFront caused the error due to resource constraints at the edge location

Origin server does not have enough capacity to support the request rate

CloudFront generates this error when the origin server is overwhelmed with incoming requests. CloudFront then relays the error back to the user. To resolve this issue, try the following solutions:

- If you use Amazon S3 as your origin server, optimize the performance of Amazon S3 by following the best practices for key naming. For more information, see Optimizing Amazon S3 performance in the *Amazon Simple Storage Service User Guide*.
- If you use Elastic Load Balancing as your origin server, see How do I troubleshoot 503 errors returned while using Classic Load Balancer?
- If you use a custom origin, examine the application logs to ensure that your origin has sufficient resources, such as memory, CPU, and disk size. If you use Amazon EC2 as the backend, make sure

that the instance type has the appropriate resources to fulfill the incoming requests. For more information, see Instance types in the *Amazon EC2 User Guide for Linux Instances*.

CloudFront caused the error due to resource constraints at the edge location

You will receive this error in the rare situation that CloudFront can't route requests to the next best available edge location, and so can't satisfy a request. This error is common when you perform load testing on your CloudFront distribution. To help prevent this, follow the the section called "Load testing CloudFront" guidelines for avoiding 503 (capacity exceeded) errors.

If this happens in your production environment, contact Amazon Support.

HTTP 504 status code (Gateway Timeout)

An HTTP 504 status code (Gateway Timeout) indicates that when CloudFront forwarded a request to the origin (because the requested object wasn't in the edge cache), one of the following happened:

- The origin returned an HTTP 504 status code to CloudFront.
- The origin didn't respond before the request expired.

CloudFront will return an HTTP 504 status code if traffic is blocked to the origin by a firewall or security group, or if the origin isn't accessible on the internet. Check for those issues first. Then, if access isn't the problem, explore application delays and server timeouts to help you identify and fix the issues.

Topics

- Configure the firewall on your origin server to allow CloudFront traffic
- Configure the security groups on your origin server to allow CloudFront traffic
- Make your custom origin server accessible on the internet
- Find and fix delayed responses from applications on your origin server

Configure the firewall on your origin server to allow CloudFront traffic

If the firewall on your origin server blocks CloudFront traffic, CloudFront returns an HTTP 504 status code, so it's good to make sure that isn't the issue before checking for other problems.

The method that you use to determine if this is an issue with your firewall depends on what system your origin server uses:

- If you use an IPTable firewall on a Linux server, you can search for tools and information to help you work with IPTables.
- If you use Windows Firewall on a Windows server, see Add or Edit Firewall Rule in the Microsoft documentation.

When you evaluate the firewall configuration on your origin server, look for any firewalls or security rules that block traffic from CloudFront edge locations, based on the published IP address range.

If the CloudFront IP address range is allowed to connect to your origin server, make sure to update your server's security rules to incorporate changes. You can subscribe to an Amazon SNS topic and receive notifications when the IP address range file is updated. After you receive the notification, you can use code to retrieve the file, parse it, and make adjustments for your local environment. For more information, see Subscribe to Amazon Public IP Address Changes via Amazon SNS on the Amazon News Blog.

Configure the security groups on your origin server to allow CloudFront traffic

If your origin uses Elastic Load Balancing, review the ELB security groups and make sure that the security groups allow inbound traffic from CloudFront.

You can also use Amazon Lambda to automatically update your security groups to allow inbound traffic from CloudFront.

Make your custom origin server accessible on the internet

If CloudFront can't access your custom origin server because it isn't publicly available on the internet, CloudFront returns an HTTP 504 error.

CloudFront edge locations connect to origin servers through the internet. If your custom origin is on a private network, CloudFront can't reach it. Because of this, you can't use private servers, including internal Classic Load Balancers, as origin servers with CloudFront.

To check that internet traffic can connect to your origin server, run the following commands (where OriginDomainName is the domain name for your server):

For HTTPS traffic:

- nc -zv OriginDomainName 443
- telnet OriginDomainName 443

For HTTP traffic:

- nc -zv OriginDomainName 80
- telnet OriginDomainName 80

Find and fix delayed responses from applications on your origin server

Server timeouts are often the result of either an application taking a very long time to respond, or a timeout value that is set too low.

A quick fix to help avoid HTTP 504 errors is to simply set a higher CloudFront timeout value for your distribution. But we recommend that you first make sure that you address any performance and latency issues with the application and origin server. Then you can set a reasonable timeout value that helps prevent HTTP 504 errors and provides good responsiveness to users.

Here's an overview of the steps you can take to find performance issues and correct them:

1. Measure the typical and high-load latency (responsiveness) of your web application.
2. Add additional resources, such as CPU or memory, if needed. Take other steps to address issues, such as tuning database queries to accommodate high-load scenarios.
3. If needed, adjust the timeout value for your CloudFront distribution.

Following are details about each step.

Measure typical and high-load latency

To determine if one or more backend web application servers are experiencing high latency, run the following Linux curl command on each server:

```
curl -w "Connect time: %{time_connect} Time to first byte:
%{time_starttransfer} Total time: %{time_total} \n" -o /dev/null https://
www.example.com/yourobject
```

> **Note**
>
> If you run Windows on your servers, you can search for and download curl for Windows to run a similar command.

As you measure and evaluate the latency of an application that runs on your server, keep in mind the following:

- Latency values are relative to each application. However, a time to first byte in milliseconds rather than seconds or more, is reasonable.
- If you measure the application latency under normal load and it's fine, be aware that viewers might still experience timeouts under high load. When there is high demand, servers can have delayed responses or not respond at all. To help prevent high-load latency issues, check your server's resources such as CPU, memory, and disk reads and writes to make sure that your servers have the capacity to scale for high load.

  You can run the following Linux command to check the memory that is used by Apache processes:

  ```
  watch -n 1 "echo -n 'Apache Processes: ' && ps -C apache2 --no-headers | wc -l && free -m"
  ```
- High CPU utilization on the server can significantly reduce an application's performance. If you use an Amazon EC2 instance for your backend server, review the CloudWatch metrics for the server to check the CPU utilization. For more information, see the Amazon CloudWatch User Guide. Or if you're using your own server, refer to the server Help documentation for instructions on how to check CPU utilization.
- Check for other potential issues under high loads, such as database queries that run slowly when there's a high volume of requests.

Add resources, and tune servers and databases

After you evaluate the responsiveness of your applications and servers, make sure that you have sufficient resources in place for typical traffic and high load situations:

- If you have your own server, make sure it has enough CPU, memory, and disk space to handle viewer requests, based on your evaluation.

- If you use an Amazon EC2 instance as your backend server, make sure that the instance type has the appropriate resources to fulfill incoming requests. For more information, see Instance types in the Amazon EC2 User Guide.

In addition, consider the following tuning steps to help avoid timeouts:

- If the Time to First Byte value that is returned by the curl command seems high, take steps to improve the performance of your application. Improving application responsiveness will in turn help reduce timeout errors.
- Tune database queries to make sure that they can handle high request volumes without slow performance.
- Set up keep-alive (persistent) connections on your backend server. This option helps to avoid latencies that occur when connections must be re-established for subsequent requests or users.
- If you use ELB as your origin, learn how you can reduce latency by reviewing the suggestions in the following Knowledge Center article: How do I troubleshoot high latency on my ELB Classic Load Balancer?

If needed, adjust the CloudFront timeout value

If you have evaluated and addressed slow application performance, origin server capacity, and other issues, but viewers are still experiencing HTTP 504 errors, then you should consider changing the time that is specified in your distribution for origin response timeout. To learn more, see the section called “Response timeout (custom origins only)”.

Load testing CloudFront

Traditional load testing methods don't work well with CloudFront because CloudFront uses DNS to balance loads across geographically dispersed edge locations and within each edge location. When a client requests content from CloudFront, the client receives a DNS response that includes a set of IP addresses. If you test by sending requests to just one of the IP addresses that DNS returns, you're testing only a small subset of the resources in one CloudFront edge location, which doesn't accurately represent actual traffic patterns. Depending on the volume of data requested, testing in this way may overload and degrade the performance of that small subset of CloudFront servers.

CloudFront is designed to scale for viewers that have different client IP addresses and different DNS resolvers across multiple geographic regions. To perform load testing that accurately assesses CloudFront performance, we recommend that you do all of the following:

- Send client requests from multiple geographic regions.
- Configure your test so each client makes an independent DNS request; each client will then receive a different set of IP addresses from DNS.
- For each client that is making requests, spread your client requests across the set of IP addresses that are returned by DNS, which ensures that the load is distributed across multiple servers in a CloudFront edge location.

Note the following restrictions for load testing CloudFront:

- Load testing is not allowed on cache behaviors that have Lambda@Edge viewer request or viewer response triggers.
- Load testing is not allowed on origins that have Origin Shield enabled.

Request and response behavior

The following sections explain how CloudFront processes viewer requests and forwards the requests to your Amazon S3 or custom origin, and how CloudFront processes responses from your origin, including how CloudFront processes and caches 4xx and 5xx HTTP status codes.

Topics

- Request and response behavior for Amazon S3 origins
- Request and response behavior for custom origins
- Request and response behavior for origin groups
- Adding custom headers to origin requests
- How CloudFront processes partial requests for an object (range GETs)
- How CloudFront processes HTTP 3xx status codes from your origin
- How CloudFront processes and caches HTTP 4xx and 5xx status codes from your origin

Request and response behavior for Amazon S3 origins

Topics

- How CloudFront processes HTTP and HTTPS requests
- How CloudFront processes and forwards requests to your Amazon S3 origin
- How CloudFront processes responses from your Amazon S3 origin

How CloudFront processes HTTP and HTTPS requests

For Amazon S3 origins, CloudFront accepts requests in both HTTP and HTTPS protocols for objects in a CloudFront distribution by default. CloudFront then forwards the requests to your Amazon S3 bucket using the same protocol in which the requests were made.

For custom origins, when you create your distribution, you can specify how CloudFront accesses your origin: HTTP only, or matching the protocol that is used by the viewer. For more information about how CloudFront handles HTTP and HTTPS requests for custom origins, see Protocols.

For information about how to restrict your distribution so that end users can only access objects using HTTPS, see Using HTTPS with CloudFront.

> **Note**
>
> The charge for HTTPS requests is higher than the charge for HTTP requests. For more information about billing rates, go to the CloudFront pricing plan.

How CloudFront processes and forwards requests to your Amazon S3 origin

This topic contains information about how CloudFront processes viewer requests and forwards the requests to your Amazon S3 origin.

Topics

- Caching duration and minimum TTL
- Client IP addresses
- Conditional GETs
- Cookies
- Cross-origin resource sharing (CORS)
- GET requests that include a body
- HTTP methods
- HTTP request headers that CloudFront removes or updates
- Maximum length of a request and maximum length of a URL
- OCSP stapling
- Protocols
- Query strings
- Origin connection timeout and attempts
- Origin response timeout
- Simultaneous requests for the same object (request collapsing)

Caching duration and minimum TTL

To control how long your objects stay in a CloudFront cache before CloudFront forwards another request to your origin, you can:

- Configure your origin to add a `Cache-Control` or an `Expires` header field to each object.
- Specify a value for Minimum TTL in CloudFront cache behaviors.
- Use the default value of 24 hours.

For more information, see Managing how long content stays in the cache (expiration).

Client IP addresses

If a viewer sends a request to CloudFront and does not include an `X-Forwarded-For` request header, CloudFront gets the IP address of the viewer from the TCP connection, adds an `X-Forwarded-For` header that includes the IP address, and forwards the request to the origin. For example, if CloudFront gets the IP address `192.0.2.2` from the TCP connection, it forwards the following header to the origin:

```
X-Forwarded-For: 192.0.2.2
```

If a viewer sends a request to CloudFront and includes an `X-Forwarded-For` request header, CloudFront gets the IP address of the viewer from the TCP connection, appends it to the end of the `X-Forwarded-For` header, and forwards the request to the origin. For example, if the viewer request includes `X-Forwarded-For: 192.0.2.4,192.0.2.3` and CloudFront gets the IP address `192.0.2.2` from the TCP connection, it forwards the following header to the origin:

```
X-Forwarded-For: 192.0.2.4,192.0.2.3,192.0.2.2
```

> **Note**
>
> The `X-Forwarded-For` header contains IPv4 addresses (such as 192.0.2.44) and IPv6 addresses (such as 2001:0db8:85a3::8a2e:0370:7334).

Conditional GETs

When CloudFront receives a request for an object that has expired from an edge cache, it forwards the request to the Amazon S3 origin either to get the latest version of the object or to get confirmation from Amazon S3 that the CloudFront edge cache already has the latest version. When Amazon S3 originally sent the object to CloudFront, it included an `ETag` value and a `LastModified` value in the response. In the new request that CloudFront forwards to Amazon S3, CloudFront adds one or both of the following:

- An `If-Match` or `If-None-Match` header that contains the `ETag` value for the expired version of the object.
- An `If-Modified-Since` header that contains the `LastModified` value for the expired version of the object.

Amazon S3 uses this information to determine whether the object has been updated and, therefore, whether to return the entire object to CloudFront or to return only an HTTP 304 status code (not modified).

Cookies

Amazon S3 doesn't process cookies. If you configure a cache behavior to forward cookies to an Amazon S3 origin, CloudFront forwards the cookies, but Amazon S3 ignores them. All future requests for the same object, regardless if you vary the cookie, are served from the existing object in the cache.

Cross-origin resource sharing (CORS)

If you want CloudFront to respect Amazon S3 cross-origin resource sharing settings, configure CloudFront to forward selected headers to Amazon S3. For more information, see Caching content based on request headers.

GET requests that include a body

If a viewer `GET` request includes a body, CloudFront returns an HTTP status code 403 (Forbidden) to the viewer.

HTTP methods

If you configure CloudFront to process all of the HTTP methods that it supports, CloudFront accepts the following requests from viewers and forwards them to your Amazon S3 origin:

- `DELETE`
- `GET`
- `HEAD`
- `OPTIONS`
- `PATCH`
- `POST`

- PUT

CloudFront always caches responses to `GET` and `HEAD` requests. You can also configure CloudFront to cache responses to `OPTIONS` requests. CloudFront does not cache responses to requests that use the other methods.

If you want to use multi-part uploads to add objects to an Amazon S3 bucket, you must add a CloudFront origin access control (OAC) to your distribution and give the OAC the needed permissions. For more information, see the section called “Restricting access to an Amazon Simple Storage Service origin”.

> **Important**
>
> If you configure CloudFront to accept and forward to Amazon S3 all of the HTTP methods that CloudFront supports, you must create a CloudFront origin access control (OAC) to restrict access to your Amazon S3 content and give the OAC the required permissions. For example, if you configure CloudFront to accept and forward these methods because you want to use `PUT`, you must configure Amazon S3 bucket policies to handle `DELETE` requests appropriately so viewers can't delete resources that you don't want them to. For more information, see the section called “Restricting access to an Amazon Simple Storage Service origin”.

For information about the operations supported by Amazon S3, see the Amazon S3 documentation.

HTTP request headers that CloudFront removes or updates

CloudFront removes or updates some headers before forwarding requests to your Amazon S3 origin. For most headers this behavior is the same as for custom origins. For a full list of HTTP request headers and how CloudFront processes them, see HTTP request headers and CloudFront behavior (custom and Amazon S3 origins).

Maximum length of a request and maximum length of a URL

The maximum length of a request, including the path, the query string (if any), and headers, is 20,480 bytes.

CloudFront constructs a URL from the request. The maximum length of this URL is 8192 bytes.

If a request or a URL exceeds these maximums, CloudFront returns HTTP status code 413, Request Entity Too Large, to the viewer, and then terminates the TCP connection to the viewer.

OCSP stapling

When a viewer submits an HTTPS request for an object, either CloudFront or the viewer must confirm with the certificate authority (CA) that the SSL certificate for the domain has not been revoked. OCSP stapling speeds up certificate validation by allowing CloudFront to validate the certificate and to cache the response from the CA, so the client doesn't need to validate the certificate directly with the CA.

The performance improvement of OCSP stapling is more pronounced when CloudFront receives a lot of HTTPS requests for objects in the same domain. Each server in a CloudFront edge location must submit a separate validation request. When CloudFront receives a lot of HTTPS requests for the same domain, every server in the edge location soon has a response from the CA that it can "staple" to a packet in the SSL handshake; when the viewer is satisfied that the certificate is valid, CloudFront can serve the requested object. If your distribution doesn't get much traffic in a CloudFront edge location, new requests are more likely to be directed to a server that hasn't validated the certificate with the CA yet. In that case, the viewer separately performs the validation step and the CloudFront server serves the object. That CloudFront server also submits a validation request to the CA, so the next time it receives a request that includes the same domain name, it has a validation response from the CA.

Protocols

CloudFront forwards HTTP or HTTPS requests to the origin server based on the protocol of the viewer request, either HTTP or HTTPS.

> **Important**
>
> If your Amazon S3 bucket is configured as a website endpoint, you cannot configure CloudFront to use HTTPS to communicate with your origin because Amazon S3 doesn't support HTTPS connections in that configuration.

Query strings

You can configure whether CloudFront forwards query string parameters to your Amazon S3 origin. For more information, see Caching content based on query string parameters.

Origin connection timeout and attempts

Origin connection timeout is the number of seconds that CloudFront waits when trying to establish a connection to the origin.

Origin connection attempts is the number of times that CloudFront attempts to connect to the origin.

Together, these settings determine how long CloudFront tries to connect to the origin before failing over to the secondary origin (in the case of an origin group) or returning an error response to the viewer. By default, CloudFront waits as long as 30 seconds (3 attempts of 10 seconds each) before attempting to connect to the secondary origin or returning an error response. You can reduce this time by specifying a shorter connection timeout, fewer attempts, or both.

For more information, see Controlling origin timeouts and attempts.

Origin response timeout

The *origin response timeout*, also known as the *origin read timeout* or *origin request timeout*, applies to both of the following:

- The amount of time, in seconds, that CloudFront waits for a response after forwarding a request to the origin.
- The amount of time, in seconds, that CloudFront waits after receiving a packet of a response from the origin and before receiving the next packet.

CloudFront behavior depends on the HTTP method of the viewer request:

- `GET` and `HEAD` requests – If the origin doesn't respond within 30 seconds or stops responding for 30 seconds, CloudFront drops the connection. If the specified number of origin connection attempts is more than 1, CloudFront tries again to get a complete response. CloudFront tries up to 3 times, as determined by the value of the *origin connection attempts* setting. If the origin doesn't respond during the final attempt, CloudFront doesn't try again until it receives another request for content on the same origin.
- `DELETE`, `OPTIONS`, `PATCH`, `PUT`, and `POST` requests – If the origin doesn't respond within 30 seconds, CloudFront drops the connection and doesn't try again to contact the origin. The client can resubmit the request if necessary.

You can't change the response timeout for an Amazon S3 origin (an S3 bucket that is *not* configured with static website hosting).

Simultaneous requests for the same object (request collapsing)

When a CloudFront edge location receives a request for an object and the object isn't in the cache or the cached object is expired, CloudFront immediately sends the request to the origin. However, if there are simultaneous requests for the same object—that is, if additional requests for the same object (with the same cache key) arrive at the edge location before CloudFront receives the response to the first request—CloudFront pauses before forwarding the additional requests to the origin. This brief pause helps to reduce the load on the origin. CloudFront sends the response from the original request to all the requests that it received while it was paused. This is called *request collapsing*. In CloudFront logs, the first request is identified as a `Miss` in the `x-edge-result-type` field, and the collapsed requests are identified as a `Hit`. For more information about CloudFront logs, see the section called "CloudFront and edge function logging".

CloudFront only collapses requests that share a *cache key*. If the additional requests do not share the same cache key because, for example, you configured CloudFront to cache based on request headers or cookies or query strings, CloudFront forwards all the requests with a unique cache key to your origin.

If you would like to prevent all request collapsing, you can use the managed cache policy `CachingDisabled`, which also prevents caching. For more information, see Using the managed cache policies.

If you would like to prevent request collapsing for specific objects, you can set the minimum TTL for the cache behavior to 0 *and* configure the origin to send `Cache-Control: private`, `Cache-Control: no-store`, `Cache-Control: no-cache`, `Cache-Control: max-age=0`, or `Cache-Control: s-maxage=0`. These configurations will increase the load on your origin and introduce additional latency for the simultaneous requests that are paused while CloudFront waits for the response to the first request.

How CloudFront processes responses from your Amazon S3 origin

This topic contains information about how CloudFront processes responses from your Amazon S3 origin.

Topics

- Canceled requests

- HTTP response headers that CloudFront removes or updates
- Maximum cacheable file size
- Redirects

Canceled requests

If an object is not in the edge cache, and if a viewer terminates a session (for example, closes a browser) after CloudFront gets the object from your origin but before it can deliver the requested object, CloudFront does not cache the object in the edge location.

HTTP response headers that CloudFront removes or updates

CloudFront removes or updates the following header fields before forwarding the response from your Amazon S3 origin to the viewer:

- `X-Amz-Id-2`
- `X-Amz-Request-Id`
- `Set-Cookie` – If you configure CloudFront to forward cookies, it will forward the `Set-Cookie` header field to clients. For more information, see Caching content based on cookies.
- `Trailer`
- `Transfer-Encoding` – If your Amazon S3 origin returns this header field, CloudFront sets the value to `chunked` before returning the response to the viewer.
- `Upgrade`
- `Via` – CloudFront sets the value to the following in the response to the viewer:

  `Via: http-version alphanumeric-string.cloudfront.net (CloudFront)`

  For example, the value is something like the following:

  `Via: 1.1 1026589cc7887e7a0dc7827b4example.cloudfront.net (CloudFront)`

Maximum cacheable file size

The maximum size of a response body that CloudFront saves in its cache is 50 GB. This includes chunked transfer responses that don't specify the `Content-Length` header value.

You can use CloudFront to cache an object that is larger than this size by using range requests to request the objects in parts that are each 50 GB or smaller. CloudFront caches these parts because each of them is 50 GB or smaller. After the viewer retrieves all the parts of the object, it can reconstruct the original, larger object. For more information, see Use range requests to cache large objects.

Redirects

You can configure an Amazon S3 bucket to redirect all requests to another host name; this can be another Amazon S3 bucket or an HTTP server. If you configure a bucket to redirect all requests and if the bucket is the origin for a CloudFront distribution, we recommend that you configure the bucket to redirect all requests to a CloudFront distribution using either the domain name for the distribution (for example, d111111abcdef8.cloudfront.net) or an alternate domain name (a CNAME) that is associated with a distribution (for example, example.com). Otherwise, viewer requests bypass CloudFront, and the objects are served directly from the new origin.

> **Note**
>
> If you redirect requests to an alternate domain name, you must also update the DNS service for your domain by adding a CNAME record. For more information, see Using custom URLs by adding alternate domain names (CNAMEs).

Here's what happens when you configure a bucket to redirect all requests:

1. A viewer (for example, a browser) requests an object from CloudFront.
2. CloudFront forwards the request to the Amazon S3 bucket that is the origin for your distribution.
3. Amazon S3 returns an HTTP status code 301 (Moved Permanently) as well as the new location.
4. CloudFront caches the redirect status code and the new location, and returns the values to the viewer. CloudFront does not follow the redirect to get the object from the new location.
5. The viewer sends another request for the object, but this time the viewer specifies the new location that it got from CloudFront:

    - If the Amazon S3 bucket is redirecting all requests to a CloudFront distribution, using either the domain name for the distribution or an alternate domain name, CloudFront requests the object from the Amazon S3 bucket or the HTTP server in the new location. When the

new location returns the object, CloudFront returns it to the viewer and caches it in an edge location.

- If the Amazon S3 bucket is redirecting requests to another location, the second request bypasses CloudFront. The Amazon S3 bucket or the HTTP server in the new location returns the object directly to the viewer, so the object is never cached in a CloudFront edge cache.

Request and response behavior for custom origins

Topics

- How CloudFront processes and forwards requests to your custom origin
- How CloudFront processes responses from your custom origin

How CloudFront processes and forwards requests to your custom origin

This topic contains information about how CloudFront processes viewer requests and forwards the requests to your custom origin.

Topics

- Authentication
- Caching duration and minimum TTL
- Client IP addresses
- Client-side SSL authentication
- Compression
- Conditional requests
- Cookies
- Cross-origin resource sharing (CORS)
- Encryption
- GET requests that include a body
- HTTP methods
- HTTP request headers and CloudFront behavior (custom and Amazon S3 origins)
- HTTP version
- Maximum length of a request and maximum length of a URL
- OCSP stapling

- Persistent connections
- Protocols
- Query strings
- Origin connection timeout and attempts
- Origin response timeout
- Simultaneous requests for the same object (request collapsing)
- User-Agent header

Authentication

If you forward the `Authorization` header to your origin, you can then configure your origin server to request client authentication for the following types of requests:

- `DELETE`
- `GET`
- `HEAD`
- `PATCH`
- `PUT`
- `POST`

For `OPTIONS` requests, client authentication can *only* be configured if you use the following CloudFront settings:

- CloudFront is configured to forward the `Authorization` header to your origin
- CloudFront is configured to *not* cache the response to `OPTIONS` requests

For more information, see Configuring CloudFront to forward the `Authorization` header.

You can use HTTP or HTTPS to forward requests to your origin server. For more information, see Using HTTPS with CloudFront.

Caching duration and minimum TTL

To control how long your objects stay in a CloudFront cache before CloudFront forwards another request to your origin, you can:

- Configure your origin to add a `Cache-Control` or an `Expires` header field to each object.
- Specify a value for Minimum TTL in CloudFront cache behaviors.
- Use the default value of 24 hours.

For more information, see Managing how long content stays in the cache (expiration).

Client IP addresses

If a viewer sends a request to CloudFront and does not include an `X-Forwarded-For` request header, CloudFront gets the IP address of the viewer from the TCP connection, adds an `X-Forwarded-For` header that includes the IP address, and forwards the request to the origin. For example, if CloudFront gets the IP address `192.0.2.2` from the TCP connection, it forwards the following header to the origin:

```
X-Forwarded-For: 192.0.2.2
```

If a viewer sends a request to CloudFront and includes an `X-Forwarded-For` request header, CloudFront gets the IP address of the viewer from the TCP connection, appends it to the end of the `X-Forwarded-For` header, and forwards the request to the origin. For example, if the viewer request includes `X-Forwarded-For: 192.0.2.4,192.0.2.3` and CloudFront gets the IP address `192.0.2.2` from the TCP connection, it forwards the following header to the origin:

```
X-Forwarded-For: 192.0.2.4,192.0.2.3,192.0.2.2
```

Some applications, such as load balancers (including Elastic Load Balancing), web application firewalls, reverse proxies, intrusion prevention systems, and API Gateway, append the IP address of the CloudFront edge server that forwarded the request onto the end of the `X-Forwarded-For` header. For example, if CloudFront includes `X-Forwarded-For: 192.0.2.2` in a request that it forwards to ELB and if the IP address of the CloudFront edge server is 192.0.2.199, the request that your EC2 instance receives contains the following header:

```
X-Forwarded-For: 192.0.2.2,192.0.2.199
```

> **Note**
>
> The `X-Forwarded-For` header contains IPv4 addresses (such as 192.0.2.44) and IPv6 addresses (such as 2001:0db8:85a3::8a2e:0370:7334).

Also note that the `X-Forwarded-For` header may be modified by every node on the path to the current server (CloudFront). For more information, see section 8.1 in RFC 7239. You can also modify the header using CloudFront edge compute functions.

Client-side SSL authentication

CloudFront does not support client authentication with client-side SSL certificates. If an origin requests a client-side certificate, CloudFront drops the request.

Compression

For more information, see Serving compressed files.

Conditional requests

When CloudFront receives a request for an object that has expired from an edge cache, it forwards the request to the origin either to get the latest version of the object or to get confirmation from the origin that the CloudFront edge cache already has the latest version. Typically, when the origin last sent the object to CloudFront, it included an `ETag` value, a `LastModified` value, or both values in the response. In the new request that CloudFront forwards to the origin, CloudFront adds one or both of the following:

- An `If-Match` or `If-None-Match` header that contains the `ETag` value for the expired version of the object.
- An `If-Modified-Since` header that contains the `LastModified` value for the expired version of the object.

The origin uses this information to determine whether the object has been updated and, therefore, whether to return the entire object to CloudFront or to return only an HTTP 304 status code (not modified).

Cookies

You can configure CloudFront to forward cookies to your origin. For more information, see Caching content based on cookies.

Cross-origin resource sharing (CORS)

If you want CloudFront to respect cross-origin resource sharing settings, configure CloudFront to forward the `Origin` header to your origin. For more information, see Caching content based on request headers.

Encryption

You can require viewers to use HTTPS to send requests to CloudFront and require CloudFront to forward requests to your custom origin by using the protocol that is used by the viewer. For more information, see the following distribution settings:

- Viewer protocol policy
- Protocol (custom origins only)

CloudFront forwards HTTPS requests to the origin server using the SSLv3, TLSv1.0, TLSv1.1, and TLSv1.2 protocols. For custom origins, you can choose the SSL protocols that you want CloudFront to use when communicating with your origin:

- If you're using the CloudFront console, choose protocols by using the **Origin SSL Protocols** check boxes. For more information, see Creating a distribution.
- If you're using the CloudFront API, specify protocols by using the `OriginSslProtocols` element. For more information, see OriginSslProtocols and DistributionConfig in the *Amazon CloudFront API Reference*.

If the origin is an Amazon S3 bucket, CloudFront always uses TLSv1.2.

> **Important**
>
> Other versions of SSL and TLS are not supported.

For more information about using HTTPS with CloudFront, see Using HTTPS with CloudFront. For lists of the ciphers that CloudFront supports for HTTPS communication between viewers and CloudFront, and between CloudFront and your origin, see Supported protocols and ciphers between viewers and CloudFront.

GET requests that include a body

If a viewer GET request includes a body, CloudFront returns an HTTP status code 403 (Forbidden) to the viewer.

HTTP methods

If you configure CloudFront to process all of the HTTP methods that it supports, CloudFront accepts the following requests from viewers and forwards them to your custom origin:

- DELETE
- GET
- HEAD
- OPTIONS
- PATCH
- POST
- PUT

CloudFront always caches responses to GET and HEAD requests. You can also configure CloudFront to cache responses to OPTIONS requests. CloudFront does not cache responses to requests that use the other methods.

For information about configuring whether your custom origin processes these methods, see the documentation for your origin.

Important

If you configure CloudFront to accept and forward to your origin all of the HTTP methods that CloudFront supports, configure your origin server to handle all methods. For example, if you configure CloudFront to accept and forward these methods because you want to use POST, you must configure your origin server to handle DELETE requests appropriately so viewers can't delete resources that you don't want them to. For more information, see the documentation for your HTTP server.

HTTP request headers and CloudFront behavior (custom and Amazon S3 origins)

The following table lists HTTP request headers that you can forward to both custom and Amazon S3 origins (with the exceptions that are noted). For each header, the table includes information about the following:

- CloudFront behavior if you don't configure CloudFront to forward the header to your origin, which causes CloudFront to cache your objects based on header values.
- Whether you can configure CloudFront to cache objects based on header values for that header.

  You can configure CloudFront to cache objects based on values in the `Date` and `User-Agent` headers, but we don't recommend it. These headers have many possible values, and caching based on their values would cause CloudFront to forward significantly more requests to your origin.

For more information about caching based on header values, see Caching content based on request headers.

Header	Behavior if you don't configure CloudFront to cache based on header values	Caching based on header values is supported
Other-defined headers	**Legacy cache settings** – CloudFront forwards the headers to your origin.	Yes
`Accept`	CloudFront removes the header.	Yes
`Accept-Charset`	CloudFront removes the header.	Yes
`Accept-Encoding`	If the value contains `gzip` or `br`, CloudFront forwards a normalized `Accept-Encoding` header to your origin.	Yes

Header	Behavior if you don't configure CloudFront to cache based on header values	Caching based on header values is supported
	For more information, see Compression support and Serving compressed files.	
`Accept-Language`	CloudFront removes the header.	Yes
`Authorization`	• `GET` and `HEAD` requests – CloudFront removes the `Authorization` header field before forwarding the request to your origin. • `OPTIONS` requests – CloudFront removes the `Authorization` header field before forwarding the request to your origin if you configure CloudFront to cache responses to `OPTIONS` requests. CloudFront forwards the `Authorization` header field to your origin if you do not configure CloudFront to cache responses to OPTIONS requests. • `DELETE`, `PATCH`, `POST`, and `PUT` requests – CloudFront does not remove the header field before forwarding the request to your origin.	Yes
`Cache-Control`	CloudFront forwards the header to your origin.	No

Header	Behavior if you don't configure CloudFront to cache based on header values	Caching based on header values is supported
`CloudFront-Forwarded-Proto`	CloudFront does not add the header before forwarding the request to your origin. For more information, see Configuring caching based on the protocol of the request.	Yes
`CloudFront-Is-Desktop-Viewer`	CloudFront does not add the header before forwarding the request to your origin. For more information, see Configuring caching based on the device type.	Yes
`CloudFront-Is-Mobile-Viewer`	CloudFront does not add the header before forwarding the request to your origin. For more information, see Configuring caching based on the device type.	Yes
`CloudFront-Is-Tablet-Viewer`	CloudFront does not add the header before forwarding the request to your origin. For more information, see Configuring caching based on the device type.	Yes
`CloudFront-Viewer-Country`	CloudFront does not add the header before forwarding the request to your origin.	Yes

Header	Behavior if you don't configure CloudFront to cache based on header values	Caching based on header values is supported
`Connection`	CloudFront replaces this header with `Connection: Keep-Alive` before forwarding the request to your origin.	No
`Content-Length`	CloudFront forwards the header to your origin.	No
`Content-MD5`	CloudFront forwards the header to your origin.	Yes
`Content-Type`	CloudFront forwards the header to your origin.	Yes
`Cookie`	If you configure CloudFront to forward cookies, it will forward the `Cookie` header field to your origin. If you don't, CloudFront removes the `Cookie` header field. For more information, see Caching content based on cookies.	No
`Date`	CloudFront forwards the header to your origin.	Yes, but not recommended
`Expect`	CloudFront removes the header.	Yes
`From`	CloudFront forwards the header to your origin.	Yes

Header	Behavior if you don't configure CloudFront to cache based on header values	Caching based on header values is supported
`Host`	CloudFront sets the value to the domain name of the origin that is associated with the requested object. You can't cache based on the Host header for Amazon S3 or MediaStore origins.	Yes (custom) No (S3 and MediaStore)
`If-Match`	CloudFront forwards the header to your origin.	Yes
`If-Modified-Since`	CloudFront forwards the header to your origin.	Yes
`If-None-Match`	CloudFront forwards the header to your origin.	Yes
`If-Range`	CloudFront forwards the header to your origin.	Yes
`If-Unmodified-Since`	CloudFront forwards the header to your origin.	Yes
`Max-Forwards`	CloudFront forwards the header to your origin.	No
`Origin`	CloudFront forwards the header to your origin.	Yes
`Pragma`	CloudFront forwards the header to your origin.	No
`Proxy-Authenticate`	CloudFront removes the header.	No

Header	Behavior if you don't configure CloudFront to cache based on header values	Caching based on header values is supported
`Proxy-Authorization`	CloudFront removes the header.	No
`Proxy-Connection`	CloudFront removes the header.	No
`Range`	CloudFront forwards the header to your origin. For more information, see How CloudFront processes partial requests for an object (range GETs).	Yes, by default
`Referer`	CloudFront removes the header.	Yes
`Request-Range`	CloudFront forwards the header to your origin.	No
`TE`	CloudFront removes the header.	No
`Trailer`	CloudFront removes the header.	No
`Transfer-Encoding`	CloudFront forwards the header to your origin.	No
`Upgrade`	CloudFront removes the header, unless you've established a WebSocket connection.	No (except for WebSocket connections)

Header	Behavior if you don't configure CloudFront to cache based on header values	Caching based on header values is supported
`User-Agent`	CloudFront replaces the value of this header field with `Amazon CloudFront` . If you want CloudFront to cache your content based on the device the user is using, see Configuring caching based on the device type.	Yes, but not recommended
`Via`	CloudFront forwards the header to your origin.	Yes
`Warning`	CloudFront forwards the header to your origin.	Yes
`X-Amz-Cf-Id`	CloudFront adds the header to the viewer request before forwarding the request to your origin. The header value contains an encrypted string that uniquely identifies the request.	No
`X-Edge-*`	CloudFront removes all `X-Edge-*` headers.	No
`X-Forwarded-For`	CloudFront forwards the header to your origin. For more information, see Client IP addresses.	Yes
`X-Forwarded-Proto`	CloudFront removes the header.	No
`X-HTTP-Method-Override`	CloudFront removes the header.	Yes
`X-Real-IP`	CloudFront removes the header.	No

HTTP version

CloudFront forwards requests to your custom origin using HTTP/1.1.

Maximum length of a request and maximum length of a URL

The maximum length of a request, including the path, the query string (if any), and headers, is 20,480 bytes.

CloudFront constructs a URL from the request. The maximum length of this URL is 8192 bytes.

If a request or a URL exceeds these maximums, CloudFront returns HTTP status code 413, Request Entity Too Large, to the viewer, and then terminates the TCP connection to the viewer.

OCSP stapling

When a viewer submits an HTTPS request for an object, either CloudFront or the viewer must confirm with the certificate authority (CA) that the SSL certificate for the domain has not been revoked. OCSP stapling speeds up certificate validation by allowing CloudFront to validate the certificate and to cache the response from the CA, so the client doesn't need to validate the certificate directly with the CA.

The performance improvement of OCSP stapling is more pronounced when CloudFront receives numerous HTTPS requests for objects in the same domain. Each server in a CloudFront edge location must submit a separate validation request. When CloudFront receives a lot of HTTPS requests for the same domain, every server in the edge location soon has a response from the CA that it can "staple" to a packet in the SSL handshake; when the viewer is satisfied that the certificate is valid, CloudFront can serve the requested object. If your distribution doesn't get much traffic in a CloudFront edge location, new requests are more likely to be directed to a server that hasn't validated the certificate with the CA yet. In that case, the viewer separately performs the validation step and the CloudFront server serves the object. That CloudFront server also submits a validation request to the CA, so the next time it receives a request that includes the same domain name, it has a validation response from the CA.

Persistent connections

When CloudFront gets a response from your origin, it tries to maintain the connection for several seconds in case another request arrives during that period. Maintaining a persistent connection saves the time required to re-establish the TCP connection and perform another TLS handshake for subsequent requests.

For more information, including how to configure the duration of persistent connections, see Keep-alive timeout (custom origins only) in the section Values that you specify when you create or update a distribution.

Protocols

CloudFront forwards HTTP or HTTPS requests to the origin server based on the following:

- The protocol of the request that the viewer sends to CloudFront, either HTTP or HTTPS.
- The value of the **Origin Protocol Policy** field in the CloudFront console or, if you're using the CloudFront API, the `OriginProtocolPolicy` element in the `DistributionConfig` complex type. In the CloudFront console, the options are **HTTP Only**, **HTTPS Only**, and **Match Viewer**.

If you specify **HTTP Only** or **HTTPS Only**, CloudFront forwards requests to the origin server using the specified protocol, regardless of the protocol in the viewer request.

If you specify **Match Viewer**, CloudFront forwards requests to the origin server using the protocol in the viewer request. Note that CloudFront caches the object only once even if viewers make requests using both HTTP and HTTPS protocols.

> **Important**
>
> If CloudFront forwards a request to the origin using the HTTPS protocol, and if the origin server returns an invalid certificate or a self-signed certificate, CloudFront drops the TCP connection.

For information about how to update a distribution using the CloudFront console, see Updating a distribution. For information about how to update a distribution using the CloudFront API, go to UpdateDistribution in the *Amazon CloudFront API Reference*.

Query strings

You can configure whether CloudFront forwards query string parameters to your origin. For more information, see Caching content based on query string parameters.

Origin connection timeout and attempts

Origin connection timeout is the number of seconds that CloudFront waits when trying to establish a connection to the origin.

Origin connection attempts is the number of times that CloudFront attempts to connect to the origin.

Together, these settings determine how long CloudFront tries to connect to the origin before failing over to the secondary origin (in the case of an origin group) or returning an error response to the viewer. By default, CloudFront waits as long as 30 seconds (3 attempts of 10 seconds each) before attempting to connect to the secondary origin or returning an error response. You can reduce this time by specifying a shorter connection timeout, fewer attempts, or both.

For more information, see Controlling origin timeouts and attempts.

Origin response timeout

The *origin response timeout*, also known as the *origin read timeout* or *origin request timeout*, applies to both of the following:

- The amount of time, in seconds, that CloudFront waits for a response after forwarding a request to the origin.
- The amount of time, in seconds, that CloudFront waits after receiving a packet of a response from the origin and before receiving the next packet.

CloudFront behavior depends on the HTTP method of the viewer request:

- `GET` and `HEAD` requests – If the origin doesn't respond or stops responding within the duration of the response timeout, CloudFront drops the connection. If the specified number of origin connection attempts is more than 1, CloudFront tries again to get a complete response. CloudFront tries up to 3 times, as determined by the value of the *origin connection attempts* setting. If the origin doesn't respond during the final attempt, CloudFront doesn't try again until it receives another request for content on the same origin.
- `DELETE`, `OPTIONS`, `PATCH`, `PUT`, and `POST` requests – If the origin doesn't respond within 30 seconds, CloudFront drops the connection and doesn't try again to contact the origin. The client can resubmit the request if necessary.

For more information, including how to configure the origin response timeout, see Response timeout (custom origins only).

Simultaneous requests for the same object (request collapsing)

When a CloudFront edge location receives a request for an object and the object isn't in the cache or the cached object is expired, CloudFront immediately sends the request to the origin. However, if there are simultaneous requests for the same object—that is, if additional requests for the same object (with the same cache key) arrive at the edge location before CloudFront receives the response to the first request—CloudFront pauses before forwarding the additional requests to the origin. This brief pause helps to reduce the load on the origin. CloudFront sends the response from the original request to all the requests that it received while it was paused. This is called *request collapsing*. In CloudFront logs, the first request is identified as a `Miss` in the `x-edge-result-type` field, and the collapsed requests are identified as a `Hit`. For more information about CloudFront logs, see the section called “CloudFront and edge function logging”.

CloudFront only collapses requests that share a *cache key*. If the additional requests do not share the same cache key because, for example, you configured CloudFront to cache based on request headers or cookies or query strings, CloudFront forwards all the requests with a unique cache key to your origin.

If you would like to prevent all request collapsing, you can use the managed cache policy `CachingDisabled`, which also prevents caching. For more information, see Using the managed cache policies.

If you would like to prevent request collapsing for specific objects, you can set the minimum TTL for the cache behavior to 0 *and* configure the origin to send `Cache-Control: private`, `Cache-Control: no-store`, `Cache-Control: no-cache`, `Cache-Control: max-age=0`, or `Cache-Control: s-maxage=0`. These configurations will increase the load on your origin and introduce additional latency for the simultaneous requests that are paused while CloudFront waits for the response to the first request.

`User-Agent` header

If you want CloudFront to cache different versions of your objects based on the device that a user is using to view your content, we recommend that you configure CloudFront to forward one or more of the following headers to your custom origin:

- `CloudFront-Is-Desktop-Viewer`
- `CloudFront-Is-Mobile-Viewer`
- `CloudFront-Is-SmartTV-Viewer`
- `CloudFront-Is-Tablet-Viewer`

Based on the value of the `User-Agent` header, CloudFront sets the value of these headers to `true` or `false` before forwarding the request to your origin. If a device falls into more than one category, more than one value might be `true`. For example, for some tablet devices, CloudFront might set both `CloudFront-Is-Mobile-Viewer` and `CloudFront-Is-Tablet-Viewer` to `true`. For more information about configuring CloudFront to cache based on request headers, see Caching content based on request headers.

You can configure CloudFront to cache objects based on values in the `User-Agent` header, but we don't recommend it. The `User-Agent` header has many possible values, and caching based on those values would cause CloudFront to forward significantly more requests to your origin.

If you do not configure CloudFront to cache objects based on values in the `User-Agent` header, CloudFront adds a `User-Agent` header with the following value before it forwards a request to your origin:

```
User-Agent = Amazon CloudFront
```

CloudFront adds this header regardless of whether the request from the viewer includes a `User-Agent` header. If the request from the viewer includes a `User-Agent` header, CloudFront removes it.

How CloudFront processes responses from your custom origin

This topic contains information about how CloudFront processes responses from your custom origin.

Topics

- 100 Continue responses
- Caching
- Canceled requests
- Content negotiation
- Cookies
- Dropped TCP connections
- HTTP response headers that CloudFront removes or replaces
- Maximum cacheable file size
- Origin unavailable
- Redirects

- Transfer-Encoding header

`100 Continue` responses

Your origin cannot send more than one 100-Continue response to CloudFront. After the first 100-Continue response, CloudFront expects an HTTP 200 OK response. If your origin sends another 100-Continue response after the first one, CloudFront will return an error.

Caching

- Ensure that the origin server sets valid and accurate values for the `Date` and `Last-Modified` header fields.
- CloudFront normally respects a `Cache-Control: no-cache` header in the response from the origin. For an exception, see Simultaneous requests for the same object (request collapsing).

Canceled requests

If an object is not in the edge cache, and if a viewer terminates a session (for example, closes a browser) after CloudFront gets the object from your origin but before it can deliver the requested object, CloudFront does not cache the object in the edge location.

Content negotiation

If your origin returns `Vary:*` in the response, and if the value of **Minimum TTL** for the corresponding cache behavior is **0**, CloudFront caches the object but still forwards every subsequent request for the object to the origin to confirm that the cache contains the latest version of the object. CloudFront doesn't include any conditional headers, such as `If-None-Match` or `If-Modified-Since`. As a result, your origin returns the object to CloudFront in response to every request.

If your origin returns `Vary:*` in the response, and if the value of **Minimum TTL** for the corresponding cache behavior is any other value, CloudFront processes the `Vary` header as described in HTTP response headers that CloudFront removes or replaces.

Cookies

If you enable cookies for a cache behavior, and if the origin returns cookies with an object, CloudFront caches both the object and the cookies. Note that this reduces cacheability for an object. For more information, see Caching content based on cookies.

Dropped TCP connections

If the TCP connection between CloudFront and your origin drops while your origin is returning an object to CloudFront, CloudFront behavior depends on whether your origin included a `Content-Length` header in the response:

- **Content-Length header** – CloudFront returns the object to the viewer as it gets the object from your origin. However, if the value of the `Content-Length` header doesn't match the size of the object, CloudFront doesn't cache the object.
- **Transfer-Encoding: Chunked** – CloudFront returns the object to the viewer as it gets the object from your origin. However, if the chunked response is not complete, CloudFront does not cache the object.
- **No Content-Length header** – CloudFront returns the object to the viewer and caches it, but the object may not be complete. Without a `Content-Length` header, CloudFront cannot determine whether the TCP connection was dropped accidentally or on purpose.

We recommend that you configure your HTTP server to add a `Content-Length` header to prevent CloudFront from caching partial objects.

HTTP response headers that CloudFront removes or replaces

CloudFront removes or updates the following header fields before forwarding the response from your origin to the viewer:

- `Set-Cookie` – If you configure CloudFront to forward cookies, it will forward the `Set-Cookie` header field to clients. For more information, see Caching content based on cookies.
- `Trailer`
- `Transfer-Encoding` – If your origin returns this header field, CloudFront sets the value to `chunked` before returning the response to the viewer.
- `Upgrade`
- `Vary` – Note the following:
  - If you configure CloudFront to forward any of the device-specific headers to your origin (`CloudFront-Is-Desktop-Viewer`, `CloudFront-Is-Mobile-Viewer`, `CloudFront-Is-SmartTV-Viewer`, `CloudFront-Is-Tablet-Viewer`) and you configure your origin to return `Vary:User-Agent` to CloudFront, CloudFront returns `Vary:User-Agent` to the viewer. For more information, see Configuring caching based on the device type.

- If you configure your origin to include either `Accept-Encoding` or `Cookie` in the `Vary` header, CloudFront includes the values in the response to the viewer.
- If you configure CloudFront to forward headers to your origin, and if you configure your origin to return the header names to CloudFront in the `Vary` header (for example, `Vary:Accept-Charset,Accept-Language`), CloudFront returns the `Vary` header with those values to the viewer.
- For information about how CloudFront processes a value of `*` in the `Vary` header, see Content negotiation.
- If you configure your origin to include any other values in the `Vary` header, CloudFront removes the values before returning the response to the viewer.

- `Via` – CloudFront sets the value to the following in the response to the viewer:

  `Via: http-version alphanumeric-string.cloudfront.net (CloudFront)`

  For example, the value is something like the following:

  `Via: 1.1 1026589cc7887e7a0dc7827b4example.cloudfront.net (CloudFront)`

Maximum cacheable file size

The maximum size of a response body that CloudFront saves in its cache is 50 GB. This includes chunked transfer responses that don't specify the `Content-Length` header value.

You can use CloudFront to cache an object that is larger than this size by using range requests to request the objects in parts that are each 50 GB or smaller. CloudFront caches these parts because each of them is 50 GB or smaller. After the viewer retrieves all the parts of the object, it can reconstruct the original, larger object. For more information, see Use range requests to cache large objects.

Origin unavailable

If your origin server is unavailable and CloudFront gets a request for an object that is in the edge cache but that has expired (for example, because the period of time specified in the `Cache-Control max-age` directive has passed), CloudFront either serves the expired version of the object or serves a custom error page. For more information about CloudFront behavior when you've configured custom error pages, see How CloudFront processes errors when you have configured custom error pages.

In some cases, an object that is seldom requested is evicted and is no longer available in the edge cache. CloudFront can't serve an object that has been evicted.

Redirects

If you change the location of an object on the origin server, you can configure your web server to redirect requests to the new location. After you configure the redirect, the first time a viewer submits a request for the object, CloudFront Front sends the request to the origin, and the origin responds with a redirect (for example, `302 Moved Temporarily`). CloudFront caches the redirect and returns it to the viewer. CloudFront does not follow the redirect.

You can configure your web server to redirect requests to one of the following locations:

- The new URL of the object on the origin server. When the viewer follows the redirect to the new URL, the viewer bypasses CloudFront and goes straight to the origin. As a result, we recommend that you not redirect requests to the new URL of the object on the origin.
- The new CloudFront URL for the object. When the viewer submits the request that contains the new CloudFront URL, CloudFront gets the object from the new location on your origin, caches it at the edge location, and returns the object to the viewer. Subsequent requests for the object will be served by the edge location. This avoids the latency and load associated with viewers requesting the object from the origin. However, every new request for the object will incur charges for two requests to CloudFront.

`Transfer-Encoding` header

CloudFront supports only the `chunked` value of the `Transfer-Encoding` header. If your origin returns `Transfer-Encoding: chunked`, CloudFront returns the object to the client as the object is received at the edge location, and caches the object in chunked format for subsequent requests.

If the viewer makes a `Range GET` request and the origin returns `Transfer-Encoding: chunked`, CloudFront returns the entire object to the viewer instead of the requested range.

We recommend that you use chunked encoding if the content length of your response cannot be predetermined. For more information, see Dropped TCP connections.

Request and response behavior for origin groups

Requests to an origin group work the same as requests to an origin that is not set up as an origin group, except when there is an origin failover. As with any other origin, when CloudFront receives

a request and the content is already cached in an edge location, the content is served to viewers from the cache. When there's a cache miss and the origin is an origin group, viewer requests are forwarded to the primary origin in the origin group.

The request and response behavior for the primary origin is the same as it is for an origin that isn't in an origin group. For more information, see Request and response behavior for Amazon S3 origins and Request and response behavior for custom origins.

The following describes the behavior for origin failover when the primary origin returns specific HTTP status codes:

- HTTP 2xx status code (success): CloudFront caches the file and returns it to the viewer.
- HTTP 3xx status code (redirection): CloudFront returns the status code to the viewer.
- HTTP 4xx or 5xx status code (client/server error): If the returned status code has been configured for failover, CloudFront sends the same request to the secondary origin in the origin group.
- HTTP 4xx or 5xx status code (client/server error): If the returned status code has not been configured for failover, CloudFront returns the error to the viewer.

CloudFront fails over to the secondary origin only when the HTTP method of the viewer request is `GET`, `HEAD`, or `OPTIONS`. CloudFront does not fail over when the viewer sends a different HTTP method (for example `POST`, `PUT`, and so on).

When CloudFront sends a request to a secondary origin, the response behavior is the same as for a CloudFront origin that's not in an origin group.

For more information about origin groups, see Optimizing high availability with CloudFront origin failover.

Adding custom headers to origin requests

You can configure CloudFront to add custom headers to the requests that it sends to your origin. These custom headers enable you to send and gather information from your origin that you don't get with typical viewer requests. These headers can even be customized for each origin. CloudFront supports custom headers for both for custom and Amazon S3 origins.

Topics

- Use cases for origin custom headers

- Configuring CloudFront to add custom headers to origin requests
- Custom headers that CloudFront can't add to origin requests
- Configuring CloudFront to forward the Authorization header

Use cases for origin custom headers

You can use custom headers for a variety of things, such as the following:

Identifying requests from CloudFront

You can identify the requests that your origin receives from CloudFront. This can be useful if you want to know if users are bypassing CloudFront, or if you're using more than one CDN and you want information about which requests are coming from each CDN.

> **Note**
>
> If you're using an Amazon S3 origin and you enable Amazon S3 server access logging, the logs don't include header information.

Determining which requests come from a particular distribution

If you configure more than one CloudFront distribution to use the same origin, you can add different custom headers in each distribution. You can then use the logs from your origin to determine which requests came from which CloudFront distribution.

Enabling cross-origin resource sharing (CORS)

If some of your viewers don't support cross-origin resource sharing (CORS), you can configure CloudFront to always add the `Origin` header to requests that it sends to your origin. Then you can configure your origin to return the `Access-Control-Allow-Origin` header for every request. You must also configure CloudFront to respect CORS settings.

Controlling access to content

You can use custom headers to control access to content. By configuring your origin to respond to requests only when they include a custom header that gets added by CloudFront, you prevent users from bypassing CloudFront and accessing your content directly on the origin. For more information, see Restricting access to files on custom origins.

Configuring CloudFront to add custom headers to origin requests

To configure a distribution to add custom headers to requests that it sends to your origin, update the origin configuration using one of the following methods:

- **CloudFront console** – When you create or update a distribution, specify header names and values in the **Origin Custom Headers** settings. For more information, see Creating a distribution or Updating a distribution.
- **CloudFront API** – For each origin that you want to add custom headers to, specify the header names and values in the `CustomHeaders` field inside `Origin`. For more information, see CreateDistribution or UpdateDistribution.

If the header names and values that you specify are not already present in the viewer request, CloudFront adds them to the origin request. If a header is present, CloudFront overwrites the header value before forwarding the request to the origin.

For the quotas (formerly known as limits) that apply to origin custom headers, see Quotas on headers.

Custom headers that CloudFront can't add to origin requests

You can't configure CloudFront to add any of the following headers to requests that it sends to your origin:

- `Cache-Control`
- `Connection`
- `Content-Length`
- `Cookie`
- `Host`
- `If-Match`
- `If-Modified-Since`
- `If-None-Match`
- `If-Range`
- `If-Unmodified-Since`
- `Max-Forwards`

- `Pragma`
- `Proxy-Authorization`
- `Proxy-Connection`
- `Range`
- `Request-Range`
- `TE`
- `Trailer`
- `Transfer-Encoding`
- `Upgrade`
- `Via`
- Headers that begin with `X-Amz-`
- Headers that begin with `X-Edge-`
- `X-Real-Ip`

Configuring CloudFront to forward the `Authorization` header

When CloudFront forwards a viewer request to your origin, CloudFront removes some viewer headers by default, including the `Authorization` header. To make sure that your origin always receives the `Authorization` header in origin requests, you have the following options:

- Add the `Authorization` header to the cache key using a cache policy. All headers in the cache key are automatically included in origin requests. For more information, see Controlling the cache key.
- Use an origin request policy that forwards all viewer headers to the origin. You cannot forward the `Authorization` header individually in an origin request policy, but when you forward all viewer headers CloudFront includes the `Authorization` header in viewer requests. CloudFront provides a managed origin request policy for this use case, called **Managed-AllViewer**. For more information, see Using the managed origin request policies.

How CloudFront processes partial requests for an object (range GETs)

For a large object, the viewer (web browser or other client) can make multiple GET requests and use the Range request header to download the object in smaller parts. These requests for ranges of bytes, sometimes known as Range GET requests, improve the efficiency of partial downloads and the recovery from partially failed transfers.

When CloudFront receives a Range GET request, it checks the cache in the edge location that received the request. If the cache in that edge location already contains the entire object or the requested part of the object, CloudFront immediately serves the requested range from the cache.

If the cache doesn't contain the requested range, CloudFront forwards the request to the origin. (To optimize performance, CloudFront may request a larger range than the client requested in the Range GET.) What happens next depends on whether the origin supports Range GET requests:

- **If the origin supports Range GET requests:** It returns the requested range. CloudFront serves the requested range and also caches it for future requests. (Amazon S3 supports Range GET requests, as do many HTTP servers.)
- **If the origin doesn't support Range GET requests:** It returns the entire object. CloudFront serves the current request by sending the entire object while also caching it for future requests. After CloudFront caches the entire object in an edge cache, it responds to new Range GET requests by serving the requested range.

In either case, CloudFront begins to serve the requested range or object to the end user as soon as the first byte arrives from the origin.

> **Note**
>
> If the viewer makes a Range GET request and the origin returns `Transfer-Encoding: chunked`, CloudFront returns the entire object to the viewer instead of the requested range.

CloudFront generally follows the RFC specification for the Range header. However, if your Range headers don't adhere to the following requirements, CloudFront returns HTTP status code 200 with the full object instead of status code 206 with the specified ranges:

- The ranges must be listed in ascending order. For example, `100-200,300-400` is valid, `300-400,100-200` is not valid.
- The ranges must not overlap. For example, `100-200,150-250` is not valid.
- All of the ranges specifications must be valid. For example, you can't specify a negative value as part of a range.

For more information about the Range request header, see Range Requests in RFC 7233, or Range in the MDN Web Docs.

Use range requests to cache large objects

When caching is enabled, CloudFront doesn't retrieve or cache an object that is larger than 50 GB. When an origin indicates that the object is larger than this size (in the `Content-Length` response header), CloudFront closes the connection to the origin and returns an error to the viewer. (With caching disabled, CloudFront can retrieve an object that is larger than this size from the origin and pass it along to the viewer. However, CloudFront doesn't cache the object.)

However, with range requests, you can use CloudFront to cache an object that is larger than the maximum cacheable file size. For example, consider an origin with a 100 GB object. With caching enabled, CloudFront doesn't retrieve or cache an object this large. However, the viewer can send multiple range requests to retrieve this object in parts, with each part smaller than 50 GB. For example, the viewer can request the object in 20 GB parts by sending a request with the header `Range: bytes=0-21474836480` to retrieve the first part, another request with the header `Range: bytes=21474836481-42949672960` to retrieve the next part, and so on. When the viewer has received all of the parts, it can combine them to construct the original 100 GB object. In this case, CloudFront caches each of the 20 GB parts of the object and can respond to subsequent requests for the same part from the cache.

How CloudFront processes HTTP 3xx status codes from your origin

When CloudFront requests an object from your Amazon S3 bucket or custom origin server, your origin sometimes returns an HTTP 3xx status code. This typically indicates one of the following:

- The object's URL has changed (for example, status codes 301, 302, 307, or 308)
- The object hasn't changed since the last time CloudFront requested it (status code 304)

CloudFront caches 3xx responses according to the settings in your CloudFront distribution and the headers in the response. CloudFront caches 307 and 308 responses only when you include the `Cache-Control` header in responses from the origin. For more information, see Managing how long content stays in the cache (expiration).

If your origin returns a redirect status code (for example, 301 or 307), CloudFront doesn't follow the redirect. CloudFront passes along the 301 or 307 response to the viewer, who can follow the redirect by sending a new request.

How CloudFront processes and caches HTTP 4xx and 5xx status codes from your origin

Topics

- How CloudFront processes errors when you have configured custom error pages
- How CloudFront processes errors when you have not configured custom error pages
- HTTP 4xx and 5xx status codes that CloudFront caches

When CloudFront requests an object from your Amazon S3 bucket or custom origin server, your origin sometimes returns an HTTP 4xx or 5xx status code, which indicates that an error has occurred. CloudFront behavior depends on:

- Whether you have configured custom error pages.
- Whether you have configured how long you want CloudFront to cache error responses from your origin (error caching minimum TTL).
- The status code.
- For 5xx status codes, whether the requested object is currently in the CloudFront edge cache.
- For some 4xx status codes, whether the origin returns a `Cache-Control max-age` or `Cache-Control s-maxage` header.

CloudFront always caches responses to `GET` and `HEAD` requests. You can also configure CloudFront to cache responses to `OPTIONS` requests. CloudFront does not cache responses to requests that use the other methods.

If the origin doesn't respond, the CloudFront request to the origin times out which is considered an HTTP 5xx error from the origin, even though the origin didn't respond with that error. In

that scenario, CloudFront continues to serve cached content. For more information, see Origin unavailable.

If you have enabled logging, CloudFront writes the results to the logs regardless of the HTTP status code.

For more information about features and options that relate to the error message returned from CloudFront, see the following:

- For information about settings for custom error pages in the CloudFront console, see Custom error pages and error caching.
- For information about the error caching minimum TTL in the CloudFront console, see Error caching minimum TTL (seconds).
- For a list of the HTTP status codes that CloudFront caches, see HTTP 4xx and 5xx status codes that CloudFront caches.

How CloudFront processes errors when you have configured custom error pages

If you have configured custom error pages, CloudFront behavior depends on whether the requested object is in the edge cache.

The requested object is not in the edge cache

CloudFront continues to try to get the requested object from your origin when all of the following are true:

- A viewer requests an object.
- The object isn't in the edge cache.
- Your origin returns an HTTP 4xx or 5xx status code and one of the following is true:
  - Your origin returns an HTTP 5xx status code instead of returning a 304 status code (Not Modified) or an updated version of the object.
  - Your origin returns an HTTP 4xx status code that is not restricted by a cache control header and is included in the following list of status codes: HTTP 4xx and 5xx status codes that CloudFront always caches.
  - Your origin returns an HTTP 4xx status code without a `Cache-Control max-age` header or a `Cache-Control s-maxage` header, and the status code is included in the following

list of status codes: Control HTTP 4xx status codes that CloudFront caches based on `Cache-Control` headers.

CloudFront does the following:

1. In the CloudFront edge cache that received the viewer request, CloudFront checks your distribution configuration and gets the path of the custom error page that corresponds with the status code that your origin returned.
2. CloudFront finds the first cache behavior in your distribution that has a path pattern that matches the path of the custom error page.
3. The CloudFront edge location sends a request for the custom error page to the origin that is specified in the cache behavior.
4. The origin returns the custom error page to the edge location.
5. CloudFront returns the custom error page to the viewer that made the request, and also caches the custom error page for the maximum of the following:
   - The amount of time specified by the error caching minimum TTL (10 seconds by default)
   - The amount of time specified by a `Cache-Control max-age` header or a `Cache-Control s-maxage` header that is returned by the origin when the first request generated the error
6. After the caching time (determined in Step 5) has elapsed, CloudFront tries again to get the requested object by forwarding another request to your origin. CloudFront continues to retry at intervals specified by the error caching minimum TTL.

The requested object is in the edge cache

CloudFront continues to serve the object that is currently in the edge cache when all of the following are true:

- A viewer requests an object.
- The object is in the edge cache but it has expired.
- Your origin returns an HTTP 5xx status code instead of returning a 304 status code (Not Modified) or an updated version of the object.

CloudFront does the following:

1. If your origin returns a 5xx status code, CloudFront serves the object even though it has expired. For the duration of the error caching minimum TTL, CloudFront continues to respond to viewer requests by serving the object from the edge cache.

   If your origin returns a 4xx status code, CloudFront returns the status code, not the requested object, to the viewer.

2. After the error caching minimum TTL has elapsed, CloudFront tries again to get the requested object by forwarding another request to your origin. Note that if the object is not requested frequently, CloudFront might evict it from the edge cache while your origin server is still returning 5xx responses. For information about how long objects stay in CloudFront edge caches, see Managing how long content stays in the cache (expiration).

How CloudFront processes errors when you have not configured custom error pages

If you have not configured custom error pages, CloudFront behavior depends on whether the requested object is in the edge cache.

The requested object is not in the edge cache

CloudFront continues to try to get the requested object from your origin when all of the following are true:

- A viewer requests an object.
- The object isn't in the edge cache.
- Your origin returns an HTTP 4xx or 5xx status code and one of the following is true:
  - Your origin returns an HTTP 5xx status code instead of returning a 304 status code (Not Modified) or an updated version of the object.
  - Your origin returns an HTTP 4xx status code that is not restricted by a cache control header and is included in the following list of status codes: HTTP 4xx and 5xx status codes that CloudFront always caches
  - Your origin returns an HTTP 4xx status code without a `Cache-Control max-age` header or a `Cache-Control s-maxage` header and the status code is included in the following list of status codes: Control HTTP 4xx status codes that CloudFront caches based on `Cache-Control` headers.

CloudFront does the following:

1. CloudFront returns the 4xx or 5xx status code to the viewer, and also caches status code in the edge cache that received the request for the maximum of the following:
   - The amount of time specified by the error caching minimum TTL (10 seconds by default)
   - The amount of time specified by a `Cache-Control max-age` header or a `Cache-Control s-maxage` header that is returned by the origin when the first request generated the error
2. For the duration of the caching time (determined in Step 1), CloudFront responds to subsequent viewer requests for the same object with the cached 4xx or 5xx status code.
3. After the caching time (determined in Step 1) has elapsed, CloudFront tries again to get the requested object by forwarding another request to your origin. CloudFront continues to retry at intervals specified by the error caching minimum TTL.

The requested object is in the edge cache

CloudFront continues to serve the object that is currently in the edge cache when all of the following are true:

- A viewer requests an object.
- The object is in the edge cache but it has expired.
- Your origin returns an HTTP 5xx status code instead of returning a 304 status code (Not Modified) or an updated version of the object.

CloudFront does the following:

1. If your origin returns a 5xx error code, CloudFront serves the object even though it has expired. For the duration of the error caching minimum TTL (10 seconds by default), CloudFront continues to respond to viewer requests by serving the object from the edge cache.

   If your origin returns a 4xx status code, CloudFront returns the status code, not the requested object, to the viewer.
2. After the error caching minimum TTL has elapsed, CloudFront tries again to get the requested object by forwarding another request to your origin. Note that if the object is not requested frequently, CloudFront might evict it from the edge cache while your origin server is still returning 5xx responses. For information about how long objects stay in CloudFront edge caches, see Managing how long content stays in the cache (expiration).

HTTP 4xx and 5xx status codes that CloudFront caches

CloudFront caches HTTP 4xx and 5xx status codes returned by your origin, depending on the specific status code that is returned and whether your origin returns specific headers in the response.

HTTP 4xx and 5xx status codes that CloudFront always caches

CloudFront always caches the following HTTP 4xx and 5xx status codes returned by your origin. If you have configured a custom error page for an HTTP status code, CloudFront caches the custom error page.

404	Not Found
414	Request-URI Too Large
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Time-out

HTTP 4xx status codes that CloudFront caches based on `Cache-Control` headers

CloudFront only caches the following HTTP 4xx status codes returned by your origin if your origin returns a `Cache-Control max-age` or `Cache-Control s-maxage` header. If you have configured a custom error page for one of these HTTP status codes—and your origin returns one of the cache control headers—CloudFront caches the custom error page.

400	Bad Request
403	Forbidden

405	Method Not Allowed
412[1]	Precondition Failed
415[1]	Unsupported Media Type

[1]CloudFront doesn't support creating custom error pages for these HTTP status codes.

Video on demand and live streaming video with CloudFront

You can use CloudFront to deliver video on demand (VOD) or live streaming video using any HTTP origin. One way you can set up video workflows in the cloud is by using CloudFront together with Amazon Media Services.

Topics

- About streaming video: video on demand and live streaming
- Delivering video on demand (VOD) with CloudFront
- Delivering live streaming video with CloudFront and Amazon Media Services

About streaming video: video on demand and live streaming

You must use an encoder to package video content before CloudFront can distribute the content. The packaging process creates *segments* that contain your audio, video, and captions content. It also generates manifest files, which describe in a specific order what segments to play and when. Common package formats are MPEG DASH, Apple HLS, Microsoft Smooth Streaming, and CMAF.

Video on demand (VOD) streaming

For video on demand (VOD) streaming, your video content is stored on a server and viewers can watch it at any time. To make an asset that viewers can stream, use an encoder, such as AWS Elemental MediaConvert, to format and package your media files.

After your video is packaged into the right formats, you can store it on a server or in an Amazon S3 bucket, and then deliver it with CloudFront as viewers request it.

Live video streaming

For live video streaming, your video content is streamed real time as live events happen, or is set up as a 24x7 live channel. To create live outputs for broadcast and streaming delivery, use an encoder such as AWS Elemental MediaLive, to compress the video and format it for viewing devices.

After your video is encoded, you can store it in AWS Elemental MediaStore or convert it into different delivery formats by using AWS Elemental MediaPackage. Use either of these origins

to set up a CloudFront distribution to deliver the content. For specific steps and guidance for creating distributions that work together with these services, see Serving video using AWS Elemental MediaStore as the origin and Serving live video formatted with AWS Elemental MediaPackage.

Wowza and Unified Streaming also provide tools that you can use for streaming video with CloudFront. For more information about using Wowza with CloudFront, see Bring your Wowza Streaming Engine license to CloudFront live HTTP streaming on the Wowza documentation website. For information about using Unified Streaming with CloudFront for VOD streaming, see CloudFront on the Unified Streaming documentation website.

Delivering video on demand (VOD) with CloudFront

To deliver video on demand (VOD) streaming with CloudFront, use the following services:

- Amazon S3 to store the content in its original format and to store the transcoded video.
- An encoder (such as AWS Elemental MediaConvert) to transcode the video into streaming formats.
- CloudFront to deliver the transcoded video to viewers. For Microsoft Smooth Streaming, see Configuring video on demand for Microsoft Smooth Streaming.

To create a VOD solution with CloudFront

1. Upload your content to an Amazon S3 bucket. To learn more about working with Amazon S3, see the Amazon Simple Storage Service User Guide.
2. Transcode your content by using a MediaConvert job. The job converts your video into the formats required by the players that your viewers use. You can also use the job to create assets that vary in resolution and bitrate. These assets are used for adaptive bitrate (ABR) streaming, which adjusts the viewing quality depending on the viewer's available bandwidth. MediaConvert stores the transcoded video in an S3 bucket.
3. Deliver your converted content by using a CloudFront distribution. Viewers can watch the content on any device, at any time.

Tip

You can explore how to use an Amazon CloudFormation template to deploy a VOD Amazon solution together with all the associated components. To see the steps for using the template, see Automated Deployment in the *Video on Demand on Amazon* guide.

Configuring video on demand for Microsoft Smooth Streaming

You have the following options for using CloudFront to distribute video on demand (VOD) content that you've transcoded into the Microsoft Smooth Streaming format:

- Specify a web server that runs Microsoft IIS and supports Smooth Streaming as the origin for your distribution.
- Enable Smooth Streaming in the cache behaviors of a CloudFront distribution. Because you can use multiple cache behaviors in a distribution, you can use one distribution for Smooth Streaming media files as well as other content.

Important

If you specify a web server running Microsoft IIS as your origin, do *not* enable Smooth Streaming in the cache behaviors of your CloudFront distribution. CloudFront can't use a Microsoft IIS server as an origin if you enable Smooth Streaming as a cache behavior.

If you enable Smooth Streaming in a cache behavior (that is, you do not have a server that is running Microsoft IIS), note the following:

- You can still distribute other content using the same cache behavior if the content matches the value of **Path Pattern** for that cache behavior.
- CloudFront can use either an Amazon S3 bucket or a custom origin for Smooth Streaming media files. CloudFront cannot use a Microsoft IIS Server as an origin if you enable Smooth Streaming for the cache behavior.
- You cannot invalidate media files in the Smooth Streaming format. If you want to update files before they expire, you must rename them. For more information, see Adding, removing, or replacing content that CloudFront distributes.

For information about Smooth Streaming clients, see Smooth Streaming on the Microsoft documentation website.

To use CloudFront to distribute Smooth Streaming files when a Microsoft IIS web server isn't the origin

1. Transcode your media files into Smooth Streaming fragmented MP4 format.
2. Do one of the following:
   - **If you're using the CloudFront console:** When you create or update a distribution, enable Smooth Streaming in one or more of the distribution's cache behaviors.
   - **If you're using the CloudFront API:** Add the `SmoothStreaming` element to the `DistributionConfig` complex type for one or more of the distribution's cache behaviors.
3. Upload the Smooth Streaming files to your origin.
4. Create either a `clientaccesspolicy.xml` or a `crossdomainpolicy.xml` file, and add it to a location that is accessible at the root of your distribution, for example, `https://d111111abcdef8.cloudfront.net/clientaccesspolicy.xml`. The following is an example policy:

   ```
   <?xml version="1.0" encoding="utf-8"?>
   <access-policy>
   <cross-domain-access>
   <policy>
   <allow-from http-request-headers="*">
   <domain uri="*"/>
   </allow-from>
   <grant-to>
   <resource path="/" include-subpaths="true"/>
   </grant-to>
   </policy>
   </cross-domain-access>
   </access-policy>
   ```

   For more information, see Making a Service Available Across Domain Boundaries on the Microsoft Developer Network website.
5. For links in your application (for example, a media player), specify the URL for the media file in the following format:

   `https://d111111abcdef8.cloudfront.net/video/presentation.ism/Manifest`

Delivering live streaming video with CloudFront and Amazon Media Services

To use Amazon Media Services with CloudFront to deliver live content to a global audience, follow the guidance included in this section.

Use AWS Elemental MediaLive to encode live video streams in real time. To encode a large video stream, MediaLive compresses it into smaller versions (*encodes*) that can be distributed to your viewers.

After you compress a live video stream, you can use either of the following two main options to prepare and serve the content:

- **Convert your content into required formats, and then serve it:** If you require content in multiple formats, use AWS Elemental MediaPackage to package the content for different device types. When you package the content, you can also implement extra features and add digital rights management (DRM) to prevent unauthorized use of your content. For step-by-step instructions for using CloudFront to serve content that MediaPackage formatted, see Serving live video formatted with AWS Elemental MediaPackage.
- **Store and serve your content using scalable origin:** If MediaLive encoded content in the formats required by all of the devices that your viewers use, use a highly scalable origin like AWS Elemental MediaStore to serve the content. For step-by-step instructions for using CloudFront to serve content that is stored in a MediaStore container, see Serving video using AWS Elemental MediaStore as the origin.

After you've set up your origin by using one of these options, you can distribute live streaming video to viewers by using CloudFront.

> **Tip**
>
> You can learn about an Amazon solution that automatically deploys services for building a highly available real-time viewing experience. To see the steps to automatically deploy this solution, see Live Streaming Automated Deployment.

Topics

- Serving video using AWS Elemental MediaStore as the origin

- Serving live video formatted with AWS Elemental MediaPackage

Serving video using AWS Elemental MediaStore as the origin

If you have video stored in an AWS Elemental MediaStore container, you can create a CloudFront distribution to serve the content.

To get started, you grant CloudFront access to your MediaStore container. Then you create a CloudFront distribution and configure it to work with MediaStore.

To serve content from an AWS Elemental MediaStore container

1. Follow the procedure at Allowing Amazon CloudFront to access your AWS Elemental MediaStore container, and then return to these steps to create your distribution.
2. Create a distribution with the following settings:

   Origin domain

   The data endpoint that is assigned to your MediaStore container. From the dropdown list, choose the MediaStore container for your live video.

   Origin path

   The folder structure in the MediaStore container where your objects are stored. For more information, see the section called “Origin path”.

   Add custom header

   Add header names and values if you want CloudFront to add custom headers when it forwards requests to your origin.

   Viewer protocol policy

   Choose **Redirect HTTP to HTTPS**. For more information, see the section called “Viewer protocol policy”.

   Cache policy and origin request policy

   For **Cache policy**, choose **Create policy**, and then create a cache policy that’s appropriate for your caching needs and segment durations. After you create the policy, refresh the list of cache policies and choose the policy that you just created.

   For **Origin request policy**, choose **CORS-CustomOrigin** from the dropdown list.

For the other settings, you can set specific values based on other technical requirements or the needs of your business. For a list of all the options for distributions and information about setting them, see the section called "Values that you specify".

3. For links in your application (for example, a media player), specify the name of the media file in the same format that you use for other objects that you're distributing using CloudFront.

Serving live video formatted with AWS Elemental MediaPackage

If you formatted a live stream by using AWS Elemental MediaPackage, you can create a CloudFront distribution and configure cache behaviors to serve the live stream. The following process assumes that you have already created a channel and added endpoints for your live video using MediaPackage.

To create a CloudFront distribution for MediaPackage manually, follow these steps:

Steps

- Step 1: Create and configure a CloudFront distribution
- Step 2: Add Origins for the domains of your MediaPackage endpoints
- Step 3: Configure cache behaviors for all endpoints
- Step 4: Enable header-based MediaPackage CDN Authorization
- Step 5: Use CloudFront to serve the live stream channel

Step 1: Create and configure a CloudFront distribution

Complete the following procedure to set up a CloudFront distribution for the live video channel that you created with MediaPackage.

To create a distribution for your live video channel

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose **Create distribution**.
3. Choose the settings for the distribution, including the following:

Origin domain

The origin where your MediaPackage live video channel and endpoints are. Choose the text field, then from the dropdown list, choose the MediaPackage origin domain for your live video. You can map one domain to several origin endpoints.

If you created your origin domain using another Amazon account, type the origin URL value into the field. The origin must be an HTTPS URL.

For example, for an HLS endpoint like `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8`, the origin domain is `3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com`.

For more information, see the section called “Origin domain”.

Origin path

The path to the MediaPackage endpoint from where the content is served.

The **Origin path** field is not filled in for you. You must manually enter the correct origin path.

For more information about how an origin path works, see the section called “Origin path”.

> **Important**
>
> The wildcard path `*` is required to route somewhere in the CloudFront distribution. To prevent requests not matching an explicit path from routing to the real origin, create a "dummy" origin for that wildcard path.

Example : Creating a "dummy" origin

In the following example, the endpoints `abc123` and `def456` route to the "real" origin, but requests for any other endpoint's video content route to `mediapackage.us-west-2.amazonaws.com` without the proper subdomain, which results in an HTTP `404` error.

MediaPackage endpoints:

```
https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/
index.m3u8
https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/def456/
index.m3u8
```

CloudFront Origin A:

```
Domain: 3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com
Path: None
```

CloudFront Origin B:

```
Domain: mediapackage.us-west-2.amazonaws.com
Path: None
```

CloudFront cache behavior:

```
1. Path: /out/v1/abc123/* forward to Origin A
2. Path: /out/v1/def456/* forward to Origin A
3. Path: * forward to Origin B
```

For the other distribution settings, set specific values based on other technical requirements or the needs of your business. For a list of all the options for distributions and information about setting them, see the section called “Values that you specify”.

When you finish choosing the other distribution settings, choose **Create distribution**.

4. Choose the distribution that you just created, then choose **Behaviors**.
5. Select the default cache behavior, then choose **Edit**. Specify the correct cache behavior settings for the channel that you chose for the origin. Later, you’ll add one or more additional origins and edit cache behavior settings for them.
6. Go to the CloudFront distributions page.
7. Wait until the value of the **Last modified** column for your distribution has changed from **Deploying** to a date and time, indicating that CloudFront has created your distribution.

Step 2: Add Origins for the domains of your MediaPackage endpoints

Repeat the steps here to add each of your MediaPackage channel endpoints to your distribution, keeping in mind the need to create a "dummy" origin.

To add other endpoints as origins

1. On the CloudFront console, choose the distribution that you created for your channel.
2. Choose **Origins**, then choose **Create origin**.
3. For **Origin domain**, in the dropdown list, choose a MediaPackage endpoint for your channel.
4. For the other settings, set the values based on other technical requirements or the needs of your business. For more information, see the section called “Origin settings”.
5. Choose **Create origin**.

Step 3: Configure cache behaviors for all endpoints

For each endpoint, you must configure cache behaviors to add path patterns that route requests correctly. The path patterns that you specify depend on the video format that you’re serving. The following procedure includes the path pattern information to use for Apple HLS, CMAF, DASH, and Microsoft Smooth Streaming formats.

You typically set up two cache behaviors for each endpoint:

- The parent manifest, which is the index to your files.
- The segments, which are the files of the video content.

To create a cache behavior for an endpoint

1. On the CloudFront console, choose the distribution that you created for your channel.
2. Choose **Behaviors**, then choose **Create behavior**.
3. For **Path pattern**, use a specific MediaPackage `OriginEndpoint` GUID as a path prefix.

   Path patterns

   For an HLS endpoint like `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8`, create the following two cache behaviors:

- For parent and child manifests, use `/out/v1/abc123/*.m3u8`.
- For the content segments, use `/out/v1/abc123/*.ts`.

For a CMAF endpoint like `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8`, create the following two cache behaviors:

- For parent and child manifests, use `/out/v1/abc123/*.m3u8`.
- For the content segments, use `/out/v1/abc123/*.mp4`.

For a DASH endpoint like `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.mpd`, create the following two cache behaviors:

- For the parent manifest, use `/out/v1/abc123/*.mpd`.
- For the content segments, use `/out/v1/abc123/*.mp4`.

For a Microsoft Smooth Streaming endpoint like `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.ism`, only a manifest is served, so you create only one cache behavior: `out/v1/abc123/index.ism/*`.

4. For each cache behavior, specify values for the following settings:

Viewer protocol policy

Choose **Redirect HTTP to HTTPS**.

Cache policy and origin request policy

For **Cache policy**, choose **Create policy**. For your new cache policy, specify the following settings:

Minimum TTL

Set to 5 seconds or less, to help prevent serving stale content.

Query strings

For **Query strings** (in **Cache key settings**), choose **Include specified query strings**. For **Allow**, add the following values by typing them and then choosing **Add item**:

- Add `m` as a query string parameter that you want CloudFront to use as the basis for caching. The MediaPackage response always includes the tag `?m=###` to capture the

modified time of the endpoint. If content is already cached with a different value for this tag, CloudFront requests a new manifest instead of serving the cached version.

- If you're using the time-shifted viewing functionality in MediaPackage, specify `start` and end as additional query string parameters on the cache behavior for manifest requests (`*.m3u8`, `*.mpd`, and `index.ism/*`). This way, content is served that's specific to the requested time period in the manifest request. For more information about time-shifted viewing and formatting content start and end request parameters, see Time-shifted viewing in the *AWS Elemental MediaPackage User Guide*.
- If you're using the manifest filtering feature in MediaPackage, specify `aws.manifestfilter` as an additional query string parameter for the cache policy that you use with the cache behavior for manifest requests (`*.m3u8`, `*.mpd`, and `index.ism/*`). This configures your distribution to forward the `aws.manifestfilter` query string to your MediaPackage origin, which is required for the manifest filtering feature to work. For more information, see Manifest filtering in the *AWS Elemental MediaPackage User Guide*.
- If you're using low-latency HLS (LL-HLS), specify `_HLS_msn` and `_HLS_part` as additional query string parameters for the cache policy that you use with the cache behavior for manifest requests (`*.m3u8`). This configures your distribution to forward the `_HLS_msn` and `_HLS_part` query strings to your MediaPackage origin, which is required for the LL-HLS blocking playlist request feature to work.

5. Choose **Create**.
6. After you create the cache policy, go back to the cache behavior creation workflow. Refresh the list of cache policies, and choose the policy that you just created.
7. Choose **Create behavior**.
8. If your endpoint is not a Microsoft Smooth Streaming endpoint, repeat these steps to create a second cache behavior.

Step 4: Enable header-based MediaPackage CDN Authorization

We recommend enabling header-based MediaPackage CDN Authorization between MediaPackage endpoints and the CloudFront distribution. For more information, see Enable CDN authorization in MediaPackage in the AWS Elemental MediaPackage User Guide.

Step 5: Use CloudFront to serve the live stream channel

After you create the distribution, add the origins, create the cache behaviors, and enable header-based CDN authorization, you can serve the live stream channel using CloudFront. CloudFront routes requests from viewers to the correct MediaPackage endpoints based on the settings that you configured for the cache behaviors.

For links in your application (for example, a media player), specify the URL for the media file in the standard format for CloudFront URLs. For more information, see the section called “Customizing file URLs”.

Customizing at the edge with functions

With Amazon CloudFront, you can write your own code to customize how your CloudFront distributions process HTTP requests and responses. The code runs close to your viewers (users) to minimize latency, and you don't have to manage servers or other infrastructure. You can write code to manipulate the requests and responses that flow through CloudFront, perform basic authentication and authorization, generate HTTP responses at the edge, and more.

The code that you write and attach to your CloudFront distribution is called an *edge function*. CloudFront provides two ways to write and manage edge functions:

- **CloudFront Functions** – With CloudFront Functions, you can write lightweight functions in JavaScript for high-scale, latency-sensitive CDN customizations. The CloudFront Functions runtime environment offers submillisecond startup times, scales immediately to handle millions of requests per second, and is highly secure. CloudFront Functions is a native feature of CloudFront, which means you can build, test, and deploy your code entirely within CloudFront.
- **Lambda@Edge** – Lambda@Edge is an extension of Amazon Lambda that offers powerful and flexible computing for complex functions and full application logic closer to your viewers, and is highly secure. Lambda@Edge functions run in a Node.js or Python runtime environment. You publish them to a single Amazon Region, but when you associate the function with a CloudFront distribution, Lambda@Edge automatically replicates your code around the world.

If you run Amazon WAF on CloudFront, you can use Amazon WAF inserted headers for both CloudFront Functions and Lambda@Edge. This works for viewer and origin requests and responses.

Choosing between CloudFront Functions and Lambda@Edge

CloudFront Functions and Lambda@Edge both provide a way to run code in response to CloudFront events. However, there are important differences that distinguish them. These differences can help you choose the one that's right for your use case. The following table lists some of the important differences between CloudFront Functions and Lambda@Edge.

	CloudFront Functions	Lambda@Edge
Programming languages	JavaScript (ECMAScript 5.1 compliant)	Node.js and Python

	CloudFront Functions	Lambda@Edge
Event sources	• Viewer request • Viewer response	• Viewer request • Viewer response • Origin request • Origin response
Scale	10,000,000 requests per second or more	Up to 10,000 requests per second per Region
Function duration	Submillisecond	Up to 5 seconds (viewer request and viewer response) Up to 30 seconds (origin request and origin response)
Maximum memory	2 MB	128 – 3,008 MB
Maximum size of the function code and included libraries	10 KB	1 MB (viewer request and viewer response) 50 MB (origin request and origin response)
Network access	No	Yes
File system access	No	Yes
Access to the request body	No	Yes
Access to geolocation and device data	Yes	No (viewer request and viewer response) Yes (origin request and origin response)

	CloudFront Functions	Lambda@Edge
Can build and test entirely within CloudFront	Yes	No
Function logging and metrics	Yes	Yes
Pricing	Free tier available; charged per request	No free tier; charged per request and function duration

CloudFront Functions is ideal for lightweight, short-running functions for use cases like the following:

- **Cache key normalization** – You can transform HTTP request attributes (headers, query strings, cookies, and even the URL path) to create an optimal cache key, which can improve your cache hit ratio.
- **Header manipulation** – You can insert, modify, or delete HTTP headers in the request or response. For example, you can add a `True-Client-IP` header to every request.
- **URL redirects or rewrites** – You can redirect viewers to other pages based on information in the request, or rewrite all requests from one path to another.
- **Request authorization** – You can validate hashed authorization tokens, such as JSON web tokens (JWT), by inspecting authorization headers or other request metadata.

To get started with CloudFront Functions, see Customizing at the edge with CloudFront Functions.

Lambda@Edge is a good fit for the following scenarios:

- Functions that take several milliseconds or more to complete.
- Functions that require adjustable CPU or memory.
- Functions that depend on third-party libraries (including the Amazon SDK, for integration with other Amazon services).
- Functions that require network access to use external services for processing.
- Functions that require file system access or access to the body of HTTP requests.

To get started with Lambda@Edge, see Customizing at the edge with Lambda@Edge.

Customizing at the edge with CloudFront Functions

With CloudFront Functions, you can write lightweight functions in JavaScript for high-scale, latency-sensitive CDN customizations. Your functions can manipulate the requests and responses that flow through CloudFront, perform basic authentication and authorization, generate HTTP responses at the edge, and more. The CloudFront Functions runtime environment offers submillisecond startup times, scales immediately to handle millions of requests per second, and is highly secure. CloudFront Functions is a native feature of CloudFront, which means you can build, test, and deploy your code entirely within CloudFront.

CloudFront Functions is ideal for lightweight, short-running functions for use cases like the following:

- **Cache key normalization** – You can transform HTTP request attributes (headers, query strings, cookies, even the URL path) to create an optimal cache key, which can improve your cache hit ratio.
- **Header manipulation** – You can insert, modify, or delete HTTP headers in the request or response. For example, you can add a `True-Client-IP` header to every request.
- **Status code modification and body generation** – You can evaluate headers and respond back to viewers with customized content.
- **URL redirects or rewrites** – You can redirect viewers to other pages based on information in the request, or rewrite all requests from one path to another.
- **Request authorization** – You can validate hashed authorization tokens, such as JSON web tokens (JWT), by inspecting authorization headers or other request metadata.

When you associate a CloudFront function with a CloudFront distribution, CloudFront intercepts requests and responses at CloudFront edge locations and passes them to your function. You can invoke CloudFront Functions when the following events occur:

- When CloudFront receives a request from a viewer (viewer request)
- Before CloudFront returns the response to the viewer (viewer response)

For a quick introduction, see Tutorial: Creating a simple function with CloudFront Functions.

You can include variables in a CloudFront function by setting up the function to use key-value pairs that are stored in a key value store. For a quick introduction to including key-value pairs in a CloudFront function, see the section called “Tutorial: A function with key values”.

To get started writing function code and to read example code, see Writing function code and Example code.

Tutorial: Creating a simple function with CloudFront Functions

This tutorial shows you how to get started with CloudFront Functions. You can create a simple function that redirects the viewer to a different URL, and also returns a custom response header.

Prerequisites

To use CloudFront Functions, you need a CloudFront distribution. If you don’t have one, follow the steps in Getting started with a basic CloudFront distribution.

Creating the function

This procedure shows you how to use the CloudFront console to create a simple function that redirects the viewer to a different URL, and also returns a custom response header.

To create a function in the CloudFront console

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, choose **Functions** and then choose **Create function**.
3. On the **Create function** page, for **Name**, enter a function name such as *`MyFunctionName`*.
4. (Optional) For **Description**, enter a description for the function such as **`Simple test function`**.
5. For **Runtime**, keep the default selected JavaScript version.
6. Choose **Create function**.
7. Copy the following function code. This function code redirects the viewer to a different URL, and also returns a custom response header.

```
function handler(event) {
    // NOTE: This example function is for a viewer request event trigger.
```

```
    // Choose viewer request for event trigger when you associate this function
 with a distribution.
    var response = {
        statusCode: 302,
        statusDescription: 'Found',
        headers: {
            'cloudfront-functions': { value: 'generated-by-CloudFront-Functions' },
            'location': { value: 'https://aws.amazon.com/cloudfront/' }
        }
    };
    return response;
}
```

8. For **Function code**, paste the code into the code editor to replace the default code.
9. Choose **Save changes**.
10. (Optional) You can test the function before you publish it. This tutorial doesn't describe how to test a function. For more information, see Testing functions.
11. Choose the **Publish** tab and then choose **Publish function**. You *must* publish the function before you can associate it with your CloudFront distribution.
12. Next, you can associate the function with a distribution or cache behavior. On the *MyFunctionName* page, choose the **Publish** tab.

> **Warning**
>
> In the following steps, choose a distribution or a cache behavior that's used for testing. Don't associate this test function with a distribution or cache behavior that's used in production.

13. Choose **Add association**.
14. On the **Associate** dialog box, choose a distribution and/or a cache behavior. For **Event type**, keep the default value.
15. Choose **Add association**.

    The **Associated distributions** table shows the associated distribution.
16. Wait a few minutes for the associated distribution to finish deploying. To check the distribution's status, select the distribution in the **Associated distributions** table and then choose **View distribution**.

    When the distribution's status is **Deployed**, you're ready to verify that the function works.

Verifying the function

To see your function in action and verify that it works, go to your distribution's domain name (for example, `https://d111111abcdef8.cloudfront.net`) in a web browser. The function returns a redirect to the browser, so the browser automatically goes to `https://aws.amazon.com/cloudfront/`.

If you send a request to your distribution's domain name using a tool like **curl**, you see the redirect response (`302 Found`) and the custom response header added by the function, as emphasized in the following example.

```
curl -v https://d111111abcdef8.cloudfront.net/
> GET / HTTP/1.1
> Host: d111111abcdef8.cloudfront.net
> User-Agent: curl/7.64.1
> Accept: */*
>
< HTTP/1.1 302 Found
< Server: CloudFront
< Date: Tue, 16 Mar 2021 18:50:48 GMT
< Content-Length: 0
< Connection: keep-alive
< Location: https://aws.amazon.com/cloudfront/
< Cloudfront-Functions: generated-by-CloudFront-Functions
< X-Cache: FunctionGeneratedResponse from cloudfront
< Via: 1.1 3035b31bddaf14eded329f8d22cf188c.cloudfront.net (CloudFront)
< X-Amz-Cf-Pop: PHX50-C2
< X-Amz-Cf-Id: ULZdIz6j43uGBlXyob_JctF9x7CCbwpNniiMlmNbmwzH1YWP9FsEHg==
```

Tutorial: Creating a function that includes key values

This tutorial shows you how to include key values with CloudFront function. Key values are part of a key value pair. You include the name (from the key value pair) in the function code. When the function runs, CloudFront replaces the name with the value.

key value pairs are variables that are stored in a key value store. When you use a key in your function (instead of hard-coded values), your function is more flexible. You can change the value of the key without having to deploy code changes. Key value pairs can also reduce the size of your function. For more information about key value pairs and key value stores, see ???.

Prerequisites

We assume that you are familiar with CloudFront Functions. If you are new to both functions and the key value store, you should first follow the tutorial in the section called "Tutorial: A simple function".

Set up the key value store

Step 1: Create the key value store

1. Plan the key value pairs you want to include in the function. Make a note of key names.

   Keep in mind that all the key value pairs you want to use in a function must be in a single key value store.
2. Decide about the order of work. There are two ways to proceed:
   - Create a key value store, and add key value pairs to the store. Then create (or modify) the function and incorporate the key names.
   - Or, create (or modify) the function and incorporate the key names you want to use. Then create a key value store, and add the key value pairs.

   This tutorial assumes that you are extending the function from the functions tutorial. It also assumes that you want to create the key value store first.
3. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
4. In the left navigation bar, choose **Functions**. On the Functions page, choose the **KeyValueStores** tab.
5. Choose **Create KeyValueStore** and complete the fields as follows:
   - Enter a name and optional description for the store.
   - Leave **S3 URI** blank because this tutorial demonstrates how to enter the key value pairs manually.
6. Choose the **Create** button. The details page for the new key value store appears. This page includes a **Key value pairs** section that is currently empty.

Step 2: Add key value pairs to the store

1. In the **Key value pairs** section, choose the **Add key value pairs** button. Choose **Add pair** and enter a name and value.

2. Choose the **Add pair** button to add another pair.
3. When you have finished, choose **Save** changes to save all the pairs in the store. On the confirmation dialog that appears, choose **Done**.

You now have a store that contains a group of key value pairs.

Set up in the function

Step 3: Associate the key value store with the function

You have now created the key value store. And you have created or modified a function that includes the key names from the key value store. You can now associate the key value store and the function. You create that association from within the function.

1. In the left navigation bar, choose **Functions**. The **Functions** tab appears on top, by default.
2. In the **Associated KeyValueStore** section, choose **Associate Existing KeyValueStore**. Select the key value store and choose the **Associate KeyValueStore** button. Note that you can associate only one key value store with each function.

Step 4: Test and publish the function code

1. You should always test the function code every time you modify it, including when you do the following:
   - Associate a key value store with the function.
   - Modify the function and its key value store to include a new key value pair.
   - Change the value of a key value pair.

   For information about how to test a function, see the section called “Testing functions”. Make sure that you choose to test the function in the `DEVELOPMENT` stage.
2. Publish the function when you are ready to use the function (with the new or revised key value pairs) in a `LIVE` environment.

   When you publish, CloudFront copies the version of the function from the `DEVELOPMENT` stage over to the live stage. The function has the new code and is associated with the key value store. (There is no need to perform the association again, in the live stage.)

   For information about how to publish the function, see the section called “Publishing functions”.

Writing function code

With CloudFront Functions in Amazon CloudFront, you can write lightweight functions in JavaScript for high-scale, latency-sensitive CDN customizations. Your function code can manipulate the requests and responses that flow through CloudFront, perform basic authentication and authorization, generate HTTP responses at the edge, and more.

The following topics can help you write function code for CloudFront Functions.

Topics

- Determine the purpose of your function
- CloudFront Functions event structure
- JavaScript runtime features for CloudFront Functions
- Helper methods for key value stores
- Example code for CloudFront Functions

Determine the purpose of your function

Before you write your function code, determine the purpose of your function. Most functions in CloudFront Functions have one of the following purposes. For more information, see the topic that corresponds to your function's purpose.

Regardless of your function's purpose, the `handler` is the entry point for any function. It takes a single argument called `event`, which is passed to the function by CloudFront. The `event` is a JSON object that contains a representation of the HTTP request (and the response, if your function modifies the HTTP response). For more information about the structure of the `event` object, see CloudFront Functions event structure.

For more information about restrictions that apply to CloudFront Functions and Lambda@Edge, see Restrictions on edge functions.

Topics

- Modify the HTTP request in a viewer request event type
- Generate an HTTP response in a viewer request event type
- Modify the HTTP response in a viewer response event type

Modify the HTTP request in a viewer request event type

Your function can modify the HTTP request that CloudFront receives from the viewer (client), and return the modified request to CloudFront for continued processing. For example, your function code might normalize the cache key or modify request headers.

When you create a function that modifies the HTTP request, make sure to choose the *viewer request* event type. This means that the function runs each time that CloudFront receives a request from a viewer, before checking to see whether the requested object is in the CloudFront cache.

The following pseudocode shows the structure of a function that modifies the HTTP request.

```
function handler(event) {
    var request = event.request;

    // Modify the request object here.

    return request;
}
```

The function returns the modified `request` object to CloudFront. CloudFront continues processing the returned request by checking the CloudFront cache for a cache hit, and sending the request to the origin if necessary.

For more information about the structure of the `event` and `request` objects, see Event structure.

Generate an HTTP response in a viewer request event type

Your function can generate an HTTP response at the edge and return it directly to the viewer (client) without checking for a cached response or any further processing by CloudFront. For example, your function code might redirect the request to a new URL, or check for authorization and return a `401` or `403` response to unauthorized requests.

When you create a function that generates an HTTP response, make sure to choose the *viewer request* event type. This means that the function runs each time CloudFront receives a request from a viewer, before CloudFront does any further processing of the request.

The following pseudocode shows the structure of a function that generates an HTTP response.

```
function handler(event) {
    var request = event.request;
```

```
    var response = ...; // Create the response object here,
                        // using the request properties if needed.

    return response;
}
```

The function returns a `response` object to CloudFront, which CloudFront immediately returns to the viewer without checking the CloudFront cache or sending a request to the origin.

For more information about the structure of the `event`, `request`, and `response` objects, see Event structure.

Modify the HTTP response in a viewer response event type

Your function can modify the HTTP response before CloudFront sends it to the viewer (client), regardless of whether the response comes from the CloudFront cache or the origin. For example, your function code might add or modify response headers, status codes, and body content.

When you create a function that modifies the HTTP response, make sure to choose the *viewer response* event type. This means that the function runs before CloudFront returns a response to the viewer, regardless of whether the response comes from the CloudFront cache or the origin.

The following pseudocode shows the structure of a function that modifies the HTTP response.

```
function handler(event) {
    var request = event.request;
    var response = event.response;

    // Modify the response object here,
    // using the request properties if needed.

    return response;
}
```

The function returns the modified `response` object to CloudFront, which CloudFront immediately returns to the viewer.

For more information about the structure of the `event` and `response` objects, see Event structure.

For more information about writing function code for CloudFront Functions, see Event structure, JavaScript runtime features, and Example code.

CloudFront Functions event structure

CloudFront Functions passes an `event` object to your function code as input when it runs the function. When you test a function, you create the `event` object and pass it to your function. When you create an `event` object for testing a function, you can omit the `distributionDomainName`, `distributionId`, and `requestId` fields in the `context` object. Make sure that the names of headers are lowercase, which is always the case in the `event` object that CloudFront Functions passes to your function in production.

The following shows an overview of the structure of this event object. For more information, see the topics that follow.

```
{
    "version": "1.0",
    "context": {
        <context object>
    },
    "viewer": {
        <viewer object>
    },
    "request": {
        <request object>
    },
    "response": {
        <response object>
    }
}
```

Topics

- Version field
- Context object
- Viewer object
- Request object
- Response object
- Status code and body
- Structure for a query string, header, or cookie
- Example response object

- Example event object

Version field

The `version` field contains a string that specifies the version of the CloudFront Functions event object. The current version is `1.0`.

Context object

The `context` object contains contextual information about the event. It includes the following fields:

`distributionDomainName`

The CloudFront domain name (for example, d111111abcdef8.cloudfront.net) of the distribution that's associated with the event.

`distributionId`

The ID of the distribution (for example, EDFDVBD6EXAMPLE) that's associated with the event.

`eventType`

The event type, either `viewer-request` or `viewer-response`.

`requestId`

A string that uniquely identifies a CloudFront request (and its associated response).

Viewer object

The `viewer` object contains an `ip` field whose value is the IP address of the viewer (client) that sent the request. If the viewer request came through an HTTP proxy or a load balancer, the value is the IP address of the proxy or load balancer.

Request object

The `request` object contains a representation of a viewer-to-CloudFront HTTP request. In the `event` object that's passed to your function, the `request` object represents the actual request that CloudFront received from the viewer.

If your function code returns a `request` object to CloudFront, it must use this same structure.

The `request` object contains the following fields:

`method`

The HTTP method of the request. If your function code returns a `request`, it cannot modify this field. This is the only read-only field in the `request` object.

`uri`

The relative path of the requested object. If your function modifies the `uri` value, note the following:

- The new `uri` value must begin with a forward slash (`/`).
- When a function changes the `uri` value, it changes the object that the viewer is requesting.
- When a function changes the `uri` value, it *doesn't* change the cache behavior for the request or the origin that an origin request is sent to.

`querystring`

An object that represents the query string in the request. If the request doesn't include a query string, the `request` object still includes an empty `querystring` object.

The `querystring` object contains one field for each query string parameter in the request.

`headers`

An object that represents the HTTP headers in the request. If the request contains any `Cookie` headers, those headers are not part of the `headers` object. Cookies are represented separately in the `cookies` object.

The `headers` object contains one field for each header in the request. Header names are converted to lowercase in the event object, and header names must be lowercase when they're added by your function code. When CloudFront Functions converts the event object back into an HTTP request, the first letter of each word in header names is capitalized. Words are separated by a hyphen (`-`). For example, if your function code adds a header named `example-header-name`, CloudFront converts this to `Example-Header-Name` in the HTTP request.

`cookies`

An object that represents the cookies in the request (`Cookie` headers).

The `cookies` object contains one field for each cookie in the request.

For more information about the structure of query strings, headers, and cookies, see Structure for a query string, header, or cookie.

For an example `event` object, see Example event object.

Response object

The `response` object contains a representation of a CloudFront-to-viewer HTTP response. In the `event` object that's passed to your function, the `response` object represents CloudFront's actual response to a viewer request.

If your function code returns a `response` object, it must use this same structure.

The `response` object contains the following fields:

`statusCode`

The HTTP status code of the response. This value is an integer, not a string.

Your function can generate or modify the `statusCode`.

`statusDescription`

The HTTP status description of the response. If your function code generates a response, this field is optional.

`headers`

An object that represents the HTTP headers in the response. If the response contains any `Set-Cookie` headers, those headers are not part of the `headers` object. Cookies are represented separately in the `cookies` object.

The `headers` object contains one field for each header in the response. Header names are converted to lowercase in the event object, and header names must be lowercase when they're added by your function code. When CloudFront Functions converts the event object back into an HTTP response, the first letter of each word in header names is capitalized. Words are separated by a hyphen (`-`). For example, if your function code adds a header named `example-header-name`, CloudFront converts this to `Example-Header-Name` in the HTTP response.

`cookies`

An object that represents the cookies in the response (`Set-Cookie` headers).

The `cookies` object contains one field for each cookie in the response.

`body`

Adding the `body` field is optional, and it will not be present in the `response` object unless you specify it in your function. Your function does not have access to the original body returned by the CloudFront cache or origin. If you don't specify the `body` field in your viewer response function, the original body returned by the CloudFront cache or origin is returned to viewer.

If you want CloudFront to return a custom body to the viewer, specify the body content in the `data` field, and the body encoding in the `encoding` field. You can specify the encoding as plain text (`"encoding": "text"`) or as Base64-encoded content (`"encoding": "base64"`).

As a shortcut, you can also specify the body content directly in the `body` field (`"body": "<specify the body content here>"`). When you do this, omit the `data` and `encoding` fields. CloudFront treats the body as plain text in this case.

`encoding`

The encoding for the `body` content (`data` field). The only valid encodings are `text` and `base64`.

If you specify `encoding` as `base64` but the body is not valid base64, CloudFront returns an error.

`data`

The `body` content.

For more information about modified status codes and body content, see Status code and body.

For more information about the structure of headers and cookies, see Structure for a query string, header, or cookie.

For an example `response` object, see Example response object.

Status code and body

With CloudFront Functions, you can update the viewer response status code, replace the entire response body with a new one, or remove the response body. Some common scenarios for updating the viewer response after evaluating aspects of the response from the CloudFront cache or origin include the following:

- Changing the status to set an HTTP 200 status code and creating static body content to return to the viewer.
- Changing the status to set an HTTP 301 or 302 status code to redirect the user to another website.
- Deciding whether to serve or drop the body of the viewer response.

> **Note**
>
> If the origin returns an HTTP error of 400 and above, the CloudFront Function will not run. For more information see Restrictions on all edge functions.

When you're working with the HTTP response, CloudFront Functions does not have access to the response body. You can replace the body content by setting it to the desired value, or you can remove the body by setting the value to be empty. If you don't update the body field in your function, the original body returned by the CloudFront cache or origin is returned back to viewer.

> **Tip**
>
> When using CloudFront Functions to replace a body, be sure to align the corresponding headers, such as `content-encoding`, `content-type`, or `content-length`, to the new body content.
> For example, if the CloudFront origin or cache returns `content-encoding: gzip` but the viewer response function sets a body that's plain text, the function also needs to change the `content-encoding` and `content-type` headers accordingly.

If your CloudFront Function is configured to return an HTTP error of 400 or above, your viewer will not see a custom error page that you have specified for the same status code.

Structure for a query string, header, or cookie

Query strings, headers, and cookies share the same structure. Query strings can appear in requests. Headers appear in requests and responses. Cookies appear in requests and responses.

Each query string, header, or cookie is a unique field within the parent `querystring`, `headers`, or `cookies` object. The field name is the name of the query string, header, or cookie. Each field contains a `value` property with the value of the query string, header, or cookie.

Topics

- Query strings values or query string objects
- Special considerations for headers
- Duplicate query strings, headers, and cookies (multiValue array)
- Cookie attributes

Query strings values or query string objects

A function can return a query string value in addition to a query string object. The query string value can be used to arrange the query string parameters in any custom order. For example, to modify a query string in your function code, use code like the following:

```
var request = event.request;
request.querystring =
 'ID=42&Exp=1619740800&TTL=1440&NoValue=&querymv=val1&querymv=val2,val3';
```

Special considerations for headers

For headers only, the header names are converted to lowercase in the event object, and header names must be lowercase when they're added by your function code. When CloudFront Functions converts the event object back into an HTTP request or response, the first letter of each word in header names is capitalized. Words are separated by a hyphen (-). For example, if your function code adds a header named `example-header-name`, CloudFront converts this to `Example-Header-Name` in the HTTP request or response.

For example, consider the following `Host` header in an HTTP request:

```
Host: video.example.com
```

This header is represented as follows in the `request` object:

```
"headers": {
    "host": {
        "value": "video.example.com"
    }
}
```

To access the `Host` header in your function code, use code like the following:

```
var request = event.request;
var host = request.headers.host.value;
```

To add or modify a header in your function code, use code like the following (this code adds a header named `X-Custom-Header` with the value `example value`):

```
var request = event.request;
request.headers['x-custom-header'] = {value: 'example value'};
```

Duplicate query strings, headers, and cookies (`multiValue` array)

An HTTP request or response can contain more than one query string, header, or cookie with the same name. In this case, the duplicate query strings, headers, or cookies are collapsed into one field in the `request` or `response` object, but this field contains an extra property named `multiValue`. The `multiValue` property contains an array with the values of each of the duplicate query strings, headers, or cookies.

For example, consider an HTTP request with the following `Accept` headers:

```
Accept: application/json
Accept: application/xml
Accept: text/html
```

These headers are represented as follows in the `request` object:

```
"headers": {
    "accept": {
        "value": "application/json",
        "multiValue": [
            {
                "value": "application/json"
            },
            {
                "value": "application/xml"
            },
            {
                "value": "text/html"
            }
        ]
    }
```

```
}
```

Note that the first header value (in this case, `application/json`) is repeated in both the `value` and `multiValue` properties. This allows you to access *all* the values by looping through the `multiValue` array.

If your function code modifies a query string, header, or cookie that has a `multiValue` array, CloudFront Functions uses the following rules to apply the changes:

1. If the `multiValue` array exists and has any modification, then that modification is applied. The first element in the `value` property is ignored.
2. Otherwise, any modification to the `value` property is applied, and subsequent values (if they exist) remain unchanged.

The `multiValue` property is used only when the HTTP request or response contains duplicate query strings, headers, or cookies with the same name, as shown in the preceding example. However, if there are multiple values in a single query string, header, or cookie, the `multiValue` property is not used.

For example, consider a request with one `Accept` header that contains three values, as in the following example:

```
Accept: application/json, application/xml, text/html
```

This header is represented as follows in the `request` object:

```
"headers": {
    "accept": {
        "value": "application/json, application/xml, text/html"
    }
}
```

Cookie attributes

In a `Set-Cookie` header in an HTTP response, the header contains the name–value pair for the cookie and optionally a set of attributes separated by semicolons. For example:

```
Set-Cookie: cookie1=val1; Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
 07:28:00 GMT
```

In the `response` object, these attributes are represented in the `attributes` property of the cookie field. For example, the preceding `Set-Cookie` header is represented as follows:

```
"cookie1": {
    "value": "val1",
    "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
 07:28:00 GMT"
}
```

Example response object

The following example shows a `response` object — the output of a viewer response function — in which the body has been replaced by a viewer response function.

```
{
  "response": {
    "statusCode": 200,
    "statusDescription": "OK",
    "headers": {
      "date": {
        "value": "Mon, 04 Apr 2021 18:57:56 GMT"
      },
      "server": {
        "value": "gunicorn/19.9.0"
      },
      "access-control-allow-origin": {
        "value": "*"
      },
      "access-control-allow-credentials": {
        "value": "true"
      },
      "content-type": {
        "value": "text/html"
      },
      "content-length": {
        "value": "86"
      }
    },
    "cookies": {
      "ID": {
```

```
        "value": "id1234",
        "attributes": "Expires=Wed, 05 Apr 2021 07:28:00 GMT"
      },
      "Cookie1": {
        "value": "val1",
        "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
 07:28:00 GMT",
        "multiValue": [
          {
            "value": "val1",
            "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
 07:28:00 GMT"
          },
          {
            "value": "val2",
            "attributes": "Path=/cat; Domain=example.com; Expires=Wed, 10 Jan 2021
 07:28:00 GMT"
          }
        ]
      }
    },

    // Adding the body field is optional and it will not be present in the response
 object
    // unless you specify it in your function.
    // Your function does not have access to the original body returned by the
 CloudFront
    // cache or origin.
    // If you don't specify the body field in your viewer response function, the
 original
    // body returned by the CloudFront cache or origin is returned to viewer.

     "body": {
      "encoding": "text",
      "data": "<!DOCTYPE html><html><body><p>Here is your custom content.</p></body></
html>"
    }
  }
}
```

Example event object

The following example shows a complete `event` object.

Note

The `event` object is the input to your function. Your function returns only the `request` or `response` object, not the complete `event` object.

```
{
    "version": "1.0",
    "context": {
        "distributionDomainName": "d111111abcdef8.cloudfront.net",
        "distributionId": "EDFDVBD6EXAMPLE",
        "eventType": "viewer-response",
        "requestId": "EXAMPLEntjQpEXAMPLE_SG5Z-EXAMPLEPmPfEXAMPLEu3EqEXAMPLE=="
    },
    "viewer": {"ip": "198.51.100.11"},
    "request": {
        "method": "GET",
        "uri": "/media/index.mpd",
        "querystring": {
            "ID": {"value": "42"},
            "Exp": {"value": "1619740800"},
            "TTL": {"value": "1440"},
            "NoValue": {"value": ""},
            "querymv": {
                "value": "val1",
                "multiValue": [
                    {"value": "val1"},
                    {"value": "val2,val3"}
                ]
            }
        },
        "headers": {
            "host": {"value": "video.example.com"},
            "user-agent": {"value": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0)
 Gecko/20100101 Firefox/83.0"},
            "accept": {
                "value": "application/json",
                "multiValue": [
                    {"value": "application/json"},
                    {"value": "application/xml"},
                    {"value": "text/html"}
                ]
```

```
            },
            "accept-language": {"value": "en-GB,en;q=0.5"},
            "accept-encoding": {"value": "gzip, deflate, br"},
            "origin": {"value": "https://website.example.com"},
            "referer": {"value": "https://website.example.com/videos/12345678?
action=play"},
            "cloudfront-viewer-country": {"value": "GB"}
        },
        "cookies": {
            "Cookie1": {"value": "value1"},
            "Cookie2": {"value": "value2"},
            "cookie_consent": {"value": "true"},
            "cookiemv": {
                "value": "value3",
                "multiValue": [
                    {"value": "value3"},
                    {"value": "value4"}
                ]
            }
        }
    },
    "response": {
        "statusCode": 200,
        "statusDescription": "OK",
        "headers": {
            "date": {"value": "Mon, 04 Apr 2021 18:57:56 GMT"},
            "server": {"value": "gunicorn/19.9.0"},
            "access-control-allow-origin": {"value": "*"},
            "access-control-allow-credentials": {"value": "true"},
            "content-type": {"value": "application/json"},
            "content-length": {"value": "701"}
        },
        "cookies": {
            "ID": {
                "value": "id1234",
                "attributes": "Expires=Wed, 05 Apr 2021 07:28:00 GMT"
            },
            "Cookie1": {
                "value": "val1",
                "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr
 2021 07:28:00 GMT",
                "multiValue": [
                    {
                        "value": "val1",
```

```
                        "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed,
 05 Apr 2021 07:28:00 GMT"
                    },
                    {
                        "value": "val2",
                        "attributes": "Path=/cat; Domain=example.com; Expires=Wed, 10
 Jan 2021 07:28:00 GMT"
                    }
                ]
            }
        }
    }
}
```

JavaScript runtime features for CloudFront Functions

The Amazon CloudFront Functions JavaScript runtime environment is compliant with ECMAScript (ES) version 5.1 and also supports some features of ES versions 6 through 12.

We recommend that you use runtime 2.0 for the most up to date features. Note that runtime 2.0 has the following changes compared to 1.0:

- Buffer module methods are available.
- The following non-standard string prototype methods are not available:
  - `String.prototype.bytesFrom()`
  - `String.prototype.fromBytes()`
  - `String.prototype.fromUTF8()`
  - `String.prototype.toBytes()`
  - `String.prototype.toUTF8()`
- The cryptographic module has the following changes:
  - `hash.digest()` - Return type is changed to `Buffer` if no encoding is provided
  - `hmac.digest()` - Return type is changed to `Buffer` if no encoding is provided
- Additional new features are noted in JavaScript runtime 2.0 features for CloudFront Functions.

Topics

- JavaScript runtime 1.0 features for CloudFront Functions
- JavaScript runtime 2.0 features for CloudFront Functions

JavaScript runtime 1.0 features for CloudFront Functions

The CloudFront Functions JavaScript runtime environment is compliant with ECMAScript (ES) version 5.1 and also supports some features of ES versions 6 through 9. It also provides some nonstandard methods that are not part of the ES specifications. The following topics list all the supported language features.

Topics

- Core features
- Primitive objects
- Built-in objects
- Error types
- Globals
- Built-in modules
- Restricted features

Core features

The following core features of ES are supported.

Types

All ES 5.1 types are supported. This includes Boolean values, numbers, strings, objects, arrays, functions, function constructors, and regular expressions.

Operators

All ES 5.1 operators are supported.

The ES 7 exponentiation operator (**) is supported.

Statements

> **Note**
>
> The `const` and `let` statements are not supported.

The following ES 5.1 statements are supported:

- `break`
- `catch`
- `continue`
- `do-while`
- `else`
- `finally`
- `for`
- `for-in`
- `if`
- `return`
- `switch`
- `throw`
- `try`
- `var`
- `while`
- Labeled statements

Literals

ES 6 template literals are supported: multiline strings, expression interpolation, and nesting templates.

Functions

All ES 5.1 function features are supported.

ES 6 arrow functions are supported, and ES 6 rest parameter syntax is supported.

Unicode

Source text and string literals can contain Unicode-encoded characters. Unicode code point escape sequences of six characters (for example, `\uXXXX`) are also supported.

Strict mode

Functions operate in strict mode by default, so you don't need to add a `use strict` statement in your function code. This cannot be changed.

Primitive objects

The following primitive objects of ES are supported.

Object

The following ES 5.1 methods on objects are supported:

- `create` (without properties list)
- `defineProperties`
- `defineProperty`
- `freeze`
- `getOwnPropertyDescriptor`
- `getOwnPropertyNames`
- `getPrototypeOf`
- `hasOwnProperty`
- `isExtensible`
- `isFrozen`
- `prototype.isPrototypeOf`
- `isSealed`
- `keys`
- `preventExtensions`
- `prototype.propertyIsEnumerable`
- `seal`
- `prototype.toString`
- `prototype.valueOf`

The following ES 6 methods on objects are supported:

- `assign`
- `is`
- `prototype.setPrototypeOf`

The following ES 8 methods on objects are supported:

- `entries`

- values

String

The following ES 5.1 methods on strings are supported:

- fromCharCode
- prototype.charAt
- prototype.concat
- prototype.indexOf
- prototype.lastIndexOf
- prototype.match
- prototype.replace
- prototype.search
- prototype.slice
- prototype.split
- prototype.substr
- prototype.substring
- prototype.toLowerCase
- prototype.trim
- prototype.toUpperCase

The following ES 6 methods on strings are supported:

- fromCodePoint
- prototype.codePointAt
- prototype.endsWith
- prototype.includes
- prototype.repeat
- prototype.startsWith

The following ES 8 methods on strings are supported:

- prototype.padStart
- prototype.padEnd

The following ES 9 methods on strings are supported:

- `prototype.trimStart`
- `prototype.trimEnd`

The following nonstandard methods on strings are supported:

- `prototype.bytesFrom(`*`array | string, encoding`*`)`

  Creates a byte string from an array of octets or an encoded string. The string encoding options are `hex`, `base64`, and `base64url`.

- `prototype.fromBytes(`*`start`*`[, `*`end`*`])`

  Creates a Unicode string from a byte string where each byte is replaced with the corresponding Unicode code point.

- `prototype.fromUTF8(`*`start`*`[, `*`end`*`])`

  Creates a Unicode string from a UTF-8 encoded byte string. If the encoding is incorrect, it returns `null`.

- `prototype.toBytes(start[, end])`

  Creates a byte string from a Unicode string. All characters must be in the [0,255] range. If not, it returns `null`.

- `prototype.toUTF8(start[, `*`end`*`])`

  Creates a UTF-8 encoded byte string from a Unicode string.

Number

All ES 5.1 methods on numbers are supported.

The following ES 6 methods on numbers are supported:

- `isFinite`
- `isInteger`
- `isNaN`
- `isSafeInteger`
- `parseFloat`
- `parseInt`
- `prototype.toExponential`

- `prototype.toFixed`
- `prototype.toPrecision`
- `EPSILON`
- `MAX_SAFE_INTEGER`
- `MAX_VALUE`
- `MIN_SAFE_INTEGER`
- `MIN_VALUE`
- `NEGATIVE_INFINITY`
- `NaN`
- `POSITIVE_INFINITY`

Built-in objects

The following built-in objects of ES are supported.

Math

All ES 5.1 math methods are supported.

> **Note**
>
> In the CloudFront Functions runtime environment, the `Math.random()` implementation uses OpenBSD `arc4random` seeded with the timestamp of when the function runs.

The following ES 6 math methods are supported:

- `acosh`
- `asinh`
- `atanh`
- `cbrt`
- `clz32`
- `cosh`
- `expm1`

- `fround`
- `hypot`
- `imul`
- `log10`
- `log1p`
- `log2`
- `sign`
- `sinh`
- `tanh`
- `trunc`
- `E`
- `LN10`
- `LN2`
- `LOG10E`
- `LOG2E`
- `PI`
- `SQRT1_2`
- `SQRT2`

Date

All ES 5.1 `Date` features are supported.

> **Note**
>
> For security reasons, `Date` always returns the same value—the function's start time—during the lifetime of a single function run. For more information, see Restricted features.

Function

The `apply`, `bind`, and `call` methods are supported.

Function constructors are not supported.

Regular expressions

All ES 5.1 regular expression features are supported. The regular expression language is Perl compatible. ES 9 named capture groups are supported.

JSON

All ES 5.1 JSON features are supported, including `parse` and `stringify`.

Array

The following ES 5.1 methods on arrays are supported:

- `isArray`
- `prototype.concat`
- `prototype.every`
- `prototype.filter`
- `prototype.forEach`
- `prototype.indexOf`
- `prototype.join`
- `prototype.lastIndexOf`
- `prototype.map`
- `prototype.pop`
- `prototype.push`
- `prototype.reduce`
- `prototype.reduceRight`
- `prototype.reverse`
- `prototype.shift`
- `prototype.slice`
- `prototype.some`
- `prototype.sort`
- `prototype.splice`
- `prototype.unshift`

The following ES 6 methods on arrays are supported:

- of
- prototype.copyWithin
- prototype.fill
- prototype.find
- prototype.findIndex

The following ES 7 methods on arrays are supported:

- prototype.includes

Typed arrays

The following ES 6 typed arrays are supported:

- Int8Array
- Uint8Array
- Uint8ClampedArray
- Int16Array
- Uint16Array
- Int32Array
- Uint32Array
- Float32Array
- Float64Array
- prototype.copyWithin
- prototype.fill
- prototype.join
- prototype.set
- prototype.slice
- prototype.subarray
- prototype.toString

ArrayBuffer

The following methods on ArrayBuffer are supported:

- prototype.isView
- prototype.slice

Promise

The following methods on promises are supported:

- `reject`
- `resolve`
- `prototype.catch`
- `prototype.finally`
- `prototype.then`

Crypto

The cryptographic module provides standard hashing and hash-based message authentication code (HMAC) helpers. You can load the module using `require('crypto')`. The module exposes the following methods that behave exactly as their Node.js counterparts:

- `createHash(`*`algorithm`*`)`
- `hash.update(`*`data`*`)`
- `hash.digest([`*`encoding`*`])`
- `createHmac(`*`algorithm, secret key`*`)`
- `hmac.update(`*`data`*`)`
- `hmac.digest([`*`encoding`*`])`

For more information, see Crypto (hash and HMAC) in the built-in modules section.

Console

This is a helper object for debugging. It only supports the `log()` method, to record log messages.

> **Note**
>
> CloudFront Functions doesn't support comma syntax, such as `console.log('a', 'b')`. Instead, use the `console.log('a' + ' ' + 'b')` format.

Error types

The following error objects are supported:

- `Error`
- `EvalError`
- `InternalError`
- `MemoryError`
- `RangeError`
- `ReferenceError`
- `SyntaxError`
- `TypeError`
- `URIError`

Globals

The `globalThis` object is supported.

The following ES 5.1 global functions are supported:

- `decodeURI`
- `decodeURIComponent`
- `encodeURI`
- `encodeURIComponent`
- `isFinite`
- `isNaN`
- `parseFloat`
- `parseInt`

The following global constants are supported:

- `NaN`
- `Infinity`
- `undefined`

Built-in modules

The following built-in modules are supported.

Modules

- Crypto (hash and HMAC)
- Query string

Crypto (hash and HMAC)

The cryptographic module (`crypto`) provides standard hashing and hash-based message authentication code (HMAC) helpers. You can load the module using `require('crypto')`. The module provides the following methods that behave exactly as their Node.js counterparts.

Hashing methods

`crypto.createHash(`*`algorithm`*`)`

Creates and returns a hash object that you can use to generate hash digests using the given algorithm: `md5`, `sha1`, or `sha256`.

`hash.update(`*`data`*`)`

Updates the hash content with the given `data`.

`hash.digest([`*`encoding`*`])`

Calculates the digest of all of the data passed using `hash.update()`. The encoding can be `hex`, `base64`, or `base64url`.

HMAC methods

`crypto.createHmac(`*`algorithm, secret key`*`)`

Creates and returns an HMAC object that uses the given `algorithm` and `secret key`. The algorithm can be `md5`, `sha1`, or `sha256`.

`hmac.update(`*`data`*`)`

Updates the HMAC content with the given `data`.

`hmac.digest([`*`encoding`*`])`

Calculates the digest of all of the data passed using `hmac.update()`. The encoding can be `hex`, `base64`, or `base64url`.

Query string

> **Note**
>
> The CloudFront Functions event object automatically parses URL query strings for you. That means that in most cases you don't need to use this module.

The query string module (`querystring`) provides methods for parsing and formatting URL query strings. You can load the module using `require('querystring')`. The module provides the following methods.

`querystring.escape(`*`string`*`)`

URL-encodes the given `string`, returning an escaped query string. The method is used by `querystring.stringify()` and should not be used directly.

`querystring.parse(`*`string`*`[, `*`separator`*`[, `*`equal`*`[, `*`options`*`]]])`

Parses a query string (`string`) and returns an object.

The `separator` parameter is a substring for delimiting key and value pairs in the query string. By default it is &.

The `equal` parameter is a substring for delimiting keys and values in the query string. By default it is =.

The `options` parameter is an object with the following keys:

`decodeURIComponent` *`function`*

A function to decode percent-encoded characters in the query string. By default it is `querystring.unescape()`.

`maxKeys` *`number`*

The maximum number of keys to parse. By default it is `1000`. Use a value of `0` to remove the limitations for counting keys.

By default, percent-encoded characters within the query string are assumed to use the UTF-8 encoding. Invalid UTF-8 sequences are replaced with the `U+FFFD` replacement character.

For example, for the following query string:

```
'name=value&abc=xyz&abc=123'
```

The return value of `querystring.parse()` is:

```
{
name: 'value',
abc: ['xyz', '123']
}
```

`querystring.decode()` is an alias for `querystring.parse()`.

`querystring.stringify(`*`object`*`[, `*`separator`*`[, `*`equal`*`[, `*`options`*`]]])`

Serializes an `object` and returns a query string.

The `separator` parameter is a substring for delimiting key and value pairs in the query string. By default it is &.

The `equal` parameter is a substring for delimiting keys and values in the query string. By default it is =.

The `options` parameter is an object with the following keys:

`encodeURIComponent` *`function`*

The function to use for converting URL-unsafe characters to percent-encoding in the query string. By default it is `querystring.escape()`.

By default, characters that require percent-encoding within the query string are encoded as UTF-8. To use a different encoding, specify the `encodeURIComponent` option.

For example, for the following code:

```
querystring.stringify({ name: 'value', abc: ['xyz', '123'], anotherName: '' });
```

The return value is:

```
'name=value&abc=xyz&abc=123&anotherName='
```

`querystring.encode()` is an alias for `querystring.stringify()`.

`querystring.unescape(`*`string`*`)`

Decodes URL percent-encoded characters in the given `string`, returning an unescaped query string. This method is used by `querystring.parse()` and should not be used directly.

Restricted features

The following JavaScript language features are either unsupported or restricted due to security concerns.

Dynamic code evaluation

Dynamic code evaluation is not supported. Both `eval()` and `Function` constructors throw an error if attempted. For example, `const sum = new Function('a', 'b', 'return a + b')` throws an error.

Timers

The `setTimeout()`, `setImmediate()`, and `clearTimeout()` functions are not supported. There is no provision to defer or yield within a function run. Your function must synchronously run to completion.

Date and timestamps

For security reasons, there is no access to high-resolution timers. All `Date` methods to query the current time always return the same value during the lifetime of a single function run. The returned timestamp is the time when the function started running. Consequently, you cannot measure elapsed time in your function.

File system access

There is no file system access. For example, there is no `fs` module for file system access like there is in Node.js.

Network access

There is no support for network calls. For example, XHR, HTTP(S), and socket are not supported.

JavaScript runtime 2.0 features for CloudFront Functions

The CloudFront Functions JavaScript runtime environment is compliant with ECMAScript (ES) version 5.1 and also supports some features of ES versions 6 through 12. It also provides some

nonstandard methods that are not part of the ES specifications. The following topics list all supported features in this runtime.

Topics

- Core features
- Primitive objects
- Built-in objects
- Error types
- Globals
- Built-in modules
- Restricted features

Core features

The following core features of ES are supported.

Types

All ES 5.1 types are supported. This includes boolean values, numbers, strings, objects, arrays, functions, and regular expressions.

Operators

All ES 5.1 operators are supported.

The ES 7 exponentiation operator (**) is supported.

Statements

The following ES 5.1 statements are supported:

- `break`
- `catch`
- `continue`
- `do-while`
- `else`
- `finally`

- `for`
- `for-in`
- `if`
- `label`
- `return`
- `switch`
- `throw`
- `try`
- `var`
- `while`

The following ES 6 statements are supported:

- `async`
- `await`
- `const`
- `let`

> **Note**
>
> `async`, `await`, `const`, and `let` are new in JavaScript runtime 2.0.

Literals

ES 6 template literals are supported: multiline strings, expression interpolation, and nesting templates.

Functions

All ES 5.1 function features are supported.

ES 6 arrow functions are supported, and ES 6 rest parameter syntax is supported.

Unicode

Source text and string literals can contain Unicode-encoded characters. Unicode code point escape sequences of six characters (for example, `\uXXXX`) are also supported.

Strict mode

Functions operate in strict mode by default, so you don't need to add a `use strict` statement in your function code. This cannot be changed.

Primitive objects

The following primitive objects of ES are supported.

Object

The following ES 5.1 methods on objects are supported:

- `Object.create()` (without properties list)
- `Object.defineProperties()`
- `Object.defineProperty()`
- `Object.freeze()`
- `Object.getOwnPropertyDescriptor()`
- `Object.getOwnPropertyDescriptors()`
- `Object.getOwnPropertyNames()`
- `Object.getPrototypeOf()`
- `Object.isExtensible()`
- `Object.isFrozen()`
- `Object.isSealed()`
- `Object.keys()`
- `Object.preventExtensions()`
- `Object.seal()`

The following ES 6 methods on objects are supported:

- `Object.assign()`

The following ES 8 methods on objects are supported:

- `Object.entries()`
- `Object.values()`

The following ES 5.1 prototype methods on objects are supported:

- `Object.prototype.hasOwnProperty()`
- `Object.prototype.isPrototypeOf()`
- `Object.prototype.propertyIsEnumerable()`
- `Object.prototype.toString()`
- `Object.prototype.valueOf()`

The following ES 6 prototype methods on objects are supported:

- `Object.prototype.is()`
- `Object.prototype.setPrototypeOf()`

String

The following ES 5.1 methods on strings are supported:

- `String.fromCharCode()`

The following ES 6 methods on strings are supported:

- `String.fromCodePoint()`

The following ES 5.1 prototype methods on strings are supported:

- `String.prototype.charAt()`
- `String.prototype.concat()`
- `String.prototype.indexOf()`
- `String.prototype.lastIndexOf()`
- `String.prototype.match()`
- `String.prototype.replace()`
- `String.prototype.search()`
- `String.prototype.slice()`
- `String.prototype.split()`
- `String.prototype.substr()`
- `String.prototype.substring()`
- `String.prototype.toLowerCase()`
- `String.prototype.trim()`

- `String.prototype.toUpperCase()`

The following ES 6 prototype methods on strings are supported:

- `String.prototype.codePointAt()`
- `String.prototype.endsWith()`
- `String.prototype.includes()`
- `String.prototype.repeat()`
- `String.prototype.startsWith()`

The following ES 8 prototype methods on strings are supported:

- `String.prototype.padStart()`
- `String.prototype.padEnd()`

The following ES 9 prototype methods on strings are supported:

- `String.prototype.trimStart()`
- `String.prototype.trimEnd()`

The following ES 12 prototype methods on strings are supported:

- `String.prototype.replaceAll()`

> **Note**
>
> `String.prototype.replaceAll()` is new in JavaScript runtime 2.0.

Number

ALL ES 5 numbers are supported.

The following ES 6 properties on numbers are supported:

- `Number.EPSILON`
- `Number.MAX_SAFE_INTEGER`
- `Number.MIN_SAFE_INTEGER`
- `Number.MAX_VALUE`
- `Number.MIN_VALUE`
- `Number.NaN`

- `Number.NEGATIVE_INFINITY`
- `Number.POSITIVE_INFINITY`

The following ES 6 methods on numbers are supported:

- `Number.isFinite()`
- `Number.isInteger()`
- `Number.isNaN()`
- `Number.isSafeInteger()`
- `Number.parseInt()`
- `Number.parseFloat()`

The following ES 5.1 prototype methods on numbers are supported:

- `Number.prototype.toExponential()`
- `Number.prototype.toFixed()`
- `Number.prototype.toPrecision()`

ES 12 numeric separators are supported.

> **Note**
>
> ES 12 numeric separators are new in JavaScript runtime 2.0.

Built-in objects

The following built-in objects of ES are supported.

Math

All ES 5.1 math methods are supported.

The following ES 6 math properties are supported:

- `Math.E`
- `Math.LN10`
- `Math.LN2`
- `Math.LOG10E`

- `Math.LOG2E`
- `Math.PI`
- `Math.SQRT1_2`
- `Math.SQRT2`

The following ES 6 math methods are supported:

- `Math.abs()`
- `Math.acos()`
- `Math.acosh()`
- `Math.asin()`
- `Math.asinh()`
- `Math.atan()`
- `Math.atan2()`
- `Math.atanh()`
- `Math.cbrt()`
- `Math.ceil()`
- `Math.clz32()`
- `Math.cos()`
- `Math.cosh()`
- `Math.exp()`
- `Math.expm1()`
- `Math.floor()`
- `Math.fround()`
- `Math.hypot()`
- `Math.imul()`
- `Math.log()`
- `Math.log1p()`
- `Math.log2()`
- `Math.log10()`
- `Math.max()`

- `Math.min()`
- `Math.pow()`
- `Math.random()`
- `Math.round()`
- `Math.sign()`
- `Math.sinh()`
- `Math.sin()`
- `Math.sqrt()`
- `Math.tan()`
- `Math.tanh()`
- `Math.trunc()`

Date

All ES 5.1 `Date` features are supported.

> **Note**
>
> For security reasons, `Date` always returns the same value—the function's start time—during the lifetime of a single function run. For more information, see Restricted features.

Function

The following ES 5.1 prototype methods are supported:

- `Function.prototype.apply()`
- `Function.prototype.bind()`
- `Function.prototype.call()`

Function constructors are not supported.

Regular expressions

All ES 5.1 regular expression features are supported. The regular expression language is Perl compatible.

The following ES 5.1 prototype accessor properties are supported:

- `RegExp.prototype.global`
- `RegExp.prototype.ignoreCase`
- `RegExp.protoype.multiline`
- `RegExp.protoype.source`
- `RegExp.prototype.sticky`
- `RegExp.prototype.flags`

> **Note**
>
> `RegExp.prototype.sticky` and `RegExp.prototype.flags` are new in JavaScript runtime 2.0.

The following ES 5.1 prototype methods are supported:

- `RegExp.prototype.exec()`
- `RegExp.prototype.test()`
- `RegExp.prototype.toString()`
- `RegExp.prototype[@@replace]()`
- `RegExp.prototype[@@split]()`

> **Note**
>
> `RegExp.prototype[@@split]()` is new in JavaScript runtime 2.0.

The following ES 5.1 instance properties are supported:

- `lastIndex`

ES 9 named capture groups are supported.

JSON

The following ES 5.1 methods are supported:

- `JSON.parse()`
- `JSON.stringify()`

Array

The following ES 5.1 methods on arrays are supported:

- `Array.isArray()`

The following ES 6 methods on arrays are supported:

- `Array.of()`

The following ES 5.1 prototype methods are supported:

- `Array.prototype.concat()`
- `Array.prototype.every()`
- `Array.prototype.filter()`
- `Array.prototype.forEach()`
- `Array.prototype.indexOf()`
- `Array.prototype.join()`
- `Array.prototype.lastIndexOf()`
- `Array.prototype.map()`
- `Array.prototype.pop()`
- `Array.prototype.push()`
- `Array.prototype.reduce()`
- `Array.prototype.reduceRight()`
- `Array.prototype.reverse()`
- `Array.prototype.shift()`
- `Array.prototype.slice()`
- `Array.prototype.some()`
- `Array.prototype.sort()`
- `Array.prototype.splice()`
- `Array.prototype.unshift()`

The following ES 6 prototype methods are supported

- `Array.prototype.copyWithin()`

- `Array.prototype.fill()`
- `Array.prototype.find()`
- `Array.prototype.findIndex()`

The following ES 7 prototype methods are supported:

- `Array.prototype.includes()`

Typed arrays

The following ES 6 typed array constructors are supported:

- `Float32Array`
- `Float64Array`
- `Int8Array`
- `Int16Array`
- `Int32Array`
- `Uint8Array`
- `Uint8ClampedArray`
- `Uint16Array`
- `Uint32Array`

The following ES 6 methods are supported:

- `TypedArray.from()`
- `TypedArray.of()`

> **Note**
>
> `TypedArray.from()` and `TypedArray.of()` are new in JavaScript runtime 2.0.

The following ES 6 prototype methods are supported:

- `TypedArray.prototype.copyWithin()`
- `TypedArray.prototype.every()`
- `TypedArray.prototype.fill()`
- `TypedArray.prototype.filter()`

- `TypedArray.prototype.find()`
- `TypedArray.prototype.findIndex()`
- `TypedArray.prototype.forEach()`
- `TypedArray.prototype.includes()`
- `TypedArray.prototype.indexOf()`
- `TypedArray.prototype.join()`
- `TypedArray.prototype.lastIndexOf()`
- `TypedArray.prototype.map()`
- `TypedArray.prototype.reduce()`
- `TypedArray.prototype.reduceRight()`
- `TypedArray.prototype.reverse()`
- `TypedArray.prototype.some()`
- `TypedArray.prototype.set()`
- `TypedArray.prototype.slice()`
- `TypedArray.prototype.sort()`
- `TypedArray.prototype.subarray()`
- `TypedArray.prototype.toString()`

> **Note**
>
> `TypedArray.prototype.every()`, `TypedArray.prototype.fill()`, `TypedArray.prototype.filter()`, `TypedArray.prototype.find()`, `TypedArray.prototype.findIndex()`, `TypedArray.prototype.forEach()`, `TypedArray.prototype.includes()`, `TypedArray.prototype.indexOf()`, `TypedArray.prototype.join()`, `TypedArray.prototype.lastIndexOf()`,`TypedArray.prototype.map()`, `TypedArray.prototype.reduce()`, `TypedArray.prototype.reduceRight()`, `TypedArray.prototype.reverse()`, and `TypedArray.prototype.some()` are new in JavaScript runtime 2.0.

ArrayBuffer

The following ES 6 methods on ArrayBuffer are supported:

- `isView()`

The following ES 6 prototype methods on ArrayBuffer are supported:

- `ArrayBuffer.prototype.slice()`

Promise

The following ES 6 methods on promises are supported:

- `Promise.all()`
- `Promise.allSettled()`
- `Promise.any()`
- `Promise.reject()`
- `Promise.resolve()`
- `Promise.race()`

> **Note**
>
> `Promise.all()`, `Promise.allSettled()`, `Promise.any()`, and `Promise.race()` are new in JavaScript runtime 2.0.

The following ES 6 prototype methods on promises are supported:

- `Promise.prototype.catch()`
- `Promise.prototype.finally()`
- `Promise.prototype.then()`

DataView

The following ES 6 prototype methods are supported:

- `DataView.prototype.getFloat32()`
- `DataView.prototype.getFloat64()`
- `DataView.prototype.getInt16()`
- `DataView.prototype.getInt32()`
- `DataView.prototype.getInt8()`
- `DataView.prototype.getUint16()`

- `DataView.prototype.getUint32()`
- `DataView.prototype.getUint8()`
- `DataView.prototype.setFloat32()`
- `DataView.prototype.setFloat64()`
- `DataView.prototype.setInt16()`
- `DataView.prototype.setInt32()`
- `DataView.prototype.setInt8()`
- `DataView.prototype.setUint16()`
- `DataView.prototype.setUint32()`
- `DataView.prototype.setUint8()`

> **Note**
>
> All Dataview ES 6 prototype methods are new in JavaScript runtime 2.0.

Symbol

The following ES 6 methods are supported:

- `Symbol.for()`
- `Symbol.keyfor()`

> **Note**
>
> All Symbol ES 6 methods are new in JavaScript runtime 2.0.

Text Decoder

The following prototype methods are supported:

- `TextDecoder.prototype.decode()`

The following prototype accessor properties are supported:

- `TextDecoder.prototype.encoding`
- `TextDecoder.prototype.fatal`
- `TextDecoder.prototype.ignoreBOM`

Text Encoder

The following prototype methods are supported:

- `TextEncoder.prototype.encode()`
- `TextEncoder.prototype.encodeInto()`

Error types

The following error objects are supported:

- `Error`
- `EvalError`
- `InternalError`
- `RangeError`
- `ReferenceError`
- `SyntaxError`
- `TypeError`
- `URIError`

Globals

The `globalThis` object is supported.

The following ES 5.1 global functions are supported:

- `decodeURI()`
- `decodeURIComponent()`
- `encodeURI()`
- `encodeURIComponent()`
- `isFinite()`
- `isNaN()`
- `parseFloat()`
- `parseInt()`

The following ES 6 global functions are supported:

- `atob()`
- `btoa()`

> **Note**
>
> `atob()` and `btoa()` are new in JavaScript runtime 2.0.

The following global constants are supported:

- `NaN`
- `Infinity`
- `undefined`
- `arguments`

Built-in modules

The following built-in modules are supported.

Modules

- Buffer
- Query string
- Crypto

Buffer

The module provides the following methods:

- `Buffer.alloc(size[, fill[, encoding]])`

  Allocate a `Buffer`.

  - `size`: Buffer size. Enter an integer.
  - `fill`: Optional. Enter a string, `Buffer`, Uint8Array, or integer. Default is `0`.
  - `encoding`: Optional. When `fill` is a string, enter one of the following: `utf8`, `hex`, `base64`, `base64url`. Default is `utf8`.
- `Buffer.allocUnsafe(size)`

Allocate a non-initialized Buffer.

- size: Enter an integer.

- Buffer.byteLength(value[, encoding])

  Return the length of a value, in bytes.

  - value: A string, Buffer, TypedArray, Dataview, or Arraybuffer.
  - encoding: Optional. When value is a string, enter one of the following: utf8, hex, base64, base64url. Default is utf8.

- Buffer.compare(buffer1, buffer2)

  Compare two Buffers to help sort arrays. Returns 0 if they're the same, -1 if buffer1 comes first, or 1 if buffer2 comes first.

  - buffer1: Enter a Buffer.
  - buffer2: Enter a different Buffer.

- Buffer.concat(list[, totalLength])

  Concatenate multiple Buffers. Returns 0 if none. Returns up to totalLength.

  - list: Enter a list of Buffers. Note this will be truncated to totalLength.
  - totalLength: Optional. Enter an unsigned integer. Use sum of Buffer instances in list if blank.

- Buffer.from(array)

  Create a Buffer from an array.

  - array: Enter a byte array from 0 to 255.

- Buffer.from(arrayBuffer, byteOffset[, length]))

  Create a view from arrayBuffer, starting at offset byteOffset with length length.

  - arrayBuffer: Enter a Buffer array.
  - byteOffset: Enter an integer.
  - length: Optional. Enter an integer.

- Buffer.from(buffer)

  Create a copy of the Buffer.

  - buffer: Enter a Buffer.

- `Buffer.from(object[, offsetOrEncoding[, length]])`

  Create a `Buffer` from an object. Returns `Buffer.from(object.valueOf(), offsetOrEncoding, length)` if `valueOf()` is not equal to the object.

  - `object`: Enter an object.
  - `offsetOrEncoding`: Optional. Enter an integer or encoding string.
  - `length`: Optional. Enter an integer.
- `Buffer.from(string[, encoding])`

  Create a `Buffer` from a string.

  - `string`: Enter a string.
  - `encoding`: Optional. Enter one of the following: `utf8`, `hex`, `base64`, `base64url`. Default is `utf8`.
- `Buffer.isBuffer(object)`

  Check if `object` is a Buffer. Returns `true` or `false`.

  - `object`: Enter an object.
- `Buffer.isEncoding(encoding)`

  Check if `encoding` is supported. Returns `true` or `false`.

  - `encoding`: Optional. Enter one of the following: `utf8`, `hex`, `base64`, `base64url`. Default is `utf8`.

The module provides the following buffer prototype methods:

- `Buffer.prototype.compare(target[, targetStart[, targetEnd[, sourceStart[, sourceEnd]]]])`

  Compare `Buffer` with target. Returns `0` if they're the same, `1` if `buffer` comes first, or `-1` if `target` comes first.

  - `target`: Enter a `Buffer`.
  - `targetStart`: Optional. Enter an integer. Default is 0.
  - `targetEnd`: Optional. Enter an integer. Default is `target` length.
  - `sourceStart`: Optional. Enter an integer. Default is 0.
  - `sourceEnd`: Optional. Enter an integer. Default is `Buffer` length.

- `Buffer.prototype.copy(target[, targetStart[, sourceStart[, sourceEnd]]])`

  Copy buffer to `target`.

  - `target`: Enter a `Buffer` or `Uint8Array`.
  - `targetStart`: Optional. Enter an integer. Default is 0.
  - `sourceStart`: Optional. Enter an integer. Default is 0.
  - `sourceEnd`: Optional. Enter an integer. Default is `Buffer` length.
- `Buffer.prototype.equals(otherBuffer)`

  Compare `Buffer` to `otherBuffer`. Returns `true` or `false`.

  - `otherBuffer`: Enter a string.
- `Buffer.prototype.fill(value[, offset[, end][, encoding])`

  Fill `Buffer` with `value`.

  - `value`: Enter a string, `Buffer`, or integer.
  - `offset`: Optional. Enter an integer.
  - end: Optional. Enter an integer.
  - `encoding`: Optional. Enter one of the following: `utf8`, `hex`, `base64`, `base64url`. Default is `utf8`.
- `Buffer.prototype.includes(value[, byteOffset][, encoding])`

  Search for `value` in `Buffer`. Returns `true` or `false`.

  - `value`: Enter a string, `Buffer`, `Uint8Array`, or integer.
  - `byteOffset`: Optional. Enter an integer.
  - `encoding`: Optional. Enter one of the following: `utf8`, `hex`, `base64`, `base64url`. Default is `utf8`.
- `Buffer.prototype.indexOf(value[, byteOffset][, encoding])`

  Search for first `value` in `Buffer`. Returns `index` if found; returns `-1` if not found.

  - `value`: Enter a string, `Buffer`, Unit8Array, or integer from 0 to 255.
  - `byteOffset`: Optional. Enter an integer.
  - `encoding`: Optional. Enter one of the following if `value` is a string: `utf8`, `hex`, `base64`, `base64url`. Default is `utf8`.
- `Buffer.prototype.lastIndexOf(value[, byteOffset][, encoding])`

Search for last `value` in `Buffer`. Returns `index` if found; returns `-1` if not found.

- `value`: Enter a string, `Buffer`, Unit8Array, or integer from 0 to 255.
- `byteOffset`: Optional. Enter an integer.
- `encoding`: Optional. Enter one of the following if `value` is a string: `utf8`, `hex`, `base64`, `base64url`. Default is `utf8`.

- `Buffer.prototype.readInt8(offset)`

  Read `Int8` at `offset` from `Buffer`.

  - `offset`: Enter an integer.

- `Buffer.prototype.readIntBE(offset, byteLength)`

  Read `Int` as big-endian at `offset` from `Buffer`.

  - `offset`: Enter an integer.
  - `byteLength`: Optional. Enter an integer from 1 to 6.

- `Buffer.prototype.readInt16BE(offset)`

  Read `Int16` as big-endian at `offset` from `Buffer`.

  - `offset`: Enter an integer.

- `Buffer.prototype.readInt32BE(offset)`

  Read `Int32` as big-endian at `offset` from `Buffer`.

  - `offset`: Enter an integer.

- `Buffer.prototype.readIntLE(offset, byteLength)`

  Read `Int` as little-endian at `offset` from `Buffer`.

  - `offset`: Enter an integer.
  - `byteLength`: Enter an integer from 1 to 6.

- `Buffer.prototype.readInt16LE(offset)`

  Read `Int16` as little-endian at `offset` from `Buffer`.

  - `offset`: Enter an integer.

- `Buffer.prototype.readInt32LE(offset)`

  Read `Int32` as little-endian at `offset` from `Buffer`.

  - `offset`: Enter an integer.
- `Buffer.prototype.readUInt8(offset)`

  Read `UInt8` at `offset` from `Buffer`.

  - `offset`: Enter an integer.
- `Buffer.prototype.readUIntBE(offset, byteLength)`

  Read `UInt` as big-endian at `offset` from `Buffer`.

  - `offset`: Enter an integer.
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.readUInt16BE(offset)`

  Read `UInt16` as big-endian at `offset` from `Buffer`.

- • `offset`: Enter an integer.
- `Buffer.prototype.readUInt32BE(offset)`

  Read `UInt32` as big-endian at `offset` from `Buffer`.

  - `offset`: Enter an integer.
- `Buffer.prototype.readUIntLE(offset, byteLength)`

  Read `UInt` as little-endian at `offset` from `Buffer`.

  - `offset`: Enter an integer.
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.readUInt16LE(offset)`

  Read `UInt16` as little-endian at `offset` from `Buffer`.

  - `offset`: Enter an integer.
- `Buffer.prototype.readUInt32LE(offset)`

  Read `UInt32` as little-endian at `offset` from `Buffer`.

  - `offset`: Enter an integer.
- `Buffer.prototype.readDoubleBE([offset])`

  Read a 64-bit double as big-endian at `offset` from `Buffer`.

  - `offset`: Optional. Enter an integer.

- `Buffer.prototype.readDoubleLE([offset])`

  Read a 64-bit double as little-endian at `offset` from `Buffer`.

  - `offset`: Optional. Enter an integer.
- `Buffer.prototype.readFloatBE([offset])`

  Read a 32-bit float as big-endian at `offset` from `Buffer`.

  - `offset`: Optional. Enter an integer.
- `Buffer.prototype.readFloatLE([offset])`

  Read a 32-bit float as little-endian at `offset` from `Buffer`.

  - `offset`: Optional. Enter an integer.
- `Buffer.prototype.subarray([start[, end]])`

  Returns a copy of `Buffer` that is offset and cropped with a new `start` and `end`.

  - `start`: Optional. Enter an integer. Default is 0.
  - `end`: Optional. Enter an integer. Default is buffer length.
- `Buffer.prototype.swap16()`

  Swap the `Buffer` array byte order, treating it as an array of 16-bit numbers. `Buffer` length must be divisible by 2, or you will receive an error.
- `Buffer.prototype.swap32()`

  Swap the `Buffer` array byte order, treating it as an array of 32-bit numbers . `Buffer` length must be divisible by 4, or you will receive an error.
- `Buffer.prototype.swap64()`

  Swap the `Buffer` array byte order, treating it as an array of 64-bit numbers. `Buffer` length must be divisible by 8, or you will receive an error.
- `Buffer.prototype.toJSON()`

  Returns `Buffer` as a JSON.
- `Buffer.prototype.toString([encoding[, start[, end]]])`

  Convert `Buffer`, from `start` to `end`, to encoded string.

  - encoding: Optional. Enter one of the following: utf8, hex, base64, or base64url. Default is utf8.
  - start: Optional. Enter an integer. Default is 0.
  - end: Optional. Enter an integer. Default is buffer length.
- Buffer.prototype.write(string[, offset[, length]][, encoding])

  Write encoded string to Buffer if there is space, or a truncated string if there is not enough space.
  - string: Enter a string.
  - offset: Optional. Enter an integer. Default is 0.
  - length: Optional. Enter an integer. Default is the length of the string.
  - encoding: Optional. Optionally enter one of the following: utf8, hex, base64, or base64url. Default is utf8.
- Buffer.prototype.writeInt8(value, offset, byteLength)

  Write Int8 value of byteLength at offset to Buffer.
  - value: Enter an integer.
  - offset: Enter an integer
  - byteLength: Enter an integer from 1 to 6.
- Buffer.prototype.writeIntBE(value, offset, byteLength)

  Write value at offset to Buffer, using big-endian.
  - value: Enter an integer.
  - offset: Enter an integer
  - byteLength: Enter an integer from 1 to 6.
- Buffer.prototype.writeInt16BE(value, offset, byteLength)

  Write value at offset to Buffer, using big-endian.
  - value: Enter an integer.
  - offset: Enter an integer
  - byteLength: Enter an integer from 1 to 6.
- Buffer.prototype.writeInt32BE(value, offset, byteLength)

  Write value at offset to Buffer, using big-endian.

  - `value`: Enter an integer.
  - `offset`: Enter an integer
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.writeIntLE(offset, byteLength)`

  Write `value` at `offset` to `Buffer`, using little-endian.
  - `offset`: Enter an integer.
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.writeInt16LE(offset, byteLength)`

  Write `value` at `offset` to `Buffer`, using little-endian.
  - `offset`: Enter an integer.
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.writeInt32LE(offset, byteLength)`

  Write `value` at `offset` to `Buffer`, using little-endian.
  - `offset`: Enter an integer.
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.writeUInt8(value, offset, byteLength)`

  Write `UInt8` `value` of `byteLength` at `offset` to `Buffer`.
  - `value`: Enter an integer.
  - `offset`: Enter an integer
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.writeUIntBE(value, offset, byteLength)`

  Write `value` at `offset` to `Buffer`, using big-endian.
  - `value`: Enter an integer.
  - `offset`: Enter an integer
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.writeUInt16BE(value, offset, byteLength)`

  Write `value` at `offset` to `Buffer`, using big-endian.
  - `value`: Enter an integer.

  - `offset`: Enter an integer
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.writeUInt32BE(value, offset, byteLength)`

  Write `value` at `offset` to `Buffer`, using big-endian.
  - `value`: Enter an integer.
  - `offset`: Enter an integer
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.writeUIntLE(value, offset, byteLength)`

  Write `value` at `offset` to `Buffer`, using little-endian.
  - `value`: Enter an integer.
  - `offset`: Enter an integer
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.writeUInt16LE(value, offset, byteLength)`

  Write `value` at `offset` to `Buffer`, using little-endian.
  - `value`: Enter an integer.
  - `offset`: Enter an integer
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.writeUInt32LE(value, offset, byteLength)`

  Write `value` at `offset` to `Buffer`, using little-endian.
  - `value`: Enter an integer.
  - `offset`: Enter an integer
  - `byteLength`: Enter an integer from 1 to 6.
- `Buffer.prototype.writeDoubleBE(value, [offset])`

  Write `value` at `offset` to `Buffer`, using big-endian.
  - `value`: Enter an integer.
  - `offset`: Optional. Enter an integer. Default is 0.
- `Buffer.prototype.writeDoubleLE(value, [offset])`

  Write `value` at `offset` to `Buffer`, using little-endian.

  - `value`: Enter an integer.
  - `offset`: Optional. Enter an integer. Default is 0.
- `Buffer.prototype.writeFloatBE(value, [offset])`

  Write `value` at `offset` to `Buffer`, using big-endian.
  - `value`: Enter an integer.
  - `offset`: Optional. Enter an integer. Default is 0.
- `Buffer.prototype.writeFloatLE(value, [offset])`

  Write `value` at `offset` to `Buffer`, using little-endian.
  - `value`: Enter an integer.
  - `offset`: Optional. Enter an integer. Default is 0.

The following instance methods are supported:

- `buffer[index]`

  Get and set octet (byte) at `index` in `Buffer`.
  - Get a number from `0` to `255`. Or set a number from from `0` to `255`.

The following instance properties are supported:

- `buffer`

  Get the `ArrayBuffer` object for the buffer.
- `byteOffset`

  Get the `byteOffset` of the buffer's `Arraybuffer` object.
- `length`

  Get the buffer byte count.

> **Note**
>
> All Buffer module methods are new in JavaScript runtime 2.0.

Query string

> **Note**
>
> The CloudFront Functions event object automatically parses URL query strings for you. That means that in most cases you don't need to use this module.

The query string module (`querystring`) provides methods for parsing and formatting URL query strings. You can load the module using `require('querystring')`. The module provides the following methods.

`querystring.escape(`*`string`*`)`

URL-encodes the given `string`, returning an escaped query string. The method is used by `querystring.stringify()` and should not be used directly.

`querystring.parse(`*`string`*`[, `*`separator`*`[, `*`equal`*`[, `*`options`*`]]])`

Parses a query string (`string`) and returns an object.

The `separator` parameter is a substring for delimiting key and value pairs in the query string. By default it is &.

The `equal` parameter is a substring for delimiting keys and values in the query string. By default it is =.

The `options` parameter is an object with the following keys:

`decodeURIComponent` *`function`*

A function to decode percent-encoded characters in the query string. By default it is `querystring.unescape()`.

`maxKeys` *`number`*

The maximum number of keys to parse. By default it is `1000`. Use a value of `0` to remove the limitations for counting keys.

By default, percent-encoded characters within the query string are assumed to use the UTF-8 encoding. Invalid UTF-8 sequences are replaced with the `U+FFFD` replacement character.

For example, for the following query string:

```
'name=value&abc=xyz&abc=123'
```

The return value of `querystring.parse()` is:

```
{
name: 'value',
abc: ['xyz', '123']
}
```

`querystring.decode()` is an alias for `querystring.parse()`.

`querystring.stringify(`*`object`*`[, `*`separator`*`[, `*`equal`*`[, `*`options`*`]]])`

Serializes an `object` and returns a query string.

The `separator` parameter is a substring for delimiting key and value pairs in the query string. By default it is &.

The `equal` parameter is a substring for delimiting keys and values in the query string. By default it is =.

The `options` parameter is an object with the following keys:

`encodeURIComponent` *`function`*

The function to use for converting URL-unsafe characters to percent-encoding in the query string. By default it is `querystring.escape()`.

By default, characters that require percent-encoding within the query string are encoded as UTF-8. To use a different encoding, specify the `encodeURIComponent` option.

For example, for the following code:

```
querystring.stringify({ name: 'value', abc: ['xyz', '123'], anotherName: '' });
```

The return value is:

```
'name=value&abc=xyz&abc=123&anotherName='
```

`querystring.encode()` is an alias for `querystring.stringify()`.

`querystring.unescape(`*`string`*`)`

Decodes URL percent-encoded characters in the given `string`, returning an unescaped query string. This method is used by `querystring.parse()` and should not be used directly.

Crypto

The cryptographic module (`crypto`) provides standard hashing and hash-based message authentication code (HMAC) helpers. You can load the module using `require('crypto')`.

Hashing methods

`crypto.createHash(`*`algorithm`*`)`

Creates and returns a hash object that you can use to generate hash digests using the given algorithm: `md5`, `sha1`, or `sha256`.

`hash.update(`*`data`*`)`

Updates the hash content with the given `data`.

`hash.digest([`*`encoding`*`])`

Calculates the digest of all of the data passed using `hash.update()`. The encoding can be `hex`, `base64`, or `base64url`.

HMAC methods

`crypto.createHmac(`*`algorithm, secret key`*`)`

Creates and returns an HMAC object that uses the given `algorithm` and `secret key`. The algorithm can be `md5`, `sha1`, or `sha256`.

`hmac.update(`*`data`*`)`

Updates the HMAC content with the given `data`.

`hmac.digest([`*`encoding`*`])`

Calculates the digest of all of the data passed using `hmac.update()`. The encoding can be `hex`, `base64`, or `base64url`.

Restricted features

The following JavaScript language features are either unsupported or restricted due to security concerns.

Dynamic code evaluation

Dynamic code evaluation is not supported. Both `eval()` and `Function` constructors throw an error if attempted. For example, `const sum = new Function('a', 'b', 'return a + b')` throws an error.

Timers

The `setTimeout()`, `setImmediate()`, and `clearTimeout()` functions are not supported. There is no provision to defer or yield within a function run. Your function must synchronously run to completion.

Date and timestamps

For security reasons, there is no access to high-resolution timers. All `Date` methods to query the current time always return the same value during the lifetime of a single function run. The returned timestamp is the time when the function started running. Consequently, you cannot measure elapsed time in your function.

File system access

There is no file system access.

Network access

There is no support for network calls. For example, XHR, HTTP(S), and socket are not supported.

Helper methods for key value stores

This section applies if you use the CloudFront Key Value Store to include key values in the function that you create. CloudFront Functions has a module that provides three helper methods to read values from the key value store.

To use this module in the function code, make sure that you have associated a key value store with the function.

Next, include the following statements in the first lines of the function code:

```
import cf from 'cloudfront';
const kvsId = "key value store ID";
const kvsHandle = cf.kvs(kvsId);
```

Your *key value store ID* might look like the following: `a1b2c3d4-5678-90ab-cdef-EXAMPLE1`

The get() method

Use this method to return the key value for the key name that you specify.

Request

```
get("key", options);
```

- `key`: The name of the key whose value needs to be fetched
- `options`: There is one option, `format`. It ensures that the function parses the data correctly. Possible values:
  - `string`: (Default) UTF8 encoded
  - `json`
  - `bytes`: Raw binary data buffer

Request example

```
const value = await kvsHandle.get("myFunctionKey", { format: "string"});
```

Response

The response is a `promise` that resolves to a value in the format requested by using `options`. By default, the value is returned as a string.

The exists() method

Use this method to identify whether or not the key exists in the key value store.

Request

```
exists("key");
```

Request example

```
const exist = await kvsHandle.exists("myFunctionkey");
```

Response

The response is a `promise` that returns a Boolean (`true` or `false`). This value specifies whether or not the key exists in the key value store.

Error handling

The `get()` method will return an error when the key that you requested doesn't exist in the associated key value store. To manage this use case, you can add a `try` and `catch` block to your code.

The meta() method

Use this method to return metadata about the key value store.

Request

```
meta();
```

Request example

```
const meta = await kvsHandle.meta();
```

Response

The response is a `promise` that resolves to an object with the following properties:

- `creationDateTime`: The date and time that the key value store was created, in ISO 8601 format.
- `lastUpdatedDateTime`: The date and time that the key value store was last synced from the source, in ISO 8601 format. The value doesn't include the propagation time to the edge.
- `keyCount`: The total number of keys in the KVS after the last sync from the source.

Response example

```
{keyCount:3,creationDateTime:2023-11-30T23:07:55.765Z,lastUpdatedDateTime:2023-12-15T03:57:52.4
```

Example code for CloudFront Functions

Use the following example functions to help you get started writing function code for CloudFront Functions. All of these examples are available in the amazon-cloudfront-functions repository on GitHub.

Examples

- Add a Cache-Control header to the response
- Add a cross-origin resource sharing (CORS) header to the response
- Add cross-origin resource sharing (CORS) header to the request
- Add security headers to the response
- Add a True-Client-IP header to the request
- Redirect the viewer to a new URL
- Add index.html to request URLs that don't include a file name
- Validate a simple token in the request
- Using async and await
- Normalize query string parameters
- Use key value pairs in a function

Add a `Cache-Control` header to the response

The following example function adds a `Cache-Control` HTTP header to the response. The header uses the `max-age` directive to tell web browsers to cache the response for a maximum of two years (63,072,000 seconds). For more information, see Cache-Control on the MDN Web Docs website.

This is a viewer response function.

See this example on GitHub.

JavaScript runtime 2.0

```
async function handler(event) {
    const response = event.response;
    const headers = response.headers;
```

```
    // Set the cache-control header
     headers['cache-control'] = {value: 'public, max-age=63072000'};

    // Return response to viewers
    return response;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var response = event.response;
    var headers = response.headers;

    // Set the cache-control header
    headers['cache-control'] = {value: 'public, max-age=63072000'};

    // Return response to viewers
    return response;
}
```

Add a cross-origin resource sharing (CORS) header to the response

The following example function adds an `Access-Control-Allow-Origin` HTTP header to the response if the response doesn't already contain this header. This header is part of cross-origin resource sharing (CORS). The header's value (*) tells web browsers to allow code from any origin to access this resource. For more information, see Access-Control-Allow-Origin on the MDN Web Docs website.

This is a viewer response function.

See this example on GitHub.

JavaScript runtime 2.0

```
async function handler(event)  {
    const request = event.request;
    const response  = event.response;

    // If Access-Control-Allow-Origin CORS header is missing, add it.
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation.
```

```
    if (!response.headers['access-control-allow-origin'] &&
 request.headers['origin']) {
        response.headers['access-control-allow-origin'] = {value:
 request.headers['origin'].value};
        console.log("Access-Control-Allow-Origin was missing, adding it now.");
    }

    return response;
}
```

JavaScript runtime 1.0

```
function handler(event)  {
    var response  = event.response;
    var headers  = response.headers;

    // If Access-Control-Allow-Origin CORS header is missing, add it.
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation.
    if (!headers['access-control-allow-origin']) {
        headers['access-control-allow-origin'] = {value: "*"};
        console.log("Access-Control-Allow-Origin was missing, adding it now.");
    }

    return response;
}
```

Add cross-origin resource sharing (CORS) header to the request

The following example function adds an `Origin` HTTP header to the request if the request doesn't already contain this header. This header is part of cross-origin resource sharing (CORS). This example sets the header's value to the value in the request's `Host` header. For more information, see Origin on the MDN Web Docs website.

This is a viewer request function.

See this example on GitHub.

JavaScript runtime 2.0

```
async function handler(event) {
```

```
    const request = event.request;
    const headers = request.headers;
    const host = request.headers.host.value;

   // If origin header is missing, set it equal to the host header.
   if (!headers.origin)
       headers.origin = {value:`https://${host}`};

   return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var request = event.request;
    var headers = request.headers;
    var host = request.headers.host.value;

   // If origin header is missing, set it equal to the host header.
   if (!headers.origin)
       headers.origin = {value:`https://${host}`};

   return request;
}
```

Add security headers to the response

The following example function adds several common security-related HTTP headers to the response. For more information, see the following pages on the MDN Web Docs website:

- Strict-Transport-Security
- Content-Security-Policy
- X-Content-Type-Options
- X-Frame-Options
- X-XSS-Protection

This is a viewer response function.

See this example on GitHub.

JavaScript runtime 2.0

```
async function handler(event) {
    const response = event.response;
    const headers = response.headers;

    // Set HTTP security headers
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation
    headers['strict-transport-security'] = { value: 'max-age=63072000;
 includeSubdomains; preload'};
    headers['content-security-policy'] = { value: "default-src 'none'; img-src
 'self'; script-src 'self'; style-src 'self'; object-src 'none'; frame-ancestors
 'none'"};
    headers['x-content-type-options'] = { value: 'nosniff'};
    headers['x-frame-options'] = {value: 'DENY'};
    headers['x-xss-protection'] = {value: '1; mode=block'};
    headers['referrer-policy'] = {value: 'same-origin'};

    // Return the response to viewers
    return response;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var response = event.response;
    var headers = response.headers;

    // Set HTTP security headers
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation
    headers['strict-transport-security'] = { value: 'max-age=63072000;
 includeSubdomains; preload'};
    headers['content-security-policy'] = { value: "default-src 'none'; img-src
 'self'; script-src 'self'; style-src 'self'; object-src 'none'"};
    headers['x-content-type-options'] = { value: 'nosniff'};
    headers['x-frame-options'] = {value: 'DENY'};
    headers['x-xss-protection'] = {value: '1; mode=block'};

    // Return the response to viewers
    return response;
}
```

Add a `True-Client-IP` header to the request

The following example function adds a `True-Client-IP` HTTP header to the request, with the IP address of the viewer as the header's value. When CloudFront sends a request to an origin, the origin can determine the IP address of the CloudFront host that sent the request but not the IP address of the viewer (client) that sent the original request to CloudFront. This function adds the `True-Client-IP` header so the origin can see the IP address of the viewer.

> **Important**
>
> To make sure that CloudFront includes this header in origin requests, you must add it to the allowed headers list in an origin request policy.

This is a viewer request function.

See this example on GitHub.

JavaScript runtime 2.0

```
async function handler(event) {
    var request = event.request;
    var clientIP = event.viewer.ip;

    //Add the true-client-ip header to the incoming request
    request.headers['true-client-ip'] = {value: clientIP};

    return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var request = event.request;
    var clientIP = event.viewer.ip;

    //Add the true-client-ip header to the incoming request
    request.headers['true-client-ip'] = {value: clientIP};

    return request;
}
```

Redirect the viewer to a new URL

The following example function generates a response to redirect the viewer to a country-specific URL when the request comes from within a particular country. This function relies on the value of the `CloudFront-Viewer-Country` header to determine the viewer's country.

> **Important**
>
> For this function to work, you must configure CloudFront to add the `CloudFront-Viewer-Country` header to incoming requests by adding it to the allowed headers in a cache policy or an origin request policy.

This example redirects the viewer to a Germany-specific URL when the viewer request comes from Germany. If the viewer request doesn't come from Germany, the function returns the original, unmodified request.

This is a viewer request function.

See this example on GitHub.

JavaScript runtime 2.0

```
async function handler(event) {
    const request = event.request;
    const headers = request.headers;
    const host = request.headers.host.value;
    const country = Symbol.for('DE'); // Choose a country code
    const newurl = `https://${host}/de/index.html`; // Change the redirect URL to
 your choice

    if (headers['cloudfront-viewer-country']) {
        const countryCode = Symbol.for(headers['cloudfront-viewer-country'].value);
        if (countryCode === country) {
            const response = {
                statusCode: 302,
                statusDescription: 'Found',
                headers:
                    { "location": { "value": newurl } }
                }
```

```
            return response;
        }
    }
    return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var request = event.request;
    var headers = request.headers;
    var host = request.headers.host.value;
    var country = 'DE' // Choose a country code
    var newurl = `https://${host}/de/index.html` // Change the redirect URL to your
 choice

    if (headers['cloudfront-viewer-country']) {
        var countryCode = headers['cloudfront-viewer-country'].value;
        if (countryCode === country) {
            var response = {
                statusCode: 302,
                statusDescription: 'Found',
                headers:
                    { "location": { "value": newurl } }
                }

            return response;
        }
    }
    return request;
}
```

For more information about rewrites and redirects, see Handling rewrites and redirects using edge functions in the Amazon workshop studio.

Add `index.html` to request URLs that don't include a file name

The following example function appends `index.html` to requests that don't include a file name or extension in the URL. This function can be useful for single page applications or statically generated websites that are hosted in an Amazon S3 bucket.

This is a viewer request function.

See this example on GitHub.

JavaScript runtime 2.0

```
async function handler(event) {
    const request = event.request;
    const uri = request.uri;

    // Check whether the URI is missing a file name.
    if (uri.endsWith('/')) {
        request.uri += 'index.html';
    }
    // Check whether the URI is missing a file extension.
    else if (!uri.includes('.')) {
        request.uri += '/index.html';
    }

    return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var request = event.request;
    var uri = request.uri;

    // Check whether the URI is missing a file name.
    if (uri.endsWith('/')) {
        request.uri += 'index.html';
    }
    // Check whether the URI is missing a file extension.
    else if (!uri.includes('.')) {
        request.uri += '/index.html';
    }

    return request;
}
```

Validate a simple token in the request

The following example function validates a JSON web token (JWT) in the query string of a request. If the token is valid, the function returns the original, unmodified request to CloudFront. If the

token is not valid, the function generates an error response. This function uses the `crypto` module. For more information, see Built-in modules.

This function assumes that requests contain a JWT value in a query string parameter named `jwt`.

> **Warning**
>
> To use this function, you must put your secret key in the function code.

This is a viewer request function.

See this example on GitHub.

JavaScript runtime 2.0

```
const crypto = require('crypto');

//Response when JWT is not valid.
const response401 = {
    statusCode: 401,
    statusDescription: 'Unauthorized'
};

function jwt_decode(token, key, noVerify, algorithm) {
    // check token
    if (!token) {
        throw new Error('No token supplied');
    }
    // check segments
    const segments = token.split('.');
    if (segments.length !== 3) {
        throw new Error('Not enough or too many segments');
    }

    // All segment should be base64
    const headerSeg = segments[0];
    const payloadSeg = segments[1];
    const signatureSeg = segments[2];

    // base64 decode and parse JSON
    const header = JSON.parse(_base64urlDecode(headerSeg));
    const payload = JSON.parse(_base64urlDecode(payloadSeg));
```

```
    if (!noVerify) {
        const signingMethod = 'sha256';
        const signingType = 'hmac';

        // Verify signature. `sign` will return base64 string.
        const signingInput = [headerSeg, payloadSeg].join('.');

        if (!_verify(signingInput, key, signingMethod, signingType, signatureSeg)) {
            throw new Error('Signature verification failed');
        }

        // Support for nbf and exp claims.
        // According to the RFC, they should be in seconds.
        if (payload.nbf && Date.now() < payload.nbf*1000) {
            throw new Error('Token not yet active');
        }

        if (payload.exp && Date.now() > payload.exp*1000) {
            throw new Error('Token expired');
        }
    }

    return payload;
}

//Function to ensure a constant time comparison to prevent
//timing side channels.
function _constantTimeEquals(a, b) {
    if (a.length != b.length) {
        return false;
    }

    var xor = 0;
    for (var i = 0; i < a.length; i++) {
    xor |= (a.charCodeAt(i) ^ b.charCodeAt(i));
    }

    return 0 === xor;
}

function _verify(input, key, method, type, signature) {
    if(type === "hmac") {
        return _constantTimeEquals(signature, _sign(input, key, method));
```

```
    }
    else {
        throw new Error('Algorithm type not recognized');
    }
}

function _sign(input, key, method) {
    return crypto.createHmac(method, key).update(input).digest('base64url');
}

function _base64urlDecode(str) {
    return Buffer.from(str, 'base64url')
}

function handler(event) {
    const request = event.request;
    //Secret key used to verify JWT token.
    //Update with your own key.
    var key = "LzdWGpAToQ1DqYuzHxE6YOqi7G3X2yvNBot9mCXfx5k";

    // If no JWT token, then generate HTTP redirect 401 response.
    if(!request.querystring.jwt) {
        console.log("Error: No JWT in the querystring");
        return response401;
    }

    const jwtToken = request.querystring.jwt.value;

    try{
        jwt_decode(jwtToken, key);
    }
    catch(e) {
        console.log(e);
        return response401;
    }

    //Remove the JWT from the query string if valid and return.
    delete request.querystring.jwt;
    console.log("Valid JWT token");
    return request;
}
```

JavaScript runtime 1.0

```
var crypto = require('crypto');

//Response when JWT is not valid.
var response401 = {
    statusCode: 401,
    statusDescription: 'Unauthorized'
};

function jwt_decode(token, key, noVerify, algorithm) {
    // check token
    if (!token) {
        throw new Error('No token supplied');
    }
    // check segments
    var segments = token.split('.');
    if (segments.length !== 3) {
        throw new Error('Not enough or too many segments');
    }

    // All segment should be base64
    var headerSeg = segments[0];
    var payloadSeg = segments[1];
    var signatureSeg = segments[2];

    // base64 decode and parse JSON
    var header = JSON.parse(_base64urlDecode(headerSeg));
    var payload = JSON.parse(_base64urlDecode(payloadSeg));

    if (!noVerify) {
        var signingMethod = 'sha256';
        var signingType = 'hmac';

        // Verify signature. `sign` will return base64 string.
        var signingInput = [headerSeg, payloadSeg].join('.');

        if (!_verify(signingInput, key, signingMethod, signingType, signatureSeg)) {
            throw new Error('Signature verification failed');
        }

        // Support for nbf and exp claims.
        // According to the RFC, they should be in seconds.
        if (payload.nbf && Date.now() < payload.nbf*1000) {
```

```
            throw new Error('Token not yet active');
        }

        if (payload.exp && Date.now() > payload.exp*1000) {
            throw new Error('Token expired');
        }
    }

    return payload;
}

function _verify(input, key, method, type, signature) {
    if(type === "hmac") {
        return (signature === _sign(input, key, method));
    }
    else {
        throw new Error('Algorithm type not recognized');
    }
}

function _sign(input, key, method) {
    return crypto.createHmac(method, key).update(input).digest('base64url');
}

function _base64urlDecode(str) {
    return String.bytesFrom(str, 'base64url')
}

function handler(event) {
    var request = event.request;

    //Secret key used to verify JWT token.
    //Update with your own key.
    var key = "LzdWGpAToQ1DqYuzHxE6YOqi7G3X2yvNBot9mCXfx5k";

    // If no JWT token, then generate HTTP redirect 401 response.
    if(!request.querystring.jwt) {
        console.log("Error: No JWT in the querystring");
        return response401;
    }

    var jwtToken = request.querystring.jwt.value;

    try{
```

```
        jwt_decode(jwtToken, key);
    }
    catch(e) {
        console.log(e);
        return response401;
    }

    //Remove the JWT from the query string if valid and return.
    delete request.querystring.jwt;
    console.log("Valid JWT token");
    return request;
}
```

Using async and await

Amazon CloudFront JavaScript runtime functions 2.0 provide `async` and `await` syntax to handle `Promise` objects. Promises represent delayed results that can be accessed via the `await` keyword in functions marked as `async`. Various new WebCrypto functions use Promises.

For more information about `Promise` objects, see Promise.

```
async function answer() {
    return 42;
}

// Note: async, await can be used only inside an async function.

async function handler(event) {
    // var answer_value = answer(); // returns Promise, not a 42 value
    let answer_value = await answer(); // resolves Promise, 42
    console.log("Answer"+answer_value);
    event.request.headers['answer'] = { value : ""+answer_value };
    return event.request;
}
```

The following example JavaScript code shows how to view promises with the `then` chain method. You can use `catch` to view errors.

```
async function answer() {
    return 42;
}
```

```
async function squared_answer() {
   // before, in NJS 0.4.3 we have to write as following
   // return answer().then(function(value) { return value * value; })

   // in NJS 0.7.11 we can simplify
   return answer().then(value => value * value)
}
// note async, await can be used only inside async function
async function handler(event) {
    // var answer_value = answer(); // returns Promise, not a 42 value
    let answer_value = await squared_answer(); // resolves Promise, 42
    console.log("Answer"+answer_value);
    event.request.headers['answer'] = { value : ""+answer_value };
    return event.request;
}
```

Note

async and await are only available when you use JavaScript runtime 2.0.

Normalize query string parameters

You can normalize query string parameters to improve the cache hit ratio.

The following example shows how to improve your cache hit ratio by putting the query strings in alphabetical order before CloudFront forwards requests to your origin.

```
function handler(event) {
     var qs=[];
     for (var key in event.request.querystring) {
         if (event.request.querystring[key].multiValue) {
             event.request.querystring[key].multiValue.forEach((mv) => {qs.push(key +
 "=" + mv.value)});
         } else {
             qs.push(key + "=" + event.request.querystring[key].value);
         }
     };

     event.request.querystring = qs.sort().join('&');
```

```
    return event.request;
}
```

Use key value pairs in a function

You can use key value pairs from a key value store in a function.

The following example shows a function that uses the content of the URL in the HTTP request to look up a custom path in the key value store. CloudFront then uses that custom path to make the request. This function helps manage the multiple paths that are part of a website.

```
import cf from 'cloudfront';

// Declare the ID of the key value store that you have associated with this function
// The import fails at runtime if the specified key value store is not associated with
 the function

const kvsId = "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111";

const kvsHandle = cf.kvs(kvsId);

async function handler(event) {
    const request = event.request;
    // Use the first segment of the pathname as key
    // For example http(s)://domain/<key>/something/else
    const pathSegments = request.uri.split('/')
    const key = pathSegments[1]
    try {
        // Replace the first path of the pathname with the value of the key
        // For example http(s)://domain/<value>/something/else
        pathSegments[1] = await kvsHandle.get(key);
        const newUri = pathSegments.join('/');
        console.log(`${request.uri} -> ${newUri}`)
        request.uri = newUri;
    } catch (err) {
        // No change to the pathname if the key is not found
        console.log(`${request.uri} | ${err}`);
    }
    return request;
}
```

Managing functions in CloudFront Functions

With CloudFront Functions, you can write lightweight functions in JavaScript for high-scale, latency-sensitive CDN customizations. After you write the function code, the following topics can help you create the function in CloudFront Functions, test it, update it, publish it, and associate it with a CloudFront distribution.

The full procedure for deploying a function is the following:

- Create the function. The function is initially created in with sample function code. This code is valid, even in the live stage. The function exists in the Development stage.

- Test, update, associate. When the function is in the Development stage, you can test the function, and update it (including associating a key value store with it).
- Publish. When you're ready to use your function with a CloudFront distribution, you publish the function, which copies it from the `DEVELOPMENT` stage to `LIVE`.
- Associate with a distribution. When the function is in the `LIVE` stage, you can associate the function with a distribution's cache behavior.

Topics

- Creating functions
- Testing functions
- Updating functions
- Publishing functions
- Associating functions with distributions

Creating functions

You create a function in two stages. First you create the function code as Java Script, outside of CloudFront. Then you use CloudFront to create the function and include the code. The code exists inside the function (not as a reference).

The new function is added to the `DEVELOPMENT` stage. You must publish the function to copy it over to the `LIVE` stage (**Published** in the console).

Console

To create a function (console)

1. Sign in to the Amazon Web Services Management Console and open the **Functions** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#/functions.
2. Choose **Create function**.
3. Enter a function name that is unique within the Amazon account, then choose the Java Script version, and then choose **Continue**. The function now exists. The details page for the new function appears.

   > **Note**
   >
   > If you want to use key value pairs in the function, you must choose Java Script 2.0.

4. In the **Function code** section, select the **Build** tab and enter your function code. The sample code that is included in the **Build** tab illustrates the basic syntax for the function code. You can complete the code as follows:

   - Use the default function just so you can get started.
   - Replace it with code that you copy from example code on GitHub.
   - Replace it with your own code.

   For more information about writing function code, see the following:

   - Writing function code
   - the section called “Event structure”
5. Choose **Save changes** as often as you want, to save the function code.
6. If the function code uses key value pairs, you must associate a key value store.

   You can associate the key value store during the initial creation of the function. Or you can associate it later, by updating the function.

   To associate a key value store now, follow these steps:

   - Go to the **Associate KeyValueStore** section and choose **Associate existing KeyValueStore**.

- Select the key value store that contains the key value pairs in the function, then choose **Associate KeyValueStore**.

CloudFront immediately associates the store with the function. You don't need to save the function.

CLI

If you use the CLI, you typically first create the function code in a file, and then create the function with the Amazon CLI.

1. Create the function code in a file, and store it in a directory that your computer can connect to. For more information about writing function code, see the following:
   - Writing function code
   - the section called “Event structure”
2. Run the command as shown in the example. This example uses the fileb:// notation to pass in the file. It also includes line breaks to make the command more readable.

```
aws cloudfront create-function \
    --name MaxAge \
    --function-config '{"Comment":"Max Age 2 years","Runtime":"cloudfront-
js-2.0","KeyValueStoreAssociations":{"Quantity":1,"Items":
[{"KeyValueStoreARN":"arn:aws:cloudfront::111122223333:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"}]}}' \
    --function-code fileb://function-max-age-v1.js
```

Notes:

- `Runtime`: The version of Java Script. If you want to use key value pairs in the function, you must specify version 2.0.
- `KeyValueStoreAssociations`: If your function uses key value pairs, you can associate the key value store during the initial creation of the function. Or you can associate it later, by using `update-function`. The `Quantity` is always 1 because each function can have only one key value store associated with it.

When the command is successful, you see output like the following.

```
ETag: ETVABCEXAMPLE
FunctionSummary:
  FunctionConfig:
    Comment: Max Age 2 years
    Runtime: cloudfront-js-2.0
    KeyValueStoreAssociations= \
      {Quantity=1, \
      Items=[{KeyValueStoreARN='arn:aws:cloudfront::111122223333:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]} \
  FunctionMetadata:
    CreatedTime: '2021-04-18T20:38:56.915000+00:00'
    FunctionARN: arn:aws:cloudfront::111122223333:function/MaxAge
    LastModifiedTime: '2023-11-19T20:38:56.915000+00:00'
    Stage: DEVELOPMENT
  Name: MaxAge
  Status: UNPUBLISHED
Location: https://cloudfront.amazonaws.com/2020-05-31/function/
arn:aws:cloudfront:::function/MaxAge
```

Most of the information is repeated from the request. Other information is added by CloudFront.

Notes

- ETag: This value changes each time you modify the key value store. You use this value and the function name to reference the function in the future. Make sure that you always use the current ETag.
- FunctionARN
- Stage
- 111122223333
- Status

Testing functions

You can test a CloudFront Function to make sure that it works as intended before you deploy the function to the live stage (production). To test a function, you provide an *event object* that

represents an HTTP request or response that your CloudFront distribution could receive in production. CloudFront Functions does the following:

1. Runs the function, using the provided event object as input.
2. Returns the function's result (the modified event object) along with any function logs or error messages and the function's *compute utilization*. For more information about compute utilization, see the section called “Understanding compute utilization”.

Topics

- Set up the event object
- Test the function
- Understanding compute utilization

Set up the event object

Before you test a function, you must set up the event object to test it with. There are several options.

Option 1: Set up an event object without saving it

You can set up an event object in the visual editor in the CloudFront console and not save it.

You can use this event object to test the function from the CloudFront console, even though it's not saved.

Option 2: Create an event object in the visual editor

You can set up an event object in the visual editor in the CloudFront console and not save it. You can create 10 event objects for each function so that you can, for example, test different possible inputs.

When you create the event object in this way, you can use the event object to test the function in the CloudFront console. You can't use it to test the function using an Amazon API or SDK.

Option 3: Create an event object using a text editor

You can use a text editor to create an event object in JSON format. For information about the structure of an event object, see Event structure.

You can use this event object to test the function using the CLI. But you can't use it to test the function in the CloudFront console.

To create using option 1 or 2

1. Display the **Functions** page in the CloudFront console and choose the function that you want to test.
2. On the function details page, choose the **Test** tab. The **Test function** section appears with buttons that include **Edit JSON** and **Test function**.
3. Complete **Event type**:
   - Choose **Viewer request** if the function modifies an HTTP request or generates a response based on the request. The **Request** section, which is already displayed applies to this type.
   - Or choose **Viewer response**. The **Request** section, which is already displayed, applies to this type. In addition, the **Response** section appears.
4. Complete all the fields that you want to include in the event. As you work, you can choose **Edit JSON** to view the raw JSON.
5. Save the event, if you want to.

   You can also choose **Edit JSON** and copy the raw JSON, and save it in your own file, outside of CloudFront.

To create using option 3

Create the event object using a text editor. Store the file in a directory that your computer can connect to.

Makes sure that you follow these guidelines:

- Omit the `distributionDomainName`, `distributionId`, and `requestId` fields.
- Make sure that the names of headers, cookies, and query strings are lowercase.

One option for creating an event object in this way is to create a sample using the visual editor. You can be sure that the sample is correctly formatted. You can then copy the raw JSON and paste it into a text editor and save the file.

For detailed information about the structure of an event, see Event structure.

Test the function

You can test a function in the CloudFront console or with the Amazon CLI.

Console

In the CloudFront console, you can test a function that you have created using the console.

To test the function

1. Display the **Functions** page in the CloudFront console and choose the function that you want to test.
2. On the function page, choose the **Test** tab. The **Test function** section appears with buttons that include **Edit JSON** and **Test function**.
3. Make sure that the correct event is displayed.

   If you want to switch from the currently displayed event, choose another event in the **Select test event** field.
4. Choose the **Test function** button. The console shows the output of the function, including function logs. It also shows the compute utilization. For more information, see the section called “Understanding compute utilization”.

CLI

You can test a function using the **aws cloudfront test-function** command.

1. Run the command as shown in the example. Run the command from the same directory that contains this file.

   This example uses the fileb:// notation to pass in the event object file. It also includes line breaks to make the command more readable.

```
aws cloudfront test-function \
    --name MaxAge \
    --if-match ETVABCEXAMPLE \
    --event-object fileb://event-maxage-test01.json \
    --stage DEVELOPMENT
```

Notes:

- You reference the function by its name and ETag (in the `if-match` parameter). You reference the event object by its location in your file system.
- The stage can be `DEVELOPMENT` or `LIVE`.

When the command is successful, you see output like the following.

```
TestResult:
  ComputeUtilization: '21'
  FunctionErrorMessage: ''
  FunctionExecutionLogs: []
  FunctionOutput: '{"response":{"headers":{"cloudfront-functions":
{"value":"generated-by-CloudFront-Functions"},"location":{"value":"https://
aws.amazon.com/cloudfront/"}},"statusDescription":"Found","cookies":
{},"statusCode":302}}'
  FunctionSummary:
    FunctionConfig:
      Comment: MaxAge function
      Runtime: cloudfront-js-2.0
      KeyValueStoreAssociations= \
      {Quantity=1, \
      Items=[{KeyValueStoreARN='arn:aws:cloudfront::111122223333:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]} \
    FunctionMetadata:
      CreatedTime: '2021-04-18T20:38:56.915000+00:00'
      FunctionARN: arn:aws:cloudfront::111122223333:function/MaxAge
      LastModifiedTime: '2023-17-20T10:38:57.057000+00:00'
      Stage: DEVELOPMENT
    Name: MaxAge
    Status: UNPUBLISHED
```

Notes

- `FunctionExecutionLogs` contains a list of log lines that the function wrote in `console.log()` statements (if any).
- `ComputeUtilization`. See the section called “Understanding compute utilization”.
- `FunctionOutput` contains the event object that the function returned.

Understanding compute utilization

Compute utilization is the amount of time that the function took to run as a percentage of the maximum allowed time. For example, a value of 35 means that the function completed in 35% of the maximum allowed time.

If a function continuously exceeds the maximum allowed time, CloudFront throttles the function. The following list explains the likelihood of a function getting throttled based on the compute utilization value.

Compute utilization value:

- **1 – 50** – The function is comfortably below the maximum allowed time and should run without throttling.
- **51 – 70** – The function is nearing the maximum allowed time. Consider optimizing the function code.
- **71 – 100** – The function is very close to or exceeds the maximum allowed time. CloudFront is likely to throttle this function if you associate it with a distribution.

Updating functions

You can update a function at any time. The changes are made only to the version of the function that is in the `DEVELOPMENT` stage. You must publish the function to copy the changes from the `DEVELOPMENT` stage to `LIVE`.

You can update a function's code in the CloudFront console or with the Amazon CLI.

Console

To update function code (console)

1. Open the **Functions** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#/functions, and then choose the function that you want to update.
2. Make changes:
   - You can choose the **Edit** button and make change the fields in the **Details** section.

- You can change or remove the associated key value store. Choose the appropriate button. For more information about key value stores, see the section called “Using CloudFront KeyValueStore”.

- You can change the function code. Choose the **Build** tab, make changes, then choose **Save changes** to save only the changes to the code.

CLI

To update function code (CLI)

Run the command as shown in the example.

This example uses the fileb:// notation to pass in the file. It also includes line breaks to make the command more readable.

```
aws cloudfront update-function \
    --name MaxAge \
    --function-config '{"Comment":"Max Age 2 years","Runtime":"cloudfront-
js-2.0","KeyValueStoreAssociations":{"Quantity":1,"Items":
[{"KeyValueStoreARN":"arn:aws:cloudfront::111122223333:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"}]}}' \
    --function-code fileb://function-max-age-v1.js \
    --if-match ETVABCEXAMPLE
```

Notes:

- You identify the function by both its name and its ETag (in the `if-match` parameter). Make sure that you use the current ETag. You can obtain it using a describe operation.
- You must include the `function-code`, even if you don't want to change it.
- Be careful with the `function-config`. You should pass everything that you want to keep in the configuration. Specifically, handle the key value store as follows:
  - If you want to retain the existing key value store association (if there is one), specify the name of the *existing* store.
  - If you want to change the association, specify the name of the *new* key value store.
  - If you want to remove the association, omit the `KeyValueStoreAssociations` parameter.

When the command is successful, you see output like the following.

```
ETag: ETVXYZEXAMPLE
FunctionSummary:
  FunctionConfig:
    Comment: Max Age 2 years \
    Runtime: cloudfront-js-2.0 \
    KeyValueStoreAssociations= \
      {Quantity=1, \
      Items=[{KeyValueStoreARN='arn:aws:cloudfront::111122223333:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]} \
  FunctionMetadata: \
    CreatedTime: '2021-04-18T20:38:56.915000+00:00' \
    FunctionARN: arn:aws:cloudfront::111122223333:function/MaxAge \
    LastModifiedTime: '2023-12-19T23:41:15.389000+00:00' \
    Stage: DEVELOPMENT \
  Name: MaxAge \
  Status: UNPUBLISHED
```

Most of the information is repeated from the request. Other information is added by CloudFront. Note the following:

- `ETag`: This value changes each time you modify the key value store.
- `FunctionARN`
- `Stage`
- `Status`

Publishing functions

Publishing a function copies it from the `DEVELOPMENT` stage to `LIVE`.

> **Important**
>
> When you publish a function, all cache behaviors that are associated with the function automatically start using the newly published copy, as soon as the distributions finish deploying.

If no cache behaviors are associated with the function, publishing it enables you to associate it with a cache behavior. You can only associate cache behaviors with functions that are in the `LIVE` stage.

You can publish a function in the CloudFront console or with the Amazon CLI.

Before you publish, you must test the function.

Console

To publish your function, you can use the CloudFront console. The console also shows the CloudFront distributions that are associated with the function.

To publish a function (console)

1. To publish a function, open the **Functions** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#/functions, and then choose the function that you want to publish.
2. On the function page, choose the **Publish** tab. Then choose the **Publish** button (or, if your function is already attached to one or more cache behaviors, the **Publish and update** button).
3. (Optional) To see the distributions that are associated with the function, choose **Associated CloudFront distributions** to expand that section.

When successful, you see a banner at the top of the page that says ***Function name*** **published successfully**. You can also choose the **Build** tab and then choose **Live** to see the live version of the function code.

CLI

To publish a function, run the **aws cloudfront publish-function** command as shown in the example. In the example, line breaks are provided to make the example more readable.

```
aws cloudfront publish-function \
    --name MaxAge \
    --if-match ETVXYZEXAMPLE
```

When the command is successful, you see output like the following.

```
FunctionSummary:
```

```
FunctionConfig:
  Comment: Max Age 2 years
  Runtime: cloudfront-js-2.0
FunctionMetadata:
  CreatedTime: '2021-04-18T21:24:21.314000+00:00'
  FunctionARN: arn:aws:cloudfront::111122223333:function/ExampleFunction
  LastModifiedTime: '2023-12-19T23:41:15.389000+00:00'
  Stage: LIVE
Name: MaxAge
Status: UNASSOCIATED
```

Associating functions with distributions

To use a function in CloudFront Functions with a CloudFront distribution, you associate the function with one or more cache behaviors in the distribution. You can associate a function with multiple cache behaviors in [multiple distributions]. Before you associate a function, you must [publish it] to the `LIVE` stage.

When you associate a function with a cache behavior, you must choose an *event type*. The event type determines when CloudFront Functions runs the function. There are two event types to choose from:

For more information about event types, see [CloudFront events that can trigger a Lambda@Edge function]. You cannot use origin-facing event types (*origin request* and *origin response*) with CloudFront Functions.

- **Viewer request** – The function runs when CloudFront receives a request from a viewer.
- **Viewer response** – The function runs before CloudFront returns a response to the viewer.

You can associate a function with a distribution in the CloudFront console or with the Amazon CLI.

Console

You can use the CloudFront console to associate a function with an existing cache behavior in an existing CloudFront distribution. For more information about creating a distribution, see [the section called “Creating a distribution”].

To associate a function with an existing cache behavior (console)

1. Open the **Functions** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#/functions, and then choose the name of the function that you want to associate.
2. On the function page, choose the **Publish** tab.
3. Choose **Publish function**.
4. Choose **Add association**. On the dialog that appears, select a distribution, an event type, and/or a cache behavior.

   For the event type, choose when you want this function to run:

   - To run the function every time CloudFront receives a request, choose **Viewer Request**.
   - To run the function every time CloudFront returns a response, choose **Viewer Response**.

   To save the configuration, choose **Add association**.

CloudFront associates the distribution with the function. Wait a few minutes for the associated distribution to finish deploying. You can choose View distribution on the function details page to check the progress.

CLI

You can associate a function with any of the following:

- An existing cache behavior.
- A new cache behavior in an existing distribution.
- A new cache behavior in a new distribution.

The following procedure shows how to associate a function with an existing cache behavior.

To associate a function with an existing cache behavior (Amazon CLI)

1. Use the following command to save the distribution configuration for the distribution whose cache behavior you want to associate with a function. This command saves the distribution configuration to a file named `dist-config.yaml`. To use this command, do the following:

- Replace *DistributionID* with the distribution's ID.
- Run the command on one line. In the example, line breaks are provided to make the example more readable.

```
aws cloudfront get-distribution-config \
    --id DistributionID \
    --output yaml > dist-config.yaml
```

When the command is successful, the Amazon CLI returns no output.

2. Open the file named `dist-config.yaml` that you just created. Edit the file to make the following changes.

   a. Rename the `ETag` field to `IfMatch`, but don't change the field's value.

   b. In the cache behavior, find the object named `FunctionAssociations`. Update this object to add a function association. The YAML syntax for a function association looks like the following example.

      - The following example shows a viewer request event type (trigger). To use a viewer response event type, replace `viewer-request` with `viewer-response`.
      - Replace *arn:aws:cloudfront::111122223333:function/ExampleFunction* with the Amazon Resource Name (ARN) of the function that you're associating with this cache behavior. To get the function ARN, you can use the **aws cloudfront list-functions** command.

```
FunctionAssociations:
  Items:
    - EventType: viewer-request
      FunctionARN: arn:aws:cloudfront::111122223333:function/ExampleFunction
  Quantity: 1
```

After making these changes, save the file.

3. Use the following command to update the distribution, adding the function association. To use this command, do the following:

   - Replace *DistributionID* with the distribution's ID.

- Run the command on one line. In the example, line breaks are provided to make the example more readable.

```
aws cloudfront update-distribution \
    --id DistributionID \
    --cli-input-yaml file://dist-config.yaml
```

When the command is successful, you see output like the following that describes the distribution that was just updated with the function association. The following example output is truncated for readability.

```
Distribution:
  ARN: arn:aws:cloudfront::111122223333:distribution/EBEDLT3BGRBBW
  ... truncated ...
  DistributionConfig:
    ... truncated ...
    DefaultCacheBehavior:
      ... truncated ...
      FunctionAssociations:
        Items:
        - EventType: viewer-request
          FunctionARN: arn:aws:cloudfront::111122223333:function/ExampleFunction
        Quantity: 1
      ... truncated ...
  DomainName: d111111abcdef8.cloudfront.net
  Id: EDFDVBD6EXAMPLE
  LastModifiedTime: '2021-04-19T22:39:09.158000+00:00'
  Status: InProgress
ETag: E2VJGGQEG1JT8S
```

Effect of associating a distribution

The distribution's `Status` changes to `InProgress` while the distribution is redeployed. As soon as the new distribution configuration reaches a CloudFront edge location, that edge location begins using the associated function. When the distribution is fully deployed, the `Status` changes back to `Deployed`, which indicates that the associated CloudFront function is live in all CloudFront edge locations worldwide. This typically takes a few minutes.

Amazon CloudFront KeyValueStore

CloudFront KeyValueStore is a secure, global, low-latency key value datastore that allows read access from within CloudFront Functions, enabling advanced customizable logic at the CloudFront edge locations.

With CloudFront KeyValueStore, you make updates to function code and updates to the data associated with a function independently of each other. This separation simplifies function code and makes it easy to update data without the need to deploy code changes.

The general procedure for using key-value pairs is as follows:

- Create key value stores, and populate it with a set of key-value pairs. You can add your key value stores to an Amazon S3 bucket or enter them manually.
- Associate the key value stores with your CloudFront function.
- Within your function code, use the name of the key to either retrieve the value associated with the key or to evaluate if a key exists. For more information about using key-value pairs in function code, and for information about helper methods, see the section called “Helper methods for key value stores”.

Use cases

Typical use cases for key-value pairs are the following:

- URL rewrites or redirects. The key-value pair could hold the rewritten URLs or the redirect URLs.
- A/B testing and feature flags. You could create a function to run experiments by assigning a percentage of traffic to a specific version of your website.
- Access authorization. You could implement access control to allow or deny requests based on criteria defined by you and the data stored in a key value store.

Supported formats for values

The value in a key-value pair can be stored in any of the following formats:

- A string
- A byte-encoded string
- JSON

Security

The CloudFront function and all its key value stores data are handled securely, as follows:

- CloudFront encrypts each key value stores at rest and during transit (when reading or writing to the key value stores) when you call the CloudFront KeyValueStore API operations.
- When the function is run, CloudFront decrypts each key-value pair in memory at the CloudFront edge locations.

To get started with CloudFront KeyValueStore, see the following topics. You can use the CloudFront console, the CloudFront API, or a supported Amazon SDK.

Topics

- Working with key value store
- Working with key value data

For more information about getting started with CloudFront KeyValueStore, see the Introducing Amazon CloudFront KeyValueStore Amazon blog post.

Working with key value store

You must create a key value store to hold the key-value pairs that you want to use in CloudFront Functions.

You can work with a key-value pair in a key value store in the following ways:

- Using the CloudFront console.
- Using your preferred CloudFront API or SDK.

After you have created the key value stores and added key-value pairs, you can use the key values in your CloudFront function code. The JavaScript runtime 2.0 includes some helper methods for working with key values in the function code. For more information, see the section called “Helper methods for key value stores”.

Topics

- Creating key value stores
- Associating a key value store with a function

- Modifying a key value stores
- Deleting a key value store
- Obtaining a reference to a key value store
- Creating a file of key-value pairs

Creating key value stores

You can create an empty key value stores then add key-value pairs later. Or you can create a key value stores and its key-value pairs at the same time.

> **Note**
>
> If you specify your data source from an Amazon S3 bucket, you must have the `s3:GetObject` and `s3:GetBucketLocation` permissions to that bucket. If you don't have these permissions, CloudFront can't successfully create your key value store.

Console

To create key value stores (console)

1. Decide if you want to add key-value pairs at the same time as you create the key value stores. (You can also add key-value pairs later.) This import feature is supported on both the CloudFront console and with CloudFront APIs and SDKs. But it is supported only when you initially create the key value stores.

   If you want to use a file, create it now.
2. Sign in to the Amazon Web Services Management Console and open the **Functions** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#/functions.
3. Choose the **KeyValueStores** tab. Choose Create KeyValueStore.
4. Enter a name and optional description for the key value stores.
5. Complete **S3 URI**:

   - If you prepared a file of key-value pairs, enter the path to the Amazon S3 bucket where you stored the file.
   - Leave this field blank if you plan to enter the key value pairs manually.

6. Choose **Create**. The key value store now exists.

   The details page for the new key value stores appears. The information on the page includes the ID and the ARN of the key value store.

   - The ID is a random string of characters that is unique in your Amazon account.
   - The ARN has this syntax:

     ```
     Amazon account:key-value-store/the key value stores ID
     ```

7. Look at the **Key value pairs** section. If you imported a file, this section shows some pairs. Otherwise it is empty. You can do the following:

   - If you didn't import a file from an Amazon S3 bucket, and if you want to add key-value pairs now, you can complete this section.
   - If you did import a file, you can also add more values manually.
   - You can leave this section empty, and add the pairs later, by editing the key value stores.

   To add the pairs now:

   - Choose the **Add key-value pairs** button.
   - Choose **Add pair** and enter a name and value.
   - Choose the **Add pair** button again, to add more pairs.

     When you have finished, choose **Save** changes to save all the pairs in the key value store. On the confirmation dialog that appears, choose **Done**.

8. Complete the **Associated functions** section if you want to associate the key value stores with a function now. You can also create this association later, either from this key value stores details page, or from the functions details page.

   To create the association now, choose the **Go to functions** button. For more information, see ??? or ???.

Programmatically

1. Decide if you want to add key-value pairs at the same time as you create the key value stores. (You can also add key-value pair later.) This import feature is supported on both the

CloudFront console and with CloudFront APIs and SDKs. But it is supported only when you initially create the key value stores.

If you want to use a file, create it now.

2. Use the create operation of your preferred CloudFront API or SDK. For example, for the REST API, use CloudFront.CreateKeyValueStore. The operation takes several parameters:
   - A name.
   - A `configuration` parameter that includes a comment.
   - An `import-source` parameter that lets you import key-value pairs from a file that is stored in an Amazon S3 bucket. Note that you can import from a file only on initial creation of the key value stores. For information about the format of the file, see the section called "Creating a file of key-value pairs".

The operation response includes the following information:

- The values passed in the request, including the name that you assigned.
- Data such as the creation time.
- An ETag (for example, ETVABCEXAMPLE2), the ARN that includes the name of the key value stores (for example, `arn:aws:cloudfront::111122223333:key-value-store/MaxAge`).

  You will use some combination of the ETag, the ARN, and the name to work with the key value stores programmatically.

Key value store statuses

When you create a key value store, the data store can have the following status values.

Value	Description
Provisioning	The key value store was created and CloudFront is processing the data source that you specified.
Ready	The key value store was created and CloudFront successfully processed the data source that you specified.

Value	Description
Import failed	CloudFront wasn't able to process the data source that you specified. This status can appear if your file format isn't valid or that it exceeds the size limit. For more information, see Creating a file of key-value pairs.

Associating a key value store with a function

You associate a key value store with a function by working in the function. You must make this association in order to use the key-value pairs from that store in that function. The following rules apply:

- One function can have one key value store.
- One key value store can be associated with multiple functions.

You can work with the association in the following ways.

- You can create an association between a function and a key value store:
  - On the CloudFront console, view the key value stores details page and choose the **Go to functions** button. The appropriate page appears — the **Functions** list (if there is currently no associated function) or the function details page (if there is currently an association). For more information, see the section called “Associating a key value store with a function”.
  - Programmatically, use the function update operation of your preferred CloudFront API or SDK.

  After you create the association (or if you change the association), you should test the function, and you *must* republish the function.

- If you modify a key value stores without changing the key-value pairs, you don't need to renew the association (which means that you don't need to publish again). But you should test the function.
- If you change the key-value pairs in the key value stores, you don't need to renew the association (which means that you don't need to publish again). But you should test the function to verify that it works with the changes to the key-value pairs.
- You can view all the functions that use a specific key value stores. On the CloudFront console, look at the key value stores details page.

Modifying a key value stores

You can work with the key-value pairs, and you can change the association between the key value stores and the function.

Using the CloudFront console

To modify a key value store by using the console

1. Sign in to the Amazon Web Services Management Console and open the **Functions** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#/functions.
2. Choose the **KeyValueStores** tab. Select the key value store that you want to change. The details page appears.
   - To work with the key-value pairs, choose the **Edit** button in the **Key value pairs** section. You can add more key-value pairs, you can delete any key-value pair, and you can change the value for an existing key-value pair. When you have finished, choose **Save changes**.
   - To work with the association for this key value stores, choose the **Go to functions** button. The appropriate page appears — the **Functions** list (if there is currently no associated function) or the function details page (if there is currently an association). For more information, see the section called “Associating a key value store with a function”.

Modifying a store programmatically

You can work with the key value stores in the following ways.

Change the key-value pairs

You can add more key-value pairs, you can delete one or more key-value pairs, and you can change the value of an existing key-value pair. For more information, see the section called “Working programmatically”.

Change the function association for the key value stores

To work with the association for this key value stores, see the section called “Updating functions”. You will need the ARN of the key value stores. For more information, see the section called “Obtaining a reference to a key value store”.

Deleting a key value store

You can delete your key value store by using the CloudFront console or API.

To delete a key value store by using the console

1. Sign in to the Amazon Web Services Management Console and open the **Functions** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#/functions.
2. Verify if the key value stores is associated with a function. If it is, remove the association. For more information on both these steps, see ???
3. Choose the **KeyValueStores** tab. Select the key value store that you want to change and then choose **Delete**.

To delete a key value store programatically

1. Obtain the ETag and the name of the key value stores. For more information, see the section called “Obtaining a reference to a key value store”.
2. Verify if the key value stores is associated with a function. If it is, remove the association. For more information on both these steps, see ???.
3. To delete the key value stores, use the delete operation of your preferred CloudFront API or SDK. For example, for the REST API, use CloudFront.DeleteKeyValueStore.

Obtaining a reference to a key value store

To work with the key value stores programmatically, you need the ETag and the name of the key value store. To obtain this data, follow these steps:

1. Use the list operation of your preferred CloudFront API or SDK. For example, for the REST API, use CloudFront.ListKeyValueStores. The response includes a list of key value stores. Find the name of the key value store you want to change.
2. Use the describe operation of your preferred CloudFront API or SDK. For example, for the REST API, use CloudFront.DescribeKeyValueStore. Pass in the name you obtained.

   For important information about the describe operation, see the section called “About CloudFront KeyValueStore”.

The response includes a UUID, the ARN of the key value stores, and the ETag of the key value stores.

- The UUID is 128 bits. For example, `a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`

- The ARN includes the Amazon account number, the constant `key-value-store`, and the UUID. For example:

  `arn:aws:cloudfront::111122223333:key-value-store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`

- An ETag looks like this: `ETVABCEXAMPLE2`

Creating a file of key-value pairs

When you create a UTF-8 encoded file, use the following JSON format:

```
{
  "data":[
    {
      "key":"key1",
      "value":"value"
    },
    {
      "key":"key2",
      "value":"value"
    }
  ]
}
```

Your file can't include duplicate keys. If you specified an invalid file in your Amazon S3 bucket, you can update the file to remove any duplicates and then try creating your key value store again.

For more information, see Creating key value stores.

Note

The file for your data source and its key-value pairs have the following limits:

- File size – 5 MB
- Key size – 512 characters
- Value size – 1024 characters

Working with key value data

You can work with key-value pairs in an existing key value stores in these ways:

- Using the CloudFront console.
- Using your preferred CloudFront KeyValueStore API or SDK.

> **Note**
>
> This is a different API or SDK from the regular CloudFront API or SDK.

This section describes how to add key-value pairs to an existing key value stores. To include key-value pairs when you initially create the key value stores, see the section called "Creating key value stores".

Topics

- Working with key-value pairs by using the CloudFront console
- Working with key-value pairs programmatically

Working with key-value pairs by using the CloudFront console

You can use the CloudFront console to work with your key-value pairs.

To use the CloudFront console to work with key-value pairs

1. Sign in to the Amazon Web Services Management Console and open the **Functions** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home#/functions.
2. Choose the **KeyValueStores** tab. Select the key value store that you want to change. The details page appears.
3. In the **Key value pairs** section, choose **Edit**.
4. You can add a key-value pair, delete a key-value pair, or change the value for an existing key-value pair.
5. When you have finished, choose **Save changes**.

Working with key-value pairs programmatically

Topics

- Obtaining a reference to a key value store
- Changing key-value pairs in a key value stores
- About CloudFront KeyValueStore
- Example code for CloudFront KeyValueStore

Obtaining a reference to a key value store

When you enter a write operation using CloudFront KeyValueStore, you need to pass in the ARN and the ETag of the key value stores. To obtain this data, do the following:

1. Use the list operation of your preferred CloudFront API or SDKs. For example, for the REST API, use CloudFront.ListKeyValueStores. The response includes a list of key value stores. Find the name of the key value store you want to change.
2. Use the describe operation of your preferred CloudFront KeyValueStore API or SDK. For example, for the REST API, use CloudFrontKeyValueStore.DescribeKeyValueStore. Pass in the name you obtained in the previous step.

   > **Note**
   >
   > Use the operation from the CloudFront KeyValueStore API, not from the CloudFront API. For more information, see the section called “About CloudFront KeyValueStore”.

   The response includes the ARN and the ETag of the key value stores.

   - The ARN includes the Amazon Web Services account number, the constant `key-value-store`, and the UUID. For example:

     ```
     arn:aws:cloudfront::111122223333:key-value-store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
     ```

   - An ETag looks like this: `ETVABCEXAMPLE2`

Changing key-value pairs in a key value stores

You can work with the key-value pairs using the following operations of your preferred CloudFront KeyValueStore API or SDK. All of these operations work on one specified key value stores:

- `CloudFrontKeyValueStore.DeleteKey`: Delete one key. See DeleteKey.
- `CloudFrontKeyValueStore.GetKey`: Get one key. See GetKey.
- `CloudFrontKeyValueStore.ListKeys`: List the keys. See ListKeys.
- `CloudFrontKeyValueStore.PutKey`: You can perform two actions:
  - Create a new key-value pair in one key value stores: In this case, pass a new key name and value.
  - Set a different value in one existing key-value pair: In this case, pass an existing key name, and a new key value.

  See PutKey.
- `CloudFrontKeyValueStore.UpdateKeys`: You can perform one or more of the following actions in one all-or-nothing operation:
  - Delete one or more key-value pairs.
  - Create one or more new key-value pairs.
  - Set a different value in one or more existing key value pairs.

  See UpdateKeys.

About CloudFront KeyValueStore

To work with key-value pairs programmatically in an *existing* key value stores, you use the CloudFront KeyValueStore service.

To include some key-value pairs in the key value stores when you initially create the key value stores, you use the CloudFront service.

The describe operation

Both the CloudFront API and the CloudFront KeyValueStore API have a describe operation that returns data about the key value stores:

- The CloudFront API provides data such as the status and the date that the store itself was last modified.
- The CloudFront KeyValueStore API provides data about the *contents* of the storage resource — the key-value pairs in the store, and the size of the contents.

The describe operations in the two APIs return slightly different data that identifies the key value stores:

- The describe operation in the CloudFront API returns an ETag, the UUID, and the ARN of the key value stores.

- The describe operation in the CloudFront KeyValueStore API returns an ETag and the ARN of the key value stores.

> **Note**
>
> Each describe operation returns a *different* ETag. The ETags are not interchangeable. When you perform an operation in one of the APIs, you must pass in the ETag from the appropriate API. For example, in the delete operation in CloudFront KeyValueStore, pass in the ETag that you obtained from the describe operation in CloudFront KeyValueStore.

Example code for CloudFront KeyValueStore

Example : Calling the `DescribeKeyValueStore` API operation

The following sample code shows you how to call the `DescribeKeyValueStore` API operation for a key value store.

```
const {
  CloudFrontKeyValueStoreClient,
  DescribeKeyValueStoreCommand,
} = require("@aws-sdk/client-cloudfront-keyvaluestore");

require("@aws-sdk/signature-v4-crt");

(async () => {
  try {
```

```
    const client = new CloudFrontKeyValueStoreClient({
      region: "us-east-1"
    });
    const input = {
      KvsARN: "arn:aws:cloudfront::123456789012:key-value-store/a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111",
    };
    const command = new DescribeKeyValueStoreCommand(input);

    const response = await client.send(command);
  } catch (e) {
    console.log(e);
  }
})();
```

Customizing at the edge with Lambda@Edge

Lambda@Edge is an extension of Amazon Lambda. Lambda@Edge is a compute service that lets you execute functions that customize the content that CloudFront delivers. You can author Node.js or Python functions in one Amazon Web Services Region, US East (N. Virginia), and then execute them in Amazon locations globally that are closer to the viewer, without provisioning or managing servers. Lambda@Edge scales automatically, from a few requests per day to thousands per second. Processing requests at Amazon locations closer to the viewer instead of on origin servers significantly reduces latency and improves the user experience.

When you associate a CloudFront distribution with a Lambda@Edge function, CloudFront intercepts requests and responses at CloudFront edge locations. You can execute Lambda functions when the following CloudFront events occur:

- When CloudFront receives a request from a viewer (viewer request)
- Before CloudFront forwards a request to the origin (origin request)
- When CloudFront receives a response from the origin (origin response)
- Before CloudFront returns the response to the viewer (viewer response)

If you're using Amazon WAF, the Lambda@Edge viewer request is executed after any Amazon WAF rules are applied.

There are many uses for Lambda@Edge processing. For example:

- A Lambda function can inspect cookies and rewrite URLs so that users see different versions of a site for A/B testing.
- CloudFront can return different objects to viewers based on the device they're using by checking the `User-Agent` header, which includes information about the devices. For example, CloudFront can return different images based on the screen size of their device. Similarly, the function could consider the value of the `Referer` header and cause CloudFront to return the images to bots that have the lowest available resolution.
- Or you could check cookies for other criteria. For example, on a retail website that sells clothing, if you use cookies to indicate which color a user chose for a jacket, a Lambda function can change the request so that CloudFront returns the image of a jacket in the selected color.
- A Lambda function can generate HTTP responses when CloudFront viewer request or origin request events occur.
- A function can inspect headers or authorization tokens, and insert a header to control access to your content before CloudFront forwards the request to your origin.
- A Lambda function can also make network calls to external resources to confirm user credentials, or fetch additional content to customize a response.

For sample code and additional examples, see Lambda@Edge example functions.

Topics

- Get started creating and using Lambda@Edge functions
- Setting IAM permissions and roles for Lambda@Edge
- Writing and creating a Lambda@Edge function
- Adding triggers for a Lambda@Edge function
- Testing and debugging Lambda@Edge functions
- Deleting Lambda@Edge functions and replicas
- Lambda@Edge event structure
- Working with requests and responses
- Lambda@Edge example functions

Get started creating and using Lambda@Edge functions

With Lambda@Edge, you can use CloudFront triggers to invoke a Lambda function. When you associate a CloudFront distribution with a Lambda function, CloudFront intercepts requests and responses at CloudFront edge locations and runs the function. Lambda functions can improve security or customize information close to your viewers to improve performance.

The following list provides a basic overview of how to create and use Lambda functions with CloudFront. For a step-by-step tutorial, see Tutorial: Creating a simple Lambda@Edge function.

1. In the Amazon Lambda console, create a Lambda function in the US East (N. Virginia) Region. (Or you can create the function programmatically by using one of the Amazon SDKs.)
2. Save and publish a numbered version of the function.

   If you want to change the function, you must edit the $LATEST version of the function in the US East (N. Virginia) Region. Then, before you set it up to work with CloudFront, you publish a new numbered version.
3. Associate the function with a CloudFront distribution and cache behavior. Then specify one or more CloudFront events (*triggers*) that cause the function to execute. For example, you can create a trigger for the function to execute when CloudFront receives a request from a viewer.
4. When you create a trigger, Lambda creates replicas of the function at Amazon locations around the world.

> **Tip**
>
> Learn more about how you can use Lambda@Edge for your own custom solutions. Learn more about creating and updating functions, the event structure, and adding CloudFront triggers. You can also find more ideas and get code samples in Lambda@Edge example functions.

Topics

- Tutorial: Creating a simple Lambda@Edge function

Tutorial: Creating a simple Lambda@Edge function

This tutorial shows you how to get started with Lambda@Edge by helping you create and add a sample Node.js function that runs in CloudFront. The example that we walk through adds HTTP security headers to a response, which can improve security and privacy for a website. You don't need a website for this walkthrough. In it, we simply add security headers to a response when CloudFront retrieves a file.

This example describes the steps to create and configure a Lambda@Edge function. When you create your own Lambda@Edge solution, you follow similar steps and choose from the same options.

Topics

- Step 1: Sign up for an Amazon Web Services account
- Step 2: Create a CloudFront distribution
- Step 3: Create your function
- Step 4: Add a CloudFront trigger to run the function
- Step 5: Verify that the function runs
- Step 6: Troubleshoot issues
- Step 7: Clean up your example resources
- Resources for learning more

Step 1: Sign up for an Amazon Web Services account

If you haven't already done so, sign up for an Amazon Web Services account. For more information, see Sign up for an Amazon Web Services account.

Step 2: Create a CloudFront distribution

Before you create the example Lambda@Edge function, you must have a CloudFront environment to work with that includes an origin to serve content from.

For this example, you create a CloudFront distribution that uses an Amazon S3 bucket as the origin for the distribution. If you already have an environment to use, you can skip this step.

To create a CloudFront distribution with an Amazon S3 origin

1. Create an Amazon S3 bucket with a file or two, such as image files, for sample content. For help, follow the steps in Upload your content to Amazon S3. Make sure that you set permissions to grant public read access to the objects in your bucket.
2. Create a CloudFront distribution and add your S3 bucket as an origin, by following the steps in Create a CloudFront web distribution. If you already have a distribution, you can add the bucket as an origin for that distribution instead.

> **Tip**
>
> Make a note of your distribution ID. Later in this tutorial when you add a CloudFront trigger for your function, you must choose the ID for your distribution in a dropdown list—for example, `E653W22221KDDL`.

Step 3: Create your function

In this step, you create a Lambda function from a blueprint template in the Lambda console. The function adds code to update security headers in your CloudFront distribution.

To create a Lambda function

1. Sign in to the Amazon Web Services Management Console and open the Amazon Lambda console at https://console.amazonaws.cn/lambda/.

> **Important**
>
> Make sure that you're in the **US-East-1 (N. Virginia)** Region (**us-east-1**). You must be in this Region to create Lambda@Edge functions.

2. Choose **Create function**.
3. On the **Create function** page, choose **Use a blueprint**, and then filter for the CloudFront blueprints by entering **`cloudfront`** in the search field.

> **Note**
>
> CloudFront blueprints are available only in the **US-East-1 (N. Virginia)** Region (**us-east-1**).

4. Choose the **Modify HTTP response header** blueprint as the template for your function.
5. Enter the following information about your function:

   Function name

   Enter a name for your function.

   Execution role

   Choose how to set the permissions for your function. To use the recommended basic Lambda@Edge permissions policy template, choose **Create a new role from Amazon policy templates**.

   Role name

   Enter a name for the role that the policy template creates.

   Policy templates

   Lambda automatically adds the policy template **Basic Lambda@Edge permissions** because you chose a CloudFront blueprint as the basis for your function. This policy template adds execution role permissions that allow CloudFront to run your Lambda function for you in CloudFront locations around the world. For more information, see Setting IAM permissions and roles for Lambda@Edge.
6. Choose **Create function**.
7. In the **Deploy to Lambda@Edge** pane that appears, choose **Cancel**. (For this tutorial, you must modify the function code before deploying the function to Lambda@Edge.)
8. Scroll down to the **Code source** section of the page.
9. Replace the template code with a function that modifies the security headers that your origin returns. For example, you could use code similar to the following:

```
'use strict';
exports.handler = (event, context, callback) => {

    //Get contents of response
```

```
    const response = event.Records[0].cf.request;
    const headers = response.headers;

    //Set new headers
    headers['strict-transport-security'] = [{key: 'Strict-Transport-Security',
 value: 'max-age= 63072000; includeSubdomains; preload'}];
    headers['content-security-policy'] = [{key: 'Content-Security-Policy', value:
 "default-src 'none'; img-src 'self'; script-src 'self'; style-src 'self'; object-
src 'none'"}];
    headers['x-content-type-options'] = [{key: 'X-Content-Type-Options', value:
 'nosniff'}];
    headers['x-frame-options'] = [{key: 'X-Frame-Options', value: 'DENY'}];
    headers['x-xss-protection'] = [{key: 'X-XSS-Protection', value: '1;
 mode=block'}];
    headers['referrer-policy'] = [{key: 'Referrer-Policy', value: 'same-origin'}];

    //Return modified response
    callback(null, response);
};
```

10. Choose **File**, **Save** to save your updated code.

Proceed to the next section to add a CloudFront trigger to run the function.

Step 4: Add a CloudFront trigger to run the function

Now that you have a Lambda function to update security headers, configure the CloudFront trigger to run your function to add the headers in any response that CloudFront receives from the origin for your distribution.

To configure the CloudFront trigger for your function

1. In the Lambda console, on the **Function overview** page for your function, choose **Add trigger**.
2. For **Trigger configuration**, choose **CloudFront**.
3. Choose **Deploy to Lambda@Edge**.
4. In the **Deploy to Lambda@Edge** pane, under **Configure CloudFront trigger**, enter the following information:

Distribution

The CloudFront distribution ID to associate with your function. In the dropdown list, choose the distribution ID.

Cache behavior

The cache behavior to use with the trigger. For this example, leave the value set to *, which means your distribution's default cache behavior. For more information, see Cache behavior settings in the Values that you specify when you create or update a distribution topic.

CloudFront event

The trigger that specifies when your function runs. We want the security headers function to run whenever CloudFront returns a response from the origin. So in the dropdown list, choose **Origin response**. For more information, see Adding triggers for a Lambda@Edge function.

5. Select the **Confirm deploy to Lambda@Edge** check box.
6. Choose **Deploy** to add the trigger and replicate the function to Amazon locations worldwide.
7. Wait for the function to replicate. This typically takes several minutes.

   You can check to see if replication is finished by going to the CloudFront console and viewing your distribution. Wait for the distribution status to change from **Deploying** to a date and time, which means that your function has been replicated. To verify that the function works, follow the steps in the next section.

Step 5: Verify that the function runs

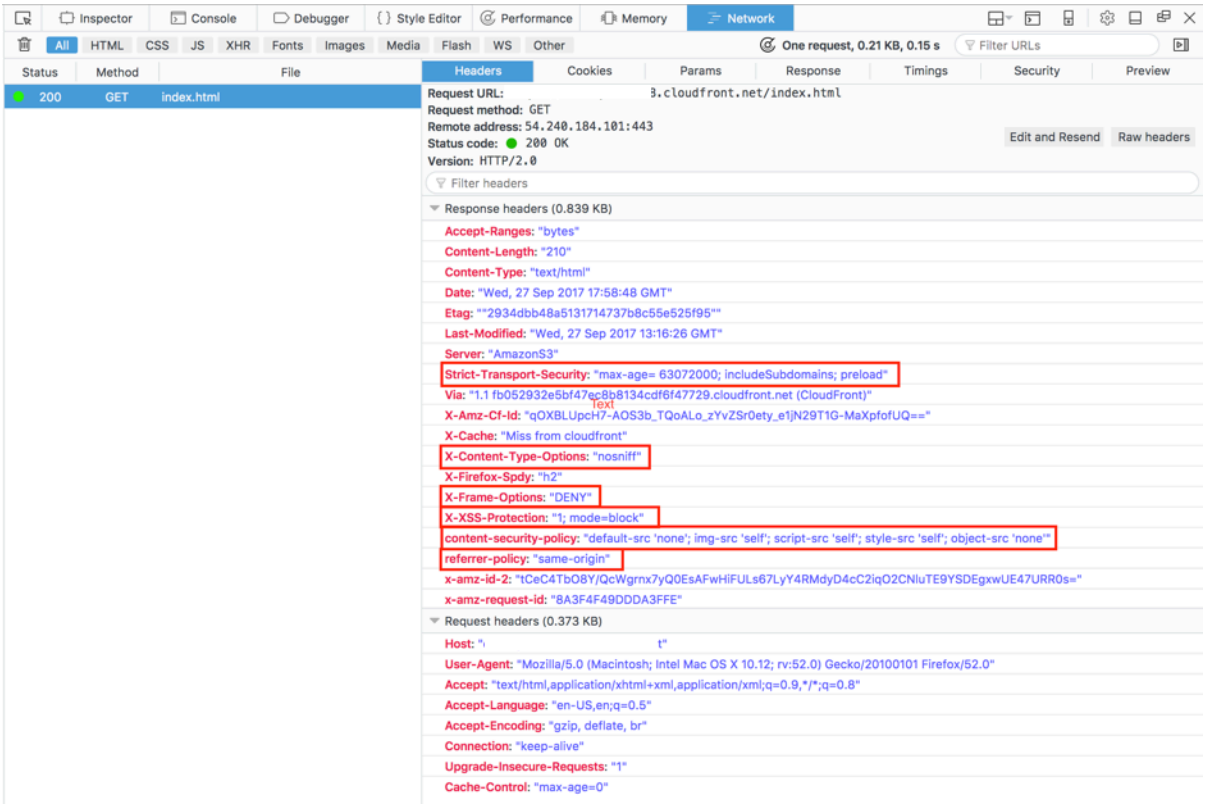
Now that you've created your Lambda function and configured a trigger to run it for a CloudFront distribution, check to make sure that the function is accomplishing what you expect it to. In this example, we check the HTTP headers that CloudFront returns, to make sure that the security headers are added.

To verify that your Lambda@Edge function adds security headers

1. In a browser, enter the URL for a file in your S3 bucket. For example, you might use a URL similar to `https://d111111abcdef8.cloudfront.net/image.jpg`.

For more information about the CloudFront domain name to use in the file URL, see [Customizing the URL format for files in CloudFront](#).

2. Open your browser's Web Developer toolbar. For example, in your browser window in Chrome, open the context (right-click) menu, and then choose **Inspect**.
3. Choose the **Network** tab.
4. Reload the page to view your image, and then choose an HTTP request on the left pane. You see the HTTP headers displayed in a separate pane.
5. Look through the list of HTTP headers to verify that the expected security headers are included in the list. For example, you might see headers similar to those shown in the following screenshot.



If the security headers are included in your headers list, great! You've successfully created your first Lambda@Edge function. If CloudFront returns errors or there are other issues, continue to the next step to troubleshoot the issues.

Step 6: Troubleshoot issues

If CloudFront returns errors or doesn't add the security headers as expected, you can investigate your function's execution by looking at CloudWatch Logs. Be sure to use the logs stored in the Amazon location that is closest to the location where the function is executed.

For example, if you view the file from London, try changing the Region in the CloudWatch console to Europe (London).

To examine CloudWatch logs for your Lambda@Edge function

1. Sign in to the Amazon Web Services Management Console and open the CloudWatch console at https://console.amazonaws.cn/cloudwatch/.
2. Change **Region** to the location that is shown when you view the file in your browser. This is where the function is executing.
3. In the left pane, choose **Logs** to view the logs for your distribution.

For more information, see Monitoring CloudFront metrics with Amazon CloudWatch.

Step 7: Clean up your example resources

If you created an Amazon S3 bucket and CloudFront distribution just for this tutorial, delete the Amazon resources that you allocated so that you no longer accrue charges. After you delete your Amazon resources, any content that you added is no longer available.

Tasks

- Delete the S3 bucket
- Delete the Lambda function
- Delete the CloudFront distribution

Delete the S3 bucket

Before you delete your Amazon S3 bucket, make sure that logging is disabled for the bucket. Otherwise, Amazon continues to write logs to your bucket as you delete it.

To disable logging for a bucket

1. Open the Amazon S3 console at https://console.amazonaws.cn/s3/.

2. Select your bucket, and then choose **Properties**.
3. From **Properties**, choose **Logging**.
4. Clear the **Enabled** check box.
5. Choose **Save**.

Now, you can delete your bucket. For more information, see Deleting a bucket in the *Amazon Simple Storage Service Console User Guide*.

Delete the Lambda function

For instructions to delete the Lambda function association and optionally the function itself, see Deleting Lambda@Edge functions and replicas.

Delete the CloudFront distribution

Before you delete a CloudFront distribution, you must disable it. A disabled distribution is no longer functional and does not accrue charges. You can enable a disabled distribution at any time. After you delete a disabled distribution, it's no longer available.

To disable and delete a CloudFront distribution

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Select the distribution that you want to disable, and then choose **Disable**.
3. When prompted for confirmation, choose **Yes, Disable**.
4. Select the disabled distribution, and then choose **Delete**.
5. When prompted for confirmation, choose **Yes, Delete**.

Resources for learning more

Now that you have a basic idea of how Lambda@Edge functions work, learn more by reading the following:

- Lambda@Edge example functions
- Lambda@Edge Design Best Practices
- Reducing Latency and Shifting Compute to the Edge with Lambda@Edge

Setting IAM permissions and roles for Lambda@Edge

To configure Lambda@Edge, you must set up specific IAM permissions and an IAM execution role. Lambda@Edge also creates service-linked roles to replicate Lambda functions to CloudFront Regions and to enable CloudWatch to use CloudFront log files.

Topics

- IAM permissions required to associate Lambda@Edge functions with CloudFront distributions
- Function execution role for service principals
- Service-linked roles for Lambda@Edge

IAM permissions required to associate Lambda@Edge functions with CloudFront distributions

In addition to the IAM permissions that you need to use Amazon Lambda, the user needs the following IAM permissions to associate Lambda functions with CloudFront distributions:

- `lambda:GetFunction`

  Allows the user to get configuration information for the Lambda function and a presigned URL to download a .zip file that contains the function.

  For the resource, specify the ARN of the function version that you want to execute when a CloudFront event occurs, as shown in the following example:

  `arn:aws:lambda:us-east-1:123456789012:function:TestFunction:2`

- `lambda:EnableReplication*`

  Adds a permission to the resource policy that gives the Lambda replication service permission to get function code and configuration.

  > **Important**
  >
  > The asterisk (*) at the end of the permission is required: `lambda:EnableReplication*`

  For the resource, specify the ARN of the function version that you want to execute when a CloudFront event occurs, as shown in the following example:

`arn:aws:lambda:us-east-1:123456789012:function:TestFunction:2`

- `lambda:DisableReplication*`

  Adds a permission to the resource policy that gives the Lambda replication service permission to allow the function to be deleted.

  > **Important**
  >
  > The asterisk (*) at the end of the permission is required: `lambda:DisableReplication*`

  For the resource, specify the ARN of the function version that you want to execute when a CloudFront event occurs, as shown in the following example:

  `arn:aws:lambda:us-east-1:123456789012:function:TestFunction:2`

- `iam:CreateServiceLinkedRole`

  Allows the user to create a service linked role that is used by Lambda@Edge to replicate Lambda functions in CloudFront. After this role has been created by the first distribution you use with Lambda@Edge, you don't need to add permission to other distributions that you use with Lambda@Edge.

- `cloudfront:UpdateDistribution` or `cloudfront:CreateDistribution`

  Use `cloudfront:UpdateDistribution` to update a distribution or `cloudfront:CreateDistribution` to create a distribution.

For more information, see the following documentation:

- Identity and Access Management for Amazon CloudFront in this guide.
- Authentication and Access Control for Amazon Lambda in the *Amazon Lambda Developer Guide*

Function execution role for service principals

You must create an IAM role that can be assumed by the service principals `lambda.amazonaws.com` and `edgelambda.amazonaws.com`. This role is assumed by the service

principals when they execute your function. For more information, see Creating roles and attaching policies (console) in the *IAM User Guide*.

You add this role under the **Trust Relationship** tab in IAM (do not add it under the **Permissions** tab).

Here's an example role trust policy:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "Service": [
               "lambda.amazonaws.com",
               "edgelambda.amazonaws.com"
            ]
         },
         "Action": "sts:AssumeRole"
      }
   ]
}
```

For information about the permissions that you need to grant to the execution role, see Manage Permissions: Using an IAM Role (Execution Role) in the *Amazon Lambda Developer Guide*. Note the following:

- By default, whenever a CloudFront event triggers a Lambda function, data is written to CloudWatch Logs. If you want to use these logs, the execution role needs permission to write data to CloudWatch Logs. You can use the predefined AWSLambdaBasicExecutionRole to grant permission to the execution role.

  For more information about CloudWatch Logs, see the section called “Edge function logs”.
- If your Lambda function code accesses other Amazon resources, such as reading an object from an S3 bucket, the execution role needs permission to perform that operation.

Service-linked roles for Lambda@Edge

Lambda@Edge uses Amazon Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to a service. Service-linked roles are predefined by the service and include all of the permissions that the service requires to call other Amazon services on your behalf.

Lambda@Edge uses the following IAM service-linked role:

- **AWSServiceRoleForLambdaReplicator** – Lambda@Edge uses this role to allow Lambda@Edge to replicate functions to Amazon Web Services Regions.
- **AWSServiceRoleForCloudFrontLogger** – CloudFront uses this role to push log files into your CloudWatch account, to help you to debug Lambda@Edge validation errors.

When you first add a Lambda@Edge trigger in CloudFront, a role named AWSServiceRoleForLambdaReplicator is automatically created to allow Lambda@Edge to replicate functions to Amazon Web Services Regions. This role is required for using Lambda@Edge functions. The ARN for the AWSServiceRoleForLambdaReplicator role looks like this:

```
arn:aws:iam::123456789012:role/aws-service-role/
replicator.lambda.amazonaws.com/AWSServiceRoleForLambdaReplicator
```

The second role, named AWSServiceRoleForCloudFrontLogger, is created automatically when you add Lambda@Edge function association to allow CloudFront to push Lambda@Edge error log files to CloudWatch. The ARN for the AWSServiceRoleForCloudFrontLogger role looks like this:

```
arn:aws:iam::account_number:role/aws-service-role/
logger.cloudfront.amazonaws.com/AWSServiceRoleForCloudFrontLogger
```

A service-linked role makes setting up and using Lambda@Edge easier because you don’t have to manually add the necessary permissions. Lambda@Edge defines the permissions of its service-linked roles, and only Lambda@Edge can assume the roles. The defined permissions include the trust policy and the permissions policy. The permissions policy cannot be attached to any other IAM entity.

You must remove any associated CloudFront or Lambda@Edge resources before you can delete a service-linked role. This helps protect your Lambda@Edge resources by making sure that you don't remove a service-linked role that is still required to access active resources.

For information about other services that support service-linked roles, see Amazon services that work with IAM and look for the services that have **Yes** in the **Service-linked roles** column.

Service-linked role permissions for Lambda@Edge

Lambda@Edge uses two service-linked roles, named **AWSServiceRoleForLambdaReplicator** and **AWSServiceRoleForCloudFrontLogger**. The following sections describe the permissions for each of these roles.

Service-linked role permissions for Lambda replicator

This service-linked role allows Lambda to replicate Lambda@Edge functions to Amazon Web Services Regions.

The AWSServiceRoleForLambdaReplicator service-linked role trusts the following service to assume the role: `replicator.lambda.amazonaws.com`

The role permissions policy allows Lambda@Edge to complete the following actions on the specified resources:

- Action: `lambda:CreateFunction` on `arn:aws:lambda:*:*:function:*`
- Action: `lambda:DeleteFunction` on `arn:aws:lambda:*:*:function:*`
- Action: `lambda:DisableReplication` on `arn:aws:lambda:*:*:function:*`
- Action: `iam:PassRole` on `all Amazon resources`
- Action: `cloudfront:ListDistributionsByLambdaFunction` on `all Amazon resources`

Service-linked role permissions for CloudFront logger

This service-linked role allows CloudFront to push log files into your CloudWatch account, to help you to debug Lambda@Edge validation errors.

The AWSServiceRoleForCloudFrontLogger service-linked role trusts the following service to assume the role: `logger.cloudfront.amazonaws.com`

The role permissions policy allows Lambda@Edge to complete the following actions on the specified resources:

- Action: `logs:CreateLogGroup` on `arn:aws:logs:*:*:log-group:/aws/cloudfront/*`
- Action: `logs:CreateLogStream` on `arn:aws:logs:*:*:log-group:/aws/cloudfront/*`
- Action: `logs:PutLogEvents` on `arn:aws:logs:*:*:log-group:/aws/cloudfront/*`

You must configure permissions to allow an IAM entity (such as a user, group, or role) to delete the Lambda@Edge service-linked roles. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating service-linked roles for Lambda@Edge

You don't typically manually create the service-linked roles for Lambda@Edge. The service creates the roles for you automatically in the following scenarios:

- When you first create a trigger, the service creates a role, AWSServiceRoleForLambdaReplicator, if the role doesn't already exist, that allows Lambda to replicate Lambda@Edge functions to Amazon Web Services Regions.

  If you delete the service-linked role, the role will be created again when you add a new trigger for Lambda@Edge in a distribution.
- When you update or create a CloudFront distribution that has a Lambda@Edge association, the service creates a role, AWSServiceRoleForCloudFrontLogger, if the role doesn't already exist, that allows CloudFront to push your log files to CloudWatch.

  If you delete the service-linked role, the role will be created again when you update or create a CloudFront distribution that has a Lambda@Edge association.

If you must manually create these service-linked roles, run the following commands using the Amazon CLI:

To create the AWSServiceRoleForLambdaReplicator role

```
aws iam create-service-linked-role --aws-service-name
replicator.lambda.amazonaws.com
```

To create the AWSServiceRoleForCloudFrontLogger role

```
aws iam create-service-linked-role --aws-service-name
logger.cloudfront.amazonaws.com
```

Editing Lambda@Edge service-linked roles

Lambda@Edge does not allow you to edit the AWSServiceRoleForLambdaReplicator or AWSServiceRoleForCloudFrontLogger service-linked roles. After the service has created a service-

linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of a role by using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Supported Amazon Web Services Regions for CloudFront service-linked roles

CloudFront supports using service-linked roles for Lambda@Edge in the following Amazon Web Services Regions:

- US East (N. Virginia) – `us-east-1`
- US East (Ohio) – `us-east-2`
- US West (N. California) – `us-west-1`
- US West (Oregon) – `us-west-2`
- Asia Pacific (Mumbai) – `ap-south-1`
- Asia Pacific (Seoul) – `ap-northeast-2`
- Asia Pacific (Singapore) – `ap-southeast-1`
- Asia Pacific (Sydney) – `ap-southeast-2`
- Asia Pacific (Tokyo) – `ap-northeast-1`
- Europe (Frankfurt) – `eu-central-1`
- Europe (Ireland) – `eu-west-1`
- Europe (London) – `eu-west-2`
- South America (São Paulo) – `sa-east-1`

Writing and creating a Lambda@Edge function

To use Lambda@Edge, you *write* the code for your Lambda function, then set up Amazon Lambda to run the function based on specific CloudFront events (triggers). To set up Lambda to run your function, you use the *create* function option in Lambda.

You can use the Amazon console to work with Lambda functions and CloudFront triggers, or you can work with Lambda@Edge programmatically by using APIs.

- If you use the console, be aware that you can use only the Amazon Lambda console to create Lambda functions. You can't use the Amazon CloudFront console to create a function.

- If you want to work with Lambda@Edge programmatically, there are several resources to help you. For more information, see Creating Lambda@Edge functions and CloudFront triggers programmatically.

> **Note**
>
> You can use either the Amazon Lambda console or CloudFront console to add triggers for Lambda@Edge functions.

Topics

- Writing content of a Lambda@Edge function
- Creating a Lambda@Edge function in the Lambda console
- Creating Lambda@Edge functions and CloudFront triggers programmatically
- Editing a Lambda@Edge function

Writing content of a Lambda@Edge function

There are several resources to help you with writing Lambda@Edge functions:

- To learn about the event structure to use with Lambda@Edge functions, see Lambda@Edge event structure.
- To see examples of Lambda@Edge functions, such as functions for A/B testing and generating an HTTP redirect, see Lambda@Edge example functions.

The programming model for using Node.js or Python with Lambda@Edge is the same as using Lambda in an Amazon Region. For more information, see Building Lambda functions with Node.js or Building Lambda functions with Python.

In your Lambda@Edge code, include the `callback` parameter and return the applicable object for request or response events:

- **Request events** – Include the `cf.request` object in the response.

  If you're generating a response, include the `cf.response` object in the response. For more information, see Generating HTTP responses in request triggers.

- **Response events** – Include the `cf.response` object in the response.

Creating a Lambda@Edge function in the Lambda console

To set up Amazon Lambda to run Lambda functions that are based on CloudFront events, follow this procedure.

To create a Lambda@Edge function

1. Sign in to the Amazon Web Services Management Console and open the Amazon Lambda console at https://console.amazonaws.cn/lambda/.
2. If you already have one or more Lambda functions, choose **Create function**.

   If you've don't have any functions, choose **Get Started Now**.
3. In the Region list at the top of the page, choose **US East (N. Virginia)**.
4. Create a function using your own code or create a function starting with a CloudFront blueprint.

   - To create a function using your own code, choose **Author from scratch**.
   - To display a list of blueprints for CloudFront, type **cloudfront** in the filter field, and then press **Enter**.

     If you find a blueprint that you want to use, choose the name of the blueprint.
5. In the **Basic information** section, specify the following values:

   Name

   Type a name for your function.

   Role

   Choose **Create new role from template(s)**.

   > **Note**
   >
   > Choosing this value will get you started quickly. Or you can choose **Choose an existing role** or **Create a custom role**. If you choose one of these, follow the prompts to complete the information for this section.

Role name

Type a name for the role.

Policy templates

Choose **Basic Edge Lambda permissions**.

6. If you chose **Author from scratch** in step 4, skip to step 7.

   If you chose a blueprint in step 4, the **cloudfront** section lets you create one trigger, which associates this function with a cache in a CloudFront distribution and a CloudFront event. We recommend that you choose **Remove** at this point, so there isn't a trigger for the function when it's created. Then you can add triggers later.

   > **Important**
   >
   > Why add triggers later? Generally it's best to test and debug the function before you add triggers. If you choose instead to add a trigger now, the function will start to run as soon as you create the function and it finishes replicating to Amazon locations around the world, and the corresponding distribution is deployed.

7. Choose **Create function**.

   Lambda creates two versions of your function: $LATEST and Version 1. You can edit only the $LATEST version, but the console initially displays Version 1.

8. To edit the function, choose **Version 1** near the top of the page, under the ARN for the function. Then, on the **Versions** tab, choose **$LATEST**. (If you left the function and then returned to it, the button label is **Qualifiers**.)

9. On the **Configuration** tab, choose the applicable **Code entry type**. Then follow the prompts to edit or upload your code.

10. For **Runtime**, choose the value based on your function's code.

11. In the **Tags** section, add any applicable tags.

12. Choose **Actions**, and then choose **Publish new version**.

13. Type a description for the new version of the function.

14. Choose **Publish**.

15. Test and debug the function. For more information about testing in the Lambda console, see the *Invoke the Lambda Function and Verify Results, Logs, and Metrics* section in Create a Lambda Function with the Console in the *Amazon Lambda Developer Guide*.
16. When you're ready to have the function execute for CloudFront events, publish another version and edit the function to add triggers. For more information, see Adding triggers for a Lambda@Edge function.

Creating Lambda@Edge functions and CloudFront triggers programmatically

You can set up Lambda@Edge functions and CloudFront triggers programmatically by using API actions instead of by using the Amazon console. For more information, see the following:

- API Reference in the *Amazon Lambda Developer Guide*
- Amazon CloudFront API Reference
- **Amazon CLI**
  - Lambda create-function command
  - CloudFront create-distribution command
  - CloudFront create-distribution-with-tags command
  - CloudFront update-distribution command
- Amazon SDKs (See the **SDKs & Toolkits** section.)
- Amazon Tools for PowerShell Cmdlet Reference

Editing a Lambda@Edge function

When you want to edit a Lambda function, note the following:

- The original version is labeled $LATEST.
- You can edit only the $LATEST version.
- Each time you edit the $LATEST version, you must publish a new numbered version.
- You can't create triggers for $LATEST.
- When you publish a new version of a function, Lambda doesn't automatically copy triggers from the previous version to the new version. You must reproduce the triggers for the new version.

- When you add a trigger for a CloudFront event to a function, if there's already a trigger for the same distribution, cache behavior, and event for an earlier version of the same function, Lambda deletes the trigger from the earlier version.
- After you make updates to a CloudFront distribution, like adding triggers, you must wait for the changes to propagate to edge locations before the functions you've specified in the triggers will work.

To edit a Lambda function (Amazon Lambda console)

1. Sign in to the Amazon Web Services Management Console and open the Amazon Lambda console at https://console.amazonaws.cn/lambda/.
2. In the Region list at the top of the page, choose **US East (N. Virginia)**.
3. In the list of functions, choose the name of the function that you want to edit.

   By default, the console displays the $LATEST version. You can view earlier versions (choose **Qualifiers**), but you can only edit $LATEST.
4. On the **Code** tab, for **Code entry type**, choose to edit the code in the browser, upload a .zip file, or upload a file from Amazon S3.
5. Choose either **Save** or **Save and test**.
6. Choose **Actions**, and choose **Publish new version**.
7. In the **Publish new version from $LATEST** dialog box, enter a description of the new version. This description appears in the list of versions, along with an automatically generated version number.
8. Choose **Publish**.

   The new version automatically becomes the latest version. The version number appears on the **Version** button in the upper-left corner of the page.
9. Choose the **Triggers** tab.
10. Choose **Add trigger**.
11. In the **Add trigger** dialog box, choose the dotted box, and then choose **CloudFront**.

> **Note**
>
> If you've already created one or more triggers for a function, CloudFront is the default service.

12. Specify the following values to indicate when you want the Lambda function to execute.

    Distribution ID

    Choose the ID of the distribution that you want to add the trigger to.

    Cache behavior

    Choose the cache behavior that specifies the objects that you want to execute the function on.

    CloudFront event

    Choose the CloudFront event that causes the function to execute.

    Enable trigger and replicate

    Select this check box so Lambda replicates the function to Regions globally.

13. Choose **Submit**.
14. To add more triggers for this function, repeat steps 10 through 13.

Adding triggers for a Lambda@Edge function

A Lambda@Edge trigger is one combination of a CloudFront distribution, cache behavior, and event that causes a function to execute. You can specify one or more CloudFront triggers that cause the function to run. For example, you can create a trigger that causes the function to execute when CloudFront receives a request from a viewer for a specific cache behavior you set up for your distribution.

> **Tip**
>
> If you're not familiar with CloudFront cache behaviors, here's a brief overview. When you create a CloudFront distribution, you specify settings that tell CloudFront how to respond when it receives different requests. The default settings are called the default cache behavior for the distribution. You can set up additional cache behaviors that define how CloudFront responds under specific circumstances, for example, when it receives a request for a specific file type. For more information, see Cache Behavior Settings.

At the time that you create a Lambda function, you can specify only one trigger. But you can add more triggers to the same function later in one of two ways: by using the Lambda console or by editing the distribution in the CloudFront console.

- Using the Lambda console works well if you want to add more triggers to a function for the same CloudFront distribution.
- Using the CloudFront console can be better if you want to add triggers for multiple distributions because it's easier to find the distribution that you want to update. You can also update other CloudFront settings at the same time.

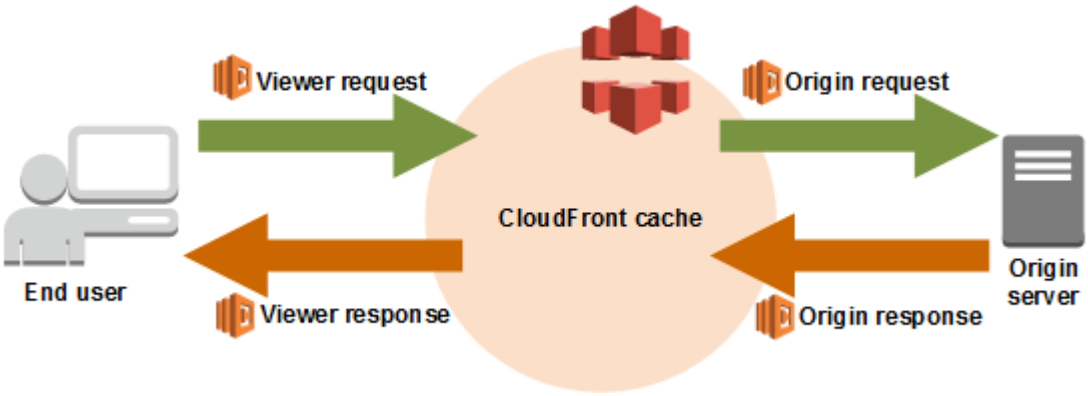
> **Note**
>
> If you want to work with Lambda@Edge programmatically, there are several resources to help you. For more information, see Creating Lambda@Edge functions and CloudFront triggers programmatically.

Topics

- CloudFront events that can trigger a Lambda@Edge function
- How to decide which CloudFront event to use to trigger a Lambda@Edge function
- Adding triggers by using the Lambda console
- Adding triggers by using the CloudFront console

CloudFront events that can trigger a Lambda@Edge function

For each cache behavior in a CloudFront distribution, you can add up to four triggers (associations) that cause a Lambda function to execute when specific CloudFront events occur. CloudFront triggers can be based on one of four CloudFront events, as shown in the following diagram.



The CloudFront events that can be used to trigger Lambda@Edge functions are the following:

Viewer request

The function executes when CloudFront receives a request from a viewer, before it checks to see whether the requested object is in the CloudFront cache.

Origin request

The function executes *only* when CloudFront forwards a request to your origin. When the requested object is in the CloudFront cache, the function doesn't execute.

Origin response

The function executes after CloudFront receives a response from the origin and before it caches the object in the response. Note that the function executes even if an error is returned from the origin.

The function doesn't execute in the following cases:

- When the requested file is in the CloudFront cache and is not expired.
- When the response is generated from a function that was triggered by an origin request event.

Viewer response

The function executes before returning the requested file to the viewer. Note that the function executes regardless of whether the file is already in the CloudFront cache.

The function doesn't execute in the following cases:

- When the origin returns an HTTP status code of 400 or higher.
- When a custom error page is returned.

- When the response is generated from a function that was triggered by a viewer request event.
- When CloudFront automatically redirects an HTTP request to HTTPS (when the value of Viewer protocol policy is **Redirect HTTP to HTTPS**).

When you add multiple triggers to the same cache behavior, you can use them to run the same function or run different functions for each trigger. You can also associate the same function with more than one distribution.

> **Note**
>
> When a CloudFront event triggers the execution of a Lambda function, the function must finish before CloudFront can continue. For example, if a Lambda function is triggered by a CloudFront viewer request event, CloudFront won't return a response to the viewer or forward the request to the origin until the Lambda function finishes running. This means that each request that triggers a Lambda function increases latency for the request, so you'll want the function to execute as fast as possible.

How to decide which CloudFront event to use to trigger a Lambda@Edge function

When you're deciding which CloudFront event you want to use to trigger a Lambda function, consider the following:

Do you want CloudFront to cache objects that are changed by a Lambda function?

If you want CloudFront to cache an object that was modified by a Lambda function so that CloudFront can serve the object from the edge location the next time it's requested, use the origin request or origin response event. This reduces the load on the origin, reduces latency for subsequent requests, and reduces the cost of invoking Lambda@Edge on subsequent requests.

For example, if you want to add, remove, or change headers for objects that are returned by the origin and you want CloudFront to cache the result, use the origin response event.

Do you want the function to execute for every request?

If you want the function to execute for every request that CloudFront receives for the distribution, use the viewer request or viewer response events. Origin request and origin

response events occur only when a requested object isn't cached in an edge location and CloudFront forwards a request to the origin.

Does the function change the cache key?

If you want the function to change a value that you're using as a basis for caching, use the viewer request event. For example, if a function changes the URL to include a language abbreviation in the path (for example, because the user chose their language from a dropdown list), use the viewer request event:

- **URL in the viewer request** – https://example.com/en/index.html
- **URL when the request comes from an IP address in Germany** – https://example.com/de/index.html

You also use the viewer request event if you're caching based on cookies or request headers.

> **Note**
>
> If the function changes cookies or headers, configure CloudFront to forward the applicable part of the request to the origin. For more information, see the following topics:
>
> - Caching content based on cookies
> - Caching content based on request headers

Does the function affect the response from the origin?

If you want the function to change the request in a way that affects the response from the origin, use the origin request event. Typically, most viewer request events aren't forwarded to the origin; CloudFront responds to a request with an object that's already in the edge cache. If the function changes the request based on an origin request event, CloudFront caches the response to the changed origin request.

Adding triggers by using the Lambda console

To add triggers to a Lambda@Edge function (Amazon Lambda console)

1. Sign in to the Amazon Web Services Management Console and open the Amazon Lambda console at https://console.amazonaws.cn/lambda/.

2. In the Region list at the top of the page, choose **US East (N. Virginia)**.
3. On the **Functions** page, choose the name of the function that you want to add triggers for.
4. On the **Function overview** page, choose the **Versions** tab.
5. Choose the version that you want to add triggers to.

   > **Important**
   >
   > You can't create triggers for the $LATEST version, you must create them for a numbered version.

   After you choose a version, the name of the button changes to **Version: $LATEST** or **Version:** *version number*.
6. Choose the **Triggers** tab.
7. Choose **Add trigger**.
8. For **Trigger configuration**, choose **Select a source** and type **`cloudfront`** in the search box. Select **CloudFront**.

   > **Note**
   >
   > If you've already created one or more triggers, CloudFront is the default service.
9. Specify the following values to indicate when you want the Lambda function to execute.

   Distribution

   Choose the distribution that you want to add the trigger to.

   Cache behavior

   Choose the cache behavior that specifies the objects that you want to execute the function on.

   > **Note**
   >
   > If you specify * for the cache behavior, the Lambda function deploys to the default cache behavior.

CloudFront event

Choose the CloudFront event that causes the function to execute.

Include body

Select this check box if you want to access the request body in your function.

Confirm deploy to Lambda@Edge

Select this check box so that Amazon Lambda replicates the function to Regions globally.

10. Choose **Add**.

The function starts to process requests for the specified CloudFront events when the updated CloudFront distribution is deployed. To determine whether a distribution is deployed, choose **Distributions** in the navigation pane. When a distribution is deployed, the value of the **Status** column for the distribution changes from **Deploying** to the date and time of deployment.

Adding triggers by using the CloudFront console

To add triggers for CloudFront events to a Lambda function (CloudFront console)

1. Get the ARN of the Lambda function that you want to add triggers for:

   a. Sign in to the Amazon Web Services Management Console and open the Amazon Lambda console at https://console.amazonaws.cn/lambda/.

   b. In the list of Regions at the top of the page, choose **US East (N. Virginia)**.

   c. In the list of functions, choose name of the function that you want to add triggers to.

   d. On the **Function overview** page, choose the **Versions** tab, and choose the numbered version that you want to add triggers to.

   > **Important**
   >
   > You can add triggers only to a numbered version, not to **$LATEST**.

   e. Choose the **Copy ARN** button to copy the ARN to your clipboard. The ARN for the Lambda function looks something like this:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:TestFunction:2
   ```

The number at the end (**2** in this example) is the version number of the function.

2. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
3. In the list of distributions, choose the ID of the distribution that you want to add triggers to.
4. Choose the **Behaviors** tab.
5. Select the cache behavior that you want to add triggers to, and then choose **Edit**.
6. For **Function associations**, in the **Function type** list, choose **Lambda@Edge** for when you want the function to execute: for viewer requests, viewer responses, origin requests, or origin responses.

   For more information, see How to decide which CloudFront event to use to trigger a Lambda@Edge function.
7. In the **Function ARN / Name** text box, paste the ARN of the Lambda function that you want to execute when the chosen event occurs. This is the value that you copied from the Lambda console.
8. Select **Include body** if you want to access the request body in your function.

   If you just want to replace the request body, you don't need to select this option.
9. To execute the same function for more event types, repeat steps 6 and 7.
10. Choose **Save changes**.
11. To add triggers to more cache behaviors for this distribution, repeat steps 5 through 10.

    The function starts to process requests for the specified CloudFront events when the updated CloudFront distribution is deployed. To determine whether a distribution is deployed, choose **Distributions** in the navigation pane. When a distribution is deployed, the value of the **Status** column for the distribution changes from **Deploying** to the time and date of deployment.

Testing and debugging Lambda@Edge functions

This topic includes sections that describe strategies for testing and debugging Lambda@Edge functions. It's important to test your Lambda@Edge function code standalone, to make sure that it completes the intended task, and to do integration testing, to make sure that the function works correctly with CloudFront.

During integration testing or after your function has been deployed, you might need to debug CloudFront errors, such as HTTP 5xx errors. Errors can be an invalid response returned from the

Lambda function, execution errors when the function is triggered, or errors due to execution throttling by the Lambda service. Sections in this topic share strategies for determining which type of failure is the issue, and then steps you can take to correct the problem.

> **Note**
>
> When you review CloudWatch log files or metrics when you're troubleshooting errors, be aware that they are displayed or stored in the Amazon Web Services Region closest to the location where the function executed. So, if you have a website or web application with users in the United Kingdom, and you have a Lambda function associated with your distribution, for example, you must change the Region to view the CloudWatch metrics or log files for the London Amazon Web Services Region. For more information, see the section called " Determining the Lambda@Edge Region".

Topics

- Testing your Lambda@Edge functions
- Identifying Lambda@Edge function errors in CloudFront
- Troubleshooting invalid Lambda@Edge function responses (validation errors)
- Troubleshooting Lambda@Edge function execution errors
- Determining the Lambda@Edge Region
- Determining if your account pushes logs to CloudWatch

Testing your Lambda@Edge functions

There are two steps to testing your Lambda function: standalone testing and integration testing.

Test standalone functionality

Before you add your Lambda function to CloudFront, make sure to test the functionality first by using the testing capabilities in the Lambda console or by using other methods. For more information about testing in the Lambda console, see the *Invoke the Lambda Function and Verify Results, Logs, and Metrics* section in Create a Lambda Function with the Console in the *Amazon Lambda Developer Guide*.

Test your function's operation in CloudFront

It's important to complete integration testing, where your function is associated with a distribution and runs based on a CloudFront event. Make sure that the function is triggered for the right event, and returns a response that is valid and correct for CloudFront. For example, make sure that the event structure is correct, that only valid headers are included, and so on.

As you iterate on integration testing with your function in the Lambda console, refer to the steps in the Lambda@Edge tutorial as you modify your code or change the CloudFront trigger that calls your function. For example, make sure that you're working in a numbered version of your function, as described in this step of the tutorial: Step 4: Add a CloudFront trigger to run the function.

As you make changes and deploy them, be aware that it will take several minutes for your updated function and CloudFront triggers to replicate across all Regions. This typically takes a few minutes but can take up to 15 minutes.

You can check to see if replication is finished by going to the CloudFront console and viewing your distribution.

To check if your replication has finished deploying

1. Open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose the distribution name.
3. Check for the distribution status to change from **In Progress** back to **Deployed**, which means that your function has been replicated. Then follow the steps in the next section to verify that the function works.

Be aware that testing in the console only validates your function's logic, and does not apply any service quotas (formerly known as limits) that are specific to Lambda@Edge.

Identifying Lambda@Edge function errors in CloudFront

After you've verified that your function logic works correctly, you might still see HTTP 5xx errors when your function runs in CloudFront. HTTP 5xx errors can be returned for a variety of reasons, which can include Lambda function errors or other issues in CloudFront.

- If you use Lambda@Edge functions, you can use graphs in the CloudFront console to help track down what's causing the error, and then work to fix it. For example, you can see if HTTP 5xx

errors are caused by CloudFront or by Lambda functions, and then, for specific functions, you can view related log files to investigate the issue.

- To troubleshoot HTTP errors in general in CloudFront, see the troubleshooting steps in the following topic: Troubleshooting error responses from your origin.

What causes Lambda@Edge function errors in CloudFront

There are several reasons why a Lambda function might cause an HTTP 5xx error, and the troubleshooting steps you should take depend on the type of error. Errors can be categorized as the following:

A Lambda function execution error

An execution error results when CloudFront doesn't get a response from Lambda because there are unhandled exceptions in the function or there's an error in the code. For example, if the code includes callback(Error). For more information, see Lambda Function Errors in the *Amazon Lambda Developer Guide*.

An invalid Lambda function response is returned to CloudFront

After the function runs, CloudFront receives a response from Lambda. An error is returned if the object structure of the response doesn't conform to the Lambda@Edge event structure, or the response contains invalid headers or other invalid fields.

The execution in CloudFront is throttled due to Lambda service quotas (formerly known as limits)

The Lambda service throttles executions in each Region, and returns an error if you exceed the quota.

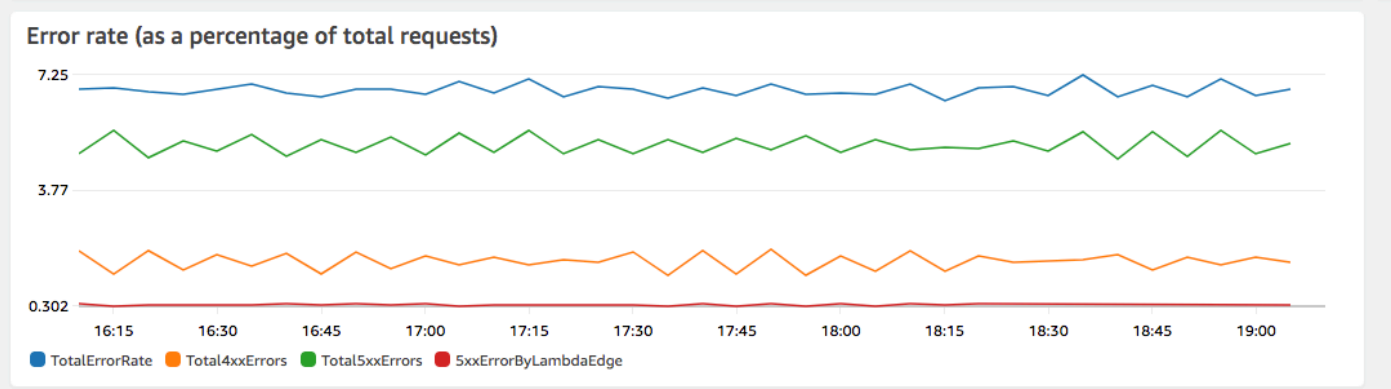
How to determine the type of failure

To help you decide where to focus as you debug and work to resolve errors returned by CloudFront, it's helpful to identify why CloudFront is returning an HTTP error. To get started, you can use the graphs provided in the **Monitoring** section of the CloudFront console on the Amazon Web Services Management Console. For more information about viewing graphs in the **Monitoring** section of the CloudFront console, see Monitoring CloudFront metrics with Amazon CloudWatch.

The following graphs can be especially helpful when you want to track down whether errors are being returned by origins or by a Lambda function, and to narrow down the type of issue when it's an error from a Lambda function.

Error rates graph

One of the graphs that you can view on the **Overview** tab for each of your distributions is an **Error rates** graph. This graph displays the rate of errors as a percentage of total requests coming to your distribution. The graph shows the total error rate, total 4xx errors, total 5xx errors, and total 5xx errors from Lambda functions. Based on the error type and volume, you can take steps to investigate and troubleshoot the cause.



- If you see Lambda errors, you can investigate further by looking at the specific types of errors that the function returns. The **Lambda@Edge errors** tab includes graphs that categorize function errors by type to help you pinpoint the issue for a specific function.
- If you see CloudFront errors, you can troubleshoot and work to fix origin errors or change your CloudFront configuration. For more information, see Troubleshooting error responses from your origin.

Execution errors and invalid function responses graphs

The **Lambda@Edge errors** tab includes graphs that categorize the Lambda@Edge errors for a specific distribution, by type. For example, one graph shows all execution errors by Amazon Web Services Region.

To make it easier to troubleshoot issues, you can look for specific problems by opening and examining the log files for specific functions by Region.

To view log files for a specific function by Region

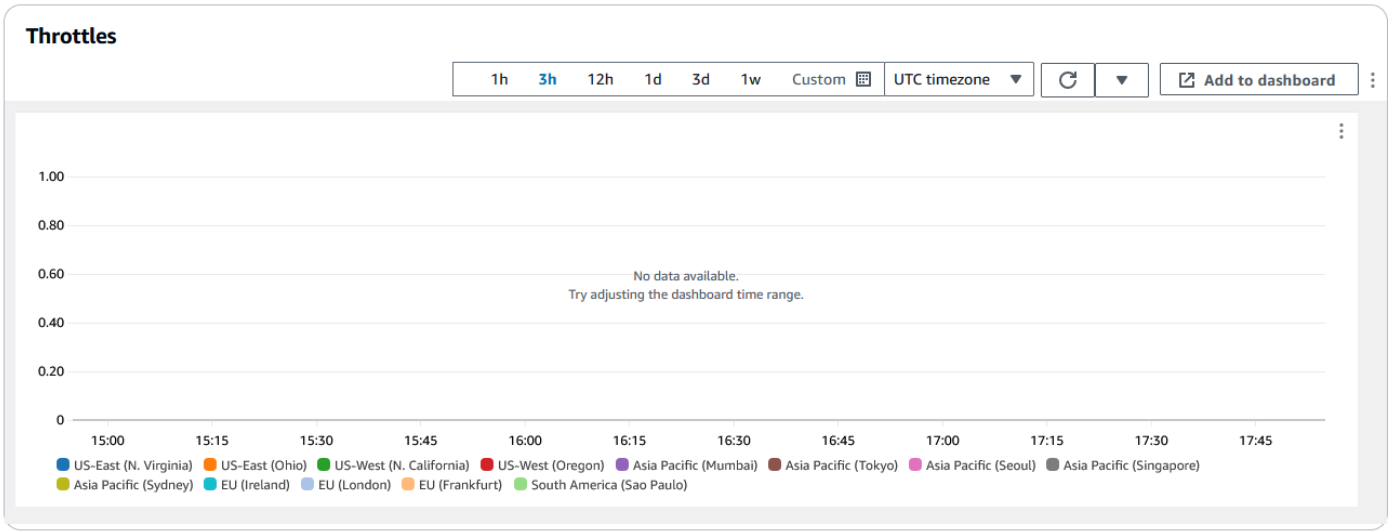
1. On the **Lambda@Edge errors** tab, under **Associated Lambda@Edge functions**, choose the function name, and then choose **View metrics**.
2. Next, on the page with your function name, in the upper-right corner, choose **View function logs**, and then choose a Region.

   For example, if you see issues in the **Errors** graph for the US West (Oregon) Region, choose that Region from the dropdown list. This opens the Amazon CloudWatch console.
3. In the CloudWatch console for that Region, under **Log streams**, choose a log stream to view the events for the function.

In addition, read the following sections in this chapter for more recommendations about troubleshooting and fixing errors.

Throttles graph

The **Lambda@Edge errors** tab also includes a **Throttles** graph. On occasion, the Lambda service throttles your function invocations on per Region basis, if you reach the regional concurrency quota (formerly known as limit). If you see a limit exceeded error, your function has reached a quota that the Lambda service imposes on executions in a Region. For more information, including how to request an increase in the quota, see Quotas on Lambda@Edge.



For an example about how to use this information in troubleshooting HTTP errors, see Four steps for debugging your content delivery on Amazon.

Troubleshooting invalid Lambda@Edge function responses (validation errors)

If you identify that your problem is a Lambda validation error, it means that your Lambda function is returning an invalid response to CloudFront. Follow the guidance in this section to take steps to review your function and make sure that your response conforms to CloudFront requirements.

CloudFront validates the response from a Lambda function in two ways:

- **The Lambda response must conform to the required object structure.** Examples of bad object structure include the following: unparsable JSON, missing required fields, and an invalid object in the response. For more information, see the Lambda@Edge event structure.
- **The response must include only valid object values.** An error will occur if the response includes a valid object but has values that are not supported. Examples include the following: adding or updating disallowed or read-only headers (see Restrictions on edge functions), exceeding the maximum body size (see *Restrictions on the Size of the Generated Response* in the Lambda@Edge Errors topic), and invalid characters or values (see the Lambda@Edge event structure).

When Lambda returns an invalid response to CloudFront, error messages are written to log files which CloudFront pushes to CloudWatch in the Region of where the Lambda function executed. It's the default behavior to send the log files to CloudWatch when there's an invalid response. However, if you associated a Lambda function with CloudFront before the functionality was released, it might not be enabled for your function. For more information, see *Determine if Your Account Pushes Logs to CloudWatch* later in the topic.

CloudFront pushes log files to the Region corresponding to where your function executed, in the log group that's associated with your distribution. Log groups have the following format: `/aws/cloudfront/LambdaEdge/DistributionId`, where *`DistributionId`* is your distribution's ID. To determine the Region where you can find the CloudWatch log files, see *Determining the Lambda@Edge Region* later in this topic.

If the error is reproducible, you can create a new request that results in the error and then find the request id in a failed CloudFront response (`X-Amz-Cf-Id` header) to locate a single failure in log files. The log file entry includes information that can help you identify why the error is being returned, and also lists the corresponding Lambda request id so you can analyze the root cause in the context of a single request.

If an error is intermittent, you can use CloudFront access logs to find the request id for a request that has failed, and then search CloudWatch logs for the corresponding error messages. For more information, see the previous section, *Determining the Type of Failure*.

Troubleshooting Lambda@Edge function execution errors

If the problem is a Lambda execution error, it can be helpful to create logging statements for Lambda functions, to write messages to CloudWatch log files that monitor the execution of your function in CloudFront and determine if it's working as expected. Then you can search for those statements in the CloudWatch log files to verify that your function is working.

> **Note**
>
> Even if you haven't changed your Lambda@Edge function, updates to the Lambda function execution environment might affect it and could return an execution error. For information about testing and migrating to a later version, see Upcoming updates to the Amazon Lambda and Amazon Lambda@Edge execution environment.

Determining the Lambda@Edge Region

To see the Regions where your Lambda@Edge function is receiving traffic, view metrics for the function on the CloudFront console on the Amazon Web Services Management Console. Metrics are displayed for each Amazon Region. On the same page, you can choose a Region and view log files for that Region so you can investigate issues. You must review CloudWatch log files in the correct Amazon Region to see the log files created when CloudFront executed your Lambda function.

For more information about viewing graphs in the **Monitoring** section of the CloudFront console, see Monitoring CloudFront metrics with Amazon CloudWatch.

Determining if your account pushes logs to CloudWatch

By default, CloudFront enables logging invalid Lambda function responses, and pushes the log files to CloudWatch by using one of the Service-linked roles for Lambda@Edge. If you have Lambda@Edge functions that you added to CloudFront before the invalid Lambda function response log feature was released, logging is enabled when you next update your Lambda@Edge configuration, for example, by adding a CloudFront trigger.

You can verify that pushing the log files to CloudWatch is enabled for your account by doing the following:

- **Check to see if the logs appear in CloudWatch.** Make sure that you look in the Region where the Lambda@Edge function executed. For more information, see Determining the Lambda@Edge Region.
- **Determine if the related service-linked role exists in your account in IAM.** To do this, open the IAM console at https://console.amazonaws.cn/iam/, and then choose **Roles** to view the list of service-linked roles for your account. Look for the following role: `AWSServiceRoleForCloudFrontLogger`.

Deleting Lambda@Edge functions and replicas

You can delete a Lambda@Edge function only when the replicas of the function have been deleted by CloudFront. Replicas of a Lambda function are automatically deleted in the following situations:

- After you remove the last association for the function from all of your CloudFront distributions. If more than one distribution uses a function, the replicas are deleted only after you remove the function association from the last distribution.
- After you delete the last distribution that a function was associated with.

Replicas are typically deleted within a few hours. You cannot manually delete Lambda@Edge function replicas. This helps prevent a situation where a replica is deleted that is still in use, which would result in an error.

Don't build applications that use Lambda@Edge function replicas outside of CloudFront. These replicas are deleted when their associations with distributions are removed, or when distributions themselves are deleted. The replica that an outside application depends on might be removed without warning, causing it to fail.

To delete a Lambda@Edge function association from a CloudFront distribution (console)

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. Choose the ID of the distribution that has the Lambda@Edge function association that you want to delete.
3. Choose the **Behaviors** tab.
4. Select the cache behavior that has the Lambda@Edge function association that you want to delete, and then choose **Edit**.

5. Under **Function associations**, **Function type**, choose **No association** to delete the Lambda@Edge function association.
6. Choose **Save changes**.

After you delete a Lambda@Edge function association from a CloudFront distribution, you can optionally delete the Lambda function or function version from Amazon Lambda. Wait a few hours after deleting the function association so that the Lambda@Edge function replicas can be cleaned up. After that, you will be able to delete the function by using the Lambda console, Amazon CLI, Lambda API, or an Amazon SDK.

You can also delete a specific *version* of a Lambda function if the version doesn't have any CloudFront distributions associated with it. After removing all the associations for a Lambda function version, wait a few hours. Then you will be able to delete the function version.

Lambda@Edge event structure

The following topics describe the request and response event objects that CloudFront passes to a Lambda@Edge function when it's triggered.

Topics

- Dynamic origin selection
- Request events
- Response events

Dynamic origin selection

You can use the path pattern in a cache behavior to route requests to an origin based on the path and name of the requested object, such as `images/*.jpg`. Using Lambda@Edge, you can also route requests to an origin based on other characteristics, such as the values in request headers.

There are a number of ways that this dynamic origin selection can be useful. For example, you can distribute requests across origins in different geographic areas to help with global load balancing. Or you can selectively route requests to different origins that each serve a particular function: bot handling, SEO optimization, authentication, and so on. For code examples that demonstrate how to use this feature, see Content-based dynamic origin selection - examples.

In the CloudFront origin request event, the `origin` object in the event structure contains information about the origin that the request would be routed to, based on the path pattern.

You can update the values in the `origin` object to route a request to a different origin. When you update the `origin` object, you don't need to define the origin in the distribution. You can also replace an Amazon S3 origin object with a custom origin object, and vice versa. You can only specify a single origin per request, though; either a custom origin or an Amazon S3 origin, but not both.

Request events

The following topics show the structure of the object that CloudFront passes to a Lambda function for viewer and origin request events. These examples show a `GET` request with no body. Following the examples is a list of all the possible fields in viewer and origin request events.

Topics

- Example viewer request
- Example origin request
- Request event fields

Example viewer request

The following example shows a viewer request event object.

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "viewer-request",
          "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "host": [
              {
                "key": "Host",
                "value": "d111111abcdef8.cloudfront.net"
              }
            ],
```

```
            "user-agent": [
              {
                "key": "User-Agent",
                "value": "curl/7.66.0"
              }
            ],
            "accept": [
              {
                "key": "accept",
                "value": "*/*"
              }
            ]
          },
          "method": "GET",
          "querystring": "",
          "uri": "/"
        }
      }
    }
  ]
}
```

Example origin request

The following example shows an origin request event object.

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "origin-request",
          "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "x-forwarded-for": [
              {
                "key": "X-Forwarded-For",
                "value": "203.0.113.178"
              }
```

```
            ],
            "user-agent": [
              {
                "key": "User-Agent",
                "value": "Amazon CloudFront"
              }
            ],
            "via": [
              {
                "key": "Via",
                "value": "2.0 2afae0d44e2540f472c0635ab62c232b.cloudfront.net
(CloudFront)"
              }
            ],
            "host": [
              {
                "key": "Host",
                "value": "example.org"
              }
            ],
            "cache-control": [
              {
                "key": "Cache-Control",
                "value": "no-cache"
              }
            ]
          },
          "method": "GET",
          "origin": {
            "custom": {
              "customHeaders": {},
              "domainName": "example.org",
              "keepaliveTimeout": 5,
              "path": "",
              "port": 443,
              "protocol": "https",
              "readTimeout": 30,
              "sslProtocols": [
                "TLSv1",
                "TLSv1.1",
                "TLSv1.2"
              ]
            }
          },
```

```
          "querystring": "",
          "uri": "/"
        }
      }
    }
  ]
}
```

Request event fields

Request event object data is contained in two subobjects: `config` (`Records.cf.config`) and `request` (`Records.cf.request`). The following lists describe each subobject's fields.

Fields in the config object

The following list describes the fields in the `config` object (`Records.cf.config`).

`distributionDomainName` (read-only)

The domain name of the distribution that's associated with the request.

`distributionID` (read-only)

The ID of the distribution that's associated with the request.

`eventType` (read-only)

The type of trigger that's associated with the request: `viewer-request` or `origin-request`.

`requestId` (read-only)

An encrypted string that uniquely identifies a viewer-to-CloudFront request. The `requestId` value also appears in CloudFront access logs as `x-edge-request-id`. For more information, see Configuring and using standard logs (access logs) and Standard log file fields.

Fields in the request object

The following list describes the fields in the `request` object (`Records.cf.request`).

`clientIp` (read-only)

The IP address of the viewer that made the request. If the viewer used an HTTP proxy or a load balancer to send the request, the value is the IP address of the proxy or load balancer.

headers (read/write)

The headers in the request. Note the following:

- The keys in the `headers` object are lowercase versions of standard HTTP header names. Using lowercase keys gives you case-insensitive access to the header values.
- Each header object (for example, `headers["accept"]` or `headers["host"]`) is an array of key–value pairs. For a given header, the array contains one key–value pair for each value in the request.
- `key` contains the case-sensitive name of the header as it appeared in the HTTP request; for example, `Host`, `User-Agent`, `X-Forwarded-For`, and so on.
- `value` contains the header value as it appeared in the HTTP request.
- When your Lambda function adds or modifies request headers and you don't include the header `key` field, Lambda@Edge automatically inserts a header `key` using the header name that you provide. Regardless of how you've formatted the header name, the header key that's inserted automatically is formatted with initial capitalization for each part, separated by hyphens (-).

  For example, you can add a header like the following, without a header `key`:

  ```
  "user-agent": [
    {
      "value": "ExampleCustomUserAgent/1.X.0"
    }
  ]
  ```

  In this example, Lambda@Edge automatically inserts `"key": "User-Agent"`.

For information about restrictions on header usage, see Restrictions on edge functions.

`method` (read-only)

The HTTP method of the request.

`querystring` (read/write)

The query string, if any, in the request. If the request doesn't include a query string, the event object still includes `querystring` with an empty value. For more information about query strings, see Caching content based on query string parameters.

`uri` (read/write)

The relative path of the requested object. If your Lambda function modifies the `uri` value, note the following:

- The new `uri` value must begin with a forward slash (/).
- When a function changes the `uri` value, that changes the object that the viewer is requesting.
- When a function changes the `uri` value, that *doesn't* change the cache behavior for the request or the origin that the request is sent to.

body (read/write)

The body of the HTTP request. The body structure can contain the following fields:

`inputTruncated` (read-only)

A Boolean flag that indicates whether the body was truncated by Lambda@Edge. For more information, see Restrictions on the request body with the include body option.

`action` (read/write)

The action that you intend to take with the body. The options for `action` are the following:

- `read-only:` This is the default. When returning the response from the Lambda function, if `action` is read-only, Lambda@Edge ignores any changes to `encoding` or `data`.
- `replace:` Specify this when you want to replace the body sent to the origin.

`encoding` (read/write)

The encoding for the body. When Lambda@Edge exposes the body to the Lambda function, it first converts the body to base64-encoding. If you choose `replace` for the `action` to replace the body, you can opt to use `base64` (the default) or `text` encoding. If you specify `encoding` as `base64` but the body is not valid base64, CloudFront returns an error.

`data` (read/write)

The request body content.

`origin` (read/write) (origin events only)

The origin to send the request to. The `origin` structure must contain exactly one origin, which can be a custom origin or an Amazon S3 origin. The origin structure can contain the following fields:

`customHeaders` (read/write) (custom and Amazon S3 origins)

You can include custom headers with the request by specifying a header name and value pair for each custom header. You can't add headers that are disallowed, and a header with the same name can't be present in `Records.cf.request.headers`. The notes about request headers also apply to custom headers. For more information, see Custom headers that CloudFront can't add to origin requests and Restrictions on edge functions.

`domainName` (read/write) (custom and Amazon S3 origins)

The domain name of the origin. The domain name can't be empty.

- **For custom origins** – Specify a DNS domain name, such as `www.example.com`. The domain name can't include a colon (:), and can't be an IP address. The domain name can be up to 253 characters.
- **For Amazon S3 origins** – Specify the DNS domain name of the Amazon S3 bucket, such as `awsexamplebucket.s3.eu-west-1.amazonaws.com`. The name can be up to 128 characters, and must be all lowercase.

`path` (read/write) (custom and Amazon S3 origins)

The directory path at the origin where the request should locate content. The path should start with a forward slash (/) but shouldn't end with one (for example, it shouldn't end with `example-path/`). For custom origins only, the path should be URL encoded and have a maximum length of 255 characters.

`keepaliveTimeout` (read/write) (custom origins only)

How long, in seconds, that CloudFront should try to maintain the connection to the origin after receiving the last packet of the response. The value must be a number from 1–60, inclusive.

`port` (read/write) (custom origins only)

The port that CloudFront should connect to at your custom origin. The port must be 80, 443, or a number in the range of 1024–65535, inclusive.

`protocol` (read/write) (custom origins only)

The connection protocol that CloudFront should use when connecting to your origin. The value can be `http` or `https`.

`readTimeout` (read/write) (custom origins only)

How long, in seconds, CloudFront should wait for a response after sending a request to your origin. This also specifies how long CloudFront should wait after receiving a packet of a response before receiving the next packet. The value must be a number from 4–60, inclusive.

If your use case requires more than 60 seconds, you can request a higher quota for `Response timeout per origin`. For more information, see General quotas on distributions.

`sslProtocols` (read/write) (custom origins only)

The minimum SSL/TLS protocol that CloudFront can use when establishing an HTTPS connection with your origin. Values can be any of the following: `TLSv1.2`, `TLSv1.1`, `TLSv1`, or `SSLv3`.

`authMethod` (read/write) (Amazon S3 origins only)

If you're using an origin access identity (OAI), set this field to `origin-access-identity`. If you aren't using an OAI, set it to none. If you set `authMethod` to `origin-access-identity`, there are several requirements:

- You must specify the `region` (see the following field).
- You must use the same OAI when you change the request from one Amazon S3 origin to another.
- You can't use an OAI when you change the request from a custom origin to an Amazon S3 origin.

> **Note**
>
> This field does not support origin access control (OAC).

`region` (read/write) (Amazon S3 origins only)

The Amazon Region of your Amazon S3 bucket. This is required only when you set `authMethod` to `origin-access-identity`.

Response events

The following topics show the structure of the object that CloudFront passes to a Lambda function for viewer and origin response events. Following the examples is a list of all the possible fields in viewer and origin response events.

Topics

- Example origin response
- Example viewer response
- Response event fields

Example origin response

The following example shows an origin response event object.

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "origin-response",
          "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "x-forwarded-for": [
              {
                "key": "X-Forwarded-For",
                "value": "203.0.113.178"
              }
            ],
            "user-agent": [
              {
                "key": "User-Agent",
                "value": "Amazon CloudFront"
              }
            ],
            "via": [
```

```
            {
              "key": "Via",
              "value": "2.0 8f22423015641505b8c857a37450d6c0.cloudfront.net
(CloudFront)"
            }
          ],
          "host": [
            {
              "key": "Host",
              "value": "example.org"
            }
          ],
          "cache-control": [
            {
              "key": "Cache-Control",
              "value": "no-cache"
            }
          ]
        },
        "method": "GET",
        "origin": {
          "custom": {
            "customHeaders": {},
            "domainName": "example.org",
            "keepaliveTimeout": 5,
            "path": "",
            "port": 443,
            "protocol": "https",
            "readTimeout": 30,
            "sslProtocols": [
              "TLSv1",
              "TLSv1.1",
              "TLSv1.2"
            ]
          }
        },
        "querystring": "",
        "uri": "/"
      },
      "response": {
        "headers": {
          "access-control-allow-credentials": [
            {
              "key": "Access-Control-Allow-Credentials",
```

```
            "value": "true"
          }
        ],
        "access-control-allow-origin": [
          {
            "key": "Access-Control-Allow-Origin",
            "value": "*"
          }
        ],
        "date": [
          {
            "key": "Date",
            "value": "Mon, 13 Jan 2020 20:12:38 GMT"
          }
        ],
        "referrer-policy": [
          {
            "key": "Referrer-Policy",
            "value": "no-referrer-when-downgrade"
          }
        ],
        "server": [
          {
            "key": "Server",
            "value": "ExampleCustomOriginServer"
          }
        ],
        "x-content-type-options": [
          {
            "key": "X-Content-Type-Options",
            "value": "nosniff"
          }
        ],
        "x-frame-options": [
          {
            "key": "X-Frame-Options",
            "value": "DENY"
          }
        ],
        "x-xss-protection": [
          {
            "key": "X-XSS-Protection",
            "value": "1; mode=block"
          }
```

```
            ],
            "content-type": [
              {
                "key": "Content-Type",
                "value": "text/html; charset=utf-8"
              }
            ],
            "content-length": [
              {
                "key": "Content-Length",
                "value": "9593"
              }
            ]
          },
          "status": "200",
          "statusDescription": "OK"
        }
      }
    }
  ]
}
```

Example viewer response

The following example shows a viewer response event object.

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "viewer-response",
          "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "host": [
              {
                "key": "Host",
                "value": "d111111abcdef8.cloudfront.net"
              }
```

```
            ],
            "user-agent": [
              {
                "key": "User-Agent",
                "value": "curl/7.66.0"
              }
            ],
            "accept": [
              {
                "key": "accept",
                "value": "*/*"
              }
            ]
          },
          "method": "GET",
          "querystring": "",
          "uri": "/"
        },
        "response": {
          "headers": {
            "access-control-allow-credentials": [
              {
                "key": "Access-Control-Allow-Credentials",
                "value": "true"
              }
            ],
            "access-control-allow-origin": [
              {
                "key": "Access-Control-Allow-Origin",
                "value": "*"
              }
            ],
            "date": [
              {
                "key": "Date",
                "value": "Mon, 13 Jan 2020 20:14:56 GMT"
              }
            ],
            "referrer-policy": [
              {
                "key": "Referrer-Policy",
                "value": "no-referrer-when-downgrade"
              }
            ],
```

```
            "server": [
              {
                "key": "Server",
                "value": "ExampleCustomOriginServer"
              }
            ],
            "x-content-type-options": [
              {
                "key": "X-Content-Type-Options",
                "value": "nosniff"
              }
            ],
            "x-frame-options": [
              {
                "key": "X-Frame-Options",
                "value": "DENY"
              }
            ],
            "x-xss-protection": [
              {
                "key": "X-XSS-Protection",
                "value": "1; mode=block"
              }
            ],
            "age": [
              {
                "key": "Age",
                "value": "2402"
              }
            ],
            "content-type": [
              {
                "key": "Content-Type",
                "value": "text/html; charset=utf-8"
              }
            ],
            "content-length": [
              {
                "key": "Content-Length",
                "value": "9593"
              }
            ]
          },
          "status": "200",
```

```
          "statusDescription": "OK"
        }
      }
    }
  ]
}
```

Response event fields

Response event object data is contained in three subobjects: `config` (`Records.cf.config`), `request` (`Records.cf.request`), and `response` (`Records.cf.response`). For more information about the fields in the request object, see Fields in the request object. The following lists describe the fields in the `config` and `response` subobjects.

Fields in the config object

The following list describes the fields in the `config` object (`Records.cf.config`).

`distributionDomainName` (read-only)

The domain name of the distribution that's associated with the response.

`distributionID` (read-only)

The ID of the distribution that's associated with the response.

`eventType` (read-only)

The type of trigger that's associated with the response: `origin-response` or `viewer-response`.

`requestId` (read-only)

An encrypted string that uniquely identifies the viewer-to-CloudFront request that this response is associated with. The `requestId` value also appears in CloudFront access logs as `x-edge-request-id`. For more information, see Configuring and using standard logs (access logs) and Standard log file fields.

Fields in the response object

The following list describes the fields in the `response` object (`Records.cf.response`). For information about using a Lambda@Edge function to generate an HTTP response, see Generating HTTP responses in request triggers.

`headers` (read/write)

The headers in the response. Note the following:

- The keys in the `headers` object are lowercase versions of standard HTTP header names. Using lowercase keys gives you case-insensitive access to the header values.
- Each header object (for example, `headers["content-type"]` or `headers["content-length"]`) is an array of key–value pairs. For a given header, the array contains one key–value pair for each value in the response.
- `key` contains the case-sensitive name of the header as it appears in the HTTP response; for example, `Content-Type`, `Content-Length`, `Cookie`, and so on.
- `value` contains the header value as it appears in the HTTP response.
- When your Lambda function adds or modifies response headers and you don't include the header `key` field, Lambda@Edge automatically inserts a header `key` using the header name that you provide. Regardless of how you've formatted the header name, the header key that's inserted automatically is formatted with initial capitalization for each part, separated by hyphens (-).

  For example, you can add a header like the following, without a header `key`:

  ```
  "content-type": [
    {
      "value": "text/html;charset=UTF-8"
    }
  ]
  ```

  In this example, Lambda@Edge automatically inserts `"key": "Content-Type"`.

For information about restrictions on header usage, see Restrictions on edge functions.

`status`

The HTTP status code of the response.

`statusDescription`

The HTTP status description of the response.

Working with requests and responses

The topics in this section explain several ways to use Lambda@Edge requests and responses.

Topics

- Using Lambda@Edge functions with origin failover
- Generating HTTP responses in request triggers
- Updating HTTP responses in origin response triggers
- Accessing the request body by choosing the include body option

Using Lambda@Edge functions with origin failover

You can use Lambda@Edge functions with CloudFront distributions that you've set up with origin groups, for example, for origin failover that you configure to help ensure high availability. To use a Lambda function with an origin group, specify the function in an origin request or origin response trigger for an origin group when you create the cache behavior.

For more information, see the following:

- **Creating origin groups:** Creating an origin group
- **How origin failover works with Lambda@Edge:** Use origin failover with Lambda@Edge functions

Generating HTTP responses in request triggers

When CloudFront receives a request, you can use a Lambda function to generate an HTTP response that CloudFront returns directly to the viewer without forwarding the response to the origin. Generating HTTP responses reduces the load on the origin, and typically also reduces latency for the viewer.

Some common scenarios for generating HTTP responses include the following:

- Returning a small webpage to the viewer
- Returning an HTTP 301 or 302 status code to redirect the user to another webpage
- Returning an HTTP 401 status code to the viewer when the user hasn't authenticated

A Lambda@Edge function can generate an HTTP response when the following CloudFront events occur:

Viewer request events

When a function is triggered by a viewer request event, CloudFront returns the response to the viewer and doesn't cache it.

Origin request events

When a function is triggered by an origin request event, CloudFront checks the edge cache for a response that was previously generated by the function.

- If the response is in the cache, the function isn't executed and CloudFront returns the cached response to the viewer.
- If the response isn't in the cache, the function is executed, CloudFront returns the response to the viewer, and also caches it.

To see some sample code for generating HTTP responses, see Lambda@Edge example functions. You can also replace the HTTP responses in response triggers. For more information, see Updating HTTP responses in origin response triggers.

Programming model

This section describes the programming model for using Lambda@Edge to generate HTTP responses.

Topics

- Response object
- Errors
- Required fields

Response object

The response you return as the `result` parameter of the `callback` method should have the following structure (note that only the `status` field is required).

```
const response = {
    body: 'content',
    bodyEncoding: 'text' | 'base64',
    headers: {
        'header name in lowercase': [{
            key: 'header name in standard case',
```

```
            value: 'header value'
         }],
         ...
    },
    status: 'HTTP status code (string)',
    statusDescription: 'status description'
};
```

The response object can include the following values:

body

The body, if any, that you want CloudFront to return in the generated response.

bodyEncoding

The encoding for the value that you specified in the `body`. The only valid encodings are `text` and `base64`. If you include `body` in the `response` object but omit `bodyEncoding`, CloudFront treats the body as text.

If you specify `bodyEncoding` as `base64` but the body is not valid base64, CloudFront returns an error.

headers

Headers that you want CloudFront to return in the generated response. Note the following:

- The keys in the `headers` object are lowercase versions of standard HTTP header names. Using lowercase keys gives you case-insensitive access to the header values.
- Each header (for example, `headers["accept"]` or `headers["host"]`) is an array of key-value pairs. For a given header, the array contains one key-value pair for each value in the generated response.
- `key` (optional) is the case-sensitive name of the header as it appears in an HTTP request; for example, `accept` or `host`.
- Specify `value` as a header value.
- If you do not include the header key portion of the key-value pair, Lambda@Edge automatically inserts a header key using the header name that you provide. Regardless of how you've formatted the header name, the header key that is inserted is automatically formatted with initial capitalization for each part, separated by hyphens (-).

  For example, you can add a header like the following, without a header key: `'content-type': [{ value: 'text/html;charset=UTF-8' }]`

In this example, Lambda@Edge creates the following header key: `Content-Type`.

For information about restrictions on header usage, see Restrictions on edge functions.

`status`

The HTTP status code. Provide the status code as a string. CloudFront uses the provided status code for the following:

- Return in the response
- Cache in the CloudFront edge cache, when the response was generated by a function that was triggered by an origin request event
- Log in CloudFront Configuring and using standard logs (access logs)

If the `status` value isn't between 200 and 599, CloudFront returns an error to the viewer.

`statusDescription`

The description that you want CloudFront to return in the response, to accompany the HTTP status code. You don't need to use standard descriptions, such as OK for an HTTP status code of 200.

Errors

The following are possible errors for generated HTTP responses.

Response Contains a Body and Specifies 204 (No Content) for Status

When a function is triggered by a viewer request, CloudFront returns an HTTP 502 status code (Bad Gateway) to the viewer when both of the following are true:

- The value of `status` is 204 (No Content)
- The response includes a value for body

This is because Lambda@Edge imposes the optional restriction found in RFC 2616, which states that an `HTTP 204` response does not need to contain a message body.

Restrictions on the Size of the Generated Response

The maximum size of a response that is generated by a Lambda function depends on the event that triggered the function:

- **Viewer request events** – 40 KB

- **Origin request events** – 1 MB

If the response is larger than the allowed size, CloudFront returns an HTTP 502 status code (Bad Gateway) to the viewer.

Required fields

The `status` field is required.

All other fields are optional.

Updating HTTP responses in origin response triggers

When CloudFront receives an HTTP response from the origin server, if there is an origin-response trigger associated with the cache behavior, you can modify the HTTP response to override what was returned from the origin.

Some common scenarios for updating HTTP responses include the following:

- Changing the status to set an HTTP 200 status code and creating static body content to return to the viewer when an origin returns an error status code (4xx or 5xx). For sample code, see Example: Using an origin response trigger to update the error status code to 200.
- Changing the status to set an HTTP 301 or HTTP 302 status code, to redirect the user to another website when an origin returns an error status code (4xx or 5xx). For sample code, see Example: Using an origin response trigger to update the error status code to 302.

> **Note**
>
> The function must return a status value between `200` and 599 (inclusive), otherwise CloudFront returns an error to the viewer.

You can also replace the HTTP responses in viewer and origin request events. For more information, see Generating HTTP responses in request triggers.

When you're working with the HTTP response, Lambda@Edge does not expose the body that is returned by the origin server to the origin-response trigger. You can generate a static content body by setting it to the desired value, or remove the body inside the function by setting the value to be

empty. If you don't update the body field in your function, the original body returned by the origin server is returned back to viewer.

Accessing the request body by choosing the include body option

You can opt to have Lambda@Edge expose the body in a request for writable HTTP methods (POST, PUT, DELETE, and so on), so that you can access it in your Lambda function. You can choose read-only access, or you can specify that you'll replace the body.

To enable this option, choose **Include Body** when you create a CloudFront trigger for your function that's for a viewer request or origin request event. For more information, see Adding triggers for a Lambda@Edge function, or to learn about using **Include Body** with your function, see Lambda@Edge event structure.

Scenarios when you might want to use this feature include the following:

- Processing web forms, like "contact us" forms, without sending customer input data back to origin servers.
- Gathering web beacon data that's sent by viewer browsers and processing it at the edge.

For sample code, see Lambda@Edge example functions.

> **Note**
>
> If the request body is large, Lambda@Edge truncates it. For detailed information about the maximum size and truncation, see Restrictions on the request body with the include body option.

Lambda@Edge example functions

See the following sections for examples of using Lambda functions with CloudFront.

Topics

- General examples
- Generating responses - examples
- Working with query strings - examples
- Personalize content by country or device type headers - examples

- Content-based dynamic origin selection - examples
- Updating error statuses - examples
- Accessing the request body - examples

General examples

The examples in this section illustrate some common ways to use Lambda@Edge in CloudFront.

Topics

- Example: A/B testing
- Example: Overriding a response header

Example: A/B testing

You can use the following example to test two different versions of an image without creating redirects or changing the URL. This example reads the cookies in the viewer request and modifies the request URL accordingly. If the viewer doesn't send a cookie with one of the expected values, the example randomly assigns the viewer to one of the URLs.

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const headers = request.headers;

    if (request.uri !== '/experiment-pixel.jpg') {
        // do not process if this is not an A-B test request
        callback(null, request);
        return;
    }

    const cookieExperimentA = 'X-Experiment-Name=A';
    const cookieExperimentB = 'X-Experiment-Name=B';
    const pathExperimentA = '/experiment-group/control-pixel.jpg';
    const pathExperimentB = '/experiment-group/treatment-pixel.jpg';

    /*
     * Lambda at the Edge headers are array objects.
```

```
    *
    * Client may send multiple Cookie headers, i.e.:
    * > GET /viewerRes/test HTTP/1.1
    * > User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1
 OpenSSL/1.0.1u zlib/1.2.3
    * > Cookie: First=1; Second=2
    * > Cookie: ClientCode=abc
    * > Host: example.com
    *
    * You can access the first Cookie header at headers["cookie"][0].value
    * and the second at headers["cookie"][1].value.
    *
    * Header values are not parsed. In the example above,
    * headers["cookie"][0].value is equal to "First=1; Second=2"
    */
   let experimentUri;
   if (headers.cookie) {
       for (let i = 0; i < headers.cookie.length; i++) {
           if (headers.cookie[i].value.indexOf(cookieExperimentA) >= 0) {
               console.log('Experiment A cookie found');
               experimentUri = pathExperimentA;
               break;
           } else if (headers.cookie[i].value.indexOf(cookieExperimentB) >= 0) {
               console.log('Experiment B cookie found');
               experimentUri = pathExperimentB;
               break;
           }
       }
   }

   if (!experimentUri) {
       console.log('Experiment cookie has not been found. Throwing dice...');
       if (Math.random() < 0.75) {
           experimentUri = pathExperimentA;
       } else {
           experimentUri = pathExperimentB;
       }
   }

   request.uri = experimentUri;
   console.log(`Request uri set to "${request.uri}"`);
   callback(null, request);
};
```

Python

```
import json
import random

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    if request['uri'] != '/experiment-pixel.jpg':
        # Not an A/B Test
        return request

    cookieExperimentA, cookieExperimentB = 'X-Experiment-Name=A', 'X-Experiment-
Name=B'
    pathExperimentA, pathExperimentB = '/experiment-group/control-pixel.jpg', '/
experiment-group/treatment-pixel.jpg'

    '''
    Lambda at the Edge headers are array objects.

    Client may send multiple cookie headers. For example:
    > GET /viewerRes/test HTTP/1.1
    > User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1
 OpenSSL/1.0.1u zlib/1.2.3
    > Cookie: First=1; Second=2
    > Cookie: ClientCode=abc
    > Host: example.com

    You can access the first Cookie header at headers["cookie"][0].value
    and the second at headers["cookie"][1].value.

    Header values are not parsed. In the example above,
    headers["cookie"][0].value is equal to "First=1; Second=2"
    '''

    experimentUri = ""

    for cookie in headers.get('cookie', []):
        if cookieExperimentA in cookie['value']:
            print("Experiment A cookie found")
            experimentUri = pathExperimentA
            break
        elif cookieExperimentB in cookie['value']:
```

```
            print("Experiment B cookie found")
            experimentUri = pathExperimentB
            break

    if not experimentUri:
        print("Experiment cookie has not been found. Throwing dice...")
        if random.random() < 0.75:
            experimentUri = pathExperimentA
        else:
            experimentUri = pathExperimentB

    request['uri'] = experimentUri
    print(f"Request uri set to {experimentUri}")
    return request
```

Example: Overriding a response header

The following example shows how to change the value of a response header based on the value of another header.

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
   const response = event.Records[0].cf.response;
   const headers = response.headers;

   const headerNameSrc = 'X-Amz-Meta-Last-Modified';
   const headerNameDst = 'Last-Modified';

   if (headers[headerNameSrc.toLowerCase()]) {
      headers[headerNameDst.toLowerCase()] = [
         headers[headerNameSrc.toLowerCase()][0],
      ];
      console.log(`Response header "${headerNameDst}" was set to ` +
               `"${headers[headerNameDst.toLowerCase()][0].value}"`);
   }

   callback(null, response);
};
```

Python

```
import json

def lambda_handler(event, context):
    response = event["Records"][0]["cf"]["response"]
    headers = response["headers"]

    headerNameSrc = "X-Amz-Meta-Last-Modified"
    headerNameDst = "Last-Modified"

    if headers.get(headerNameSrc.lower(), None):
        headers[headerNameDst.lower()] = [headers[headerNameSrc.lower()][0]]
        print(f"Response header {headerNameDst.lower()} was set to
 {headers[headerNameSrc.lower()][0]}")

    return response
```

Generating responses - examples

The examples in this section show how you can use Lambda@Edge to generate responses.

Topics

- Example: Serving static content (generated response)
- Example: Generating an HTTP redirect (generated response)

Example: Serving static content (generated response)

The following example shows how to use a Lambda function to serve static website content, which reduces the load on the origin server and reduces overall latency.

> **Note**
>
> You can generate HTTP responses for viewer request and origin request events. For more information, see the section called “Generating HTTP responses in request triggers”.
> You can also replace or remove the body of the HTTP response in origin response events. For more information, see the section called “Updating HTTP responses in origin response triggers”.

Node.js

```
'use strict';

const content = `
<\!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Simple Lambda@Edge Static Content Response</title>
  </head>
  <body>
    <p>Hello from Lambda@Edge!</p>
  </body>
</html>
`;

exports.handler = (event, context, callback) => {
    /*
     * Generate HTTP OK response using 200 status code with HTML body.
     */
    const response = {
        status: '200',
        statusDescription: 'OK',
        headers: {
            'cache-control': [{
                key: 'Cache-Control',
                value: 'max-age=100'
            }],
            'content-type': [{
                key: 'Content-Type',
                value: 'text/html'
            }]
        },
        body: content,
    };
    callback(null, response);
};
```

Python

```
import json
```

```
CONTENT = """
<\!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Simple Lambda@Edge Static Content Response</title>
</head>
<body>
    <p>Hello from Lambda@Edge!</p>
</body>
</html>
"""

def lambda_handler(event, context):
    # Generate HTTP OK response using 200 status code with HTML body.
    response = {
        'status': '200',
        'statusDescription': 'OK',
        'headers': {
            'cache-control': [
                {
                    'key': 'Cache-Control',
                    'value': 'max-age=100'
                }
            ],
            "content-type": [
                {
                    'key': 'Content-Type',
                    'value': 'text/html'
                }
            ]
        },
        'body': CONTENT
    }
    return response
```

Example: Generating an HTTP redirect (generated response)

The following example shows how to generate an HTTP redirect.

Note

You can generate HTTP responses for viewer request and origin request events. For more information, see Generating HTTP responses in request triggers.

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    /*
     * Generate HTTP redirect response with 302 status code and Location header.
     */
    const response = {
        status: '302',
        statusDescription: 'Found',
        headers: {
            location: [{
                key: 'Location',
                value: 'https://docs.aws.amazon.com/lambda/latest/dg/lambda-
edge.html',
            }],
        },
    };
    callback(null, response);
};
```

Python

```
def lambda_handler(event, context):

    # Generate HTTP redirect response with 302 status code and Location header.

    response = {
        'status': '302',
        'statusDescription': 'Found',
        'headers': {
            'location': [{
                'key': 'Location',
                'value': 'https://docs.aws.amazon.com/lambda/latest/dg/lambda-
edge.html'
```

```
            }]
        }
    }

    return response
```

Working with query strings - examples

The examples in this section include ways that you can use Lambda@Edge with query strings.

Topics

- Example: Adding a header based on a query string parameter
- Example: Normalizing query string parameters to improve the cache hit ratio
- Example: Redirecting unauthenticated users to a sign-in page

Example: Adding a header based on a query string parameter

The following example shows how to get the key-value pair of a query string parameter, and then add a header based on those values.

Node.js

```
'use strict';

const querystring = require('querystring');
exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;

    /* When a request contains a query string key-value pair but the origin server
     * expects the value in a header, you can use this Lambda function to
     * convert the key-value pair to a header. Here's what the function does:
     * 1. Parses the query string and gets the key-value pair.
     * 2. Adds a header to the request using the key-value pair that the function
 got in step 1.
     */

    /* Parse request querystring to get javascript object */
    const params = querystring.parse(request.querystring);

    /* Move auth param from querystring to headers */
```

```
    const headerName = 'Auth-Header';
    request.headers[headerName.toLowerCase()] = [{ key: headerName, value:
 params.auth }];
    delete params.auth;

    /* Update request querystring */
    request.querystring = querystring.stringify(params);

    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs, urlencode

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    '''
    When a request contains a query string key-value pair but the origin server
    expects the value in a header, you can use this Lambda function to
    convert the key-value pair to a header. Here's what the function does:
        1. Parses the query string and gets the key-value pair.
        2. Adds a header to the request using the key-value pair that the function
 got in step 1.
    '''

    # Parse request querystring to get dictionary/json
    params = {k : v[0] for k, v in parse_qs(request['querystring']).items()}

    # Move auth param from querystring to headers
    headerName = 'Auth-Header'
    request['headers'][headerName.lower()] = [{'key': headerName, 'value':
 params['auth']}]
    del params['auth']

    # Update request querystring
    request['querystring'] = urlencode(params)

    return request
```

Example: Normalizing query string parameters to improve the cache hit ratio

The following example shows how to improve your cache hit ratio by making the following changes to query strings before CloudFront forwards requests to your origin:

- Alphabetize key-value pairs by the name of the parameter.
- Change the case of key-value pairs to lowercase.

For more information, see Caching content based on query string parameters.

Node.js

```
'use strict';

const querystring = require('querystring');

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    /* When you configure a distribution to forward query strings to the origin and
     * to cache based on an allowlist of query string parameters, we recommend
     * the following to improve the cache-hit ratio:
     * - Always list parameters in the same order.
     * - Use the same case for parameter names and values.
     *
     * This function normalizes query strings so that parameter names and values
     * are lowercase and parameter names are in alphabetical order.
     *
     * For more information, see:
     * https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/
QueryStringParameters.html
     */

    console.log('Query String: ', request.querystring);

    /* Parse request query string to get javascript object */
    const params = querystring.parse(request.querystring.toLowerCase());
    const sortedParams = {};

    /* Sort param keys */
    Object.keys(params).sort().forEach(key => {
        sortedParams[key] = params[key];
    });
```

```
    /* Update request querystring with normalized  */
    request.querystring = querystring.stringify(sortedParams);

    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs, urlencode

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    '''
    When you configure a distribution to forward query strings to the origin and
    to cache based on an allowlist of query string parameters, we recommend
    the following to improve the cache-hit ratio:
    Always list parameters in the same order.
    - Use the same case for parameter names and values.

    This function normalizes query strings so that parameter names and values
    are lowercase and parameter names are in alphabetical order.

    For more information, see:
    https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/
QueryStringParameters.html
    '''
    print("Query string: ", request["querystring"])

    # Parse request query string to get js object
    params = {k : v[0] for k, v in parse_qs(request['querystring'].lower()).items()}

    # Sort param keys
    sortedParams = sorted(params.items(), key=lambda x: x[0])

    # Update request querystring with normalized
    request['querystring'] = urlencode(sortedParams)

    return request
```

Example: Redirecting unauthenticated users to a sign-in page

The following example shows how to redirect users to a sign-in page if they haven't entered their credentials.

Node.js

```
'use strict';

function parseCookies(headers) {
    const parsedCookie = {};
    if (headers.cookie) {
        headers.cookie[0].value.split(';').forEach((cookie) => {
            if (cookie) {
                const parts = cookie.split('=');
                parsedCookie[parts[0].trim()] = parts[1].trim();
            }
        });
    }
    return parsedCookie;
}

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const headers = request.headers;

    /* Check for session-id in request cookie in viewer-request event,
     * if session-id is absent, redirect the user to sign in page with original
     * request sent as redirect_url in query params.
     */

    /* Check for session-id in cookie, if present then proceed with request */
    const parsedCookies = parseCookies(headers);
    if (parsedCookies && parsedCookies['session-id']) {
        callback(null, request);
        return;
    }

    /* URI encode the original request to be sent as redirect_url in query params */
    const encodedRedirectUrl = encodeURIComponent(`https://
${headers.host[0].value}${request.uri}?${request.querystring}`);
    const response = {
        status: '302',
        statusDescription: 'Found',
```

```
        headers: {
            location: [{
                key: 'Location',
                value: `https://www.example.com/signin?redirect_url=
${encodedRedirectUrl}`,
            }],
        },
    };
    callback(null, response);
};
```

Python

```
import urllib

def parseCookies(headers):
    parsedCookie = {}
    if headers.get('cookie'):
        for cookie in headers['cookie'][0]['value'].split(';'):
            if cookie:
                parts = cookie.split('=')
                parsedCookie[parts[0].strip()] = parts[1].strip()
    return parsedCookie

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    '''
    Check for session-id in request cookie in viewer-request event,
    if session-id is absent, redirect the user to sign in page with original
    request sent as redirect_url in query params.
    '''

    # Check for session-id in cookie, if present, then proceed with request
    parsedCookies = parseCookies(headers)

    if parsedCookies and parsedCookies['session-id']:
        return request

    # URI encode the original request to be sent as redirect_url in query params
    redirectUrl = "https://%s%s?%s" % (headers['host'][0]['value'], request['uri'],
 request['querystring'])
```

```
    encodedRedirectUrl = urllib.parse.quote_plus(redirectUrl.encode('utf-8'))

    response = {
        'status': '302',
        'statusDescription': 'Found',
        'headers': {
            'location': [{
                'key': 'Location',
                'value': 'https://www.example.com/signin?redirect_url=%s' %
encodedRedirectUrl
            }]
        }
    }
    return response
```

Personalize content by country or device type headers - examples

The examples in this section illustrate how you can use Lambda@Edge to customize behavior based on location or the type of device used by the viewer.

Topics

- Example: Redirecting viewer requests to a country-specific URL
- Example: Serving different versions of an object based on the device

Example: Redirecting viewer requests to a country-specific URL

The following example shows how to generate an HTTP redirect response with a country-specific URL and return the response to the viewer. This is useful when you want to provide country-specific responses. For example:

- If you have country-specific subdomains, such as us.example.com and tw.example.com, you can generate a redirect response when a viewer requests example.com.
- If you're streaming video but you don't have rights to stream the content in a specific country, you can redirect users in that country to a page that explains why they can't view the video.

Note the following:

- You must configure your distribution to cache based on the `CloudFront-Viewer-Country` header. For more information, see Cache based on selected request headers.
- CloudFront adds the `CloudFront-Viewer-Country` header after the viewer request event. To use this example, you must create a trigger for the origin request event.

Node.js

```
'use strict';

/* This is an origin request function */
exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const headers = request.headers;

    /*
     * Based on the value of the CloudFront-Viewer-Country header, generate an
     * HTTP status code 302 (Redirect) response, and return a country-specific
     * URL in the Location header.
     * NOTE: 1. You must configure your distribution to cache based on the
     *          CloudFront-Viewer-Country header. For more information, see
     *          https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-
headers
     *       2. CloudFront adds the CloudFront-Viewer-Country header after the
 viewer
     *          request event. To use this example, you must create a trigger for
 the
     *          origin request event.
     */

    let url = 'https://example.com/';
    if (headers['cloudfront-viewer-country']) {
        const countryCode = headers['cloudfront-viewer-country'][0].value;
        if (countryCode === 'TW') {
            url = 'https://tw.example.com/';
        } else if (countryCode === 'US') {
            url = 'https://us.example.com/';
        }
    }

    const response = {
        status: '302',
        statusDescription: 'Found',
```

```
        headers: {
            location: [{
                key: 'Location',
                value: url,
            }],
        },
    };
    callback(null, response);
};
```

Python

```
# This is an origin request function

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    '''
    Based on the value of the CloudFront-Viewer-Country header, generate an
    HTTP status code 302 (Redirect) response, and return a country-specific
    URL in the Location header.
    NOTE: 1. You must configure your distribution to cache based on the
            CloudFront-Viewer-Country header. For more information, see
            https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
          2. CloudFront adds the CloudFront-Viewer-Country header after the viewer
            request event. To use this example, you must create a trigger for the
            origin request event.
    '''

    url = 'https://example.com/'
    viewerCountry = headers.get('cloudfront-viewer-country')
    if viewerCountry:
        countryCode = viewerCountry[0]['value']
        if countryCode == 'TW':
            url = 'https://tw.example.com/'
        elif countryCode == 'US':
            url = 'https://us.example.com/'

    response = {
        'status': '302',
        'statusDescription': 'Found',
        'headers': {
```

```
            'location': [{
                'key': 'Location',
                'value': url
            }]
        }
    }

    return response
```

Example: Serving different versions of an object based on the device

The following example shows how to serve different versions of an object based on the type of device that the user is using, for example, a mobile device or a tablet. Note the following:

- You must configure your distribution to cache based on the `CloudFront-Is-*-Viewer` headers. For more information, see Cache based on selected request headers.
- CloudFront adds the `CloudFront-Is-*-Viewer` headers after the viewer request event. To use this example, you must create a trigger for the origin request event.

Node.js

```
'use strict';

/* This is an origin request function */
exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const headers = request.headers;

    /*
     * Serve different versions of an object based on the device type.
     * NOTE: 1. You must configure your distribution to cache based on the
     *          CloudFront-Is-*-Viewer headers. For more information, see
     *          the following documentation:
     *          https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-
headers
     *          https://docs.aws.amazon.com/console/cloudfront/cache-on-device-type
     *       2. CloudFront adds the CloudFront-Is-*-Viewer headers after the viewer
     *          request event. To use this example, you must create a trigger for
 the
     *          origin request event.
     */
```

```
    const desktopPath = '/desktop';
    const mobilePath = '/mobile';
    const tabletPath = '/tablet';
    const smarttvPath = '/smarttv';

    if (headers['cloudfront-is-desktop-viewer']
        && headers['cloudfront-is-desktop-viewer'][0].value === 'true') {
        request.uri = desktopPath + request.uri;
    } else if (headers['cloudfront-is-mobile-viewer']
               && headers['cloudfront-is-mobile-viewer'][0].value === 'true') {
        request.uri = mobilePath + request.uri;
    } else if (headers['cloudfront-is-tablet-viewer']
               && headers['cloudfront-is-tablet-viewer'][0].value === 'true') {
        request.uri = tabletPath + request.uri;
    } else if (headers['cloudfront-is-smarttv-viewer']
               && headers['cloudfront-is-smarttv-viewer'][0].value === 'true') {
        request.uri = smarttvPath + request.uri;
    }
    console.log(`Request uri set to "${request.uri}"`);

    callback(null, request);
};
```

Python

```
# This is an origin request function
def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    '''
    Serve different versions of an object based on the device type.
    NOTE: 1. You must configure your distribution to cache based on the
            CloudFront-Is-*-Viewer headers. For more information, see
            the following documentation:
            https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
            https://docs.aws.amazon.com/console/cloudfront/cache-on-device-type
          2. CloudFront adds the CloudFront-Is-*-Viewer headers after the viewer
            request event. To use this example, you must create a trigger for the
            origin request event.
    '''
```

```
    desktopPath = '/desktop';
    mobilePath = '/mobile';
    tabletPath = '/tablet';
    smarttvPath = '/smarttv';

    if 'cloudfront-is-desktop-viewer' in headers and headers['cloudfront-is-desktop-
viewer'][0]['value'] == 'true':
        request['uri'] = desktopPath + request['uri']
    elif 'cloudfront-is-mobile-viewer' in headers and headers['cloudfront-is-mobile-
viewer'][0]['value'] == 'true':
        request['uri'] = mobilePath + request['uri']
    elif 'cloudfront-is-tablet-viewer' in headers and headers['cloudfront-is-tablet-
viewer'][0]['value'] == 'true':
        request['uri'] = tabletPath + request['uri']
    elif 'cloudfront-is-smarttv-viewer' in headers and headers['cloudfront-is-
smarttv-viewer'][0]['value'] == 'true':
        request['uri'] = smarttvPath + request['uri']

    print("Request uri set to %s" % request['uri'])

    return request
```

Content-based dynamic origin selection - examples

The examples in this section show how you can use Lambda@Edge to route to different origins based on information in the request.

Topics

- Example: Using an origin request trigger to change from a custom origin to an Amazon S3 origin
- Example: Using an origin-request trigger to change the Amazon S3 origin Region
- Example: Using an origin request trigger to change from an Amazon S3 origin to a custom origin
- Example: Using an origin request trigger to gradually transfer traffic from one Amazon S3 bucket to another
- Example: Using an origin request trigger to change the origin domain name based on the country header

Example: Using an origin request trigger to change from a custom origin to an Amazon S3 origin

This function demonstrates how an origin-request trigger can be used to change from a custom origin to an Amazon S3 origin from which the content is fetched, based on request properties.

Node.js

```
'use strict';

 const querystring = require('querystring');

 exports.handler = (event, context, callback) => {
     const request = event.Records[0].cf.request;

     /**
      * Reads query string to check if S3 origin should be used, and
      * if true, sets S3 origin properties.
      */

     const params = querystring.parse(request.querystring);

     if (params['useS3Origin']) {
         if (params['useS3Origin'] === 'true') {
             const s3DomainName = 'my-bucket.s3.amazonaws.com';

             /* Set S3 origin fields */
             request.origin = {
                 s3: {
                     domainName: s3DomainName,
                     region: '',
                     authMethod: 'none',
                     path: '',
                     customHeaders: {}
                 }
             };
             request.headers['host'] = [{ key: 'host', value: s3DomainName}];
         }
     }

    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    '''
    Reads query string to check if S3 origin should be used, and
    if true, sets S3 origin properties
    '''
    params = {k: v[0] for k, v in parse_qs(request['querystring']).items()}
    if params.get('useS3Origin') == 'true':
        s3DomainName = 'my-bucket.s3.amazonaws.com'

        # Set S3 origin fields
        request['origin'] = {
            's3': {
                'domainName': s3DomainName,
                'region': '',
                'authMethod': 'none',
                'path': '',
                'customHeaders': {}
            }
        }
        request['headers']['host'] = [{'key': 'host', 'value': s3DomainName}]
    return request
```

Example: Using an origin-request trigger to change the Amazon S3 origin Region

This function demonstrates how an origin-request trigger can be used to change the Amazon S3 origin from which the content is fetched, based on request properties.

In this example, we use the value of the `CloudFront-Viewer-Country` header to update the S3 bucket domain name to a bucket in a Region that is closer to the viewer. This can be useful in several ways:

- It reduces latencies when the Region specified is nearer to the viewer's country.
- It provides data sovereignty by making sure that data is served from an origin that's in the same country that the request came from.

To use this example, you must do the following:

- Configure your distribution to cache based on the `CloudFront-Viewer-Country` header. For more information, see Cache based on selected request headers.
- Create a trigger for this function in the origin request event. CloudFront adds the `CloudFront-Viewer-Country` header after the viewer request event, so to use this example, you must make sure that the function executes for an origin request.

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;

    /**
     * This blueprint demonstrates how an origin-request trigger can be used to
     * change the origin from which the content is fetched, based on request
 properties.
     * In this example, we use the value of the CloudFront-Viewer-Country header
     * to update the S3 bucket domain name to a bucket in a Region that is closer to
     * the viewer.
     *
     * This can be useful in several ways:
     *      1) Reduces latencies when the Region specified is nearer to the viewer's
     *         country.
     *      2) Provides data sovereignty by making sure that data is served from an
     *         origin that's in the same country that the request came from.
     *
     * NOTE: 1. You must configure your distribution to cache based on the
     *          CloudFront-Viewer-Country header. For more information, see
     *          https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-
headers
     *       2. CloudFront adds the CloudFront-Viewer-Country header after the
 viewer
     *          request event. To use this example, you must create a trigger for
 the
     *          origin request event.
     */

    const countryToRegion = {
        'DE': 'eu-central-1',
```

```
        'IE': 'eu-west-1',
        'GB': 'eu-west-2',
        'FR': 'eu-west-3',
        'JP': 'ap-northeast-1',
        'IN': 'ap-south-1'
    };

    if (request.headers['cloudfront-viewer-country']) {
        const countryCode = request.headers['cloudfront-viewer-country'][0].value;
        const region = countryToRegion[countryCode];

        /**
         * If the viewer's country is not in the list you specify, the request
         * goes to the default S3 bucket you've configured.
         */
        if (region) {
            /**
             * If you've set up OAI, the bucket policy in the destination bucket
             * should allow the OAI GetObject operation, as configured by default
             * for an S3 origin with OAI. Another requirement with OAI is to provide
             * the Region so it can be used for the SIGV4 signature. Otherwise, the
             * Region is not required.
             */
            request.origin.s3.region = region;
            const domainName = `my-bucket-in-${region}.s3.amazonaws.com`;
            request.origin.s3.domainName = domainName;
            request.headers['host'] = [{ key: 'host', value: domainName }];
        }
    }

    callback(null, request);
};
```

Python

```
def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    '''
    This blueprint demonstrates how an origin-request trigger can be used to
    change the origin from which the content is fetched, based on request
 properties.
    In this example, we use the value of the CloudFront-Viewer-Country header
```

```
    to update the S3 bucket domain name to a bucket in a Region that is closer to
    the viewer.

    This can be useful in several ways:
        1) Reduces latencies when the Region specified is nearer to the viewer's
            country.
        2) Provides data sovereignty by making sure that data is served from an
            origin that's in the same country that the request came from.

    NOTE: 1. You must configure your distribution to cache based on the
            CloudFront-Viewer-Country header. For more information, see
            https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
          2. CloudFront adds the CloudFront-Viewer-Country header after the viewer
            request event. To use this example, you must create a trigger for the
            origin request event.
    '''

    countryToRegion = {
        'DE': 'eu-central-1',
        'IE': 'eu-west-1',
        'GB': 'eu-west-2',
        'FR': 'eu-west-3',
        'JP': 'ap-northeast-1',
        'IN': 'ap-south-1'
    }

    viewerCountry = request['headers'].get('cloudfront-viewer-country')
    if viewerCountry:
        countryCode = viewerCountry[0]['value']
        region = countryToRegion.get(countryCode)

        # If the viewer's country in not in the list you specify, the request
        # goes to the default S3 bucket you've configured
        if region:
            '''
            If you've set up OAI, the bucket policy in the destination bucket
            should allow the OAI GetObject operation, as configured by default
            for an S3 origin with OAI. Another requirement with OAI is to provide
            the Region so it can be used for the SIGV4 signature. Otherwise, the
            Region is not required.
            '''
            request['origin']['s3']['region'] = region
            domainName = 'my-bucket-in-%s.s3.amazonaws.com' % region
            request['origin']['s3']['domainName'] = domainName
```

```
            request['headers']['host'] = [{'key': 'host', 'value': domainName}]

    return request
```

Example: Using an origin request trigger to change from an Amazon S3 origin to a custom origin

This function demonstrates how an origin-request trigger can be used to change the custom origin from which the content is fetched, based on request properties.

Node.js

```
'use strict';

const querystring = require('querystring');

 exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;

    /**
     * Reads query string to check if custom origin should be used, and
     * if true, sets custom origin properties.
     */

    const params = querystring.parse(request.querystring);

    if (params['useCustomOrigin']) {
        if (params['useCustomOrigin'] === 'true') {

            /* Set custom origin fields*/
            request.origin = {
                custom: {
                    domainName: 'www.example.com',
                    port: 443,
                    protocol: 'https',
                    path: '',
                    sslProtocols: ['TLSv1', 'TLSv1.1'],
                    readTimeout: 5,
                    keepaliveTimeout: 5,
                    customHeaders: {}
                }
            };
```

```
            request.headers['host'] = [{ key: 'host', value: 'www.example.com'}];
        }
     }
    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    # Reads query string to check if custom origin should be used, and
    # if true, sets custom origin properties

    params = {k: v[0] for k, v in parse_qs(request['querystring']).items()}

    if params.get('useCustomOrigin') == 'true':
            # Set custom origin fields
            request['origin'] = {
                'custom': {
                    'domainName': 'www.example.com',
                    'port': 443,
                    'protocol': 'https',
                    'path': '',
                    'sslProtocols': ['TLSv1', 'TLSv1.1'],
                    'readTimeout': 5,
                    'keepaliveTimeout': 5,
                    'customHeaders': {}
                }
            }
            request['headers']['host'] = [{'key': 'host', 'value':
 'www.example.com'}]

    return request
```

Example: Using an origin request trigger to gradually transfer traffic from one Amazon S3 bucket to another

This function demonstrates how you can gradually transfer traffic from one Amazon S3 bucket to another, in a controlled way.

Node.js

```
'use strict';

    function getRandomInt(min, max) {
        /* Random number is inclusive of min and max*/
        return Math.floor(Math.random() * (max - min + 1)) + min;
 }

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const BLUE_TRAFFIC_PERCENTAGE = 80;

    /**
      * This Lambda function demonstrates how to gradually transfer traffic from
      * one S3 bucket to another in a controlled way.
      * We define a variable BLUE_TRAFFIC_PERCENTAGE which can take values from
      * 1 to 100. If the generated randomNumber less than or equal to
 BLUE_TRAFFIC_PERCENTAGE, traffic
      * is re-directed to blue-bucket. If not, the default bucket that we've
 configured
      * is used.
      */

    const randomNumber = getRandomInt(1, 100);

if (randomNumber <= BLUE_TRAFFIC_PERCENTAGE) {
         const domainName = 'blue-bucket.s3.amazonaws.com';
         request.origin.s3.domainName = domainName;
         request.headers['host'] = [{ key: 'host', value: domainName}];
     }
    callback(null, request);
};
```

Python

```
import math
import random

def getRandomInt(min, max):
    # Random number is inclusive of min and max
    return math.floor(random.random() * (max - min + 1)) + min
```

```
def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    BLUE_TRAFFIC_PERCENTAGE = 80

    '''
    This Lambda function demonstrates how to gradually transfer traffic from
    one S3 bucket to another in a controlled way.
    We define a variable BLUE_TRAFFIC_PERCENTAGE which can take values from
    1 to 100. If the generated randomNumber less than or equal to
 BLUE_TRAFFIC_PERCENTAGE, traffic
    is re-directed to blue-bucket. If not, the default bucket that we've configured
    is used.
    '''

    randomNumber = getRandomInt(1, 100)

    if randomNumber <= BLUE_TRAFFIC_PERCENTAGE:
        domainName = 'blue-bucket.s3.amazonaws.com'
        request['origin']['s3']['domainName'] = domainName
        request['headers']['host'] = [{'key': 'host', 'value': domainName}]

    return request
```

Example: Using an origin request trigger to change the origin domain name based on the country header

This function demonstrates how you can change the origin domain name based on the `CloudFront-Viewer-Country` header, so content is served from an origin closer to the viewer's country.

Implementing this functionality for your distribution can have advantages such as the following:

- Reducing latencies when the Region specified is nearer to the viewer's country
- Providing data sovereignty by making sure that data is served from an origin that's in the same country that the request came from

Note that to enable this functionality you must configure your distribution to cache based on the `CloudFront-Viewer-Country` header. For more information, see the section called “Cache based on selected request headers”.

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
     const request = event.Records[0].cf.request;

  if (request.headers['cloudfront-viewer-country']) {
        const countryCode = request.headers['cloudfront-viewer-country'][0].value;
        if (countryCode === 'GB' || countryCode === 'DE' || countryCode === 'IE' )
 {
            const domainName = 'eu.example.com';
            request.origin.custom.domainName = domainName;
            request.headers['host'] = [{key: 'host', value: domainName}];
        }
     }

    callback(null, request);
};
```

Python

```
def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    viewerCountry = request['headers'].get('cloudfront-viewer-country')
    if viewerCountry:
        countryCode = viewerCountry[0]['value']
        if countryCode == 'GB' or countryCode == 'DE' or countryCode == 'IE':
            domainName = 'eu.example.com'
            request['origin']['custom']['domainName'] = domainName
            request['headers']['host'] = [{'key': 'host', 'value': domainName}]
    return request
```

Updating error statuses - examples

The examples in this section provide guidance for how you can use Lambda@Edge to change the error status that is returned to users.

Topics

- Example: Using an origin response trigger to update the error status code to 200

- Example: Using an origin response trigger to update the error status code to 302

Example: Using an origin response trigger to update the error status code to 200

This function demonstrates how you can update the response status to 200 and generate static body content to return to the viewer in the following scenario:

- The function is triggered in an origin response.
- The response status from the origin server is an error status code (4xx or 5xx).

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const response = event.Records[0].cf.response;

    /**
     * This function updates the response status to 200 and generates static
     * body content to return to the viewer in the following scenario:
     * 1. The function is triggered in an origin response
     * 2. The response status from the origin server is an error status code (4xx or
 5xx)
     */

    if (response.status >= 400 && response.status <= 599) {
        response.status = 200;
        response.statusDescription = 'OK';
        response.body = 'Body generation example';
    }

    callback(null, response);
};
```

Python

```
def lambda_handler(event, context):
    response = event['Records'][0]['cf']['response']

    '''
    This function updates the response status to 200 and generates static
```

```
    body content to return to the viewer in the following scenario:
    1. The function is triggered in an origin response
    2. The response status from the origin server is an error status code (4xx or
 5xx)
    '''

    if int(response['status']) >= 400 and int(response['status']) <= 599:
        response['status'] = 200
        response['statusDescription'] = 'OK'
        response['body'] = 'Body generation example'
    return response
```

Example: Using an origin response trigger to update the error status code to 302

This function demonstrates how you can update the HTTP status code to 302 to redirect to another path (cache behavior) that has a different origin configured. Note the following:

- The function is triggered in an origin response.
- The response status from the origin server is an error status code (4xx or 5xx).

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const response = event.Records[0].cf.response;
    const request = event.Records[0].cf.request;

    /**
     * This function updates the HTTP status code in the response to 302, to
 redirect to another
     * path (cache behavior) that has a different origin configured. Note the
 following:
     * 1. The function is triggered in an origin response
     * 2. The response status from the origin server is an error status code (4xx or
 5xx)
     */

    if (response.status >= 400 && response.status <= 599) {
        const redirect_path = `/plan-b/path?${request.querystring}`;
```

```
        response.status = 302;
        response.statusDescription = 'Found';

        /* Drop the body, as it is not required for redirects */
        response.body = '';
        response.headers['location'] = [{ key: 'Location', value: redirect_path }];
    }

    callback(null, response);
};
```

Python

```
def lambda_handler(event, context):
    response = event['Records'][0]['cf']['response']
    request = event['Records'][0]['cf']['request']

    '''
    This function updates the HTTP status code in the response to 302, to redirect
 to another
    path (cache behavior) that has a different origin configured. Note the
 following:
    1. The function is triggered in an origin response
    2. The response status from the origin server is an error status code (4xx or
 5xx)
    '''

    if int(response['status']) >= 400 and int(response['status']) <= 599:
        redirect_path = '/plan-b/path?%s' % request['querystring']

        response['status'] = 302
        response['statusDescription'] = 'Found'

        # Drop the body as it is not required for redirects
        response['body'] = ''
        response['headers']['location'] = [{'key': 'Location', 'value':
 redirect_path}]

    return response
```

Accessing the request body - examples

The examples in this section illustrate how you can use Lambda@Edge to work with POST requests.

> **Note**
>
> To use these examples, you must enable the *include body* option in the distribution's Lambda function association. It is not enabled by default.
>
> - To enable this setting in the CloudFront console, select the check box for **Include Body** in the **Lambda Function Association**.
> - To enable this setting in the CloudFront API or with Amazon CloudFormation, set the `IncludeBody` field to `true` in `LambdaFunctionAssociation`.

Topics

- Example: Using a request trigger to read an HTML form
- Example: Using a request trigger to modify an HTML form

Example: Using a request trigger to read an HTML form

This function demonstrates how you can process the body of a POST request generated by an HTML form (web form), such as a "contact us" form. For example, you might have an HTML form like the following:

```
<html>
  <form action="https://example.com" method="post">
    Param 1: <input type="text" name="name1"><br>
    Param 2: <input type="text" name="name2"><br>
    input type="submit" value="Submit">
  </form>
</html>
```

For the example function that follows, the function must be triggered in a CloudFront viewer request or origin request.

Node.js

```
'use strict';
```

```
const querystring = require('querystring');

/**
 * This function demonstrates how you can read the body of a POST request
 * generated by an HTML form (web form). The function is triggered in a
 * CloudFront viewer request or origin request event type.
 */

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;

    if (request.method === 'POST') {
        /* HTTP body is always passed as base64-encoded string. Decode it. */
        const body = Buffer.from(request.body.data, 'base64').toString();

        /* HTML forms send the data in query string format. Parse it. */
        const params = querystring.parse(body);

        /* For demonstration purposes, we only log the form fields here.
         * You can put your custom logic here. For example, you can store the
         * fields in a database, such as Amazon DynamoDB, and generate a response
         * right from your Lambda@Edge function.
         */
        for (let param in params) {
            console.log(`For "${param}" user submitted "${params[param]}".\n`);
        }
    }
    return callback(null, request);
};
```

Python

```
import base64
from urllib.parse import parse_qs

'''
Say there is a POST request body generated by an HTML such as:

<html>
<form action="https://example.com" method="post">
    Param 1: <input type="text" name="name1"><br>
    Param 2: <input type="text" name="name2"><br>
```

```
    input type="submit" value="Submit">
</form>
</html>

'''

'''
This function demonstrates how you can read the body of a POST request
generated by an HTML form (web form). The function is triggered in a
CloudFront viewer request or origin request event type.
'''

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    if request['method'] == 'POST':
        # HTTP body is always passed as base64-encoded string. Decode it
        body = base64.b64decode(request['body']['data'])

        # HTML forms send the data in query string format. Parse it
        params = {k: v[0] for k, v in parse_qs(body).items()}

        '''
        For demonstration purposes, we only log the form fields here.
        You can put your custom logic here. For example, you can store the
        fields in a database, such as Amazon DynamoDB, and generate a response
        right from your Lambda@Edge function.
        '''
        for key, value in params.items():
            print("For %s use submitted %s" % (key, value))

    return request
```

Example: Using a request trigger to modify an HTML form

This function demonstrates how you can modify the body of a POST request generated by an HTML form (web form). The function is triggered in a CloudFront viewer request or origin request.

Node.js

```
'use strict';
```

```
const querystring = require('querystring');

exports.handler = (event, context, callback) => {
    var request = event.Records[0].cf.request;
    if (request.method === 'POST') {
        /* Request body is being replaced. To do this, update the following
        /* three fields:
         *    1) body.action to 'replace'
         *    2) body.encoding to the encoding of the new data.
         *
         *       Set to one of the following values:
         *
         *           text - denotes that the generated body is in text format.
         *               Lambda@Edge will propagate this as is.
         *           base64 - denotes that the generated body is base64 encoded.
         *               Lambda@Edge will base64 decode the data before sending
         *               it to the origin.
         *    3) body.data to the new body.
         */
        request.body.action = 'replace';
        request.body.encoding = 'text';
        request.body.data = getUpdatedBody(request);
    }
    callback(null, request);
};

function getUpdatedBody(request) {
    /* HTTP body is always passed as base64-encoded string. Decode it. */
    const body = Buffer.from(request.body.data, 'base64').toString();

    /* HTML forms send data in query string format. Parse it. */
    const params = querystring.parse(body);

    /* For demonstration purposes, we're adding one more param.
     *
     * You can put your custom logic here. For example, you can truncate long
     * bodies from malicious requests.
     */
    params['new-param-name'] = 'new-param-value';
    return querystring.stringify(params);
}
```

Python

```
import base64
from urllib.parse import parse_qs, urlencode

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    if request['method'] == 'POST':
        '''
        Request body is being replaced. To do this, update the following
        three fields:
            1) body.action to 'replace'
            2) body.encoding to the encoding of the new data.

            Set to one of the following values:

                text - denotes that the generated body is in text format.
                    Lambda@Edge will propagate this as is.
                base64 - denotes that the generated body is base64 encoded.
                    Lambda@Edge will base64 decode the data before sending
                    it to the origin.
            3) body.data to the new body.
        '''
        request['body']['action'] = 'replace'
        request['body']['encoding'] = 'text'
        request['body']['data'] = getUpdatedBody(request)
    return request

def getUpdatedBody(request):
    # HTTP body is always passed as base64-encoded string. Decode it
    body = base64.b64decode(request['body']['data'])

    # HTML forms send data in query string format. Parse it
    params = {k: v[0] for k, v in parse_qs(body).items()}

    # For demonstration purposes, we're adding one more param

    # You can put your custom logic here. For example, you can truncate long
    # bodies from malicious requests
    params['new-param-name'] = 'new-param-value'
    return urlencode(params)
```

Restrictions on edge functions

The following topics describe the restrictions that apply to CloudFront Functions and Lambda@Edge. Some restrictions apply to all edge functions, while others apply only to CloudFront Functions or Lambda@Edge.

For information about quotas (formerly referred to as limits), see Quotas on CloudFront Functions and Quotas on Lambda@Edge.

Topics

- Restrictions on all edge functions
- Restrictions on CloudFront Functions
- Restrictions on Lambda@Edge

Restrictions on all edge functions

The following restrictions apply to all edge functions, both CloudFront Functions and Lambda@Edge.

Topics

- Amazon Web Services account ownership
- Combining CloudFront Functions with Lambda@Edge
- HTTP status codes
- HTTP headers
- Query strings
- URI
- URI and query string encoding
- Microsoft Smooth Streaming
- Tagging

Amazon Web Services account ownership

To associate an edge function with a CloudFront distribution, the function and distribution must be owned by the same Amazon Web Services account.

Combining CloudFront Functions with Lambda@Edge

For a given cache behavior, the following restrictions apply:

- Each event type (viewer request, origin request, origin response, and viewer response) can have only one edge function association.
- You cannot combine CloudFront Functions and Lambda@Edge in viewer events (viewer request and viewer response).

All other combinations of edge functions are allowed. The following table explains the allowed combinations.

		CloudFront Functions	
		Viewer request	**Viewer response**
Lambda@Edge	**Viewer request**	Not allowed	Not allowed
	Origin request	Allowed	Allowed
	Origin response	Allowed	Allowed
	Viewer response	Not allowed	Not allowed

HTTP status codes

CloudFront does not invoke edge functions for viewer response events when the origin returns HTTP status code 400 or higher.

Lambda@Edge functions for origin response events are invoked for *all* origin responses, including when the origin returns HTTP status code 400 or higher. For more information, see Updating HTTP responses in origin response triggers.

HTTP headers

Certain HTTP headers are disallowed, which means they're not exposed to edge functions and functions can't add them. Other headers are read-only, which means functions can read them but can't add or modify them.

Topics

- Disallowed headers
- Read-only headers

Disallowed headers

The following HTTP headers are not exposed to edge functions, and functions can't add them. If your function adds one of these headers, it fails CloudFront validation and CloudFront returns HTTP status code 502 (Bad Gateway) to the viewer.

- `Connection`
- `Expect`
- `Keep-Alive`
- `Proxy-Authenticate`
- `Proxy-Authorization`
- `Proxy-Connection`
- `Trailer`
- `Upgrade`
- `X-Accel-Buffering`
- `X-Accel-Charset`
- `X-Accel-Limit-Rate`
- `X-Accel-Redirect`
- `X-Amz-Cf-*`
- `X-Amzn-Auth`
- `X-Amzn-Cf-Billing`
- `X-Amzn-Cf-Id`
- `X-Amzn-Cf-Xff`
- `X-Amzn-Errortype`
- `X-Amzn-Fle-Profile`
- `X-Amzn-Header-Count`
- `X-Amzn-Header-Order`

- `X-Amzn-Lambda-Integration-Tag`
- `X-Amzn-RequestId`
- `X-Cache`
- `X-Edge-*`
- `X-Forwarded-Proto`
- `X-Real-IP`

Read-only headers

The following headers are read-only. Your function can read them and use them as input to the function logic, but it can't change the values. If your function adds or edits a read-only header, the request fails CloudFront validation and CloudFront returns HTTP status code 502 (Bad Gateway) to the viewer.

Read-only headers in viewer request events

The following headers are read-only in viewer request events.

- `Content-Length`
- `Host`
- `Transfer-Encoding`
- `Via`

Read-only headers in origin request events (Lambda@Edge only)

The following headers are read-only in origin request events, which exist only in Lambda@Edge.

- `Accept-Encoding`
- `Content-Length`
- `If-Modified-Since`
- `If-None-Match`
- `If-Range`
- `If-Unmodified-Since`
- `Transfer-Encoding`

- `Via`

Read-only headers in origin response events (Lambda@Edge only)

The following headers are read-only in origin response events, which exist only in Lambda@Edge.

- `Transfer-Encoding`
- `Via`

Read-only headers in viewer response events

The following headers are read-only in viewer response events for both CloudFront Functions and Lambda@Edge.

- `Warning`
- `Via`

The following headers are read-only in viewer response events for Lambda@Edge.

- `Content-Length`
- `Content-Encoding`
- `Transfer-Encoding`

Query strings

The following restrictions apply to functions that read, update, or create a query string in a request URI.

- (Lambda@Edge only) To access the query string in an origin request or origin response function, your cache policy or origin request policy must be set to **All** for **Query strings**.
- A function can create or update a query string for viewer request and origin request events (origin request events exist only in Lambda@Edge).
- A function can read a query string, but cannot create or update one, for origin response and viewer response events (origin response events exist only in Lambda@Edge).
- If a function creates or updates a query string, the following restrictions apply:
  - The query string can't include spaces, control characters, or the fragment identifier (#).

- The total size of the URI, including the query string, must be less than 8,192 characters.
- We recommend that you use percent encoding for the URI and query string. For more information, see URI and query string encoding.

URI

If a function changes the URI for a request, that doesn't change the cache behavior for the request or the origin that the request is forwarded to.

The total size of the URI, including the query string, must be less than 8,192 characters.

URI and query string encoding

URI and query string values passed to edge functions are UTF-8 encoded. Your function should use UTF-8 encoding for the URI and query string values that it returns. Percent encoding is compatible with UTF-8 encoding.

The following list explains how CloudFront handles URI and query string value encoding:

- When values in the request are UTF-8 encoded, CloudFront forwards the values to your function without changing them.
- When values in the request are ISO-8859-1 encoded, CloudFront converts the values to UTF-8 encoding before forwarding them to your function.
- When values in the request are encoded using some other character encoding, CloudFront assumes that they're ISO-8859-1 encoded and tries to convert from ISO-8859-1 to UTF-8.

> **Important**
>
> The converted characters might be an inaccurate interpretation of the values in the original request. This might cause your function or your origin to produce an unintended result.

The URI and query string values that CloudFront forwards to your origin depend on whether a function changes the values:

- If a function does not change the URI or query string, CloudFront forwards the values that it received in the request to your origin.

- If a function changes the URI or query string, CloudFront forwards the UTF-8 encoded values.

Microsoft Smooth Streaming

You cannot use edge functions with a CloudFront distribution that you're using for streaming media files that you've transcoded into the Microsoft Smooth Streaming format.

Tagging

You cannot add tags to edge functions. To learn more about tagging in CloudFront, see Tagging Amazon CloudFront distributions.

Restrictions on CloudFront Functions

The following restrictions apply only to CloudFront Functions.

For information about quotas (formerly referred to as limits), see Quotas on CloudFront Functions.

Logs

Function logs in CloudFront Functions are truncated at 10 KB.

Request body

CloudFront Functions cannot access the body of the HTTP request.

Regional Amazon Security Token Service endpoints when using the CloudFront KeyValueStore API

When you call the CloudFront KeyValueStore API by using Signature Version 4A (SigV4A) with temporary security credentials—for example, when using Amazon Identity and Access Management (IAM) roles—make sure that you request the temporary credentials from a Regional endpoint in Amazon STS. If you use the global endpoint for Amazon STS (`sts.amazonaws.com`), Amazon STS will generate temporary credentials from a global endpoint, which isn't supported by SigV4A. As a result, you will receive an authentication error. To resolve this issue, use any of the listed Regional endpoints for Amazon STS in the *IAM User Guide*. If you're configuring SAML to use Amazon STS regional endpoints, see the How to use regional SAML endpoints for failover blog post.

Runtime

The CloudFront Functions runtime environment does not support dynamic code evaluation, and it restricts access to the network, file system, and timers. For more information, see Restricted features.

Compute utilization

CloudFront Functions have a limit on the time they can take to run, measured as *compute utilization*. Compute utilization is a number between 0 and 100 that indicates the amount of time that the function took to run as a percentage of the maximum allowed time. For example, a compute utilization of 35 means that the function completed in 35% of the maximum allowed time.

When you test a function, you can see the compute utilization value in the output of the test event. For production functions, you can view the compute utilization metric on the Monitoring page in the CloudFront console, or in CloudWatch.

Restrictions on Lambda@Edge

The following restrictions apply only to Lambda@Edge.

For information about quotas, see Quotas on Lambda@Edge.

DNS resolution

CloudFront performs a DNS resolution on the origin domain name *before* it executes your origin request Lambda@Edge function. If the DNS service for your domain is experiencing issues and CloudFront can't resolve the domain name to get the IP address, your Lambda@Edge function will not invoke. CloudFront will return an HTTP 502 status code (Bad Gateway) to the client. For more information, see HTTP 502 status code (DNS error).

For more information about managing DNS failover, see Configuring DNS failover in the *Amazon Route 53 Developer Guide*.

HTTP status codes

Lambda@Edge functions for viewer response events cannot modify the HTTP status code of the response, regardless of whether the response came from the origin or the CloudFront cache.

Lambda function version

You must use a numbered version of the Lambda function, not `$LATEST` or aliases.

Lambda Region

The Lambda function must be in the US East (N. Virginia) Region.

Lambda role permissions

The IAM execution role associated with the Lambda function must allow the service principals `lambda.amazonaws.com` and `edgelambda.amazonaws.com` to assume the role. For more information, see Setting IAM permissions and roles for Lambda@Edge.

Lambda features

The following Lambda features are not supported by Lambda@Edge:

- Lambda runtime management configurations other than **Auto** (default)
- Configuration of your Lambda function to access resources inside your VPC
- Lambda function dead letter queues
- Lambda environment variables (except for reserved environment variables, which are automatically supported)
- Lambda functions with Amazon Lambda layers
- Using Amazon X-Ray
- Lambda provisioned concurrency

> **Note**
>
> Lambda@Edge functions have the same Regional concurrency capabilities as Lambda functions. However, when the quota is increased for concurrent Lambda@Edge executions, the quota is increased for all the Amazon Web Services Regions where the Lambda@Edge function is replicated. For more information, see Quotas on Lambda@Edge.

- Lambda functions defined as container images
- Lambda functions that use the arm64 architecture
- Lambda functions with more than 512 MB of ephemeral storage

Supported runtimes

Lambda@Edge supports Lambda functions with the following runtimes:

Node.js	Python
• Node.js 20 • Node.js 18 • Node.js 16[1] • Node.js 14[2] • Node.js 12[2] • Node.js 10[2] • Node.js 8[2] • Node.js 6[2]	• Python 3.12 • Python 3.11 • Python 3.10 • Python 3.9 • Python 3.8 • Python 3.7

[1]This version of Node.js has reached end of life, and will soon be deprecated by Amazon Lambda.

[2]This version of Node.js has reached end of life, and is fully deprecated by Amazon Lambda.

You can't create or update functions with deprecated versions of Node.js. You can only associate existing functions with these versions with CloudFront distributions. Functions with these versions that are associated with distributions will continue to run. However, we recommend that you move your function to newer versions of Node.js. For more information, see Runtime deprecation policy in the *Amazon Lambda Developer Guide* and the Node.js release schedule on GitHub.

> **Tip**
>
> As a best practice, use the latest versions of the provided runtimes for performance improvements and new features.

CloudFront headers

Lambda@Edge functions can read, edit, remove, or add any of the CloudFront headers listed in Adding CloudFront request headers.

Notes

- If you want CloudFront to add these headers, you must configure CloudFront to add them by using a cache policy or origin request policy.
- CloudFront adds the headers *after* the viewer request event, which means the headers aren't available to Lambda@Edge functions in a viewer request. The headers are only available to Lambda@Edge functions in an origin request and origin response.
- If the viewer request includes headers that have these names, and you configured CloudFront to add these headers using a cache policy or origin request policy, then CloudFront overwrites the header values that were in the viewer request. Viewer-facing functions see the header value from the viewer request, while origin-facing functions see the header value that CloudFront added.
- If a viewer request function adds the `CloudFront-Viewer-Country` header, it fails validation and CloudFront returns HTTP status code 502 (Bad Gateway) to the viewer.

Restrictions on the request body with the include body option

When you choose the **Include Body** option to expose the request body to your Lambda@Edge function, the following information and size quotas apply to the portions of the body that are exposed or replaced.

- CloudFront always base64 encodes the request body before exposing it to Lambda@Edge.
- If the request body is large, CloudFront truncates it before exposing it to Lambda@Edge, as follows:
  - For viewer request events, the body is truncated at 40 KB.
  - For origin request events, the body is truncated at 1 MB.
- If you access the request body as read-only, CloudFront sends the full original request body to the origin.
- If your Lambda@Edge function replaces the request body, the following size quotas apply to the body that the function returns:
  - If the Lambda@Edge function returns the body as plain text:
    - For viewer request events, the body is truncated at 40 KB.
    - For origin request events, the body is truncated at 1 MB.

- If the Lambda@Edge function returns the body as base64 encoded text:
  - For viewer request events, the body is truncated at 53.2 KB.
  - For origin request events, the body is truncated at 1.33 MB.

Reports, metrics, and logs

CloudFront provides several options for reporting, monitoring, and logging your CloudFront resources:

- You can view and download reports to see usage and activity for your CloudFront distributions, including billing reports, cache statistics, popular content, and top referrers.
- You can monitor and track CloudFront, including your edge computing functions, directly in the CloudFront console or by using Amazon CloudWatch. CloudFront sends various metrics to CloudWatch for distributions and edge functions, both Lambda@Edge and CloudFront Functions.
- You can view logs for the viewer requests that your CloudFront distributions receive with standard logs or real-time logs. In addition to viewer request logs, you can use CloudWatch Logs to get logs for your edge functions, both Lambda@Edge and CloudFront Functions. You can also use Amazon CloudTrail to get logs of the CloudFront API activity in your Amazon Web Services account.
- You can track configuration changes to your CloudFront resources using Amazon Config.

For more information about each of these options, see the following topics.

Topics

- Amazon billing and usage reports for CloudFront
- CloudFront reports in the console
- Monitoring CloudFront metrics with Amazon CloudWatch
- CloudFront and edge function logging
- Tracking configuration changes with Amazon Config

Amazon billing and usage reports for CloudFront

Amazon provides two usage reports for CloudFront:

- The billing report is a high-level view of all of the activity for the Amazon services that you're using, including CloudFront. For more information, see the section called “Amazon billing report for CloudFront”.

- The usage report is a summary of activity for a specific service, aggregated by hour, day, or month. It also includes usage charts that provide a graphical representation of your CloudFront usage. For more information, see the section called "Amazon usage report for CloudFront".

To help you understand these reports, see the detailed information in the section called "Interpreting your Amazon bill and the Amazon usage report for CloudFront".

> **Note**
>
> Like other Amazon services, CloudFront charges you for only what you use. For more information, see CloudFront pricing.

Topics

- Amazon billing report for CloudFront
- Amazon usage report for CloudFront
- Interpreting your Amazon bill and the Amazon usage report for CloudFront

Amazon billing report for CloudFront

You can view a summary of your Amazon usage and charges, listed by service, on the bills page in the Amazon Web Services Management Console.

You can also download a more detailed version of the report in CSV format. The detailed billing report includes the following values that apply to CloudFront:

- **ProductCode** — `AmazonCloudFront`
- **UsageType** — One of the following values:
  - A code that identifies the type of data transfer
  - `Invalidations`
  - `SSL-Cert-Custom`

  For more information, see the section called "Interpreting your Amazon bill and the Amazon usage report for CloudFront".
- **ItemDescription** — A description of the billing rate for the **UsageType**.

- **Usage Start Date/Usage End Date** — The day that the usage applies to, in Coordinated Universal Time (UTC).
- **Usage Quantity** — One of the following values:
  - The number of requests during the specified time period
  - The amount of data transferred in gigabytes
  - The number of objects invalidated
  - The sum of the prorated months that you had SSL certificates associated with enabled CloudFront distributions. For example, if you have one certificate associated with an enabled distribution for an entire month and another certificate associated with an enabled distribution for half of the month, this value will be 1.5.

To display summary billing information and download the detailed billing report

1. Sign in to the Amazon Web Services Management Console at https://console.amazonaws.cn/console/home.
2. In the title bar, choose your user name, then choose **Billing Dashboard**.
3. In the navigation pane, choose **Bills**.
4. To view summary information for CloudFront, under **Details**, choose **CloudFront**.
5. To download a detailed billing report in CSV format, choose **Download CSV**, then follow the on-screen prompts to save the report.

Amazon usage report for CloudFront

Amazon provides a CloudFront usage report that is more detailed than the billing report but less detailed than CloudFront access logs. The usage report provides aggregate usage data by hour, day, or month, and it lists operations by region and usage type, such as data transferred out of the Australia region.

The CloudFront usage report includes the following values:

- **Service** — `AmazonCloudFront`
- **Operation** — HTTP method. Values include `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST`, and `PUT`.
- **UsageType** — One of the following values:
  - A code that identifies the type of data transfer

- `Invalidations`
- `SSL-Cert-Custom`

For more information, see the section called “Interpreting your Amazon bill and the Amazon usage report for CloudFront”.

- **Resource** — Either the ID of the CloudFront distribution associated with the usage or the certificate ID of an SSL certificate that you have associated with a CloudFront distribution.
- **StartTime/EndTime** — The day that the usage applies to, in Coordinated Universal Time (UTC).
- **UsageValue** — (1) The number of requests during the specified time period or (2) the amount of data transferred in bytes.

If you're using Amazon S3 as the origin for CloudFront, consider running the usage report for Amazon S3, too. However, if you use Amazon S3 for purposes other than as an origin for your CloudFront distributions, it might not be clear what portion applies to your CloudFront usage.

> **Tip**
>
> For detailed information about every request that CloudFront receives for your objects, turn on CloudFront access logs for your distribution. For more information, see the section called “Using standard logs (access logs)”.

Interpreting your Amazon bill and the Amazon usage report for CloudFront

Your Amazon bill for CloudFront includes codes and abbreviations that might not be immediately obvious. The first column in the following table lists items that appear in your bill and explains what each means.

In addition, you can get an Amazon usage report for CloudFront that contains more detail than the Amazon bill for CloudFront. The second column in the table lists items that appear in the usage report and shows the correlation between bill items and usage report items.

Most codes in both columns include a two-letter abbreviation that indicates the location of the activity. In the following table, *`region`* in a code is replaced in your Amazon bill and in the usage report by one of the following two-letter abbreviations:

- **AP:** Hong Kong, Philippines, South Korea, and Singapore (Asia Pacific)
- **AU:** Australia
- **CA:** Canada
- **EU:** Europe and Israel
- **IN:** India
- **JP:** Japan
- **ME:** Middle East
- **SA:** South America
- **US:** United States
- **ZA:** South Africa

For more information about pricing by region, see Amazon CloudFront Pricing.

> **Note**
>
> This table doesn't include charges for transferring your objects from an Amazon S3 bucket to CloudFront edge locations. These charges, if any, appear in the **Amazon Data Transfer** portion of your Amazon bill.

Items in your CloudFront bill	Values in the usage type column in the CloudFront usage report
region-**DataTransfer-Out-Bytes** Total bytes served from CloudFront edge locations in *region* in response to user GET and HEAD requests.	*region*-**Out-Bytes-HTTP-Static:** Bytes served via HTTP for objects with TTL ≥ 3,600 seconds. *region*-**Out-Bytes-HTTPS-Static:** Bytes served via HTTPS for objects with TTL ≥ 3,600 seconds. *region*-**Out-Bytes-HTTP-Dynamic:**

Items in your CloudFront bill	Values in the usage type column in the CloudFront usage report
	Bytes served via HTTP for objects with TTL < 3,600 seconds. *region*-**Out-Bytes-HTTPS-Dynamic:** Bytes served via HTTPS for objects with TTL < 3,600 seconds. *region*-**Out-Bytes-HTTP-Proxy:** Bytes returned from CloudFront to viewers via HTTP in response to DELETE, OPTIONS, PATCH, POST, and PUT requests. *region*-**Out-Bytes-HTTPS-Proxy:** Bytes returned from CloudFront to viewers via HTTPS in response to DELETE, OPTIONS, PATCH, POST, and PUT requests.
region-**DataTransfer-Out-OBytes** Total bytes transferred from CloudFront edge locations to your origin or edge function in response to DELETE, OPTIONS, PATCH, POST, and PUT requests. The charges include data transfer for WebSocket data from client to server.	*region*-**Out-OBytes-HTTP-Proxy** Total bytes transferred via HTTP from CloudFront edge locations to your origin or edge function in response to DELETE, OPTIONS, PATCH, POST, and PUT requests. *region*-**Out-OBytes-HTTPS-Proxy** Total bytes transferred via HTTPS from CloudFront edge locations to your origin or edge function in response to DELETE, OPTIONS, PATCH, POST, and PUT requests.

Items in your CloudFront bill	Values in the usage type column in the CloudFront usage report
***region*-Requests-Tier1** Number of HTTP GET and HEAD requests.	***region*-Requests-HTTP-Static** Number of HTTP GET and HEAD requests served for objects with TTL ≥ 3,600 seconds. ***region*-Requests-HTTP-Dynamic** Number of HTTP GET and HEAD requests served for objects with TTL < 3,600 seconds.
***region*-Requests-Tier2-HTTPS** Number of HTTPS GET and HEAD requests.	***region*-Requests-HTTPS-Static** Number of HTTPS GET and HEAD requests served for objects with TTL ≥ 3,600 seconds. ***region*-Requests-HTTPS-Dynamic** Number of HTTPS GET and HEAD requests served for objects with TTL < 3,600 seconds.
***region*-Requests-HTTP-Proxy** Number of HTTP DELETE, OPTIONS, PATCH, POST, and PUT requests that CloudFront forwards to your origin or edge function. Also includes the number of HTTP WebSocket requests (GET requests with the Upgrade: websocket header) that CloudFront forwards to your origin or edge function.	***region*-Requests-HTTP-Proxy** Same as the corresponding item in your CloudFront bill.

Items in your CloudFront bill	Values in the usage type column in the CloudFront usage report
***region*-Requests-HTTPS-Proxy** Number of HTTPS DELETE, OPTIONS, PATCH, POST, and PUT requests that CloudFront forwards to your origin or edge function. Also includes the number of HTTPS WebSocket requests (GET requests with the Upgrade: websocket header) that CloudFront forwards to your origin or edge function.	***region*-Requests-HTTPS-Proxy** Same as the corresponding item in your CloudFront bill.
***region*-Requests-HTTPS-Proxy-FLE** Number of HTTPS DELETE, OPTIONS, PATCH, and POST requests processed with field-level encryption that CloudFront forwards to your origin or edge function.	***region*-Requests-HTTPS-Proxy-FLE** Same as the corresponding item in your CloudFront bill.
***region*-Bytes-OriginShield** Total bytes transferred from the origin to any regional edge cache, including the regional edge cache that is enabled as Origin Shield.	***region*-Bytes-OriginShield** Total bytes transferred from the origin to any regional edge cache, including the regional edge cache that is enabled as Origin Shield.
***region*-OBytes-OriginShield** Total bytes transferred to the origin from any regional edge cache, including the regional edge cache that is enabled as Origin Shield.	***region*-OBytes-OriginShield** Total bytes transferred to the origin from any regional edge cache, including the regional edge cache that is enabled as Origin Shield.

Items in your CloudFront bill	Values in the usage type column in the CloudFront usage report
region-Requests-OriginShield Number of requests that go to Origin Shield as an incremental layer. For dynamic (non-cacheable) requests that are proxied to the origin, Origin Shield is always an incremental layer. For cacheable requests, Origin Shield is sometimes an incremental layer. For more information, see the section called "Estimating Origin Shield costs".	*region*-Requests-OriginShield Number of requests that go to Origin Shield as an incremental layer. For dynamic (non-cacheable) requests that are proxied to the origin, Origin Shield is always an incremental layer. For cacheable requests, Origin Shield is sometimes an incremental layer. For more information, see the section called "Estimating Origin Shield costs".
Invalidations The charge for invalidating objects (removing the objects from CloudFront edge locations); for more information, see Paying for file invalidation.	**Invalidations** Same as the corresponding item in your CloudFront bill.
SSL-Cert-Custom The charge for using an SSL certificate with a CloudFront alternate domain name such as example.com instead of using the default CloudFront SSL certificate and the domain name that CloudFront assigned to your distribution.	**SSL-Cert-Custom** Same as the corresponding item in your CloudFront bill.

CloudFront reports in the console

The CloudFront console includes a variety of reports about your CloudFront activity, including the following:

- CloudFront cache statistics reports
- CloudFront popular objects report

- CloudFront top referrers report
- CloudFront usage reports
- CloudFront viewers reports

Most of these reports are based on the data in CloudFront access logs, which contain detailed information about every user request that CloudFront receives. You don't need to enable access logs to view the reports. For more information, see Configuring and using standard logs (access logs). The CloudFront usage report is based on the Amazon usage report for CloudFront, which also doesn't require any special configuration. For more information, see Amazon usage report for CloudFront.

CloudFront cache statistics reports

The CloudFront cache statistics report includes the following information:

- **Total requests** – Shows the total number of requests for all HTTP status codes (for example, 200 or 404) and all methods (for example, GET, HEAD, or POST).
- **Percentage of viewer requests by result type** – Shows hits, misses, and errors as a percentage of total viewer requests for the selected CloudFront distribution.
- **Bytes transferred to viewers** – Shows total bytes and bytes from misses.
- **HTTP status codes** – Shows viewer requests by HTTP status code.
- **Percentage of GET requests that didn't finish downloading**– Shows viewer GET requests that didn't finish downloading the requested object as a percentage of total requests.

For more information, see CloudFront cache statistics reports.

CloudFront popular objects report

The CloudFront popular objects report lists the 50 most popular objects and statistics about those objects, including the number of requests for the object, the number of hits and misses, the hit ratio, the number of bytes served for misses, the total bytes served, the number of incomplete downloads, and the number of requests by HTTP status code (2xx, 3xx, 4xx, and 5xx).

For more information, see CloudFront popular objects report.

CloudFront top referrers report

The CloudFront top referrers report includes the top 25 referrers, the number of requests from a referrer, and the number of requests from a referrer as a percentage of the total number of requests during the specified period.

For more information, see CloudFront top referrers report.

CloudFront usage reports

The CloudFront usage reports include the following information:

- **Number of requests** – Shows the total number of requests that CloudFront responds to from edge locations in the selected region during each time interval for the specified CloudFront distribution.
- **Data transferred by protocol** and **data transferred by destination** – Both show the total amount of data transferred from CloudFront edge locations in the selected region during each time interval for the specified CloudFront distribution. They separate the data differently, as follows:
  - **By protocol** – Separates the data by protocol: HTTP or HTTPS.
  - **By destination** – Separates the data by destination: to your users or to your origin.

For more information, see CloudFront usage reports.

CloudFront viewers reports

The CloudFront viewers reports include the following information:

- **Devices** – Shows the types of devices (for example, Desktop or Mobile) that your users use to access your content
- **Browsers** – Shows the name (or the name and version) of the browsers that your users use most frequently to access your content, for example, Chrome or Firefox
- **Operating systems** – Shows the name (or the name and version) of the operating system that viewers run on most frequently when accessing your content, for example, Linux, macOS, or Windows
- **Locations** – Shows the locations, by country or by U.S. state/territory, of the viewers that access your content most frequently

For more information, see CloudFront viewers reports.

CloudFront cache statistics reports

You can use the Amazon CloudFront console to display a graphical representation of statistics related to CloudFront edge locations. Data for these statistics are drawn from the same source as CloudFront access logs. You can display charts for a specified date range in the last 60 days, with data points every hour or every day. You can usually view data about requests that CloudFront received as recently as an hour ago, but data can occasionally be delayed by as much as 24 hours.

> **Note**
>
> You don't need to enable access logging to view cache statistics.

To display CloudFront cache statistics

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, click **Cache Statistics**.
3. In the **CloudFront Cache Statistics Reports** pane, for **Start Date** and **End Date**, select the date range for which you want to display cache statistics charts. Available ranges depend on the value that you select for **Granularity**:

   - **Daily** – To display charts with one data point per day, select any date range in the previous 60 days.
   - **Hourly** – To display charts with one data point every hour, select any date range of up to 14 days within the previous 60 days.

   Dates and times are in Coordinated Universal Time (UTC).
4. For **Granularity**, specify whether to display one data point per day or one data point per hour in the charts. If you specify a date range greater than 14 days, the option to specify one data point per hour is not available.
5. For **Viewer Location**, choose the continent from which viewer requests originated, or choose **All Locations**. Cache statistics charts include data for requests that CloudFront received from the specified location.
6. In the **Distribution** list, select the distributions for which you want to display data in the usage charts:

- **An individual distribution** – The charts display data for the selected CloudFront distribution. The **Distribution** list displays the distribution ID and alternate domain names (CNAMEs) for the distribution, if any. If a distribution has no alternate domain names, the list includes origin domain names for the distribution.
- **All distributions** – The charts display summed data for all distributions that are associated with the current Amazon account, excluding distributions that you have deleted.

7. Click **Update**.
8. To view data for a daily or hourly data point within a chart, move your mouse pointer over the data point.
9. For charts that show data transferred, note that you can change the vertical scale to gigabytes, megabytes, or kilobytes for each chart.

Topics

- Downloading data in CSV format
- How cache statistics charts are related to data in the CloudFront standard logs (access logs)

Downloading data in CSV format

You can download the cache statistics report in CSV format. This section explains how to download the report and describes the values in the report.

To download the cache statistics report in CSV format

1. While viewing the Cache Statistics report, click **CSV**.
2. In the **Opening *file name*** dialog box, choose whether to open or save the file.

Information about the report

The first few rows of the report include the following information:

Version

The version of the format for this CSV file.

Report

The name of the report.

DistributionID

The ID of the distribution that you ran the report for, or ALL if you ran the report for all distributions.

StartDateUTC

The beginning of the date range for which you ran the report, in Coordinated Universal Time (UTC).

EndDateUTC

The end of the date range for which you ran the report, in Coordinated Universal Time (UTC).

GeneratedTimeUTC

The date and time on which you ran the report, in Coordinated Universal Time (UTC).

Granularity

Whether each row in the report represents one hour or one day.

ViewerLocation

The continent that viewer requests originated from, or ALL, if you chose to download the report for all locations.

Data in the cache statistics report

The report includes the following values:

DistributionID

The ID of the distribution that you ran the report for, or ALL if you ran the report for all distributions.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

ViewerLocation

The continent that viewer requests originated from, or ALL, if you chose to download the report for all locations.

TimeBucket

The hour or the day that data applies to, in Coordinated Universal Time (UTC).

RequestCount

The total number of requests for all HTTP status codes (for example, 200 or 404) and all methods (for example, GET, HEAD, or POST).

HitCount

The number of viewer requests for which the object is served from a CloudFront edge cache.

MissCount

The number of viewer requests for which the object isn't currently in an edge cache, so CloudFront must get the object from your origin.

ErrorCount

The number of viewer requests that resulted in an error, so CloudFront didn't serve the object.

IncompleteDownloadCount

The number of viewer requests for which the viewer started but didn't finish downloading the object.

HTTP2xx

The number of viewer requests for which the HTTP status code was a 2xx value (succeeded).

HTTP3xx

The number of viewer requests for which the HTTP status code was a 3xx value (additional action is required).

HTTP4xx

The number of viewer requests for which the HTTP status code was a 4xx value (client error).

HTTP5xx

The number of viewer requests for which the HTTP status code was a 5xx value (server error).

TotalBytes

The total number of bytes served to viewers by CloudFront in response to all requests for all HTTP methods.

BytesFromMisses

The number of bytes served to viewers for objects that were not in the edge cache at the time of the request. This value is a good approximation of bytes transferred from your origin to CloudFront edge caches. However, it excludes requests for objects that are already in the edge cache but that have expired.

How cache statistics charts are related to data in the CloudFront standard logs (access logs)

The following table shows how cache statistics charts in the CloudFront console correspond with values in CloudFront access logs. For more information about CloudFront access logs, see Configuring and using standard logs (access logs).

Total requests

This chart shows the total number of requests for all HTTP status codes (for example, 200 or 404) and all methods (for example, `GET`, `HEAD`, or `POST`). Total requests shown in this chart equal the total number of requests in the access log files for the same time period.

Percentage of viewer requests by result type

This chart shows hits, misses, and errors as a percentage of total viewer requests for the selected CloudFront distribution:

- **Hit** – A viewer request for which the object is served from a CloudFront edge cache. In access logs, these are requests for which the value of `x-edge-response-result-type` is `Hit`.
- **Miss** – A viewer request for which the object isn't currently in an edge cache, so CloudFront must get the object from your origin. In access logs, these are requests for which the value of `x-edge-response-result-type` is `Miss`.
- **Error** – A viewer request that resulted in an error, so CloudFront didn't serve the object. In access logs, these are requests for which the value of `x-edge-response-result-type` is `Error`, `LimitExceeded`, or `CapacityExceeded`.

The chart does not include refresh hits—requests for objects that are in the edge cache but that have expired. In access logs, refresh hits are requests for which the value of `x-edge-response-result-type` is `RefreshHit`.

Bytes transferred to viewers

This chart shows two values:

- **Total bytes** – The total number of bytes served to viewers by CloudFront in response to all requests for all HTTP methods. In CloudFront access logs, **Total Bytes** is the sum of the values in the `sc-bytes` column for all of the requests during the same time period.
- **Bytes from misses** – The number of bytes served to viewers for objects that were not in the edge cache at the time of the request. In CloudFront access logs, **bytes from misses** is the sum of the values in the `sc-bytes` column for requests for which the value of `x-edge-result-type` is `Miss`. This value is a good approximation of bytes transferred from your origin to CloudFront edge caches. However, it excludes requests for objects that are already in the edge cache but that have expired.

HTTP status codes

This chart shows viewer requests by HTTP status code. In CloudFront access logs, status codes appear in the `sc-status` column:

- **2xx** – The request succeeded.
- **3xx** – Additional action is required. For example, 301 (Moved Permanently) means that the requested object has moved to a different location.
- **4xx** – The client apparently made an error. For example, 404 (Not Found) means that the client requested an object that could not be found.
- **5xx** – The origin server didn't fill the request. For example, 503 (Service Unavailable) means that the origin server is currently unavailable.

Percentage of GET requests that didn't finish downloading

This chart shows viewer `GET` requests that didn't finish downloading the requested object as a percentage of total requests. Typically, downloading an object doesn't complete because the viewer canceled the download, for example, by clicking a different link or by closing the browser. In CloudFront access logs, these requests have a value of `200` in the `sc-status` column and a value of `Error` in the `x-edge-result-type` column.

CloudFront popular objects report

The Amazon CloudFront console can display a list of the 50 most popular objects for a distribution during a specified date range in the previous 60 days.

Data for the popular objects report is drawn from the same source as CloudFront access logs. To get an accurate count of the top 50 objects, CloudFront counts the requests for all of your objects

in 10-minute intervals beginning at midnight and keeps a running total of the top 150 objects for the next 24 hours. (CloudFront also retains daily totals for the top 150 objects for 60 days.) Near the bottom of the list, objects constantly rise onto or drop off of the list, so the totals for those objects are approximations. The fifty objects at the top of the list of 150 objects may rise and fall within the list, but they rarely drop off of the list altogether, so the totals for those objects typically are more reliable.

When an object drops off of the list of the top 150 objects and then rises onto the list again over the course of a day, CloudFront adds an estimated number of requests for the period that the object was missing from the list. The estimate is based on the number of requests received by whichever object was at the bottom of the list during that time period. If the object rises into the top 50 objects later in the day, the estimates of the number of requests that CloudFront received while the object was out of the top 150 objects usually causes the number of requests in the popular objects report to exceed the number of requests that appear in the access logs for that object.

> **Note**
>
> You don't need to enable access logging to view a list of popular objects.

To display popular objects for a distribution

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, click **Popular Objects**.
3. In the **CloudFront Popular Objects Report** pane, for **Start Date** and **End Date**, select the date range for which you want to display a list of popular objects. You can choose any date range in the previous 60 days.

   Dates and times are in Coordinated Universal Time (UTC).
4. In the **Distribution** list, select the distribution for which you want to display a list of popular objects.
5. Click **Update**.

Topics

- Downloading data in CSV format

- How data in the popular objects report is related to data in the CloudFront standard logs (access logs)

Downloading data in CSV format

You can download the popular objects report in CSV format. This section explains how to download the report and describes the values in the report.

To download the popular objects report in CSV format

1. While viewing the popular objects report, click **CSV**.
2. In the **Opening *file name*** dialog box, choose whether to open or save the file.

Information about the report

The first few rows of the report include the following information:

Version

The version of the format for this CSV file.

Report

The name of the report.

DistributionID

The ID of the distribution that you ran the report for.

StartDateUTC

The beginning of the date range for which you ran the report, in Coordinated Universal Time (UTC).

EndDateUTC

The end of the date range for which you ran the report, in Coordinated Universal Time (UTC).

GeneratedTimeUTC

The date and time on which you ran the report, in Coordinated Universal Time (UTC).

Data in the popular objects report

The report includes the following values:

DistributionID

The ID of the distribution that you ran the report for.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

Object

The last 500 characters of the URL for the object.

RequestCount

The total number of requests for this object.

HitCount

The number of viewer requests for which the object is served from a CloudFront edge cache.

MissCount

The number of viewer requests for which the object isn't currently in an edge cache, so CloudFront must get the object from your origin.

HitCountPct

The value of `HitCount` as a percentage of the value of `RequestCount`.

BytesFromMisses

The number of bytes served to viewers for this object when the object was not in the edge cache at the time of the request.

TotalBytes

The total number of bytes served to viewers by CloudFront for this object in response to all requests for all HTTP methods.

IncompleteDownloadCount

The number of viewer requests for this object for which the viewer started but didn't finish downloading the object.

HTTP2xx

The number of viewer requests for which the HTTP status code was a 2xx value (succeeded).

HTTP3xx

The number of viewer requests for which the HTTP status code was a 3xx value (additional action is required).

HTTP4xx

The number of viewer requests for which the HTTP status code was a 4xx value (client error).

HTTP5xx

The number of viewer requests for which the HTTP status code was a 5xx value (server error).

How data in the popular objects report is related to data in the CloudFront standard logs (access logs)

The following list shows how values in the popular objects report in the CloudFront console correspond with values in CloudFront access logs. For more information about CloudFront access logs, see Configuring and using standard logs (access logs).

URL

The last 500 characters of the URL that viewers use to access the object.

Requests

The total number of requests for the object. This value generally corresponds closely with the number of `GET` requests for the object in CloudFront access logs.

Hits

The number of viewer requests for which the object was served from a CloudFront edge cache. In access logs, these are requests for which the value of `x-edge-response-result-type` is `Hit`.

Misses

The number of viewer requests for which the object wasn't in an edge cache, so CloudFront retrieved the object from your origin. In access logs, these are requests for which the value of `x-edge-response-result-type` is `Miss`.

Hit ratio

The value of the **Hits** column as a percentage of the value of the **Requests** column.

Bytes from misses

The number of bytes served to viewers for objects that were not in the edge cache at the time of the request. In CloudFront access logs, **bytes from misses** is the sum of the values in the `sc-bytes` column for requests for which the value of `x-edge-result-type` is `Miss`.

Total bytes

The total number of bytes that CloudFront served to viewers in response to all requests for the object for all HTTP methods. In CloudFront access logs, **total bytes** is the sum of the values in the `sc-bytes` column for all of the requests during the same time period.

Incomplete downloads

The number of viewer requests that did not finish downloading the requested object. Typically, the reason that a download doesn't complete is that the viewer canceled it, for example, by clicking a different link or by closing the browser. In CloudFront access logs, these requests have a value of `200` in the `sc-status` column and a value of `Error` in the `x-edge-result-type` column.

2xx

The number of requests for which the HTTP status code is `2xx`, `Successful`. In CloudFront access logs, status codes appear in the `sc-status` column.

3xx

The number of requests for which the HTTP status code is `3xx`, `Redirection`. `3xx` status codes indicate that additional action is required. For example, 301 (Moved Permanently) means that the requested object has moved to a different location.

4xx

The number of requests for which the HTTP status code is `4xx`, `Client Error`. `4xx` status codes indicate that the client apparently made an error. For example, 404 (Not Found) means that the client requested an object that could not be found.

5xx

The number of requests for which the HTTP status code is `5xx`, `Server Error`. `5xx` status codes indicate that the origin server didn't fill the request. For example, 503 (Service Unavailable) means that the origin server is currently unavailable.

CloudFront top referrers report

The CloudFront console can display a list of the 25 domains of the websites that originated the most HTTP and HTTPS requests for objects that CloudFront is distributing for a specified distribution. These top referrers can be search engines, other websites that link directly to your objects, or your own website. For example, if https://example.com/index.html links to 10 graphics, example.com is the referrer for all 10 graphics. You can display the top referrers report for any date range in the previous 60 days.

> **Note**
>
> If a user enters a URL directly into the address line of a browser, there is no referrer for the requested object.

Data for the top referrers report is drawn from the same source as CloudFront access logs. To get an accurate count of the top 25 referrers, CloudFront counts the requests for all of your objects in 10-minute intervals and keeps a running total of the top 75 referrers. Near the bottom of the list, referrers constantly rise onto or drop off of the list, so the totals for those referrers are approximations. The 25 referrers at the top of the list of 75 referrers may rise and fall within the list, but they rarely drop off of the list altogether, so the totals for those referrers typically are more reliable.

> **Note**
>
> You don't need to enable access logging to view a list of top referrers.

To display top referrers for a distribution

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, click **Top Referrers**.
3. In the **CloudFront Top Referrers Report** pane, for **Start Date** and **End Date**, select the date range for which you want to display a list of top referrers.

   Dates and times are in Coordinated Universal Time (UTC).

4. In the **Distribution** list, select the distribution for which you want to display a list of top referrers.
5. Click **Update**.

Topics

- Downloading data in CSV format
- How data in the top referrers report is related to data in the CloudFront standard logs (access logs)

Downloading data in CSV format

You can download the top referrers report in CSV format. This section explains how to download the report and describes the values in the report.

To download the top referrers report in CSV format

1. While viewing the Top Referrers report, click **CSV**.
2. In the **Opening *file name*** dialog box, choose whether to open or save the file.

Information about the report

The first few rows of the report include the following information:

Version

The version of the format for this CSV file.

Report

The name of the report.

DistributionID

The ID of the distribution that you ran the report for, or ALL if you ran the report for all distributions.

StartDateUTC

The beginning of the date range for which you ran the report, in Coordinated Universal Time (UTC).

EndDateUTC

The end of the date range for which you ran the report, in Coordinated Universal Time (UTC).

GeneratedTimeUTC

The date and time on which you ran the report, in Coordinated Universal Time (UTC).

Data in the top referrers report

The report includes the following values:

DistributionID

The ID of the distribution that you ran the report for, or `ALL` if you ran the report for all distributions.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

Referrer

The domain name of the referrer.

RequestCount

The total number of requests from the domain name in the `Referrer` column.

RequestsPct

The number of requests submitted by the referrer as a percentage of the total number of requests during the specified period.

How data in the top referrers report is related to data in the CloudFront standard logs (access logs)

The following list shows how values in the Top Referrers report in the CloudFront console correspond with values in CloudFront access logs. For more information about CloudFront access logs, see Configuring and using standard logs (access logs).

Referrer

The domain name of the referrer. In access logs, referrers are listed in the `cs(Referer)` column.

Request count

The total number of requests from the domain name in the **Referrer** column. This value generally corresponds closely with the number of `GET` requests from the referrer in CloudFront access logs.

Request %

The number of requests submitted by the referrer as a percentage of the total number of requests during the specified period. If you have more than 25 referrers, then you can't calculate **Request %** based on the data in this table because the **request count** column doesn't include all of the requests during the specified period.

CloudFront usage reports

The Amazon CloudFront console can display a graphical representation of your CloudFront usage that is based on a subset of the usage report data. You can display charts for a specified date range in the last 60 days, with data points every hour or every day. You can usually view data about requests that CloudFront received as recently as four hours ago, but data can occasionally be delayed by as much as 24 hours.

For more information, see How the usage charts are related to data in the CloudFront usage report.

To display CloudFront usage charts

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In **navigation** pane, click **Usage Reports**.
3. In the **CloudFront Usage Reports** pane, for **Start Date** and **End Date**, select the date range for which you want to display usage charts. Available ranges depend on the value that you select for **Granularity**:

   - **Daily** — To display charts with one data point per day, select any date range in the previous 60 days.

- **Hourly** — To display charts with one data point every hour, select any date range of up to 14 days within the previous 60 days.

Dates and times are in Coordinated Universal Time (UTC).

4. For **Granularity**, specify whether to display one data point per day or one data point per hour in the charts. If you specify a date range greater than 14 days, the option to specify one data point per hour is not available.
5. For **Billing Region**, choose the CloudFront billing region that has the data you want to view, or choose **All Regions**. Usage charts include data for requests that CloudFront processes in edge locations in the specified region. The region where CloudFront processes requests might or might not correspond with the location of your users.

   Select only regions that are included in the price class for your distribution; otherwise, the usage charts probably won't contain any data. For example, if you chose Price Class 200 for your distribution, the South America and Australia billing regions are not included, so CloudFront generally won't process your requests from those regions. For more information about price classes, see CloudFront pricing.
6. In the **Distribution** list, select the distributions for which you want to display data in the usage charts:

   - **An individual distribution** — The charts display data for the selected CloudFront distribution. The **Distribution** list displays the distribution ID and alternate domain names (CNAMEs) for the distribution, if any. If a distribution has no alternate domain names, the list includes origin domain names for the distribution.
   - **All distributions (excludes deleted)** — The charts display summed data for all distributions that are associated with the current Amazon account, excluding distributions that you have deleted.
   - **All Deleted Distributions** — The charts display summed data for all distributions that are associated with the current Amazon account and that were deleted in the last 60 days.
7. Click **Update Graphs**.
8. To view data for a daily or hourly data point within a chart, move your mouse pointer over the data point.
9. For charts that show data transferred, note that you can change the vertical scale to gigabytes, megabytes, or kilobytes for each chart.

Topics

- Downloading data in CSV format
- How the usage charts are related to data in the CloudFront usage report

Downloading data in CSV format

You can download the Usage report in CSV format. This section explains how to download the report and describes the values in the report.

To download the usage report in CSV format

1. While viewing the Usage report, click **CSV**.
2. In the **Opening *file name*** dialog box, choose whether to open or save the file.

Information about the report

The first few rows of the report include the following information:

Version

The version of the format for this CSV file.

Report

The name of the report.

DistributionID

The ID of the distribution that you ran the report for, `ALL` if you ran the report for all distributions, or `ALL_DELETED` if you ran the report for all deleted distributions.

StartDateUTC

The beginning of the date range for which you ran the report, in Coordinated Universal Time (UTC).

EndDateUTC

The end of the date range for which you ran the report, in Coordinated Universal Time (UTC).

GeneratedTimeUTC

The date and time on which you ran the report, in Coordinated Universal Time (UTC).

Granularity

Whether each row in the report represents one hour or one day.

BillingRegion

The continent that viewer requests originated from, or ALL, if you chose to download the report for all billing regions.

Data in the usage report

The report includes the following values:

DistributionID

The ID of the distribution that you ran the report for, ALL if you ran the report for all distributions, or ALL_DELETED if you ran the report for all deleted distributions.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

BillingRegion

The CloudFront billing region that you ran the report for, or ALL.

TimeBucket

The hour or the day that data applies to, in Coordinated Universal Time (UTC).

HTTP

The number of HTTP requests that CloudFront responded to from edge locations in the selected region during each time interval for the specified CloudFront distribution. Values include:

- The number of GET and HEAD requests, which cause CloudFront to transfer data to your users
- The number of DELETE, OPTIONS, PATCH, POST, and PUT requests, which cause CloudFront to transfer data to your origin

HTTPS

The number of HTTPS requests that CloudFront responded to from edge locations in the selected region during each time interval for the specified CloudFront distribution. Values include:

- The number of GET and HEAD requests, which cause CloudFront to transfer data to your users
- The number of DELETE, OPTIONS, PATCH, POST, and PUT requests, which cause CloudFront to transfer data to your origin

HTTPBytes

The total amount of data transferred over HTTP from CloudFront edge locations in the selected billing region during the time period for the specified CloudFront distribution. Values include:

- Data transferred from CloudFront to your users in response to GET and HEAD requests
- Data transferred from CloudFront to your origin for DELETE, OPTIONS, PATCH, POST, and PUT requests
- Data transferred from CloudFront to your users in response to DELETE, OPTIONS, PATCH, POST, and PUT requests

HTTPSBytes

The total amount of data transferred over HTTPS from CloudFront edge locations in the selected billing region during the time period for the specified CloudFront distribution. Values include:

- Data transferred from CloudFront to your users in response to GET and HEAD requests
- Data transferred from CloudFront to your origin for DELETE, OPTIONS, PATCH, POST, and PUT requests
- Data transferred from CloudFront to your users in response to DELETE, OPTIONS, PATCH, POST, and PUT requests

BytesIn

The total amount of data transferred from CloudFront to your origin for DELETE, OPTIONS, PATCH, POST, and PUT requests in the selected region during each time interval for the specified CloudFront distribution.

BytesOut

The total amount of data transferred over HTTP and HTTPS from CloudFront to your users in the selected region during each time interval for the specified CloudFront distribution. Values include:

- Data transferred from CloudFront to your users in response to GET and HEAD requests
- Data transferred from CloudFront to your users in response to DELETE, OPTIONS, PATCH, POST, and PUT requests

How the usage charts are related to data in the CloudFront usage report

The following list shows how the usage charts in the CloudFront console correspond with values in the **Usage Type** column in the CloudFront usage report.

Topics

- Number of requests
- Data transferred by protocol
- Data transferred by destination

Number of requests

This chart shows the total number of requests that CloudFront responds to from edge locations in the selected region during each time interval for the specified CloudFront distribution, separated by protocol (HTTP or HTTPS) and type (static, dynamic, or proxy).

Number of HTTP requests

- *region*-**Requests-HTTP-Static:** Number of HTTP `GET` and `HEAD` requests served for objects with TTL ≥ 3600 seconds
- *region*-**Requests-HTTP-Dynamic:** Number of HTTP `GET` and `HEAD` requests served for objects with TTL < 3600 seconds
- *region*-**Requests-HTTP-Proxy:** Number of HTTP `DELETE`, `OPTIONS`, `PATCH`, `POST`, and `PUT` requests that CloudFront forwards to your origin

Number of HTTPS requests

- *region*-**Requests-HTTPS-Static:** Number of HTTPS `GET` and `HEAD` requests served for objects with TTL ≥ 3600 seconds
- *region*-**Requests-HTTPS-Dynamic:** Number of HTTPS `GET` and `HEAD` requests served for objects with TTL < 3600 seconds
- *region*-**Requests-HTTPS-Proxy:** Number of HTTPS `DELETE`, `OPTIONS`, `PATCH`, `POST`, and `PUT` requests that CloudFront forwards to your origin

Data transferred by protocol

This chart shows the total amount of data transferred from CloudFront edge locations in the selected region during each time interval for the specified CloudFront distribution, separated by protocol (HTTP or HTTPS), type (static, dynamic, or proxy), and destination (users or origin).

Data transferred over HTTP

- *region*-**Out-Bytes-HTTP-Static:** Bytes served via HTTP for objects with TTL ≥ 3600 seconds
- *region*-**Out-Bytes-HTTP-Dynamic:** Bytes served via HTTP for objects with TTL < 3600 seconds
- *region*-**Out-Bytes-HTTP-Proxy:** Bytes returned from CloudFront to viewers via HTTP in response to DELETE, OPTIONS, PATCH, POST, and PUT requests
- *region*-**Out-OBytes-HTTP-Proxy:** Total bytes transferred via HTTP from CloudFront edge locations to your origin in response to DELETE, OPTIONS, PATCH, POST, and PUT requests

Data transferred over HTTPS

- *region*-**Out-Bytes-HTTPS-Static:** Bytes served via HTTPS for objects with TTL ≥ 3600 seconds
- *region*-**Out-Bytes-HTTPS-Dynamic:** Bytes served via HTTPS for objects with TTL < 3600 seconds
- *region*-**Out-Bytes-HTTPS-Proxy:** Bytes returned from CloudFront to viewers via HTTPS in response to DELETE, OPTIONS, PATCH, POST, and PUT requests
- *region*-**Out-OBytes-HTTPS-Proxy:** Total bytes transferred via HTTPS from CloudFront edge locations to your origin in response to DELETE, OPTIONS, PATCH, POST, and PUT requests

Data transferred by destination

This chart shows the total amount of data transferred from CloudFront edge locations in the selected region during each time interval for the specified CloudFront distribution, separated by destination (users or origin), protocol (HTTP or HTTPS), and type (static, dynamic, or proxy).

Data transferred from CloudFront to your users

- *region*-**Out-Bytes-HTTP-Static:** Bytes served via HTTP for objects with TTL ≥ 3600 seconds
- *region*-**Out-Bytes-HTTPS-Static:** Bytes served via HTTPS for objects with TTL ≥ 3600 seconds
- *region*-**Out-Bytes-HTTP-Dynamic:** Bytes served via HTTP for objects with TTL < 3600 seconds
- *region*-**Out-Bytes-HTTPS-Dynamic:** Bytes served via HTTPS for objects with TTL < 3600 seconds
- *region*-**Out-Bytes-HTTP-Proxy:** Bytes returned from CloudFront to viewers via HTTP in response to DELETE, OPTIONS, PATCH, POST, and PUT requests

- *region*-**Out-Bytes-HTTPS-Proxy:** Bytes returned from CloudFront to viewers via HTTPS in response to DELETE, OPTIONS, PATCH, POST, and PUT requests

Data transferred from CloudFront to your origin

- *region*-**Out-OBytes-HTTP-Proxy:** Total bytes transferred via HTTP from CloudFront edge locations to your origin in response to DELETE, OPTIONS, PATCH, POST, and PUT requests
- *region*-**Out-OBytes-HTTPS-Proxy:** Total bytes transferred via HTTPS from CloudFront edge locations to your origin in response to DELETE, OPTIONS, PATCH, POST, and PUT requests

CloudFront viewers reports

The CloudFront console can display four reports about the physical devices (desktop computers, mobile devices) and about the viewers (typically web browsers) that are accessing your content:

- **Devices** – The type of the devices that your users use most frequently to access your content, for example, Desktop or Mobile.
- **Browsers** – The name (or the name and version) of the browsers that your users use most frequently to access your content, for example, Chrome or Firefox. The report lists the top 10 browsers.
- **Operating systems** – The name (or the name and version) of the operating system that viewers run on most frequently when accessing your content, for example, Linux, macOS, or Windows. The report lists the top 10 operating systems.
- **Locations** – The locations, by country or by U.S. state/territory, of the viewers that access your content most frequently. The report lists the top 50 countries or U.S. states/territories.

You can display all four Viewers reports for any date range in the previous 60 days. For the Locations report, you can also display the report with data points every hour for any date range of up to 14 days in the previous 60 days.

> **Note**
>
> You don't need to enable access logging to view Viewers charts and reports.

Topics

- Displaying viewers charts and reports

- Downloading data in CSV format
- How data in the locations report is related to data in the CloudFront standard logs (access logs)

Displaying viewers charts and reports

To display CloudFront viewers charts and reports, perform the following procedure.

To display CloudFront viewers charts and reports

1. Sign in to the Amazon Web Services Management Console and open the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home.
2. In the navigation pane, click **Viewers**.
3. In the **CloudFront Viewers** pane, for **Start Date** and **End Date**, select the date range for which you want to display viewer charts and reports.

   For the Locations chart, available ranges depend on the value that you select for **Granularity**:

   - **Daily** – To display charts with one data point per day, select any date range in the previous 60 days.
   - **Hourly** – To display charts with one data point every hour, select any date range of up to 14 days within the previous 60 days.

   Dates and times are in Coordinated Universal Time (UTC).
4. (Browsers and Operating Systems charts only) For **Grouping**, specify whether you want to group browsers and operating systems by name (Chrome, Firefox) or by name and version (Chrome 40.0, Firefox 35.0).
5. (Locations chart only) For **Granularity**, specify whether to display one data point per day or one data point per hour in the charts. If you specify a date range greater than 14 days, the option to specify one data point per hour is not available.
6. (Locations chart only) For **Details**, specify whether to display the top locations by countries or by U.S. states.
7. In the **Distribution** list, select the distribution for which you want to display data in the usage charts:

   - **An individual distribution** – The charts display data for the selected CloudFront distribution. The **Distribution** list displays the distribution ID and an alternate domain name

(CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

- **All distributions (excludes deleted)** – The charts display summed data for all distributions that are associated with the current Amazon account, excluding distributions that you have deleted.

8. Click **Update**.
9. To view data for a daily or hourly data point within a chart, move your mouse pointer over the data point.

Downloading data in CSV format

You can download each of the viewer reports in CSV format. This section explains how to download the reports and describes the values in the report.

To download the viewer reports in CSV format

1. While viewing the Viewer report, click **CSV**.
2. Choose the data that you want to download, for example, **Devices** or **Devices Trends**.
3. In the **Opening *file name*** dialog box, choose whether to open or save the file.

Topics

- Information about the reports
- Devices report
- Device trends report
- Browsers report
- Browser trends report
- Operating systems report
- Operating system trends report
- Locations report
- Location trends report

Information about the reports

The first few rows of each report includes the following information:

Version

The version of the format for this CSV file.

Report

The name of the report.

DistributionID

The ID of the distribution that you ran the report for, or ALL if you ran the report for all distributions.

StartDateUTC

The beginning of the date range for which you ran the report, in Coordinated Universal Time (UTC).

EndDateUTC

The end of the date range for which you ran the report, in Coordinated Universal Time (UTC).

GeneratedTimeUTC

The date and time on which you ran the report, in Coordinated Universal Time (UTC).

Grouping (browsers and operating systems reports only)

Whether the data is grouped by the name or by the name and version of the browser or operating system.

Granularity

Whether each row in the report represents one hour or one day.

Details (locations report only)

Whether requests are listed by country or by U.S. state.

Devices report

The report includes the following values:

DistributionID

The ID of the distribution that you ran the report for, or ALL if you ran the report for all distributions.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

Requests

The number of requests that CloudFront received from each type of device.

RequestsPct

The number of requests that CloudFront received from each type of device as a percentage of the total number of requests that CloudFront received from all devices.

Device trends report

The report includes the following values:

DistributionID

The ID of the distribution that you ran the report for, or `ALL` if you ran the report for all distributions.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

TimeBucket

The hour or the day that the data applies to, in Coordinated Universal Time (UTC).

Desktop

The number of requests that CloudFront received from desktop computers during the period.

Mobile

The number of requests that CloudFront received from mobile devices during the period. Mobile devices can include both tablets and mobile phones. If CloudFront can't determine whether a request originated from a mobile device or a tablet, it's counted in the `Mobile` column.

Smart-TV

The number of requests that CloudFront received from smart TVs during the period.

Tablet

The number of requests that CloudFront received from tablets during the period. If CloudFront can't determine whether a request originated from a mobile device or a tablet, it's counted in the `Mobile` column.

Unknown

Requests for which the value of the `User-Agent` HTTP header was not associated with one of the standard device types, for example, `Desktop` or `Mobile`.

Empty

The number of requests that CloudFront received that didn't include a value in the HTTP `User-Agent` header during the period.

Browsers report

The report includes the following values:

DistributionID

The ID of the distribution that you ran the report for, or `ALL` if you ran the report for all distributions.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

Group

The browser or the browser and version that CloudFront received requests from, depending on the value of `Grouping`. In addition to browser names, possible values include the following:

- **Bot/Crawler** – primarily requests from search engines that are indexing your content.
- **Empty** – requests for which the value of the `User-Agent` HTTP header was empty.
- **Other** – browsers that CloudFront identified but that aren't among the most popular. If `Bot/Crawler`, `Empty`, and/or `Unknown` don't appear among the first nine values, then they're also included in `Other`.
- **Unknown** – requests for which the value of the `User-Agent` HTTP header was not associated with a standard browser. Most requests in this category come from custom applications or scripts.

Requests

The number of requests that CloudFront received from each type of browser.

RequestsPct

The number of requests that CloudFront received from each type of browser as a percentage of the total number of requests that CloudFront received during the time period.

Browser trends report

The report includes the following values:

DistributionID

The ID of the distribution that you ran the report for, or `ALL` if you ran the report for all distributions.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

TimeBucket

The hour or the day that the data applies to, in Coordinated Universal Time (UTC).

(Browsers)

The remaining columns in the report list the browsers or the browsers and their versions, depending on the value of `Grouping`. In addition to browser names, possible values include the following:

- **Bot/Crawler** – primarily requests from search engines that are indexing your content.
- **Empty** – requests for which the value of the `User-Agent` HTTP header was empty.
- **Other** – browsers that CloudFront identified but that aren't among the most popular. If `Bot/Crawler`, `Empty`, and/or `Unknown` don't appear among the first nine values, then they're also included in `Other`.
- **Unknown** – requests for which the value of the `User-Agent` HTTP header was not associated with a standard browser. Most requests in this category come from custom applications or scripts.

Operating systems report

The report includes the following values:

DistributionID

The ID of the distribution that you ran the report for, or `ALL` if you ran the report for all distributions.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

Group

The operating system or the operating system and version that CloudFront received requests from, depending on the value of `Grouping`. In addition to operating system names, possible values include the following:

- **Bot/Crawler** – primarily requests from search engines that are indexing your content.
- **Empty** – requests for which the value of the `User-Agent` HTTP header was empty.
- **Other** – operating systems that CloudFront identified but that aren't among the most popular. If `Bot/Crawler`, `Empty`, and/or `Unknown` don't appear among the first nine values, then they're also included in `Other`.
- **Unknown** – requests for which the value of the `User-Agent` HTTP header was not associated with a standard browser. Most requests in this category come from custom applications or scripts.

Requests

The number of requests that CloudFront received from each type of operating system.

RequestsPct

The number of requests that CloudFront received from each type of operating system as a percentage of the total number of requests that CloudFront received during the time period.

Operating system trends report

The report includes the following values:

DistributionID

The ID of the distribution that you ran the report for, or `ALL` if you ran the report for all distributions.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

TimeBucket

The hour or the day that the data applies to, in Coordinated Universal Time (UTC).

(Operating systems)

The remaining columns in the report list the operating systems or the operating systems and their versions, depending on the value of `Grouping`. In addition to operating system names, possible values include the following:

- **Bot/Crawler** – primarily requests from search engines that are indexing your content.
- **Empty** – requests for which the value of the `User-Agent` HTTP header was empty.
- **Other** – operating systems that CloudFront identified but that aren't among the most popular. If `Bot/Crawler`, `Empty`, and/or `Unknown` don't appear among the first nine values, then they're also included in `Other`.
- **Unknown** – requests for which the operating system isn't specified in the `User-Agent` HTTP header.

Locations report

The report includes the following values:

DistributionID

The ID of the distribution that you ran the report for, or `ALL` if you ran the report for all distributions.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

LocationCode

The abbreviation for the location that CloudFront received requests from. For more information about possible values, see the description of Location in How data in the locations report is related to data in the CloudFront standard logs (access logs).

LocationName

The name of the location that CloudFront received requests from.

Requests

The number of requests that CloudFront received from each location.

RequestsPct

The number of requests that CloudFront received from each location as a percentage of the total number of requests that CloudFront received from all locations during the time period.

TotalBytes

The number of bytes that CloudFront served to viewers in this country or state, for the specified distribution and period.

Location trends report

The report includes the following values:

DistributionID

The ID of the distribution that you ran the report for, or ALL if you ran the report for all distributions.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

TimeBucket

The hour or the day that the data applies to, in Coordinated Universal Time (UTC).

(Locations)

The remaining columns in the report list the locations that CloudFront received requests from. For more information about possible values, see the description of Location in How data in the locations report is related to data in the CloudFront standard logs (access logs).

How data in the locations report is related to data in the CloudFront standard logs (access logs)

The following list shows how data in the Locations report in the CloudFront console corresponds with values in CloudFront access logs. For more information about CloudFront access logs, see Configuring and using standard logs (access logs).

Location

The country or U.S. state that the viewer is in. In access logs, the `c-ip` column contains the IP address of the device that the viewer is running on. We use geolocation data to identify the geographic location of the device based on the IP address.

If you're displaying the **Locations** report by country, note that the country list is based on ISO 3166-2, *Codes for the representation of names of countries and their subdivisions – Part 2: Country subdivision code*. The country list includes the following additional values:

- **Anonymous Proxy** – The request originated from an anonymous proxy.
- **Satellite Provider** – The request originated from a satellite provider that provides internet service to multiple countries. Users might be in countries with a high risk of fraud.
- **Europe (Unknown)** – The request originated from an IP in a block that is used by multiple European countries. The country that the request originated from cannot be determined. CloudFront uses **Europe (Unknown)** as the default.
- **Asia/Pacific (Unknown)** – The request originated from an IP in a block that is used by multiple countries in the Asia/Pacific region. The country that the request originated from cannot be determined. CloudFront uses **Asia/Pacific (Unknown)** as the default.

If you display the **Locations** report by U.S. state, note that the report can include U.S. territories and U.S. Armed Forces regions.

> **Note**
>
> If CloudFront can't determine a user's location, the location will appear as Unknown in viewer reports.

Request Count

The total number of requests from the country or U.S. state that the viewer is in, for the specified distribution and period. This value generally corresponds closely with the number of GET requests from IP addresses in that country or state in CloudFront access logs.

Request %

One of the following, depending on the value that you selected for **Details**:

- **Countries** – The requests from this country as a percentage of the total number of requests.
- **U.S. States** – The requests from this state as a percentage of the total number of requests from the United States.

If requests came from more than 50 countries, then you can't calculate **Request %** based on the data in this table because the **Request Count** column doesn't include all of the requests during the specified period.

Bytes

The number of bytes that CloudFront served to viewers in this country or state, for the specified distribution and period. To change the display of data in this column to KB, MB, or GB, click the link in the column heading.

Monitoring CloudFront metrics with Amazon CloudWatch

Amazon CloudFront is integrated with Amazon CloudWatch and automatically publishes operational metrics for distributions and edge functions (both Lambda@Edge and CloudFront Functions). Many of these metrics are displayed in a set of graphs in the CloudFront console, and are also accessible by using the CloudFront API or CLI. All of these metrics are available in the CloudWatch console or through the CloudWatch API or CLI. The CloudFront metrics don't count against CloudWatch quotas (formerly known as limits) and don't incur any additional cost.

In addition to the default metrics for CloudFront distributions, you can turn on additional metrics for an additional cost. The additional metrics apply to CloudFront distributions, and must be turned on for each distribution separately. For more information about the cost, see the section called "Estimating cost for the additional CloudFront metrics".

Viewing these metrics can help you troubleshoot, track, and debug issues. To view these metrics in the CloudFront console, see the Monitoring page. To view graphs about the activity for a specific

CloudFront distribution or edge function, choose one, and then choose **View distribution metrics** or **View metrics**.

You can also set alarms based on these metrics in the CloudFront console, or in the CloudWatch console, API, or CLI (standard CloudWatch pricing applies). For example, you can set an alarm based on the `5xxErrorRate` metric, which represents the percentage of all viewer requests for which the response's HTTP status code is in the range of `500` to 599, inclusive. When the error rate reaches a certain value for a certain amount of time, for example, 5% of requests for 5 continuous minutes, the alarm is triggered. You specify the alarm's value and its time unit when you create the alarm. For more information, see Creating alarms.

> **Note**
>
> When you create a CloudWatch alarm in the CloudFront console, it creates one for you in the US East (N. Virginia) Region (`us-east-1`). If you create an alarm from the CloudWatch console, you must use the same Region. Because CloudFront is a global service, metrics for the service are sent to US East (N. Virginia).

Topics

- Viewing CloudFront and edge function metrics
- Creating alarms for metrics
- Downloading metrics data in CSV format
- Getting metrics using the CloudWatch API

Viewing CloudFront and edge function metrics

You can view operational metrics about your CloudFront distributions and edge functions in the CloudFront console. To view these metrics, see the Monitoring page in the CloudFront console. To view graphs about the activity for a specific CloudFront distribution or edge function, choose one, and then choose **View distribution metrics** or **View metrics**.

Topics

- Viewing the default CloudFront distribution metrics
- Turning on additional CloudFront distribution metrics
- Viewing the default Lambda@Edge function metrics

- Viewing the default CloudFront Functions metrics

Viewing the default CloudFront distribution metrics

The following default metrics are included for all CloudFront distributions, at no additional cost:

Requests

The total number of viewer requests received by CloudFront, for all HTTP methods and for both HTTP and HTTPS requests.

Bytes downloaded

The total number of bytes downloaded by viewers for `GET`, `HEAD`, and `OPTIONS` requests.

Bytes uploaded

The total number of bytes that viewers uploaded to your origin with CloudFront, using `POST` and `PUT` requests.

4xx error rate

The percentage of all viewer requests for which the response's HTTP status code is `4xx`.

5xx error rate

The percentage of all viewer requests for which the response's HTTP status code is `5xx`.

Total error rate

The percentage of all viewer requests for which the response's HTTP status code is `4xx` or `5xx`.

These metrics are shown in graphs for each CloudFront distribution on the Monitoring page in the CloudFront console. On each graph, the totals are displayed at 1-minute granularity. In addition to viewing the graphs, you can also download metrics reports as CSV files.

You can customize the graphs by doing the following:

- To change the time range for the information displayed in the graphs, choose 1h (1 hour), 3h (3 hours), or another range, or specify a custom range.
- To change how often CloudFront updates the information in the graph, choose the down arrow next to the refresh icon, and then choose a refresh rate. The default refresh rate is 1 minute, but you can choose 10 seconds, 2 minutes, or other options.

To view CloudFront graphs in the CloudWatch console, choose **Add to dashboard**.

Turning on additional CloudFront distribution metrics

In addition to the default metrics, you can turn on additional metrics for an additional cost. For more information about the cost, see the section called “Estimating cost for the additional CloudFront metrics”.

These additional metrics must be turned on for each distribution separately:

Cache hit rate

The percentage of all cacheable requests for which CloudFront served the content from its cache. HTTP POST and PUT requests, and errors, are not considered cacheable requests.

Origin latency

The total time spent from when CloudFront receives a request to when it starts providing a response to the network (not the viewer), for requests that are served from the origin, not the CloudFront cache. This is also known as *first byte latency*, or *time-to-first-byte*.

Error rate by status code

The percentage of all viewer requests for which the response's HTTP status code is a particular code in the `4xx` or `5xx` range. This metric is available for all of the following error codes: `401`, `403`, `404`, `502`, `503`, and `504`.

Turning on additional metrics

You can turn on additional metrics in the CloudFront console, with Amazon CloudFormation, with the Amazon Command Line Interface (Amazon CLI), or with the CloudFront API.

Console

To turn on additional metrics (console)

1. Sign in to the Amazon Web Services Management Console and open the Monitoring page in the CloudFront console.
2. Choose the distribution to turn on additional metrics for, and then choose **View distribution metrics**.
3. Choose **Manage additional metrics**.

4. In the **Manage additional metrics** window, turn on **Enabled**. After you turn on the additional metrics, you can close the **Manage additional metrics** window.

After you turn on the additional metrics, they are shown in graphs. On each graph, the totals are displayed at 1-minute granularity. In addition to viewing the graphs, you can also download metrics reports as CSV files.

You can customize the graphs by doing the following:

- To change the time range for the information displayed in the graphs, choose 1h (1 hour), 3h (3 hours), or another range, or specify a custom range.
- To change how often CloudFront updates the information in the graph, choose the down arrow next to the refresh icon, and then choose a refresh rate. The default refresh rate is 1 minute, but you can choose 10 seconds, 2 minutes, or other options.

To view CloudFront graphs in the CloudWatch console, choose **Add to dashboard**.

Amazon CloudFormation

To turn on additional metrics with Amazon CloudFormation, use the `AWS::CloudFront::MonitoringSubscription` resource type. The following example shows the Amazon CloudFormation template syntax, in YAML format, for enabling additional metrics.

```
Type: AWS::CloudFront::MonitoringSubscription
Properties:
  DistributionId: EDFDVBD6EXAMPLE
  MonitoringSubscription:
    RealtimeMetricsSubscriptionConfig:
      RealtimeMetricsSubscriptionStatus: Enabled
```

CLI

To manage additional metrics with the Amazon Command Line Interface (Amazon CLI), use one of the following commands:

To turn on additional metrics for a distribution (CLI)

- Use the **create-monitoring-subscription** command, as in the following example. Replace *EDFDVBD6EXAMPLE* with the ID of the distribution that you are enabling additional metrics for.

```
aws cloudfront create-monitoring-subscription --
distribution-id EDFDVBD6EXAMPLE --monitoring-subscription
 RealtimeMetricsSubscriptionConfig={RealtimeMetricsSubscriptionStatus=Enabled}
```

To see whether additional metrics are turned on for a distribution (CLI)

- Use the **get-monitoring-subscription** command, as in the following example. Replace *EDFDVBD6EXAMPLE* with the ID of the distribution that you are checking.

```
aws cloudfront get-monitoring-subscription --distribution-id EDFDVBD6EXAMPLE
```

To turn off additional metrics for a distribution (CLI)

- Use the **delete-monitoring-subscription** command, as in the following example. Replace *EDFDVBD6EXAMPLE* with the ID of the distribution that you are turning off additional metrics for.

```
aws cloudfront delete-monitoring-subscription --distribution-id EDFDVBD6EXAMPLE
```

API

To manage additional metrics with the CloudFront API, use one of the following API operations.

- To turn on additional metrics for a distribution, use CreateMonitoringSubscription.
- To see whether additional metrics are turned on for a distribution, use GetMonitoringSubscription.
- To turn off additional metrics for a distribution, use DeleteMonitoringSubscription.

For more information about these API calls, see the API reference documentation for your Amazon SDK or other API client.

Estimating cost for the additional CloudFront metrics

When you turn on additional metrics for a distribution, CloudFront sends up to 8 metrics to CloudWatch in the US East (N. Virginia) Region. CloudWatch charges a low, fixed rate for each metric. This rate is charged only once per month, per metric (up to 8 metrics per distribution). This is a fixed rate, so your cost remains the same regardless of the number of requests or responses that the CloudFront distribution receives or sends. For the per-metric rate, see the Amazon CloudWatch pricing page and the CloudWatch pricing calculator. Additional API charges apply when you retrieve the metrics with the CloudWatch API.

Viewing the default Lambda@Edge function metrics

You can use CloudWatch metrics to monitor, in real time, problems with your Lambda@Edge functions. There's no additional charge for these metrics.

When you attach a Lambda@Edge function to a cache behavior in a CloudFront distribution, Lambda begins sending metrics to CloudWatch automatically. Metrics are available for all Lambda Regions, but to view metrics in the CloudWatch console or get the metric data from the CloudWatch API, you must use the US East (N. Virginia) Region (`us-east-1`). The metric group name is formatted as: `AWS/CloudFront/`*`distribution-ID`*, where *`distribution-ID`* is the ID of the CloudFront distribution that the Lambda@Edge function is associated with. For more information about CloudWatch metrics, see the Amazon CloudWatch User Guide.

The following default metrics are shown in graphs for each Lambda@Edge function on the Monitoring page in the CloudFront console:

- `5xx` error rate for Lambda@Edge
- Lambda execution errors
- Lambda invalid responses
- Lambda throttles

The graphs include the number of invocations, errors, throttles, and so on. On each graph, the totals are displayed at 1-minute granularity, grouped by Amazon Region.

If you see a spike in errors that you want to investigate, you can choose a function and then view log files by Amazon Region, until you determine which function is causing the problems and in which Amazon Region. For more information about troubleshooting Lambda@Edge errors, see:

- the section called "How to determine the type of failure"
- Four Steps for Debugging your Content Delivery on Amazon

You can customize the graphs by doing the following:

- To change the time range for the information displayed in the graphs, choose 1h (1 hour), 3h (3 hours), or another range, or specify a custom range.
- To change how often CloudFront updates the information in the graph, choose the down arrow next to the refresh icon, and then choose a refresh rate. The default refresh rate is 1 minute, but you can choose 10 seconds, 2 minutes, or other options.

To view the graphs in the CloudWatch console, choose **Add to dashboard**. You must use the US East (N. Virginia) Region (us-east-1) to view the graphs in the CloudWatch console.

Viewing the default CloudFront Functions metrics

CloudFront Functions sends operational metrics to Amazon CloudWatch so that you can monitor your functions. Viewing these metrics can help you troubleshoot, track, and debug issues. CloudFront Functions publishes the following metrics to CloudWatch:

- **Invocations** (`FunctionInvocations`) – The number of times the function was started (invoked) in a given time period.
- **Validation errors** (`FunctionValidationErrors`) – The number of validation errors produced by the function in a given time period. Validation errors occur when the function runs successfully but returns invalid data (an invalid event object).
- **Execution errors** (`FunctionExecutionErrors`) – The number of execution errors that occurred in a given time period. Execution errors occur when the function fails to complete successfully.
- **Compute utilization** (`FunctionComputeUtilization`) – The amount of time that the function took to run as a percentage of the maximum allowed time. For example, a value of 35 means that the function completed in 35% of the maximum allowed time. This metric is a number between 0 and 100.

If this value reaches or is near 100, the function has used or is close to using the allowed execution time and subsequent requests might be throttled. If your function is running at 80% or more utilization, we recommend that you review your function to reduce the execution time and improve utilization. For example, you might want to only log errors, simplify any complex regex expressions, or remove unnecessary parsing of complex JSON objects.

- **Throttles** (`FunctionThrottles`) – The number of times that the function was throttled in a given time period. Functions can be throttled for the following reasons:
  - The function continuously exceeds the maximum time allowed for execution
  - The function results in compilation errors
  - There is an unusually high number of requests per second

CloudFront KeyValueStore also sends the following operational metrics to Amazon CloudWatch:

- **Read requests** (`KvsReadRequests`) – The number of times the function successfully read from the key value store within a given time period.
- **Read errors** (`KvsReadErrors`) – The number of times the function failed to read from the key value store within a given time period.

To view these metrics in the CloudFront console, go to the Monitoring page. To view graphs for a specific function, choose **Functions**, select the function, and then choose **View function metrics**.

All of these metrics are published to CloudWatch in the US East (N. Virginia) Region (`us-east-1`), in the CloudFront namespace. You can also view these metrics in the CloudWatch console. In the CloudWatch console, you can view the metrics per function or per function per distribution.

You can also use CloudWatch to set alarms based on these metrics. For example, you can set an alarm based on the execution time (`FunctionComputeUilization`) metric, which represents the percentage of available time that your function took to run. When the execution time reaches a certain value for a certain amount of time—for example, greater than 70% of available time for 15 continuous minutes—the alarm is triggered. You specify the alarms value and its time unit when you create the alarm.

> **Note**
>
> CloudFront Functions sends metrics to CloudWatch only for functions in the `LIVE` stage that run in response to production requests and responses. When you test a function,

CloudFront doesn't send any metrics to CloudWatch. The test output contains information about errors, compute utilization, and function logs (`console.log()` statements), but this information is not sent to CloudWatch.

For information about how to get these metrics with the CloudWatch API, see the section called "Getting metrics using the API".

Creating alarms for metrics

In the CloudFront console, you can set alarms to notify you by Amazon Simple Notification Service (Amazon SNS) based on specific CloudFront metrics. You can set an alarm on the Alarms page in the CloudFront console.

To create an alarm in the console, specify the following values:

Metric

The metric for which you are creating the alarm.

Distribution

The CloudFront distribution for which you are creating the alarm.

Name of alarm

A name for the alarm.

Send a notification to

The Amazon SNS topic to send notification to if this metric triggers an alarm.

Whenever *<metric> <operator> <value>*

Specify when CloudWatch should trigger an alarm and send a notification to the Amazon SNS topic. For example, to receive a notification when the 5xx error rate exceeds 1%, specify the following:

Whenever Average of 5xxErrorRate > 1

Note the following about specifying values:

- Enter only whole numbers without punctuation. For example, to specify one thousand, enter **1000**.

- For 4xx, 5xx, and total error rates, the value that you specify is a percentage.
- For requests, bytes downloaded, and bytes uploaded, the value that you specify is units. For example, 1073742000 bytes.

For at least *<number>* consecutive periods of *<time period>*

Specify how many consecutive time periods of the specified duration the metric must meet the criteria before CloudWatch triggers an alarm. When you choose a value, aim for an appropriate balance between a value that does not alarm for temporary or fleeting problems, but does alarm for sustained or real problems.

Downloading metrics data in CSV format

You can download the CloudWatch metrics data for a CloudFront distribution in CSV format. You can download the data when you **View distribution metrics** for a particular distribution in the CloudFront console.

Information about the report

The first few rows of the report include the following information:

Version

The CloudFront reporting version.

Report

The name of the report.

DistributionID

The ID of the distribution for which you ran the report.

StartDateUTC

The beginning of the date range for which you ran the report, in Coordinated Universal Time (UTC).

EndDateUTC

The end of the date range for which you ran the report, in Coordinated Universal Time (UTC).

GeneratedTimeUTC

The date and time on which you ran the report, in Coordinated Universal Time (UTC).

Granularity

The time period for each row in the report, for example, `ONE_MINUTE`.

Data in the metrics report

The report includes the following values:

DistributionID

The ID of the distribution for which you ran the report.

FriendlyName

An alternate domain name (CNAME) for the distribution, if any. If a distribution has no alternate domain names, the list includes an origin domain name for the distribution.

TimeBucket

The hour or the day that the data applies to, in Coordinated Universal Time (UTC).

Requests

The total number of requests for all HTTP status codes (for example, `200`, `404`, and so on) and all methods (for example, `GET`, `HEAD`, `POST`, and so on) during the time period.

BytesDownloaded

The number of bytes that viewers downloaded for the specified distribution during the time period.

BytesUploaded

The number of bytes that viewers uploaded for the specified distribution during the time period.

TotalErrorRatePct

The percentage of requests for which the HTTP status code was a `4xx` or `5xx` error for the specified distribution during the time period.

4xxErrorRatePct

The percentage of requests for which the HTTP status code was a `4xx` error for the specified distribution during the time period.

5xxErrorRatePct

The percentage of requests for which the HTTP status code was a `5xx` error for the specified distribution during the time period.

If you have turned on additional metrics for your distribution, then the report also includes the following additional values:

401ErrorRatePct

The percentage of requests for which the HTTP status code was a `401` error for the specified distribution during the time period.

403ErrorRatePct

The percentage of requests for which the HTTP status code was a `403` error for the specified distribution during the time period.

404ErrorRatePct

The percentage of requests for which the HTTP status code was a `404` error for the specified distribution during the time period.

502ErrorRatePct

The percentage of requests for which the HTTP status code was a `502` error for the specified distribution during the time period.

503ErrorRatePct

The percentage of requests for which the HTTP status code was a `503` error for the specified distribution during the time period.

504ErrorRatePct

The percentage of requests for which the HTTP status code was a `504` error for the specified distribution during the time period.

OriginLatency

The total time spent, in milliseconds, from when CloudFront received a request to when it started providing a response to the network (not the viewer), for requests that were served from the origin, not the CloudFront cache. This is also known as *first byte latency*, or *time-to-first-byte*.

CacheHitRate

The percentage of all cacheable requests for which CloudFront served the content from its cache. HTTP POST and PUT requests, and errors, are not considered cacheable requests.

Getting metrics using the CloudWatch API

You can use the Amazon CloudWatch API or CLI to get the CloudFront metrics in programs or applications that you build. You can use the raw data to build your own custom dashboards, your own alarming tools, and so on.

To get the CloudFront metrics from the CloudWatch API, you must use the US East (N. Virginia) Region (`us-east-1`). You also need to know certain values and types for each metric.

Topics

- Values for all CloudFront metrics
- Values for CloudFront distribution metrics
- Values for CloudFront function metrics

Values for all CloudFront metrics

The following values apply to all CloudFront metrics:

`Namespace`

The value for `Namespace` is always `AWS/CloudFront`.

Dimensions

Each CloudFront metric has the following two dimensions:

`DistributionId`

The ID of the CloudFront distribution for which you want to get metrics.

`FunctionName`

The name of the function (in CloudFront Functions) for which you want to get metrics.

This dimension applies only to functions.

`Region`

The value for `Region` is always `Global`, because CloudFront is a global service.

> **Note**
>
> To get the CloudFront metrics from the CloudWatch API, you must use the US East (N. Virginia) Region (us-east-1).

Values for CloudFront distribution metrics

Use information from the following list to get details about specific CloudFront distribution metrics from the CloudWatch API. Some of these metrics are available only when you have turned on additional metrics for the distribution.

> **Note**
>
> Only one statistic, `Average` or `Sum`, is applicable for each metric. The following list specifies which statistic is applicable to that metric.

4xx error rate

The percentage of all viewer requests for which the response's HTTP status code is `4xx`.

- Metric name: `4xxErrorRate`
- Valid statistic: `Average`
- Unit: `Percent`

401 error rate

The percentage of all viewer requests for which the response's HTTP status code is `401`. To get this metric, you must first turn on additional metrics.

- Metric name: `401ErrorRate`
- Valid statistic: `Average`
- Unit: `Percent`

403 error rate

The percentage of all viewer requests for which the response's HTTP status code is `403`. To get this metric, you must first turn on additional metrics.

- Metric name: `403ErrorRate`
- Valid statistic: `Average`
- Unit: `Percent`

404 error rate

The percentage of all viewer requests for which the response's HTTP status code is `404`. To get this metric, you must first turn on additional metrics.

- Metric name: `404ErrorRate`
- Valid statistic: `Average`
- Unit: `Percent`

5xx error rate

The percentage of all viewer requests for which the response's HTTP status code is `5xx`.

- Metric name: `5xxErrorRate`
- Valid statistic: `Average`
- Unit: `Percent`

502 error rate

The percentage of all viewer requests for which the response's HTTP status code is `502`. To get this metric, you must first turn on additional metrics.

- Metric name: `502ErrorRate`
- Valid statistic: `Average`
- Unit: `Percent`

503 error rate

The percentage of all viewer requests for which the response's HTTP status code is `503`. To get this metric, you must first turn on additional metrics.

- Metric name: `503ErrorRate`
- Valid statistic: `Average`
- Unit: `Percent`

504 error rate

The percentage of all viewer requests for which the response's HTTP status code is `504`. To get this metric, you must first turn on additional metrics.

- Metric name: `504ErrorRate`
- Valid statistic: `Average`
- Unit: `Percent`

Bytes downloaded

The total number of bytes downloaded by viewers for `GET`, `HEAD`, and `OPTIONS` requests.

- Metric name: `BytesDownloaded`
- Valid statistic: `Sum`
- Unit: `None`

Bytes uploaded

The total number of bytes that viewers uploaded to your origin with CloudFront, using `POST` and `PUT` requests.

- Metric name: `BytesUploaded`
- Valid statistic: `Sum`
- Unit: `None`

Cache hit rate

The percentage of all cacheable requests for which CloudFront served the content from its cache. HTTP `POST` and `PUT` requests, and errors, are not considered cacheable requests. To get this metric, you must first turn on additional metrics.

- Metric name: `CacheHitRate`
- Valid statistic: `Average`
- Unit: `Percent`

Origin latency

The total time spent, in milliseconds, from when CloudFront receives a request to when it starts providing a response to the network (not the viewer), for requests that are served from the origin, not the CloudFront cache. This is also known as *first byte latency*, or *time-to-first-byte*. To get this metric, you must first turn on additional metrics.

- Metric name: `OriginLatency`
- Valid statistic: `Percentile`
- Unit: `Milliseconds`

> **Note**
>
> To get a `Percentile` statistic from the CloudWatch API, use the `ExtendedStatistics` parameter, not `Statistics`. For more information, see GetMetricStatistics in the *Amazon CloudWatch API Reference*, or the reference documentation for the Amazon SDKs.

Requests

The total number of viewer requests received by CloudFront, for all HTTP methods and for both HTTP and HTTPS requests.

- Metric name: `Requests`
- Valid statistic: `Sum`
- Unit: None

Total error rate

The percentage of all viewer requests for which the response's HTTP status code is `4xx` or `5xx`.

- Metric name: `TotalErrorRate`
- Valid statistic: `Average`
- Unit: `Percent`

Values for CloudFront function metrics

Use information from the following list to get details about specific CloudFront function metrics from the CloudWatch API.

> **Note**
>
> Only one statistic, `Average` or `Sum`, is applicable for each metric. The following list specifies which statistic is applicable to that metric.

Invocations

The number of times the function was started (invoked) in a given time period.

- Metric name: `FunctionInvocations`
- Valid statistic: `Sum`
- Unit: `None`

Validation errors

The number of validation errors produced by the function in a given time period. Validation errors occur when the function runs successfully but returns invalid data (an invalid event object).

- Metric name: `FunctionValidationErrors`
- Valid statistic: `Sum`
- Unit: `None`

Execution errors

The number of execution errors that occurred in a given time period. Execution errors occur when the function fails to complete successfully.

- Metric name: `FunctionExecutionErrors`
- Valid statistic: `Sum`
- Unit: `None`

Compute utilization

The amount of time (0-100) that the function took to run as a percentage of the maximum allowed time. For example, a value of 35 means that the function completed in 35% of the maximum allowed time.

- Metric name: `FunctionComputeUtilization`
- Valid statistic: `Average`
- Unit: `Percent`

Throttles

The number of times that the function was throttled in a given time period.

- Metric name: `FunctionThrottles`
- Valid statistic: `Sum`
- Unit: `None`

CloudFront and edge function logging

Amazon CloudFront provides different kinds of logging. You can log the viewer requests that come to your CloudFront distributions, or you can log the CloudFront service activity (API activity) in your Amazon account. You can also get logs from your edge computing functions.

Logging requests

CloudFront provides the following ways to log the requests that come to your distributions.

Standard logs (access logs)

CloudFront standard logs provide detailed records about every request that's made to a distribution. These logs are useful for many scenarios, including security and access audits.

CloudFront standard logs are delivered to the Amazon S3 bucket of your choice. CloudFront doesn't charge for standard logs, though you incur Amazon S3 charges for storing and accessing the log files.

For more information, see Using standard logs (access logs).

Real-time logs

CloudFront real-time logs provide information about requests made to a distribution, in real time (log records are delivered within seconds of receiving the requests). You can choose the *sampling rate* for your real-time logs—that is, the percentage of requests for which you want to receive real-time log records. You can also choose the specific fields that you want to receive in the log records.

CloudFront real-time logs are delivered to the data stream of your choice in Amazon Kinesis Data Streams. CloudFront charges for real-time logs, in addition to the charges you incur for using Kinesis Data Streams.

For more information, see Real-time logs.

Logging edge functions

You can use Amazon CloudWatch Logs to get logs for your edge functions, both Lambda@Edge and CloudFront Functions. You can access the logs using the CloudWatch console or the CloudWatch Logs API. For more information, see the section called “Edge function logs”.

Logging service activity

You can use Amazon CloudTrail to log the CloudFront service activity (API activity) in your Amazon account. CloudTrail provides a record of API actions taken by a user, role, or Amazon service in CloudFront. Using the information collected by CloudTrail, you can determine the API request that was made to CloudFront, the IP address from which the request was made, who made the request, when it was made, and additional details.

For more information, see Logging Amazon CloudFront API calls using Amazon CloudTrail.

Topics

- Configuring and using standard logs (access logs)
- Real-time logs
- Edge function logs
- Logging Amazon CloudFront API calls using Amazon CloudTrail

Configuring and using standard logs (access logs)

You can configure CloudFront to create log files that contain detailed information about every user request that CloudFront receives. These are called *standard logs*, also known as *access logs*. If you enable standard logs, you can also specify the Amazon S3 bucket that you want CloudFront to save files in.

You can enable standard logs when you create or update a distribution. For more information, see Values that you specify when you create or update a distribution.

CloudFront also offers real-time logs, which give you information about requests made to a distribution in real time (logs are delivered within seconds of receiving the requests). You can use real-time logs to monitor, analyze, and take action based on content delivery performance. For more information, see Real-time logs.

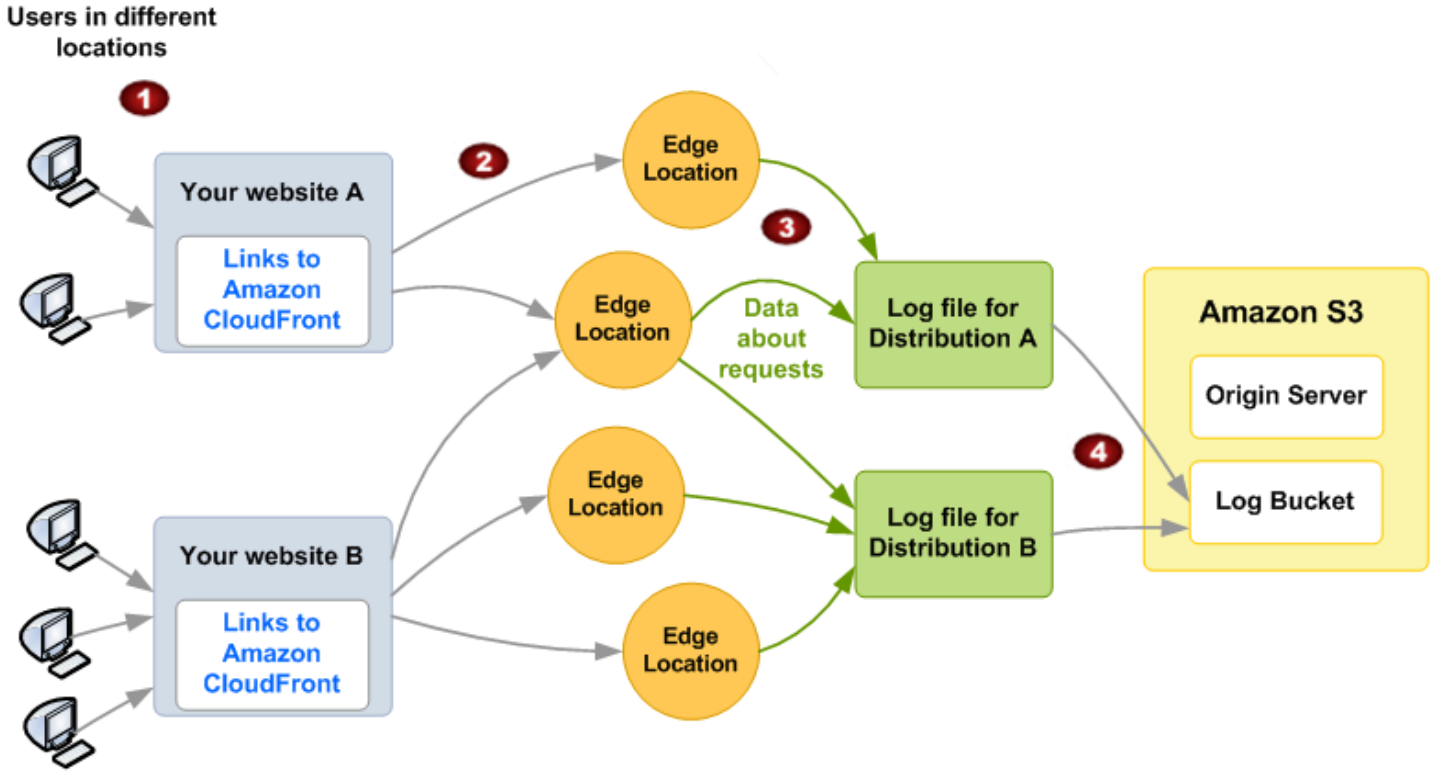
Topics

- How standard logging works
- Choosing an Amazon S3 bucket for your standard logs
- Permissions required to configure standard logging and to access your log files
- Required key policy for SSE-KMS buckets

- File name format
- Timing of standard log file delivery
- How requests are logged when the request URL or headers exceed the maximum size
- Analyzing standard logs
- Editing your standard logging settings
- Deleting standard log files from an Amazon S3 bucket
- Standard log file format
- Charges for standard logs

How standard logging works

The following diagram shows how CloudFront logs information about requests for your objects.



The following explains how CloudFront logs information about requests for your objects, as illustrated in the previous diagram.

1. In this diagram, you have two websites, A and B, and two corresponding CloudFront distributions. Users request your objects using URLs that are associated with your distributions.
2. CloudFront routes each request to the appropriate edge location.

3. CloudFront writes data about each request to a log file specific to that distribution. In this example, information about requests related to Distribution A goes into a log file just for Distribution A, and information about requests related to Distribution B goes into a log file just for Distribution B.
4. CloudFront periodically saves the log file for a distribution in the Amazon S3 bucket that you specified when you enabled logging. CloudFront then starts saving information about subsequent requests in a new log file for the distribution.

If no users access your content during a given hour, you don't receive any log files for that hour.

Each entry in a log file gives details about a single request. For more information about log file format, see Standard log file format.

> **Note**
>
> We recommend that you use the logs to understand the nature of the requests for your content, not as a complete accounting of all requests. CloudFront delivers access logs on a best-effort basis. The log entry for a particular request might be delivered long after the request was actually processed and, in rare cases, a log entry might not be delivered at all. When a log entry is omitted from access logs, the number of entries in the access logs won't match the usage that appears in the Amazon billing and usage reports.

Choosing an Amazon S3 bucket for your standard logs

When you enable logging for a distribution, you specify the Amazon S3 bucket that you want CloudFront to store log files in. If you're using Amazon S3 as your origin, we recommend that you don't use the same bucket for your log files; using a separate bucket simplifies maintenance.

> **Important**
>
> Don't choose an Amazon S3 bucket with S3 Object Ownership set to **bucket owner enforced**. That setting disables ACLs for the bucket and the objects in it, which prevents CloudFront from delivering log files to the bucket.

Important

Don't choose an Amazon S3 bucket in any of the following Regions, because CloudFront doesn't deliver standard logs to buckets in these Regions:

- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asia Pacific (Hyderabad)
- Asia Pacific (Jakarta)
- Asia Pacific (Melbourne)
- Canada West (Calgary)
- Europe (Milan)
- Europe (Spain)
- Europe (Zurich)
- Israel (Tel Aviv)
- Middle East (Bahrain)
- Middle East (UAE)

You can store the log files for multiple distributions in the same bucket. When you enable logging, you can specify an optional prefix for the file names, so you can keep track of which log files are associated with which distributions.

Permissions required to configure standard logging and to access your log files

Important

Starting in April 2023, you will need to enable S3 access control lists (ACLs) for new S3 buckets being used for CloudFront standard logs. ACLs can be enabled during the bucket creation steps, or after a bucket has been created.
For more information about the changes, see Default settings for new S3 buckets FAQ in the *Amazon Simple Storage Service User Guide* and Heads-Up: Amazon S3 Security Changes Are Coming in April of 2023 in the *Amazon News Blog*.

Your Amazon account must have the following permissions for the bucket that you specify for log files:

- The S3 access control list (ACL) for the bucket must grant you `FULL_CONTROL`. If you're the bucket owner, your account has this permission by default. If you're not, the bucket owner must update the ACL for the bucket.
- `s3:GetBucketAcl`
- `s3:PutBucketAcl`

Note the following:

ACL for the bucket

When you create or update a distribution and enable logging, CloudFront uses these permissions to update the ACL for the bucket to give the `awslogsdelivery` account `FULL_CONTROL` permission. The `awslogsdelivery` account writes log files to the bucket. If your account doesn't have the required permissions to update the ACL, creating or updating the distribution will fail.

In some circumstances, if you programmatically submit a request to create a bucket but a bucket with the specified name already exists, S3 resets permissions on the bucket to the default value. If you configured CloudFront to save access logs in an S3 bucket and you stop getting logs in that bucket, check permissions on the bucket to ensure that CloudFront has the necessary permissions.

Restoring the ACL for the bucket

If you remove permissions for the `awslogsdelivery` account, CloudFront won't be able to save logs to the S3 bucket. To enable CloudFront to start saving logs for your distribution again, restore the ACL permission by doing one of the following:

- Disable logging for your distribution in CloudFront, and then enable it again. For more information, see Values that you specify when you create or update a distribution.
- Add the ACL permission for `awslogsdelivery` manually by navigating to the S3 bucket in the Amazon S3 console and adding permission. To add the ACL for `awslogsdelivery`, you must provide the canonical ID for the account, which is the following:

```
a52cb28745c0c06e84ec548334e44bfa7fc2a85c54af20cd59e4969344b7af56
```

For more information about adding ACLs to S3 buckets, see How Do I Set ACL Bucket Permissions? in the *Amazon Simple Storage Service User Guide*.

ACL for each log file

In addition to the ACL on the bucket, there's an ACL on each log file. The bucket owner has `FULL_CONTROL` permission on each log file, the distribution owner (if different from the bucket owner) has no permission, and the `awslogsdelivery` account has read and write permissions.

Disabling logging

If you disable logging, CloudFront doesn't delete the ACLs for either the bucket or the log files. If you want, you can do that yourself.

Required key policy for SSE-KMS buckets

If the S3 bucket for your standard logs uses server-side encryption with Amazon KMS keys (SSE-KMS) using a customer managed key, you must add the following statement to the key policy for your customer managed key. This allows CloudFront to write log files to the bucket. (You can't use SSE-KMS with the Amazon managed key because CloudFront won't be able to write log files to the bucket.)

```
{
    "Sid": "Allow CloudFront to use the key to deliver logs",
    "Effect": "Allow",
    "Principal": {
        "Service": "delivery.logs.amazonaws.com"
    },
    "Action": "kms:GenerateDataKey*",
    "Resource": "*"
}
```

If the S3 bucket for your standard logs uses SSE-KMS with an S3 Bucket Key, you also need to add the `kms:Decrypt` permission to policy statement. In that case, the full policy statement looks like the following.

```
{
    "Sid": "Allow CloudFront to use the key to deliver logs",
    "Effect": "Allow",
    "Principal": {
```

```
        "Service": "delivery.logs.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey*",
        "kms:Decrypt"
    ],
    "Resource": "*"
}
```

File name format

The name of each log file that CloudFront saves in your Amazon S3 bucket uses the following file name format:

<optional prefix>/*<distribution ID>*.*YYYY*-*MM*-*DD*-*HH*.*unique-ID*.gz

The date and time are in Coordinated Universal Time (UTC).

For example, if you use `example-prefix` as the prefix, and your distribution ID is `EMLARXS9EXAMPLE`, your file names look similar to this:

```
example-prefix/EMLARXS9EXAMPLE.2019-11-14-20.RT4KCN4SGK9.gz
```

When you enable logging for a distribution, you can specify an optional prefix for the file names, so you can keep track of which log files are associated with which distributions. If you include a value for the log file prefix and your prefix doesn't end with a forward slash (/), CloudFront appends one automatically. If your prefix does end with a forward slash, CloudFront doesn't add another one.

The `.gz` at the end of the file name indicates that CloudFront has compressed the log file using gzip.

Timing of standard log file delivery

CloudFront delivers standard logs for a distribution up to several times an hour. In general, a log file contains information about the requests that CloudFront received during a given time period. CloudFront usually delivers the log file for that time period to your Amazon S3 bucket within an hour of the events that appear in the log. Note, however, that some or all log file entries for a time period can sometimes be delayed by up to 24 hours. When log entries are delayed, CloudFront saves them in a log file for which the file name includes the date and time of the period in which the requests occurred, not the date and time when the file was delivered.

When creating a log file, CloudFront consolidates information for your distribution from all of the edge locations that received requests for your objects during the time period that the log file covers.

CloudFront can save more than one file for a time period depending on how many requests CloudFront receives for the objects associated with a distribution.

CloudFront begins to reliably deliver access logs about four hours after you enable logging. You might get a few access logs before that time.

> **Note**
>
> If no users request your objects during the time period, you don't receive any log files for that period.

CloudFront also offers real-time logs, which give you information about requests made to a distribution in real time (logs are delivered within seconds of receiving the requests). You can use real-time logs to monitor, analyze, and take action based on content delivery performance. For more information, see Real-time logs.

How requests are logged when the request URL or headers exceed the maximum size

If the total size of all request headers, including cookies, exceeds 20 KB, or if the URL exceeds 8192 bytes, CloudFront can't parse the request completely and can't log the request. Because the request isn't logged, you won't see in the log files the HTTP error status code returned.

If the request body exceeds the maximum size, the request is logged, including the HTTP error status code.

Analyzing standard logs

Because you can receive multiple access logs per hour, we recommend that you combine all the log files you receive for a given time period into one file. You can then analyze the data for that period more accurately and completely.

One way to analyze your access logs is to use Amazon Athena. Athena is an interactive query service that can help you analyze data for Amazon services, including CloudFront. To learn more, see Querying Amazon CloudFront Logs in the *Amazon Athena User Guide*.

In addition, the following Amazon blog posts discuss some ways to analyze access logs.

- Amazon CloudFront Request Logging (for content delivered via HTTP)
- Enhanced CloudFront Logs, Now With Query Strings

> **Important**
>
> We recommend that you use the logs to understand the nature of the requests for your content, not as a complete accounting of all requests. CloudFront delivers access logs on a best-effort basis. The log entry for a particular request might be delivered long after the request was actually processed and, in rare cases, a log entry might not be delivered at all. When a log entry is omitted from access logs, the number of entries in the access logs won't match the usage that appears in the Amazon usage and billing reports.

Editing your standard logging settings

You can enable or disable logging, change the Amazon S3 bucket where your logs are stored, and change the prefix for log files by using the CloudFront console or the CloudFront API. Your changes to logging settings take effect within 12 hours.

For more information, see the following topics:

- To update a distribution using the CloudFront console, see Updating a distribution.
- To update a distribution using the CloudFront API, see UpdateDistribution in the *Amazon CloudFront API Reference*.

Deleting standard log files from an Amazon S3 bucket

CloudFront does not automatically delete log files from your Amazon S3 bucket. For information about deleting log files from an Amazon S3 bucket, see the following topics:

- Using the Amazon S3 console: Deleting Objects in the *Amazon Simple Storage Service Console User Guide*.
- Using the REST API: DeleteObject in the *Amazon Simple Storage Service API Reference*.

Standard log file format

Each entry in a log file gives details about a single viewer request. The log files have the following characteristics:

- Use the W3C extended log file format.
- Contain tab-separated values.
- Contain records that are not necessarily in chronological order.
- Contain two header lines: one with the file format version, and another that lists the W3C fields included in each record.
- Contain URL-encoded equivalents for spaces and certain other characters in field values.

  URL-encoded equivalents are used for the following characters:

  - ASCII character codes 0 through 32, inclusive
  - ASCII character codes 127 and higher
  - All characters in the following table

  The URL encoding standard is defined in RFC 1738.

URL-Encoded value	Character
%3C	<
%3E	>
%22	"
%23	#
%25	%
%7B	{
%7D	}
%7C	|
%5C	\

URL-Encoded value	Character
%5E	^
%7E	~
%5B	[
%5D	]
%60	`
%27	'
%20	space

Standard log file fields

The log file for a distribution contains 33 fields. The following list contains each field name, in order, along with a description of the information in that field.

1. **`date`**

   The date on which the event occurred in the format `YYYY-MM-DD`. For example, `2019-06-30`. The date and time are in Coordinated Universal Time (UTC). For WebSocket connections, this is the date when the connection closed.

2. **`time`**

   The time when the CloudFront server finished responding to the request (in UTC), for example, `01:42:39`. For WebSocket connections, this is the time when the connection is closed.

3. **`x-edge-location`**

   The edge location that served the request. Each edge location is identified by a three-letter code and an arbitrarily assigned number (for example, DFW3). The three-letter code typically corresponds with the International Air Transport Association (IATA) airport code for an airport near the edge location's geographic location. (These abbreviations might change in the future.)

4. **`sc-bytes`**

The total number of bytes that the server sent to the viewer in response to the request, including headers. For WebSocket connections, this is the total number of bytes sent from the server to the client through the connection.

5. **`c-ip`**

   The IP address of the viewer that made the request, for example, `192.0.2.183` or `2001:0db8:85a3::8a2e:0370:7334`. If the viewer used an HTTP proxy or a load balancer to send the request, the value of this field is the IP address of the proxy or load balancer. See also the `x-forwarded-for` field.

6. **`cs-method`**

   The HTTP request method received from the viewer.

7. **`cs(Host)`**

   The domain name of the CloudFront distribution (for example, d111111abcdef8.cloudfront.net).

8. **`cs-uri-stem`**

   The portion of the request URL that identifies the path and object (for example, `/images/cat.jpg`). Question marks (?) in URLs and query strings are not included in the log.

9. **`sc-status`**

   Contains one of the following values:

   - The HTTP status code of the server's response (for example, `200`).
   - `000`, which indicates that the viewer closed the connection before the server could respond to the request. If the viewer closes the connection after the server starts to send the response, this field contains the HTTP status code of the response that the server started to send.

10. **`cs(Referer)`**

    The value of the `Referer` header in the request. This is the name of the domain that originated the request. Common referrers include search engines, other websites that link directly to your objects, and your own website.

11. **`cs(User-Agent)`**

    The value of the `User-Agent` header in the request. The `User-Agent` header identifies the source of the request, such as the type of device and browser that submitted the request or, if the request came from a search engine, which search engine.

12 **cs-uri-query**

The query string portion of the request URL, if any.

When a URL doesn't contain a query string, this field's value is a hyphen (-). For more information, see Caching content based on query string parameters.

13 **cs(Cookie)**

The `Cookie` header in the request, including name—value pairs and the associated attributes.

If you enable cookie logging, CloudFront logs the cookies in all requests regardless of which cookies you choose to forward to the origin. When a request doesn't include a cookie header, this field's value is a hyphen (-). For more information about cookies, see Caching content based on cookies.

14 **x-edge-result-type**

How the server classified the response after the last byte left the server. In some cases, the result type can change between the time that the server is ready to send the response and the time that it finishes sending the response. See also the `x-edge-response-result-type` field.

For example, in HTTP streaming, suppose the server finds a segment of the stream in the cache. In that scenario, the value of this field would ordinarily be `Hit`. However, if the viewer closes the connection before the server has delivered the entire segment, the final result type (and the value of this field) is `Error`.

WebSocket connections will have a value of `Miss` for this field because the content is not cacheable and is proxied directly to the origin.

Possible values include:

- `Hit` – The server served the object to the viewer from the cache.
- `RefreshHit` – The server found the object in the cache but the object had expired, so the server contacted the origin to verify that the cache had the latest version of the object.
- `Miss` – The request could not be satisfied by an object in the cache, so the server forwarded the request to the origin and returned the result to the viewer.
- `LimitExceeded` – The request was denied because a CloudFront quota (formerly referred to as a limit) was exceeded.
- `CapacityExceeded` – The server returned an HTTP 503 status code because it didn't have enough capacity at the time of the request to serve the object.

- `Error` – Typically, this means the request resulted in a client error (the value of the `sc-status` field is in the `4xx` range) or a server error (the value of the `sc-status` field is in the `5xx` range). If the value of the `sc-status` field is `200`, or if the value of this field is `Error` and the value of the `x-edge-response-result-type` field is not `Error`, it means the HTTP request was successful but the client disconnected before receiving all of the bytes.
- `Redirect` – The server redirected the viewer from HTTP to HTTPS according to the distribution settings.

15**`x-edge-request-id`**

An opaque string that uniquely identifies a request. CloudFront also sends this string in the `x-amz-cf-id` response header.

16**`x-host-header`**

The value that the viewer included in the `Host` header of the request. If you're using the CloudFront domain name in your object URLs (such as d111111abcdef8.cloudfront.net), this field contains that domain name. If you're using alternate domain names (CNAMEs) in your object URLs (such as www.example.com), this field contains the alternate domain name.

If you're using alternate domain names, see `cs(Host)` in field 7 for the domain name that is associated with your distribution.

17**`cs-protocol`**

The protocol of the viewer request (`http`, `https`, `ws`, or `wss`).

18**`cs-bytes`**

The total number of bytes of data that the viewer included in the request, including headers. For WebSocket connections, this is the total number of bytes sent from the client to the server on the connection.

19**`time-taken`**

The number of seconds (to the thousandth of a second, for example, 0.082) from when the server receives the viewer's request to when the server writes the last byte of the response to the output queue, as measured on the server. From the perspective of the viewer, the total time to get the full response will be longer than this value because of network latency and TCP buffering.

20**`x-forwarded-for`**

If the viewer used an HTTP proxy or a load balancer to send the request, the value of the `c-ip` field is the IP address of the proxy or load balancer. In that case, this field is the IP address of the viewer that originated the request. This field can contain multiple comma-separated IP addresses. Each IP address can be an IPv4 address (for example, `192.0.2.183`) or an IPv6 address (for example, `2001:0db8:85a3::8a2e:0370:7334`).

If the viewer did not use an HTTP proxy or a load balancer, the value of this field is a hyphen (-).

21 **`ssl-protocol`**

When the request used HTTPS, this field contains the SSL/TLS protocol that the viewer and server negotiated for transmitting the request and response. For a list of possible values, see the supported SSL/TLS protocols in Supported protocols and ciphers between viewers and CloudFront.

When `cs-protocol` in field 17 is `http`, the value for this field is a hyphen (-).

22 **`ssl-cipher`**

When the request used HTTPS, this field contains the SSL/TLS cipher that the viewer and server negotiated for encrypting the request and response. For a list of possible values, see the supported SSL/TLS ciphers in Supported protocols and ciphers between viewers and CloudFront.

When `cs-protocol` in field 17 is `http`, the value for this field is a hyphen (-).

23 **`x-edge-response-result-type`**

How the server classified the response just before returning the response to the viewer. See also the `x-edge-result-type` field. Possible values include:

- `Hit` – The server served the object to the viewer from the cache.
- `RefreshHit` – The server found the object in the cache but the object had expired, so the server contacted the origin to verify that the cache had the latest version of the object.
- `Miss` – The request could not be satisfied by an object in the cache, so the server forwarded the request to the origin server and returned the result to the viewer.
- `LimitExceeded` – The request was denied because a CloudFront quota (formerly referred to as a limit) was exceeded.
- `CapacityExceeded` – The server returned a 503 error because it didn't have enough capacity at the time of the request to serve the object.

- `Error` – Typically, this means the request resulted in a client error (the value of the `sc-status` field is in the `4xx` range) or a server error (the value of the `sc-status` field is in the `5xx` range).

  If the value of the `x-edge-result-type` field is `Error` and the value of this field is not `Error`, the client disconnected before finishing the download.
- `Redirect` – The server redirected the viewer from HTTP to HTTPS according to the distribution settings.

24**`cs-protocol-version`**

The HTTP version that the viewer specified in the request. Possible values include `HTTP/0.9`, `HTTP/1.0`, `HTTP/1.1`, `HTTP/2.0`, and `HTTP/3.0`.

25**`fle-status`**

When field-level encryption is configured for a distribution, this field contains a code that indicates whether the request body was successfully processed. When the server successfully processes the request body, encrypts values in the specified fields, and forwards the request to the origin, the value of this field is `Processed`. The value of `x-edge-result-type` can still indicate a client-side or server-side error in this case.

Possible values for this field include:

- `ForwardedByContentType` – The server forwarded the request to the origin without parsing or encryption because no content type was configured.
- `ForwardedByQueryArgs` – The server forwarded the request to the origin without parsing or encryption because the request contains a query argument that wasn't in the configuration for field-level encryption.
- `ForwardedDueToNoProfile` – The server forwarded the request to the origin without parsing or encryption because no profile was specified in the configuration for field-level encryption.
- `MalformedContentTypeClientError` – The server rejected the request and returned an HTTP 400 status code to the viewer because the value of the `Content-Type` header was in an invalid format.
- `MalformedInputClientError` – The server rejected the request and returned an HTTP 400 status code to the viewer because the request body was in an invalid format.
- `MalformedQueryArgsClientError` – The server rejected the request and returned an HTTP 400 status code to the viewer because a query argument was empty or in an invalid format.

- `RejectedByContentType` – The server rejected the request and returned an HTTP 400 status code to the viewer because no content type was specified in the configuration for field-level encryption.
- `RejectedByQueryArgs` – The server rejected the request and returned an HTTP 400 status code to the viewer because no query argument was specified in the configuration for field-level encryption.
- `ServerError` – The origin server returned an error.

If the request exceeds a field-level encryption quota (formerly referred to as a limit), this field contains one of the following error codes, and the server returns HTTP status code 400 to the viewer. For a list of the current quotas on field-level encryption, see Quotas on field-level encryption.

- `FieldLengthLimitClientError` – A field that is configured to be encrypted exceeded the maximum length allowed.
- `FieldNumberLimitClientError` – A request that the distribution is configured to encrypt contains more than the number of fields allowed.
- `RequestLengthLimitClientError` – The length of the request body exceeded the maximum length allowed when field-level encryption is configured.

If field-level encryption is not configured for the distribution, the value of this field is a hyphen (-).

26 **`fle-encrypted-fields`**

The number of field-level encryption fields that the server encrypted and forwarded to the origin. CloudFront servers stream the processed request to the origin as they encrypt data, so this field can have a value even if the value of `fle-status` is an error.

If field-level encryption is not configured for the distribution, the value of this field is a hyphen (-).

27 **`c-port`**

The port number of the request from the viewer.

28 **`time-to-first-byte`**

The number of seconds between receiving the request and writing the first byte of the response, as measured on the server.

29**x-edge-detailed-result-type**

This field contains the same value as the `x-edge-result-type` field, except in the following cases:

- When the object was served to the viewer from the Origin Shield layer, this field contains `OriginShieldHit`.
- When the object was not in the CloudFront cache and the response was generated by an origin request Lambda@Edge function, this field contains `MissGeneratedResponse`.
- When the value of the `x-edge-result-type` field is `Error`, this field contains one of the following values with more information about the error:
  - `AbortedOrigin` – The server encountered an issue with the origin.
  - `ClientCommError` – The response to the viewer was interrupted due to a communication problem between the server and the viewer.
  - `ClientGeoBlocked` – The distribution is configured to refuse requests from the viewer's geographic location.
  - `ClientHungUpRequest` – The viewer stopped prematurely while sending the request.
  - `Error` – An error occurred for which the error type doesn't fit any of the other categories. This error type can occur when the server serves an error response from the cache.
  - `InvalidRequest` – The server received an invalid request from the viewer.
  - `InvalidRequestBlocked` – Access to the requested resource is blocked.
  - `InvalidRequestCertificate` – The distribution doesn't match the SSL/TLS certificate for which the HTTPS connection was established.
  - `InvalidRequestHeader` – The request contained an invalid header.
  - `InvalidRequestMethod` – The distribution is not configured to handle the HTTP request method that was used. This can happen when the distribution supports only cacheable requests.
  - `OriginCommError` – The request timed out while connecting to the origin, or reading data from the origin.
  - `OriginConnectError` – The server couldn't connect to the origin.
  - `OriginContentRangeLengthError` – The `Content-Length` header in the origin's response doesn't match the length in the `Content-Range` header.
  - `OriginDnsError` – The server couldn't resolve the origin's domain name.
  - `OriginError` – The origin returned an incorrect response.

- `OriginHeaderTooBigError` – A header returned by the origin is too big for the edge server to process.
- `OriginInvalidResponseError` – The origin returned an invalid response.
- `OriginReadError` – The server couldn't read from the origin.
- `OriginWriteError` – The server couldn't write to the origin.
- `OriginZeroSizeObjectError` – A zero size object sent from the origin resulted in an error.
- `SlowReaderOriginError` – The viewer was slow to read the message that caused the origin error.

30**sc-content-type**

The value of the HTTP `Content-Type` header of the response.

31**sc-content-len**

The value of the HTTP `Content-Length` header of the response.

32**sc-range-start**

When the response contains the HTTP `Content-Range` header, this field contains the range start value.

33**sc-range-end**

When the response contains the HTTP `Content-Range` header, this field contains the range end value.

The following is an example log file for a distribution:

```
#Version: 1.0
#Fields: date time x-edge-location sc-bytes c-ip cs-method cs(Host) cs-uri-stem sc-
status cs(Referer) cs(User-Agent) cs-uri-query cs(Cookie) x-edge-result-type x-edge-
request-id x-host-header cs-protocol cs-bytes time-taken x-forwarded-for ssl-protocol
 ssl-cipher x-edge-response-result-type cs-protocol-version fle-status fle-encrypted-
fields c-port time-to-first-byte x-edge-detailed-result-type sc-content-type sc-
content-len sc-range-start sc-range-end
2019-12-04 21:02:31 LAX1 392 192.0.2.100 GET d111111abcdef8.cloudfront.net /
index.html 200 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;
%20x64)%20AppleWebKit/537.36%20(KHTML,%20like
%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
```

```
 SOX4xwn4XV6Q4rgb7XiVGOHms_BGlTAC4KyHmureZmBNrjGdRLiNIQ== d111111abcdef8.cloudfront.net
 https 23 0.001 - TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 Hit HTTP/2.0 - - 11040 0.001 Hit
 text/html 78 - -
2019-12-04 21:02:31 LAX1 392 192.0.2.100 GET d111111abcdef8.cloudfront.net /
index.html 200 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;
%20x64)%20AppleWebKit/537.36%20(KHTML,%20like
%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
 k6WGMNkEzR5BEM_SaF47gjtX9zBDO2m349OY2an0QPEaUum1ZOLrow== d111111abcdef8.cloudfront.net
 https 23 0.000 - TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 Hit HTTP/2.0 - - 11040 0.000 Hit
 text/html 78 - -
2019-12-04 21:02:31 LAX1 392 192.0.2.100 GET d111111abcdef8.cloudfront.net /
index.html 200 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;
%20x64)%20AppleWebKit/537.36%20(KHTML,%20like
%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
 f37nTMVvnKvV2ZSvEsivup_c2kZ7VXzYdjC-GUQZ5qNs-89BlWazbw== d111111abcdef8.cloudfront.net
 https 23 0.001 - TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 Hit HTTP/2.0 - - 11040 0.001 Hit
 text/html 78 - -
2019-12-13 22:36:27 SEA19-C1 900 192.0.2.200 GET d111111abcdef8.cloudfront.net /
favicon.ico 502 http://www.example.com/ Mozilla/5.0%20(Windows
%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Error
 1pkpNfBQ39sYMnjjUQjmH2w1wdJnbHYTbag21o_3OfcQgPzdL2RSSQ== www.example.com http 675
 0.102 - - - Error HTTP/1.1 - - 25260 0.102 OriginDnsError text/html 507 - -
2019-12-13 22:36:26 SEA19-C1 900 192.0.2.200 GET d111111abcdef8.cloudfront.net / 502
 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Error
 3AqrZGCnF_g0-5KOvfA7c9XLcf4YGvMFSeFdIetR1N_2y8jSis8Zxg== www.example.com http 735
 0.107 - - - Error HTTP/1.1 - - 3802 0.107 OriginDnsError text/html 507 - -
2019-12-13 22:37:02 SEA19-C2 900 192.0.2.200 GET d111111abcdef8.cloudfront.net / 502
 - curl/7.55.1 - - Error kBkDzGnceVtWHqSCqBUqtA_cEs2T3tFUBbnBNkB9El_uVRhHgcZfcw==
 www.example.com http 387 0.103 - - - Error HTTP/1.1 - - 12644 0.103 OriginDnsError
 text/html 507 - -
```

Charges for standard logs

Standard logging is an optional feature of CloudFront. There is no extra charge for enabling standard logging. However, you accrue the usual Amazon S3 charges for storing and accessing the files on Amazon S3 (you can delete them at any time).

For more information about Amazon S3 pricing, see Amazon S3 Pricing.

For more information about CloudFront pricing, see CloudFront Pricing.

Real-time logs

With CloudFront real-time logs, you can get information about requests made to a distribution in real time (logs are delivered within seconds of receiving the requests). You can use real-time logs to monitor, analyze, and take action based on content delivery performance.

CloudFront real-time logs are configurable. You can choose:

- The *sampling rate* for your real-time logs—that is, the percentage of requests for which you want to receive real-time log records.
- The specific fields that you want to receive in the log records.
- The specific cache behaviors (path patterns) that you want to receive real-time logs for.

CloudFront real-time logs are delivered to the data stream of your choice in Amazon Kinesis Data Streams. You can build your own Kinesis data stream consumer, or use Amazon Data Firehose to send the log data to Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon OpenSearch Service (OpenSearch Service), or a third-party log processing service.

CloudFront charges for real-time logs, in addition to the charges you incur for using Kinesis Data Streams. For more information about pricing, see Amazon CloudFront Pricing and Amazon Kinesis Data Streams pricing.

> **Important**
>
> We recommend that you use the logs to understand the nature of the requests for your content, not as a complete accounting of all requests. CloudFront delivers real-time logs on a best-effort basis. The log entry for a particular request might be delivered long after the request was actually processed and, in rare cases, a log entry might not be delivered at all. When a log entry is omitted from real-time logs, the number of entries in the real-time logs won't match the usage that appears in the Amazon billing and usage reports.

Understanding real-time log configurations

To use CloudFront real-time logs, you start by creating a real-time log configuration. The real-time log configuration contains information about which log fields you want to receive, the *sampling rate* for log records, and the Kinesis data stream where you want to deliver the logs.

Specifically, a real-time log configuration contains the following settings:

- Name
- Sampling rate
- Fields
- Endpoint (Kinesis data stream)
- IAM role

Name

A name to identify the real-time log configuration.

Sampling rate

The sampling rate is a whole number between 1 and 100 (inclusive) that determines the percentage of viewer requests that are sent to Kinesis Data Streams as real-time log records. To include every viewer request in your real-time logs, specify 100 for the sampling rate. You might choose a lower sampling rate to reduce costs while still receiving a representative sample of request data in your real-time logs.

Fields

A list of the fields that are included in each real-time log record. Each log record can contain up to 40 fields, and you can choose to receive all of the available fields, or only the fields that you need for monitoring and analyzing performance.

The following list contains each field name and a description of the information in that field. The fields are listed in the order in which they appear in the log records that are delivered to Kinesis Data Streams.

Fields 46-63 are common media client data (CMCD) that media player clients can send to CDNs with each request. You can use this data to understand each request, such as the media type (audio, video), playback rate, and streaming length. These fields will only appear in your real-time logs if they're sent to CloudFront.

1. **`timestamp`**

The date and time at which the edge server finished responding to the request.

2. **`c-ip`**

The IP address of the viewer that made the request, for example, `192.0.2.183` or `2001:0db8:85a3::8a2e:0370:7334`. If the viewer used an HTTP proxy or a load balancer to send the request, the value of this field is the IP address of the proxy or load balancer. See also the `x-forwarded-for` field.

3. **`time-to-first-byte`**

   The number of seconds between receiving the request and writing the first byte of the response, as measured on the server.

4. **`sc-status`**

   The HTTP status code of the server's response (for example, `200`).

5. **`sc-bytes`**

   The total number of bytes that the server sent to the viewer in response to the request, including headers. For WebSocket connections, this is the total number of bytes sent from the server to the client through the connection.

6. **`cs-method`**

   The HTTP request method received from the viewer.

7. **`cs-protocol`**

   The protocol of the viewer request (`http`, `https`, `ws`, or `wss`).

8. **`cs-host`**

   The value that the viewer included in the `Host` header of the request. If you're using the CloudFront domain name in your object URLs (such as d111111abcdef8.cloudfront.net), this field contains that domain name. If you're using alternate domain names (CNAMEs) in your object URLs (such as www.example.com), this field contains the alternate domain name.

9. **`cs-uri-stem`**

   The entire request URL, including the query string (if one exists), but without the domain name. For example, `/images/cat.jpg?mobile=true`.

> **Note**
>
> In standard logs, the `cs-uri-stem` value doesn't include the query string.

10**cs-bytes**

The total number of bytes of data that the viewer included in the request, including headers. For WebSocket connections, this is the total number of bytes sent from the client to the server on the connection.

11**x-edge-location**

The edge location that served the request. Each edge location is identified by a three-letter code and an arbitrarily assigned number (for example, DFW3). The three-letter code typically corresponds with the International Air Transport Association (IATA) airport code for an airport near the edge location's geographic location. (These abbreviations might change in the future.)

12**x-edge-request-id**

An opaque string that uniquely identifies a request. CloudFront also sends this string in the `x-amz-cf-id` response header.

13**x-host-header**

The domain name of the CloudFront distribution (for example, d111111abcdef8.cloudfront.net).

14**time-taken**

The number of seconds (to the thousandth of a second, for example, 0.082) from when the server receives the viewer's request to when the server writes the last byte of the response to the output queue, as measured on the server. From the perspective of the viewer, the total time to get the full response will be longer than this value because of network latency and TCP buffering.

15**cs-protocol-version**

The HTTP version that the viewer specified in the request. Possible values include `HTTP/0.9`, `HTTP/1.0`, `HTTP/1.1`, `HTTP/2.0`, and `HTTP/3.0`.

16**c-ip-version**

The IP version of the request (IPv4 or IPv6).

17**cs-user-agent**

The value of the `User-Agent` header in the request. The `User-Agent` header identifies the source of the request, such as the type of device and browser that submitted the request or, if the request came from a search engine, which search engine.

18 **`cs-referer`**

The value of the `Referer` header in the request. This is the name of the domain that originated the request. Common referrers include search engines, other websites that link directly to your objects, and your own website.

19 **`cs-cookie`**

The `Cookie` header in the request, including name—value pairs and the associated attributes.

> **Note**
>
> This field is truncated to 800 bytes.

20 **`cs-uri-query`**

The query string portion of the request URL, if any.

21 **`x-edge-response-result-type`**

How the server classified the response just before returning the response to the viewer. See also the `x-edge-result-type` field. Possible values include:

- `Hit` – The server served the object to the viewer from the cache.
- `RefreshHit` – The server found the object in the cache but the object had expired, so the server contacted the origin to verify that the cache had the latest version of the object.
- `Miss` – The request could not be satisfied by an object in the cache, so the server forwarded the request to the origin server and returned the result to the viewer.
- `LimitExceeded` – The request was denied because a CloudFront quota (formerly referred to as a limit) was exceeded.
- `CapacityExceeded` – The server returned a 503 error because it didn't have enough capacity at the time of the request to serve the object.
- `Error` – Typically, this means the request resulted in a client error (the value of the `sc-status` field is in the `4xx` range) or a server error (the value of the `sc-status` field is in the `5xx` range).

  If the value of the `x-edge-result-type` field is `Error` and the value of this field is not `Error`, the client disconnected before finishing the download.
- `Redirect` – The server redirected the viewer from HTTP to HTTPS according to the distribution settings.

22**x-forwarded-for**

If the viewer used an HTTP proxy or a load balancer to send the request, the value of the `c-ip` field is the IP address of the proxy or load balancer. In that case, this field is the IP address of the viewer that originated the request. This field can contain multiple comma-separated IP addresses. Each IP address can be an IPv4 address (for example, `192.0.2.183`) or an IPv6 address (for example, `2001:0db8:85a3::8a2e:0370:7334`).

23**ssl-protocol**

When the request used HTTPS, this field contains the SSL/TLS protocol that the viewer and server negotiated for transmitting the request and response. For a list of possible values, see the supported SSL/TLS protocols in Supported protocols and ciphers between viewers and CloudFront.

24**ssl-cipher**

When the request used HTTPS, this field contains the SSL/TLS cipher that the viewer and server negotiated for encrypting the request and response. For a list of possible values, see the supported SSL/TLS ciphers in Supported protocols and ciphers between viewers and CloudFront.

25**x-edge-result-type**

How the server classified the response after the last byte left the server. In some cases, the result type can change between the time that the server is ready to send the response and the time that it finishes sending the response. See also the `x-edge-response-result-type` field.

For example, in HTTP streaming, suppose the server finds a segment of the stream in the cache. In that scenario, the value of this field would ordinarily be `Hit`. However, if the viewer closes the connection before the server has delivered the entire segment, the final result type (and the value of this field) is `Error`.

WebSocket connections will have a value of `Miss` for this field because the content is not cacheable and is proxied directly to the origin.

Possible values include:

- `Hit` – The server served the object to the viewer from the cache.
- `RefreshHit` – The server found the object in the cache but the object had expired, so the server contacted the origin to verify that the cache had the latest version of the object.

- `Miss` – The request could not be satisfied by an object in the cache, so the server forwarded the request to the origin and returned the result to the viewer.
- `LimitExceeded` – The request was denied because a CloudFront quota (formerly referred to as a limit) was exceeded.
- `CapacityExceeded` – The server returned an HTTP 503 status code because it didn't have enough capacity at the time of the request to serve the object.
- `Error` – Typically, this means the request resulted in a client error (the value of the `sc-status` field is in the `4xx` range) or a server error (the value of the `sc-status` field is in the `5xx` range). If the value of the `sc-status` field is `200`, or if the value of this field is `Error` and the value of the `x-edge-response-result-type` field is not `Error`, it means the HTTP request was successful but the client disconnected before receiving all of the bytes.
- `Redirect` – The server redirected the viewer from HTTP to HTTPS according to the distribution settings.

26. **`fle-encrypted-fields`**

    The number of field-level encryption fields that the server encrypted and forwarded to the origin. CloudFront servers stream the processed request to the origin as they encrypt data, so this field can have a value even if the value of `fle-status` is an error.

27. **`fle-status`**

    When field-level encryption is configured for a distribution, this field contains a code that indicates whether the request body was successfully processed. When the server successfully processes the request body, encrypts values in the specified fields, and forwards the request to the origin, the value of this field is `Processed`. The value of `x-edge-result-type` can still indicate a client-side or server-side error in this case.

    Possible values for this field include:

    - `ForwardedByContentType` – The server forwarded the request to the origin without parsing or encryption because no content type was configured.
    - `ForwardedByQueryArgs` – The server forwarded the request to the origin without parsing or encryption because the request contains a query argument that wasn't in the configuration for field-level encryption.
    - `ForwardedDueToNoProfile` – The server forwarded the request to the origin without parsing or encryption because no profile was specified in the configuration for field-level encryption.

- `MalformedContentTypeClientError` – The server rejected the request and returned an HTTP 400 status code to the viewer because the value of the `Content-Type` header was in an invalid format.
- `MalformedInputClientError` – The server rejected the request and returned an HTTP 400 status code to the viewer because the request body was in an invalid format.
- `MalformedQueryArgsClientError` – The server rejected the request and returned an HTTP 400 status code to the viewer because a query argument was empty or in an invalid format.
- `RejectedByContentType` – The server rejected the request and returned an HTTP 400 status code to the viewer because no content type was specified in the configuration for field-level encryption.
- `RejectedByQueryArgs` – The server rejected the request and returned an HTTP 400 status code to the viewer because no query argument was specified in the configuration for field-level encryption.
- `ServerError` – The origin server returned an error.

If the request exceeds a field-level encryption quota (formerly referred to as a limit), this field contains one of the following error codes, and the server returns HTTP status code 400 to the viewer. For a list of the current quotas on field-level encryption, see Quotas on field-level encryption.

- `FieldLengthLimitClientError` – A field that is configured to be encrypted exceeded the maximum length allowed.
- `FieldNumberLimitClientError` – A request that the distribution is configured to encrypt contains more than the number of fields allowed.
- `RequestLengthLimitClientError` – The length of the request body exceeded the maximum length allowed when field-level encryption is configured.

28 **`sc-content-type`**

The value of the HTTP `Content-Type` header of the response.

29 **`sc-content-len`**

The value of the HTTP `Content-Length` header of the response.

30 **`sc-range-start`**

When the response contains the HTTP `Content-Range` header, this field contains the range start value.

31**sc-range-end**

When the response contains the HTTP `Content-Range` header, this field contains the range end value.

32**c-port**

The port number of the request from the viewer.

33**x-edge-detailed-result-type**

This field contains the same value as the `x-edge-result-type` field, except in the following cases:

- When the object was served to the viewer from the Origin Shield layer, this field contains `OriginShieldHit`.
- When the object was not in the CloudFront cache and the response was generated by an origin request Lambda@Edge function, this field contains `MissGeneratedResponse`.
- When the value of the `x-edge-result-type` field is `Error`, this field contains one of the following values with more information about the error:
  - `AbortedOrigin` – The server encountered an issue with the origin.
  - `ClientCommError` – The response to the viewer was interrupted due to a communication problem between the server and the viewer.
  - `ClientGeoBlocked` – The distribution is configured to refuse requests from the viewer's geographic location.
  - `ClientHungUpRequest` – The viewer stopped prematurely while sending the request.
  - `Error` – An error occurred for which the error type doesn't fit any of the other categories. This error type can occur when the server serves an error response from the cache.
  - `InvalidRequest` – The server received an invalid request from the viewer.
  - `InvalidRequestBlocked` – Access to the requested resource is blocked.
  - `InvalidRequestCertificate` – The distribution doesn't match the SSL/TLS certificate for which the HTTPS connection was established.
  - `InvalidRequestHeader` – The request contained an invalid header.
  - `InvalidRequestMethod` – The distribution is not configured to handle the HTTP request method that was used. This can happen when the distribution supports only cacheable requests.

- `OriginCommError` – The request timed out while connecting to the origin, or reading data from the origin.
- `OriginConnectError` – The server couldn't connect to the origin.
- `OriginContentRangeLengthError` – The `Content-Length` header in the origin's response doesn't match the length in the `Content-Range` header.
- `OriginDnsError` – The server couldn't resolve the origin's domain name.
- `OriginError` – The origin returned an incorrect response.
- `OriginHeaderTooBigError` – A header returned by the origin is too big for the edge server to process.
- `OriginInvalidResponseError` – The origin returned an invalid response.
- `OriginReadError` – The server couldn't read from the origin.
- `OriginWriteError` – The server couldn't write to the origin.
- `OriginZeroSizeObjectError` – A zero size object sent from the origin resulted in an error.
- `SlowReaderOriginError` – The viewer was slow to read the message that caused the origin error.

34**`c-country`**

A country code that represents the viewer's geographic location, as determined by the viewer's IP address. For a list of country codes, see ISO 3166-1 alpha-2.

35**`cs-accept-encoding`**

The value of the `Accept-Encoding` header in the viewer request.

36**`cs-accept`**

The value of the `Accept` header in the viewer request.

37**`cache-behavior-path-pattern`**

The path pattern that identifies the cache behavior that matched the viewer request.

38**`cs-headers`**

The HTTP headers (names and values) in the viewer request.

> **Note**
>
> This field is truncated to 800 bytes.

39 **`cs-header-names`**

The names of the HTTP headers (not values) in the viewer request.

> **Note**
>
> This field is truncated to 800 bytes.

40 **`cs-headers-count`**

The number of HTTP headers in the viewer request.

41 **`origin-fbl`**

The number of seconds of first-byte latency between CloudFront and your origin.

42 **`origin-lbl`**

The number of seconds of last-byte latency between CloudFront and your origin.

43 **`asn`**

The autonomous system number (ASN) of the viewer.

44 **`primary-distribution-id`**

When continuous deployment is enabled, this ID identifies which distribution is the primary in the current distribution.

45 **`primary-distribution-dns-name`**

When continuous deployment is enabled, this value shows the primary domain name that is related to the current CloudFront distribution (for example, d111111abcdef8.cloudfront.net).

> **CMCD fields in real-time logs**
>
> For more information about these fields, see the CTA Specification Web Application Video Ecosystem - Common Media Client Data CTA-5004 document.

46**cmcd-encoded-bitrate**

The encoded bitrate of the requested audio or video object.

47**cmcd-buffer-length**

The buffer length of the requested media object.

48**cmcd-buffer-starvation**

Whether the buffer was starved at some point between the prior request and the object request. This can cause the player to be in a rebuffering stat, which can stall the video or audio playback.

49**cmcd-content-id**

A unique string that identifies the current content.

50**cmcd-object-duration**

The playback duration of the requested object (in milliseconds).

51**cmcd-deadline**

The deadline from the request time that the first sample of this object must be available, so that a buffer underrun state or other playback problems are avoided.

52**cmcd-measured-throughput**

The throughput between the client and server, as measured by the client.

53**cmcd-next-object-request**

The relative path of the next requested object.

54**cmcd-next-range-request**

If the next request is a partial object request, this string denotes the byte range to be requested.

55**cmcd-object-type**

The media type of the current object being requested.

56**cmcd-playback-rate**

1 if real-time, 2 if double-speed, 0 if not playing.

57**cmcd-requested-maximum-throughput**

The requested maximum throughput that the client considers sufficient for asset delivery.

58**cmcd-streaming-format**

The streaming format that defines the current request.

59**cmcd-session-id**

A GUID identifying the current playback session.

60**cmcd-stream-type**

Token identifying segment availability. `v` = all segments are available. `l` = segments become available over time.

61**cmcd-startup**

Key is included without a value if the object is needed urgently during startup, seeking, or recovery after a buffer-empty event.

62**cmcd-top-bitrate**

The highest bitrate rendition that the client can play.

63**cmcd-version**

The version of this specification used for interpreting the defined key names and values. If this key is omitted, the client and server *must* interpret the values as being defined by version 1.

Endpoint (Kinesis data stream)

The endpoint contains information about the Kinesis data stream where you want to send real-time logs. You provide the Amazon Resource Name (ARN) of the data stream.

For more information about creating a Kinesis data stream, see the following topics in the *Amazon Kinesis Data Streams Developer Guide*.

- Managing Streams Using the Console
- Perform Basic Kinesis Data Stream Operations Using the Amazon CLI
- Creating a Stream (uses the Amazon SDK for Java)

When you create a data stream, you need to specify the number of shards. Use the following information to help you estimate the number of shards you need.

To estimate the number of shards for your Kinesis data stream

1. Calculate (or estimate) the number of requests per second that your CloudFront distribution receives.

   You can use the CloudFront usage reports (in the CloudFront console) and the CloudFront metrics (in the CloudFront and Amazon CloudWatch consoles) to help you calculate your requests per second.

2. Determine the typical size of a single real-time log record.

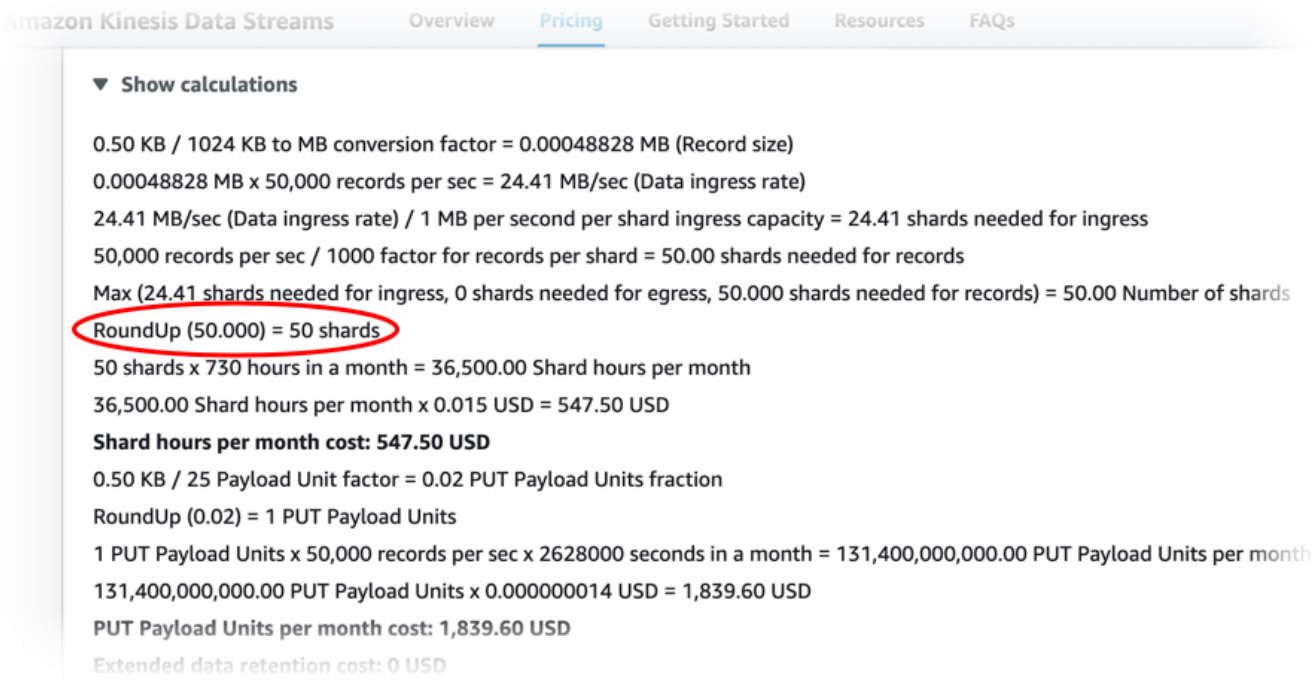
   In general, a single log record is around 500 bytes. A large record that includes all available fields is generally around 1 KB.

   If you're not sure what your log record size is, you can enable real-time logs with a low sampling rate (for example, 1%), and then calculate the average record size using monitoring data in Kinesis Data Streams (total incoming bytes divided by total number of records).

3. In the Pricing calculator on the Amazon Kinesis Data Streams pricing page, enter the number of requests (records) per second, and the average record size of a single log record. Then choose **Show calculations**.

   The pricing calculator shows you the number of shards you need. (It also shows you the estimated cost.)

   The following example shows that for an average record size of 0.5 KB, and 50,000 requests per second, you need 50 shards.



IAM role

The Amazon Identity and Access Management (IAM) role that gives CloudFront permission to deliver real-time logs to your Kinesis data stream.

When you create a real-time log configuration with the CloudFront console, you can choose **Create new service role** to let the console create the IAM role for you.

When you create a real-time log configuration with Amazon CloudFormation or the CloudFront API (Amazon CLI or SDK), you must create the IAM role yourself and provide the role ARN. To create the IAM role yourself, use the following policies.

IAM role trust policy

To use the following IAM role trust policy, replace *111122223333* with your Amazon Web Services account number. The `Condition` element in this policy helps to prevent the confused deputy problem because CloudFront can only assume this role on behalf of a distribution in your Amazon Web Services account.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                }
            }
        }
    ]
}
```

IAM role permissions policy for an unencrypted data stream

To use the following policy, replace *arn:aws:kinesis:us-west-2:123456789012:stream/StreamName* with the ARN of your Kinesis data stream.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamSummary",
                "kinesis:DescribeStream",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:us-west-2:123456789012:stream/StreamName"
            ]
        }
    ]
}
```

IAM role permissions policy for an encrypted data stream

To use the following policy, replace *arn:aws:kinesis:us-west-2:123456789012:stream/StreamName* with the ARN of your Kinesis data stream

and *arn:aws:kms:us-west-2:123456789012:key/e58a3d0b-fe4f-4047-a495-ae03cc73d486* with the ARN of your Amazon KMS key.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamSummary",
                "kinesis:DescribeStream",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:us-west-2:123456789012:stream/StreamName"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-west-2:123456789012:key/e58a3d0b-fe4f-4047-a495-
ae03cc73d486"
            ]
        }
    ]
}
```

Creating and using real-time log configurations

You can use a real-time log configurations to get information about requests made to a distribution in real time (logs are delivered within seconds of receiving the requests). You can create a real-time log configuration in the CloudFront console, with the Amazon Command Line Interface (Amazon CLI), or with the CloudFront API.

To use a real-time log configuration, you attach it to one or more cache behaviors in a CloudFront distribution.

Create a real-time log configuration (console)

To create a real-time log configuration

1. Sign in to the Amazon Web Services Management Console and open the **Logs** page in the CloudFront console at https://console.amazonaws.cn/cloudfront/v4/home?#/logs.
2. Choose the **Real-time configurations** tab.
3. Choose **Create configuration**.
4. For **Name**, enter a name for the configuration.
5. For **Sampling rate**, enter the percentage of requests for which you want to receive log records.
6. For **Fields**, choose the fields to receive in the real-time logs.
   - To include all CMCD fields for your logs, choose **CMCD all keys**.
7. For **Endpoint**, choose one or more Kinesis data streams to receive real-time logs.

   > **Note**
   >
   > CloudFront real-time logs are delivered to the data stream that you specify in Kinesis Data Streams. To read and analyze your real-time logs, you can build your own Kinesis data stream consumer. You can also use Firehose to send the log data to Amazon S3, Amazon Redshift, Amazon OpenSearch Service, or a third-party log processing service.

8. For **IAM role**, choose **Create new service role** or choose an existing role. You must have permission to create IAM roles.
9. (Optional) For **Distribution**, choose a CloudFront distribution and cache behavior to attach to the real-time log configuration.
10. Choose **Create configuration**.

If successful, the console shows the details of the real-time log configuration that you just created.

For more information, see Understanding real-time log configurations.

Create a real-time log configuration (Amazon CLI)

To create a real-time log configuration with the Amazon Command Line Interface (Amazon CLI), use the **aws cloudfront create-realtime-log-config** command. You can use an input file to provide the command's input parameters, rather than specifying each individual parameter as command line input.

To create a real-time log configuration (CLI with input file)

1. Use the following command to create a file named `rtl-config.yaml` that contains all of the input parameters for the **create-realtime-log-config** command.

```
aws cloudfront create-realtime-log-config --generate-cli-skeleton yaml-input > rtl-config.yaml
```

2. Open the file named `rtl-config.yaml` that you just created. Edit the file to specify the real-time log configuration settings that you want, then save the file. Note the following:

   - For `StreamType`, the only valid value is `Kinesis`.

   For more information about the real-time long configuration settings, see Understanding real-time log configurations.

3. Use the following command to create the real-time log configuration using input parameters from the `rtl-config.yaml` file.

```
aws cloudfront create-realtime-log-config --cli-input-yaml file://rtl-config.yaml
```

If successful, the command's output shows the details of the real-time log configuration that you just created.

To attach a real-time log configuration to an existing distribution (CLI with input file)

1. Use the following command to save the distribution configuration for the CloudFront distribution that you want to update. Replace *`distribution_ID`* with the distribution's ID.

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml > dist-config.yaml
```

2. Open the file named `dist-config.yaml` that you just created. Edit the file, making the following changes to each cache behavior that you are updating to use a real-time log configuration.

- In the cache behavior, add a field named `RealtimeLogConfigArn`. For the field's value, use the ARN of the real-time log configuration that you want to attach to this cache behavior.
- Rename the `ETag` field to `IfMatch`, but don't change the field's value.

Save the file when finished.

3. Use the following command to update the distribution to use the real-time log configuration. Replace *`distribution_ID`* with the distribution's ID.

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

If successful, the command's output shows the details of the distribution that you just updated.

Create a real-time log configuration (API)

To create a real-time log configuration with the CloudFront API, use CreateRealtimeLogConfig. For more information about the parameters that you specify in this API call, see Understanding real-time log configurations and the API reference documentation for your Amazon SDK or other API client.

After you create a real-time log configuration, you can attach it to a cache behavior, using one of the following API calls:

- To attach it to a cache behavior in an existing distribution, use UpdateDistribution.
- To attach it to a cache behavior in a new distribution, use CreateDistribution.

For both of these API calls, provide the ARN of the real-time log configuration in the `RealtimeLogConfigArn` field, inside a cache behavior. For more information about the other fields that you specify in these API calls, see Values that you specify when you create or update a distribution and the API reference documentation for your Amazon SDK or other API client.

Creating a Kinesis Data Streams consumer

To read and analyze your real-time logs, you build or use a Kinesis Data Streams *consumer*. When you build a consumer for CloudFront real-time logs, it's important to know that the fields in every

real-time log record are always delivered in the same order, as listed in the Fields section. Make sure that you build your consumer to accommodate this fixed order.

For example, consider a real-time log configuration that includes only these three fields: `time-to-first-byte`, `sc-status`, and `c-country`. In this scenario, the last field, `c-country`, is always field number 3 in every log record. However, if you later add fields to the real-time log configuration, the placement of each field in a record can change.

For example, if you add the fields `sc-bytes` and `time-taken` to the real-time log configuration, these fields are inserted into each log record according to the order shown in the Fields section. The resulting order of all five fields is `time-to-first-byte`, `sc-status`, `sc-bytes`, `time-taken`, and `c-country`. The `c-country` field was originally field number 3, but is now field number 5. Make sure that your consumer application can handle fields that change position in a log record, in case you add fields to your real-time log configuration.

Troubleshooting real-time logs

After you create a real-time log configuration, you might find that no records (or not all records) are delivered to Kinesis Data Streams. In this case, you should first verify that your CloudFront distribution is receiving viewer requests. If it is, you can check the following setting to continue troubleshooting.

IAM role permissions

To deliver real-time log records to your Kinesis data stream, CloudFront uses the IAM role in the real-time log configuration. Make sure that the role trust policy and the role permissions policy match the policies shown in IAM role.

Kinesis Data Streams throttling

If CloudFront writes real-time log records to your Kinesis data stream faster than the stream can handle, Kinesis Data Streams might throttle the requests from CloudFront. In this case, you can increase the number of shards in your Kinesis data stream. Each shard can support writes up to 1,000 records per second, up to a maximum data write of 1 MB per second.

Edge function logs

You can use Amazon CloudWatch Logs to get logs for your edge functions, both Lambda@Edge and CloudFront Functions. Access the logs using the CloudWatch console or the CloudWatch Logs API.

Important

We recommend that you use the logs to understand the nature of the requests for your content, not as a complete accounting of all requests. CloudFront delivers edge function logs on a best-effort basis. The log entry for a particular request might be delivered long after the request was actually processed and, in rare cases, a log entry might not be delivered at all. When a log entry is omitted from edge function logs, the number of entries in the edge function logs won't match the usage that appears in the Amazon billing and usage reports.

Lambda@Edge logs

Lambda@Edge automatically sends function logs to CloudWatch Logs, creating log streams in the Amazon Web Services Regions where the functions are run. The log group name is formatted as `/aws/lambda/us-east-1.function-name`, where *`function-name`* is the name that you gave to the function when you created it, and *`us-east-1`* is the Region code for the Amazon Web Services Region where the function ran.

Note

Lambda@Edge throttles logs based on the request volume and the size of logs.

You must review CloudWatch log files in the correct Amazon Web Services Region to see your Lambda@Edge function log files. To see the Regions where your Lambda@Edge function is running, view graphs of metrics for the function in the CloudFront console. Metrics are displayed for each Amazon Web Services Region. On the same page, you can choose a Region and then view log files for that Region to investigate issues.

To learn more about how to use CloudWatch Logs with Lambda@Edge functions, see the following:

- For more information about viewing graphs in the **Monitoring** section of the CloudFront console, see the section called “Monitoring CloudFront metrics with Amazon CloudWatch”.
- For information about the permissions required to send data to CloudWatch Logs, see the section called “Setting IAM permissions and roles”.

- For information about adding logging to a Lambda@Edge function, see Amazon Lambda function logging in Node.js or Amazon Lambda function logging in Python in the *Amazon Lambda Developer Guide*.
- For information about CloudWatch Logs quotas (formerly known as limits), see CloudWatch Logs quotas in the *Amazon CloudWatch Logs User Guide*.

CloudFront Functions logs

If a CloudFront function's code contains `console.log()` statements, CloudFront Functions automatically sends these log lines to CloudWatch Logs. If there are no `console.log()` statements, nothing is sent to CloudWatch Logs.

CloudFront Functions always creates log streams in the US East (N. Virginia) Region (`us-east-1`), no matter which edge location ran the function. The log group name is in the format `/aws/cloudfront/function/`*`FunctionName`*, where *`FunctionName`* is the name that you gave to the function when you created it. The log stream name is in the format `YYYY/M/D/`*`UUID`*.

The following shows an example log message sent to CloudWatch Logs. Each line begins with an ID that uniquely identifies a CloudFront request. The message begins with a `START` line that includes the CloudFront distribution ID, and ends with an `END` line. Between the `START` and `END` lines are the log lines generated by `console.log()` statements in the function.

```
U7b4hR_RaxMADupvKAvr8_m9gsGXvioUggLV5Oyq-vmAtH8HADpjhw== START DistributionID:
 E3E5D42GADAXZZ
U7b4hR_RaxMADupvKAvr8_m9gsGXvioUggLV5Oyq-vmAtH8HADpjhw== Example function log output
U7b4hR_RaxMADupvKAvr8_m9gsGXvioUggLV5Oyq-vmAtH8HADpjhw== END
```

> **Note**
>
> CloudFront Functions sends logs to CloudWatch only for functions in the `LIVE` stage that run in response to production requests and responses. When you test a function, CloudFront doesn't send any logs to CloudWatch. The test output contains information about errors, compute utilization, and function logs (`console.log()` statements), but this information is not sent to CloudWatch.

CloudFront Functions uses an Amazon Identity and Access Management (IAM) service-linked role to send logs to CloudWatch Logs in your account. A service-linked role is an IAM role that is

linked directly to an Amazon service. Service-linked roles are predefined by the service and include all of the permissions that the service requires to call other Amazon services on your behalf. CloudFront Functions uses a service-linked role called **AWSServiceRoleForCloudFrontLogger**. For more information about this role, see the section called "Service-linked roles for Lambda@Edge" (Lambda@Edge uses the same service-linked role).

When a function fails with a validation error or an execution error, information is logged in CloudFront's standard logs and real-time logs. Information about the error is logged in the `x-edge-result-type`, `x-edge-response-result-type`, and `x-edge-detailed-result-type` fields.

Logging Amazon CloudFront API calls using Amazon CloudTrail

CloudFront is integrated with Amazon CloudTrail, a service that provides a record of actions taken by a user, role, or an Amazon Web Service. CloudTrail captures all API calls for CloudFront as events. The calls captured include calls from the CloudFront console and code calls to the CloudFront API operations. Using the information collected by CloudTrail, you can determine the request that was made to CloudFront, the IP address from which the request was made, when it was made, and additional details.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or user credentials.
- Whether the request was made on behalf of an IAM Identity Center user.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another Amazon Web Service.

CloudTrail is active in your Amazon Web Services account when you create the account and you automatically have access to the CloudTrail **Event history**. The CloudTrail **Event history** provides a viewable, searchable, downloadable, and immutable record of the past 90 days of recorded management events in an Amazon Web Services Region. For more information, see Working with CloudTrail Event history in the *Amazon CloudTrail User Guide*. There are no CloudTrail charges for viewing the **Event history**.

For an ongoing record of events in your Amazon Web Services account past 90 days, create a trail or a CloudTrail Lake event data store.

CloudTrail trails

A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. All trails created using the Amazon Web Services Management Console are multi-Region. You can create a single-Region or a multi-Region trail by using the Amazon CLI. Creating a multi-Region trail is recommended because you capture activity in all Amazon Web Services Regions in your account. If you create a single-Region trail, you can view only the events logged in the trail's Amazon Web Services Region. For more information about trails, see Creating a trail for your Amazon Web Services account and Creating a trail for an organization in the *Amazon CloudTrail User Guide*.

You can deliver one copy of your ongoing management events to your Amazon S3 bucket at no charge from CloudTrail by creating a trail, however, there are Amazon S3 storage charges. For more information about CloudTrail pricing, see Amazon CloudTrail Pricing. For information about Amazon S3 pricing, see Amazon S3 Pricing.

CloudTrail Lake event data stores

CloudTrail Lake lets you run SQL-based queries on your events. CloudTrail Lake converts existing events in row-based JSON format to Apache ORC format. ORC is a columnar storage format that is optimized for fast retrieval of data. Events are aggregated into *event data stores*, which are immutable collections of events based on criteria that you select by applying advanced event selectors. The selectors that you apply to an event data store control which events persist and are available for you to query. For more information about CloudTrail Lake, see Working with Amazon CloudTrail Lake in the *Amazon CloudTrail User Guide*.

CloudTrail Lake event data stores and queries incur costs. When you create an event data store, you choose the pricing option you want to use for the event data store. The pricing option determines the cost for ingesting and storing events, and the default and maximum retention period for the event data store. For more information about CloudTrail pricing, see Amazon CloudTrail Pricing.

Note

CloudFront is a global service. CloudTrail records events for CloudFront in the US East (N. Virginia) Region. For more information, see Global service events in the *Amazon CloudTrail User Guide*.
If you use temporary security credentials by using Amazon Security Token Service, calls to regional endpoints, such as `us-west-2`, are logged in CloudTrail to their appropriate Region.

For more information about CloudFront endpoints, see CloudFront endpoints and quotas in the *Amazon Web Services General Reference*.

CloudFront data events in CloudTrail

Data events provide information about the resource operations performed on or in a resource (for example, reading or writing to a CloudFront distribution). These are also known as data plane operations. Data events are often high-volume activities. By default, CloudTrail doesn't log data events. The CloudTrail **Event history** doesn't record data events.

Additional charges apply for data events. For more information about CloudTrail pricing, see Amazon CloudTrail Pricing.

You can log data events for the CloudFront resource types by using the CloudTrail console, Amazon CLI, or CloudTrail API operations. For more information about how to log data events, see Logging data events with the Amazon Web Services Management Console and Logging data events with the Amazon Command Line Interface in the *Amazon CloudTrail User Guide*.

The following table lists the CloudFront resource types for which you can log data events. The **Data event type (console)** column shows the value to choose from the **Data event type** list on the CloudTrail console. The **resources.type value** column shows the `resources.type` value, which you would specify when configuring advanced event selectors using the Amazon CLI or CloudTrail APIs. The **Data APIs logged to CloudTrail** column shows the API calls logged to CloudTrail for the resource type.

Data event type (console)	resources.type value	Data APIs logged to CloudTrail
CloudFront KeyValueStore	`AWS::CloudFront::KeyValueStore`	• DeleteKeys • DescribeKeyValueStore • GetKey • ListKeys • PutKeys • UpdateKeys

You can configure advanced event selectors to filter on the `eventName`, `readOnly`, and `resources.ARN` fields to log only those events that are important to you. For more information about these fields, see AdvancedFieldSelector in the *Amazon CloudTrail API Reference*.

CloudFront management events in CloudTrail

Management events provide information about management operations that are performed on resources in your Amazon Web Services account. These are also known as control plane operations. By default, CloudTrail logs management events.

Amazon CloudFront logs all CloudFront control plane operations as management events. For a list of the Amazon CloudFront control plane operations that CloudFront logs to CloudTrail, see the Amazon CloudFront API Reference.

CloudFront event examples

An event represents a single request from any source and includes information about the requested API operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so events don't appear in any specific order.

Contents

- Example: UpdateDistribution
- Example: UpdateKeys

Example: UpdateDistribution

The following example shows a CloudTrail event that demonstrates the UpdateDistribution operation.

For calls to the CloudFront API, the `eventSource` is `cloudfront.amazonaws.com`.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:role-session-name",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/role-session-name",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
```

```
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-02-02T19:23:50Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2024-02-02T19:26:01Z",
    "eventSource": "cloudfront.amazonaws.com",
    "eventName": "UpdateDistribution",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "52.94.133.137",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/121.0.0.0 Safari/537.36",
    "requestParameters": {
        "distributionConfig": {
            "defaultRootObject": "",
            "aliases": {
                "quantity": 3,
                "items": [
                    "alejandro_rosalez.awsps.myinstance.com",
                    "cross-testing.alejandro_rosalez.awsps.myinstance.com",
                    "*.alejandro_rosalez.awsps.myinstance.com"
                ]
            },
            "cacheBehaviors": {
                "quantity": 0,
                "items": []
            },
            "httpVersion": "http2and3",
            "originGroups": {
                "quantity": 0,
                "items": []
            },
            "viewerCertificate": {
                "minimumProtocolVersion": "TLSv1.2_2021",
                "cloudFrontDefaultCertificate": false,
```

```
                "aCMCertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                "sSLSupportMethod": "sni-only"
            },
            "webACLId": "arn:aws:wafv2:us-east-1:111122223333:global/webacl/testing-
acl/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "customErrorResponses": {
                "quantity": 0,
                "items": []
            },
            "logging": {
                "includeCookies": false,
                "prefix": "",
                "enabled": false,
                "bucket": ""
            },
            "priceClass": "PriceClass_All",
            "restrictions": {
                "geoRestriction": {
                    "restrictionType": "none",
                    "quantity": 0,
                    "items": []
                }
            },
            "isIPV6Enabled": true,
            "callerReference": "1578329170895",
            "continuousDeploymentPolicyId": "",
            "enabled": true,
            "defaultCacheBehavior": {
                "targetOriginId": "d111111abcdef8",
                "minTTL": 0,
                "compress": false,
                "maxTTL": 31536000,
                "functionAssociations": {
                    "quantity": 0,
                    "items": []
                },
                "trustedKeyGroups": {
                    "quantity": 0,
                    "items": [],
                    "enabled": false
                },
                "smoothStreaming": false,
                "fieldLevelEncryptionId": "",
```

```
                "defaultTTL": 86400,
                "lambdaFunctionAssociations": {
                    "quantity": 0,
                    "items": []
                },
                "viewerProtocolPolicy": "redirect-to-https",
                "forwardedValues": {
                    "cookies": {"forward": "none"},
                    "queryStringCacheKeys": {
                        "quantity": 0,
                        "items": []
                    },
                    "queryString": false,
                    "headers": {
                        "quantity": 1,
                        "items": ["*"]
                    }
                },
                "trustedSigners": {
                    "items": [],
                    "enabled": false,
                    "quantity": 0
                },
                "allowedMethods": {
                    "quantity": 2,
                    "items": [
                        "HEAD",
                        "GET"
                    ],
                    "cachedMethods": {
                        "quantity": 2,
                        "items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                }
            },
            "staging": false,
            "origins": {
                "quantity": 1,
                "items": [
                    {
                        "originPath": "",
```

```
                        "connectionTimeout": 10,
                        "customOriginConfig": {
                            "originReadTimeout": 30,
                            "hTTPSPort": 443,
                            "originProtocolPolicy": "https-only",
                            "originKeepaliveTimeout": 5,
                            "hTTPPort": 80,
                            "originSslProtocols": {
                                "quantity": 3,
                                "items": [
                                    "TLSv1",
                                    "TLSv1.1",
                                    "TLSv1.2"
                                ]
                            }
                        },
                        "id": "d111111abcdef8",
                        "domainName": "d111111abcdef8.cloudfront.net",
                        "connectionAttempts": 3,
                        "customHeaders": {
                            "quantity": 0,
                            "items": []
                        },
                        "originShield": {"enabled": false},
                        "originAccessControlId": ""
                    }
                ]
            },
            "comment": "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "id": "EDFDVBD6EXAMPLE",
        "ifMatch": "E1RTLUR9YES76O"
    },
    "responseElements": {
        "distribution": {
            "activeTrustedSigners": {
                "quantity": 0,
                "enabled": false
            },
            "id": "EDFDVBD6EXAMPLE",
            "domainName": "d111111abcdef8.cloudfront.net",
            "distributionConfig": {
                "defaultRootObject": "",
                "aliases": {
```

```
                    "quantity": 3,
                    "items": [
                        "alejandro_rosalez.awsps.myinstance.com",
                        "cross-testing.alejandro_rosalez.awsps.myinstance.com",
                        "*.alejandro_rosalez.awsps.myinstance.com"
                    ]
                },
                "cacheBehaviors": {"quantity": 0},
                "httpVersion": "http2and3",
                "originGroups": {"quantity": 0},
                "viewerCertificate": {
                    "minimumProtocolVersion": "TLSv1.2_2021",
                    "cloudFrontDefaultCertificate": false,
                    "aCMCertificateArn": "arn:aws:acm:us-
east-1:111122223333:certificate/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                    "sSLSupportMethod": "sni-only",
                    "certificateSource": "acm",
                    "certificate": "arn:aws:acm:us-east-1:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
                },
                "webACLId": "arn:aws:wafv2:us-east-1:111122223333:global/webacl/
testing-acl/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
                "customErrorResponses": {"quantity": 0},
                "logging": {
                    "includeCookies": false,
                    "prefix": "",
                    "enabled": false,
                    "bucket": ""
                },
                "priceClass": "PriceClass_All",
                "restrictions": {
                    "geoRestriction": {
                        "restrictionType": "none",
                        "quantity": 0
                    }
                },
                "isIPV6Enabled": true,
                "callerReference": "1578329170895",
                "continuousDeploymentPolicyId": "",
                "enabled": true,
                "defaultCacheBehavior": {
                    "targetOriginId": "d111111abcdef8",
                    "minTTL": 0,
                    "compress": false,
```

```
                    "maxTTL": 31536000,
                    "functionAssociations": {"quantity": 0},
                    "trustedKeyGroups": {
                        "quantity": 0,
                        "enabled": false
                    },
                    "smoothStreaming": false,
                    "fieldLevelEncryptionId": "",
                    "defaultTTL": 86400,
                    "lambdaFunctionAssociations": {"quantity": 0},
                    "viewerProtocolPolicy": "redirect-to-https",
                    "forwardedValues": {
                        "cookies": {"forward": "none"},
                        "queryStringCacheKeys": {"quantity": 0},
                        "queryString": false,
                        "headers": {
                            "quantity": 1,
                            "items": ["*"]
                        }
                    },
                    "trustedSigners": {
                        "enabled": false,
                        "quantity": 0
                    },
                    "allowedMethods": {
                        "quantity": 2,
                        "items": [
                            "HEAD",
                            "GET"
                        ],
                        "cachedMethods": {
                            "quantity": 2,
                            "items": [
                                "HEAD",
                                "GET"
                            ]
                        }
                    }
                },
                "staging": false,
                "origins": {
                    "quantity": 1,
                    "items": [
                        {
```

```
                        "originPath": "",
                        "connectionTimeout": 10,
                        "customOriginConfig": {
                            "originReadTimeout": 30,
                            "hTTPSPort": 443,
                            "originProtocolPolicy": "https-only",
                            "originKeepaliveTimeout": 5,
                            "hTTPPort": 80,
                            "originSslProtocols": {
                                "quantity": 3,
                                "items": [
                                    "TLSv1",
                                    "TLSv1.1",
                                    "TLSv1.2"
                                ]
                            }
                        },
                        "id": "d111111abcdef8",
                        "domainName": "d111111abcdef8.cloudfront.net",
                        "connectionAttempts": 3,
                        "customHeaders": {"quantity": 0},
                        "originShield": {"enabled": false},
                        "originAccessControlId": ""
                    }
                ]
            },
            "comment": "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "aliasICPRecordals": [
            {
                "cNAME": "alejandro_rosalez.awsps.myinstance.com",
                "iCPRecordalStatus": "APPROVED"
            },
            {
                "cNAME": "cross-testing.alejandro_rosalez.awsps.myinstance.com",
                "iCPRecordalStatus": "APPROVED"
            },
            {
                "cNAME": "*.alejandro_rosalez.awsps.myinstance.com",
                "iCPRecordalStatus": "APPROVED"
            }
        ],
        "aRN": "arn:aws:cloudfront::111122223333:distribution/EDFDVBD6EXAMPLE",
        "status": "InProgress",
```

```
            "lastModifiedTime": "Feb 2, 2024 7:26:01 PM",
            "activeTrustedKeyGroups": {
                "enabled": false,
                "quantity": 0
            },
            "inProgressInvalidationBatches": 0
        },
        "eTag": "E1YHBLAB2BJY1G"
    },
    "requestID": "4e6b66f9-d548-11e3-a8a9-73e33example",
    "eventID": "5ab02562-0fc5-43d0-b7b6-90293example",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "apiVersion": "2020_05_31",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "cloudfront.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
}
```


Example: UpdateKeys

The following example shows a CloudTrail event that demonstrates the UpdateKeys operation.

For calls to the CloudFront KeyValueStore API, the `eventSource` is `edgekeyvaluestore.amazonaws.com` instead of `cloudfront.amazonaws.com`.


```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:role-session-name",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/role-session-name",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
```

```
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2023-11-01T23:41:14Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-11-01T23:41:28Z",
    "eventSource": "edgekeyvaluestore.amazonaws.com",
    "eventName": "UpdateKeys",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "3.235.183.252",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/121.0.0.0 Safari/537.36,
    "requestParameters": {
        "kvsARN": "arn:aws:cloudfront::111122223333:key-value-store/a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111",
        "ifMatch": "KV3O6B1CX531EBP",
        "deletes": [
            {"key": "key1"}
        ]
    },
    "responseElements": {
        "itemCount": 0,
        "totalSizeInBytes": 0,
        "eTag": "KVDC9VEVZ71ZGO"
    },
    "requestID": "5ccf104c-acce-4ea1-b7fc-73e33example",
    "eventID": "a0b1b5c7-906c-439d-9925-90293example",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::CloudFront::KeyValueStore",
            "ARN": "arn:aws:cloudfront::111122223333:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
```

```
    "recipientAccountId": "111122223333",
    "eventCategory": "Data",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "111122223333.cloudfront-kvs.global.api.aws"
    }
}
```

For information about CloudTrail record contents, see CloudTrail record contents in the *Amazon CloudTrail User Guide*.

Tracking configuration changes with Amazon Config

You can use Amazon Config to record configuration changes for CloudFront distribution settings changes. For example, you can capture changes to distribution states, price classes, origins, geographic restriction settings, and Lambda@Edge configurations.

> **Note**
>
> Amazon Config does not record key–value tags for CloudFront streaming distributions.

Set up Amazon Config with CloudFront

When you set up Amazon Config, you can choose to record all supported Amazon resources, or you can specify only certain resources to record configuration changes for, such as just recording changes for CloudFront. To see the specific resources supported for CloudFront, see the list of Supported Amazon Resource Types in the *Amazon Config Developer Guide*.

To track configuration changes to your CloudFront distribution, you must log in to the Amazon Console in the US East (N. Virginia) public region.

> **Note**
>
> There might be a delay in recording resources with Amazon Config. Amazon Config records resources only after it discovers the resources.

Set up Amazon Config with CloudFront by using the Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Amazon Config console at https://console.amazonaws.cn/config/.
2. Choose **Get Started Now**.
3. On the **Settings** page, for **Resource types to record**, specify the Amazon resource types that you want Amazon Config to record. If you want to record only CloudFront changes, choose **Specific types**, and then, under **CloudFront**, choose the distribution or streaming distribution that you want to track changes for.

   To add or change which distributions to track, choose **Settings** on the left, after completing your initial setup.
4. Specify additional required options for Amazon Config: set up a notification, specify a location for the configuration information, and add rules for evaluating resource types.

For more information, see Setting up Amazon Config with the Console in the *Amazon Config Developer Guide*.

To set up Amazon Config with CloudFront by using the Amazon CLI or by using an API, see one of the following:

- **Use the Amazon CLI:** Setting up Amazon Config with the Amazon CLI in the *Amazon Config Developer Guide*
- **Use an API:** The StartConfigurationRecorder action and other information in the *Amazon Config API Reference*

View CloudFront configuration history

After Amazon Config starts recording configuration changes to your distributions, you can get the configuration history of any distribution that you have configured for CloudFront.

You can view configuration histories in any of the following ways:

- **Use the Amazon Config console.** For each recorded resource, you can view a timeline page, which provides a history of configuration details. To view this page, choose the gray icon in

the **Config Timeline** column of the **Dedicated Hosts** page. For more information, see Viewing Configuration Details in the Amazon Config Console in the *Amazon Config Developer Guide*.

- **Run Amazon CLI commands.** To get a list of all your distributions, use the list-discovered-resources command. To get the configuration details of a distribution for a specific time interval, use the get-resource-config-history command. For more information, see View Configuration Details Using the CLI in the *Amazon Config Developer Guide*.
- **Use the Amazon Config API in your applications.** To get a list of all your distributions use the ListDiscoveredResources action. To get the configuration details of a distribution for a specific time interval, use the GetResourceConfigHistory action. For more information, see the Amazon Config API Reference.

For example, to get a list of all of your distributions from Amazon Config, you could run a CLI command such as the following:

```
aws configservice list-discovered-resources --resource-type
AWS::CloudFront::Distribution
```

Security in Amazon CloudFront

Cloud security at Amazon is the highest priority. As an Amazon customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between Amazon and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – Amazon is responsible for protecting the infrastructure that runs Amazon services in the Amazon Cloud. Amazon also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the Amazon compliance programs. To learn about the compliance programs that apply to Amazon CloudFront, see Amazon Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the Amazon service that you use. You are also responsible for other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using CloudFront. The following topics show you how to configure CloudFront to meet your security and compliance objectives. You also learn how to use other Amazon services that help you to monitor and secure your CloudFront resources.

Topics

- Data protection in Amazon CloudFront
- Identity and Access Management for Amazon CloudFront
- Logging and monitoring in Amazon CloudFront
- Compliance validation for Amazon CloudFront
- Resilience in Amazon CloudFront
- Infrastructure security in Amazon CloudFront

Data protection in Amazon CloudFront

The Amazon shared responsibility model applies to data protection in Amazon CloudFront. As described in this model, Amazon is responsible for protecting the global infrastructure that runs all

of the Amazon Web Services Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the Amazon Web Services that you use. For more information about data privacy, see the Data Privacy FAQ.

For data protection purposes, we recommend that you protect Amazon Web Services account credentials and set up individual users with Amazon IAM Identity Center or Amazon Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with Amazon resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with Amazon CloudTrail.
- Use Amazon encryption solutions, along with all default security controls within Amazon Web Services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing Amazon through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with CloudFront or other Amazon Web Services using the console, API, Amazon CLI, or Amazon SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Amazon CloudFront provides several options that you can use to help secure the content that it delivers:

- Configure HTTPS connections.
- Configure field-level encryption to provide additional security for specific data during transit.
- Restrict access to content so that only specific people, or people in a specific area, can view it.

The following topics explain the options in more detail.

Topics

- Encryption in transit
- Encryption at rest
- Restrict access to content

Encryption in transit

To encrypt your data during transit, you configure Amazon CloudFront to require that viewers use HTTPS to request your files, so that connections are encrypted when CloudFront communicates with viewers. You also can configure CloudFront to use HTTPS to get files from your origin, so that connections are encrypted when CloudFront communicates with your origin.

For more information, see Using HTTPS with CloudFront.

Field-level encryption adds an additional layer of security along with HTTPS that lets you protect specific data throughout system processing so that only certain applications can see it. By configuring field-level encryption in CloudFront, you can securely upload user-submitted sensitive information to your web servers. The sensitive information provided by your clients is encrypted at the edge closer to the user. It remains encrypted throughout your entire application stack, ensuring that only applications that need the data—and have the credentials to decrypt it—are able to do so.

For more information, see Using field-level encryption to help protect sensitive data.

The CloudFront API endpoints, `cloudfront.amazonaws.com` and `cloudfront-fips.amazonaws.com`, only accept HTTPS traffic. This means that when you send and receive information using the CloudFront API, your data—including distribution configurations, cache policies and origin request policies, key groups and public keys, and function code in CloudFront Functions—is always encrypted in transit. In addition, all requests sent to the CloudFront API endpoints are signed with Amazon credentials and logged in Amazon CloudTrail.

Function code and configuration in CloudFront Functions is always encrypted in transit when copied to the edge location points of presence (POPs), and between other storage locations used by CloudFront.

Encryption at rest

Function code and configuration in CloudFront Functions is always stored in an encrypted format on the edge location POPs, and in other storage locations used by CloudFront.

Restrict access to content

Many companies that distribute content over the internet want to restrict access to documents, business data, media streams, or content that is intended for a subset of users. To securely serve this content by using Amazon CloudFront, you can do one or more of the following:

Use signed URLs or cookies

You can restrict access to content that is intended for selected users—for example, users who have paid a fee—by serving this private content through CloudFront using signed URLs or signed cookies. For more information, see Serving private content with signed URLs and signed cookies.

Restrict access to content in Amazon S3 buckets

If you restrict access to your content by using, for example, CloudFront signed URLs or signed cookies, you also won't want people to view files by using the direct URL for the file. Instead, you want them to access the files only by using the CloudFront URL, so that your protections work.

If you use an Amazon S3 bucket as the origin for a CloudFront distribution, you can set up an origin access control (OAC) which makes it possible to restrict access to the S3 bucket. For more information, see the section called “Restricting access to an Amazon Simple Storage Service origin”.

Restrict access to content served by an Application Load Balancer

When you use CloudFront with an Application Load Balancer in Elastic Load Balancing as the origin, you can configure CloudFront to prevent users from directly accessing the Application Load Balancer. This allows users to access the Application Load Balancer only through CloudFront, ensuring that you get the benefits of using CloudFront. For more information, see Restricting access to Application Load Balancers.

Use Amazon WAF web ACLs

You can use Amazon WAF, a web application firewall service, to create a web access control list (web ACL) to restrict access to your content. Based on conditions that you specify, such as the

IP addresses that requests originate from or the values of query strings, CloudFront responds to requests either with the requested content or with an HTTP 403 status code (Forbidden). For more information, see Using Amazon WAF protections.

Use geo restriction

You can use *geo restriction*, also known as *geoblocking*, to prevent users in specific geographic locations from accessing content that you serve through a CloudFront distribution. There are several options to choose from when you configure geo restrictions. For more information, see Restricting the geographic distribution of your content.

Identity and Access Management for Amazon CloudFront

Amazon Identity and Access Management (IAM) is an Amazon Web Service that helps an administrator securely control access to Amazon resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use CloudFront resources. IAM is an Amazon Web Service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How Amazon CloudFront works with IAM
- Identity-based policy examples for Amazon CloudFront
- Amazon managed policies for Amazon CloudFront
- Troubleshooting Amazon CloudFront identity and access

Audience

How you use Amazon Identity and Access Management (IAM) differs, depending on the work that you do in CloudFront.

Service user – If you use the CloudFront service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more CloudFront features to do your work, you might need additional permissions. Understanding how access is managed can

help you request the right permissions from your administrator. If you cannot access a feature in CloudFront, see Troubleshooting Amazon CloudFront identity and access.

Service administrator – If you're in charge of CloudFront resources at your company, you probably have full access to CloudFront. It's your job to determine which CloudFront features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with CloudFront, see How Amazon CloudFront works with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to CloudFront. To view example CloudFront identity-based policies that you can use in IAM, see Identity-based policy examples for Amazon CloudFront.

Authenticating with identities

Authentication is how you sign in to Amazon using your identity credentials. You must be *authenticated* (signed in to Amazon) as the Amazon Web Services account root user, as an IAM user, or by assuming an IAM role.

If you access Amazon programmatically, Amazon provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use Amazon tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing Amazon API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, Amazon recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using multi-factor authentication (MFA) in Amazon in the *IAM User Guide*.

Amazon Web Services account root user

When you create an Amazon Web Services account, you begin with one sign-in identity that has complete access to all Amazon Web Services and resources in the account. This identity is called the Amazon Web Services account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access Amazon Web Services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the Amazon Directory Service, or any user that accesses Amazon Web Services by using credentials provided through an identity source. When federated identities access Amazon Web Services accounts, they assume roles, and the roles provide temporary credentials.

IAM users and groups

An *IAM user* is an identity within your Amazon Web Services account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your Amazon Web Services account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the Amazon Web Services Management Console by switching roles. You can assume a role by calling an Amazon CLI or Amazon API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some Amazon Web Services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **Cross-service access** – Some Amazon Web Services use features in other Amazon Web Services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in Amazon, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an Amazon Web Service, combined with the requesting Amazon Web Service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other Amazon Web Services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an Amazon Web Service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an Amazon Web Service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your Amazon Web Services account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making Amazon CLI or

Amazon API requests. This is preferable to storing access keys within the EC2 instance. To assign an Amazon role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in Amazon by creating policies and attaching them to Amazon identities or resources. A policy is an object in Amazon that, when associated with an identity or resource, defines their permissions. Amazon evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in Amazon as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use Amazon JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the Amazon Web Services Management Console, the Amazon CLI, or the Amazon API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your Amazon Web Services account. Managed policies include Amazon managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or Amazon Web Services.

Resource-based policies are inline policies that are located in that service. You can't use Amazon managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, Amazon WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

Amazon supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based

policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.

- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in Amazon Organizations. Amazon Organizations is a service for grouping and centrally managing multiple Amazon Web Services accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each Amazon Web Services account root user. For more information about Organizations and SCPs, see How SCPs work in the *Amazon Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how Amazon determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Amazon CloudFront works with IAM

Before you use IAM to manage access to CloudFront, learn what IAM features are available to use with CloudFront.

IAM features you can use with Amazon CloudFront

IAM feature	CloudFront support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes

IAM feature	CloudFront support
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Forward access sessions (FAS)	No
Service roles	No
Service-linked roles	Yes

To get a high-level view of how CloudFront and other Amazon services work with most IAM features, see Amazon services that work with IAM in the *IAM User Guide*.

Identity-based policies for CloudFront

Supports identity-based policies	Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for CloudFront

To view examples of CloudFront identity-based policies, see Identity-based policy examples for Amazon CloudFront.

Resource-based policies within CloudFront

Supports resource-based policies	No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or Amazon Web Services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different Amazon Web Services accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Policy actions for CloudFront

Supports policy actions	Yes

Administrators can use Amazon JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated Amazon API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of CloudFront actions, see Actions defined by Amazon CloudFront in the *Service Authorization Reference*.

Policy actions in CloudFront use the following prefix before the action:

```
cloudfront
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "cloudfront:action1",
      "cloudfront:action2"
         ]
```

To view examples of CloudFront identity-based policies, see Identity-based policy examples for Amazon CloudFront.

Policy resources for CloudFront

Supports policy resources	Yes

Administrators can use Amazon JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of CloudFront resource types and their ARNs, see Resources defined by Amazon CloudFront in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Amazon CloudFront.

To view examples of CloudFront identity-based policies, see Identity-based policy examples for Amazon CloudFront.

Policy condition keys for CloudFront

Supports service-specific policy condition keys	Yes

Administrators can use Amazon JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, Amazon evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, Amazon evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

Amazon supports global condition keys and service-specific condition keys. To see all Amazon global condition keys, see Amazon global condition context keys in the *IAM User Guide*.

To see a list of CloudFront condition keys, see Condition keys for Amazon CloudFront in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by Amazon CloudFront.

To view examples of CloudFront identity-based policies, see Identity-based policy examples for Amazon CloudFront.

ACLs in CloudFront

Supports ACLs	No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with CloudFront

Supports ABAC (tags in policies)	Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In Amazon, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many Amazon resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with CloudFront

Supports temporary credentials	Yes

Some Amazon Web Services don't work when you sign in using temporary credentials. For additional information, including which Amazon Web Services work with temporary credentials, see Amazon Web Services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the Amazon Web Services Management Console using any method except a user name and password. For example, when you access Amazon using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the Amazon CLI or Amazon API. You can then use those temporary credentials to access Amazon. Amazon recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Forward access sessions for CloudFront

Supports forward access sessions (FAS)	No

When you use an IAM user or role to perform actions in Amazon, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an Amazon Web Service, combined with the requesting Amazon Web Service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other Amazon Web Services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service roles for CloudFront

Supports service roles	No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an Amazon Web Service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break CloudFront functionality. Edit service roles only when CloudFront provides guidance to do so.

Service-linked roles for CloudFront

Supports service-linked roles	Yes

A service-linked role is a type of service role that is linked to an Amazon Web Service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your Amazon Web Services account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see Amazon services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for Amazon CloudFront

By default, users and roles don't have permission to create or modify CloudFront resources. They also can't perform tasks by using the Amazon Web Services Management Console, Amazon Command Line Interface (Amazon CLI), or Amazon API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by CloudFront, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon CloudFront in the *Service Authorization Reference*.

Topics

- Policy best practices
- Using the CloudFront console

- Allow users to view their own permissions
- Permissions to access CloudFront programmatically
- Permissions required to use the CloudFront console
- Amazon managed (predefined) policies for CloudFront
- Customer managed policy examples

Policy best practices

Identity-based policies determine whether someone can create, access, or delete CloudFront resources in your account. These actions can incur costs for your Amazon Web Services account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with Amazon managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *Amazon managed policies* that grant permissions for many common use cases. They are available in your Amazon Web Services account. We recommend that you reduce permissions further by defining Amazon customer managed policies that are specific to your use cases. For more information, see Amazon managed policies or Amazon managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific Amazon Web Service, such as Amazon CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.

- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your Amazon Web Services account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the CloudFront console

To access the Amazon CloudFront console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the CloudFront resources in your Amazon Web Services account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the Amazon CLI or the Amazon API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the CloudFront console, also attach the CloudFront `ConsoleAccess` or `ReadOnly` Amazon managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the Amazon CLI or Amazon API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
```

```
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws-cn:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Permissions to access CloudFront programmatically

The following shows a permissions policy. The `Sid`, or statement ID, is optional.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowAllCloudFrontPermissions",
         "Effect": "Allow",
         "Action": ["cloudfront:*"],
         "Resource": "*"
      }
   ]
}
```

The policy grants permissions to perform all CloudFront operations, which is sufficient to access CloudFront programmatically. If you're using the console to access CloudFront, see Permissions required to use the CloudFront console.

For a list of actions and the ARN that you specify to grant or deny permission to use each action, see Actions, resources, and condition keys for Amazon CloudFront in the *Service Authorization Reference*.

Permissions required to use the CloudFront console

To grant full access to the CloudFront console, you grant the permissions in the following permissions policy:

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm:ListCertificates",
            "cloudfront:*",
            "cloudwatch:DescribeAlarms",
            "cloudwatch:PutMetricAlarm",
            "cloudwatch:GetMetricStatistics",
            "elasticloadbalancing:DescribeLoadBalancers",
            "iam:ListServerCertificates",
            "sns:ListSubscriptionsByTopic",
            "sns:ListTopics",
            "waf:GetWebACL",
            "waf:ListWebACLs"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets",
            "s3:PutBucketPolicy"
         ],
         "Resource":"arn:aws:s3:::*"
      }
   ]
}
```

Here's why the permissions are required:

`acm:ListCertificates`

When you're creating and updating distributions by using the CloudFront console and you want to configure CloudFront to require HTTPS between the viewer and CloudFront or between CloudFront and the origin, lets you view a list of ACM certificates.

This permission isn't required if you aren't using the CloudFront console.

`cloudfront:*`

Lets you perform all CloudFront actions.

`cloudwatch:DescribeAlarms` and `cloudwatch:PutMetricAlarm`

Let you create and view CloudWatch alarms in the CloudFront console. See also `sns:ListSubscriptionsByTopic` and `sns:ListTopics`.

These permissions aren't required if you aren't using the CloudFront console.

`cloudwatch:GetMetricStatistics`

Lets CloudFront render CloudWatch metrics in the CloudFront console.

This permission isn't required if you aren't using the CloudFront console.

`elasticloadbalancing:DescribeLoadBalancers`

When creating and updating distributions, lets you view a list of Elastic Load Balancing load balancers in the list of available origins.

This permission isn't required if you aren't using the CloudFront console.

`iam:ListServerCertificates`

When you're creating and updating distributions by using the CloudFront console and you want to configure CloudFront to require HTTPS between the viewer and CloudFront or between CloudFront and the origin, lets you view a list of certificates in the IAM certificate store.

This permission isn't required if you aren't using the CloudFront console.

`s3:ListAllMyBuckets`

When you're creating and updating distributions, lets you perform the following operations:

- View a list of S3 buckets in the list of available origins

- View a list of S3 buckets that you can save access logs in

This permission isn't required if you aren't using the CloudFront console.

`S3:PutBucketPolicy`

When you're creating or updating distributions that restrict access to S3 buckets, lets a user update the bucket policy to grant access to the CloudFront origin access identity. For more information, see the section called “Using an origin access identity (legacy, not recommended)”.

This permission isn't required if you aren't using the CloudFront console.

`sns:ListSubscriptionsByTopic` and `sns:ListTopics`

When you create CloudWatch alarms in the CloudFront console, lets you choose an SNS topic for notifications.

These permissions aren't required if you aren't using the CloudFront console.

`waf:GetWebACL` and `waf:ListWebACLs`

Lets you view a list of Amazon WAF web ACLs in the CloudFront console.

These permissions aren't required if you aren't using the CloudFront console.

Amazon managed (predefined) policies for CloudFront

Amazon addresses many common use cases by providing standalone IAM policies that are created and administered by Amazon. These Amazon managed policies grant necessary permissions for common use cases so that you can avoid having to investigate what permissions are needed. For more information, see Amazon Managed Policies in the *IAM User Guide*. For CloudFront, IAM provides two managed policies:

- **CloudFrontFullAccess** – Grants full access to CloudFront resources.

> **Important**
>
> If you want CloudFront to create and save access logs, you need to grant additional permissions. For more information, see Permissions required to configure standard logging and to access your log files.

- **CloudFrontReadOnlyAccess** – Grants read-only access to CloudFront resources.

Customer managed policy examples

You can create your own custom IAM policies to allow permissions for CloudFront API actions. You can attach these custom policies to the IAM users or groups that require the specified permissions. These policies work when you are using the CloudFront API, the Amazon SDKs, or the Amazon CLI. The following examples show permissions for a few common use cases. For the policy that grants a user full access to CloudFront, see Permissions required to use the CloudFront console.

Examples

- Example 1: Allow read access to all distributions
- Example 2: Allow creating, updating, and deleting distributions
- Example 3: Allow creating and listing invalidations
- Example 4: Allow creating a distribution

Example 1: Allow read access to all distributions

The following permissions policy grants the user permissions to view all distributions in the CloudFront console:

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm:ListCertificates",
            "cloudfront:GetDistribution",
            "cloudfront:GetDistributionConfig",
            "cloudfront:ListDistributions",
            "cloudfront:ListCloudFrontOriginAccessIdentities",
            "elasticloadbalancing:DescribeLoadBalancers",
            "iam:ListServerCertificates",
            "sns:ListSubscriptionsByTopic",
            "sns:ListTopics",
            "waf:GetWebACL",
            "waf:ListWebACLs"
         ],
         "Resource":"*"
      },
      {
```

```
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets"
         ],
         "Resource":"arn:aws:s3:::*"
      }
   ]
}
```

Example 2: Allow creating, updating, and deleting distributions

The following permissions policy allows users to create, update, and delete distributions by using the CloudFront console:

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm:ListCertificates",
            "cloudfront:CreateDistribution",
            "cloudfront:DeleteDistribution",
            "cloudfront:GetDistribution",
            "cloudfront:GetDistributionConfig",
            "cloudfront:ListDistributions",
            "cloudfront:UpdateDistribution",
            "cloudfront:ListCloudFrontOriginAccessIdentities",
            "elasticloadbalancing:DescribeLoadBalancers",
            "iam:ListServerCertificates",
            "sns:ListSubscriptionsByTopic",
            "sns:ListTopics",
            "waf:GetWebACL",
            "waf:ListWebACLs"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets",
            "s3:PutBucketPolicy"
         ],
```

```
         "Resource":"arn:aws:s3:::*"
      }
   ]
}
```

The `cloudfront:ListCloudFrontOriginAccessIdentities` permission allows users to automatically grant to an existing origin access identity the permission to access objects in an Amazon S3 bucket. If you also want users to be able to create origin access identities, you also need to allow the `cloudfront:CreateCloudFrontOriginAccessIdentity` permission.

Example 3: Allow creating and listing invalidations

The following permissions policy allows users to create and list invalidations. It includes read access to CloudFront distributions because you create and view invalidations by first displaying settings for a distribution:

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm:ListCertificates",
            "cloudfront:GetDistribution",
            "cloudfront:GetStreamingDistribution",
            "cloudfront:GetDistributionConfig",
            "cloudfront:ListDistributions",
            "cloudfront:ListCloudFrontOriginAccessIdentities",
            "cloudfront:CreateInvalidation",
            "cloudfront:GetInvalidation",
            "cloudfront:ListInvalidations",
            "elasticloadbalancing:DescribeLoadBalancers",
            "iam:ListServerCertificates",
            "sns:ListSubscriptionsByTopic",
            "sns:ListTopics",
            "waf:GetWebACL",
            "waf:ListWebACLs"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
```

```
            "s3:ListAllMyBuckets"
         ],
         "Resource":"arn:aws:s3:::*"
      }
   ]
}
```

Example 4: Allow creating a distribution

The following permission policy grants the user permission to create and list distributions in the CloudFront console. For the `CreateDistribution` action, specify the wildcard (*) character for the `Resource` instead of a wildcard for the distribution ARN (`arn:aws:cloudfront::123456789012:distribution/*`). For more information about the `Resource` element, see IAM JSON policy elements: Resource in the *IAM User Guide*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "cloudfront:CreateDistribution",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "cloudfront:ListDistributions",
            "Resource": "*"
        }
    ]
}
```

Amazon managed policies for Amazon CloudFront

To add permissions to users, groups, and roles, it's easier to use Amazon managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your users with only the permissions they need. To get started quickly, you can use

our Amazon managed policies. These policies cover common use cases and are available in your Amazon Web Services account. For more information about Amazon managed policies, see Amazon managed policies in the *IAM User Guide*.

Amazon services maintain and update Amazon managed policies. You can't change the permissions in Amazon managed policies. Services occasionally add additional permissions to an Amazon managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an Amazon managed policy when a new feature is launched or when new permissions become available. Services do not remove permissions from an Amazon managed policy, so policy updates won't break your existing permissions.

Additionally, Amazon supports managed policies for job functions that span multiple services. For example, the **ReadOnlyAccess** Amazon managed policy provides read-only access to all Amazon services and resources. When a service launches a new feature, Amazon adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see Amazon managed policies for job functions in the *IAM User Guide*.

Amazon managed policy: CloudFrontReadOnlyAccess

You can attach the **CloudFrontReadOnlyAccess** policy to your IAM identities. This policy allows read-only permissions to CloudFront resources. It also allows read-only permissions to other Amazon service resources that are related to CloudFront and that are visible in the CloudFront console.

Permissions details

This policy includes the following permissions.

- `cloudfront:Describe*` – Allows principals to get information about metadata about CloudFront resources.
- `cloudfront:Get*` – Allows principals to get detailed information and configurations for CloudFront resources.
- `cloudfront:List*` – Allows principals to get lists of CloudFront resources.
- `cloudfront-keyvaluestore:Describe*` - Allows principals to get information about the key value store.

- `cloudfront-keyvaluestore:Get*` - Allows principals to get detailed information and configurations for the key value store.
- `cloudfront-keyvaluestore:List*` - Allows principals to get lists of the key value stores.
- `acm:ListCertificates` – Allows principals to get a list of ACM certificates.
- `iam:ListServerCertificates` – Allows principals to get a list of server certificates stored in IAM.
- `route53:List*` – Allows principals to get lists of Route 53 resources.
- `waf:ListWebACLs` – Allows principals to get a list of web ACLs in Amazon WAF.
- `waf:GetWebACL` – Allows principals to get detailed information about web ACLs in Amazon WAF.
- `wafv2:ListWebACLs` – Allows principals to get a list of web ACLs in Amazon WAF.
- `wafv2:GetWebACL` – Allows principals to get detailed information about web ACLs in Amazon WAF.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "cfReadOnly",
      "Effect": "Allow",
      "Action": [
        "acm:ListCertificates",
        "cloudfront:Describe*",
        "cloudfront:Get*",
        "cloudfront:List*",
        "cloudfront-keyvaluestore:Describe*",
        "cloudfront-keyvaluestore:Get*",
        "cloudfront-keyvaluestore:List*",
        "iam:ListServerCertificates",
        "route53:List*",
        "waf:ListWebACLs",
        "waf:GetWebACL",
        "wafv2:ListWebACLs",
        "wafv2:GetWebACL"
      ],
      "Resource": "*"
    }
  ]
```

```
}
```

Amazon managed policy: CloudFrontFullAccess

You can attach the **CloudFrontFullAccess** policy to your IAM identities. This policy allows administrative permissions to CloudFront resources. It also allows read-only permissions to other Amazon service resources that are related to CloudFront and that are visible in the CloudFront console.

Permissions details

This policy includes the following permissions.

- `s3:ListAllMyBuckets` – Allows principals to get a list of all Amazon S3 buckets.
- `acm:ListCertificates` – Allows principals to get a list of ACM certificates.
- `cloudfront:*` – Allows principals to perform all actions on all CloudFront resources.
- `cloudfront-keyvaluestore:*` - Allows principals to perform all actions on the key value store.
- `iam:ListServerCertificates` – Allows principals to get a list of server certificates stored in IAM.
- `waf:ListWebACLs` – Allows principals to get a list of web ACLs in Amazon WAF.
- `waf:GetWebACL` – Allows principals to get detailed information about web ACLs in Amazon WAF.
- `wafv2:ListWebACLs` – Allows principals to get a list of web ACLs in Amazon WAF.
- `wafv2:GetWebACL` – Allows principals to get detailed information about web ACLs in Amazon WAF.
- `kinesis:ListStreams` – Allows principals to get a list of Amazon Kinesis streams.
- `kinesis:DescribeStream` – Allows principals to get detailed information about a Kinesis stream.
- `iam:ListRoles` – Allows principals to get a list of roles in IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "cfflistbuckets",
```

```
      "Action": [
        "s3:ListAllMyBuckets"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "cffullaccess",
      "Action": [
        "acm:ListCertificates",
        "cloudfront:*",
        "cloudfront-keyvaluestore:*",
        "iam:ListServerCertificates",
        "waf:ListWebACLs",
        "waf:GetWebACL",
        "wafv2:ListWebACLs",
        "wafv2:GetWebACL",
        "kinesis:ListStreams"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "cffdescribestream",
      "Action": [
        "kinesis:DescribeStream"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:kinesis:*:*:*"
    },
    {
      "Sid": "cfflistroles",
      "Action": [
        "iam:ListRoles"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:*"
    }
  ]
}
```

Amazon managed policy: AWSCloudFrontLogger

You can't attach the **AWSCloudFrontLogger** policy to your IAM identities. This policy is attached to a service-linked role that allows CloudFront to perform actions on your behalf. For more information, see the section called "Service-linked roles for Lambda@Edge".

This policy allows CloudFront to push log files to Amazon CloudWatch. For details about the permissions included in this policy, see the section called "Service-linked role permissions for CloudFront logger".

Amazon managed policy: AWSLambdaReplicator

You can't attach the **AWSLambdaReplicator** policy to your IAM identities. This policy is attached to a service-linked role that allows CloudFront to perform actions on your behalf. For more information, see the section called "Service-linked roles for Lambda@Edge".

This policy allows CloudFront to create, delete, and disable functions in Amazon Lambda to replicate Lambda@Edge functions to Amazon Web Services Regions. For details about the permissions included in this policy, see the section called "Service-linked role permissions for Lambda replicator".

CloudFront updates to Amazon managed policies

View details about updates to Amazon managed policies for CloudFront since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the CloudFront Document history page.

Change	Description	Date
CloudFrontReadOnlyAccess and CloudFrontFullAccess - Update to two existing policies.	CloudFront added new permissions for key value stores. The new permissions allow users to get information about, and take action on, key value stores.	December 19, 2023

Change	Description	Date
CloudFrontReadOnlyAccess – Update to an existing policy	CloudFront added a new permission to describe CloudFront Functions. This permission allows the user, group, or role to read information and metadata about a function, but not the function's code.	September 8, 2021
CloudFront started tracking changes	CloudFront started tracking changes for its Amazon managed policies.	September 8, 2021

Troubleshooting Amazon CloudFront identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with CloudFront and IAM.

Topics

- I am not authorized to perform an action in CloudFront
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my Amazon Web Services account to access my CloudFront resources

I am not authorized to perform an action in CloudFront

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `cloudfront:GetWidget` permissions.

```
User: arn:aws-cn:iam::123456789012:user/mateojackson is not authorized to perform:
 cloudfront:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *my-example-widget* resource by using the `cloudfront:GetWidget` action.

If you need help, contact your Amazon administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to CloudFront.

Some Amazon Web Services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in CloudFront. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws-cn:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your Amazon administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my Amazon Web Services account to access my CloudFront resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether CloudFront supports these features, see How Amazon CloudFront works with IAM.
- To learn how to provide access to your resources across Amazon Web Services accounts that you own, see Providing access to an IAM user in another Amazon Web Services account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party Amazon Web Services accounts, see Providing access to Amazon Web Services accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Logging and monitoring in Amazon CloudFront

Monitoring is an important part of maintaining the availability and performance of CloudFront and your Amazon solutions. You should collect monitoring data from all of the parts of your Amazon solution so that you can more easily debug a multi-point failure if one occurs. Amazon provides several tools for monitoring your CloudFront resources and activity, and responding to potential incidents:

Amazon CloudWatch alarms

Using CloudWatch alarms, you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon SNS topic or Amazon Auto Scaling policy. CloudWatch alarms do not invoke actions when a metric is in a particular state. Rather the state must have changed and been maintained for a specified number of periods. For more information, see Monitoring CloudFront metrics with Amazon CloudWatch.

Amazon CloudTrail logs

CloudTrail provides a record of API actions taken by a user, role, or an Amazon service in CloudFront. Using the information collected by CloudTrail, you can determine the API request that was made to CloudFront, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Logging Amazon CloudFront API calls using Amazon CloudTrail.

CloudFront standard logs and real-time logs

CloudFront logs provide detailed records about requests that are made to a distribution. These logs are useful for many applications. For example, log information can be useful in security and access audits. For more information, see CloudFront and edge function logging.

Edge function logs

Logs generated by edge functions, both CloudFront Functions and Lambda@Edge, are sent directly to Amazon CloudWatch Logs and are not stored anywhere by CloudFront. CloudFront Functions uses an Amazon Identity and Access Management (IAM) service-linked role to send customer-generated logs directly to CloudWatch Logs in your account.

CloudFront console reports

The CloudFront console includes a variety of reports, including the cache statistics report, the popular objects report, and the top referrers report. Most CloudFront console reports are based on the data in CloudFront access logs, which contain detailed information about every user request that CloudFront receives. However, you don't need to enable access logs to view the reports. For more information, see CloudFront reports in the console.

Compliance validation for Amazon CloudFront

Third-party auditors assess the security and compliance of Amazon CloudFront as part of multiple Amazon compliance programs. These include SOC, PCI, HIPAA, and others.

For a list of Amazon services in scope of specific compliance programs, see Amazon Services in Scope by Compliance Program. For general information, see Amazon Compliance Programs.

You can download third-party audit reports using Amazon Artifact. For more information, see Downloading Reports in Amazon Artifact.

Your compliance responsibility when using CloudFront is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. Amazon provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on Amazon.
- Architecting for HIPAA Security and Compliance on Amazon – This whitepaper describes how companies can use Amazon to create HIPAA-compliant applications.

The Amazon HIPAA compliance program includes CloudFront (excluding content delivery through CloudFront Embedded POPs) as a HIPAA eligible service. If you have an executed Business Associate Addendum (BAA) with Amazon, you can use CloudFront (excluding content delivery through CloudFront Embedded POPs) to deliver content that contains protected health information (PHI). For more information, see HIPAA Compliance.

- Amazon Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Amazon Config – This Amazon service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- Amazon Security Hub – This Amazon service uses security controls to evaluate resource configurations and security standards to help you comply with various compliance frameworks. For more information about using Security Hub to evaluate CloudFront resources, see Amazon CloudFront controls in the *Amazon Security Hub User Guide*.

CloudFront compliance best practices

This section provides best practices and recommendations for compliance when you use Amazon CloudFront to serve your content.

If you run PCI-compliant or HIPAA-compliant workloads that are based on the Amazon shared responsibility model, we recommend that you log your CloudFront usage data for the last 365 days for future auditing purposes. To log usage data, you can do the following:

- Enable CloudFront access logs. For more information, see Configuring and using standard logs (access logs).
- Capture requests that are sent to the CloudFront API. For more information, see Logging Amazon CloudFront API calls using Amazon CloudTrail.

In addition, see the following for details about how CloudFront is compliant with the PCI DSS and SOC standards.

Payment Card Industry Data Security Standard (PCI DSS)

CloudFront (excluding content delivery through CloudFront Embedded POPs) supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard

(DSS). For more information about PCI DSS, including how to request a copy of the Amazon PCI Compliance Package, see PCI DSS Level 1.

As a security best practice, we recommend that you don't cache credit card information in CloudFront edge caches. For example, you can configure your origin to include a `Cache-Control:no-cache="field-name"` header in responses that contain credit card information, such as the last four digits of a credit card number and the card owner's contact information.

System and Organization Controls (SOC)

CloudFront (excluding content delivery through CloudFront Embedded POPs) is compliant with System and Organization Controls (SOC) measures, including SOC 1, SOC 2, and SOC 3. SOC reports are independent, third-party examination reports that demonstrate how Amazon achieves key compliance controls and objectives. These audits ensure that the appropriate safeguards and procedures are in place to protect against risks that might affect the security, confidentiality, and availability of customer and company data. The results of these third-party audits are available on the Amazon SOC Compliance website, where you can view the published reports to get more information about the controls that support Amazon operations and compliance.

Resilience in Amazon CloudFront

The Amazon global infrastructure is built around Amazon Regions and Availability Zones. Amazon Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about Amazon Regions and Availability Zones, see Amazon Global Infrastructure.

CloudFront origin failover

In addition to the support of Amazon global infrastructure, Amazon CloudFront offers an *origin failover* feature to help support your data resiliency needs. CloudFront is a global service that delivers your content through a worldwide network of data centers called *edge locations* or *points of presence* (POPs). If your content is not already cached in an edge location, CloudFront retrieves it from an origin that you've identified as the source for the definitive version of the content.

You can improve resiliency and increase availability for specific scenarios by setting up CloudFront with origin failover. To get started, you create an origin group in which you designate a primary origin for CloudFront plus a second origin. CloudFront automatically switches to the second origin when the primary origin returns specific HTTP status code failure responses. For more information, see Optimizing high availability with CloudFront origin failover.

Infrastructure security in Amazon CloudFront

As a managed service, Amazon CloudFront is protected by Amazon global network security. For information about Amazon security services and how Amazon protects infrastructure, see Amazon Cloud Security. To design your Amazon environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar Amazon Well-Architected Framework*.

You use Amazon published API calls to access CloudFront through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the Amazon Security Token Service (Amazon STS) to generate temporary security credentials to sign requests.

CloudFront Functions uses a highly secure isolation barrier between Amazon accounts, ensuring that customer environments are secure against side-channel attacks like Spectre and Meltdown. Functions cannot access or modify data belonging to other customers. Functions run in a dedicated single-threaded process on a dedicated CPU without hyperthreading. In any given CloudFront edge location point of presence (POP), CloudFront Functions only serves one customer at a time, and all customer-specific data is cleared between function executions.

Quotas

CloudFront is subject to the following quotas.

Topics

- General quotas
- General quotas on distributions
- General quotas on policies
- Quotas on CloudFront Functions
- Quotas on key value stores
- Quotas on Lambda@Edge
- Quotas on SSL certificates
- Quotas on invalidations
- Quotas on key groups
- Quotas on WebSocket connections
- Quotas on field-level encryption
- Quotas on cookies (legacy cache settings)
- Quotas on query strings (legacy cache settings)
- Quotas on headers

General quotas

Entity	Default quota
Data transfer rate per distribution	150 Gbps Request a higher quota
Requests per second per distribution	250,000 Request a higher quota

Entity	Default quota
Tags that can be added to a distribution	50
Files that you can serve per distribution	No quota
Maximum length of a request or an origin response, including headers and query strings, but not including the body content	20,480 bytes
Maximum length of a URL	8,192 bytes

General quotas on distributions

Entity	Default quota
Alternate domain names (CNAMEs) per distribution For more information, see Using custom URLs by adding alternate domain names (CNAMEs).	100 Request a higher quota
Cache behaviors per distribution	25 Request a higher quota
Connection attempts per origin For more information, see Connection attempts.	1-3
Connection timeout per origin For more information, see Connection timeout.	1-10 seconds
Distributions per Amazon Web Services account For more information, see Creating a distribution.	200 Request a higher quota
Distributions per origin access control	100

Entity	Default quota
	Request a higher quota
File compression: range of file sizes that CloudFront compresses For more information, see Serving compressed files.	1,000 to 10,000,000 bytes
Keep-alive timeout per origin For more information, see Keep-alive timeout (custom origins only).	1-60 seconds Request a higher quota
Maximum cacheable file size per HTTP GET response. Only the responses for an HTTP GET are cached. Responses for POST or PUT are not cached.	50 GB
Origin access controls per Amazon Web Services account	100
Origin access identities per Amazon Web Services account	100 Request a higher quota
Origins per distribution	25 Request a higher quota
Origin groups per distribution	10 Request a higher quota
Response timeout per origin For more information, see Response timeout (custom origins only).	1-60 seconds Request a higher quota

Entity	Default quota
Staging distributions per Amazon Web Services account	20
For more information, see the section called “Using continuous deployment to safely test changes”.	Request a higher quota

General quotas on policies

Entity	Default quota
Cache policies per Amazon Web Services account	20
Distributions associated with the same cache policy	100
Query strings per cache policy	10 Request a higher quota
Headers per cache policy	10 Request a higher quota
Cookies per cache policy	10 Request a higher quota
Total combined length of all query string, header, and cookie names in a cache policy	1024
Origin request policies per Amazon Web Services account	20
Distributions associated with the same origin request policy	100
Query strings per origin request policy	10

Entity	Default quota
	Request a higher quota
Headers per origin request policy	10 Request a higher quota
Cookies per origin request policy	10 Request a higher quota
Total combined length of all query string, header, and cookie names in an origin request policy	1024
Response headers policies per Amazon Web Services account	20 Request a higher quota
Distributions associated with the same response headers policy	100 Request a higher quota
Custom headers per response headers policy	10 Request a higher quota
Continuous deployment policies per Amazon Web Services account	20 Request a higher quota

Quotas on CloudFront Functions

Entity	Default quota
Functions per Amazon Web Services account	100
Maximum function size	10 KB Request a higher quota
Maximum function memory	2 MB
Distributions associated with the same function	100

In addition to these quotas, there are some other restrictions when using CloudFront Functions. For more information, see Restrictions on CloudFront Functions.

Quotas on key value stores

Entity	Default quota
Maximum size of a key in a key-value pair	512 Bytes
Maximum size of the value in a key-value pair	1 KB
Maximum key values pairs that you can update in a single API request	50 keys or 3 MB payload, whichever is reached first
Maximum size of an individual key value store	5 MB
Maximum number of functions that a single key value store can be associated with	10
Maximum number of key value stores per function	1
Maximum number of key value stores per account	50

Entity	Default quota
	Request a higher quota

Quotas on Lambda@Edge

The quotas in this section apply to Lambda@Edge. These quotas are in addition to the default Amazon Lambda quotas, which also apply. For the Lambda quotas, see Quotas in the *Amazon Lambda Developer Guide*.

> **Note**
>
> Lambda dynamically scales capacity in response to increased traffic, within your Amazon Web Services account's quotas. For more information, see Function scaling in the *Amazon Lambda Developer Guide*.

General quotas

Entity	Default quota
Distributions per Amazon Web Services account that can have Lambda@Edge functions	500 Request a higher quota
Lambda@Edge functions per distribution	100 Request a higher quota
Requests per second	10,000 (in each Amazon Web Services Region) Request a higher quota

Entity	Default quota
Concurrent executions For more information, see Function scaling in the *Amazon Lambda Developer Guide*.	1,000 (in each Amazon Web Services Region) Request a higher quota
Distributions associated with the same function	500

Quotas that differ by event type

Entity	Viewer request and viewer response events	Origin request and origin response events
Function memory size	128 MB	Same as Lambda quotas
Function timeout. The function can make network calls to resources such as Amazon S3 buckets, DynamoDB tables, or Amazon EC2 instances in Amazon Web Services Regions.	5 seconds	30 seconds
Size of a response that is generated by a Lambda function, including headers and body	40 KB	1 MB
Maximum compressed size of a Lambda function and any included libraries	1 MB	50 MB

In addition to these quotas, there are some other restrictions when using Lambda@Edge functions. For more information, see Restrictions on Lambda@Edge.

Quotas on SSL certificates

Entity	Default quota
SSL certificates per Amazon Web Services account when serving HTTPS requests using dedicated IP addresses (no quota when serving HTTPS requests using SNI) For more information, see Using HTTPS with CloudFront.	2 Request a higher quota
SSL certificates that can be associated with a CloudFront distribution	1

Quotas on invalidations

Entity	Default quota
File invalidation: maximum number of files allowed in active invalidation requests, excluding wildcard invalidations For more information, see Invalidating files.	3,000
File invalidation: maximum number of active wildcard invalidations allowed	15
File invalidation: maximum number of files that one wildcard invalidation can process	No quota

Quotas on key groups

Entity	Default quota
Public keys in a single key group	5 Request a higher quota

Entity	Default quota
Key groups associated with a single cache behavior	4 Request a higher quota
Key groups per Amazon Web Services account	10 Request a higher quota
Distributions associated with a single key group	100 Request a higher quota

Quotas on WebSocket connections

Entity	Default quota
Origin response timeout (idle timeout)	10 minutes If CloudFront hasn't detected any bytes sent from the origin to the client within the past 10 minutes, the connection is assumed to be idle and is closed.

Quotas on field-level encryption

Entity	Default quota
Maximum length of a field to encrypt	16 KB

Entity	Default quota
For more information, see Using field-level encryption to help protect sensitive data.	
Maximum number of fields in a request body when field-level encryption is configured	10
Maximum length of a request body when field-level encryption is configured	1 MB
Maximum number of field-level encryption configurations that can be associated with one Amazon Web Services account	10
Maximum number of field-level encryption profiles that can be associated with one Amazon Web Services account	10
Maximum number of public keys that can be added to one Amazon Web Services account	10
Maximum number of fields to encrypt that can be specified in one profile	10
Maximum number of CloudFront distributions that can be associated with a field-level encryption configuration	20
Maximum number of query argument profile mappings that can be included in a field-level encryption configuration	5

Quotas on cookies (legacy cache settings)

These quotas apply to CloudFront's legacy cache settings. We recommend using a cache policy or origin request policy instead of the legacy settings.

Entity	Default quota
Cookies per cache behavior For more information, see Caching content based on cookies.	10

Entity	Default quota
	Request a higher quota
Total number of bytes in cookie names (doesn't apply if you configure CloudFront to forward all cookies to the origin)	512 minus the number of cookies

Quotas on query strings (legacy cache settings)

These quotas apply to CloudFront's legacy cache settings. We recommend using a cache policy or origin request policy instead of the legacy settings.

Entity	Default quota
Maximum number of characters in a query string	128 characters
Maximum number of characters total for all query strings in the same parameter	512 characters
Query strings per cache behavior	10
For more information, see Caching content based on query string parameters.	Request a higher quota

Quotas on headers

Entity	Default quota
Headers per cache behavior (legacy cache settings)	10
For more information, see the section called “Caching content based on request headers”.	Request a higher quota
Custom headers: maximum number of custom headers that you can configure CloudFront to add to origin requests	10

Entity	Default quota
For more information, see the section called “Adding custom headers to origin requests”.	Request a higher quota
Custom headers: maximum number of custom headers that you can add to a response headers policy	10 Request a higher quota
Custom headers: maximum length of a header name	256 characters
Custom headers: maximum length of a header value	1,783 characters
Custom headers: maximum length of all header values and names combined	10,240 characters
Maximum length of the value of the `Content-Security-Policy` header	1,783 characters Request a higher quota

Code examples for CloudFront using Amazon SDKs

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

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 and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Code examples

- Actions for CloudFront using Amazon SDKs
  - Use CreateDistribution with an Amazon SDK or command line tool
  - Use CreateFunction with an Amazon SDK or command line tool
  - Use CreateInvalidation with an Amazon SDK or command line tool
  - Use CreateKeyGroup with an Amazon SDK or command line tool
  - Use CreatePublicKey with an Amazon SDK or command line tool
  - Use DeleteDistribution with an Amazon SDK or command line tool
  - Use GetCloudFrontOriginAccessIdentity with an Amazon SDK or command line tool
  - Use GetCloudFrontOriginAccessIdentityConfig with an Amazon SDK or command line tool
  - Use GetDistribution with an Amazon SDK or command line tool
  - Use GetDistributionConfig with an Amazon SDK or command line tool
  - Use ListCloudFrontOriginAccessIdentities with an Amazon SDK or command line tool
  - Use ListDistributions with an Amazon SDK or command line tool
  - Use UpdateDistribution with an Amazon SDK or command line tool
- Scenarios for CloudFront using Amazon SDKs
  - Delete CloudFront signing resources using Amazon SDK
  - Create signed URLs and cookies using an Amazon SDK

Actions for CloudFront using Amazon SDKs

The following code examples demonstrate how to perform individual CloudFront actions with Amazon SDKs. These excerpts call the CloudFront API and are code excerpts from larger programs that must be run in context. 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 CloudFront API Reference.

Examples

- Use CreateDistribution with an Amazon SDK or command line tool
- Use CreateFunction with an Amazon SDK or command line tool
- Use CreateInvalidation with an Amazon SDK or command line tool
- Use CreateKeyGroup with an Amazon SDK or command line tool
- Use CreatePublicKey with an Amazon SDK or command line tool
- Use DeleteDistribution with an Amazon SDK or command line tool
- Use GetCloudFrontOriginAccessIdentity with an Amazon SDK or command line tool
- Use GetCloudFrontOriginAccessIdentityConfig with an Amazon SDK or command line tool
- Use GetDistribution with an Amazon SDK or command line tool
- Use GetDistributionConfig with an Amazon SDK or command line tool
- Use ListCloudFrontOriginAccessIdentities with an Amazon SDK or command line tool
- Use ListDistributions with an Amazon SDK or command line tool
- Use UpdateDistribution with an Amazon SDK or command line tool

Use `CreateDistribution` with an Amazon SDK or command line tool

The following code examples show how to use `CreateDistribution`.

CLI

Amazon CLI

To create a CloudFront distribution

The following example creates a distribution for an S3 bucket named `awsexamplebucket`, and also specifies `index.html` as the default root object, using command line arguments:

```
aws cloudfront create-distribution \
    --origin-domain-name awsexamplebucket.s3.amazonaws.com \
    --default-root-object index.html
```

Instead of using command line arguments, you can provide the distribution configuration in a JSON file, as shown in the following example:

```
aws cloudfront create-distribution \
    --distribution-config file://dist-config.json
```

The file `dist-config.json` is a JSON document in the current folder that contains the following:

```
{
    "CallerReference": "cli-example",
    "Aliases": {
        "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
                "DomainName": "awsexamplebucket.s3.amazonaws.com",
                "OriginPath": "",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "OriginGroups": {
        "Quantity": 0
    },
    "DefaultCacheBehavior": {
```

```
        "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
            },
            "Headers": {
                "Quantity": 0
            },
            "QueryStringCacheKeys": {
                "Quantity": 0
            }
        },
        "TrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ViewerProtocolPolicy": "allow-all",
        "MinTTL": 0,
        "AllowedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ]
            }
        },
        "SmoothStreaming": false,
        "DefaultTTL": 86400,
        "MaxTTL": 31536000,
        "Compress": false,
        "LambdaFunctionAssociations": {
            "Quantity": 0
        },
        "FieldLevelEncryptionId": ""
    },
    "CacheBehaviors": {
        "Quantity": 0
```

```
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Comment": "",
    "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
}
```

Whether you provide the distribution information with a command line argument or a JSON file, the output is the same:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EMLARXS9EXAMPLE",
    "ETag": "E9LHASXEXAMPLE",
    "Distribution": {
        "Id": "EMLARXS9EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-11-22T00:55:15.705Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
```

```
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
                        "DomainName": "awsexamplebucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-
example",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
```

```
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
```

```
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- For API details, see CreateDistribution 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 uses an Amazon Simple Storage Service (Amazon S3) bucket as a content origin.

After creating the distribution, the code creates a CloudFrontWaiter to wait until the distribution is deployed before returning the distribution.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
```

```
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateDistribution {

        private static final Logger logger =
 LoggerFactory.getLogger(CreateDistribution.class);

        public static Distribution createDistribution(CloudFrontClient
 cloudFrontClient, S3Client s3Client,
                        final String bucketName, final String keyGroupId, final
 String originAccessControlId) {

                final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
                                .get("x-amz-bucket-region").get(0);
                final String originDomain = bucketName + ".s3." + region +
 ".amazonaws.com";
                String originId = originDomain; // Use the originDomain value for
 the originId.

                // The service API requires some deprecated methods, such as
                // DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
                CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                                .distributionConfig(b1 -> b1
                                                .origins(b2 -> b2
                                                                .quantity(1)
                                                                .items(b3 -> b3

 .domainName(originDomain)

 .id(originId)

 .s3OriginConfig(builder4 -> builder4
```

```
                .originAccessIdentity(
                                    ""))
.originAccessControlId(
                originAccessControlId)))
                                                .defaultCacheBehavior(b2 -> b2
.viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
.targetOriginId(originId)
                                                                    .minTTL(200L)
.forwardedValues(b5 -> b5
.cookies(cp -> cp
                .forward(ItemSelection.NONE))
.queryString(true))
.trustedKeyGroups(b3 -> b3
.quantity(1)
.items(keyGroupId)
.enabled(true))
.allowedMethods(b4 -> b4
.quantity(2)
.items(Method.HEAD, Method.GET)
.cachedMethods(b5 -> b5
                .quantity(2)
                .items(Method.HEAD,
                                    Method.GET))))
```

```
                                                .cacheBehaviors(b -> b
                                                                .quantity(1)
                                                                .items(b2 -> b2

.pathPattern("/index.html")

.viewerProtocolPolicy(

              ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3

              .quantity(1)

              .items(keyGroupId)

              .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4

              .cookies(cp -> cp

                              .forward(ItemSelection.NONE))

              .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)

              .items(Method.HEAD,

                             Method.GET)

              .cachedMethods(b6 -> b6

                              .quantity(2)

                              .items(Method.HEAD,

                                                    Method.GET)))))
                                                     .enabled(true)
```

```
                                                .comment("Distribution built with
 java")

 .callerReference(Instant.now().toString())));

                final Distribution distribution =
 createDistResponse.distribution();
                logger.info("Distribution created. DomainName: [{}]  Id: [{}]",
 distribution.domainName(),
                                distribution.id());
                logger.info("Waiting for distribution to be deployed ...");
                try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                        .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                                        .matched();
                        responseOrException.response()
                                        .orElseThrow(() -> new
 RuntimeException("Distribution not created"));
                        logger.info("Distribution deployed. DomainName: [{}]  Id:
 [{}]", distribution.domainName(),
                                        distribution.id());
                }
                return distribution;
        }
}
```

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

PowerShell

Tools for PowerShell

Example 1: Creates a basic CloudFront distribution, configured with logging and caching.

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "ps-cmdlet-sample.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
```

```
New-CFDistribution `
      -DistributionConfig_Enabled $true `
      -DistributionConfig_Comment "Test distribution" `
      -Origins_Item $origin `
      -Origins_Quantity 1 `
      -Logging_Enabled $true `
      -Logging_IncludeCookie $true `
      -Logging_Bucket ps-cmdlet-sample-logging.s3.amazonaws.com `
      -Logging_Prefix "help/" `
      -DistributionConfig_CallerReference Client1 `
      -DistributionConfig_DefaultRootObject index.html `
      -DefaultCacheBehavior_TargetOriginId $origin.Id `
      -ForwardedValues_QueryString $true `
      -Cookies_Forward all `
      -WhitelistedNames_Quantity 0 `
      -TrustedSigners_Enabled $false `
      -TrustedSigners_Quantity 0 `
      -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
      -DefaultCacheBehavior_MinTTL 1000 `
      -DistributionConfig_PriceClass "PriceClass_All" `
      -CacheBehaviors_Quantity 0 `
      -Aliases_Quantity 0
```

- For API details, see CreateDistribution in *Amazon Tools for PowerShell Cmdlet Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateFunction` with an Amazon SDK or command line tool

The following code example shows how to use `CreateFunction`.

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.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**
 * 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 CreateFunction {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <functionName> <filePath>

                Where:
                    functionName - The name of the function to create.\s
                    filePath - The path to a file that contains the application
 logic for the function.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();
```

```
        String funArn = createNewFunction(cloudFrontClient, functionName,
 filePath);
        System.out.println("The function ARN is " + funArn);
        cloudFrontClient.close();
    }

    public static String createNewFunction(CloudFrontClient cloudFrontClient,
 String functionName, String filePath) {
        try {
            InputStream fileIs =
 CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
            SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

            FunctionConfig config = FunctionConfig.builder()
                    .comment("Created by using the CloudFront Java API")
                    .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
                    .build();

            CreateFunctionRequest functionRequest =
 CreateFunctionRequest.builder()
                    .name(functionName)
                    .functionCode(functionCode)
                    .functionConfig(config)
                    .build();

            CreateFunctionResponse response =
 cloudFrontClient.createFunction(functionRequest);
            return response.functionSummary().functionMetadata().functionARN();

        } catch (CloudFrontException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see CreateFunction in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateInvalidation` with an Amazon SDK or command line tool

The following code examples show how to use `CreateInvalidation`.

CLI

Amazon CLI

To create an invalidation for a CloudFront distribution

The following `create-invalidation` example creates an invalidation for the specified files in the specified CloudFront distribution:

```
aws cloudfront create-invalidation \
    --distribution-id EDFDVBD6EXAMPLE \
    --paths "/example-path/example-file.jpg" "/example-path/example-file2.png"
```

Output:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EDFDVBD6EXAMPLE/invalidation/I1JLWSDAP8FU89",
    "Invalidation": {
        "Id": "I1JLWSDAP8FU89",
        "Status": "InProgress",
        "CreateTime": "2019-12-05T18:24:51.407Z",
        "InvalidationBatch": {
            "Paths": {
                "Quantity": 2,
                "Items": [
                    "/example-path/example-file2.png",
                    "/example-path/example-file.jpg"
                ]
            },
            "CallerReference": "cli-1575570291-670203"
        }
    }
}
```

In the previous example, the Amazon CLI automatically generated a random `CallerReference`. To specify your own `CallerReference`, or to avoid passing the invalidation parameters as command line arguments, you can use a JSON file. The following example creates an invalidation for two files, by providing the invalidation parameters in a JSON file named `inv-batch.json`:

```
aws cloudfront create-invalidation \
    --distribution-id EDFDVBD6EXAMPLE \
    --invalidation-batch file://inv-batch.json
```

Contents of `inv-batch.json`:

```
{
    "Paths": {
        "Quantity": 2,
        "Items": [
            "/example-path/example-file.jpg",
            "/example-path/example-file2.png"
        ]
    },
    "CallerReference": "cli-example"
}
```

Output:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EDFDVBD6EXAMPLE/invalidation/I2J0I21PCUYOIK",
    "Invalidation": {
        "Id": "I2J0I21PCUYOIK",
        "Status": "InProgress",
        "CreateTime": "2019-12-05T18:40:49.413Z",
        "InvalidationBatch": {
            "Paths": {
                "Quantity": 2,
                "Items": [
                    "/example-path/example-file.jpg",
                    "/example-path/example-file2.png"
                ]
            },
            "CallerReference": "cli-example"
        }
```

```
    }
}
```

- For API details, see CreateInvalidation in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell

Example 1: This example creates a new invalidation on a distribution with an ID of EXAMPLENSTXAXE. The CallerReference is a unique ID chosen by the user; in this case, a time stamp representing May 15, 2019 at 9:00 a.m. is used. The $Paths variable stores three paths to image and media files that the user does not want as part of the distribution's cache. The -Paths_Quantity parameter value is the total number of paths specified in the -Paths_Item parameter.

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -
Paths_Quantity 3
```

Output:

```
Invalidation                          Location

------------                          --------

Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- For API details, see CreateInvalidation in *Amazon Tools for PowerShell Cmdlet Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateKeyGroup` with an Amazon SDK or command line tool

The following code example shows how to use `CreateKeyGroup`.

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 key group requires at least one public key that is used to verify signed URLs or cookies.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

public class CreateKeyGroup {
    private static final Logger logger =
 LoggerFactory.getLogger(CreateKeyGroup.class);

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
 publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b ->
 b.keyGroupConfig(c -> c
                .items(publicKeyId)
                .name("JavaKeyGroup" + UUID.randomUUID())))
                .keyGroup().id();
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- For API details, see CreateKeyGroup in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreatePublicKey` with an Amazon SDK or command line tool

The following code examples show how to use `CreatePublicKey`.

CLI

Amazon CLI

To create a CloudFront public key

The following example creates a CloudFront public key by providing the parameters in a JSON file named `pub-key-config.json`. Before you can use this command, you must have a PEM-encoded public key. For more information, see Create an RSA Key Pair in the *Amazon CloudFront Developer Guide*.

```
aws cloudfront create-public-key \
    --public-key-config file://pub-key-config.json
```

The file `pub-key-config.json` is a JSON document in the current folder that contains the following. Note that the public key is encoded in PEM format.

```
{
    "CallerReference": "cli-example",
    "Name": "ExampleKey",
    "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPMbCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1UOWcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAzO3ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95ylUQd9LfYTBzVZYG2tSesplcOkjM3\n2Uu
+oMWxQAw1NINnSLPinMVsutJy6ZqlV3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLKOX5Rgb/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nrwIDAQAB\n-----
END PUBLIC KEY-----\n",
    "Comment": "example public key"
}
```

Output:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/public-key/
KDFB19YGCR002",
```

```
    "ETag": "E2QWRUHEXAMPLE",
    "PublicKey": {
        "Id": "KDFB19YGCR002",
        "CreatedTime": "2019-12-05T18:51:43.781Z",
        "PublicKeyConfig": {
            "CallerReference": "cli-example",
            "Name": "ExampleKey",
            "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPMbCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1UOWcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAzO3ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95ylUQd9LfYTBzVZYG2tSesplcOkjM3\n2Uu
+oMWxQAw1NINnSLPinMVsutJy6ZqlV3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLKOX5Rgb/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nrwIDAQAB\n-----
END PUBLIC KEY-----\n",
            "Comment": "example public key"
        }
    }
}
```

- For API details, see CreatePublicKey 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 code example reads in a public key and uploads it to Amazon CloudFront.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
```

```
import java.util.UUID;

public class CreatePublicKey {
    private static final Logger logger =
 LoggerFactory.getLogger(CreatePublicKey.class);

    public static String createPublicKey(CloudFrontClient cloudFrontClient,
 String publicKeyFileName) {
        try (InputStream is =
 CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
                    .createPublicKey(b -> b.publicKeyConfig(c -> c
                            .name("JavaCreatedPublicKey" + UUID.randomUUID())
                            .encodedKey(publicKeyString)
                            .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

- For API details, see CreatePublicKey in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteDistribution` with an Amazon SDK or command line tool

The following code examples show how to use `DeleteDistribution`.

CLI

Amazon CLI

To delete a CloudFront distribution

The following example deletes the CloudFront distribution with the ID EDFDVBD6EXAMPLE. Before you can delete a distribution, you must disable it. To disable a distribution, use the update-distribution command. For more information, see the update-distribution examples.

When a distribution is disabled, you can delete it. To delete a distribution, you must use the `--if-match` option to provide the distribution's ETag. To get the ETag, use the get-distribution or get-distribution-config command.

```
aws cloudfront delete-distribution \
    --id EDFDVBD6EXAMPLE \
    --if-match E2QWRUHEXAMPLE
```

When successful, this command has no output.

- For API details, see DeleteDistribution 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 code example updates a distribution to *disabled*, uses a waiter that waits for the change to be deployed, then deletes the distribution.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

public class DeleteDistribution {
```

```
        private static final Logger logger =
LoggerFactory.getLogger(DeleteDistribution.class);

        public static void deleteDistribution(final CloudFrontClient
cloudFrontClient, final String distributionId) {
                // First, disable the distribution by updating it.
                GetDistributionResponse response =
cloudFrontClient.getDistribution(b -> b
                                .id(distributionId));
                String etag = response.eTag();
                DistributionConfig distConfig =
response.distribution().distributionConfig();

                cloudFrontClient.updateDistribution(builder -> builder
                                .id(distributionId)
                                .distributionConfig(builder1 -> builder1

.cacheBehaviors(distConfig.cacheBehaviors())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                                                        .enabled(false)
                                                        .origins(distConfig.origins())
                                                        .comment(distConfig.comment())

.callerReference(distConfig.callerReference())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())

.priceClass(distConfig.priceClass())
                                                        .aliases(distConfig.aliases())
                                                        .logging(distConfig.logging())

.defaultRootObject(distConfig.defaultRootObject())

.customErrorResponses(distConfig.customErrorResponses())

.httpVersion(distConfig.httpVersion())

.isIPV6Enabled(distConfig.isIPV6Enabled())

.restrictions(distConfig.restrictions())

.viewerCertificate(distConfig.viewerCertificate())
                                                        .webACLId(distConfig.webACLId())
```

```
 .originGroups(distConfig.originGroups()))
                                .ifMatch(etag));

                logger.info("Distribution [{}] is DISABLED, waiting for
 deployment before deleting ...",
                                distributionId);
                GetDistributionResponse distributionResponse;
                try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                        .waitUntilDistributionDeployed(builder ->
 builder.id(distributionId)).matched();
                        distributionResponse = responseOrException.response()
                                        .orElseThrow(() -> new
 RuntimeException("Could not disable distribution"));
                }

                DeleteDistributionResponse deleteDistributionResponse =
 cloudFrontClient
                                .deleteDistribution(builder -> builder
                                                .id(distributionId)

 .ifMatch(distributionResponse.eTag()));
                if (deleteDistributionResponse.sdkHttpResponse().isSuccessful())
 {
                        logger.info("Distribution [{}] DELETED", distributionId);
                }
        }
}
```

- For API details, see DeleteDistribution in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetCloudFrontOriginAccessIdentity` with an Amazon SDK or command line tool

The following code examples show how to use `GetCloudFrontOriginAccessIdentity`.

CLI

Amazon CLI

To get a CloudFront origin access identity

The following example gets the CloudFront origin access identity (OAI) with the ID `E74FTE3AEXAMPLE`, including its `ETag` and the associated S3 canonical ID. The OAI ID is returned in the output of the create-cloud-front-origin-access-identity and list-cloud-front-origin-access-identities commands.

```
aws cloudfront get-cloud-front-origin-access-identity --id E74FTE3AEXAMPLE
```

Output:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "CloudFrontOriginAccessIdentity": {
        "Id": "E74FTE3AEXAMPLE",
        "S3CanonicalUserId":
 "cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
        "CloudFrontOriginAccessIdentityConfig": {
            "CallerReference": "cli-example",
            "Comment": "Example OAI"
        }
    }
}
```

- For API details, see [GetCloudFrontOriginAccessIdentity](#) in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell

Example 1: This example returns a specific Amazon CloudFront origin access identity, specified by the -Id parameter. Although the -Id parameter is not required, if you do not specify it, no results are returned.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

Output:

```
     CloudFrontOriginAccessIdentityConfig    Id
    S3CanonicalUserId
     ------------------------------------    --
    -----------------
     Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
    4b6e...
```

- For API details, see GetCloudFrontOriginAccessIdentity in *Amazon Tools for PowerShell Cmdlet Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetCloudFrontOriginAccessIdentityConfig` with an Amazon SDK or command line tool

The following code examples show how to use `GetCloudFrontOriginAccessIdentityConfig`.

CLI

Amazon CLI

To get a CloudFront origin access identity configuration

The following example gets metadata about the CloudFront origin access identity (OAI) with the ID `E74FTE3AEXAMPLE`, including its `ETag`. The OAI ID is returned in the output of

the create-cloud-front-origin-access-identity and list-cloud-front-origin-access-identities commands.

```
aws cloudfront get-cloud-front-origin-access-identity-config --id E74FTE3AEXAMPLE
```

Output:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "CloudFrontOriginAccessIdentityConfig": {
        "CallerReference": "cli-example",
        "Comment": "Example OAI"
    }
}
```

- For API details, see GetCloudFrontOriginAccessIdentityConfig in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell

Example 1: This example returns configuration information about a single Amazon CloudFront origin access identity, specified by the -Id parameter. Errors occur if no -Id parameter is specified..

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

Output:

```
     CallerReference                                          Comment
     ---------------                                          -------
     mycallerreference: 2/1/2011 1:16:32 PM                   Caller
reference: 2/1/2011 1:16:32 PM
```

- For API details, see GetCloudFrontOriginAccessIdentityConfig in *Amazon Tools for PowerShell Cmdlet Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetDistribution` with an Amazon SDK or command line tool

The following code examples show how to use `GetDistribution`.

CLI

Amazon CLI

To get a CloudFront distribution

The following example gets the CloudFront distribution with the ID `EDFDVBD6EXAMPLE`, including its `ETag`. The distribution ID is returned in the create-distribution and list-distributions commands.

```
aws cloudfront get-distribution --id EDFDVBD6EXAMPLE
```

Output:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "Distribution": {
        "Id": "EDFDVBD6EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
        "Status": "Deployed",
        "LastModifiedTime": "2019-12-04T23:35:41.433Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
```

```
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
                        "DomainName": "awsexamplebucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-
example",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
```

```
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
```

```
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- For API details, see GetDistribution in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell

Example 1: Retrieves the information for a specific distribution.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- For API details, see GetDistribution in *Amazon Tools for PowerShell Cmdlet Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetDistributionConfig` with an Amazon SDK or command line tool

The following code examples show how to use `GetDistributionConfig`.

CLI

Amazon CLI

To get a CloudFront distribution configuration

The following example gets metadata about the CloudFront distribution with the ID EDFDVBD6EXAMPLE, including its ETag. The distribution ID is returned in the create-distribution and list-distributions commands.

```
aws cloudfront get-distribution-config --id EDFDVBD6EXAMPLE
```

Output:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "DistributionConfig": {
        "CallerReference": "cli-example",
        "Aliases": {
            "Quantity": 0
        },
        "DefaultRootObject": "index.html",
        "Origins": {
            "Quantity": 1,
            "Items": [
                {
                    "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
                    "DomainName": "awsexamplebucket.s3.amazonaws.com",
                    "OriginPath": "",
                    "CustomHeaders": {
                        "Quantity": 0
                    },
                    "S3OriginConfig": {
                        "OriginAccessIdentity": ""
                    }
                }
            ]
        },
        "OriginGroups": {
            "Quantity": 0
        },
        "DefaultCacheBehavior": {
            "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
            "ForwardedValues": {
                "QueryString": false,
                "Cookies": {
                    "Forward": "none"
                },
                "Headers": {
                    "Quantity": 0
                },
                "QueryStringCacheKeys": {
                    "Quantity": 0
                }
            },
            "TrustedSigners": {
```

```
                "Enabled": false,
                "Quantity": 0
            },
            "ViewerProtocolPolicy": "allow-all",
            "MinTTL": 0,
            "AllowedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ],
                "CachedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ]
                }
            },
            "SmoothStreaming": false,
            "DefaultTTL": 86400,
            "MaxTTL": 31536000,
            "Compress": false,
            "LambdaFunctionAssociations": {
                "Quantity": 0
            },
            "FieldLevelEncryptionId": ""
        },
        "CacheBehaviors": {
            "Quantity": 0
        },
        "CustomErrorResponses": {
            "Quantity": 0
        },
        "Comment": "",
        "Logging": {
            "Enabled": false,
            "IncludeCookies": false,
            "Bucket": "",
            "Prefix": ""
        },
        "PriceClass": "PriceClass_All",
        "Enabled": true,
        "ViewerCertificate": {
```

```
            "CloudFrontDefaultCertificate": true,
            "MinimumProtocolVersion": "TLSv1",
            "CertificateSource": "cloudfront"
        },
        "Restrictions": {
            "GeoRestriction": {
                "RestrictionType": "none",
                "Quantity": 0
            }
        },
        "WebACLId": "",
        "HttpVersion": "http2",
        "IsIPV6Enabled": true
    }
}
```

- For API details, see GetDistributionConfig in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell

Example 1: Retrieves the configuration for a specific distribution.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- For API details, see GetDistributionConfig in *Amazon Tools for PowerShell Cmdlet 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 CloudFrontWrapper:
```

```
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- For API details, see GetDistributionConfig in *Amazon SDK for Python (Boto3) API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListCloudFrontOriginAccessIdentities` with an Amazon SDK or command line tool

The following code examples show how to use `ListCloudFrontOriginAccessIdentities`.

CLI

Amazon CLI

To list CloudFront origin access identities

The following example gets a list of the CloudFront origin access identities (OAIs) in your Amazon account:

```
aws cloudfront list-cloud-front-origin-access-identities
```

Output:

```
{
    "CloudFrontOriginAccessIdentityList": {
        "Items": [
            {
                "Id": "E74FTE3AEXAMPLE",
                "S3CanonicalUserId":
 "cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
                "Comment": "Example OAI"
            },
            {
                "Id": "EH1HDMBEXAMPLE",
                "S3CanonicalUserId":
 "1489f6f2e6faacaae7ff64c4c3e6956c24f78788abfc1718c3527c263bf7a17EXAMPLE",
                "Comment": "Test OAI"
            },
            {
                "Id": "E2X2C9TEXAMPLE",
                "S3CanonicalUserId":
 "cbfeebb915a64749f9be546a45b3fcfd3a31c779673c13c4dd460911ae402c2EXAMPLE",
                "Comment": "Example OAI #2"
            }
        ]
    }
}
```

- For API details, see ListCloudFrontOriginAccessIdentities in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell

Example 1: This example returns a list of Amazon CloudFront origin access identities. Because the -MaxItem parameter specifies a value of 2, the results include two identities.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

Output:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- For API details, see ListCloudFrontOriginAccessIdentities in *Amazon Tools for PowerShell Cmdlet Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListDistributions` with an Amazon SDK or command line tool

The following code examples show how to use `ListDistributions`.

CLI

Amazon CLI

To list CloudFront distributions

The following example gets a list of the CloudFront distributions in your Amazon account:

```
aws cloudfront list-distributions
```

Output:

```
{
    "DistributionList": {
        "Items": [
            {
                "Id": "EMLARXS9EXAMPLE",
                "ARN": "arn:aws:cloudfront::123456789012:distribution/
EMLARXS9EXAMPLE",
                "Status": "InProgress",
                "LastModifiedTime": "2019-11-22T00:55:15.705Z",
                "InProgressInvalidationBatches": 0,
                "DomainName": "d111111abcdef8.cloudfront.net",
                "ActiveTrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "DistributionConfig": {
                    "CallerReference": "cli-example",
                    "Aliases": {
                        "Quantity": 0
                    },
                    "DefaultRootObject": "index.html",
                    "Origins": {
                        "Quantity": 1,
                        "Items": [
                            {
                                "Id": "awsexamplebucket.s3.amazonaws.com-cli-
example",
                                "DomainName":
 "awsexamplebucket.s3.amazonaws.com",
                                "OriginPath": "",
                                "CustomHeaders": {
                                    "Quantity": 0
                                },
                                "S3OriginConfig": {
                                    "OriginAccessIdentity": ""
                                }
                            }
                        ]
                    },
```

```
                    "OriginGroups": {
                        "Quantity": 0
                    },
                    "DefaultCacheBehavior": {
                        "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-
example",
                        "ForwardedValues": {
                            "QueryString": false,
                            "Cookies": {
                                "Forward": "none"
                            },
                            "Headers": {
                                "Quantity": 0
                            },
                            "QueryStringCacheKeys": {
                                "Quantity": 0
                            }
                        },
                        "TrustedSigners": {
                            "Enabled": false,
                            "Quantity": 0
                        },
                        "ViewerProtocolPolicy": "allow-all",
                        "MinTTL": 0,
                        "AllowedMethods": {
                            "Quantity": 2,
                            "Items": [
                                "HEAD",
                                "GET"
                            ],
                            "CachedMethods": {
                                "Quantity": 2,
                                "Items": [
                                    "HEAD",
                                    "GET"
                                ]
                            }
                        },
                        "SmoothStreaming": false,
                        "DefaultTTL": 86400,
                        "MaxTTL": 31536000,
                        "Compress": false,
                        "LambdaFunctionAssociations": {
                            "Quantity": 0
```

```
                        },
                        "FieldLevelEncryptionId": ""
                    },
                    "CacheBehaviors": {
                        "Quantity": 0
                    },
                    "CustomErrorResponses": {
                        "Quantity": 0
                    },
                    "Comment": "",
                    "Logging": {
                        "Enabled": false,
                        "IncludeCookies": false,
                        "Bucket": "",
                        "Prefix": ""
                    },
                    "PriceClass": "PriceClass_All",
                    "Enabled": true,
                    "ViewerCertificate": {
                        "CloudFrontDefaultCertificate": true,
                        "MinimumProtocolVersion": "TLSv1",
                        "CertificateSource": "cloudfront"
                    },
                    "Restrictions": {
                        "GeoRestriction": {
                            "RestrictionType": "none",
                            "Quantity": 0
                        }
                    },
                    "WebACLId": "",
                    "HttpVersion": "http2",
                    "IsIPV6Enabled": true
                }
            },
            {
                "Id": "EDFDVBD6EXAMPLE",
                "ARN": "arn:aws:cloudfront::123456789012:distribution/
EDFDVBD6EXAMPLE",
                "Status": "InProgress",
                "LastModifiedTime": "2019-12-04T23:35:41.433Z",
                "InProgressInvalidationBatches": 0,
                "DomainName": "d930174dauwrn8.cloudfront.net",
                "ActiveTrustedSigners": {
                    "Enabled": false,
```

```
                    "Quantity": 0
                },
                "DistributionConfig": {
                    "CallerReference": "cli-example",
                    "Aliases": {
                        "Quantity": 0
                    },
                    "DefaultRootObject": "index.html",
                    "Origins": {
                        "Quantity": 1,
                        "Items": [
                            {
                                "Id": "awsexamplebucket1.s3.amazonaws.com-cli-
example",
                                "DomainName":
 "awsexamplebucket1.s3.amazonaws.com",
                                "OriginPath": "",
                                "CustomHeaders": {
                                    "Quantity": 0
                                },
                                "S3OriginConfig": {
                                    "OriginAccessIdentity": ""
                                }
                            }
                        ]
                    },
                    "OriginGroups": {
                        "Quantity": 0
                    },
                    "DefaultCacheBehavior": {
                        "TargetOriginId": "awsexamplebucket1.s3.amazonaws.com-
cli-example",
                        "ForwardedValues": {
                            "QueryString": false,
                            "Cookies": {
                                "Forward": "none"
                            },
                            "Headers": {
                                "Quantity": 0
                            },
                            "QueryStringCacheKeys": {
                                "Quantity": 0
                            }
                        },
```

```
                        "TrustedSigners": {
                            "Enabled": false,
                            "Quantity": 0
                        },
                        "ViewerProtocolPolicy": "allow-all",
                        "MinTTL": 0,
                        "AllowedMethods": {
                            "Quantity": 2,
                            "Items": [
                                "HEAD",
                                "GET"
                            ],
                            "CachedMethods": {
                                "Quantity": 2,
                                "Items": [
                                    "HEAD",
                                    "GET"
                                ]
                            }
                        },
                        "SmoothStreaming": false,
                        "DefaultTTL": 86400,
                        "MaxTTL": 31536000,
                        "Compress": false,
                        "LambdaFunctionAssociations": {
                            "Quantity": 0
                        },
                        "FieldLevelEncryptionId": ""
                    },
                    "CacheBehaviors": {
                        "Quantity": 0
                    },
                    "CustomErrorResponses": {
                        "Quantity": 0
                    },
                    "Comment": "",
                    "Logging": {
                        "Enabled": false,
                        "IncludeCookies": false,
                        "Bucket": "",
                        "Prefix": ""
                    },
                    "PriceClass": "PriceClass_All",
                    "Enabled": true,
```

```
                    "ViewerCertificate": {
                        "CloudFrontDefaultCertificate": true,
                        "MinimumProtocolVersion": "TLSv1",
                        "CertificateSource": "cloudfront"
                    },
                    "Restrictions": {
                        "GeoRestriction": {
                            "RestrictionType": "none",
                            "Quantity": 0
                        }
                    },
                    "WebACLId": "",
                    "HttpVersion": "http2",
                    "IsIPV6Enabled": true
                }
            },
            {
                "Id": "E1X5IZQEXAMPLE",
                "ARN": "arn:aws:cloudfront::123456789012:distribution/
E1X5IZQEXAMPLE",
                "Status": "Deployed",
                "LastModifiedTime": "2019-11-06T21:31:48.864Z",
                "DomainName": "d2e04y12345678.cloudfront.net",
                "Aliases": {
                    "Quantity": 0
                },
                "Origins": {
                    "Quantity": 1,
                    "Items": [
                        {
                            "Id": "awsexamplebucket2",
                            "DomainName": "awsexamplebucket2.s3.us-
west-2.amazonaws.com",
                            "OriginPath": "",
                            "CustomHeaders": {
                                "Quantity": 0
                            },
                            "S3OriginConfig": {
                                "OriginAccessIdentity": ""
                            }
                        }
                    ]
                },
                "OriginGroups": {
```

```
                    "Quantity": 0
                },
                "DefaultCacheBehavior": {
                    "TargetOriginId": "awsexamplebucket2",
                    "ForwardedValues": {
                        "QueryString": false,
                        "Cookies": {
                            "Forward": "none"
                        },
                        "Headers": {
                            "Quantity": 0
                        },
                        "QueryStringCacheKeys": {
                            "Quantity": 0
                        }
                    },
                    "TrustedSigners": {
                        "Enabled": false,
                        "Quantity": 0
                    },
                    "ViewerProtocolPolicy": "allow-all",
                    "MinTTL": 0,
                    "AllowedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ],
                        "CachedMethods": {
                            "Quantity": 2,
                            "Items": [
                                "HEAD",
                                "GET"
                            ]
                        }
                    },
                    "SmoothStreaming": false,
                    "DefaultTTL": 86400,
                    "MaxTTL": 31536000,
                    "Compress": false,
                    "LambdaFunctionAssociations": {
                        "Quantity": 0
                    },
                    "FieldLevelEncryptionId": ""
```

```
                },
                "CacheBehaviors": {
                    "Quantity": 0
                },
                "CustomErrorResponses": {
                    "Quantity": 0
                },
                "Comment": "",
                "PriceClass": "PriceClass_All",
                "Enabled": true,
                "ViewerCertificate": {
                    "CloudFrontDefaultCertificate": true,
                    "MinimumProtocolVersion": "TLSv1",
                    "CertificateSource": "cloudfront"
                },
                "Restrictions": {
                    "GeoRestriction": {
                        "RestrictionType": "none",
                        "Quantity": 0
                    }
                },
                "WebACLId": "",
                "HttpVersion": "HTTP1_1",
                "IsIPV6Enabled": true
            }
        ]
    }
}
```

- For API details, see ListDistributions in *Amazon CLI Command Reference*.

PowerShell

Tools for PowerShell

Example 1: Returns distributions.

```
Get-CFDistributionList
```

- For API details, see ListDistributions in *Amazon Tools for PowerShell Cmdlet 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 CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def list_distributions(self):
        print("CloudFront distributions:\n")
        distributions = self.cloudfront_client.list_distributions()
        if distributions["DistributionList"]["Quantity"] > 0:
            for distribution in distributions["DistributionList"]["Items"]:
                print(f"Domain: {distribution['DomainName']}")
                print(f"Distribution Id: {distribution['Id']}")
                print(
                    f"Certificate Source: "
                    f"{distribution['ViewerCertificate']['CertificateSource']}"
                )
                if distribution["ViewerCertificate"]["CertificateSource"] ==
 "acm":
                    print(
                        f"Certificate: {distribution['ViewerCertificate']
['Certificate']}"
                    )
                print("")
        else:
            print("No CloudFront distributions detected.")
```

- For API details, see ListDistributions in *Amazon SDK for Python (Boto3) API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UpdateDistribution` with an Amazon SDK or command line tool

The following code examples show how to use `UpdateDistribution`.

CLI

Amazon CLI

To update a CloudFront distribution's default root object

The following example updates the default root object to `index.html` for the CloudFront distribution with the ID `EDFDVBD6EXAMPLE`:

```
aws cloudfront update-distribution --id EDFDVBD6EXAMPLE \
    --default-root-object index.html
```

Output:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "Distribution": {
        "Id": "EDFDVBD6EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-12-06T18:55:39.870Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "6b10378d-49be-4c4b-a642-419ccaf8f3b5",
            "Aliases": {
                "Quantity": 0
            },
```

```
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "example-website",
                        "DomainName": "www.example.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "CustomOriginConfig": {
                            "HTTPPort": 80,
                            "HTTPSPort": 443,
                            "OriginProtocolPolicy": "match-viewer",
                            "OriginSslProtocols": {
                                "Quantity": 2,
                                "Items": [
                                    "SSLv3",
                                    "TLSv1"
                                ]
                            },
                            "OriginReadTimeout": 30,
                            "OriginKeepaliveTimeout": 5
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "example-website",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 1,
                        "Items": [
                            "*"
                        ]
                    },
```

```
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
```

```
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http1.1",
            "IsIPV6Enabled": true
        }
    }
}
```

To update a CloudFront distribution

The following example disables the CloudFront distribution with the ID `EMLARXS9EXAMPLE` by providing the distribution configuration in a JSON file named `dist-config-disable.json`. To update a distribution, you must use the `--if-match` option to provide the distribution's `ETag`. To get the `ETag`, use the get-distribution or get-distribution-config command.

After you use the following example to disable a distribution, you can use the delete-distribution command to delete it.

```
aws cloudfront update-distribution \
    --id EMLARXS9EXAMPLE \
    --if-match E2QWRUHEXAMPLE \
    --distribution-config file://dist-config-disable.json
```

The file `dist-config-disable.json` is a JSON document in the current folder that contains the following. Note that the `Enabled` field is set to `false`:

```
{
```

```
    "CallerReference": "cli-1574382155-496510",
    "Aliases": {
        "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "awsexamplebucket.s3.amazonaws.com-1574382155-273939",
                "DomainName": "awsexamplebucket.s3.amazonaws.com",
                "OriginPath": "",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "OriginGroups": {
        "Quantity": 0
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-1574382155-273939",
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
            },
            "Headers": {
                "Quantity": 0
            },
            "QueryStringCacheKeys": {
                "Quantity": 0
            }
        },
        "TrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ViewerProtocolPolicy": "allow-all",
        "MinTTL": 0,
```

```
        "AllowedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ]
            }
        },
        "SmoothStreaming": false,
        "DefaultTTL": 86400,
        "MaxTTL": 31536000,
        "Compress": false,
        "LambdaFunctionAssociations": {
            "Quantity": 0
        },
        "FieldLevelEncryptionId": ""
    },
    "CacheBehaviors": {
        "Quantity": 0
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Comment": "",
    "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": false,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
```

```
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
}
```

Output:

```
{
    "ETag": "E9LHASXEXAMPLE",
    "Distribution": {
        "Id": "EMLARXS9EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-12-06T18:32:35.553Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-1574382155-496510",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id":
 "awsexamplebucket.s3.amazonaws.com-1574382155-273939",
                        "DomainName": "awsexamplebucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
```

```
                        "OriginAccessIdentity": ""
                    }
                }
            ]
        },
        "OriginGroups": {
            "Quantity": 0
        },
        "DefaultCacheBehavior": {
            "TargetOriginId":
"awsexamplebucket.s3.amazonaws.com-1574382155-273939",
            "ForwardedValues": {
                "QueryString": false,
                "Cookies": {
                    "Forward": "none"
                },
                "Headers": {
                    "Quantity": 0
                },
                "QueryStringCacheKeys": {
                    "Quantity": 0
                }
            },
            "TrustedSigners": {
                "Enabled": false,
                "Quantity": 0
            },
            "ViewerProtocolPolicy": "allow-all",
            "MinTTL": 0,
            "AllowedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ],
                "CachedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ]
                }
            },
            "SmoothStreaming": false,
```

```
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": false,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- For API details, see UpdateDistribution 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.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import
 software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * 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 ModifyDistribution {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <id>\s

                Where:
                    id - the id value of the distribution.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String id = args[0];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        modDistribution(cloudFrontClient, id);
        cloudFrontClient.close();
    }

    public static void modDistribution(CloudFrontClient cloudFrontClient, String
idVal) {
        try {
            // Get the Distribution to modify.
            GetDistributionRequest disRequest = GetDistributionRequest.builder()
                    .id(idVal)
                    .build();

            GetDistributionResponse response =
cloudFrontClient.getDistribution(disRequest);
            Distribution disObject = response.distribution();
            DistributionConfig config = disObject.distributionConfig();

            // Create a new DistributionConfig object and add new values to
comment and
            // aliases
            DistributionConfig config1 = DistributionConfig.builder()
                    .aliases(config.aliases()) // You can pass in new values here
                    .comment("New Comment")
                    .cacheBehaviors(config.cacheBehaviors())
                    .priceClass(config.priceClass())
                    .defaultCacheBehavior(config.defaultCacheBehavior())
                    .enabled(config.enabled())
                    .callerReference(config.callerReference())
                    .logging(config.logging())
                    .originGroups(config.originGroups())
                    .origins(config.origins())
                    .restrictions(config.restrictions())
                    .defaultRootObject(config.defaultRootObject())
                    .webACLId(config.webACLId())
                    .httpVersion(config.httpVersion())
                    .viewerCertificate(config.viewerCertificate())
```

```
                    .customErrorResponses(config.customErrorResponses())
                    .build();

            UpdateDistributionRequest updateDistributionRequest =
 UpdateDistributionRequest.builder()
                    .distributionConfig(config1)
                    .id(disObject.id())
                    .ifMatch(response.eTag())
                    .build();

            cloudFrontClient.updateDistribution(updateDistributionRequest);

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

- For API details, see UpdateDistribution 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 CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client
```

```
    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- For API details, see UpdateDistribution in *Amazon SDK for Python (Boto3) API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Scenarios for CloudFront using Amazon SDKs

The following code examples show you how to implement common scenarios in CloudFront with Amazon SDKs. These scenarios show you how to accomplish specific tasks by calling multiple functions within CloudFront. Each scenario includes a link to GitHub, where you can find instructions on how to set up and run the code.

Examples

- Delete CloudFront signing resources using Amazon SDK
- Create signed URLs and cookies using an Amazon SDK

Delete CloudFront signing resources using Amazon SDK

The following code example shows how to delete resources that are used to gain access to restricted content in 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.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

public class DeleteSigningResources {
    private static final Logger logger =
 LoggerFactory.getLogger(DeleteSigningResources.class);

    public static void deleteOriginAccessControl(final CloudFrontClient
 cloudFrontClient,
            final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
                .getOriginAccessControl(b -> b.id(originAccessControlId));
```

```
        DeleteOriginAccessControlResponse deleteResponse =
 cloudFrontClient.deleteOriginAccessControl(builder -> builder
                .id(originAccessControlId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Origin Access Control [{}]",
 originAccessControlId);
        }
    }

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient,
 final String keyGroupId) {

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
 b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
 cloudFrontClient.deleteKeyGroup(builder -> builder
                .id(keyGroupId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
 final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
 b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
 cloudFrontClient.deletePublicKey(builder -> builder
                .id(publicKeyId)
                .ifMatch(getResponse.eTag()));

        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.
  - DeleteKeyGroup

- DeleteOriginAccessControl
- DeletePublicKey

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Create signed URLs and cookies using an Amazon SDK

The following code example shows how to create signed URLs and cookies that allow access to restricted 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.

Use the CannedSignerRequest class to sign URLs or cookies with a *canned* policy.

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;
```

```
        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expirationDate)
                .build();
    }
}
```

Use the CustomSignerRequest class to sign URLs or cookies with a *custom* policy. The `activeDate` and `ipRange` are optional methods.

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);
```

```
        return CustomSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expireDate)
                .activeDate(activeDate) // Optional.
                // .ipRange("192.168.0.1/24") // Optional.
                .build();
    }
}
```

The following example demonstrates the use of the CloudFrontUtilities class to produce signed cookies and URLs. View this code example on GitHub.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
 LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
 CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
 cannedSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
 customSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
```

```
        return signedUrl;
    }

    public static CookiesForCannedPolicy
 getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
                .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
 cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCannedPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }

    public static CookiesForCustomPolicy
 getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
        CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
                .getCookiesForCustomPolicy(customSignerRequest);
        logger.info("Cookie POLICY header [{}]",
 cookiesForCustomPolicy.policyHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCustomPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCustomPolicy.signatureHeaderValue());
        return cookiesForCustomPolicy;
    }
}
```

- For API details, see CloudFrontUtilities in *Amazon SDK for Java 2.x API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using CloudFront with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Amazon CloudFront related information

The information and resources listed here can help you learn more about CloudFront.

Topics

- Additional Amazon CloudFront documentation
- Getting support
- CloudFront developer tools and SDKs
- Tips from the Amazon Web Services blog

Additional Amazon CloudFront documentation

The following related resources can help you as you work with this service.

- Amazon CloudFront API Reference – Gives complete descriptions of the API actions, parameters, and data types, and a list of errors that the service returns.
- CloudFront What's New – Announcements of new CloudFront features and recently added edge locations.
- Amazon CloudFront product information – The primary web page for information about CloudFront, including features and pricing information.

Getting support

Support for CloudFront is available in a number of forms.

- Amazon re:Post – A community-based question and answer site for developers to discuss technical questions related to CloudFront.
- Amazon Support Center – This site brings together information about your recent support cases and results from Amazon Trusted Advisor and health checks, as well as providing links to discussion forums, technical FAQs, the service health dashboard, and information about Amazon support plans.
- Amazon Premium Support – The primary web page for information about Amazon Premium Support, a one-on-one, fast-response support channel to help you build and run applications on Amazon.

- Amazon IQ – Get help from Amazon Certified professionals and experts.
- Contact Us – Links for inquiring about your billing or account. For technical questions, use the discussion forums or support links above.

CloudFront developer tools and SDKs

See the Tools page for links to developer resources that provide documentation, code samples, release notes, and other information to help you build innovative applications with Amazon.

In addition, Amazon Web Services provides software development kits for accessing CloudFront programmatically. The SDK libraries automate a number of common tasks, including cryptographically signing your service requests, retrying requests, and handling error responses.

Tips from the Amazon Web Services blog

The Amazon Blog has a number of posts to help you use CloudFront, in the Networking & Content Delivery category.

Document history

The following table describes the important changes made to CloudFront documentation. For notification of updates, you can subscribe to the RSS feed.

Change	Description	Date
Added origin access control support	You can now create an origin access control (OAC) for AWS Elemental MediaPackage V2 and Amazon Lambda function URL.	April 11, 2024
Real-time log fields for CMCD	Added 18 common media client data (CMCD) fields for real-time logging.	April 9, 2024
Getting started with a basic CloudFront distribution	Updated tutorial for a basic distribution that uses an Amazon S3 origin with origin access control (OAC).	March 18, 2024
Code examples for CloudFront using Amazon SDKs	Added code examples that show how to use CloudFront with an Amazon software development kit (SDK). The examples are divided into code excerpts that show you how to call individual service functions and examples that show you how to accomplish a specific task by calling multiple functions within the same service.	February 16, 2024
Amazon managed policy update	The `CloudFrontReadOnlyAccess` and `CloudFron`	December 19, 2023

	`tFullAccess` IAM policies now support `KeyValueS tore` operations.	
JavaScript runtime 2.0	Added JavaScript runtime 2.0 features for CloudFront Functions.	November 21, 2023
CloudFront KeyValueStore	Amazon CloudFront now supports CloudFront KeyValueStore. This feature is a secure, global, low-laten cy key value datastore that allows read access from within CloudFront Functions, enabling advanced customiza ble logic at the CloudFront edge locations.	November 21, 2023
Lambda@Edge supports newer runtime version	Lambda@Edge now supports Lambda functions with the Node.js 20 runtime.	November 15, 2023
Security dashboard	CloudFront creates a security dashboard when you create a distribution. Enable Amazon WAF, manage geo restrictions, and view high-level data for requests, bots, and logs.	November 8, 2023
Sorting query strings in functions	CloudFront now supports query string sorting using CloudFront Functions	October 3, 2023
Amazon WAF security recommendations	Amazon CloudFront now displays Amazon WAF security recommendations on the CloudFront console.	September 26, 2023

Support for serving stale (expired) cache content	CloudFront supports the `Stale-While-Revalidate` and `Stale-If-Error` cache control directives.	May 15, 2023
Enable Amazon WAF protections with one click	A streamlined method for adding Amazon WAF security protections to CloudFront distributions.	May 10, 2023
Enable ACLs for new S3 buckets used for standard logs	Added note and links to address the default ACL setting for new S3 buckets.	April 11, 2023
Create an origin using Amazon S3 Object Lambda	You can use an Amazon S3 Object Lambda Access Point alias as an origin for your distribution.	March 31, 2023
Customize HTTP status and body using CloudFront Functions	You can use CloudFront Functions to update the viewer response status code and replace or remove the response body.	March 29, 2023
Added CORS headers wildcard options for ports	You can now include wildcard configurations for ports in CORS access-control headers.	March 20, 2023
Added new link for the Amazon Security Hub User Guide	Updated language and added link to the reorganized Amazon CloudFront controls in the Amazon Security Hub User Guide.	March 9, 2023

CloudFront now supports block lists ("all except") in origin request policies	Use block lists in origin request policies to include all query strings, HTTP headers, or cookies, ***except*** for the ones specified, in requests that CloudFront sends to the origin.	February 22, 2023
CloudFront adds a new managed origin request policy to forward all viewer headers except the Host header	Use CloudFront's new managed origin request policy to include all headers from the viewer request, ***except*** for the `Host` header, in requests that CloudFront sends to the origin.	February 22, 2023
Updated restrictions on Lambda@Edge	Lambda@Edge supports Lambda runtime management configurations set to **Auto**.	February 16, 2023
Updated the IAM guidance for CloudFront	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM.	February 15, 2023
Enhanced security with origin access control	You can now secure MediaStore origins by permitting access to only the designated CloudFront distributions.	February 9, 2023
New headers for determining viewer's header structure	You can now add header order and header count to help identify the viewer based on the headers that it sends.	January 13, 2023

Lambda@Edge supports newer runtime version	Lambda@Edge now supports Lambda functions with the Node.js 18 runtime.	January 12, 2023
Remove response headers using a response headers policy	You can now use a CloudFront response headers policy to remove headers that CloudFront received in the response from the origin. The specified headers are not included in the response that CloudFront sends to viewers.	January 3, 2023
New managed origin request policy	Added the `AllViewerAndCloudFrontHeaders-2022-06` origin access policy.	December 2, 2022
Continuous deployment for safely testing configuration changes	You can now deploy changes to your CDN configuration by testing with a subset of production traffic.	November 18, 2022
Release of `CloudFront-Viewer-JA3-Fingerprint` header	You can now use the JA3 fingerprint to help determine whether the request comes from a known client.	November 16, 2022
Added CORS headers wildcard options	You can now use various wildcard configurations in some CORS access-control headers.	November 11, 2022
Additional metrics for CloudFront distributions	Support for `MonitoringSubscription` in the CloudFront API and Amazon CloudFormation.	October 3, 2022

Enhanced security with origin access control	You can now secure Amazon S3 origins by permitting access to only the designated CloudFront distributions.	August 24, 2022
HTTP/3 support for CloudFront distributions	You can now choose HTTP/3 for your CloudFront distribution.	August 15, 2022
Add handshake details to CloudFront-Viewer-TLS header	You can new view information about the SSL/TLS handshake used.	June 27, 2022
New metric in Server-Timing header	Added the new `cdn-downstream-fbl` metric to `Server-Timing` headers.	June 13, 2022
New header to get information about TLS version and cipher	You can now use the `CloudFront-Viewer-TLS` header to get information about the version of TLS (or SSL) and the cipher that was used for the connection between the viewer and CloudFront.	May 23, 2022
New FunctionThrottles metric for CloudFront Functions	With Amazon CloudWatch, you can now monitor the number of times that a CloudFront Function was throttled in a given time period.	May 4, 2022
CloudFront supports Lambda function URLs	If you build a serverless web application using Lambda functions with function URLs, you can now add CloudFront for an array of benefits.	April 6, 2022

Server-Timing header in HTTP responses	You can now enable the `Server-Timing` header in HTTP responses sent from CloudFront to view metrics that can help you gain insights about the behavior and performance of CloudFront.	March 30, 2022
Use Amazon-managed prefix list to limit inbound traffic	You can now limit the inbound HTTP and HTTPS traffic to your origins from only the IP addresses that belong to CloudFront's origin-facing servers.	February 7, 2022

For earlier entries, see Updates before 2022.

Updates before 2022

The following table describes the important changes made to CloudFront documentation before 2022.

Change	Description	Date
New feature	CloudFront adds support for *response headers policies*, which allow you to specify the HTTP headers that CloudFront adds to HTTP responses that it sends to viewers (web browsers or other clients). You can specify the desired headers (and their values) without making any changes to the origin or writing any code. For more information, see the section called "Adding or removing response headers".	November 2, 2021
New `CloudFront-Viewer-`	CloudFront adds support for a new header, `CloudFront-Viewer-Address` , that contains the IP address of the viewer	October 25, 2021

Change	Description	Date
`Address` request header	that sent the HTTP request to CloudFront. For more informati on, see the section called “Adding CloudFront request headers”.	
Lambda@Ed ge supports new runtime version	Lambda@Edge now supports Lambda functions with the Python 3.9 runtime. For more information, see the section called “Supported runtimes”.	September 22, 2021
Amazon managed policy update	CloudFront updated the **CloudFrontReadOnlyAccess** policy. For more information, see the section called “Policy updates”.	September 8, 2021
New feature	CloudFront now supports ECDSA certificates for viewer-facing HTTPS connections. For more information, see the section called “Supported protocols and ciphers between viewers and CloudFront” and the section called “Requirements for using SSL/TLS certificates with CloudFront”.	July 14, 2021
New feature	CloudFront now supports more ways to move an alternate domain name from one distribution to another, without contacting Amazon Web Services Support. For more informati on, see the section called “Moving an alternate domain name to a different distribution”.	July 7, 2021
New security policy	CloudFront now supports a new security policy, **TLSv1.2_2021**, with a smaller set of supported ciphers. For more informati on, see Supported protocols and ciphers between viewers and CloudFront.	June 23, 2021
New feature	Amazon CloudFront now supports CloudFront Functions , a native feature of CloudFront that enables you to write lightweight functions in JavaScript for high-scale, latency-sensitive CDN customizations. For more information, see Customizing at the edge with CloudFront Functions.	May 3, 2021

Change	Description	Date
Lambda@Edge supports newer runtime versions	Lambda@Edge now supports Lambda functions with the Node.js 14 runtime. For more information, see Supported runtimes.	April 29, 2021
Remove documentation for RTMP distributions	Amazon CloudFront deprecated real-time messaging protocol (RTMP) distributions on December 31, 2020. Documentation for RTMP distributions is now removed from the *Amazon CloudFront Developer Guide*.	February 10, 2021
New pricing option	Amazon CloudFront introduces the *CloudFront security savings bundle*, a simple way to save up to 30% on the CloudFront charges on your Amazon bill. For more information, see the Savings Bundle FAQs.	February 5, 2021
New tutorial	The *Amazon CloudFront Developer Guide* now includes a tutorial for using Amazon CloudFront to restrict access to an Application Load Balancer in Elastic Load Balancing. For more information, see Restricting access to Application Load Balancers.	December 18, 2020
New option for public key management	CloudFront now supports public key management for signed URLs and signed cookies through the CloudFront console and API, without requiring access to the Amazon account root user. For more information, see Specifying the signers that can create signed URLs and signed cookies.	October 22, 2020
New feature – Origin Shield	CloudFront now supports CloudFront Origin Shield, an additional layer in the CloudFront caching infrastructure that helps to minimize your origin's load, improve its availability, and reduce its operating costs. For more information, see Using Amazon CloudFront Origin Shield.	October 20, 2020

Change	Description	Date
New compression format	CloudFront now supports the Brotli compression formation when you configure CloudFront to compress objects at CloudFront edge locations. You can also configure CloudFront to cache Brotli objects using a normalized `Accept-Encoding` header. For more information, see Serving compressed files and Compression support.	September 14, 2020
New TLS protocol	CloudFront now supports the TLS 1.3 protocol for HTTPS connections between viewers and CloudFront distributions. TLS 1.3 is enabled by default in all CloudFront security policies. For more information, see Supported protocols and ciphers between viewers and CloudFront.	September 3, 2020
New real-time logs	CloudFront now supports configurable real-time logs. With real-time logs, you can get information about requests made to a distribution in real time. You can use real-time logs to monitor, analyze, and take action based on content delivery performance. For more information, see Real-time logs.	August 31, 2020
API support for additional metrics	CloudFront now supports enabling eight additional real-time metrics with the CloudFront API. For more information, see Turning on additional metrics.	August 28, 2020
New CloudFront HTTP headers	CloudFront added additional HTTP headers for determining information about the viewer such as device type, geographic location, and more. For more information, see the section called “Adding CloudFront request headers”.	July 23, 2020
New feature	CloudFront now supports *cache policies* and *origin request polices*, which give you more granular control over the *cache key* and *origin requests* for your CloudFront distributions. For more information, see Working with policies.	July 22, 2020

Change	Description	Date
New security policy	CloudFront now supports a new security policy, **TLSv1.2_2019**, with a smaller set of supported ciphers. For more informati on, see Supported protocols and ciphers between viewers and CloudFront.	July 8, 2020
New settings to control origin timeouts and attempts	CloudFront added new settings that control origin timeouts and attempts. For more information, see Controlling origin timeouts and attempts.	June 5, 2020
New documenta tion for getting started with CloudFront by creating a secure static website	Get started with CloudFront by creating a secure static website using Amazon S3, CloudFront, Lambda@Edge, and more, all deployed with Amazon CloudFormation. For more information, see Getting started with a secure static website.	June 2, 2020
Lambda@Ed ge supports newer runtime versions	Lambda@Edge now supports Lambda functions with the Node.js 12 and Python 3.8 runtimes. For more information, see Supported runtimes.	February 27, 2020
New real-time metrics in CloudWatc h	Amazon CloudFront now offers eight additional real-time metrics in Amazon CloudWatch. For more information, see Turning on additional CloudFront distribution metrics.	December 19, 2019
New fields in access logs	CloudFront adds seven new fields to access logs. For more information, see Standard log file fields.	December 12, 2019

Change	Description	Date
Amazon WordPress plugin	You can use the Amazon WordPress plugin to provide visitors to your WordPress website an accelerated viewing experienc e using CloudFront. (Update: as of September 30, 2022, the Amazon for WordPress plugin is deprecated.)	October 30, 2019
Tag-based and resource-level IAM permissions policies	CloudFront now supports two additional ways of specifying IAM permission policies: tag-based and resource-level policy permissions. For more information, see Managing Access to Resources.	August 8, 2019
Support for Python programmi ng language	You can now use the Python programming language to develop functions in Lambda@Edge, in addition to Node.js. For example functions that cover a variety of scenarios, see Lambda@Edge Example Functions.	August 1, 2019
Updated monitoring graphs	Content updates to describe new ways for you to monitor Lambda functions associated with your CloudFront distribut ions directly from the CloudFront console to more easily track and debug errors. For more information, see Monitoring CloudFront.	June 20, 2019
Consolida ted security content	A new Security chapter consolidates information about CloudFront's features around and implementation of data protection, IAM, logging, compliance, and more. For more information, see Security.	May 24, 2019
Domain validation is now required	CloudFront now requires that you use an SSL certificate to verify that you have permission to use an alternate domain name with a distribution. For more information, see Using Alternate Domain Names and HTTPS.	April 9, 2019
Updated PDF filename	The new filename for the Amazon CloudFront Developer Guide is: AmazonCloudFront_DevGuide. The previous name was: cf-dg.	January 7, 2019

Change	Description	Date
New features	CloudFront now supports WebSocket, a TCP-based protocol that is useful when you need long-lived connections between clients and servers. You can also now set up CloudFront with origin failover for scenarios that require high availability. For more information, see Using WebSocket with CloudFront Distributions and Optimizing High Availability with CloudFront Origin Failover.	November 20, 2018
New feature	CloudFront now supports detailed error logging for HTTP requests that run Lambda functions. You can store the logs in CloudWatch and use them to help troubleshoot HTTP 5xx errors when your function returns an invalid response. For more information, see CloudWatch Metrics and CloudWatch Logs for Lambda Functions.	October 8, 2018
New feature	You can now opt to have Lambda@Edge expose the body in a request for writable HTTP methods (POST, PUT, DELETE, and so on), so that you can access it in your Lambda function. You can choose read-only access, or you can specify that you'll replace the body. For more information, see Accessing the Request Body by Choosing the Include Body Option.	August 14, 2018
New feature	CloudFront now supports serving content compressed by using brotli or other compression algorithms, in addition to or instead of gzip. For more information, see Serving Compressed Files.	July 25, 2018
Reorganization	The Amazon CloudFront Developer Guide has been reorganized to simplify finding related content, and to improve scanability and navigation.	June 28, 2018

Change	Description	Date
New Feature	Lambda@Edge now enables you to further customize the delivery of content stored in an Amazon S3 bucket, by allowing you to access additional headers, including custom headers, within origin-facing events. For more information, see these examples showing personalization of content based on viewer location and viewer device type.	March 20, 2018
New Feature	You can now use Amazon CloudFront to negotiate HTTPS connections to origins using Elliptic Curve Digital Signature Algorithm (ECDSA). ECDSA uses smaller keys that are faster, yet, just as secure, as the older RSA algorithm. For more information, see Supported SSL/TLS Protocols and Ciphers for Communication Between CloudFront and Your Origin and About RSA and ECDSA Ciphers.	March 15, 2018
New Feature	Lambda@Edge enables you to customize error responses from your origin, by allowing you to execute Lambda functions in response to HTTP errors that Amazon CloudFront receives from your origin. For more information, see these examples showing redirects to another location and response generation with 200 status code (OK).	December 21, 2017
New Feature	A new CloudFront capability, field-level encryption, helps you to further enhance the security of sensitive data, like credit card numbers or personally identifiable information (PII) like social security numbers. For more information, see Using field-level encryption to help protect sensitive data.	December 14, 2017
Doc history archived	Older doc history was archived.	December, 2017

Amazon Glossary

For the latest Amazon terminology, see the Amazon glossary in the *Amazon Web Services Glossary Reference*.