亚马逊云科技

Developer Guide

Amazon Cloud Map

Amazon Cloud Map: Developer Guide

Table of Contents

What Is Amazon Cloud Map?

Amazon Cloud Map is a fully managed service that you can use to create and maintain a map of the backend services and resources that your applications depend on. Here's how Amazon Cloud Map works:

1. You create a namespace that identifies the name that you want to use to locate your resources and also specifies how you want to locate resources: using Amazon Cloud Map DiscoverInstances API calls, DNS queries in a VPC, or public DNS queries. In most cases, a namespace contains all the services for an application, such as a billing application.
2. You create an Amazon Cloud Map service for each type of resource for which you want to use Amazon Cloud Map to locate endpoints. For example, you might create services for web servers and database servers.

   A service is a template that Amazon Cloud Map uses when your application adds another resource, such as another web server. If you chose to locate resources using DNS when you created the namespace, a service contains information about the types of records that you want to use to locate the web server. A service also indicates whether you want to check the health of the resource and, if so, whether you want to use Amazon Route 53 health checks or a third-party health checker.
3. When your application adds a resource, it can call the Amazon Cloud Map RegisterInstance API action, which creates a service instance. The service instance contains information about how your application can locate the resource, whether using DNS or using the Amazon Cloud Map DiscoverInstances API action.
4. When your application needs to connect to a resource, it calls DiscoverInstances and specifies the namespace and service that are associated with the resource. Amazon Cloud Map returns information about how to locate one or more resources. If you specified health checking when you created the service, Amazon Cloud Map returns only healthy instances.

Amazon Cloud Map is tightly integrated with Amazon Elastic Container Service (Amazon ECS). As new container tasks spin up or down, they automatically register with Amazon Cloud Map. You can use the Kubernetes ExternalDNS connector to integrate Amazon Elastic Kubernetes Service with Amazon Cloud Map. You can also use Amazon Cloud Map to register and locate any cloud resources, such as Amazon EC2 instances, Amazon DynamoDB tables, Amazon S3 buckets, Amazon Simple Queue Service (Amazon SQS) queues, or APIs deployed on top of Amazon API Gateway, among others. You can specify attribute values for services instances, and clients can use these attributes to filter the resources that Amazon Cloud Map returns. For example, an application can request resources in a particular deployment stage, like BETA or PROD.

Topics

Accessing Amazon Cloud Map

You can access Amazon Cloud Map 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 Amazon Cloud Map. SDKs simplify authentication, integrate easily with your development environment, and provide access to Amazon Cloud Map commands. For more information, see Tools for Amazon Web Services.
- **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*.
- **Amazon Cloud Map API –** If you're using a programming language that an SDK isn't available for, see the Amazon Cloud Map API Reference for information about API actions and about how to make API requests.

Note

IPv6 Client Support – As of June 22nd, 2023 in all new regions, any commands sent to Amazon Cloud Map from `IPv6` clients are routed to a new **dualstack endpoint** (`servicediscovery.<region>.api.aws`). Amazon Cloud Map `IPv6`-only networks are reachable for both **legacy** (`servicediscovery.<region>.amazonaws.com`) and **dualstack endpoint**s in the following regions that were released prior to June 22nd, 2023:

- cn-northwest-1
- cn-north-1

Note

IPv6 Client Support – As of June 22nd, 2023 in all new regions, any commands sent to Amazon Cloud Map from `IPv6` clients are routed to a new **dualstack endpoint** (`servicediscovery.<region>.api.aws`). Amazon Cloud Map `IPv6`-only networks are reachable for both **legacy** (`servicediscovery.<region>.amazonaws.com`) and **dualstack endpoints** in the following regions that were released prior to June 22nd, 2023:

- US East (Ohio) – us-east-2
- US East (N. Virginia) – us-east-1
- US West (N. California) – us-west-1
- US West (Oregon) – us-west-2
- Africa (Cape Town) – af-south-1
- Asia Pacific (Hong Kong) – ap-east-1
- Asia Pacific (Hyderabad) – ap-south-2
- Asia Pacific (Jakarta) – ap-southeast-3
- Asia Pacific (Melbourne) – ap-southeast-4
- Asia Pacific (Mumbai) – ap-south-1
- Asia Pacific (Osaka) – ap-northeast-3
- Asia Pacific (Seoul) – ap-northeast-2
- Asia Pacific (Singapore) – ap-southeast-1
- Asia Pacific (Sydney) – ap-southeast-2
- Asia Pacific (Tokyo) – ap-northeast-1
- Canada (Central) – ca-central-1
- Europe (Frankfurt) – eu-central-1
- Europe (Ireland) – eu-west-1
- Europe (London) – eu-west-2
- Europe (Milan) – eu-south-1

- Europe (Paris) – eu-west-3
- Europe (Spain) – eu-south-2
- Europe (Stockholm) – eu-north-1
- Europe (Zurich) – eu-central-2
- Middle East (Bahrain) – me-south-1
- Middle East (UAE) – me-central-1
- South America (São Paulo) – sa-east-1
- Amazon GovCloud (US-East) – us-gov-east-1
- Amazon GovCloud (US-West) – us-gov-west-1

Amazon Identity and Access Management

Amazon Cloud Map integrates with Amazon Identity and Access Management (IAM), a service that your organization can use to do the following actions:

- Create users and groups under your organization's Amazon account
- Share your Amazon account resources among the users in the account in an efficient manner
- Assign unique security credentials to each user
- Granularly control user access to services and resources

For example, you can use IAM with Amazon Cloud Map to control which users in your Amazon account can create a new namespace or register instances.

For general information about IAM, see the following resources:

- Amazon Identity and Access Management in Amazon Cloud Map
- Amazon Identity and Access Management
- IAM User Guide

Amazon Cloud Map Pricing

Amazon Cloud Map pricing is based on resources that you register in the service registry and API calls that you make to discover them. With Amazon Cloud Map there are no upfront payments, and you only pay for what you use.

Optionally, you can enable DNS-based discovery for the resources with IP addresses. You can also enable health checking for your resources using Amazon Route 53 health checks, whether you're discovering instances using API calls or DNS queries. You will incur additional charges related to Route 53 DNS and health check usage.

For more information, see Amazon Cloud Map Pricing.

Amazon Cloud Map and Amazon Cloud Compliance

For information about Amazon Cloud Map compliance with various security compliance regulations and audits standards, see the following pages:

- Amazon Cloud Compliance
- Amazon Services in Scope by Compliance Program

Setting Up Amazon Cloud Map

The overview and procedures in this section are meant to help you get started with Amazon.

Topics

Sign Up for Amazon

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

Access the API, Amazon CLI, Amazon Tools for Windows PowerShell, or the Amazon SDKs

To use the API, the Amazon CLI, Amazon Tools for Windows PowerShell, or the Amazon SDKs, you must create *access keys*. These keys consist of an access key ID and secret access key, which are used to sign programmatic requests that you make to Amazon.

Users need programmatic access if they want to interact with Amazon outside of the Amazon Web Services Management Console. The Amazon APIs and the Amazon Command Line Interface require access keys. Whenever possible, create temporary credentials that consist of an access key ID, a secret access key, and a security token that indicates when the credentials expire.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	Use short-term credentials to sign programmatic requests to the Amazon CLI or Amazon APIs (directly or by using the Amazon SDKs).	Following the instructions in Using temporary credentials with Amazon resources in the *IAM User Guide*.
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the Amazon CLI or Amazon APIs (directly or by using the Amazon SDKs).	Following the instructions in Managing access keys for IAM users in the *IAM User Guide*.

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 Cloud Map API. Using an SDK has several benefits. SDKs make authentication simpler, integrate easily with your development environment, and provide access to Amazon Cloud Map commands. For more information, see Tools for Amazon Web Services.

Using Amazon Cloud Map

Amazon Cloud Map is a managed solution that you can use to map logical names to the resources for an application. It also helps your applications discover resources using one of the Amazon SDKs, RESTful API calls, or DNS queries. Amazon Cloud Map serves only healthy resources, which can be Amazon DynamoDB (DynamoDB) tables, Amazon Simple Queue Service (Amazon SQS) queues, or any higher-level application services that are built using Amazon Elastic Compute Cloud (Amazon EC2) instances or Amazon Elastic Container Service (Amazon ECS) tasks.

Topics

Overview of How to Use Amazon Cloud Map

The following is an overview of how you can use Amazon Cloud Map:

1. Create a namespace, which is a logical grouping of services. When you create a namespace, you specify the name that you want your applications to use to discover instances. You also specify how you want to discover service instances that you register with Amazon Cloud Map: using API calls or using DNS queries.

   For more information, see the following topics:

   - Creating an Amazon Cloud Map namespace
   - CreatePublicDnsNamespace, CreatePrivateDnsNamespace, and CreateHttpNamespace in the *Amazon Cloud Map API Reference*

   If you create a public or private DNS namespace, Amazon Cloud Map automatically creates an Amazon Route 53 public or private hosted zone that has the same name as the namespace. Even with public and private DNS namespaces, you can still discover instances using Amazon Cloud Map DiscoverInstances requests.

   For a list of the endpoints that you can submit Amazon Cloud Map API requests to, see Amazon Cloud Map in the "Amazon Regions and Endpoints" chapter in the *Amazon Web Services General Reference*.

2. If you created a public DNS namespace, perform the following steps to change the name servers for the domain registration to the name servers for the Route 53 hosted zone that Amazon Cloud Map created when you created the namespace:

    a. If you already registered a domain that has the same name as the public DNS namespace, skip to step 2b.

       If you haven't registered a domain that has the same name as the namespace, register a domain. If you want to use Route 53 for domain registration, see Registering a New Domain in the *Amazon Route 53 Developer Guide*. Then skip to step 3.

    b. Use the `OperationId` that was returned when you created the namespace to get the namespace ID. For more information, see GetOperation.

       Note

       If you're using a programmatic method to perform these steps, you'll also use the namespace ID later in the process to create a service.

    c. Use the namespace ID that you got in step 2b to get the ID of the Route 53 hosted zone that Amazon Cloud Map created. For more information, see GetNamespace in the *Amazon Cloud Map API Reference*.

    d. Using the hosted zone ID that you got in step 2c, get the names of the name servers that Route 53 assigned to your hosted zone. For more information, see Getting the Name Servers for a Public Hosted Zone.

    e. Change the name servers that are assigned to the domain. If the domain is registered with Route 53, see Adding or Changing Name Servers and Glue Records for a Domain for more information.

3. Create a service, which contains the service instances that identify how to contact the resources for an application, such as a web server, a DynamoDB table, or an Amazon S3 bucket.

   If you created a public or private DNS namespace in step 1, the name that you specify for the service becomes part of the names of records in the Route 53 public or private hosted zone that Amazon Cloud Map created automatically in step 1. When you register an instance in the next step, Amazon Cloud Map creates records in the hosted zone. The record names are a combination of the name of the service (such as `backend`) and the name of the namespace (such as `example.com`): `backend.example.com`.

When you create a service, you can also choose whether you want to check the health of the resources that service instances point to:

- If you choose no health checking, Amazon Cloud Map or Route 53 return service instances regardless of the health of the corresponding resources.
- If you choose Route 53 health checking (only available for public DNS namespaces), Amazon Cloud Map automatically creates a Route 53 health check and associates it with the corresponding Route 53 record. Route 53 responds to DNS queries only with records for healthy resources.
- If you choose custom health checking, you use a third-party application to determine the health of your resources. Based on the results of the third-party health checks, you send UpdateInstanceCustomHealthStatus requests to Amazon Cloud Map to update the status of the service instances.

If you configure health checking, either Amazon Cloud Map or Route 53 returns only service instances for healthy resources in response to DiscoverInstances requests or DNS queries.

For more information, see the following topics:

- Creating an Amazon Cloud Map service
- CreateService in the *Amazon Cloud Map API Reference*

4. Register one or more service instances. Each service instance contains information about how your application can contact one resource for an application.

   For more information, see the following topics:

   - Registering an Amazon Cloud Map service instance
   - RegisterInstance in the *Amazon Cloud Map API Reference*

5. Write your application to discover instances using either the Amazon Cloud Map DiscoverInstances API action or using DNS queries:

   - If your application uses DiscoverInstances, Amazon Cloud Map returns information about the available instances that meet the specified criteria.
   - If your application uses DNS queries, Route 53 returns one or more records.

If you specified settings for a health check when you created the service, Amazon Cloud Map or Route 53 returns values only for healthy instances.

6. When you want to stop using a resource, deregister the corresponding service instance. Amazon Cloud Map automatically deletes the associated Route 53 record and health check, if any.

   For more information, see the following topics:

   - Deregistering an Amazon Cloud Map service instance
   - DeregisterInstance in the *Amazon Cloud Map API Reference*
7. If you don't need a service and namespace any longer, you can delete them. Note the following:

   - Before you can delete a service, you must deregister all instances that were registered using the service.
   - Before you can delete a namespace, you must delete all services that were created in the namespace.

   For more information, see the following topics:

   - Deleting an Amazon Cloud Map service
   - Deleting an Amazon Cloud Map namespace
   - DeleteService in the *Amazon Cloud Map API Reference*
   - DeleteNamespace in the *Amazon Cloud Map API Reference*

Configuring Amazon Cloud Map

The following sections explain how to use the Amazon Cloud Map console and Amazon CLI to create, view, and delete namespaces and services, and register and deregister instances.

In a production environment, you'll probably perform most Amazon Cloud Map actions programmatically. For more information about programmatic access to Amazon Cloud Map, see the following pages for documentation and downloads:

- Setting Up Amazon Cloud Map

- Tools for Amazon Web Services lists SDKs, command line tools, and other developer resources.
- Amazon Cloud Map API Reference provides information about using the Amazon Cloud Map API when you're using a programming language that Amazon doesn't provide an SDK for.

Topics

Working with Amazon Cloud Map namespaces

A namespace is a way to group services for an application. When you create a namespace, you specify how you want to discover service instances that you register with Amazon Cloud Map: using API calls or using DNS queries. You also specify the name that you want your application to use to discover instances.

Topics

Creating an Amazon Cloud Map namespace

To create a namespace, perform the following procedure.

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. Choose **Create namespace**.
3. On the **Create namespace** page, enter the applicable values. For more information, see Values that you specify when you create a namespace.
4. Choose **Create namespace**.

Amazon CLI

- Create a namespace with the command for the instance discovery type you would prefer (replace the *red* values with your own).

  - Create an HTTP namespace using create-http-namespace. Service instances registered using an HTTP namespace can be discovered using a `DiscoverInstances` request, but they can't be discovered using DNS.

    ```
    aws servicediscovery create-http-namespace --name name-of-namespace
    ```

  - Create a private namespace based on DNS and only visible inside a specified Amazon VPC using create-private-dns-namespace. You can discover instances that were registered with a private DNS namespace by using either a `DiscoverInstances` request or using DNS

    ```
    aws servicediscovery create-private-dns-namespace --name name-of-namespace --
    vpc vpc-xxxxxxxxx
    ```

  - Create a public namespace based on DNS that is visible on the internet using create-public-dns-namespace. You can discover instances that were registered with a public DNS namespace by using either a `DiscoverInstances` request or using DNS.

    ```
    aws servicediscovery create-public-dns-namespace --name name-of-namespace
    ```

    Note

    Namespace requirements:

    - Namespaces configured for public DNS queries must end with a top level domain (e.g .com).
    - The namespace name can have up to 1,024 characters, and must start and end with a letter.
    - Valid characters: a-z, A-Z, 0-9, . (period), _ (underscore), and - (hyphen).

Amazon SDK for Python (Boto3)

1. If you don't already have `Boto3` installed, you can find instructions for installing, configuring, and using `Boto3` here.
2. Import `Boto3` and use `servicediscovery` as your service.

   ```
   import boto3
   client = boto3.client('servicediscovery')
   ```

3. Create a namespace with the command for the instance discovery type you would prefer (replace the *red* values with your own):

   - Create an HTTP namespace using `create_http_namespace()`. Service instances registered using an HTTP namespace can be discovered using `discover_instances()`, but they can't be discovered using DNS.

     ```
     response = client.create_http_namespace(
         Name='name-of-namespace',
     )
     # If you want to see the response
     print(response)
     ```

   - Create a private namespace based on DNS and only visible inside a specified Amazon VPC using `create_private_dns_namespace()`. You can discover instances that were registered with a private DNS namespace by using either `discover_instances()` or using DNS

     ```
     response = client.create_private_dns_namespace(
         Name='name-of-namespace',
         Vpc='vpc-1c56417b',
     )
     # If you want to see the response
     print(response)
     ```

   - Create a public namespace based on DNS that is visible on the internet using `create_public_dns_namespace()`. You can discover instances that were registered with a public DNS namespace by using either `discover_instances()` or using DNS.

     ```
     response = client.create_public_dns_namespace(
         Name='name-of-namespace',
     ```

```
)
# If you want to see the response
print(response)
```

- Example response output

```
{
    'OperationId': 'gv4g5meo7ndmeh4fqskygvk23d2fijwa-k9302yzd',
    'ResponseMetadata': {
        '...': '...',
    },
}
```

> **Note**
>
> **Namespace requirements:**
>
> - Namespaces configured for public DNS queries must end with a top level domain (e.g .com).
> - The namespace name can have up to 1,024 characters, and must start and end with a letter.
> - Valid characters: a-z, A-Z, 0-9, . (period), _ (underscore), and - (hyphen).

Values that you specify when you create a namespace

When you create an Amazon Cloud Map namespace, you specify the following values.

> **Note**
>
> After you create a namespace, you can change tags. However, you can't change any other values.

Values

- Namespace name
- Namespace description
- Instance discovery

- Tags
- VPC

Namespace name

The name that you specify for a namespace depends on how you want your application to discover instances. The method of how instances are discovered is determined by the option that you choose for **Instance discovery**. The options appear later on the current page in the console. They are as follows:

API calls

If you choose this option, your application discovers service instances by specifying the namespace name and service name in a DiscoverInstances request. For more information, see DiscoverInstances in the *Amazon Cloud Map API Reference*.

You can specify a name that's up to 1,024 characters in length. A name can contain both uppercase and lowercase letters, numbers, underscores (_), and hyphens (-).

API calls and DNS queries in VPCs

Enter the domain name that you want your applications in a VPC to use when they discover instances by submitting DNS queries. Amazon Cloud Map automatically creates an Amazon Route 53 private hosted zone that has this name. When you register service instances, Amazon Cloud Map creates DNS records in the hosted zone that have names in the following format:

```
service-name.namespace-name
```

If you choose this option, your application can also discover instances by specifying the namespace name and service name in a DiscoverInstances request. For more information, see DiscoverInstances in the *Amazon Cloud Map API Reference*.

You can specify an internationalized domain name (IDN) if you convert the name to Punycode first. For information about online converters, perform an internet search on "punycode converter".

You can also convert an internationalized domain name to Punycode when you create namespaces programmatically. For example, if you're using Java, you can convert a Unicode value to Punycode by using the `toASCII` method of the java.net.IDN library.

API calls and public DNS queries

Enter the domain name that you want your applications to use when they discover instances by submitting public DNS queries. This must be a domain name that you have registered. When you create the namespace, Amazon Cloud Map automatically creates an Amazon Route 53 public hosted zone that has the same name. When you register service instances, Amazon Cloud Map creates DNS records in the hosted zone that have names in the following format:

`service-name.namespace-name`

If you choose this option, your application can also discover instances by specifying the namespace name and service name in a DiscoverInstances request. For more information, see DiscoverInstances in the *Amazon Cloud Map API Reference*.

You can specify an internationalized domain name (IDN) if you convert the name to Punycode first. For information about online converters, perform an internet search on "punycode converter".

You can also convert an internationalized domain name to Punycode when you create namespaces programmatically. For example, if you're using Java, you can convert a Unicode value to Punycode by using the `toASCII` method of the java.net.IDN library.

Namespace description

Enter a description for the namespace. The value that you enter here appears on the **Namespaces** page and on the detail page for each namespace.

Instance discovery

Choose how you want your application to discover registered instances:

API calls

Choose this option if you want your application to use only API calls to discover registered instances.

API calls and DNS queries in VPCs

Choose this option if you want your application to be able to discover instances using either API calls or using DNS queries in a VPC. You aren't required to use both methods.

API calls and public DNS queries

Choose this option if you want your application to be able to discover instances using either API calls or using public DNS queries. You aren't required to use both methods.

SOA TTL

For **API calls and DNS queries in VPCs** or **API calls and public DNS queries**, the time to live (TTL) value for the start of authority (SOA) DNS record of the Route 53 hosted zone created with your namespace. The value determines how long DNS resolvers cache information for this record before the resolvers forward another DNS query to Amazon Route 53 to get updated settings. A smaller value will also reduce the time a missing entry will be cached (negative caching) at the expense of additional queries for that namespace.

Tags

You can specify one or more tags to add to your namespace. A tag is an optional label that you can assign to an Amazon resource. Each tag consists of a key and a value. For example, you can define a tag with Key = Environment and Value = Production. Tags enable you to categorize your Amazon resources so you can more easily manage them.

You can update or remove tags on your namespaces after they have been created. For more information, see Tagging your Amazon Cloud Map resources.

VPC

When you choose **API calls and DNS queries in VPCs** for the value of **Instance discovery**, Amazon Cloud Map creates an Amazon Route 53 private hosted zone that has the same name. Amazon Cloud Map associates the VPC that you choose in the **VPC** list with that private hosted zone.

Route 53 Resolver resolves DNS queries that originate in the VPC using records in the private hosted zone. If the private hosted zone doesn't include a record that matches the domain name in a DNS query, Route 53 responds to the query with `NXDOMAIN` (non-existent domain).

You can associate additional VPCs with the private hosted zone. For more information, see AssociateVPCWithHostedZone in the *Amazon Route 53 API Reference*.

Viewing your Amazon Cloud Map namespaces

To view a list of the namespaces you've created, perform the following procedure.

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. In the navigation pane, choose **Namespaces**.

Amazon CLI

- List namespaces with the `list-namespaces` command.

```
aws servicediscovery list-namespaces
```

Amazon SDK for Python (Boto3)

1. If you don't already have `Boto3` installed, you can find instructions for installing, configuring, and using `Boto3` here.
2. Import `Boto3` and use `servicediscovery` as your service.

```
import boto3
client = boto3.client('servicediscovery')
```

3. List namespaces with `list_namespaces()`.

```
response = client.list_namespaces()
# If you want to see the response
print(response)
```

Example response output

```
{
    'Namespaces': [
        {
            'Arn': 'arn:aws-cn::servicediscovery:us-
west-2:123456789012:namespace/ns-xxxxxxxxxxxxxxx',
            'CreateDate': 1585354387.357,
            'Id': 'ns-xxxxxxxxxxxxxxx',
            'Name': 'myFirstNamespace',
            'Properties': {
                'DnsProperties': {
                    'HostedZoneId': 'Z06752353VBUDTC32S84S',
                },
                'HttpProperties': {
                    'HttpName': 'myFirstNamespace',
                },
            },
            'Type': 'DNS_PRIVATE',
        },
        {
            'Arn': 'arn:aws-cn::servicediscovery:us-
west-2:123456789012:namespace/ns-xxxxxxxxxxxxxxx',
            'CreateDate': 1586468974.698,
            'Description': 'My second namespace',
            'Id': 'ns-xxxxxxxxxxxxxxx',
            'Name': 'mySecondNamespace.com',
            'Properties': {
                'DnsProperties': {
                },
                'HttpProperties': {
                    'HttpName': 'mySecondNamespace.com',
                },
            },
            'Type': 'HTTP',
        },
        {
            'Arn': 'arn:aws-cn::servicediscovery:us-
west-2:123456789012:namespace/ns-xxxxxxxxxxxxxxx',
            'CreateDate': 1587055896.798,
            'Id': 'ns-xxxxxxxxxxxxxxx',
            'Name': 'myThirdNamespace.com',
            'Properties': {
                'DnsProperties': {
                    'HostedZoneId': 'Z09983722P0QME1B3KC8I',
                },
                'HttpProperties': {
                    'HttpName': 'myThirdNamespace.com',
                },
            },
            'Type': 'DNS_PRIVATE',
        },
    ],
    'ResponseMetadata': {
        '...': '...',
    },
}
```

Deleting an Amazon Cloud Map namespace

When you delete a namespace, you can no longer use it to register or discover service instances. Note the following:

- Before you can delete a namespace, you must delete all the services that were created in the namespace. For more information, see Deleting an Amazon Cloud Map service.
- Before you can delete a service, you must deregister all the service instances that were registered using the service. For more information, see Deregistering an Amazon Cloud Map service instance.
- When you create a namespace, if you specify that you want to discover service instances using either public DNS queries or DNS queries in VPCs, Amazon Cloud Map creates an Amazon Route 53 public or private hosted zone. When you delete the namespace, Amazon Cloud Map deletes the corresponding hosted zone.

To delete a namespace, perform the following procedure.

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. In the navigation pane, choose **Namespaces**.
3. Select the namespace that you want to delete, then choose **Delete**.
4. Confirm that you want to delete the service by choosing **Delete** again.

Amazon CLI

- Delete a namespace with the `delete-namespace` command (replace the *red* value with your own). If the namespace still contains one or more services, the request fails.

```
aws servicediscovery delete-namespace --id ns-xxxxxxxxxxx
```

Amazon SDK for Python (Boto3)

1. If you don't already have `Boto3` installed, you can find instructions for installing, configuring, and using `Boto3` here.
2. Import `Boto3` and use `servicediscovery` as your service.

   ```
   import boto3
   client = boto3.client('servicediscovery')
   ```

3. Delete a namespace with `delete_namespace()` (replace the *red* value with your own). If the namespace still contains one or more services, the request fails.

   ```
   response = client.delete_namespace(
       Id='ns-xxxxxxxxxxx',
   )
   # If you want to see the response
   print(response)
   ```

   Example response output

   ```
   {
       'OperationId': 'gv4g5meo7ndmeh4fqskygvk23d2fijwa-k98y6drk',
       'ResponseMetadata': {
           '...': '...',
       },
   }
   ```

Working with Amazon Cloud Map services

A service is a template for registering service instances, which allow you to locate the resources for an application using DNS queries or the Amazon Cloud Map DiscoverInstances API action, depending on how you configured the namespace.

Topics

Creating an Amazon Cloud Map service

To create a service, perform the following procedure.

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. In the navigation pane, choose **Namespaces**.
3. On the **Namespaces** page, choose the namespace that you want to add the service to.
4. On the **Namespace: *namespace-name*** page, choose **Create service**.
5. On the **Create service** page, enter the applicable values. For more information, see Values that you specify when you create services.
6. Choose **Create service**.

Amazon CLI

- Create a service with the `create-service` command (replace the *red* value with your own).

```
aws servicediscovery create-service \
    --name service-name \
    --namespace-id  ns-xxxxxxxxxxx \
    --dns-config "NamespaceId=ns-
xxxxxxxxxxx,RoutingPolicy=MULTIVALUE,DnsRecords=[{Type=A,TTL=60}]"
```

Output:

```
{
        "Service": {
        "Id": "srv-xxxxxxxxxxx",
        "Arn": "arn:aws-cn:servicediscovery:us-west-2:123456789012:service/srv-
xxxxxxxxxxx",
        "Name": "service-name",
        "NamespaceId": "ns-xxxxxxxxxxx",
        "DnsConfig": {
            "NamespaceId": "ns-xxxxxxxxxxx",
            "RoutingPolicy": "MULTIVALUE",
            "DnsRecords": [
                {
                    "Type": "A",
                    "TTL": 60
                }
            ]
        },
        "CreateDate": 1587081768.334,
        "CreatorRequestId": "567c1193-6b00-4308-bd57-ad38a8822d25"
    }
}
```

Amazon SDK for Python (Boto3)

1. If you don't already have `Boto3` installed, you can find instructions for installing, configuring, and using `Boto3` here.
2. Import `Boto3` and use `servicediscovery` as your service.

```
import boto3
client = boto3.client('servicediscovery')
```

3. Create a service with `create_service()` (replace the *red* value with your own).

```
response = client.create_service(
    DnsConfig={
        'DnsRecords': [
            {
                'TTL': 60,
                'Type': 'A',
            },
        ],
        'NamespaceId': 'ns-xxxxxxxxxxx',
        'RoutingPolicy': 'MULTIVALUE',
    },
    Name='service-name',
    NamespaceId='ns-xxxxxxxxxxx',
)
```

Example response output

```
{
    'Service': {
        'Arn': 'arn:aws-cn:servicediscovery:us-west-2:123456789012:service/srv-
xxxxxxxxxxx',
        'CreateDate': 1587081768.334,
        'DnsConfig': {
            'DnsRecords': [
                {
                    'TTL': 60,
                    'Type': 'A',
                },
            ],
            'NamespaceId': 'ns-xxxxxxxxxxx',
            'RoutingPolicy': 'MULTIVALUE',
        },
        'Id': 'srv-xxxxxxxxxxx',
        'Name': 'service-name',
        'NamespaceId': 'ns-xxxxxxxxxxx',
    },
    'ResponseMetadata': {
        '...': '...',
    },
}
```

Note

For services that are accessible by DNS queries, you cannot create multiple services with names that differ only by case (such as EXAMPLE and example). Otherwise, these services will have the same DNS name. If you use a namespace that's only accessible by API calls, then you can create services that with names that differ only by case.

Values that you specify when you create services

When you create an Amazon Cloud Map service, you specify the following values.

Note

You can only change tags in a service after you create it.

Values

- Service name
- Service description
- Service discovery configuration
- Routing policy
- Record type
- TTL
- Health check options
- Failure threshold
- Health check protocol
- Health check path
- Tags

Service name

Enter a name that describes the instances that you register when using this service. The value is used to discover Amazon Cloud Map service instances either in API calls or in DNS queries. This depends on the instance discovery method that you chose when you created the namespace. You can use one of the following methods:

- **API calls** – When your application calls DiscoverInstances, the API call includes the namespace and service names.
- **API calls and DNS queries in VPCs** or **API calls and public DNS queries** – When you register service instances and create the namespace, Amazon Cloud Map creates an Amazon Route 53 private or public hosted zone. It also create DNS records in that hosted zone. The names of the records are in the following format:

  ```
  service-name.namespace-name
  ```

  When your application submits a DNS query to discover service instances, the query is for a record that includes the name of the service in the record name.

> **Note**
>
> When creating a service in a namespace that supports DNS queries, you can choose to have the service instances for that service discoverable only with calls

to the DiscoverInstances API operation and not DNS queries. See Service discovery configuration.

If you want Amazon Cloud Map to create an **SRV** record when you register an instance and you're using a system that requires a specific **SRV** format (such as HAProxy), specify the following for **Service name**:

- Start the name with an underscore (_), for example **_exampleservice**.
- End the name with `._protocol`, for example **._tcp**.

When you register an instance, Amazon Cloud Map creates an **SRV** record and assigns a name by concatenating the service name and the namespace name, for example:

_exampleservice._tcp.example.com

Note

For services that are discoverable by DNS queries, you can't create multiple services with names that differ only by case (such as EXAMPLE and example). Otherwise, these services have the same DNS name and can't be distinguished.

Service description

Enter a description for the service. The value that you enter here appears on the **Services** page and on the detail page for each service.

Service discovery configuration

If the namespace supports DNS queries, Amazon Cloud Map supports the following service discovery options:

API and DNS

Amazon Cloud Map will create **SRV** records when you register an instance for the service. Service instances can also be discovered using the DiscoverInstances API operation.

API only

Amazon Cloud Map will not create **SRV** records for instance for the service. Service instances can be discovered only using the DiscoverInstances API operation.

Routing policy (public and private DNS namespaces only)

If you're using a public or private DNS namespace to create the service, choose the Amazon Route 53 routing policy for the DNS records that Amazon Cloud Map creates when you register instances. (Public DNS namespaces have a value of **API calls and public DNS queries** for **Instance discovery**, and private DNS namespaces have a value of **API calls and DNS queries in VPCs**.)

> **Note**
>
> You can't use the console to configure Amazon Cloud Map to create a Route 53 alias record when you register an instance. If you want Amazon Cloud Map to create alias records for Elastic Load Balancing load balancer when you register instances programmatically, choose **Weighted routing** for **Routing policy**.

Amazon Cloud Map supports the following Route 53 routing policies:

Weighted routing

Route 53 returns the applicable value from one randomly selected instance from among the instances that you registered using the same service. All records have the same weight, so you can't route more or less traffic to any instances.

For example, suppose the service includes configurations for one **A** record and a health check, and you use the service to register 10 instances. Route 53 responds to DNS queries with the IP address for one randomly selected instance from among the healthy instances. If no instances are healthy, Route 53 responds to DNS queries as if all the instances were healthy.

If you don't define a health check for the service, Route 53 assumes that all instances are healthy and returns the applicable value for one randomly selected instance.

For more information, see Weighted Routing in the *Amazon Route 53 Developer Guide*.

Multivalue answer routing

If you define a health check for the service and the result of the health check is healthy, Route 53 returns the applicable value for up to eight instances.

For example, suppose that the service includes configurations for one **A** record and a health check. You use the service to register 10 instances. Route 53 responds to DNS queries with

IP addresses for only a maximum of eight healthy instances. If fewer than eight instances are healthy, Route 53 responds to every DNS query with the IP addresses for all the healthy instances.

If you don't define a health check for the service, Route 53 assumes that all instances are healthy and returns the values for up to eight instances.

For more information, see Multivalue Answer Routing in the *Amazon Route 53 Developer Guide*.

Record type (public and private DNS namespaces only)

If you're using a public or private DNS namespace to create the service, choose the DNS record type for the records that Amazon Cloud Map creates when you register instances. Amazon Route 53 returns the applicable value in response to DNS queries for registered instances.

The following record types are supported:

A

When you register an instance, you specify the IP address of the resource in IPv4 format, such as **192.0.2.44**.

AAAA

When you register an instance, you specify the IP address of the resource in IPv6 format, such as **2001:0db8:85a3:0000:0000:abcd:0001:2345**.

CNAME

When you register an instance, you specify the domain name of the resource (such as www.example.com). Note the following:

- If you want to choose **CNAME**, you must choose **Weighted routing** for **Routing policy**.
- If you choose **CNAME**, you can't choose **Route 53 health check** for **Health check options**.

SRV

The value for an **SRV** record uses the following values:

```
priority weight port service-hostname
```

Note the following about the values:

- The values of `priority` and `weight` are both set to 1 and can't be changed.

- For `port`, Amazon Cloud Map uses the value that you specify for **Port** when you register an instance.
- The value of `service-hostname` is a concatenation of the following values:
  - The value that you specify for **Service instance ID** when you register an instance
  - The name of the service
  - The name of the namespace

  For example, suppose you specify **test** for **Service instance ID** when you register an instance. The name of the service is **backend** and the name of the namespace is **example.com**. Amazon Cloud Map assigns the following value to the `service-hostname` attribute in the **SRV** record:

  ```
  test.backend.example.com
  ```

If you specify settings for an **SRV** record, note the following:

- If you specify values for **IPv4 address**, **IPv6 address**, or both, Amazon Cloud Map automatically creates **A** and/or **AAAA** records that have the same name as the value of `service-hostname` in the **SRV** record.
- If you're using a system that requires a specific **SRV** format, such as HAProxy, see service name for information about how to specify the correct name format.

You can specify record types in the following combinations:

- **A**
- **AAAA**
- **A** and **AAAA**
- **CNAME**
- **SRV**

If you specify **A** and **AAAA** record types, you can specify an IPv4 IP address, an IPv6 IP address, or both when you register an instance.

TTL (public and private DNS namespaces only)

If you're using a public or private DNS namespace to create the service, enter a value for **TTL**, or time to live. The value of **TTL** determines how long DNS resolvers cache information for this record before the resolvers forward another DNS query to Amazon Route 53 to get updated settings.

Health check options

No health check

If you don't configure a health check, traffic is routed to service instances regardless of whether they're healthy.

Route 53 health check (not supported for private DNS namespaces)

If you specify settings for an Amazon Route 53 health check, Amazon Cloud Map creates a Route 53 health check whenever you register an instance and deletes the health check when you deregister the instance.

For public DNS namespaces, Amazon Cloud Map associates the health check with the Route 53 record that Amazon Cloud Map creates when you register an instance.

For namespaces that you use API calls to discover instances for, Amazon Cloud Map creates a Route 53 health check. However, there's no DNS record for Amazon Cloud Map to associate the health check with. To determine whether a health check is healthy, you can configure monitoring using either the Route 53 console or using Amazon CloudWatch. For more information about using the Route 53 console, see Get Notified When a Health Check Fails in the *Amazon Route 53 Developer Guide*. For more information about using CloudWatch, see PutMetricAlarm in the *Amazon CloudWatch API Reference*.

For information about the charges for Route 53 health checks, see Route 53 Pricing.

Custom health check

If you configure Amazon Cloud Map to use a custom health check when you register an instance, you must use a third-party health checker to evaluate the health of your resources. Custom health checks are useful in the following circumstances:

- You can't use a Route 53 health check because the resource isn't available over the internet. For example, suppose that you have an instance that's located in an Amazon VPC. You can use a custom health check for this instance. However, for the health check to work,your health checker must also be in the same VPC as your instance.
- You want to use a third-party health checker regardless of where your resources are.

Failure threshold (Route 53 health check only)

The number of consecutive Route 53 health checks that a resource must pass or fail for Amazon Route 53 to change the current status of the resource from healthy to unhealthy or the opposite situation. For more information, see How Amazon Route 53 Determines Whether a Health Check Is Healthy *Amazon Route 53 Developer Guide*.

Health check protocol (Route 53 health check only)

The method that you want Amazon Route 53 to use to check the health of your resource:

HTTP

Route 53 tries to establish a TCP connection. If successful, Route 53 submits an HTTP request and waits for an HTTP status code of a 2xx or 3xx format.

HTTPS

Route 53 tries to establish a TCP connection. If successful, Route 53 submits an HTTPS request and waits for an HTTP status code of a 2xx or 3xx format.

Important

If you choose HTTPS, the resource must support TLS v1.0 or later.

If you choose HTTPS for the value of **Health check protocol**, additional charges apply. For more information, see Route 53 Pricing.

TCP

Route 53 tries to establish a TCP connection.

For more information, see How Amazon Route 53 Determines Whether a Health Check Is Healthy.

Health check path (Route 53 HTTP and HTTPS health checks only)

The path that you want Amazon Route 53 to request when performing health checks. The path can be any value such as the file `/docs/route53-health-check.html`. When the resource is healthy, the returned value is an HTTP status code of a 2xx or 3xx format. You can also include query string parameters, for example, `/welcome.html?language=jp&login=y`. The Amazon Cloud Map console automatically adds a leading slash (/) character.

Tags

You can specify one or more tags to add to your service. A tag is an optional label that you can assign to an Amazon resource. Each tag consists of a key and a value. For example, you can define a tag with Key = Environment and Value = Production. Using tags to categorize Amazon resources can make managing those resources easier.

After your tags are created, you can always update or remove tags on your namespaces. For more information, see Tagging your Amazon Cloud Map resources.

Updating an Amazon Cloud Map service

To update a service, perform the following procedure.

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. In the navigation pane, choose **Namespaces**.
3. On the **Namespaces** page, choose the namespace that you want to edit the service for.
4. On the **Namespace:** ***namespace-name*** page, select the service you want to edit and click **Edit**.
5. On the **Service:** ***service-name*** page, click **Edit**.
6. On the **Edit service** page, enter the applicable values.
7. Click **Update service**.

Amazon CLI

- Update a service with the `update-service` command (replace the *red* value with your own).

```
aws servicediscovery update-service \
    --id  srv-xxxxxxxxxxx \
    --service "Description=new
 description,DnsConfig={DnsRecords=[{Type=A,TTL=60}]}"
```

Output:

```
{
    "OperationId": "l3pfx7f4ynndrbj3cfq5fm2qy2z37bms-5m6iaoty"
}
```

Amazon SDK for Python (Boto3)

1. If you don't already have `Boto3` installed, you can find instructions for installing, configuring, and using `Boto3` here.
2. Import `Boto3` and use `servicediscovery` as your service.

```
import boto3
client = boto3.client('servicediscovery')
```

3. Update a service with `update_service()` (replace the *red* value with your own).

```
response = client.update_service(
    Id='srv-xxxxxxxxxxx',
    Service={
        'DnsConfig': {
            'DnsRecords': [
                {
                    'TTL': 300,
                    'Type': 'A',
                },
            ],
        },
        'Description': "new description",
    }
)
```

Example response output

```
{
    "OperationId": "l3pfx7f4ynndrbj3cfq5fm2qy2z37bms-5m6iaoty"
}
```

Viewing the services in a namespace

To view a list of the services that you created in a namespace, perform the following procedure.

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.

2. In the navigation pane, choose **Namespaces**.
3. Choose the name of the namespace that contains the services that you want to list.

Amazon CLI

- List services with the `list-services` command.

```
aws servicediscovery list-services
```

Amazon SDK for Python (Boto3)

1. If you don't already have `Boto3` installed, you can find instructions for installing, configuring, and using `Boto3` here.
2. Import `Boto3` and use `servicediscovery` as your service.

```
import boto3
client = boto3.client('servicediscovery')
```

3. List services with `list_services()`.

```
response = client.list_services()
# If you want to see the response
print(response)
```

Example response output

```
{
    'Services': [
        {
            'Arn': 'arn:aws-cn:servicediscovery:us-west-2:123456789012:service/
srv-xxxxxxxxxxxxxxxx',
            'CreateDate': 1587081768.334,
            'DnsConfig': {
                'DnsRecords': [
                    {
                        'TTL': 60,
                        'Type': 'A',
                    },
                ],
                'RoutingPolicy': 'MULTIVALUE',
            },
            'Id': 'srv-xxxxxxxxxxxxxxxx',
            'Name': 'myservice',
        },
    ],
    'ResponseMetadata': {
        '...': '...',
    },
}
```

Deleting an Amazon Cloud Map service

Before you can delete a service, you must deregister all service instances that were registered using the service. For more information, see Deregistering an Amazon Cloud Map service instance.

To delete a service, perform the following procedure.

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. In the navigation pane, choose **Namespaces**.
3. Choose the option for the namespace that contains the service that you want to delete.
4. On the **Namespace: *namespace-name*** page, choose the option for the service that you want to delete.
5. Choose **Delete**.
6. Confirm that you want to delete the service.

Amazon CLI

- Delete a service with the `delete-service` command (replace the *red* value with your own).

```
aws servicediscovery delete-service --id srv-xxxxxx
```

Amazon SDK for Python (Boto3)

1. If you don't already have `Boto3` installed, you can find instructions for installing, configuring, and using `Boto3` here.
2. Import `Boto3` and use `servicediscovery` as your service.

   ```
   import boto3
   client = boto3.client('servicediscovery')
   ```

3. Delete a service with `delete_service()` (replace the *red* value with your own).

   ```
   response = client.delete_service(
       Id='srv-xxxxxx',
   )
   # If you want to see the response
   print(response)
   ```

   Example response output

   ```
   {
       'ResponseMetadata': {
           '...': '...',
       },
   }
   ```

Working with Amazon Cloud Map service instances

A service instance contains information about how to locate a resource, such as a web server, for an application. After you register instances, you locate them by using DNS queries or the Amazon Cloud Map DiscoverInstances API action.

Topics

Registering an Amazon Cloud Map service instance

To register a service instance, perform the following procedure.

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. In the navigation pane, choose **Namespaces**.
3. On the **Namespaces** page, choose the namespace that contains the service that you want to use as a template for registering a service instance.
4. On the **Namespace: *namespace-name*** page, choose the service that you want to use.
5. On the **Service: *service-name*** page, choose **Register service instance**.
6. On the **Register service instance** page, enter the applicable values. For more information, see Values that you specify when you register or update a service instance.
7. Choose **Register service instance**.

Amazon CLI

- When you submit a `RegisterInstance` request:
  - For each DNS record that you define in the service that's specified by `ServiceId`, a record is created or updated in the hosted zone that's associated with the corresponding namespace.
  - If the service includes `HealthCheckConfig`, a health check is created based on the settings in the health check configuration.
  - Any health checks are associated with each of the new or updated records.

  Register a service instance with the `register-instance` command (replace the *red* values with your own).

```
aws servicediscovery register-instance \
    --service-id srv-xxxxxxxxx \
    --instance-id myservice-xx \
    --attributes=AWS_INSTANCE_IPV4=172.2.1.3,AWS_INSTANCE_PORT=808
```

Amazon SDK for Python (Boto3)

1. If you don't already have `Boto3` installed, you can find instructions for installing, configuring, and using `Boto3` here.
2. Import `Boto3` and use `servicediscovery` as your service.

```
import boto3
client = boto3.client('servicediscovery')
```

3. When you submit a `RegisterInstance` request:

   - For each DNS record that you define in the service that's specified by `ServiceId`, a record is created or updated in the hosted zone that's associated with the corresponding namespace.
   - If the service includes `HealthCheckConfig`, a health check is created based on the settings in the health check configuration.
   - Any health checks are associated with each of the new or updated records.

   Register a service instance with `register_instance()` (replace the *red* values with your own).

```
response = client.register_instance(
    Attributes={
        'AWS_INSTANCE_IPV4': '172.2.1.3',
        'AWS_INSTANCE_PORT': '808',
    },
    InstanceId='myservice-xx',
    ServiceId='srv-xxxxxxxxx',
)
# If you want to see the response
print(response)
```

   Example response output

```
{
    'OperationId': '4yejorelbukcjzpnr6tlmrghsjwpngf4-k95yg2u7',
    'ResponseMetadata': {
        '...': '...',
    },
}
```

Values that you specify when you register or update a service instance

When you register a service instance, you specify the following values.

Values

- Instance type
- Service instance ID
- IPv4 address
- IPv6 address
- Port
- EC2 instance ID
- Custom attributes

Instance type

Each of the following instance types is available for selected configurations only.

IP address

Choose this option when the resource that's associated with the service instance is accessible using an IP address.

You can choose this option for all three types of namespaces: HTTP, public DNS, and private DNS.

EC2 Instance

Choose this option when the resource that's associated with the service instance is accessible through an EC2 instance.

You can choose this option for HTTP.

Identifying information for another resource

Choose this option when the resource that's associated with the service instance is accessible using values other than an IP address or an EC2 instance. Specify the other values in **Custom attributes**.

You can choose this option for all three types of namespaces: HTTP, public DNS, and private DNS.

Service instance ID

An identifier that you want to associate with the instance. Note the following:

- To register a new instance, you must specify a value that's unique among instances that you register by using the same service.
- If the service that's specified by **Service instance ID** includes settings for an **SRV** record, the value of **Service instance ID** is automatically included as part of the value for the **SRV** record. For more information, see **Record type** in the section Values that you specify when you create services.
- You can update an existing instance programmatically. Call RegisterInstance, specify the value of **Service instance ID** and **Service ID**, and specify the new settings for the service instance. If Amazon Cloud Map created a health check when you registered the instance originally, Amazon Cloud Map deletes the old health check and creates a new one.

> **Note**
>
> The health check isn't deleted immediately, so it will still appear for a while if you submit an Amazon Route 53 `ListHealthChecks` request, for example.

IPv4 address

The IPv4 IP address, if any, where your applications can access the resource that's associated with this service instance.

IPv6 address

The IPv6 IP address, if any, where your applications can access the resource that's associated with this service instance.

Port

The port, if any, that your applications must include to access the resource that's associated with this service instance. **Port** is required when the service includes an **SRV** record or an Amazon Route 53 health check.

EC2 instance ID

The instance Id in EC2 instance Id format for the resource.

Custom attributes

Specify key-value pairs that you want to associate with the resource, if any.

You can add up to 30 custom attributes. Note the following:

- You must specify both **Key** and **Value**.
- **Key** can be up to 255 characters long and can include the characters a-z, A-Z, 0-9 and other printable ASCII characters between 33 and 126 (Decimal). Spaces, tabs, and other whitespace characters are not allowed.
- **Value** can be up to 1,024 characters long and can include the characters a-z, A-Z, 0-9, other printable ASCII characters between 33 and 126 (Decimal), space, and tab.

Updating an Amazon Cloud Map service instance

You can update service instances in two ways, depending on which values you want to update:

- **Update any values**: If you want to update any of the values that you specified for a service instance when you registered it, including custom attributes, you reregister the service instance and respecify all values. See Updating the details of a service instance.
- **Update only custom attributes**: If you want to update only the custom attributes for a service instance, you don't need to reregister the instance. You can update only those values. See Updating the custom attributes for a service instance.

Updating the details of a service instance

To update a service instance

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. In the navigation pane, choose **Namespaces**.
3. On the **Namespaces** page, choose the namespace that contains the service that you originally used to register the service instance.
4. On the **Namespace:** ***`namespace-name`*** page, choose the service that you used to register the service instance.
5. On the **Service:** ***`service-name`*** page, copy the ID of the service instance that you want to update.

6. Choose **Register service instance**.
7. On the **Register service instance** page, paste the ID that you copied in step 5 into **Service instance ID**.
8. Enter all the other values that you want to apply to the service instance. The previous values for the service instance are not retained. For more information, see Values that you specify when you register or update a service instance.
9. Choose **Register service instance**.

Updating the custom attributes for a service instance

To update only custom attributes for a service instance

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. In the navigation pane, choose **Namespaces**.
3. On the **Namespaces** page, choose the namespace that contains the service that you originally used to register the service instance.
4. On the **Namespace:** ***`namespace-name`*** page, choose the service that you used to register the service instance.
5. On the **Service:** ***`service-name`*** page, choose the name of the service instance that you want to update.
6. In the **Custom attributes** section, choose **Edit**.
7. On the **Edit service instance:** ***`instance-name`*** page, add, remove, or update custom attributes. You can update both keys and values for existing attributes.
8. Choose **Update service instance**.

Viewing your Amazon Cloud Map service instances

To view a list of the service instances that you registered using a service, perform the following procedure.

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.

2. In the navigation pane, choose **Namespaces**.
3. Choose the name of the namespace that contains the service for which you want to list service instances.
4. Choose the name of the service that you used to create the service instances.

Amazon CLI

- List service instances with the `list-instances` command (replace the *red* value with your own).

```
aws servicediscovery list-instances --service-id srv-xxxxxxxxx
```

Amazon SDK for Python (Boto3)

1. If you don't already have `Boto3` installed, you can find instructions for installing, configuring, and using `Boto3` here.
2. Import `Boto3` and use `servicediscovery` as your service.

```
import boto3
client = boto3.client('servicediscovery')
```

3. List service instances with `list_instances()` (replace the *red* value with your own).

```
response = client.list_instances(
    ServiceId='srv-xxxxxxxxx',
)
# If you want to see the response
print(response)
```

Example response output

```
{
    'Instances': [
        {
            'Attributes': {
                'AWS_INSTANCE_IPV4': '172.2.1.3',
                'AWS_INSTANCE_PORT': '808',
            },
            'Id': 'i-xxxxxxxxxxxxxxxxx',
        },
    ],
    'ResponseMetadata': {
        '...': '...',
    },
}
```

Deregistering an Amazon Cloud Map service instance

Before you can delete a service, you must deregister all service instances that were registered using the service.

To deregister a service instance, perform the following procedure.

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. In the navigation pane, choose **Namespaces**.
3. Choose the option for the namespace that contains the service instance that you want to deregister.
4. On the **Namespace:** ***namespace-name*** page, choose the option for the service you used to register the service instance.
5. On the **Service:** ***service-name*** page, choose the option for the service instance that you want to deregister.
6. Choose **Deregister**.
7. Confirm that you want to deregister the service instance.

Amazon CLI

- Deregister a service instance with the `deregister-instance` command (replace the *red* values with your own). This command deletes the Amazon Route 53 DNS records and any health checks that Amazon Cloud Map created for the specified instance.

```
aws servicediscovery deregister-instance \
    --service-id srv-xxxxxxxxx \
    --instance-id myservice-53
```

Amazon SDK for Python (Boto3)

1. If you don't already have `Boto3` installed, you can find instructions for installing, configuring, and using `Boto3` here.
2. Import `Boto3` and use `servicediscovery` as your service.

   ```
   import boto3
   client = boto3.client('servicediscovery')
   ```

3. Deregister a service instance with `deregister-instance()` (replace the *`red`* values with your own). This command deletes the Amazon Route 53 DNS records and any health checks that Amazon Cloud Map created for the specified instance.

   ```
   response = client.deregister_instance(
       InstanceId='myservice-53',
       ServiceId='srv-xxxxxxxxx',
   )
   # If you want to see the response
   print(response)
   ```

   Example response output

   ```
   {
       'OperationId': '4yejorelbukcjzpnr6tlmrghsjwpngf4-k98rnaiq',
       'ResponseMetadata': {
           '...': '...',
       },
   }
   ```

Amazon Cloud Map features that are unavailable in the Amazon Cloud Map console

The following Amazon Cloud Map features are unavailable on the Amazon Cloud Map console. To use these features, you must use a programmatic method to access Amazon Cloud Map.

Creating Route 53 alias records when you register service instances

When you register a service instance using the console, you can't create an alias record that routes traffic to an Elastic Load Balancing (ELB) load balancer. Note the following:

- When you create a service, you must specify `WEIGHTED` for `RoutingPolicy`. You can do this using the console. For more information, see Creating an Amazon Cloud Map service.

  For information about creating a service using the Amazon Cloud Map API, see CreateService in the *Amazon Cloud Map API Reference*.
- When you register an instance, you must include the `AWS_ALIAS_DNS_NAME` attribute. For more information, see RegisterInstance in the *Amazon Cloud Map API Reference*.

Specifying the initial health status for custom health checks

If you register an instance using a service that includes a custom health check, you can't specify the initial status for the custom health check. By default, the initial status of a custom health checks is **Healthy**. If you want the initial health status to be **Unhealthy**, register the instance programmatically and include the `AWS_INIT_HEALTH_STATUS` attribute. For more information, see RegisterInstance in the *Amazon Cloud Map API Reference*.

Getting the status of an incomplete operation

If you close a browser window after you create a namespace but before creating the namespace has completed, the console doesn't provide a way to see the current status. You can get the status by using ListOperations. For more information, see ListOperations in the *Amazon Cloud Map API Reference*.

Tutorials

The following tutorials show you how to perform common tasks using Amazon Cloud Map namespaces.

Topics

Tutorial: Using Amazon Cloud Map service discovery with DNS queries

This tutorial simulates a microservice architecture with two backend services. The first service will be discoverable using a DNS query. The second service will be discoverable using the Amazon Cloud Map API only.

Note

For the purposes of this tutorial, the resource details, like domain names and IP addresses, are for simulation purposes only. They can't be resolved over the internet.

Prerequisites

The following prerequisites must be met to complete this tutorial successfully.

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

Install the Amazon Command Line Interface

If you have not yet installed the Amazon Command Line Interface, follow the steps at Installing or updating the latest version of the Amazon CLI to install it.

The tutorial requires a command line terminal or shell to run commands. In Linux and macOS, use your preferred shell and package manager.

> **Note**
>
> In Windows, some Bash CLI commands that you commonly use with Lambda (such as `zip`) are not supported by the operating system's built-in terminals. To get a Windows-integrated version of Ubuntu and Bash, install the Windows Subsystem for Linux.

Have access to the dig utility

The tutorial requires a local environment with the `dig` DNS lookup utility command. For more information about the `dig` command, see dig - DNS lookup utility.

Step 1: Create an Amazon Cloud Map namespace

In this step, you create a public Amazon Cloud Map namespace. Amazon Cloud Map creates a Route 53 hosted zone on your behalf with this same name. This gives you the ability to discovery the service instances created in this namespace either using public DNS records or by using Amazon Cloud Map API calls.

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. Choose **Create namespace**.
3. For **Namespace name**, specify `cloudmap-tutorial.com`.

   > **Note**
   >
   > If you were going to use this in production, you'd want to ensure that you specified the name of a domain you owned or had access to. But for the purposes of this tuturial, it's not necessary for it to be an actual domain that's being used.

4. (Optional) For **Namespace description**, specify a description for what you intend to use the namespace for.
5. For **Instance discovery**, select **API calls and public DNS queries**.
6. Leave the rest of the default values and choose **Create namespace**.

Step 2: Create the Amazon Cloud Map services

In this step, you create two services. The first service will be discoverable using public DNS and API calls. The second service will be discoverable using API calls only.

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. In the left navigation pane, choose **Namespaces** to list the namespaces you've created.
3. From the list of namespaces, select the `cloudmap-tutorial.com` namespace and choose **View details**.
4. In the **Services** section, choose **Create service** and do the following to create the first service.

   a. For **Service name**, enter `public-service`. The service name will be applied to the DNS records that Amazon Cloud Map creates. The format that is used is *`<service-name>`*`.`*`<namespace-name>`*.

   b. For **Service Discovery Configuration**, select **API and DNS**.

   c. In the **DNS configuration** section, for **Routing policy**, select **Multivalue answer routing**.

      Note

      The console will translate this to **MULTIVALUE** after it is selected. For more information about available routing options, see Choosing a routing policy in the *Route 53 Developer Guide*.

   d. Leave the rest of the default values and choose **Create service** which will return you to the namespace details page.

5. In the **Services** section, choose **Create service** and do the following to create the second service.

   a. For **Service name**, enter `backend-service`.

   b. For **Service Discovery Configuration**, select **API only**.

   c. Leave the rest of the default values and choose **Create service**.

Step 3: Create the Amazon Cloud Map service instances

In this step, you create two service instances, one for each service in our namespace.

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.

2. From the list of namespaces, select the namespace you created in step 1 and choose **View details**.

3. On the namespace details page, from the list of services, select the `public-service` service and choose **View details**.

4. In the **Service instances** section, choose **Register service instance** and do the following to create the first service instance.

   a. For **Service instance ID**, specify `first`.

   b. For **IPv4 address**, specify `192.168.2.1`.

   c. Leave the rest of the default values and choose **Register service instance**.

5. Using the breadcrumb at the top of the page, select **cloudmap-tutorial.com** to navigate back to the namespace detail page.

6. On the namespace details page, from the list of services, select the **backend-service** service and choose **View details**.

7. In the **Service instances** section, choose **Register service instance** and do the following to create the second service instance.

   a. For **Service instance ID**, specify `second` to indicate that this is the second service instance.

   b. For **Instance type**, select **Identifying information for another resource**.

   c. For **Custom attributes**, add a key-value pair with `service-name` as the key and `backend` as the value.

   d. Choose **Register service instance**.

Step 4: Discover the Amazon Cloud Map service instances

Now that the Amazon Cloud Map namespace, services, and service instances are created, you can verify everything is working by discovering the instances. Use the `dig` command to verify the public DNS settings and the Amazon Cloud Map API to verify the backend service. For more information about the `dig` command, see dig - DNS lookup utility.

1. Sign in to the Amazon Web Services Management Console and open the Route 53 console at https://console.amazonaws.cn/route53/.

2. In the left navigation, choose **Hosted zones**.

3. Select the **cloudmap-tutorial.com** hosted zone. This displays the hosted zone details in a separate pane. Take note of the **Name servers** associated with your hosted zone as we will use those in the next step.

4. Using the dig command and one of the Route 53 name servers for your hosted zone, query the DNS records for your service instance.

   ```
   dig @hosted-zone-nameserver public-service.cloudmap-tutorial.com
   ```

The `ANSWER SECTION` in the output should display the IPv4 address you associated with your `public-service` service.

```
;; ANSWER SECTION:
public-service.cloudmap-tutorial.com. 300 IN A 192.168.2.1
```

5. Using the Amazon CLI, query the attributes for your second service instances.

```
aws servicediscovery discover-instances --namespace-name cloudmap-tutorial.com --
service-name backend-service --region region
```

The output displays the attributes you associated with the service as key-value pairs.

```
{
    "Instances": [
        {
            "InstanceId": "second",
            "NamespaceName": "cloudmap-tutorial.com",
            "ServiceName": "backend-service",
            "HealthStatus": "UNKNOWN",
            "Attributes": {
                "service-name": "backend"
            }
        }
    ],
    "InstancesRevision": 71462688285136850
}
```

Step 5: Clean up the resources

Once you have completed the tutorial, you can delete the resources. Amazon Cloud Map requires that you clean them up in reverse order, the service instances first, then the services, and finally the namespace. Amazon Cloud Map will clean up the Route 53 resources on your behalf when you go through these steps.

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.

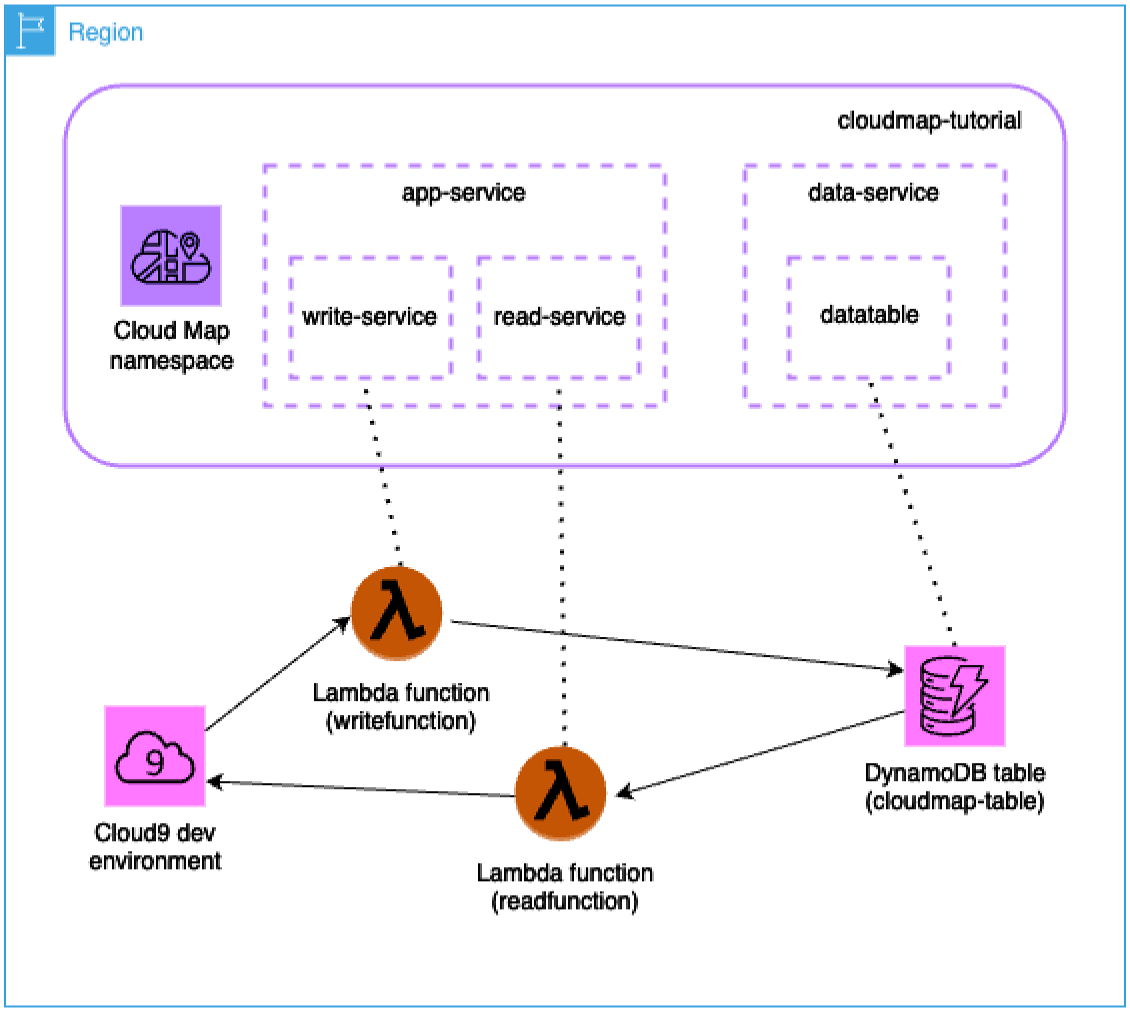
2. From the list of namespaces, select the `cloudmap-tutorial.com` namespace and choose **View details**.
3. On the namespace details page, from the list of services, select the `public-service` service and choose **View details**.
4. In the **Service instances** section, select the `first` instance and choose **Deregister**.
5. Using the breadcrumb at the top of the page, select **cloudmap-tutorial.com** to navigate back to the namespace detail page.
6. On the namespace details page, from the list of services, select the **public-service** service and choose **Delete**.
7. Repeat steps 3-6 for the `backend-service`.
8. In the left navigation, choose **Namespaces**.
9. Select the `cloudmap-tutorial.com` namespace and choose **Delete**.

> **Note**
>
> Although Amazon Cloud Map cleans up the Route 53 resources on your behalf, you can navigate to the Route 53 console to verify that the `cloudmap-tutorial.com` hosted zone is deleted.

Tutorial: Using Amazon Cloud Map service discovery with custom attributes

This tutorial demonstrates how you can use Amazon Cloud Map service discovery with custom attributes that are discoverable using the Amazon Cloud Map API. This tutorial walks you through creating a client application in an Amazon Cloud9 environment that uses two Lambda functions to write data to a DynamoDB table and then read from the table. The Lambda functions and DynamoDB table are registered in Amazon Cloud Map as service instances. The code in the client application and Lambda functions uses Amazon Cloud Map custom attributes to discover the resources needed to perform the job.

The following diagram demonstrates the high level architecture this tutorial uses.



Important

You will create Amazon resources during the workshop which will incur a cost in your Amazon account. It is recommended to clean-up the resources as soon as you finish the workshop to minimize the cost.

Prerequisites

The following prerequisites must be met to complete this tutorial successfully.

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

Step 1: Create an Amazon Cloud Map namespace

In this step, you create an Amazon Cloud Map namespace. A namespace is a construct used to group services for an application. When you create the namespace, you specify how the resources will be discoverable. For this tutorial, the resources created in this namespace will be discoverable with Amazon Cloud Map API calls using custom attributes. You will learn about this more in a later step.

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. Choose **Create namespace**.

3. For **Namespace name**, specify `cloudmap-tutorial`.
4. (Optional) For **Namespace description**, specify a description for what you intend to use the namespace for.
5. For **Instance discovery**, select **API calls**.
6. Leave the rest of the default values and choose **Create namespace**.

Step 2: Create a DynamoDB table

In this step, you create a DynamoDB table which is used to store and retrieve data for the sample application created later in this tutorial.

1. Sign in to the Amazon Web Services Management Console and open the DynamoDB console at https://console.amazonaws.cn/dynamodb/.
2. In the left navigation pane, choose **Tables**, **Create table**.
3. On the **Create table** page, do the following.

   a. For **Table name**, specify `cloudmap-table`.

   b. For **Partition key**, specify `id`.

   c. Leave the rest of the default values and choose **Create table**.

Step 3: Create the Amazon Cloud Map data service

In this step, you create a Amazon Cloud Map service and then register the DynamoDB table created in the last step as a service instance.

1. Open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/
2. From the list of namespaces, select the `cloudmap-tutorial` namespace and choose **View details**.
3. In the **Services** section, choose **Create service** and do the following.

   a. For **Service name**, enter `data-service`.

   b. Leave the rest of the default values and choose **Create service**.
4. In the **Services** section, select the `data-service` service and choose **View details**.
5. In the **Service instances** section, choose **Register service instance**.

6. On the **Register service instance** page, do the following.

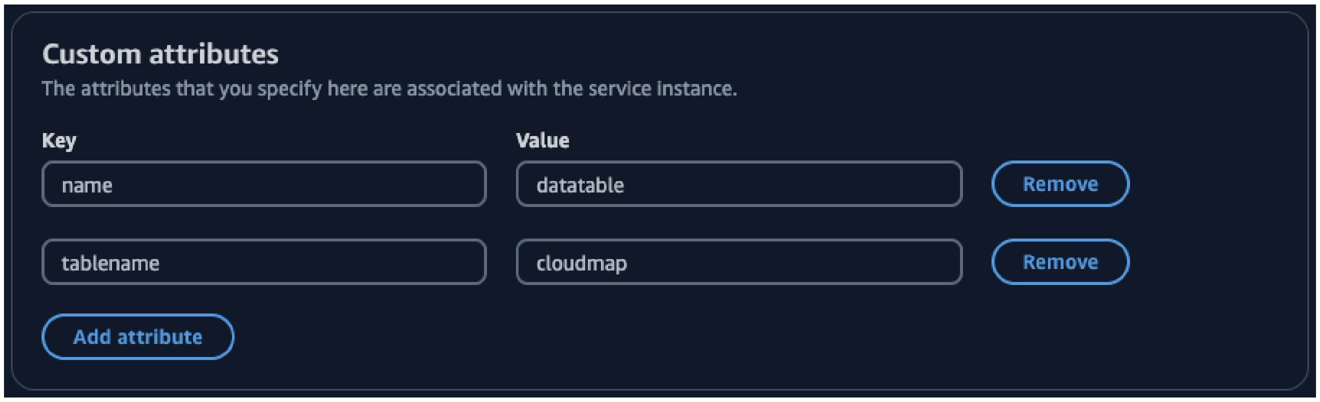
   a. For **Instance type**, select **Identifying information for another resource**.

   b. For **Service instance id**, specify `data-instance`.

   c. In the **Custom attributes** section, specify the following key-value pairs.

      - **key** = `name`, **value** = `datatable`
      - **key** = `tablename`, **value** = `cloudmap`

   d. Verify the attributes match the image below and choose **Register service instance**.



Custom attributes

The attributes that you specify here are associated with the service instance.

Key	Value	
name	datatable	Remove
tablename	cloudmap	Remove

Step 4: Create an Amazon Lambda execution role

In this step, you create an IAM role that the Amazon Lambda function we create in the next step uses. You can name the role `cloudmap-role` and omit the permissions boundary as this IAM role is only used for this tutorial and you can delete it afterwards.

To create the service role for Lambda (IAM console)

1. Sign in to the Amazon Web Services Management Console and open the IAM console at https://console.amazonaws.cn/iam/.
2. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
3. For **Trusted entity type**, choose **Amazon Web Service**.
4. For **Service or use case**, choose **Lambda**, and then choose the **Lambda** use case.
5. Choose **Next**.
6. Search for, and select the box next to, the `PowerUserAccess` policy and then choose **Next**.
7. Choose **Next**.
8. For **Role name**, specify `cloudmap-tutorial-role`.

9. Review the role, and then choose **Create role**.

Step 5: Create the Lambda function to write data

In this step, you create a Lambda function that writes data to the DynamoDB table by using the Amazon Cloud Map API to query the Amazon Cloud Map service you created.

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 left navigation, choose **Functions**, **Create function**.
3. On the **Create function** page, do the following.
    a. Select **Author from scratch**.
    b. For **Function name**, specify `writefunction`.
    c. For **Runtime**, select `Python 3.12`.
    d. For **Architecture**, select `x86_64`.
    e. In the **Permissions** section, do the following.
        i. Expand the **Change default execution role** option and select **Use an existing role**.
        ii. For **Existing role**, use the dropdown menu to select the IAM role you created in Step 4: Create an Amazon Lambda execution role.
        iii. Leave the rest of the default values and choose **Create function**.
    f. On the **Code** tab, in the **Code source** section, update the example code to reflect the following Python code. Take note that you're specifying the `datatable` custom attribute you associated with the Amazon Cloud Map service instance you created for the DynamoDB table.

```
import json
import boto3
import random

def lambda_handler(event, context):

    serviceclient = boto3.client('servicediscovery')

    response = serviceclient.discover_instances(
        NamespaceName='cloudmap-tutorial',
        ServiceName='data-service',
        QueryParameters={ 'name': 'datatable' })

    tablename = response["Instances"][0]["Attributes"]["tablename"]

    dynamodbclient = boto3.resource('dynamodb')

    table = dynamodbclient.Table('cloudmap-table')

    response = table.put_item(
        Item={ 'id': str(random.randint(1,100)), 'todo': event })

    return {
        'statusCode': 200,
        'body': json.dumps(response)
    }
```

g. Choose **Deploy** to update the function.

Step 6: Create the Amazon Cloud Map app service

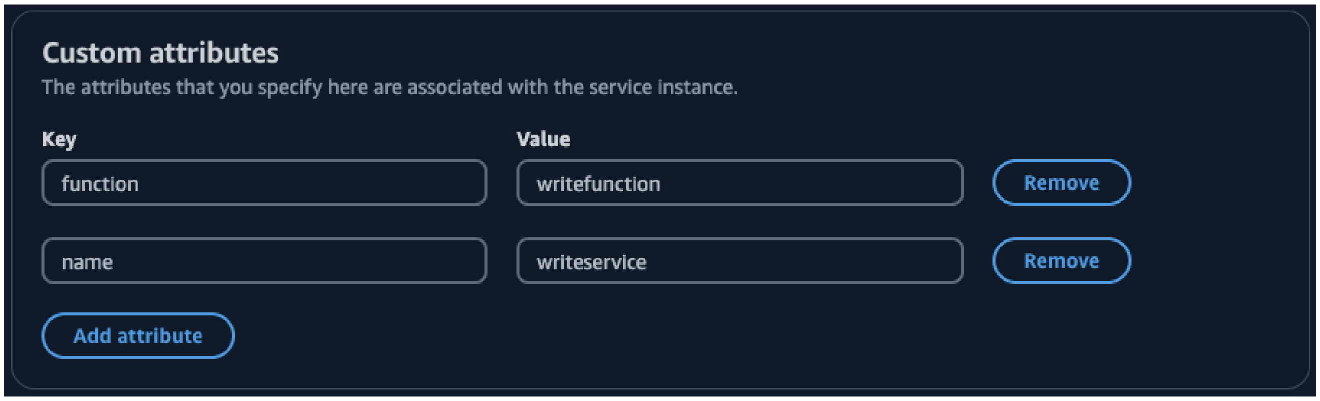
In this step, you create an Amazon Cloud Map service and then register the Lambda write function as a service instance.

1. Open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/
2. In the left navigation, choose **Namespaces**.
3. From the list of namespaces, select the `cloudmap-tutorial` namespace and choose **View details**.
4. In the **Services** section, choose **Create service** and do the following.
   a. For **Service name**, enter `app-service`.
   b. Leave the rest of the default values and choose **Create service**.
5. In the **Services** section, select the `app-service` service and choose **View details**.
6. In the **Service instances** section, choose **Register service instance**.
7. On the **Register service instance** page, do the following.
   a. For **Instance type**, select **Identifying information for another resource**.
   b. For **Service instance id**, specify `write-instance`.

c. In the **Custom attributes** section, specify the following key-value pairs.

- **key** = `name`, **value** = `writeservice`
- **key** = `function`, **value** = `writefunction`

d. Verify the attributes match the image below and choose **Register service instance**.


Custom attributes
The attributes that you specify here are associated with the service instance.
Key
Value
Remove
Remove
Add attribute

Key	Value
function	writefunction
name	writeservice

Step 7: Create the Lambda function to read data

In this step, you create a Lambda function that writes data to the DynamoDB table you created.

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 left navigation, choose **Functions**, **Create function**.
3. On the **Create function** page, do the following.

   a. Select **Author from scratch**.

   b. For **Function name**, specify `readfunction`.

   c. For **Runtime**, select `Python 3.12`.

   d. For **Architecture**, select `x86_64`.

   e. In the **Permissions** section, do the following.

      i. Expand the **Change default execution role** option and select **Use an existing role**.

      ii. For **Existing role**, use the dropdown menu to select the IAM role you created in Step 4: Create an Amazon Lambda execution role.

      iii. Leave the rest of the default values and choose **Create function**.

f. On the **Code** tab, in the **Code source** section, update the example code to reflect the following Python code.

```
import json
import boto3

def lambda_handler(event, context):
    serviceclient = boto3.client('servicediscovery')

    response = serviceclient.discover_instances(NamespaceName='cloudmap-
tutorial', ServiceName='data-service', QueryParameters={ 'name': 'datatable' })

    tablename = response["Instances"][0]["Attributes"]["tablename"]

    dynamodbclient = boto3.resource('dynamodb')

    table = dynamodbclient.Table('cloudmap-table')

    response = table.get_item(Key={'id': event})

    return {
        'statusCode': 200,
        'body': json.dumps(response)
    }
```

g. Choose **Deploy** to update the function.

Step 8: Create an Amazon Cloud Map service instance

In this step, you register the Lambda read function as a service instance in the `app-service` service you previously created.

1. Open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/
2. In the left navigation, choose **Namespaces**.
3. From the list of namespaces, select the `cloudmap-tutorial` namespace and choose **View details**.
4. In the **Services** section, select the `app-service` service and choose **View details**.
5. In the **Service instances** section, choose **Register service instance**.
6. On the **Register service instance** page, do the following.

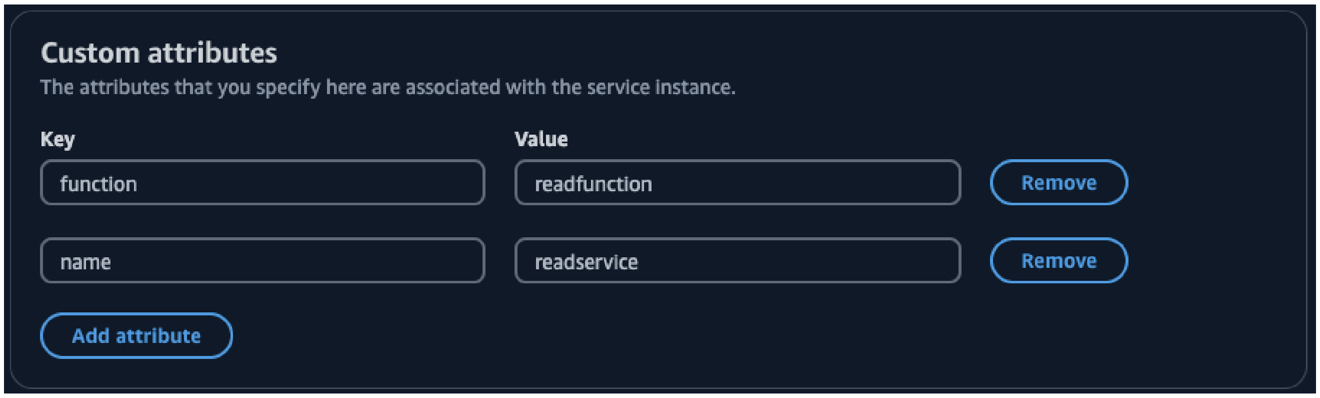
a. For **Instance type**, select **Identifying information for another resource**.

b. For **Service instance id**, specify `read-instance`.

c. In the **Custom attributes** section, specify the following key-value pairs.

- **key** = name, **value** = `readservice`
- **key** = `function`, **value** = `readfunction`

d. Verify the attributes match the image below and choose **Register service instance**.


Custom attributes
The attributes that you specify here are associated with the service instance.

Key	Value
function	readfunction
name	readservice

Step 9: Create a development environment

Amazon Cloud9 is an integrated development environment (IDE) managed by Amazon. The Amazon Cloud9 IDE provides the software and toooling needed for dynamic programming. In this step, we create an Amazon Cloud9 environment and configure it with the Amazon SDK for Python (Boto3) which you will to program with the Amazon API.

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud9 console at https://console.amazonaws.cn/cloud9/.
2. In the left navigation menu, select **My environments** and then choose **Create environment**.
3. On the **Create environment** page, do the following to create your development environment.

   a. For **Name**, use `cloudmap-tutorial`.

   b. For **Environment type**, select **New EC2 instance**.

   c. For **Instance type**, select **t2.micro**.

   d. For **Platform**, use the drop down menu to select **Ubuntu Server 22.04 LTS**.

   e. Leave the rest of the default selections and choose **Create**.

4. Once your Amazon Cloud9 environment is created, select the `cloudmap-tutorial` environment and choose **Open in Cloud9**. This opens the development environment in a new tab and provides you with a bash shell to work with.

   Important

   If you have issues opening your Amazon Cloud9 environment, see Amazon Cloud9 troubleshooting: Can't open an environment in the *Amazon Cloud9 User Guide*.

5. Using the bash shell, run the following commands to configure the environment.

   a. Update the environment.

   ```
   sudo apt-get -y update
   ```

   b. Verify that `python3` is installed.

   ```
   python3 --version
   ```

   c. Install the Boto3 package in the environment.

   ```
   sudo apt install -y python3-boto3
   ```

Step 10: Create a frontend client

Using the Amazon Cloud9 development environment created in the previous step, you create a frontend client that uses code that discovers the services you configured in Amazon Cloud Map and makes calls to these services.

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud9 console at https://console.amazonaws.cn/cloud9/.
2. In the left navigation menu, select **My environments** and then select your `cloudmap-tutorial` environment and choose **Open in Cloud9**.
3. In the Amazon Cloud9 environment, in the **File** menu, choose **New file** which creates a file named `Untitled1`.
4. In the `Untitled1` file, copy and paste the following code. This code discovers the Lambda function to write data by searching for the custom attribute `name=writeservice` in the

`app-service` service. The name of the Lambda function is returned which is responsible for writing data to the DynamoDB table. Then the Lambda function is invoked, passing a sample payload.

```
import boto3

serviceclient = boto3.client('servicediscovery')

response = serviceclient.discover_instances(NamespaceName='cloudmap-tutorial',
 ServiceName='app-service', QueryParameters={ 'name': 'writeservice' })

functionname = response["Instances"][0]["Attributes"]["function"]

lambdaclient = boto3.client('lambda')

resp = lambdaclient.invoke(FunctionName=functionname, Payload='"This is a test
 data"')

print(resp["Payload"].read())
```

5. From the **File** menu, choose **Save As...** and save the file as `writeclient.py`.
6. From the bash shell in your Amazon Cloud9 environment, use the following command to run the Python code.

```
python3 writeclient.py
```

The output should be a `200` response, similar to the following.

```
b'{"statusCode": 200, "body": "{\\"ResponseMetadata\\": {\\"RequestId\\": \
\"Q0M038IT0BPBVBJK8OCKK6I6M7VV4KQNSO5AEMVJF66Q9ASUAAJG\\", \\"HTTPStatusCode\
\": 200, \\"HTTPHeaders\\": {\\"server\\": \\"Server\\", \\"date\\": \\"Wed, 06
 Mar 2024 22:46:09 GMT\\", \\"content-type\\": \\"application/x-amz-json-1.0\\",
 \\"content-length\\": \\"2\\", \\"connection\\": \\"keep-alive\\", \\"x-amzn-
requestid\\": \\"Q0M038IT0BPBVBJK8OCKK6I6M7VV4KQNSO5AEMVJF66Q9ASUAAJG\\", \\"x-amz-
crc32\\": \\"2745614147\\"}, \\"RetryAttempts\\": 0}}"}'
```

7. To verify the write was successful in the previous step, create a read client.

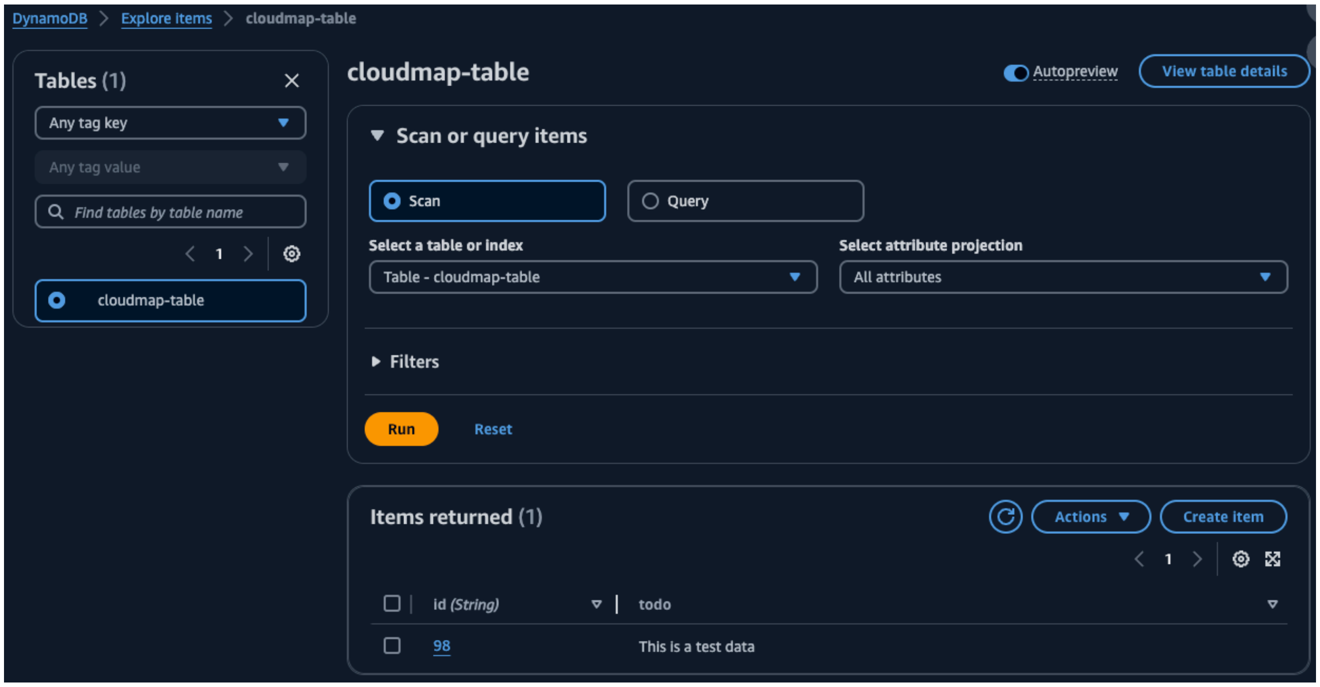
   a. Sign in to the Amazon Web Services Management Console and open the DynamoDB console at https://console.amazonaws.cn/dynamodb/.

   b. In the left navigation pane, choose **Tables**.

c. From the list of tables, select your **cloudmap-table** and use the **Actions** menu to choose **Explore items**.

d. In the **Items returned** section, take note of the numerical value in the **id(String)** column.

   The following shows an example, where the **id(String)** value is 98.



cloudmap-table

Select a table or index: Table - cloudmap-table

Select attribute projection: All attributes

Items returned (1)

id (String)	todo
98	This is a test data

e. In the Amazon Cloud9 environment, in the **File** menu, choose **New file** which creates a file named `Untitled1`.

f. In the `Untitled1` file, copy and paste the following code. Replace the `Payload` value with the `id (String)` value from your DynamoDB table in the previous step. This code reads from the table and will return the value that you wrote to the table in the previous step.

```
import boto3

serviceclient = boto3.client('servicediscovery')

response = serviceclient.discover_instances(NamespaceName='cloudmap-tutorial',
 ServiceName='app-service', QueryParameters={ 'name': 'readservice' })

functionname = response["Instances"][0]["Attributes"]["function"]

lambdaclient = boto3.client('lambda')
```

```
resp = lambdaclient.invoke(FunctionName=functionname,
 InvocationType='RequestResponse', Payload='"98"')

print(resp["Payload"].read())
```

g. From the **File** menu, choose **Save As...** and save the file as `readclient.py`.

h. From the bash shell in your Amazon Cloud9 environment, use the following command to run the Python code.

```
python3 readclient.py
```

The output should look similar to the following.

```
b'{"statusCode": 200, "body": "{\\"Item\\": {\\"id\\": \\"98\\", \\"todo\
\": \\"This is a test data\\"}, \\"ResponseMetadata\\": {\\"RequestId\\": \
\"JSO5DLRGF0JUPQN4NCH369ABMBVV4KQNSO5AEMVJF66Q9ASUAAJG\\", \\"HTTPStatusCode\
\": 200, \\"HTTPHeaders\\": {\\"server\\": \\"Server\\", \\"date\\": \\"Wed, 06
 Mar 2024 23:03:38 GMT\\", \\"content-type\\": \\"application/x-amz-json-1.0\
\", \\"content-length\\": \\"61\\", \\"connection\\": \\"keep-alive\\", \\"x-
amzn-requestid\\": \\"JSO5DLRGF0JUPQN4NCH369ABMBVV4KQNSO5AEMVJF66Q9ASUAAJG\\",
 \\"x-amz-crc32\\": \\"3104232745\\"}, \\"RetryAttempts\\": 0}}"}'
```

Step 11: Clean up the resources

Once you have completed the tutorial, to ensure you don't incur any additional charges, you can delete the resources. Amazon Cloud Map requires that you clean them up in reverse order, the service instances first, then the services, and finally the namespace. The following steps walk you through cleaning up the Amazon Cloud Map, Lambda, DynamoDB, and Amazon Cloud9 resources used in this tutorial.

To delete the Amazon Cloud9 resource

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud9 console at https://console.amazonaws.cn/cloud9/.
2. In the left navigation menu, select **My environments**.
3. Select your `cloudmap-tutorial` environment and choose **Delete**.
4. Confirm the deletion by typing `Delete` and then choose **Delete**.

To delete the Lambda functions

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 left navigation, choose **Functions**.
3. Select both the `writefunction` and `readfunction` functions.
4. From the **Actions** menu, choose **Delete**.
5. Confirm the deletion by typing `delete` and then choose **Delete**.

To delete the DynamoDB table

1. Sign in to the Amazon Web Services Management Console and open the DynamoDB console at https://console.amazonaws.cn/dynamodb/.
2. In the left navigation pane, choose **Tables**.
3. Select the `cloudmap-table` table and choose **Delete**.
4. Confirm the deletion by typing `confirm` and then choose **Delete**.

To delete the Amazon Cloud Map resources

1. Sign in to the Amazon Web Services Management Console and open the Amazon Cloud Map console at https://console.amazonaws.cn/cloudmap/.
2. From the list of namespaces, select the `cloudmap-tutorial` namespace and choose **View details**.
3. On the namespace details page, from the list of services, select the `data-service` service and choose **View details**.
4. In the **Service instances** section, select the `data-instance` instance and choose **Deregister**.
5. Using the breadcrumb at the top of the page, select **cloudmap-tutorial.com** to navigate back to the namespace detail page.
6. On the namespace details page, from the list of services, select the **data-service** service and choose **Delete**.
7. Repeat steps 3-6 for the `app-service` service and the `write-instance` and `read-instance` service instances.
8. In the left navigation, choose **Namespaces**.
9. Select the `cloudmap-tutorial` namespace and choose **Delete**.

Security in Amazon Cloud Map

Cloud security at Amazon is the highest priority. As an Amazon customer, you benefit from a data center and network architecture that's 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 Cloud Map, 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 company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon Cloud Map. The following topics show you how to configure Amazon Cloud Map to meet your security and compliance objectives. You also learn how to use other Amazon services that help you to monitor and secure your Amazon Cloud Map resources.

Topics

Amazon Identity and Access Management in Amazon Cloud Map

To perform any action on Amazon Cloud Map resources, such as registering a domain or updating a record, Amazon Identity and Access Management (IAM) requires you to authenticate that you're an approved Amazon user. If you're using the Amazon Cloud Map console, you authenticate your identity by providing your Amazon user name and a password. If you're accessing Amazon Cloud Map programmatically, your application authenticates your identity for you by using access keys or by signing requests.

After you authenticate your identity, IAM controls your access to Amazon by verifying that you have permissions to perform actions and to access resources. If you are an account administrator, you can use IAM to control the access of other users to the resources that are associated with your account.

This chapter explains how to use IAM and Amazon Cloud Map to help secure your resources.

Topics

Authentication

You can access Amazon as any of the following:

- **Amazon Web Services account root user –** When you first create an Amazon account, you begin with a single sign-in identity that has complete access to all Amazon 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. 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*.

- **IAM user** – An IAM user is an identity within your Amazon account that has specific custom permissions (for example, permissions to create an HTTP namespace in Amazon Cloud Map). You can use your IAM sign-in credentials to secure Amazon webpages like the Amazon Web Services Management Console, Amazon Discussion Forums, or the Amazon Web Services Support Center.

  In addition to sign-in credentials, you can also generate access keys for each user. You can use these keys when you access Amazon services programmatically, either through one of the several SDKs or by using the Amazon Command Line Interface. The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use Amazon tools, you must sign the request yourself. Amazon Cloud Map supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the *Amazon Web Services General Reference*.
- **IAM role** – An IAM role is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user in that it is an Amazon identity with permissions policies that determine what the identity can and cannot do in Amazon. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session. IAM roles with temporary credentials are useful in the following situations:

  - **Federated user access** – Instead of creating an IAM user, you can use existing user identities from Amazon Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. Amazon assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the *IAM User Guide*.
  - **Amazon service access** – You can use an IAM role in your account to grant an Amazon service permissions to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an Amazon Service in the *IAM User Guide*.
  - **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an Amazon EC2 instance and making Amazon API requests. This is preferable to storing access keys within the Amazon EC2 instance. To assign an Amazon role to an Amazon EC2 instance and make it available to all of its applications, you create an instance profile that's attached to the instance. An instance profile

contains the role and enables programs that are running on the Amazon 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*.

Access Control

To create, update, delete, or list Amazon Cloud Map resources, you need permissions to perform the action, and you need permission to access the corresponding resources. In addition, to perform the action programmatically, you need valid access keys.

The following sections describe how to manage permissions for Amazon Cloud Map. We recommend that you read the overview first.

- Overview of Managing Access Permissions to Your Amazon Cloud Map Resources
- Using Identity-Based Policies (IAM Policies) for Amazon Cloud Map
- Amazon Cloud Map API Permissions: Actions, Resources, and Conditions Reference

Overview of Managing Access Permissions to Your Amazon Cloud Map Resources

Every Amazon resource is owned by an Amazon account, and permissions to create or access a resource are governed by permissions policies.

Note

An *account administrator* (or administrator user) is a user that has administrator privileges. For more information about administrators, see IAM Best Practices in the *IAM User Guide*.

When you grant permissions, you decide who gets the permissions, the resources they get permissions for, and the actions that they get permissions to perform.

Topics

- ARNs for Amazon Cloud Map Resources
- Understanding Resource Ownership
- Managing Access to Resources

- Specifying Policy Elements: Resources, Actions, Effects, and Principals
- Specifying Conditions in an IAM Policy

ARNs for Amazon Cloud Map Resources

You can grant or deny resource-level permissions for namespaces and services for selected operations. For more information, see Amazon Cloud Map API Permissions: Actions, Resources, and Conditions Reference.

Understanding Resource Ownership

An Amazon account owns the resources that are created in the account, regardless of who created the resources. Specifically, the resource owner is the Amazon account of the principal entity (that is, the root user account, an IAM user, or an IAM role) that authenticates the resource creation request.

The following examples illustrate how this works:

- If you use the root user account credentials of your Amazon account to create an HTTP namespace, your Amazon account is the owner of the resource.
- If you create an IAM user in your Amazon account and grant permissions to create an HTTP namespace to that user, the user can create an HTTP namespace. However, your Amazon account, to which the user belongs, owns the HTTP namespace resource.
- If you create an IAM role in your Amazon account with permissions to create an HTTP namespace, anyone who can assume the role can create an HTTP namespace. Your Amazon account, to which the role belongs, owns the HTTP namespace resource.

Managing Access to Resources

A *permissions policy* specifies who has access to what. This section explains the options for creating permissions policies for Amazon Cloud Map. For general information about IAM policy syntax and descriptions, see the IAM Policy Reference in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based* policies (IAM policies), and policies attached to a resource are referred to as *resource-based* policies. Amazon Cloud Map supports only identity-based policies (IAM policies).

Topics

- Identity-Based Policies (IAM Policies)
- Resource-Based Policies

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – An account administrator can use a permissions policy that's associated with a particular user to grant permissions for that user to create Amazon Cloud Map resources.
- **Attach a permissions policy to a role (grant cross-account permissions)** – You can grant permission to perform Amazon Cloud Map actions to a user that was created by another Amazon account. To do so, you attach a permissions policy to an IAM role, and then you allow the user in the other account to assume the role. The following example explains how this works for two Amazon accounts, account A and account B:
  1. Account A administrator creates an IAM role and attaches to the role a permissions policy that grants permissions to create or access resources that are owned by account A.
  2. Account A administrator attaches a trust policy to the role. The trust policy identifies account B as the principal that can assume the role.
  3. Account B administrator can then delegate permissions to assume the role to users or groups in Account B. This allows users in account B to create or access resources in account A.

  For more information about how to delegate permissions to users in another Amazon account, see Access Management in the *IAM User Guide*.

The following example policy allows a user to perform the CreatePublicDnsNamespace action to create a public DNS namespace for any Amazon account. The Amazon Route 53 permissions are required because when you create a public DNS namespace, Amazon Cloud Map also creates a Route 53 hosted zone:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "servicediscovery:CreatePublicDnsNamespace",
            "route53:CreateHostedZone",
            "route53:GetHostedZone",
            "route53:ListHostedZonesByName"
         ],
         "Resource":"*"
      }
   ]
}
```

If you want the policy to instead apply to private DNS namespaces, you need to grant permissions to use the Amazon Cloud Map CreatePrivateDnsNamespace action. In addition, you grant permission to use the same Route 53 actions as in the previous example because Amazon Cloud Map creates a Route 53 private hosted zone. You also grant permission to use two Amazon EC2 actions, `DescribeVpcs` and `DescribeRegions`:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "servicediscovery:CreatePrivateDnsNamespace",
            "route53:CreateHostedZone",
            "route53:GetHostedZone",
            "route53:ListHostedZonesByName"
         ],
         "Resource":"*"
      },
      {
         "Effect": "Allow",
         "Action": [
            "ec2:DescribeVpcs",
            "ec2:DescribeRegions"
         ],
         "Resource":"*"
      }
   ]
}
```

For more information about attaching policies to identities for Amazon Cloud Map, see Using Identity-Based Policies (IAM Policies) for Amazon Cloud Map. For more information about users, groups, roles, and permissions, see Identities (Users, Groups, and Roles) in the *IAM User Guide*.

Resource-Based Policies

Other services, such as Amazon S3, also support attaching permissions policies to resources. For example, you can attach a policy to an S3 bucket to manage access permissions to that bucket. Amazon Cloud Map doesn't support attaching policies to resources.

Specifying Policy Elements: Resources, Actions, Effects, and Principals

Amazon Cloud Map includes API actions (see the Amazon Cloud Map API Reference) that you can use on each Amazon Cloud Map resource (see ARNs for Amazon Cloud Map Resources). You can grant a user or a federated user permissions to perform any or all of these actions. Note that some API actions, such as creating a public DNS namespace, require permissions to perform more than one action.

The following are the basic policy elements:

- **Resource** – You use an Amazon Resource Name (ARN) to identify the resource that the policy applies to. For more information, see ARNs for Amazon Cloud Map Resources.
- **Action** – You use action keywords to identify resource actions that you want to allow or deny. For example, depending on the specified `Effect`, the `servicediscovery:CreateHttpNamespace` permission allows or denies a user the ability to perform the Amazon Cloud Map CreateHttpNamespace action.
- **Effect** – You specify the effect, either allow or deny, when a user tries to perform the action on the specified resource. If you don't explicitly grant access to an action, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user can't access it, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only). Amazon Cloud Map doesn't support resource-based policies.

For more information about IAM policy syntax and descriptions, see the IAM Policy Reference in the *IAM User Guide*.

For a list of the Amazon Cloud Map API actions and the resources that they apply to, see Amazon Cloud Map API Permissions: Actions, Resources, and Conditions Reference.

Specifying Conditions in an IAM Policy

When you grant permissions, you can use the IAM policy language to specify when a policy should take effect. For example, you might want a policy to be applied only after a specified date, or you might want a policy to apply only to a specified namespace.

To express conditions, you use predefined condition keys. Amazon Cloud Map defines its own set of condition keys and also supports using some global condition keys. For more information, see the following topics:

- For information about Amazon Cloud Map condition keys, see Amazon Cloud Map API Permissions: Actions, Resources, and Conditions Reference.
- For information about Amazon global condition keys, see Amazon Global Condition Context Keys in the *IAM User Guide*.
- For information about specifying conditions in a policy language, IAM JSON Policy Elements: Condition in the *IAM User Guide*.

Using Identity-Based Policies (IAM Policies) for Amazon Cloud Map

This topic provides examples of identity-based policies that demonstrate how an account administrator can attach permissions policies to IAM identities (users, groups, and roles) and thereby grant permissions to perform actions on Amazon Cloud Map resources.

Important

We recommend that you first review the introductory topics that explain the basic concepts and options to manage access to your Amazon Cloud Map resources. For more information, see Overview of Managing Access Permissions to Your Amazon Cloud Map Resources.

Topics

- Permissions Required to Use the Amazon Cloud Map Console

The following example shows a permissions policy that grants a user permission to register, deregister, and register service instances. The `Sid`, or statement ID, is optional:

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
          "Sid" : "AllowInstancePermissions",
          "Effect": "Allow",
          "Action": [
             "servicediscovery:RegisterInstance",
             "servicediscovery:DeregisterInstance",
             "servicediscovery:DiscoverInstances",
             "servicediscovery:Get*",
             "servicediscovery:List*",
             "route53:GetHostedZone",
             "route53:ListHostedZonesByName",
             "route53:ChangeResourceRecordSets",
             "route53:CreateHealthCheck",
             "route53:GetHealthCheck",
             "route53:DeleteHealthCheck",
             "route53:UpdateHealthCheck",
             "ec2:DescribeInstances"
          ],
          "Resource": "*"
       }
    ]
}
```

The policy grants permissions to the actions that are required to register and manage service instances. The Route 53 permission is required if you're using public or private DNS namespaces because Amazon Cloud Map creates, updates, and deletes Route 53 records and health checks when you register and deregister instances. The wildcard character (*) in `Resource` grants access to all Amazon Cloud Map instances, and Route 53 records and health checks that are owned by the current Amazon account.

For a list of actions and the ARN that you specify to grant or deny permission to use each action, see Amazon Cloud Map API Permissions: Actions, Resources, and Conditions Reference.

Permissions Required to Use the Amazon Cloud Map Console

To grant full access to the Amazon Cloud Map console, you grant the permissions in the following permissions policy:

```
{
    "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "servicediscovery:*",
            "route53:GetHostedZone",
            "route53:ListHostedZonesByName",
            "route53:CreateHostedZone",
            "route53:DeleteHostedZone",
            "route53:ChangeResourceRecordSets",
            "route53:CreateHealthCheck",
            "route53:GetHealthCheck",
            "route53:DeleteHealthCheck",
            "route53:UpdateHealthCheck",
            "ec2:DescribeInstances",
            "ec2:DescribeVpcs",
            "ec2:DescribeRegions"
         ],
         "Resource":"*"
      }
   ]
}
```

Here's why the permissions are required:

`servicediscovery:*`

Lets you perform all Amazon Cloud Map actions.

`route53:CreateHostedZone`, `route53:GetHostedZone`, `route53:ListHostedZonesByName`, `route53:DeleteHostedZone`

Lets Amazon Cloud Map manage hosted zones when you create and delete public and private DNS namespaces.

`route53:CreateHealthCheck`, `route53:GetHealthCheck`, `route53:DeleteHealthCheck`, `route53:UpdateHealthCheck`

Lets Amazon Cloud Map manage health checks when you include Amazon Route 53 health checks when you create a service.

`ec2:DescribeVpcs` and `ec2:DescribeRegions`

Let Amazon Cloud Map manage private hosted zones.

Amazon managed policies for Amazon Cloud Map

An Amazon managed policy is a standalone policy that is created and administered by Amazon. Amazon managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that Amazon managed policies might not grant least-privilege permissions for your specific use cases because they're available for all Amazon customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in Amazon managed policies. If Amazon updates the permissions defined in an Amazon managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. Amazon is most likely to update an Amazon managed policy when a new Amazon Web Service is launched or new API operations become available for existing services.

For more information, see Amazon managed policies in the *IAM User Guide*.

Amazon managed policy: AWSCloudMapDiscoverInstanceAccess

You can attach `AWSCloudMapDiscoverInstanceAccess` to your IAM entities. Provides access to Amazon Cloud Map Discovery API.

To view the permissions for this policy, see AWSCloudMapDiscoverInstanceAccess in the *Amazon Managed Policy Reference*.

Amazon managed policy: AWSCloudMapReadOnlyAccess

You can attach `AWSCloudMapReadOnlyAccess` to your IAM entities. Grants read-only access to all Amazon Cloud Map actions.

To view the permissions for this policy, see AWSCloudMapReadOnlyAccess in the *Amazon Managed Policy Reference*.

Amazon managed policy: AWSCloudMapRegisterInstanceAccess

You can attach `AWSCloudMapRegisterInstanceAccess` to your IAM entities. Grants read-only access to namespaces and services and grants permission to register and deregister service instances.

To view the permissions for this policy, see AWSCloudMapRegisterInstanceAccess in the *Amazon Managed Policy Reference*.

Amazon managed policy: AWSCloudMapFullAccess

You can attach `AWSCloudMapFullAccess` to your IAM entities. Provides full access to all Amazon Cloud Map actions

To view the permissions for this policy, see AWSCloudMapFullAccess in the *Amazon Managed Policy Reference*.

Amazon Cloud Map updates to Amazon managed policies

View details about updates to Amazon managed policies for Amazon Cloud Map since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Amazon Cloud Map Document history page.

Change	Description	Date
AWSCloudMapDiscoverInstanceAccess, AWSCloudMapRegisterInstanceAccess, AWSCloudMapReadOnlyAccess – Updates to existing policies.	Amazon Cloud Map updated these policies to provide access to the new Amazon Cloud Map `DiscoverInstanceRevision` API operations.	August 15, 2023

Customer Managed Policy Examples

You can create your own custom IAM policies to allow permissions for Amazon Cloud Map 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 Amazon Cloud Map API, the Amazon SDKs, or the Amazon CLI. The following examples show permissions for several common use cases. For the policy that grants a user full access to Amazon Cloud Map, see Permissions Required to Use the Amazon Cloud Map Console.

Examples

- Example 1: Allow Read Access to All Amazon Cloud Map Resources

- Example 2: Allow Creation of All Types of Namespaces

Example 1: Allow Read Access to All Amazon Cloud Map Resources

The following permissions policy grants the user read-only access to all Amazon Cloud Map resources:

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "servicediscovery:Get*",
            "servicediscovery:List*",
            "servicediscovery:DiscoverInstances"
         ],
         "Resource":"*"
      }
   ]
}
```

Example 2: Allow Creation of All Types of Namespaces

The following permissions policy allows users to create all types of namespaces:

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "servicediscovery:CreateHttpNamespace",
            "servicediscovery:CreatePrivateDnsNamespace",
            "servicediscovery:CreatePublicDnsNamespace",
            "route53:CreateHostedZone",
            "route53:GetHostedZone",
            "route53:ListHostedZonesByName",
            "ec2:DescribeVpcs",
            "ec2:DescribeRegions"
         ],
         "Resource":"*"
```

```
        }
    ]
}
```

To provide access, add permissions to your users, groups, or roles:

- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Creating a role for a third-party identity provider (federation) in the *IAM User Guide*.
- IAM users:
  - Create a role that your user can assume. Follow the instructions in Creating a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

Amazon Cloud Map API Permissions: Actions, Resources, and Conditions Reference

When you set up Access Control and write a permissions policy that you can attach to an IAM identity (identity-based policies), you can use the following lists as a reference. The lists include each Amazon Cloud Map API action, the actions that you must grant permissions access to, and the Amazon resource that you must grant access to. You specify the actions in the `Action` field for the policy, and you specify the resource value in the `Resource` field for the policy.

You can use Amazon Cloud Map–specific condition keys in your IAM policies for some operations. For more information, see Amazon Cloud Map Condition Keys Reference. You can also use Amazon wide condition keys. For a complete list of Amazon wide keys, see Available Keys in the *IAM User Guide*.

To specify an action, use the `servicediscovery` prefix followed by the API action name, for example, `servicediscovery:CreatePublicDnsNamespace` and `route53:CreateHostedZone`.

Topics

Required Permissions for Amazon Cloud Map Actions

CreateHttpNamespace

Required Permissions (API Action):

- `servicediscovery:CreateHttpNamespace`

Resources: *

CreatePrivateDnsNamespace

Required Permissions (API Action):

- `servicediscovery:CreatePrivateDnsNamespace`
- `route53:CreateHostedZone`
- `route53:GetHostedZone`
- `route53:ListHostedZonesByName`
- `ec2:DescribeVpcs`
- `ec2:DescribeRegions`

Resources: *

CreatePublicDnsNamespace

Required Permissions (API Action):

- `servicediscovery:CreatePublicDnsNamespace`
- `route53:CreateHostedZone`
- `route53:GetHostedZone`
- `route53:ListHostedZonesByName`

Resources: *

CreateService

Required Permissions (API Action): `servicediscovery:CreateService`

Resources: *

DeleteNamespace

Required Permissions (API Action):

- `servicediscovery:DeleteNamespace`

  Resources: *, `arn:aws-cn:servicediscovery:`*`region`*`:`*`account-id`*`:namespace/`*`namespace-id`*

[DeleteService]

  Required Permissions (API Action): `servicediscovery:DeleteService`

  Resources: *, `arn:aws-cn:servicediscovery:`*`region`*`:`*`account-id`*`:service/`*`service-id`*

[DeregisterInstance]

  Required Permissions (API Action):

- `servicediscovery:DeregisterInstance`
- `route53:GetHealthCheck`
- `route53:DeleteHealthCheck`
- `route53:UpdateHealthCheck`
- `route53:ChangeResourceRecordSets`

  Resources: *

[DiscoverInstances]

  Required Permissions (API Action): `servicediscovery:DiscoverInstances`

  Resources: *

[GetInstance]

  Required Permissions (API Action): `servicediscovery:GetInstance`

  Resources: *

[GetInstancesHealthStatus]

  Required Permissions (API Action): `servicediscovery:GetInstancesHealthStatus`

  Resources: *

[GetNamespace]

  Required Permissions (API Action): `servicediscovery:GetNamespace`

Resources: *, `arn:aws-cn:servicediscovery:`*`region`*`:`*`account-id`*`:namespace/`*`namespace-id`*

[GetOperation]

Required Permissions (API Action): `servicediscovery:GetOperation`

Resources: *

[GetService]

Required Permissions (API Action): `servicediscovery:GetService`

Resources: *, `arn:aws-cn:servicediscovery:`*`region`*`:`*`account-id`*`:service/`*`service-id`*

[ListInstances]

Required Permissions (API Action): `servicediscovery:ListInstances`

Resources: *

[ListNamespaces]

Required Permissions (API Action): `servicediscovery:ListNamespaces`

Resources: *

[ListOperations]

Required Permissions (API Action): `servicediscovery:ListOperations`

Resources: *

[ListServices]

Required Permissions (API Action): `servicediscovery:ListServices`

Resources: *

[ListTagsForResource]

Required Permissions (API Action): `servicediscovery:ListTagsForResource`

Resources: *

[RegisterInstance]

Required Permissions (API Action):

- `servicediscovery:RegisterInstance`
- `route53:GetHealthCheck`
- `route53:CreateHealthCheck`
- `route53:UpdateHealthCheck`
- `route53:ChangeResourceRecordSets`
- `ec2:DescribeInstances`

Resources: *

[TagResource]

Required Permissions (API Action): `servicediscovery:TagResource`

Resources: *

[UntagResource]

Required Permissions (API Action): `servicediscovery:UntagResource`

Resources: *

[UpdateHttpNamespace]

Required Permissions (API Action): `servicediscovery:UpdateHttpNamespace`

Resources: *, `arn:aws-cn:servicediscovery:`*`region`*`:`*`account-id`*`:namespace/`*`namespace-id`*

[UpdateInstanceCustomHealthStatus]

Required Permissions (API Action): `servicediscovery:UpdateInstanceCustomHealthStatus`

Resources: *

[UpdatePrivateDnsNamespace]

Required Permissions (API Action):

- `servicediscovery:UpdatePrivateDnsNamespace`

- `route53:ChangeResourceRecordSets`

  Resources: *, `arn:aws-cn:servicediscovery:`*`region`*`:`*`account-id`*`:namespace/`*`namespace-id`*

UpdatePublicDnsNamespace

  Required Permissions (API Action):

- `servicediscovery:UpdatePublicDnsNamespace`
- `route53:ChangeResourceRecordSets`

  Resources: *, `arn:aws-cn:servicediscovery:`*`region`*`:`*`account-id`*`:namespace/`*`namespace-id`*

UpdateService

  Required Permissions (API Action):

- `servicediscovery:UpdateService`
- `route53:GetHealthCheck`
- `route53:CreateHealthCheck`
- `route53:DeleteHealthCheck`
- `route53:UpdateHealthCheck`
- `route53:ChangeResourceRecordSets`

  Resources: *, `arn:aws-cn:servicediscovery:`*`region`*`:`*`account-id`*`:service/`*`service-id`*

Amazon Cloud Map Condition Keys Reference

Amazon Cloud Map defines the following condition keys that can be used in the `Condition` element of an IAM policy for specific Amazon Cloud Map actions. You can use these keys to further refine the conditions under which the policy statement applies. For details on which Amazon Cloud Map actions accept these condition keys, see Actions defined by Amazon Cloud Map. For more information about condition keys in general, see Specifying Conditions in an IAM Policy.

`servicediscovery:NamespaceArn`

A filter that lets you get objects by specifying the Amazon Resource Name (ARN) for the related namespace.

`servicediscovery:NamespaceName`

A filter that lets you get objects by specifying the name of the related namespace.

`servicediscovery:ServiceArn`

A filter that lets you get objects by specifying the Amazon Resource Name (ARN) for the related service.

`servicediscovery:ServiceName`

A filter that lets you get objects by specifying the name of the related service.

Logging and Monitoring in Amazon Cloud Map

Monitoring is an important part of maintaining the reliability, availability, and performance of 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. However, before you start monitoring, you should create a monitoring plan that includes answers to the following questions:

- What are your monitoring goals?
- What resources will you monitor?
- How often will you monitor these resources?
- What monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

Compliance Validation for Amazon Cloud Map

The security and compliance of Amazon Cloud Map is assessed by third-party auditors as part of multiple Amazon compliance programs, including Health Insurance Portability and Accountability Act (HIPAA), Payment Card Industry Data Security Standard (PCI DSS), ISO, and FIPS.

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 Amazon services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. Amazon provides 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 Whitepaper – This paper describes how companies can use Amazon to create HIPAA-compliant applications.
- 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 provides a comprehensive view of your security state within Amazon that helps you check your compliance with security industry standards and best practices.

Resilience in Amazon Cloud Map

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.

Amazon Cloud Map is primarily a global service. However, you can use Amazon Cloud Map to create Route 53 health checks that check the health of resources in specific Regions, such as Amazon EC2 instances and Elastic Load Balancing load balancers.

For more information about Amazon Regions and Availability Zones, see Amazon Global Infrastructure.

Infrastructure Security in Amazon Cloud Map

As a managed service, Amazon Cloud Map 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 Amazon Cloud Map 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.

You can improve the security posture of your VPC by configuring Amazon Cloud Map to use an interface VPC endpoint. For more information, see Access Amazon Cloud Map using an interface endpoint (Amazon PrivateLink).

Access Amazon Cloud Map using an interface endpoint (Amazon PrivateLink)

You can use Amazon PrivateLink to create a private connection between your VPC and Amazon Cloud Map. You can access Amazon Cloud Map as if it were in your VPC, without the use of an internet gateway, NAT device, VPN connection, or Amazon Direct Connect connection. Instances in your VPC don't need public IP addresses to access Amazon Cloud Map.

You establish this private connection by creating an *interface endpoint*, powered by Amazon PrivateLink. We create an endpoint network interface in each subnet that you enable for the interface endpoint. These are requester-managed network interfaces that serve as the entry point for traffic destined for Amazon Cloud Map.

For more information, see Access Amazon Web Services through Amazon PrivateLink in the *Amazon PrivateLink Guide*.

Considerations for Amazon Cloud Map

Before you set up an interface endpoint for Amazon Cloud Map, review Considerations in the *Amazon PrivateLink Guide*.

If your Amazon VPC doesn't have an internet gateway and your tasks use the `awslogs` log driver to send log information to CloudWatch Logs, you must create an interface VPC endpoint for CloudWatch Logs. For more information, see Using CloudWatch Logs with Interface VPC Endpoints in the Amazon CloudWatch Logs User Guide.

VPC endpoints don't support Amazon cross-Region requests. Ensure that you create your endpoint in the same Region where you plan to issue your API calls to Amazon Cloud Map.

VPC endpoints only support Amazon-provided DNS through Amazon Route 53. If you want to use your own DNS, you can use conditional DNS forwarding. For more information, see DHCP Options Sets in the Amazon VPC User Guide.

The security group attached to the VPC endpoint must allow incoming connections on port 443 from the private subnet of the Amazon VPC.

Create an interface endpoint for Amazon Cloud Map

You can create an interface endpoint for Amazon Cloud Map using either the Amazon VPC console or the Amazon Command Line Interface (Amazon CLI). For more information, see Create an interface endpoint in the *Amazon PrivateLink Guide*.

Create an interface endpoint for Amazon Cloud Map using the following service names:

Note

`DiscoverInstances` API won't be available over these two endpoints.

```
com.amazonaws.region.servicediscovery
```

```
com.amazonaws.region.servicediscovery-fips
```

Create an interface endpoint for Amazon Cloud Map data plane to access the `DiscoverInstances` API using the following service names:

```
com.amazonaws.region.data-servicediscovery
```

```
com.amazonaws.region.data-servicediscovery-fips
```

Note

You'll need to disable host prefix injection when you call `DiscoverInstances` with the regional or zonal VPCE DNS names for data plane endpoints. The Amazon CLI and Amazon SDKs prepend the service endpoint with various host prefixes when you call each API operation, which produces invalid URLS when you specify a VPC endpoint.

If you enable private DNS for the interface endpoint, you can make API requests to Amazon Cloud Map using its default Regional DNS name. For example, `servicediscovery.us-east-1.amazonaws.com`.

VPCE Amazon PrivateLink connection is supported in any Region where Amazon Cloud Map is supported; however, a customer needs to check which Availability Zones support VPCE before defining an endpoint. To find out which Availability Zones are supported with interface VPC endpoints in a Region, use the describe-vpc-endpoint-services command or use the Amazon Web Services Management Console. For example, the following commands return the availability zones to which you can deploy an Amazon Cloud Map interface VPC endpoints within the US East (Ohio) Region:

```
aws --region us-east-2 ec2 describe-vpc-endpoint-services --query 'ServiceDetails[?
ServiceName==`com.amazonaws.us-east-2.servicediscovery`].AvailabilityZones[]'
```

Logging Amazon Cloud Map API calls using Amazon CloudTrail

Amazon Cloud Map 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 Amazon Cloud Map as events. The calls captured include calls from the Amazon Cloud Map console and code calls to the Amazon Cloud Map API operations. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Cloud Map, 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.

Amazon Cloud Map data events in CloudTrail

Data events provide information about the resource operations performed on or in a resource (for example, discovering a registered instance in a namespace). 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 Amazon Cloud Map 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 Amazon Cloud Map 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
AwsApiCall	`AWS::ServiceDiscovery::Namespace`	• DiscoverInstances • DiscoverInstancesRevision
AwsApiCall	`AWS::ServiceDiscovery::Service`	• DiscoverInstances • DiscoverInstancesRevision

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*.

The following example shows how to configure advanced event selectors to log all Amazon Cloud Map data events.

```
"AdvancedEventSelectors":
[
    {
        "Name": "Log all AWS Cloud Map data events",
        "FieldSelectors": [
            { "Field": "eventCategory", "Equals": ["Data"] },
            { "Field": "resources.type", "Equals":
 ["AWS::ServiceDiscovery::Namespace"] }
        ]
    }
]
```

Amazon Cloud Map 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 Cloud Map logs all Amazon Cloud Map control plane operations as management events. For a list of the Amazon Cloud Map control plane operations that Amazon Cloud Map logs to CloudTrail, see the Amazon Cloud Map API Reference.

Amazon Cloud Map 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.

The following example shows a CloudTrail management event that demonstrates the `CreateHTTPNamespace` operation.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:alejandro_rosalez",
        "arn": "arn:aws:sts::111122223333:assumed-role/users/alejandro_rosalez",
        "accountId": "111122223333",
        "accessKeyId": "AIDACKCEVSQ6C2EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA123456789EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/readonly-role",
                "accountId": "111122223333",
                "userName": "alejandro_rosalez"
            },
            "attributes": {
                "creationDate": "2024-03-19T16:15:37Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2024-03-19T19:23:13Z",
    "eventSource": "servicediscovery.amazonaws.com",
    "eventName": "CreateHttpNamespace",
    "awsRegion": "eu-west-3",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36",
    "requestParameters": {
        "name": "example-namespace",
        "creatorRequestId": "eda8b524-ca14-4f68-a176-dc4dfd165c26",
        "tags": []
    },
    "responseElements": {
        "operationId": "7xm4i7ghhkaalma666nrg6itf2eylcbp-gwipo38o"
    },
    "requestID": "641274d0-dbbe-4e64-9b53-685769a086c7",
    "eventID": "4a1ab076-ef1b-4bcf-aa95-cec5fb64f2bd",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "servicediscovery.eu-west-3.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
```

```
}
```

The following example shows a CloudTrail data event that demonstrates the `DiscoverInstances` operation.

```
{
            "eventVersion": "1.09",
            "userIdentity": {
                "type": "AssumedRole",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE:alejandro_rosalez",
                "arn": "arn:aws:sts::111122223333:assumed-role/role/Admin",
                "accountId": "111122223333",
                "accessKeyId": "AIDACKCEVSQ6C2EXAMPLE",
                "sessionContext": {
                    "sessionIssuer": {
                        "type": "Role",
                        "principalId": "AROA123456789EXAMPLE",
                        "arn": "arn:aws:iam::"111122223333":role/Admin",
                        "accountId": "111122223333",
                        "userName": "Admin"
                    },
                    "attributes": {
                        "creationDate": "2024-03-19T16:15:37Z",
                        "mfaAuthenticated": "false"
                    }
                }
            },
            "eventTime": "2024-03-19T21:19:12Z",
            "eventSource": "servicediscovery.amazonaws.com",
            "eventName": "DiscoverInstances",
            "awsRegion": "eu-west-3",
            "sourceIPAddress": "13.38.34.79",
            "userAgent": "Boto3/1.20.34 md/Botocore#1.34.60 ua/2.0 os/linux#6.5.0-1014-
aws md/arch#x86_64 lang/python#3.10.12 md/pyimpl#CPython cfg/retry-mode#legacy
 Botocore/1.34.60",
            "requestParameters": {
                "namespaceName": "example-namespace",
                "serviceName": "example-service",
                "queryParameters": {"example-key": "example-value"}
            },
            "responseElements": null,
            "requestID": "e5ee36f1-edb0-4814-a4ba-2e8c97621c79",
            "eventID": "503cedb6-9906-4ee5-83e0-a64dde27bab0",
            "readOnly": true,
            "resources": [
                {
                    "accountId": "111122223333",
                    "type": "AWS::ServiceDiscovery::Namespace",
                    "ARN": "arn:aws:servicediscovery:eu-west-3:111122223333:namespace/
ns-vh4nbmhEXAMPLE"
                },
                {
                    "accountId": "111122223333",
                    "type": "AWS::ServiceDiscovery::Service",
                    "ARN": "arn:aws:servicediscovery:eu-west-3:111122223333:service/
srv-h46op6ylEXAMPLE"
                }
            ],
            "eventType": "AwsApiCall",
            "managementEvent": false,
            "recipientAccountId": "111122223333",
            "eventCategory": "Data",
            "tlsDetails": {
                "tlsVersion": "TLSv1.3",
                "cipherSuite": "TLS_AES_128_GCM_SHA256",
                "clientProvidedHostHeader": "data-servicediscovery.eu-
west-3.amazonaws.com"
            },
            "sessionCredentialFromConsole": "true"
        }
```

For information about CloudTrail record contents, see CloudTrail record contents in the *Amazon CloudTrail User Guide*.

Tagging your Amazon Cloud Map resources

To help you manage your Amazon Cloud Map resources, you can assign your own metadata to each resource in the form of *tags*. This topic describes tags and shows you how to create them.

Contents

Tag basics

A tag is a label that you assign to an Amazon resource. Each tag consists of a *key* and an optional *value*, both of which you define.

Tags enable you to categorize your Amazon resources by, for example, purpose, owner, or environment. When you have many resources of the same type, you can quickly identify a specific resource based on the tags you've assigned to it. For example, you can define a set of tags for your Amazon Cloud Map services to help you track each service's owner and stack level. We recommend that you devise a consistent set of tag keys for each resource type.

Tags are not automatically assigned to your resources. After you add a tag, you can edit tag keys and values or remove tags from a resource at any time. If you delete a resource, any tags for the resource are also deleted.

Tags don't have any semantic meaning to Amazon Cloud Map and are interpreted strictly as a string of characters. You can set the value of a tag to an empty string, but you can't set the value of a tag to null. If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value.

You can work with tags using the Amazon Web Services Management Console, the Amazon CLI, and the Amazon Cloud Map API.

If you're using Amazon Identity and Access Management (IAM), you can control which users in your Amazon account have permission to create, edit, or delete tags.

Tagging your resources

You can tag new or existing Amazon Cloud Map namespaces and services..

If you're using the Amazon Cloud Map console, you can apply tags to new resources when they are created or to existing resources at any time using the **Tags** tab on the relevant resource page.

If you're using the Amazon Cloud Map API, the Amazon CLI, or an Amazon SDK, you can apply tags to new resources using the `tags` parameter on the relevant API action or to existing resources using the TagResource API action. For more information, see TagResource.

Some resource-creating actions enable you to specify tags for a resource when the resource is created. If tags cannot be applied during resource creation, the resource creation process fails. This ensures that resources you intended to tag on creation are either created with specified tags or not created at all. If you tag resources at the time of creation, you don't need to run custom tagging scripts after resource creation.

The following table describes the Amazon Cloud Map resources that can be tagged, and the resources that can be tagged on creation.

Tagging support for Amazon Cloud Map resources

Resource	Supports tags	Supports tag propagation	Supports tagging on creation (Amazon Cloud Map API, Amazon CLI, Amazon SDK)
Amazon Cloud Map namespaces	Yes	No. Namespace tags don't propagate to any other resources associated with the namespace.	Yes
Amazon Cloud Map services	Yes	No. Service tags don't propagate to any other resources associated with the service.	Yes

Tag restrictions

The following basic restrictions apply to tags:

- Maximum number of tags for each resource – 50
- For each resource, each tag key must be unique, and each tag key can have only one value.
- Maximum key length – 128 Unicode characters in UTF-8
- Maximum value length – 256 Unicode characters in UTF-8
- If your tagging schema is used across multiple Amazon services and resources, remember that other services might have restrictions on allowed characters. Generally allowed characters are letters, numbers, spaces representable in UTF-8, and the following characters: + - = . _ : / @.
- Tag keys and values are case sensitive.
- Don't use `aws:`, `AWS:`, or any upper or lowercase combination of such as a prefix for either keys or values, as it is reserved for Amazon use. You can't edit or delete tag keys or values with this prefix. Tags with this prefix don't count against your tags-per-resource limit.

Working with tags using the CLI or API

Use the following Amazon CLI commands or Amazon Cloud Map API operations to add, update, list, and delete the tags for your resources.

Tagging support for Amazon Cloud Map resources

Task	API action	Amazon CLI	Amazon Tools for Windows PowerShell
Add or overwrite one or more tags.	TagResource	tag-resource	Add-SDResourceTag
Delete one or more tags.	UntagResource	untag-resource	Remove-SDResourceTag
List tags for a resource	ListTagsForResource	list-tags-for-resource	Get-SDResourceTag

The following examples show how to tag or untag resources using the Amazon CLI.

Example 1: Tag an existing resource

The following command tags an existing resource.

```
aws servicediscovery tag-resource --resource-arn resource_ARN --tags team=devs
```

Example 2: Untag an existing resource

The following command deletes a tag from an existing resource.

```
aws servicediscovery untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Example 3: List tags for a resource

The following command lists the tags associated with an existing resource.

```
aws servicediscovery list-tags-for-resource --resource-arn resource_ARN
```

Some resource-creating actions enable you to specify tags when you create the resource. The following actions support tagging on creation.

Task	API action	Amazon CLI	Amazon Tools for Windows PowerShell
Create an HTTP namespace	CreateHttpNamespace	create-http-namespace	New-SDHttpNamespace
Create a private namespace based on DNS	CreatePrivateDnsNamespace	create-private-dns-namespace	New-SDPrivateDnsNamespace
Create a public namespace based on DNS	CreatePublicDnsNamespace	create-public-dns-namespace	New-SDPublicDnsNamespace
Create a service	CreateService	create-service	New-SDService

Amazon Cloud Map service quotas

Amazon Cloud Map resources are subject to the following account-level service quotas. Each quota listed applies to each Amazon Region where you create Amazon Cloud Map resources.

Name	Default	Adjus e	Description
Custom attributes per instance	Each supported Region: 30	No	The maximum number of custom attributes that you can specify when you register an instance.
DiscoverInstances operation per account burst rate	Each supported Region: 2,000	Yes	The maximum burst rate to call DiscoverInstances operation from a single account.
DiscoverInstances operation per account steady rate	Each supported Region: 1,000	Yes	The maximum steady rate to call DiscoverI nstances operation from a single account.
DiscoverInstancesRevision operation per account rate	Each supported Region: 3,000	Yes	The maximum rate to call DiscoverInstancesR evision operation from a single account.
Instances per namespace	Each supported Region: 2,000	Yes	The maximum number of service instances that you can register using the same namespace.
Instances per service	Each supported Region: 1,000	No	The maximum number of instances that you can register in a Region using the same service.

Name	Default	Adjustable	Description
Namespaces per Region	Each supported Region: 50	Yes	The maximum number of namespaces that you can create per Region.

* When you create a namespace, we automatically create an Amazon Route 53 hosted zone. This hosted zone counts against the quota on the number of hosted zones that you can create with an Amazon account. For more information, see Quotas on hosted zones in the *Amazon Route 53 Developer Guide*.

** Increasing the instances for DNS namespaces for Amazon Cloud Map requires an increase to the records per hosted zone Route 53 limit, which incurs additional charges.

Managing your Amazon Cloud Map service quotas

You can request an increase for your Amazon Cloud Map quotas for your Amazon account. To request a quota adjustment, contact the Amazon Web Services Support Center.

Amazon Cloud Map DiscoverInstances API request throttling

Amazon Cloud Map throttles DiscoverInstances API requests for each Amazon account on a per-Region basis. Throttling helps improve the performance of the service and helps provide fair usage for all Amazon Cloud Map customers. Throttling ensures that calls to the Amazon Cloud Map DiscoverInstances API doesn't exceed the maximum allowed DiscoverInstances API request quotas. DiscoverInstances API calls originating from any of the following sources are subject to the request quotas:

- A third-party application
- A command line tool
- The Amazon Cloud Map console

If you exceed an API throttling quota, you get the `RequestLimitExceeded` error code. For more information, see the section called "Request rate limiting".

How throttling is applied

Amazon Cloud Map uses the token bucket algorithm to implement API throttling. With this algorithm, your account has a *bucket* that holds a specific number of *tokens*. The number of tokens in the bucket represents your throttling quota at any given second. There is one bucket for a single Region, and it applies to all endpoints in the Region.

Request rate limiting

Throttling limits the number of DiscoverInstances API requests that you can make. Each request removes one token from the bucket. For example, the bucket size for the DiscoverInstances API operation is 2,000 tokens, so you can make up to 2,000 DiscoverInstances requests in one second. If you exceed 2,000 requests in one second, you're throttled and the remaining requests within that second fail.

Buckets automatically refill at a set rate. If the bucket isn't at capacity, a set number of tokens is added back every second until the bucket reaches capacity. If the bucket is at capacity when refill tokens arrive, then these tokens are discarded. The bucket size for the DiscoverInstances API operation is 2,000 tokens, and the refill rate is 1,000 tokens every second. If you make 2,000 DiscoverInstances API requests in a second, the bucket is immediately reduced to zero (0) tokens. The bucket is then refilled by up to 1,000 tokens every second until it reaches its maximum capacity of 2,000 tokens.

You can use tokens as they are added to the bucket. You don't need to wait for the bucket to be at maximum capacity before you make API requests. If you deplete the bucket by making 2,000 DiscoverInstances API requests in one second, you can still make up to 1,000 DiscoverInstances API requests every second after that for as long as you need. This means that you can immediately use the refill tokens as they are added to your bucket. The bucket only starts to refill to the maximum capacity when you make fewer API requests every second than the refill rate.

Retries or batch processing

If an API request fails, your application might need to retry the request. To reduce the number of API requests, use an appropriate sleep interval between successive requests. For best results, use an increasing or variable sleep interval.

Calculating the sleep interval

When you have to poll or retry an API request, we recommend using an exponential backoff algorithm to calculate the sleep interval between API calls. By using progressively longer wait times between retries for consecutive error responses, you can reduce the number of failed requests. For more information and implementation examples of this algorithm, see Error Retries and Exponential Backoff in Amazon.

Adjusting API throttling quotas

You can request an increase to API throttling quotas for your Amazon account. To request a quota adjustment, contact the Amazon Web Services Support Center.

Related Information

The following related resources can help you as you work with Amazon Cloud Map.

Topics

- Amazon resources
- Third-Party Tools and Libraries

Amazon resources

The following related resources can help you as you work with this service.

- Getting Started Resource Center – Learn how to set up your Amazon Web Services account, join the Amazon community, and launch your first application.
- Amazon Web Services Support Center – The hub for creating and managing your Amazon Web Services Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and Amazon Trusted Advisor.
- Amazon Web Services Support – The primary webpage for information about Amazon Web Services Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.
- Contact Us – A central contact point for inquiries concerning Amazon billing, account, events, abuse, and other issues.
- Amazon Site Terms – Detailed information about our copyright and trademark; your account, license, and site access; and other topics.

Third-Party Tools and Libraries

In addition to Amazon resources, the following third-party tools and libraries work with Amazon Cloud Map.

- Cloud Application Framework (Amazon Cloud Map) – Library that handles common cloud platform tasks, such as queuing messages, publishing events, and calling cloud functions, with the help of Amazon Cloud Map.
- ExternalDNS for Kubernetes – Tool for configuring external DNS services including Amazon Route 53, and Amazon Cloud Map for Kubernetes Ingresses and Services.

Document history for Amazon Cloud Map

The following table describes the major updates and new features for the *Amazon Cloud Map Developer Guide*. We also update the documentation frequently to address the feedback that you send us.

Change	Description	Date
Tutorials added	Two tutorials showing common use cases for using Amazon Cloud Map added.	March 27, 2024
CloudTrail integration documentation updated	The documentation describin g the Amazon Cloud Map integration with CloudTrail to log API activity has been updated.	March 20, 2024
Managed policy updates	`AWSCloudMapDiscoverInstanceAccess` , `AWSCloudMapRegisterInstanceAccess` , and `AWSCloudMapReadOnlyAccess` policies were updated.	September 20, 2023
Cloud Map and Amazon PrivateLink	You can now use an Amazon PrivateLink to create a private connection between your VPC and Amazon Cloud Map.	September 15, 2023
Managed policy update	`AWSCloudMapDiscoverInstanceAccess` policy was updated.	August 15, 2023
Amazon SDK for Python	Added Python command line examples.	September 13, 2022

IPv6 support	API endpoints are now available in `IPv6`-only networks.	January 28, 2022
Service instance discovery	Amazon Cloud Map added support for creating services in a namespace that supports DNS queries that are discoverable only using the DiscoverInstances API operation and not using DNS queries.	March 24, 2021
Resource tagging	Amazon Cloud Map added support for adding metadata tags to your namespaces and services using the Amazon Web Services Management Console.	February 8, 2021
Resource tagging	Amazon Cloud Map added support for adding metadata tags to your namespaces and services using the Amazon CLI and APIs.	June 22, 2020
Initial Release	This is the first release of *Amazon Cloud Map Developer Guide*.	November 28, 2018

Amazon Glossary

For the latest Amazon terminology, see the Amazon glossary in the *Amazon Web Services Glossary Reference*.