# Amazon Elasticsearch Service

## Developer Guide

## API Version 2015-01-01

# Amazon Elasticsearch Service: Developer Guide

Copyright © 2020 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

# Table of Contents

# What Is Amazon Elasticsearch Service?

Amazon Elasticsearch Service (Amazon ES) is a managed service that makes it easy to deploy, operate, and scale Elasticsearch clusters in the AWS Cloud. Elasticsearch is a popular open-source search and analytics engine for use cases such as log analytics, real-time application monitoring, and clickstream analysis. With Amazon ES, you get direct access to the Elasticsearch APIs; existing code and applications work seamlessly with the service.

Amazon ES provisions all the resources for your Elasticsearch cluster and launches it. It also automatically detects and replaces failed Elasticsearch nodes, reducing the overhead associated with self-managed infrastructures. You can scale your cluster with a single API call or a few clicks in the console.

To get started using Amazon ES, you create a *domain*. An Amazon ES domain is synonymous with an Elasticsearch cluster. Domains are clusters with the settings, instance types, instance counts, and storage resources that you specify.

You can use the Amazon ES console to set up and configure a domain in minutes. If you prefer programmatic access, you can use the AWS CLI or the AWS SDKs.

**Topics**

## Features of Amazon Elasticsearch Service

Amazon ES includes the following features:

**Scale**

- Numerous configurations of CPU, memory, and storage capacity, known as *instance types*
- Up to 3 PB of attached storage
- Cost-effective UltraWarm (p. 160) storage for read-only data

**Security**

- AWS Identity and Access Management (IAM) access control
- Easy integration with Amazon VPC and VPC security groups
- Encryption of data at rest and node-to-node encryption
- Amazon Cognito or HTTP basic authentication for Kibana
- Index-level, document-level, and field-level security
- Kibana multi-tenancy

**Stability**

- Numerous geographical locations for your resources, known as *Regions* and *Availability Zones*
- Node allocation across two or three Availability Zones in the same AWS Region, known as *Multi-AZ*
- Dedicated master nodes to offload cluster management tasks
- Automated snapshots to back up and restore Amazon ES domains

**Flexibility**

- SQL support for integration with business intelligence (BI) applications
- Custom packages to improve search results

**Integration with Popular Services**

- Data visualization using Kibana
- Integration with Amazon CloudWatch for monitoring Amazon ES domain metrics and setting alarms
- Integration with AWS CloudTrail for auditing configuration API calls to Amazon ES domains
- Integration with Amazon S3, Amazon Kinesis, and Amazon DynamoDB for loading streaming data into Amazon ES
- Alerts from Amazon SNS when your data exceeds certain thresholds

## Supported Elasticsearch Versions

Amazon ES currently supports the following Elasticsearch versions:

- 7.4, 7.1
- 6.8, 6.7, 6.5, 6.4, 6.3, 6.2, 6.0
- 5.6, 5.5, 5.3, 5.1
- 2.3
- 1.5

Compared to earlier versions of Elasticsearch, the 7.*x* and 6.*x* versions offer powerful features that make them faster, more secure, and easier to use. Here are a few highlights:

- **Higher indexing performance** – Newer versions of Elasticsearch provide superior indexing capabilities that significantly increase the throughput of data updates.
- **Better safeguards** – Newer versions of Elasticsearch help prevent overly broad or complex queries from negatively affecting the performance and stability of the cluster.
- **Vega visualizations** – Kibana 6.2 and later versions support the Vega visualization language, which lets you make context-aware Elasticsearch queries, combine multiple data sources into a single graph, add user interactivity to graphs, and much more.
- **Java high-level REST client** – Compared to the low-level client, this client offers a simplified development experience and supports most Elasticsearch APIs. For a code example, see Signing HTTP Requests (p. 113).

For more information, see the section called "Supported Elasticsearch Operations" (p. 183), the section called "Features by Elasticsearch Version" (p. 181), and the section called "Plugins by Elasticsearch Version" (p. 182).

If you start a new Elasticsearch project, we strongly recommend that you choose the latest supported Elasticsearch version. If you have an existing domain that uses an older Elasticsearch version, you can choose to keep the domain or migrate your data. For more information, see the section called "Upgrading Elasticsearch" (p. 51).

## Pricing for Amazon Elasticsearch Service

For Amazon ES, you pay for each hour of use of an EC2 instance and for the cumulative size of any EBS storage volumes attached to your instances. Standard AWS data transfer charges also apply.

However, some notable data transfer exceptions exist. If a domain uses multiple Availability Zones (p. 17), Amazon ES does not bill for traffic between the Availability Zones. Significant data transfer occurs within a domain during shard allocation and rebalancing. Amazon ES neither meters nor bills for this traffic. Similarly, Amazon ES does not bill for data transfer between UltraWarm (p. 160) nodes and Amazon S3.

For full pricing details, see Amazon Elasticsearch Service Pricing. For information about charges incurred during configuration changes, see the section called "Charges for Configuration Changes" (p. 15).

## Getting Started with Amazon Elasticsearch Service

To get started, sign up for an AWS account if you don't already have one. After you are set up with an account, complete the Getting Started (p. 5) tutorial for Amazon Elasticsearch Service. Consult the following introductory topics if you need more information while learning about the service:

- Create a domain (p. 9)
- Size the domain (p. 171) appropriately for your workload
- Control access to your domain using a domain access policy (p. 64) or fine-grained access control (p. 76)
- Index data manually (p. 130) or from other AWS services (p. 132)
- Use Kibana (p. 156) to search your data and create visualizations

For information on migrating to Amazon ES from a self-managed Elasticsearch cluster, see the section called "Migrating to Amazon ES" (p. 207).

## Related Services

Amazon ES commonly is used with the following services:

Amazon CloudWatch

Amazon ES domains automatically send metrics to CloudWatch so that you can monitor domain health and performance. For more information, see Monitoring Cluster Metrics with Amazon CloudWatch (p. 28).

CloudWatch Logs can also go the other direction. You might configure CloudWatch Logs to stream data to Amazon ES for analysis. To learn more, see the section called "Loading Streaming Data into Amazon ES from Amazon CloudWatch" (p. 142).

AWS CloudTrail

Use AWS CloudTrail to get a history of the Amazon ES configuration API calls and related events for your account. For more information, see Logging and Monitoring in Amazon Elasticsearch Service (p. 95).

Amazon Kinesis

Kinesis is a managed service for real-time processing of streaming data at a massive scale. For more information, see the section called “Loading Streaming Data into Amazon ES from Amazon Kinesis Data Streams” (p. 137) and the section called “Loading Streaming Data into Amazon ES from Amazon Kinesis Data Firehose” (p. 142).

Amazon S3

Amazon Simple Storage Service (Amazon S3) provides storage for the internet. This guide provides Lambda sample code for integration with Amazon S3. For more information, see the section called “Loading Streaming Data into Amazon ES from Amazon S3” (p. 132).

AWS IAM

AWS Identity and Access Management (IAM) is a web service that you can use to manage access to your Amazon ES domains. For more information, see the section called “Identity and Access Management” (p. 64).

AWS Lambda

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. This guide provides Lambda sample code to stream data from DynamoDB, Amazon S3, and Kinesis. For more information, see the section called “Loading Streaming Data into Amazon ES” (p. 132).

Amazon DynamoDB

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. To learn more about streaming data to Amazon ES, see the section called “Loading Streaming Data into Amazon ES from Amazon DynamoDB” (p. 139).

# Getting Started with Amazon Elasticsearch Service

This tutorial shows you how to use Amazon Elasticsearch Service (Amazon ES) to create and configure a test domain. An Amazon ES domain is synonymous with an Elasticsearch cluster. Domains are clusters with the settings, instance types, instance counts, and storage resources that you specify.

The tutorial walks you through the basic steps to get a domain up and running quickly. For more detailed information, see *Creating and Managing Amazon ES Domains* (p. 9) and the other topics within this guide. For information on migrating to Amazon ES from a self-managed Elasticsearch cluster, see the section called “Migrating to Amazon ES” (p. 207).

You can complete the following steps by using the Amazon ES console, the AWS CLI, or the AWS SDK:

1. Create an Amazon ES domain (p. 5)
2. Upload data to an Amazon ES domain for indexing (p. 6)
3. Search documents in an Amazon ES domain (p. 7)
4. Delete an Amazon ES domain (p. 8)

For information about installing and setting up the AWS CLI, see the AWS Command Line Interface User Guide.

## Step 1: Create an Amazon ES Domain

**Important**
This process is a concise tutorial for configuring a *test domain*. It shouldn't be used to create production domains. For a comprehensive version of the same process, see *Creating and Managing Amazon ES Domains* (p. 9).

An Amazon ES domain is synonymous with an Elasticsearch cluster. Domains are clusters with the settings, instance types, instance counts, and storage resources that you specify. You can create an Amazon ES domain by using the console, the AWS CLI, or the AWS SDKs.

**To create an Amazon ES domain (console)**

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. Choose **Create a new domain**.
4. On the **Create Elasticsearch domain** page, choose **Development and testing**.
5. For **Elasticsearch version**, choose the latest version and **Next**.
6. Enter a name for the domain. In this tutorial, we use the domain name *movies* for the examples that we provide later in the tutorial.
7. For **Data nodes**, choose the `c5.large.elasticsearch` instance type. Use the default value of 1 instance.
8. For **Data nodes storage**, use the default values.
9. For now, you can ignore the **Dedicated master nodes**, **UltraWarm data nodes**, **Snapshot configuration**, and **Optional Elasticsearch cluster settings** sections.

10. Choose **Next**.
11. For simplicity in this tutorial, we recommend a public access domain. For **Network configuration**, choose **Public access**.
12. For **Fine-grained access control**, choose **Create master user**. Specify a username and password.
13. For now, you can ignore **Amazon Cognito Authentication**.
14. For **Access policy**, choose **Allow open access to the domain**. In this tutorial, fine-grained access control handles authentication, not the domain access policy.
15. Leave the encryption settings at their default values, and choose **Next**.
16. On the **Review** page, double-check your configuration and choose **Confirm**. New domains typically take 15-30 minutes to initialize, but can take longer depending on the configuration. After your domain initializes, make note of its endpoint.

## Step 2: Upload Data to an Amazon ES Domain for Indexing

**Important**
This process is a concise tutorial for uploading a small amount of test data. For more information, see *Indexing Data* (p. 130).

You can upload data to an Amazon Elasticsearch Service domain using the command line or most programming languages.

The following example requests use curl, a common HTTP client, for brevity and convenience. Clients like curl can't perform the request signing that is required if your access policies specify IAM users or roles. To successfully perform the instructions in this step, you must use fine-grained access control with a master user name and password, like you configured in step 1 (p. 5).

You can install curl on Windows and use it from the command prompt, but we recommend a tool like Cygwin or the Windows Subsystem for Linux. macOS and most Linux distributions come with curl pre-installed.

### To upload a single document to an Amazon ES domain

- Run the following command to add a single document to the *movies* domain:

```
curl -XPUT -u master-user:master-user-password domain-endpoint/movies/_doc/1 -d
 '{"director": "Burton, Tim", "genre": ["Comedy","Sci-Fi"], "year": 1996, "actor":
 ["Jack Nicholson","Pierce Brosnan","Sarah Jessica Parker"], "title": "Mars Attacks!"}'
 -H 'Content-Type: application/json'
```

For a detailed explanation of this command and how to make signed requests to Amazon ES, see *Indexing Data* (p. 130).

### To upload a JSON file that contains multiple documents to an Amazon ES domain

1. Create a file called `bulk_movies.json`. Copy and paste the following content into it, and add a trailing newline:

```
{ "index" : { "_index": "movies", "_id" : "2" } }
{"director": "Frankenheimer, John", "genre": ["Drama", "Mystery", "Thriller", "Crime"],
 "year": 1962, "actor": ["Lansbury, Angela", "Sinatra, Frank", "Leigh, Janet", "Harvey,
 Laurence", "Silva, Henry", "Frees, Paul", "Gregory, James", "Bissell, Whit", "McGiver,
 John", "Parrish, Leslie", "Edwards, James", "Flowers, Bess", "Dhiegh, Khigh", "Payne,
 Julie", "Kleeb, Helen", "Gray, Joe", "Nalder, Reggie", "Stevens, Bert", "Masters,
 Michael", "Lowell, Tom"], "title": "The Manchurian Candidate"}
{ "index" : { "_index": "movies", "_id" : "3" } }
{"director": "Baird, Stuart", "genre": ["Action", "Crime", "Thriller"], "year": 1998,
 "actor": ["Downey Jr., Robert", "Jones, Tommy Lee", "Snipes, Wesley", "Pantoliano,
 Joe", "Jacob, Ir\u00e8ne", "Nelligan, Kate", "Roebuck, Daniel", "Malahide, Patrick",
 "Richardson, LaTanya", "Wood, Tom", "Kosik, Thomas", "Stellate, Nick", "Minkoff,
 Robert", "Brown, Spitfire", "Foster, Reese", "Spielbauer, Bruce", "Mukherji, Kevin",
 "Cray, Ed", "Fordham, David", "Jett, Charlie"], "title": "U.S. Marshals"}
{ "index" : { "_index": "movies", "_id" : "4" } }
{"director": "Ray, Nicholas", "genre": ["Drama", "Romance"], "year": 1955, "actor":
 ["Hopper, Dennis", "Wood, Natalie", "Dean, James", "Mineo, Sal", "Backus, Jim",
 "Platt, Edward", "Ray, Nicholas", "Hopper, William", "Allen, Corey", "Birch, Paul",
 "Hudson, Rochelle", "Doran, Ann", "Hicks, Chuck", "Leigh, Nelson", "Williams, Robert",
 "Wessel, Dick", "Bryar, Paul", "Sessions, Almira", "McMahon, David", "Peters Jr.,
 House"], "title": "Rebel Without a Cause"}
```

2. Run the following command to upload the file to the *movies* domain:

```
curl -XPOST -u master-user:master-user-password domain-endpoint/_bulk --data-binary
 @bulk_movies.json -H 'Content-Type: application/json'
```

For more information about the bulk file format, see *Indexing Data* (p. 130).

## Step 3: Search Documents in an Amazon ES Domain

To search documents in an Amazon Elasticsearch Service domain, use the Elasticsearch search API. Or you can use Kibana (p. 156) to search documents in the domain.

**To search documents from the command line**

- Run the following command to search the *movies* domain for the word *mars*:

```
curl -XGET -u master-user:master-user-password 'domain-endpoint/movies/_search?
q=mars&pretty=true'
```

If you used the bulk data on the previous page, try searching for *rebel* instead.

**To search documents from an Amazon ES domain by using Kibana**

1. Point your browser to the Kibana plugin for your Amazon ES domain. You can find the Kibana endpoint on your domain dashboard on the Amazon ES console. The URL follows this format:

```
domain-endpoint/_plugin/kibana/
```

2. Log in using your master user name and password.
3. To use Kibana, you must configure at least one index pattern. Kibana uses these patterns to identify which indices you want to analyze. For this tutorial, enter *movies*, and then choose **Create**.
4. The **Index Patterns** page shows your various document fields, such as `actor` and `director`. For now, choose **Discover** to search your data.
5. In the search bar, enter *mars*, and then press **Enter**. Note how the similarity score (`_score`) increases when you search for the phrase *mars attacks*.

## Step 4: Delete an Amazon ES Domain

Because the *movies* domain from this tutorial is for test purposes, you should delete it when you are finished experimenting to avoid incurring charges.

**To delete an Amazon ES domain (console)**

1. Sign in to the **Amazon Elasticsearch Service** console.
2. In the navigation pane, under **My domains**, choose the *movies* domain.
3. Choose **Actions**, **Delete domain**.
4. Select the **Delete the domain** check box, and then choose **Delete**.

# Creating and Managing Amazon Elasticsearch Service Domains

This chapter describes how to create and manage Amazon Elasticsearch Service (Amazon ES) domains. An Amazon ES domain is synonymous with an Elasticsearch cluster. Domains are clusters with the settings, instance types, instance counts, and storage resources that you specify.

Unlike the brief instructions in the Getting Started (p. 5) tutorial, this chapter describes all options and provides relevant reference information. You can complete each procedure by using instructions for the Amazon ES console, the AWS Command Line Interface (AWS CLI), or the AWS SDKs.

## Creating Amazon ES Domains

This section describes how to create Amazon ES domains by using the Amazon ES console or by using the AWS CLI with the `create-elasticsearch-domain` command.

### Creating Amazon ES Domains (Console)

Use the following procedure to create an Amazon ES domain by using the console.

**To create an Amazon ES domain (console)**

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. Choose **Create a new domain**.
4. For **Choose deployment type**, choose the option that best matches the purpose of your domain:

   - **Production** domains use Multi-AZ and dedicated master nodes for higher availability.
   - **Development and testing** domains use a single Availability Zone.
   - **Custom** domains let you choose from all configuration options.

   **Important**
   Different deployment types present different options on subsequent pages. These steps include all options (the **Custom** deployment type).
5. For **Elasticsearch version**, we recommend that you choose the latest version. For more information, see the section called "Supported Elasticsearch Versions" (p. 2).
6. Choose **Next**.
7. For **Elasticsearch domain name**, enter a domain name. The name must meet the following criteria:

   - Unique to your account and Region
   - Starts with a lowercase letter
   - Contains between 3 and 28 characters
   - Contains only lowercase letters a-z, the numbers 0-9, and the hyphen (-)
8. For **Availability Zones**, choose **1-AZ**, **2-AZ**, or **3-AZ**. For more information, see the section called "Configuring a Multi-AZ Domain" (p. 17).
9. For **Instance type**, choose an instance type for the data nodes. For more information, see Supported Instance Types (p. 180).

> **Note**
> Not all Availability Zones support all instance types. If you choose **3-AZ**, we recommend choosing current-generation instance types such as R5 or I3.

10. For **Number of nodes**, choose the number of data nodes.

    For maximum values, see the section called "Cluster and Instance Limits" (p. 197). Single-node clusters are fine for development and testing, but should not be used for production workloads. For more guidance, see the section called "Sizing Amazon ES Domains" (p. 171) and the section called "Configuring a Multi-AZ Domain" (p. 17).
11. For **Data nodes storage type**, choose either **Instance** (default) or **EBS**.

    For guidance on creating especially large domains, see Petabyte Scale (p. 175). If you choose **EBS**, the following options appear:

    a. For **EBS volume type**, choose an EBS volume type.

       If you choose **Provisioned IOPS (SSD)** for the EBS volume type, for **Provisioned IOPS**, enter the baseline IOPS performance that you want. For more information, see Amazon EBS Volumes in the Amazon EC2 documentation.
    b. For **EBS storage size per node**, enter the size of the EBS volume that you want to attach to each data node.

       **EBS volume size** is per node. You can calculate the total cluster size for the Amazon ES domain by multiplying the number of data nodes by the EBS volume size. The minimum and maximum size of an EBS volume depends on both the specified EBS volume type and the instance type that it's attached to. To learn more, see EBS Volume Size Limits (p. 198).
12. (Optional) Enable or disable dedicated master nodes (p. 176). Dedicated master nodes increase cluster stability and are required for domains that have instance counts greater than 10. We recommend three dedicated master nodes for production domains.

    > **Note**
    > You can choose different instance types for your dedicated master nodes and data nodes. For example, you might select general purpose or storage-optimized instances for your data nodes, but compute-optimized instances for your dedicated master nodes.
13. (Optional) To enable UltraWarm storage (p. 160), choose **Enable UltraWarm data nodes**. Each instance type has a maximum amount of storage (p. 198) that it can address. Multiply that amount by the number of warm data nodes for the total addressable warm storage.
14. (Optional) For domains running Elasticsearch 5.3 and later, **Automated snapshot start hour** has no effect. For more information about automated snapshots, see the section called "Working with Index Snapshots" (p. 44).
15. (Optional) Choose **Optional Elasticsearch cluster settings**. For a summary of these options, see the section called "Advanced Options" (p. 13).
16. Choose **Next**.
17. In the **Network configuration** section, choose either **VPC access** or **Public access**. If you choose **Public access**, skip to the next step. If you choose **VPC access**, ensure that you have met the prerequisites (p. 25), and then do the following:

    a. For **VPC**, choose the ID of the VPC that you want to use.

       > **Note**
       > The VPC and domain must be in the same AWS Region, and you must select a VPC with tenancy set to **Default**. Amazon ES does not yet support VPCs that use dedicated tenancy.
    b. For **Subnet**, choose a subnet. If you enabled Multi-AZ, you must choose two or three subnets. Amazon ES will place a VPC endpoint and *elastic network interfaces* in the subnets.

> **Note**
> You must reserve sufficient IP addresses for the network interfaces in the subnet (or subnets). For more information, see Reserving IP Addresses in a VPC Subnet (p. 27).

c. For **Security groups**, choose the VPC security groups that need access to the Amazon ES domain. For more information, see the section called “VPC Support” (p. 21).

d. For **IAM role**, keep the default role. Amazon ES uses this predefined role (also known as a *service-linked role*) to access your VPC and to place a VPC endpoint and network interfaces in the subnet of the VPC. For more information, see Service-Linked Role for VPC Access (p. 27).

18. In the **Fine-grained access control** section, enable or disable fine-grained access control:

    - If you want to use IAM for user management, choose **Set IAM role as master user** and specify the ARN for an IAM role.
    - If you want to user the internal user database, choose **Create a master user** and specify a user name and password.

    Whichever option you choose, the master user can access all indices in the cluster and all Elasticsearch APIs. For guidance on which option to choose, see the section called “Key Concepts” (p. 81).

    If you disable fine-grained access control, you can still control access to your domain by placing it within a VPC, applying a restrictive access policy, or both. You must enable node-to-node encryption and encryption at rest to use fine-grained access control.
19. (Optional) If you want to use Amazon Cognito authentication for Kibana, choose **Enable Amazon Cognito authentication**.

    - Choose the Amazon Cognito user pool and identity pool that you want to use for Kibana authentication. For guidance on creating these resources, see the section called “Authentication for Kibana” (p. 99).
20. For **Domain access policy**, add the ARNs or IP addresses that you want or choose a preconfigured policy from the dropdown list. For more information, see the section called “Identity and Access Management” (p. 64) and the section called “About Access Policies on VPC Domains” (p. 24).

    > **Note**
    > If you chose **VPC access** in step 17, IP-based policies are prohibited. Instead, you can use security groups to control which IP addresses can access the domain. For more information, see the section called “About Access Policies on VPC Domains” (p. 24).
21. (Optional) To require that all requests to the domain arrive over HTTPS, select the **Require HTTPS for all traffic to the domain** check box.
22. (Optional) To enable node-to-node encryption, select the **Node-to-node encryption** check box. For more information, see the section called “Node-to-node Encryption” (p. 63).
23. (Optional) To enable encryption of data at rest, select the **Enable encryption of data at rest** check box.

    Select **(Default) aws/es** to have Amazon ES create a KMS encryption key on your behalf (or use the one that it already created). Otherwise, choose your own KMS encryption key from the **KMS master key** menu. For more information, see the section called “Encryption at Rest” (p. 61).
24. Choose **Next**.
25. On the **Review** page, review your domain configuration, and then choose **Confirm**.

## Creating Amazon ES Domains (AWS CLI)

Instead of creating an Amazon ES domain by using the console, you can use the AWS CLI. For syntax, see Amazon Elasticsearch Service in the AWS CLI Command Reference.

## Example Commands

This first example demonstrates the following Amazon ES domain configuration:

- Creates an Amazon ES domain named *mylogs* with Elasticsearch version 5.5
- Populates the domain with two instances of the `m4.large.elasticsearch` instance type
- Uses a 100 GiB Magnetic disk EBS volume for storage for each data node
- Allows anonymous access, but only from a single IP address: 192.0.2.0/32

```
aws es create-elasticsearch-domain --domain-name mylogs --elasticsearch-version 5.5 --
elasticsearch-cluster-config  InstanceType=m4.large.elasticsearch,InstanceCount=2 --ebs-
options EBSEnabled=true,VolumeType=standard,VolumeSize=100 --access-policies '{"Version":
 "2012-10-17", "Statement": [{"Action": "es:*", "Principal":"*","Effect": "Allow",
 "Condition": {"IpAddress":{"aws:SourceIp":["192.0.2.0/32"]}}}]}'
```

The next example demonstrates the following Amazon ES domain configuration:

- Creates an Amazon ES domain named *mylogs* with Elasticsearch version 5.5
- Populates the domain with six instances of the `m4.large.elasticsearch` instance type
- Uses a 100 GiB General Purpose (SSD) EBS volume for storage for each data node
- Restricts access to the service to a single user, identified by the user's AWS account ID: 555555555555
- Distributes instances across three Availability Zones

```
aws es create-elasticsearch-domain --domain-name mylogs --
elasticsearch-version 5.5 --elasticsearch-cluster-config
 InstanceType=m4.large.elasticsearch,InstanceCount=6,ZoneAwarenessEnabled=true,ZoneAwarenessConfig={Ava
 --ebs-options EBSEnabled=true,VolumeType=gp2,VolumeSize=100 --access-policies
 '{"Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": {"AWS":
 "arn:aws:iam::555555555555:root" }, "Action":"es:*", "Resource": "arn:aws:es:us-
east-1:555555555555:domain/mylogs/*" } ] }'
```

The next example demonstrates the following Amazon ES domain configuration:

- Creates an Amazon ES domain named *mylogs* with Elasticsearch version 5.5
- Populates the domain with ten instances of the `m4.xlarge.elasticsearch` instance type
- Populates the domain with three instances of the `m4.large.elasticsearch` instance type to serve as dedicated master nodes
- Uses a 100 GiB Provisioned IOPS EBS volume for storage, configured with a baseline performance of 1000 IOPS for each data node
- Restricts access to a single user and to a single subresource, the `_search` API

```
aws es create-elasticsearch-domain --domain-name mylogs --
elasticsearch-version 5.5 --elasticsearch-cluster-config
 InstanceType=m4.xlarge.elasticsearch,InstanceCount=10,DedicatedMasterEnabled=true,DedicatedMasterType=
 --ebs-options EBSEnabled=true,VolumeType=io1,VolumeSize=100,Iops=1000 --access-policies
 '{"Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "AWS":
 "arn:aws:iam::555555555555:root" }, "Action": "es:*", "Resource": "arn:aws:es:us-
east-1:555555555555:domain/mylogs/_search" } ] }'
```

**Note**
If you attempt to create an Amazon ES domain and a domain with the same name already exists, the CLI does not report an error. Instead, it returns details for the existing domain.

## Creating Amazon ES Domains (AWS SDKs)

The AWS SDKs (except the Android and iOS SDKs) support all the actions defined in the Amazon ES Configuration API Reference (p. 236), including `CreateElasticsearchDomain`. For sample code, see the section called "Using the AWS SDKs" (p. 121). For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

# Configuring Access Policies

Amazon Elasticsearch Service offers several ways to configure access to your Amazon ES domains. For more information, see the section called "Identity and Access Management" (p. 64) and the section called "Fine-Grained Access Control" (p. 76).

The console provides preconfigured access policies that you can customize for the specific needs of your domain. You also can import access policies from other Amazon ES domains. For information about how these access policies interact with VPC access, see the section called "About Access Policies on VPC Domains" (p. 24).

**To configure access policies (console)**

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. In the navigation pane, under **My domains**, choose the domain that you want to update.
4. Choose **Actions** and **Modify access policy**.
5. Edit the access policy JSON, or use the dropdown list to choose a preconfigured option.
6. Choose **Submit**.

# Advanced Options

Use advanced options to configure the following:

**rest.action.multi.allow_explicit_index**

Specifies whether explicit references to indices are allowed inside the body of HTTP requests. Setting this property to `false` prevents users from bypassing access control for subresources. By default, the value is `true`. For more information, see the section called "Advanced Options and API Considerations" (p. 73).

**indices.fielddata.cache.size**

Specifies the percentage of Java heap space that is allocated to field data. By default, this setting is unbounded.

> **Note**
> Many customers query rotating daily indices. We recommend that you begin benchmark testing with `indices.fielddata.cache.size` configured to 40% of the JVM heap for most such use cases. However, if you have very large indices you might need a large field data cache.

**indices.query.bool.max_clause_count**

Specifies the maximum number of clauses allowed in a Lucene boolean query. The default is 1,024. Queries with more than the permitted number of clauses result in a `TooManyClauses` error. For more information, see the Lucene documentation.

## Configuration Changes

Amazon ES uses a *blue/green* deployment process when updating domains. Blue/green typically refers to the practice of running two production environments, one live and one idle, and switching the two as you make software changes. In the case of Amazon ES, it refers to the practice of creating a new environment for domain updates and routing users to the new environment after those updates are complete. The practice minimizes downtime and maintains the original environment in the event that deployment to the new environment is unsuccessful.

The following operations cause blue/green deployments:

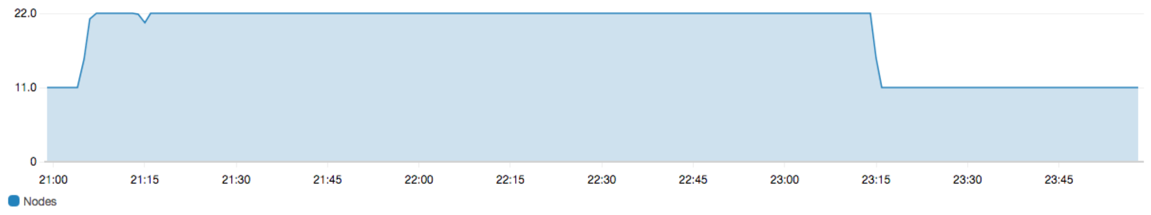
- Changing instance type
- If your domain *doesn't* have dedicated master nodes, changing data instance count
- Enabling or disabling dedicated master nodes
- Changing dedicated master node count
- Enabling or disabling Multi-AZ
- Changing storage type, volume type, or volume size
- Choosing different VPC subnets
- Adding or removing VPC security groups
- Enabling or disabling Amazon Cognito authentication for Kibana
- Choosing a different Amazon Cognito user pool or identity pool
- Modifying advanced settings
- Enabling or disabling the publication of error logs or slow logs to CloudWatch
- Upgrading to a new Elasticsearch version
- Enabling or disabling **Require HTTPS**
- Enabling UltraWarm storage

In *most* cases, the following operations do not cause blue/green deployments:

- Changing access policy
- Changing the automated snapshot hour
- If your domain has dedicated master nodes, changing data node or UltraWarm node count

There are some exceptions. For example, if you haven't reconfigured your domain since the launch of three Availability Zone support, Amazon ES might perform a one-time blue/green deployment to redistribute your dedicated master nodes across Availability Zones.

If you initiate a configuration change, the domain state changes to **Processing** while Amazon ES creates a new environment with the latest service software (p. 15). During certain service software updates, the state remains **Active**. In both cases, you can review the cluster health and Amazon CloudWatch metrics and see that the number of nodes in the cluster temporarily increases—often doubling—while the domain update occurs. In the following illustration, you can see the number of nodes doubling from 11 to 22 during a configuration change and returning to 11 when the update is complete.



This temporary increase can strain the cluster's dedicated master nodes (p. 176), which suddenly might have many more nodes to manage. It's important to maintain sufficient capacity on dedicated master nodes to handle the overhead that is associated with these blue/green deployments.

**Important**
You do *not* incur any additional charges during configuration changes and service maintenance. You are billed only for the number of nodes that you request for your cluster. For specifics, see the section called "Charges for Configuration Changes" (p. 15).

To prevent overloading dedicated master nodes, you can monitor usage with the Amazon CloudWatch metrics (p. 28). For recommended maximum values, see the section called "Recommended CloudWatch Alarms" (p. 178).

## Charges for Configuration Changes

If you change the configuration for a domain, Amazon ES creates a new cluster as described in the section called "Configuration Changes" (p. 14). During the migration of old to new, you incur the following charges:

- If you change the instance type, you are charged for both clusters for the first hour. After the first hour, you are charged only for the new cluster.

  **Example:** You change the configuration from three `m3.xlarge` instances to four `m4.large` instances. For the first hour, you are charged for both clusters (3 * `m3.xlarge` + 4 * `m4.large`). After the first hour, you are charged only for the new cluster (4 * `m4.large`).
- If you don't change the instance type, you are charged only for the largest cluster for the first hour. After the first hour, you are charged only for the new cluster.

  **Example:** You change the configuration from six `m3.xlarge` instances to three `m3.xlarge` instances. For the first hour, you are charged for the largest cluster (6 * `m3.xlarge`). After the first hour, you are charged only for the new cluster (3 * `m3.xlarge`).

# Service Software Updates

**Note**
Service software updates differ from Elasticsearch version upgrades. For information about upgrading to a later version of Elasticsearch, see the section called "Upgrading Elasticsearch" (p. 51).

Amazon ES regularly releases system software updates that add features or otherwise improve your domains. The console is the easiest way to see if an update is available. When new service software becomes available, you can request an update to your domain and benefit from new features more quickly. You might also want to start the update at a low traffic time.

Some updates are required. Others are optional.

- If you take no action on required updates, we still update the service software automatically after a certain timeframe (typically two weeks).
- If the console does not include an automatic deployment date, the update is optional.

Your domain might be ineligible for a service software update if it is in any of the states that are shown in the following table.

| State | Description |
|---|---|
| Domain in processing | The domain is in the middle of a configuration change. Check update eligibility after the operation completes. |
| Red cluster status | One or more indices in the cluster is red. For troubleshooting steps, see the section called "Red Cluster Status" (p. 228). |
| High error rate | The Elasticsearch cluster is returning a large number of 5*xx* errors when attempting to process requests. This problem is usually the result of too many simultaneous read or write requests. Consider reducing traffic to the cluster or scaling your domain. |
| Split brain | *Split brain* means that your Elasticsearch cluster has more than one master node and has split into two clusters that never will rejoin on their own. You can avoid split brain by using the recommended number of dedicated master nodes (p. 176). For help recovering from split brain, contact AWS Support. |
| Amazon Cognito integration issue | Your domain uses authentication for Kibana (p. 99), and Amazon ES can't find one or more Amazon Cognito resources. This problem usually occurs if the Amazon Cognito user pool is missing. To correct the issue, recreate the missing resource and configure the Amazon ES domain to use it. |
| Other Amazon ES service issue | Issues with Amazon ES itself might cause your domain to display as ineligible for an update. If none of the previous conditions apply to your domain and the problem persists for more than a day, contact AWS Support. |

### To request a service software update (console)

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. In the navigation pane, under **My domains**, choose the domain that you want to update.
4. For **Service software release**, use the documentation link to compare your current version to the latest version. Then choose **Update**.

### To request a service software update (AWS CLI and AWS SDKs)

You can use the following commands to see if an update is available, check upgrade eligibility, and request an update:

- `describe-elasticsearch-domain` (`DescribeElasticsearchDomain`)
- `start-elasticsearch-service-software-update` (`StartElasticsearchServiceSoftwareUpdate`)

For more information, see the AWS CLI Command Reference and *Amazon ES Configuration API Reference* (p. 236).

**Tip**
After requesting an update, you might have a narrow window of time in which you can cancel it. Use the console or `stop-elasticsearch-service-software-update` (`StopElasticsearchServiceSoftwareUpdate`) command.

# Configuring a Multi-AZ Domain

Each AWS Region is a separate geographic area with multiple, isolated locations known as *Availability Zones*. To prevent data loss and minimize cluster downtime in the event of a service disruption, you can distribute nodes across two or three Availability Zones in the same Region, a configuration known as *Multi-AZ*.

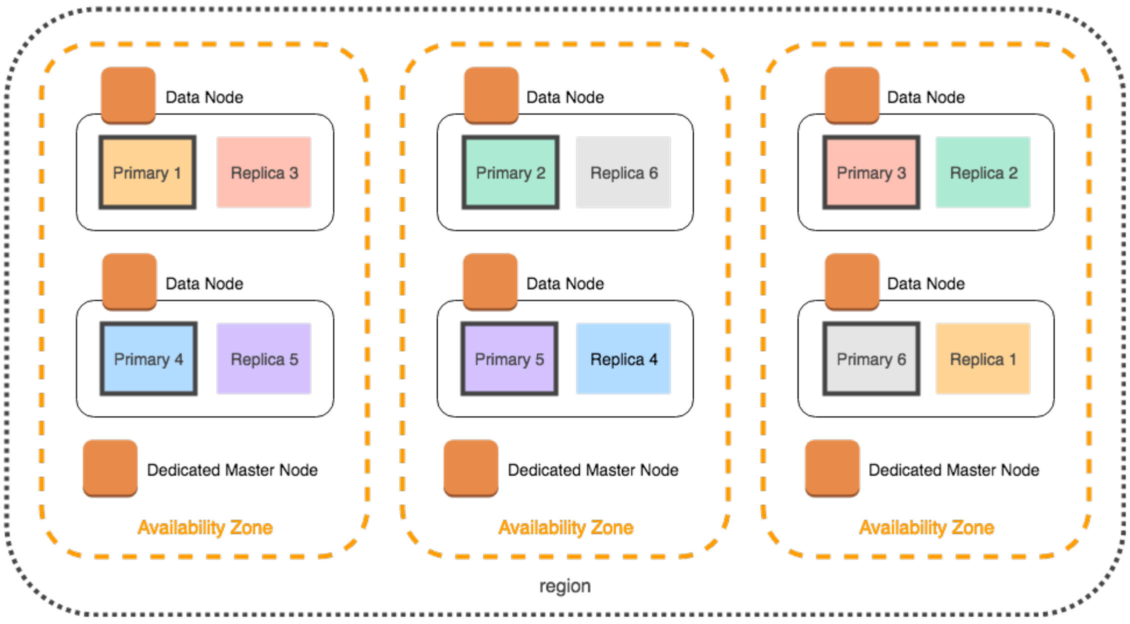
For domains that run production workloads, we recommend the following configuration:

- Choose a Region that supports three Availability Zones with Amazon ES:
  - US East (N. Virginia, Ohio)
  - US West (Oregon)
  - EU (Frankfurt, Ireland, London, Paris, Milan)
  - Asia Pacific (Singapore, Sydney, Tokyo, Hong Kong, Mumbai)
  - Middle East (Bahrain)
  - Africa (Cape Town)
  - China (Ningxia)
- Deploy the domain across three zones.
- Choose current-generation instance types for dedicated master nodes and data nodes.
- Use three dedicated master nodes and at least three data nodes.
- Create at least one replica for each index in your cluster.

The rest of this section provides explanations for and context around these recommendations.

## Shard Distribution

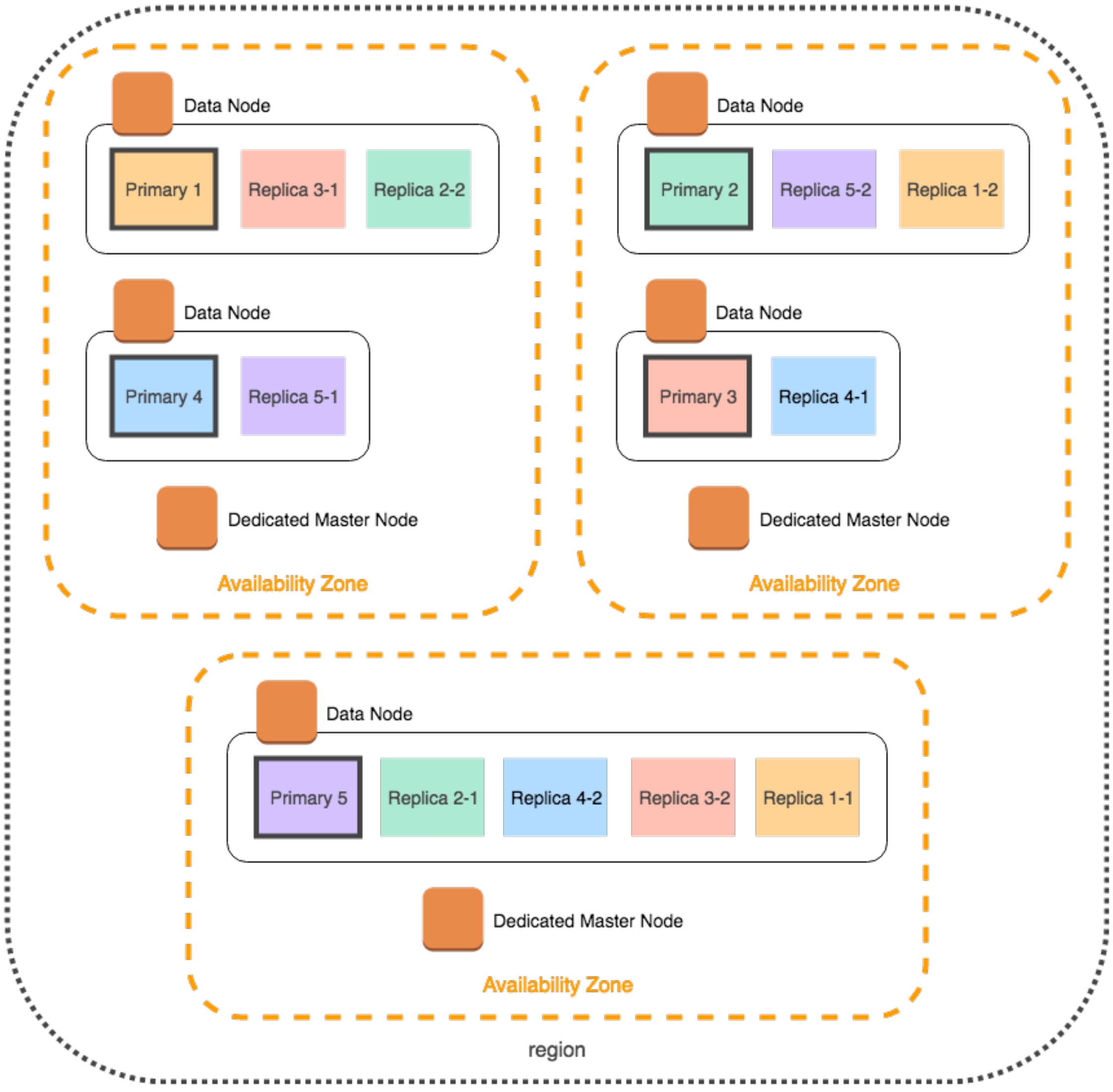
If you enable Multi-AZ, you should create at least one replica for each index in your cluster. Without replicas, Amazon ES can't distribute copies of your data to other Availability Zones, which largely defeats the purpose of Multi-AZ. Fortunately, the default configuration for any index is a replica count of 1. As the following diagram shows, Amazon ES makes a best effort to distribute primary shards and their corresponding replica shards to different zones.



In addition to distributing shards by Availability Zone, Amazon ES distributes them by node. Still, certain domain configurations can result in imbalanced shard counts. Consider the following domain:

- 5 data nodes
- 5 primary shards
- 2 replicas
- 3 Availability Zones

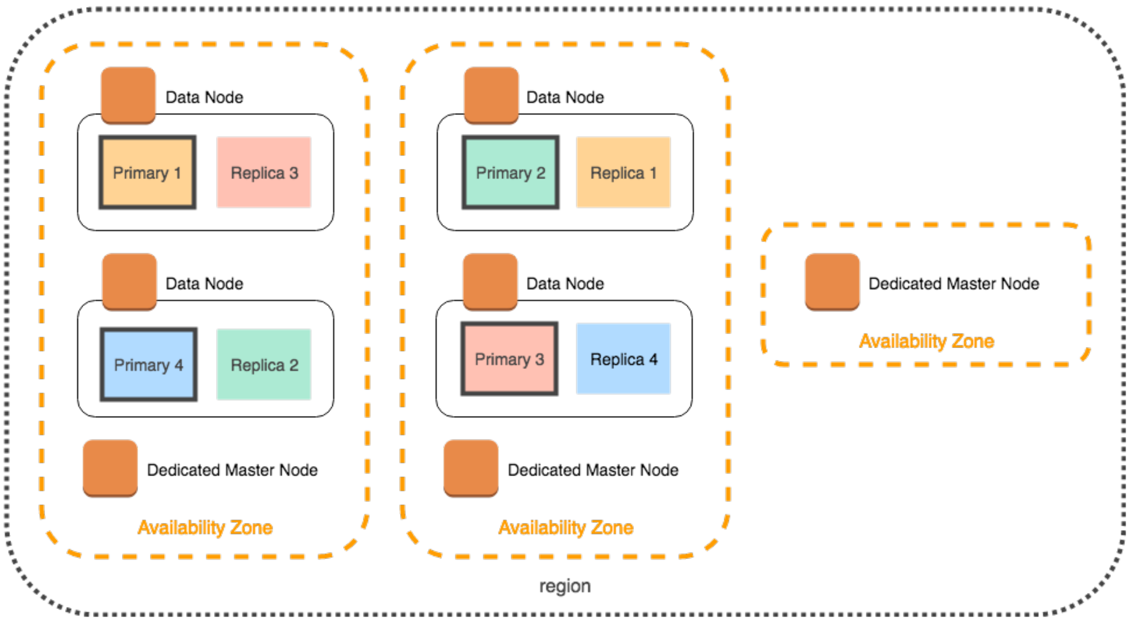
In this situation, Amazon ES has to overload one node in order to distribute the primary and replica shards across the zones, as shown in the following diagram.



To avoid these kinds of situations, which can strain individual nodes and hurt performance, we recommend that you choose an instance count that is a multiple of three if you plan to have two or more replicas per index.

## Dedicated Master Node Distribution

Even if you select two Availability Zones when configuring your domain, Amazon ES automatically distributes dedicated master nodes (p. 176) across three Availability Zones. This distribution helps prevent cluster downtime if a zone experiences a service disruption. If you use the recommended three dedicated master nodes and one Availability Zone goes down, your cluster still has a quorum (2) of dedicated master nodes and can elect a new master. The following diagram demonstrates this configuration.



This automatic distribution has some notable exceptions:

- If you choose an older-generation instance type that is not available in three Availability Zones, the following scenarios apply:
  - If you chose three Availability Zones for the domain, Amazon ES throws an error. Choose a different instance type, and try again.
  - If you chose two Availability Zones for the domain, Amazon ES distributes the dedicated master nodes across two zones.
- Not all AWS Regions have three Availability Zones. In these Regions, you can only configure a domain to use two zones (and Amazon ES can only distribute dedicated master nodes across two zones). For the list of Regions that support three Availability Zones, see the section called “Configuring a Multi-AZ Domain” (p. 17).

## Availability Zone Disruptions

Availability Zone disruptions are rare, but do occur. The following table lists different Multi-AZ configurations and behaviors during a disruption.

| Number of Availability Zones in a Region | Number of Availability Zones That You Chose | Number of Dedicated Master Nodes | Behavior if One Availability Zone Experiences a Disruption |
|---|---|---|---|
| 2 or more | 2 | 0 | Downtime. Your cluster loses half of its data nodes and must replace at least one in the remaining Availability Zone before it can elect a master. |
| 2 | 2 | 3 | 50/50 chance of downtime. Amazon ES distributes two dedicated master nodes into one Availability Zone and one into the other: |

| Number of Availability Zones in a Region | Number of Availability Zones That You Chose | Number of Dedicated Master Nodes | Behavior if One Availability Zone Experiences a Disruption |
|---|---|---|---|
| | | | • If the Availability Zone with one dedicated master node experiences a disruption, the two dedicated master nodes in the remaining Availability Zone can elect a master.<br>• If the Availability Zone with two dedicated master nodes experiences a disruption, the cluster is unavailable until the remaining Availability Zone can add a dedicated master node and elect a master. |
| 3 or more | 2 | 3 | No downtime. Amazon ES automatically distributes the dedicated master nodes across three Availability Zones, so the remaining two dedicated master nodes can elect a master. |
| 3 or more | 3 | 0 | No downtime. Roughly two-thirds of your data nodes are still available to elect a master. |
| 3 or more | 3 | 3 | No downtime. The remaining two dedicated master nodes can elect a master. |

In *all* configurations, regardless of the cause, node failures can cause the cluster's remaining data nodes to experience a period of increased load while Amazon ES automatically configures new nodes to replace the now-missing ones.

For example, in the event of an Availability Zone disruption in a three-zone configuration, two-thirds as many data nodes have to process just as many requests to the cluster. As they process these requests, the remaining nodes are also replicating shards onto new nodes as they come online, which can further impact performance. If availability is critical to your workload, consider adding resources to your cluster to alleviate this concern.

**Note**
Amazon ES manages Multi-AZ domains transparently, so you can't manually simulate Availability Zone disruptions.

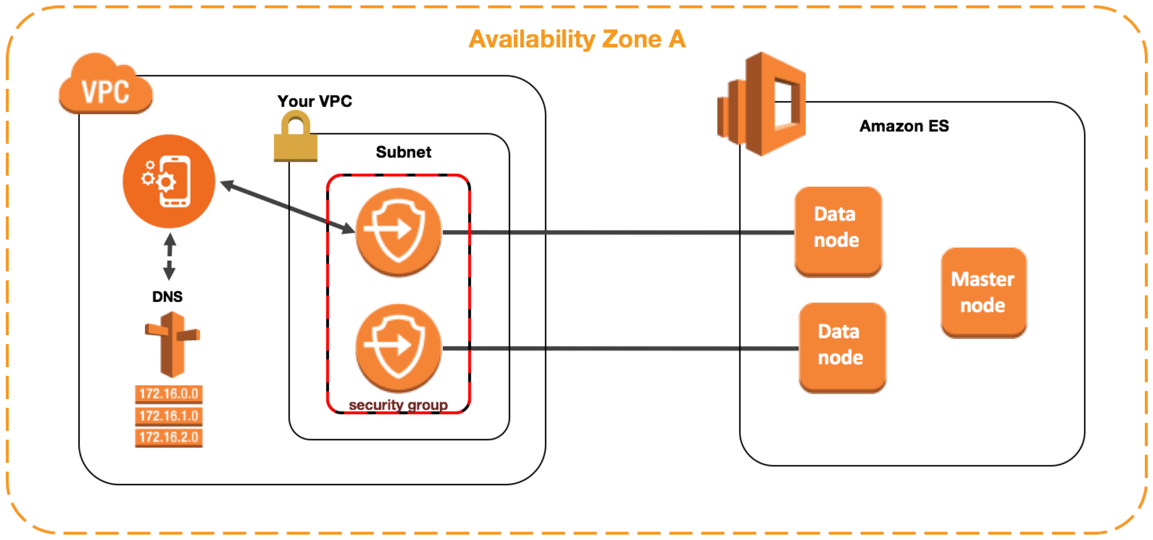
## VPC Support for Amazon Elasticsearch Service Domains

A *virtual private cloud* (VPC) is a virtual network that is dedicated to your AWS account. It's logically isolated from other virtual networks in the AWS Cloud. You can launch AWS resources, such as Amazon ES domains, into your VPC.

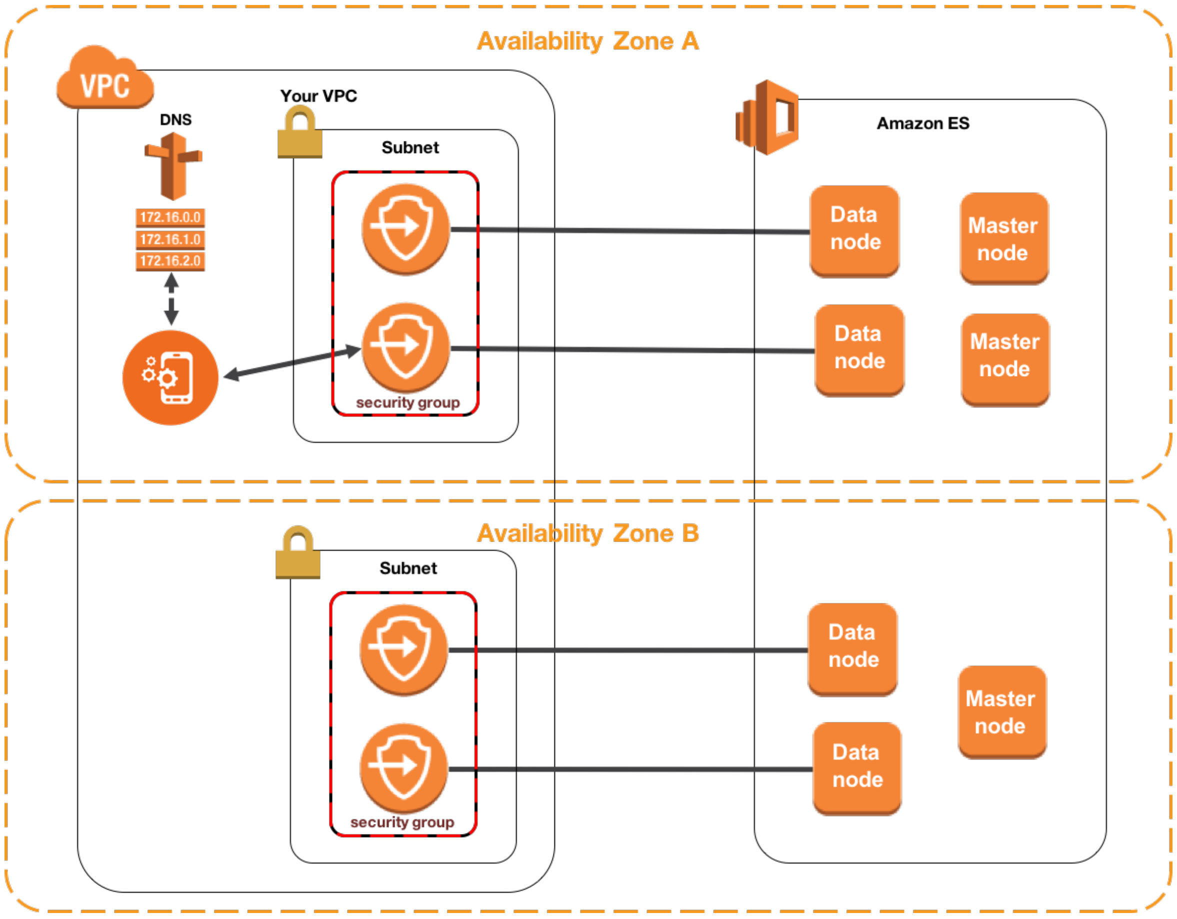
Placing an Amazon ES domain within a VPC enables secure communication between Amazon ES and other services within the VPC without the need for an internet gateway, NAT device, or VPN connection. All traffic remains securely within the AWS Cloud. Because of their logical isolation, domains that reside within a VPC have an extra layer of security when compared to domains that use public endpoints.

To support VPCs, Amazon ES places an endpoint into one, two, or three subnets of your VPC. A *subnet* is a range of IP addresses in your VPC. If you enable multiple Availability Zones (p. 17) for your domain, each subnet must be in a different Availability Zone in the same region. If you only use one Availability Zone, Amazon ES places an endpoint into only one subnet.

The following illustration shows the VPC architecture for one Availability Zone.



The following illustration shows the VPC architecture for two Availability Zones.



Amazon ES also places an *elastic network interface* (ENI) in the VPC for each of your data nodes. Amazon ES assigns each ENI a private IP address from the IPv4 address range of your subnet. The service also assigns a public DNS hostname (which is the domain endpoint) for the IP addresses. You must use a public DNS service to resolve the endpoint (which is a DNS hostname) to the appropriate IP addresses for the data nodes:

- If your VPC uses the Amazon-provided DNS server by setting the `enableDnsSupport` option to `true` (the default value), resolution for the Amazon ES endpoint will succeed.
- If your VPC uses a private DNS server and the server can reach the public authoritative DNS servers to resolve DNS hostnames, resolution for the Amazon ES endpoint will also succeed.

Because the IP addresses might change, you should resolve the domain endpoint periodically so that you can always access the correct data nodes. We recommend that you set the DNS resolution interval to one minute. If you're using a client, you should also ensure that the DNS cache in the client is cleared.

**Note**
Amazon ES doesn't support IPv6 addresses with a VPC. You can use a VPC that has IPv6 enabled, but the domain will use IPv4 addresses.

**Topics**

## Limitations

Currently, operating an Amazon ES domain within a VPC has the following limitations:

- You can either launch your domain within a VPC or use a public endpoint, but you can't do both. You must choose one or the other when you create your domain.
- If you launch a new domain within a VPC, you can't later switch it to use a public endpoint. The reverse is also true: If you create a domain with a public endpoint, you can't later place it within a VPC. Instead, you must create a new domain and migrate your data.
- You can't launch your domain within a VPC that uses dedicated tenancy. You must use a VPC with tenancy set to **Default**.
- After you place a domain within a VPC, you can't move it to a different VPC. However, you can change the subnets and security group settings.
- Compared to public domains, VPC domains display less information in the Amazon ES console. Specifically, the **Cluster health** tab does not include shard information, and the **Indices** tab is not present at all.
- To access the default installation of Kibana for a domain that resides within a VPC, users must have access to the VPC. This process varies by network configuration, but likely involves connecting to a VPN or managed network or using a proxy server. To learn more, see the section called "About Access Policies on VPC Domains" (p. 24), the Amazon VPC User Guide, and the section called "Controlling Access to Kibana" (p. 156).

## About Access Policies on VPC Domains

Placing your Amazon ES domain within a VPC provides an inherent, strong layer of security. When you create a domain with public access, the endpoint takes the following form:

```
https://search-domain-name-identifier.region.es.amazonaws.com
```

As the "public" label suggests, this endpoint is accessible from any internet-connected device, though you can (and should) control access to it (p. 64). If you access the endpoint in a web browser, you might receive a `Not Authorized` message, but the request reaches the domain.

When you create a domain with VPC access, the endpoint *looks* similar to a public endpoint:

```
https://vpc-domain-name-identifier.region.es.amazonaws.com
```

If you try to access the endpoint in a web browser, however, you might find that the request times out. To perform even basic `GET` requests, your computer must be able to connect to the VPC. This connection often takes the form of a VPN, managed network, or proxy server. For details on the various forms it can take, see Scenarios and Examples in the *Amazon VPC User Guide*. For a development-focused example, see the section called "Testing VPC Domains" (p. 24).

In addition to this connectivity requirement, VPCs let you manage access to the domain through security groups. For many use cases, this combination of security features is sufficient, and you might feel comfortable applying an open access policy to the domain.

Operating with an open access policy does *not* mean that anyone on the internet can access the Amazon ES domain. Rather, it means that if a request reaches the Amazon ES domain and the associated security groups permit it, the domain accepts the request without further security checks.

For an additional layer of security, we recommend using access policies that specify IAM users or roles. Applying these policies means that, for the domain to accept a request, the security groups must permit it *and* it must be signed with valid credentials.

**Note**
Because security groups already enforce IP-based access policies, you can't apply IP-based access policies to Amazon ES domains that reside within a VPC. If you use public access, IP-based policies are still available.

## Testing VPC Domains

The enhanced security of a VPC can make connecting to your domain and running basic tests a real challenge. If you already have an Amazon ES VPC domain and would rather not create a VPN server, try the following process:

1. For your domain's access policy, choose **Allow open access to the domain**. You can always update this setting after you finish testing.
2. Create an Amazon Linux Amazon EC2 instance in the same VPC, subnet, and security group as your Amazon ES domain.

   Because this instance is for testing purposes and needs to do very little work, choose an inexpensive instance type like `t2.micro`. Assign the instance a public IP address and either create a new key pair or choose an existing one. If you create a new key, download it to your `~/.ssh` directory.

   To learn more about creating instances, see Getting Started with Amazon EC2 Linux Instances.
3. Add an internet gateway to your VPC.

4. In the route table for your VPC, add a new route. For **Destination**, specify a CIDR block that contains your computer's public IP address. For **Target**, specify the internet gateway you just created.

   For example, you might specify `123.123.123.123/32` for just your computer or `123.123.123.0/24` for a range of computers.
5. For the security group, specify two inbound rules:

   | Type | Protocol | Port Range | Source |
   |---|---|---|---|
   | SSH (22) | TCP (6) | 22 | *your-cidr-block* |
   | HTTPS (443) | TCP (6) | 443 | *your-security-group-id* |

   The first rule lets you SSH into your EC2 instance. The second allows the EC2 instance to communicate with the Amazon ES domain over HTTPS.
6. From the terminal, run the following command:

   ```
   ssh -i ~/.ssh/your-key.pem ec2-user@your-ec2-instance-public-ip -N -L 9200:vpc-your-
   amazon-es-domain.region.es.amazonaws.com:443
   ```

   This command creates an SSH tunnel that forwards requests to https://localhost:9200 to your Amazon ES domain through the EC2 instance. By default, Elasticsearch listens for traffic on port 9200. Specifying this port simulates a local Elasticsearch install, but use whichever port you'd like.

   The command provides no feedback and runs indefinitely. To stop it, press `Ctrl + C`.
7. Navigate to https://localhost:9200/_plugin/kibana/ in your web browser. You might need to acknowledge a security exception.

   Alternately, you can send requests to https://localhost:9200 using curl, Postman, or your favorite programming language.

   **Tip**
   If you encounter curl errors due to a certificate mismatch, try the `--insecure` flag.

As an alternative to this approach, if your domain is in a region that AWS Cloud9 supports, you can create an EC2 environment in the same VPC as your domain, add the environment's security group to your Amazon ES domain configuration, add the HTTPS rule from step 5 to your security group, and use the web-based Bash in AWS Cloud9 to issue curl commands.

## Before You Begin: Prerequisites for VPC Access

Before you can enable a connection between a VPC and your new Amazon ES domain, you must do the following:

- **Create a VPC**

  To create your VPC, you can use the Amazon VPC console, the AWS CLI, or one of the AWS SDKs. For more information, see Creating A VPC (p. 26). If you already have a VPC, you can skip this step.
- **Reserve IP addresses**

  Amazon ES enables the connection of a VPC to a domain by placing network interfaces in a subnet of the VPC. Each network interface is associated with an IP address. You must reserve a sufficient number of IP addresses in the subnet for the network interfaces. For more information, see Reserving IP Addresses in a VPC Subnet (p. 27).

## Creating a VPC

To create your VPC, you can use one of the following: the Amazon VPC console, the AWS CLI, or one of the AWS SDKs. The VPC must have between one and three subnets, depending on the number of Availability Zones (p. 17) for your domain.

The following procedure shows how to use the Amazon VPC console to create a VPC with a public subnet, reserve IP addresses for the subnet, and create a security group to control access to your Amazon ES domain. For other VPC configurations, see Scenarios and Examples in the *Amazon VPC User Guide*.

### To create a VPC (console)

1. Sign in to the AWS Management Console, and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **VPC Dashboard**.
3. Choose **Start VPC Wizard**.
4. On the **Select a VPC Configuration** page, select **VPC with a Single Public Subnet**.
5. On the **VPC with a Single Public Subnet** page, keep the default options, and then choose **Create VPC**.
6. In the confirmation message that appears, choose **Close**.
7. If you intend to enable multiple Availability Zones (p. 17) for your Amazon ES domain, you must create additional subnets. Otherwise, skip to step 8.

   a. In the navigation pane, choose **Subnets.**
   b. Choose **Create Subnet**.
   c. In the **Create Subnet** dialog box, optionally create a name tag to help you identify the subnet later.
   d. For **VPC**, choose the VPC that you just created.
   e. For **Availability Zone**, choose an Availability Zone that differs from that of the first subnet. The Availability Zones for all subnets must be in the same region.
   f. For **IPv4 CIDR block**, configure a CIDR block large enough to provide sufficient IP addresses for Amazon ES to use during maintenance activities. For more information, see Reserving IP Addresses in a VPC Subnet (p. 27).

      **Note**
      Amazon ES domains using VPC access don't support IPv6 addresses. You can use a VPC that has IPv6 enabled, but the ENIs will have IPv4 addresses.
   g. Choose **Yes, Create**.
8. In the navigation pane, choose **Subnets**.
9. In the list of subnets, find your subnet (or subnets, if you created a second subnet in step 7). In the **Available IPv4** column, confirm that you have a sufficient number of IPv4 addresses.
10. Make a note of the subnet ID and Availability Zone. You need this information later when you launch your Amazon ES domain and add an Amazon EC2 instance to your VPC.
11. Create an Amazon VPC security group. You use this security group to control access to your Amazon ES domain.

    a. In the navigation pane, choose **Security Groups**.
    b. Choose **Create Security Group**.
    c. In the **Create Security Group** dialog box, type a name tag, a group name, and a description. For **VPC**, choose the ID of your VPC.
    d. Choose **Yes, Create**.

12. Define a network ingress rule for your security group. This rule allows you to connect to your Amazon ES domain.

    a. In the navigation pane, choose **Security Groups**, and then select the security group that you just created.
    b. At the bottom of the page, choose the **Inbound Rules** tab.
    c. Choose **Edit**, and then choose **HTTPS (443)**.
    d. Choose **Save**.

Now you are ready to launch an Amazon ES domain (p. 9) in your Amazon VPC.

## Reserving IP Addresses in a VPC Subnet

Amazon ES connects a domain to a VPC by placing network interfaces in a subnet of the VPC (or multiple subnets of the VPC if you enable multiple Availability Zones (p. 17)). Each network interface is associated with an IP address. Before you create your Amazon ES domain, you must have a sufficient number of IP addresses available in the VPC subnet to accommodate the network interfaces.

The number of IP addresses that Amazon ES requires depends on the following:

- Number of data nodes in your domain. (Master nodes are not included in the number.)
- Number of Availability Zones. If you enable two or three Availability Zones, you need only half or one-third the number of IP addresses per subnet that you need for one Availability Zone.

Here is the basic formula: The number of IP addresses reserved in each subnet is three times the number of nodes, divided by the number of Availability Zones.

**Examples**

- If a domain has 10 data nodes and two Availability Zones, the IP count is 10 / 2 * 3 = 15.
- If a domain has 10 data nodes and one Availability Zone, the IP count is 10 * 3 = 30.

When you create the domain, Amazon ES reserves the IP addresses, uses some for the domain, and reserves the rest for blue/green deployments (p. 14). You can see the network interfaces and their associated IP addresses in the **Network Interfaces** section of the Amazon EC2 console at https://console.aws.amazon.com/ec2/. The **Description** column shows which Amazon ES domain the network interface is associated with.

**Tip**
We recommend that you create dedicated subnets for the Amazon ES reserved IP addresses. By using dedicated subnets, you avoid overlap with other applications and services and ensure that you can reserve additional IP addresses if you need to scale your cluster in the future. To learn more, see Creating a Subnet in Your VPC.

## Service-Linked Role for VPC Access

A service-linked role is a unique type of IAM role that delegates permissions to a service so that it can create and manage resources on your behalf. Amazon ES requires a service-linked role to access your VPC, create the domain endpoint, and place network interfaces in a subnet of your VPC.

Amazon ES automatically creates the role when you use the Amazon ES console to create a domain within a VPC. For this automatic creation to succeed, you must have permissions for the `es:CreateElasticsearchServiceRole` and `iam:CreateServiceLinkedRole` actions. To learn more, see Service-Linked Role Permissions in the *IAM User Guide*.

After Amazon ES creates the role, you can view it (`AWSServiceRoleForAmazonElasticsearchService`) using the IAM console.

**Note**
If you create a domain that uses a public endpoint, Amazon ES doesn't need the service-linked role and doesn't create it.

For full information on this role's permissions and how to delete it, see the section called "Using Service-Linked Roles" (p. 110).

## Migrating from Public Access to VPC Access

When you create a domain, you specify whether it should have a public endpoint or reside within a VPC. Once created, you cannot switch from one to the other. Instead, you must create a new domain and either manually reindex or migrate your data. Snapshots offer a convenient means of migrating data. For information about taking and restoring snapshots, see the section called "Working with Index Snapshots" (p. 44).

## Amazon VPC Documentation

Amazon VPC has its own set of documentation to describe how to create and use your Amazon VPC. The following table provides links to the Amazon VPC guides.

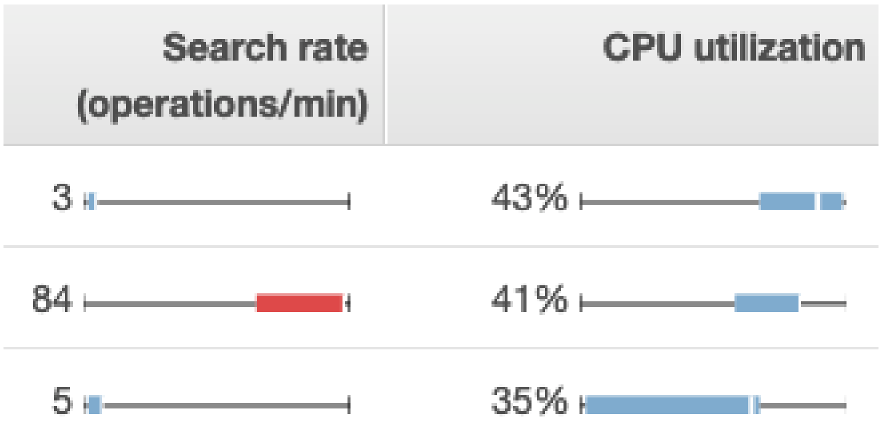
| Description | Documentation |
|---|---|
| How to get started using Amazon VPC | Amazon VPC Getting Started Guide |
| How to use Amazon VPC through the AWS Management Console | Amazon VPC User Guide |
| Complete descriptions of all the Amazon VPC commands | Amazon EC2 Command Line Reference (The Amazon VPC commands are part of the Amazon EC2 reference.) |
| Complete descriptions of the Amazon VPC API actions, data types, and errors | Amazon EC2 API Reference (The Amazon VPC API actions are part of the Amazon EC2 reference.) |
| Information for the network administrator who configures the gateway at your end of an optional IPsec VPN connection | AWS Site-to-Site VPN Network Administrator Guide |

For more detailed information about Amazon Virtual Private Cloud, see Amazon Virtual Private Cloud.

# Monitoring Cluster Metrics with Amazon CloudWatch

## Interpreting Health Dashboards

The **Instance health** tab in the Amazon ES console uses box charts to provide at-a-glance visibility into the health of each Elasticsearch node.



- Each colored box shows the range of values for the node over the specified time period.
- Blue boxes represent values that are consistent with other nodes. Red boxes represent outliers.
- The white line within each box shows the node's current value.
- The “whiskers” on either side of each box show the minimum and maximum values for all nodes over the time period.

Amazon ES domains send performance metrics to Amazon CloudWatch every minute. If you use General Purpose or Magnetic EBS volumes, the EBS volume metrics update only every five minutes. To view these metrics, use the **Cluster health** and **Instance health** tabs in the Amazon Elasticsearch Service console. The metrics are provided at no extra charge.

If you make configuration changes to your domain, the list of individual instances in the **Cluster health** and **Instance health** tabs often double in size for a brief period before returning to the correct number. For an explanation of this behavior, see the section called “Configuration Changes” (p. 14).

Amazon ES metrics fall into these categories:

- the section called “Cluster Metrics” (p. 29)
- the section called “Dedicated Master Node Metrics” (p. 32)
- the section called “EBS Volume Metrics” (p. 33)
- the section called “Instance Metrics” (p. 34)
- the section called “UltraWarm Metrics” (p. 37)
- the section called “Alerting Metrics” (p. 38)
- the section called “SQL Metrics” (p. 39)

**Note**
The service archives metrics for two weeks before discarding them.

## Cluster Metrics

The `AWS/ES` namespace includes the following metrics for clusters.

| Metric | Description |
|---|---|
| `ClusterStatus.green` | A value of 1 indicates that all index shards are allocated to nodes in the cluster. |

| Metric | Description |
|---|---|
| | Relevant statistics: Maximum |
| `ClusterStatus.yellow` | A value of 1 indicates that the primary shards for all indices are allocated to nodes in the cluster, but replica shards for at least one index are not. For more information, see the section called "Yellow Cluster Status" (p. 230).<br><br>Relevant statistics: Maximum |
| `ClusterStatus.red` | A value of 1 indicates that the primary and replica shards for at least one index are not allocated to nodes in the cluster. For more information, see the section called "Red Cluster Status" (p. 228).<br><br>Relevant statistics: Maximum |
| `Nodes` | The number of nodes in the Amazon ES cluster, including dedicated master nodes and UltraWarm nodes. For more information, see the section called "Configuration Changes" (p. 14).<br><br>Relevant statistics: Maximum |
| `SearchableDocuments` | The total number of searchable documents across all data nodes in the cluster.<br><br>Relevant statistics: Minimum, Maximum, Average |
| `DeletedDocuments` | The total number of documents marked for deletion across all data nodes in the cluster. These documents no longer appear in search results, but Elasticsearch only removes deleted documents from disk during segment merges. This metric increases after delete requests and decreases after segment merges.<br><br>Relevant statistics: Minimum, Maximum, Average |
| `CPUUtilization` | The percentage of CPU usage for data nodes in the cluster. Maximum shows the node with the highest CPU usage. Average represents all nodes in the cluster. This metric is also available for individual nodes.<br><br>Relevant statistics: Maximum, Average |

| Metric | Description |
|---|---|
| `FreeStorageSpace` | The free space for data nodes in the cluster. `Sum` shows total free space for the cluster, but you must leave the period at one minute to get an accurate value. `Minimum` and `Maximum` show the nodes with the most and least free space, respectively. This metric is also available for individual nodes. Amazon ES throws a `ClusterBlockException` when this metric reaches `0`. To recover, you must either delete indices, add larger instances, or add EBS-based storage to existing instances. To learn more, see the section called "Lack of Available Storage Space" (p. 230).<br><br>The Amazon ES console displays this value in GiB. The Amazon CloudWatch console displays it in MiB.<br><br>**Note**<br>`FreeStorageSpace` will always be lower than the value that the Elasticsearch `_cluster/stats` API provides. Amazon ES reserves a percentage of the storage space on each instance for internal operations.<br><br>Relevant statistics: Minimum, Maximum, Average, Sum |
| `ClusterUsedSpace` | The total used space for the cluster. You must leave the period at one minute to get an accurate value.<br><br>The Amazon ES console displays this value in GiB. The Amazon CloudWatch console displays it in MiB.<br><br>Relevant statistics: Minimum, Maximum |
| `ClusterIndexWritesBlocked` | Indicates whether your cluster is accepting or blocking incoming write requests. A value of 0 means that the cluster is accepting requests. A value of 1 means that it is blocking requests.<br><br>Many factors can cause a cluster to begin blocking requests. Some common factors include the following: `FreeStorageSpace` is too low, `JVMMemoryPressure` is too high, or `CPUUtilization` is too high. To alleviate this issue, consider adding more disk space or scaling your cluster.<br><br>Relevant statistics: Maximum |
| `JVMMemoryPressure` | The maximum percentage of the Java heap used for all data nodes in the cluster. Amazon ES uses half of an instance's RAM for the Java heap, up to a heap size of 32 GiB. You can scale instances vertically up to 64 GiB of RAM, at which point you can scale horizontally by adding instances. See the section called "Recommended CloudWatch Alarms" (p. 178).<br><br>Relevant statistics: Maximum |
| `AutomatedSnapshotFailure` | The number of failed automated snapshots for the cluster. A value of `1` indicates that no automated snapshot was taken for the domain in the previous 36 hours.<br><br>Relevant statistics: Minimum, Maximum |

| Metric | Description |
|---|---|
| `CPUCreditBalance` | The remaining CPU credits available for data nodes in the cluster. A CPU credit provides the performance of a full CPU core for one minute. For more information, see CPU Credits in the *Amazon EC2 Developer Guide*. This metric is available only for the `t2.micro.elasticsearch`, `t2.small.elasticsearch`, and `t2.medium.elasticsearch` instance types.<br><br>Relevant statistics: Minimum |
| `KibanaHealthyNodes` | A health check for Kibana. A value of 1 indicates normal behavior. A value of 0 indicates that Kibana is inaccessible. In most cases, the health of Kibana mirrors the health of the cluster.<br><br>Relevant statistics: Minimum |
| `KMSKeyError` | A value of 1 indicates that the KMS customer master key used to encrypt data at rest has been disabled. To restore the domain to normal operations, re-enable the key. The console displays this metric only for domains that encrypt data at rest.<br><br>Relevant statistics: Minimum, Maximum |
| `KMSKeyInaccessible` | A value of 1 indicates that the KMS customer master key used to encrypt data at rest has been deleted or revoked its grants to Amazon ES. You can't recover domains that are in this state. If you have a manual snapshot, though, you can use it to migrate the domain's data to a new domain. The console displays this metric only for domains that encrypt data at rest.<br><br>Relevant statistics: Minimum, Maximum |
| `InvalidHostHeaderRequests` | The number of HTTP requests made to the Elasticsearch cluster that included an invalid (or missing) host header. Valid requests include the domain hostname as the host header value. Amazon ES rejects invalid requests for public access domains that don't have a restrictive access policy. We recommend applying a restrictive access policy to all domains.<br><br>If you see large values for this metric, confirm that your Elasticsearch clients include the domain hostname (and not, for example, its IP address) in their requests.<br><br>Relevant statistics: Sum |
| `ElasticsearchRequests` | The number of requests made to the Elasticsearch cluster.<br><br>Relevant statistics: Sum |
| `2xx, 3xx, 4xx, 5xx` | The number of requests to the domain that resulted in the given HTTP response code (2*xx*, 3*xx*, 4*xx*, 5*xx*).<br><br>Relevant statistics: Sum |

## Dedicated Master Node Metrics

The `AWS/ES` namespace includes the following metrics for dedicated master nodes (p. 176).

| Metric | Description |
|---|---|
| `MasterCPUUtilization` | The maximum percentage of CPU resources used by the dedicated master nodes. We recommend increasing the size of the instance type when this metric reaches 60 percent.<br><br>Relevant statistics: Average |
| `MasterFreeStorageSpace` | This metric is not relevant and can be ignored. The service does not use master nodes as data nodes. |
| `MasterJVMMemoryPressure` | The maximum percentage of the Java heap used for all dedicated master nodes in the cluster. We recommend moving to a larger instance type when this metric reaches 85 percent.<br><br>Relevant statistics: Maximum |
| `MasterCPUCreditBalance` | The remaining CPU credits available for dedicated master nodes in the cluster. A CPU credit provides the performance of a full CPU core for one minute. For more information, see CPU Credits in the *Amazon EC2 User Guide for Linux Instances*. This metric is available only for the `t2.micro.elasticsearch`, `t2.small.elasticsearch`, and `t2.medium.elasticsearch` instance types.<br><br>Relevant statistics: Minimum |
| `MasterReachableFromNode` | A health check for `MasterNotDiscovered` exceptions. A value of 1 indicates normal behavior. A value of 0 indicates that `/_cluster/health/` is failing.<br><br>Failures mean that the master node stopped or is not reachable. They are usually the result of a network connectivity issue or AWS dependency problem.<br><br>Relevant statistics: Minimum |
| `MasterSysMemoryUtilization` | The percentage of the master node's memory that is in use.<br><br>Relevant statistics: Maximum |

## EBS Volume Metrics

The `AWS/ES` namespace includes the following metrics for EBS volumes.

| Metric | Description |
|---|---|
| `ReadLatency` | The latency, in seconds, for read operations on EBS volumes.<br><br>Relevant statistics: Minimum, Maximum, Average |
| `WriteLatency` | The latency, in seconds, for write operations on EBS volumes.<br><br>Relevant statistics: Minimum, Maximum, Average |
| `ReadThroughput` | The throughput, in bytes per second, for read operations on EBS volumes.<br><br>Relevant statistics: Minimum, Maximum, Average |

| Metric | Description |
|---|---|
| `WriteThroughput` | The throughput, in bytes per second, for write operations on EBS volumes.<br><br>Relevant statistics: Minimum, Maximum, Average |
| `DiskQueueDepth` | The number of pending input and output (I/O) requests for an EBS volume.<br><br>Relevant statistics: Minimum, Maximum, Average |
| `ReadIOPS` | The number of input and output (I/O) operations per second for read operations on EBS volumes.<br><br>Relevant statistics: Minimum, Maximum, Average |
| `WriteIOPS` | The number of input and output (I/O) operations per second for write operations on EBS volumes.<br><br>Relevant statistics: Minimum, Maximum, Average |

## Instance Metrics

The `AWS/ES` namespace includes the following metrics for each instance in a domain. Amazon ES also aggregates these instance metrics to provide insight into overall cluster health. You can verify this behavior using the **Data samples** statistic in the console. Note that each metric in the following table has relevant statistics for the node *and* the cluster.

**Important**
Different versions of Elasticsearch use different thread pools to process calls to the `_index` API. Elasticsearch 1.5 and 2.3 use the index thread pool. Elasticsearch 5.*x*, 6.0, and 6.2 use the bulk thread pool. 6.3 and later use the write thread pool. Currently, the Amazon ES console doesn't include a graph for the bulk thread pool.

| Metric | Description |
|---|---|
| `IndexingLatency` | The average time, in milliseconds, that it takes a shard to complete an indexing operation.<br><br>Relevant node statistics: Average<br><br>Relevant cluster statistics: Average, Maximum |
| `IndexingRate` | The number of indexing operations per minute. A single call to the `_bulk` API that adds two documents and updates two counts as four operations, which might be spread across one or more nodes. If that index has one or more replicas, other nodes in the cluster also record a total of four indexing operations. Document deletions do not count towards this metric.<br><br>Relevant node statistics: Average<br><br>Relevant cluster statistics: Average, Maximum, Sum |
| `SearchLatency` | The average time, in milliseconds, that it takes a shard on a data node to complete a search operation.<br><br>Relevant node statistics: Average<br><br>Relevant cluster statistics: Average, Maximum |

| Metric | Description |
|---|---|
| `SearchRate` | The total number of search requests per minute for all shards on a data node. A single call to the `_search` API might return results from many different shards. If five of these shards are on one node, the node would report 5 for this metric, even though the client only made one request.<br><br>Relevant node statistics: Average<br><br>Relevant cluster statistics: Average, Maximum, Sum |
| `SysMemoryUtilization` | The percentage of the instance's memory that is in use. High values for this metric are normal and usually do not represent a problem with your cluster. For a better indicator of potential performance and stability issues, see the `JVMMemoryPressure` metric.<br><br>Relevant node statistics: Minimum, Maximum, Average<br><br>Relevant cluster statistics: Minimum, Maximum, Average |
| `JVMGCYoungCollectionCount` | The number of times that "young generation" garbage collection has run. A large, ever-growing number of runs is a normal part of cluster operations.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Sum, Maximum, Average |
| `JVMGCYoungCollectionTime` | The amount of time, in milliseconds, that the cluster has spent performing "young generation" garbage collection.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Sum, Maximum, Average |
| `JVMGCOldCollectionCount` | The number of times that "old generation" garbage collection has run. In a cluster with sufficient resources, this number should remain small and grow infrequently.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Sum, Maximum, Average |
| `JVMGCOldCollectionTime` | The amount of time, in milliseconds, that the cluster has spent performing "old generation" garbage collection.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Sum, Maximum, Average |
| `ThreadpoolForce_mergeQueue` | The number of queued tasks in the force merge thread pool. If the queue size is consistently high, consider scaling your cluster.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Sum, Maximum, Average |

| Metric | Description |
|---|---|
| `ThreadpoolForce_mergeRejected` | The number of rejected tasks in the force merge thread pool. If this number continually grows, consider scaling your cluster.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Sum |
| `ThreadpoolForce_mergeThreads` | The size of the force merge thread pool.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Average, Sum |
| `ThreadpoolIndexQueue` | The number of queued tasks in the index thread pool. If the queue size is consistently high, consider scaling your cluster. The maximum index queue size is 200.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Sum, Maximum, Average |
| `ThreadpoolIndexRejected` | The number of rejected tasks in the index thread pool. If this number continually grows, consider scaling your cluster.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Sum |
| `ThreadpoolIndexThreads` | The size of the index thread pool.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Average, Sum |
| `ThreadpoolSearchQueue` | The number of queued tasks in the search thread pool. If the queue size is consistently high, consider scaling your cluster. The maximum search queue size is 1,000.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Sum, Maximum, Average |
| `ThreadpoolSearchRejected` | The number of rejected tasks in the search thread pool. If this number continually grows, consider scaling your cluster.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Sum |
| `ThreadpoolSearchThreads` | The size of the search thread pool.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Average, Sum |

| Metric | Description |
|---|---|
| `ThreadpoolBulkQueue` | The number of queued tasks in the bulk thread pool. If the queue size is consistently high, consider scaling your cluster.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Sum, Maximum, Average |
| `ThreadpoolBulkRejected` | The number of rejected tasks in the bulk thread pool. If this number continually grows, consider scaling your cluster.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Sum |
| `ThreadpoolBulkThreads` | The size of the bulk thread pool.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Average, Sum |
| `ThreadpoolWriteThreads` | The size of the write thread pool.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Average, Sum |
| `ThreadpoolWriteRejected` | The number of rejected tasks in the write thread pool.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Average, Sum |
| `ThreadpoolWriteQueue` | The number of queued tasks in the write thread pool.<br><br>Relevant node statistics: Maximum<br><br>Relevant cluster statistics: Average, Sum |

## UltraWarm Metrics

The `AWS/ES` namespace includes the following metrics for UltraWarm (p. 160) nodes.

| Metric | Description |
|---|---|
| `WarmCPUUtilization` | The percentage of CPU usage for UltraWarm nodes in the cluster. Maximum shows the node with the highest CPU usage. Average represents all UltraWarm nodes in the cluster. This metric is also available for individual UltraWarm nodes.<br><br>Relevant statistics: Maximum, Average |
| `WarmFreeStorageSpace` | The amount of free warm storage space in MiB. Because UltraWarm uses Amazon S3 rather than attached disks, `Sum` is the only relevant statistic. You must leave the period at one minute to get an accurate value.<br><br>Relevant statistics: Sum |

| Metric | Description |
|---|---|
| `WarmJVMMemoryPressure` | The maximum percentage of the Java heap used for the UltraWarm nodes.<br><br>Relevant statistics: Maximum |
| `WarmSearchableDocuments` | The total number of searchable documents across all warm indices in the cluster. You must leave the period at one minute to get an accurate value.<br><br>Relevant statistics: Sum |
| `WarmSearchLatency` | The average time, in milliseconds, that it takes a shard on an UltraWarm node to complete a search operation.<br><br>Relevant node statistics: Average<br><br>Relevant cluster statistics: Average, Maximum |
| `WarmSearchRate` | The total number of search requests per minute for all shards on an UltraWarm node. A single call to the `_search` API might return results from many different shards. If five of these shards are on one node, the node would report 5 for this metric, even though the client only made one request.<br><br>Relevant node statistics: Average<br><br>Relevant cluster statistics: Average, Maximum, Sum |
| `WarmStorageSpaceUtilization` | The total amount of warm storage space that the cluster is using. The Amazon ES console displays this value in GiB. The Amazon CloudWatch console displays it in MiB.<br><br>Relevant statistics: Max |
| `HotStorageSpaceUtilization` | The total amount of hot storage space that the cluster is using. The Amazon ES console displays this value in GiB. The Amazon CloudWatch console displays it in MiB.<br><br>Relevant statistics: Max |
| `WarmSysMemoryUtilization` | The percentage of the warm node's memory that is in use.<br><br>Relevant statistics: Maximum |
| `HotToWarmMigrationQueueSize` | The number of indices currently migrating from hot to warm storage.<br><br>Relevant statistics: Maximum |
| `WarmToHotMigrationQueueSize` | The number of indices currently migrating from warm to hot storage.<br><br>Relevant statistics: Maximum |

## Alerting Metrics

The `AWS/ES` namespace includes the following metrics for the alerting feature (p. 169).

| Metric | Description |
|---|---|
| `AlertingDegraded` | A value of 1 means that either the alerting index is red or one or more nodes is not on schedule. A value of 0 indicates normal behavior.<br><br>Relevant statistics: Maximum |
| `AlertingIndexExists` | A value of 1 means the `.opendistro-alerting-config` index exists. A value of 0 means it does not. Until you use the alerting feature for the first time, this value remains 0.<br><br>Relevant statistics: Maximum |
| `AlertingIndexStatus.green` | The health of the index. A value of 1 means green. A value of 0 means that the index either doesn't exist or isn't green.<br><br>Relevant statistics: Maximum |
| `AlertingIndexStatus.red` | The health of the index. A value of 1 means red. A value of 0 means that the index either doesn't exist or isn't red.<br><br>Relevant statistics: Maximum |
| `AlertingIndexStatus.yellow` | The health of the index. A value of 1 means yellow. A value of 0 means that the index either doesn't exist or isn't yellow.<br><br>Relevant statistics: Maximum |
| `AlertingNodesNotOnSchedule` | A value of 1 means some jobs are not running on schedule. A value of 0 means that all alerting jobs are running on schedule (or that no alerting jobs exist). Check the Amazon ES console or make a `_nodes/stats` request to see if any nodes show high resource usage.<br><br>Relevant statistics: Maximum |
| `AlertingNodesOnSchedule` | A value of 1 means that all alerting jobs are running on schedule (or that no alerting jobs exist). A value of 0 means some jobs are not running on schedule.<br><br>Relevant statistics: Maximum |
| `AlertingScheduledJobEnabled` | A value of 1 means that the `opendistro.scheduled_jobs.enabled` cluster setting is true. A value of 0 means it is false, and scheduled jobs are disabled.<br><br>Relevant statistics: Maximum |

## SQL Metrics

The `AWS/ES` namespace includes the following metrics for SQL support (p. 153).

| Metric | Description |
|---|---|
| `SQLFailedRequestCountByCusErr` | The number of requests to the `_opendistro/_sql` API that failed due to a client issue. For example, a request might return HTTP status code 400 due to an `IndexNotFoundException`. |

| Metric | Description |
|---|---|
| | Relevant statistics: Sum |
| `SQLFailedRequestCountBySysErr` | The number of requests to the `_opendistro/_sql` API that failed due to a server problem or feature limitation. For example, a request might return HTTP status code 503 due to a `VerificationException`.<br><br>Relevant statistics: Sum |
| `SQLRequestCount` | The number of requests to the `_opendistro/_sql` API.<br><br>Relevant statistics: Sum |
| `SQLUnhealthy` | A value of 1 indicates that, in response to certain requests, the SQL plugin is returning 5*xx* response codes or passing invalid query DSL to Elasticsearch. Other requests should continue to succeed. A value of 0 indicates no recent failures. If you see a sustained value of 1, troubleshoot the requests your clients are making to the plugin.<br><br>Relevant statistics: Maximum |

## KNN Metrics

The `AWS/ES` namespace includes metrics for KNN (p. 153). For a summary of each, see the Open Distro for Elasticsearch documentation.

# Configuring Logs

Amazon ES exposes three Elasticsearch logs through Amazon CloudWatch Logs: error logs, search slow logs, and index slow logs. These logs are useful for troubleshooting performance and stability issues, but are *disabled* by default. If enabled, standard CloudWatch pricing applies.

**Note**
Error logs are available only for Elasticsearch versions 5.1 and greater. Slow logs are available for all Elasticsearch versions.

For its logs, Elasticsearch uses Apache Log4j 2 and its built-in log levels (from least to most severe) of `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`, and `FATAL`.

If you enable error logs, Amazon ES publishes log lines of `WARN`, `ERROR`, and `FATAL` to CloudWatch. Amazon ES also publishes several exceptions from the `DEBUG` level, including the following:

- `org.elasticsearch.index.mapper.MapperParsingException`
- `org.elasticsearch.index.query.QueryShardException`
- `org.elasticsearch.action.search.SearchPhaseExecutionException`
- `org.elasticsearch.common.util.concurrent.EsRejectedExecutionException`
- `java.lang.IllegalArgumentException`

Error logs can help with troubleshooting in many situations, including the following:

- Painless script compilation issues
- Invalid queries
- Indexing issues

- Snapshot failures

## Enabling Log Publishing (Console)

The Amazon ES console is the simplest way to enable the publishing of logs to CloudWatch.

**To enable log publishing to CloudWatch (console)**

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. In the navigation pane, under **My domains**, choose the domain that you want to update.
4. On the **Logs** tab, choose **Enable** for the log that you want.
5. Create a CloudWatch log group, or choose an existing one.

   **Note**
   If you plan to enable multiple logs, we recommend publishing each to its own log group. This separation makes the logs easier to scan.
6. Choose an access policy that contains the appropriate permissions, or create a policy using the JSON that the console provides:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "es.amazonaws.com"
      },
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogStream"
      ],
      "Resource": "cw_log_group_arn"
    }
  ]
}
```

   **Important**
   CloudWatch Logs supports 10 resource policies per Region. If you plan to enable logs for several Amazon ES domains, you should create and reuse a broader policy that includes multiple log groups to avoid reaching this limit.
7. Choose **Enable**.

   The status of your domain changes from **Active** to **Processing**. The status must return to **Active** before log publishing is enabled. This change typically takes 30 minutes, but can take longer depending on your domain configuration.

If you enabled one of the slow logs, see the section called “Setting Elasticsearch Logging Thresholds for Slow Logs” (p. 43). If you enabled only error logs, you don't need to perform any additional configuration steps.

## Enabling Log Publishing (AWS CLI)

Before you can enable log publishing, you need a CloudWatch log group. If you don't already have one, you can create one using the following command:

```
aws logs create-log-group --log-group-name my-log-group
```

Enter the next command to find the log group's ARN, and then *make a note of it*:

```
aws logs describe-log-groups --log-group-name my-log-group
```

Now you can give Amazon ES permissions to write to the log group. You must provide the log group's ARN near the end of the command:

```
aws logs put-resource-policy --policy-name my-policy --policy-document
 '{ "Version": "2012-10-17", "Statement": [{ "Sid": "", "Effect":
 "Allow", "Principal": { "Service": "es.amazonaws.com"}, "Action":
[ "logs:PutLogEvents","logs:PutLogEventsBatch","logs:CreateLogStream"],"Resource":
 "cw_log_group_arn"}]}'
```

**Important**
CloudWatch Logs supports 10 resource policies per Region. If you plan to enable slow logs for several Amazon ES domains, you should create and reuse a broader policy that includes multiple log groups to avoid reaching this limit.

Finally, you can use the `--log-publishing-options` option to enable publishing. The syntax for the option is the same for both the `create-elasticsearch-domain` and `update-elasticsearch-domain-config` commands.

| Parameter | Valid Values |
|---|---|
| `--log-publishing-options` | `SEARCH_SLOW_LOGS={CloudWatchLogsLogGroupArn=cw_log_group_arn,Enab false}` |
| | `INDEX_SLOW_LOGS={CloudWatchLogsLogGroupArn=cw_log_group_arn,Enabl false}` |
| | `ES_APPLICATION_LOGS={CloudWatchLogsLogGroupArn=cw_log_group_arn,E false}` |

**Note**
If you plan to enable multiple logs, we recommend publishing each to its own log group. This separation makes the logs easier to scan.

**Example**

The following example enables the publishing of search and index slow logs for the specified domain:

```
aws es update-elasticsearch-domain-config --domain-name my-domain --log-publishing-options
 "SEARCH_SLOW_LOGS={CloudWatchLogsLogGroupArn=arn:aws:logs:us-east-1:123456789012:log-
group:my-log-
group,Enabled=true},INDEX_SLOW_LOGS={CloudWatchLogsLogGroupArn=arn:aws:logs:us-
east-1:123456789012:log-group:my-other-log-group,Enabled=true}"
```

To disable publishing to CloudWatch, run the same command with `Enabled=false`.

If you enabled one of the slow logs, see the section called “Setting Elasticsearch Logging Thresholds for Slow Logs” (p. 43). If you enabled only error logs, you don't need to perform any additional configuration steps.

## Enabling Log Publishing (AWS SDKs)

Before you can enable log publishing, you must first create a CloudWatch log group, get its ARN, and give Amazon ES permissions to write to it. The relevant operations are documented in the Amazon CloudWatch Logs API Reference:

- `CreateLogGroup`
- `DescribeLogGroup`
- `PutResourcePolicy`

You can access these operations using the AWS SDKs.

The AWS SDKs (except the Android and iOS SDKs) support all the operations that are defined in *Amazon ES Configuration API Reference* (p. 236), including the `--log-publishing-options` option for `CreateElasticsearchDomain` and `UpdateElasticsearchDomainConfig`.

If you enabled one of the slow logs, see the section called “Setting Elasticsearch Logging Thresholds for Slow Logs” (p. 43). If you enabled only error logs, you don't need to perform any additional configuration steps.

## Setting Elasticsearch Logging Thresholds for Slow Logs

Elasticsearch disables slow logs by default. After you enable the *publishing* of slow logs to CloudWatch, you still must specify logging thresholds for each Elasticsearch index. These thresholds define precisely what should be logged and at which log level.

You specify these settings through the Elasticsearch REST API:

```
PUT elasticsearch_domain_endpoint/index/_settings
{
  "index.search.slowlog.threshold.query.warn": "5s",
  "index.search.slowlog.threshold.query.info": "2s"
}
```

To test that slow logs are publishing successfully, consider starting with very low values to verify that logs appear in CloudWatch, and then increase the thresholds to more useful levels.

If the logs don't appear, check the following:

- Does the CloudWatch log group exist? Check the CloudWatch console.
- Does Amazon ES have permissions to write to the log group? Check the Amazon ES console.
- Is the Amazon ES domain configured to publish to the log group? Check the Amazon ES console, use the AWS CLI `describe-elasticsearch-domain-config` option, or call `DescribeElasticsearchDomainConfig` using one of the SDKs.
- Are the Elasticsearch logging thresholds low enough that your requests are exceeding them? To review your thresholds for an index, use the following command:

  ```
  GET elasticsearch_domain_endpoint/index/_settings?pretty
  ```

If you want to disable slow logs for an index, return any thresholds that you changed to their default values of `-1`.

Disabling publishing to CloudWatch using the Amazon ES console or AWS CLI does *not* stop Elasticsearch from generating logs; it only stops the *publishing* of those logs. Be sure to check your index settings if you no longer need the slow logs.

## Viewing Logs

Viewing the application and slow logs in CloudWatch is just like viewing any other CloudWatch log. For more information, see View Log Data in the *Amazon CloudWatch Logs User Guide*.

Here are some considerations for viewing the logs:

- Amazon ES publishes only the first 255,000 characters of each line to CloudWatch. Any remaining content is truncated.
- In CloudWatch, the log stream names have suffixes of `-index-slow-logs`, `-search-slow-logs`, and `-es-application-logs` to help identify their contents.

# Working with Amazon Elasticsearch Service Index Snapshots

Snapshots are backups of a cluster's indices and *state*. State includes cluster settings, node information, index settings, and shard allocation.

On Amazon Elasticsearch Service, snapshots come in two forms: automated and manual.

- Automated snapshots are *only* for cluster recovery. You can use them to restore your domain (p. 49) in the event of red cluster status (p. 228) or other data loss. Amazon ES stores automated snapshots in a preconfigured Amazon S3 bucket at no additional charge.
- Manual snapshots are for cluster recovery *or* moving data from one cluster to another. As the name suggests, you have to initiate manual snapshots. These snapshots are stored in your own Amazon S3 bucket, and standard S3 charges apply. If you have a snapshot from a self-managed Elasticsearch cluster, you can even use that snapshot to migrate to an Amazon ES domain.

All Amazon ES domains take automated snapshots, but frequency differs:

- For domains running Elasticsearch 5.3 and later, Amazon ES takes hourly automated snapshots and retains up to 336 of them for 14 days.
- For domains running Elasticsearch 5.1 and earlier, Amazon ES takes daily automated snapshots (during the hour you specify) and retains up to 14 of them for 30 days.

If your cluster enters red status, Amazon ES stops taking automated snapshots. If you don't correct the problem within two weeks, you can permanently lose your cluster's data. For troubleshooting steps, see the section called “Red Cluster Status” (p. 228).

**Topics**

## Manual Snapshot Prerequisites

To create snapshots manually, you must work with IAM and Amazon S3. Verify that you have met the following prerequisites before you attempt to take a snapshot.

| Prerequisite | Description |
|---|---|
| S3 bucket | Stores manual snapshots for your Amazon ES domain. Make a note of the bucket's name. You need it in two places: • `Resource` statement of the IAM policy that is attached to your IAM role • Python client that is used to register a snapshot repository For more information, see Create a Bucket in the *Amazon Simple Storage Service Getting Started Guide*. **Important** Do **not** apply an S3 Glacier lifecycle rule to this bucket. Manual snapshots do not support the S3 Glacier storage class. |
| IAM role | Delegates permissions to Amazon Elasticsearch Service. The rest of this chapter refers to this role as `TheSnapshotRole`. The trust relationship for the role must specify Amazon Elasticsearch Service in the `Principal` statement, as shown in the following example: |

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "",
    "Effect": "Allow",
    "Principal": {
      "Service": "es.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
  }]
}
```

The role must have the following policy attached to it:

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Action": [
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::s3-bucket-name"
      ]
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Effect": "Allow",
      "Resource": [
```

| Prerequisite | Description |
|---|---|
| | |

```
           "arn:aws:s3:::s3-bucket-name/*"
         ]
       }
     ]
}
```

| Prerequisite | Description |
|---|---|
| | For more information, see Adding IAM Identity Permissions in the *IAM User Guide*. |
| Permissions | You must be able to assume `TheSnapshotRole` in order to register the snapshot repository. You also need access to the `es:ESHttpPut` action. The following policy includes these permissions: |

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/TheSnapshotRole"
    },
    {
      "Effect": "Allow",
      "Action": "es:ESHttpPut",
      "Resource": "arn:aws:es:region:123456789012:domain/my-domain/*"
    }
  ]
}
```

If you don't have `iam:PassRole` permissions to assume `TheSnapshotRole`, you might encounter the following common error:

```
$ python register-repo.py
{"Message":"User: arn:aws:iam::123456789012:user/MyUserAccount
is not authorized to perform: iam:PassRole on resource:
arn:aws:iam::123456789012:role/TheSnapshotRole"}
```

## Registering a Manual Snapshot Repository

You must register a snapshot repository with Amazon Elasticsearch Service before you can take manual index snapshots. This one-time operation requires that you sign your AWS request with credentials that are allowed to access `TheSnapshotRole`, as described in the section called "Manual Snapshot Prerequisites" (p. 45).

You can't use `curl` to perform this operation, because it doesn't support AWS request signing. Instead, use the sample Python client (p. 47), Postman, or some other method to send a signed request (p. 113) to register the snapshot repository. The request takes the following form:

```
PUT elasticsearch-domain-endpoint/_snapshot/my-snapshot-repo-name
{
  "type": "s3",
  "settings": {
    "bucket": "s3-bucket-name",
    "region": "region",
    "role_arn": "arn:aws:iam::123456789012:role/TheSnapshotRole"
  }
}
```

Registering a snapshot directory is a one-time operation, but to migrate from one domain to another, you must register the same snapshot repository on the old domain and the new domain. The repository name is arbitrary.

**Important**

If the S3 bucket is in the us-east-1 region, you must use `"endpoint": "s3.amazonaws.com"` instead of `"region": "us-east-1"`.

To enable server-side encryption with S3-managed keys for the snapshot repository, add `"server_side_encryption": true` to the `"settings"` JSON.

If your domain resides within a VPC, your computer must be connected to the VPC in order for the request to successfully register the snapshot repository. Accessing a VPC varies by network configuration, but likely involves connecting to a VPN or corporate network. To check that you can reach the Amazon ES domain, navigate to `https://`*`your-vpc-domain`*`.`*`region`*`.es.amazonaws.com` in a web browser and verify that you receive the default JSON response.

If you use fine-grained access control, see the section called "Manual Snapshots" (p. 93) for an additional step.

## Sample Python Client

Save the following sample Python code as a Python file, such as `register-repo.py`. The client requires the AWS SDK for Python (Boto 3), requests and requests-aws4auth packages. The client contains commented-out examples for other snapshot operations.

**Tip**

A Java-based code sample is available in Signing HTTP Requests (p. 113).

You must update the following variables: `host`, `region`, `path`, and `payload`.

```
import boto3
import requests
from requests_aws4auth import AWS4Auth

host = '' # include https:// and trailing /
region = '' # e.g. us-west-1
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

# Register repository

path = '_snapshot/my-snapshot-repo-name' # the Elasticsearch API endpoint
url = host + path

payload = {
  "type": "s3",
  "settings": {
    "bucket": "s3-bucket-name",
    # "endpoint": "s3.amazonaws.com", # for us-east-1
    "region": "us-west-1", # for all other regions
    "role_arn": "arn:aws:iam::123456789012:role/TheSnapshotRole"
  }
}

headers = {"Content-Type": "application/json"}

r = requests.put(url, auth=awsauth, json=payload, headers=headers)

print(r.status_code)
print(r.text)

# # Take snapshot
#
# path = '_snapshot/my-snapshot-repo/my-snapshot'
# url = host + path
#
# r = requests.put(url, auth=awsauth)
#
# print(r.text)
#
# # Delete index
#
# path = 'my-index'
# url = host + path
#
# r = requests.delete(url, auth=awsauth)
#
# print(r.text)
#
# # Restore snapshot (all indices except Kibana and fine-grained access control)
#
# path = '_snapshot/my-snapshot-repo/my-snapshot/_restore'
# url = host + path
#
# payload = {
#   "indices": "-.kibana*,-.opendistro_security",
#   "include_global_state": false
# }
#
# headers = {"Content-Type": "application/json"}
#
# r = requests.post(url, auth=awsauth, json=payload, headers=headers)
#
# # Restore snapshot (one index)
#
# path = '_snapshot/my-snapshot-repo/my-snapshot/_restore'
# url = host + path
#
# payload = {"indices": "my-index"}
#
# headers = {"Content-Type": "application/json"}
#
# r = requests.post(url, auth=awsauth, json=payload, headers=headers)
#
# print(r.text)
```

## Taking Manual Snapshots

Snapshots are not instantaneous; they take time to complete and do not represent perfect point-in-time views of the cluster. While a snapshot is in-progress, you can still index documents and make other requests to the cluster, but new documents (and updates to existing documents) generally aren't included in the snapshot. The snapshot includes primary shards as they existed when Elasticsearch initiated the snapshot. Depending on the size of your snapshot thread pool, different shards might be included in the snapshot at slightly different times.

Elasticsearch snapshots are incremental, meaning that they only store data that has changed since the last successful snapshot. This incremental nature means that the difference in disk usage between frequent and infrequent snapshots is often minimal. In other words, taking hourly snapshots for a week (for a total of 168 snapshots) might not use much more disk space than taking a single snapshot at the end of the week. Also, the more frequently you take snapshots, the less time they take to complete. Some Elasticsearch users take snapshots as often as every half hour.

You specify two pieces of information when you create a snapshot:

- Name of your snapshot repository
- Name for the snapshot

The examples in this chapter use curl, a common HTTP client, for convenience and brevity. If your access policies specify IAM users or roles, however, you must sign your snapshot requests. You can use the commented-out examples in the sample Python client (p. 47) to make signed HTTP requests to the same endpoints that the curl commands use.

### To manually take a snapshot

1. You can't take a snapshot if one is currently in progress. To check, run the following command:

```
curl -XGET 'elasticsearch-domain-endpoint/_snapshot/_status'
```

2. Run the following command to manually take a snapshot:

```
curl -XPUT 'elasticsearch-domain-endpoint/_snapshot/repository/snapshot-name'
```

**Note**
The time required to take a snapshot increases with the size of the Amazon ES domain. Long-running snapshot operations sometimes encounter the following error: `504 GATEWAY_TIMEOUT`. Typically, you can ignore these errors and wait for the operation to complete successfully. Use the following command to verify the state of all snapshots of your domain:

```
curl -XGET 'elasticsearch-domain-endpoint/_snapshot/repository/_all?pretty'
```

## Restoring Snapshots

**Warning**
If you use index aliases, cease write requests to an alias (or switch the alias to another index) prior to deleting its index. Halting write requests helps avoid the following scenario:

1. You delete an index, which also deletes its alias.
2. An errant write request to the now-deleted alias creates a new index with the same name as the alias.
3. You can no longer use the alias due to a naming conflict with the new index.

If you switched the alias to another index, specify `"include_aliases": false` when you restore from a snapshot.

### To restore a snapshot

1. Identify the snapshot that you want to restore. To see all snapshot repositories, run the following command:

```
curl -XGET 'elasticsearch-domain-endpoint/_snapshot?pretty'
```

After you identify the repository, run the following command to see all snapshots:

```
curl -XGET 'elasticsearch-domain-endpoint/_snapshot/repository/_all?pretty'
```

> **Note**
> Most automated snapshots are stored in the `cs-automated` repository. If your domain encrypts data at rest, they are stored in the `cs-automated-enc` repository. If you don't see the manual snapshot repository that you're looking for, make sure that you registered it (p. 46) to the domain.

2. (Optional) Delete or rename one or more indices in the Amazon ES domain. You don't need to perform this step if you have no naming conflicts between indices on the cluster and indices in the snapshot.

   You can't restore a snapshot of your indices to an Elasticsearch cluster that already contains indices with the same names. Currently, Amazon ES does not support the Elasticsearch `_close` API, so you must use one of the following alternatives:

   - Delete the indices on the same Amazon ES domain, and then restore the snapshot.
   - Rename the indices as you restore them from the snapshot (p. 232), and later, reindex them.
   - Restore the snapshot to a different Amazon ES domain (only possible with manual snapshots).

   The following example shows how to delete *all* existing indices for a domain:

   ```
   curl -XDELETE 'elasticsearch-domain-endpoint/_all'
   ```

   If you don't plan to restore all indices, though, you might want to delete only one:

   ```
   curl -XDELETE 'elasticsearch-domain-endpoint/index-name'
   ```

3. To restore a snapshot, run the following command:

   ```
   curl -XPOST 'elasticsearch-domain-endpoint/_snapshot/repository/snapshot/_restore'
   ```

   Due to special permissions on the Kibana and fine-grained access control indices, attempts to restore all indices might fail, especially if you try to restore from an automated snapshot. The following example restores just one index, `my-index`, from `2017-snapshot` in the `cs-automated` snapshot repository:

   ```
   curl -XPOST 'elasticsearch-domain-endpoint/_snapshot/cs-automated/2017-snapshot/
   _restore' -d '{"indices": "my-index"}' -H 'Content-Type: application/json'
   ```

   Alternately, you might want to restore all indices *except* for the Kibana and fine-grained access control indices:

   ```
   curl -XPOST 'elasticsearch-domain-endpoint/_snapshot/cs-automated/2017-snapshot/
   _restore' -d '{"indices": "-.kibana*, -.opendistro_security"}' -H 'Content-Type:
    application/json'
   ```

   > **Note**
   > If not all primary shards were available for the indices involved, a snapshot might have a `state` of `PARTIAL`. This value indicates that data from at least one shard was not stored successfully. You can still restore from a partial snapshot, but you might need to use older snapshots to restore any missing indices.

## Using Curator for Snapshots

Some users find tools like Curator convenient for index and snapshot management. Use pip to install Curator:

```
pip install elasticsearch-curator
```

Curator offers advanced filtering functionality that can help simplify management tasks on complex clusters. Amazon ES supports Curator on domains running Elasticsearch version 5.1 and above. You can use Curator as a command line interface (CLI) or Python API. If you use the CLI, export your credentials at the command line and configure `curator.yml` as follows:

```
client:
  hosts: search-my-domain.us-west-1.es.amazonaws.com
  port: 443
  use_ssl: True
  aws_region: us-west-1
  aws_sign_request: True
  ssl_no_validate: False
  timeout: 60

logging:
  loglevel: INFO
```

For sample Lambda functions that use the Python API, see the section called "Using Curator to Rotate Data" (p. 125).

# Upgrading Elasticsearch

**Note**
Elasticsearch version upgrades differ from service software updates. For information on updating the service software for your Amazon ES domain, see the section called "Service Software Updates" (p. 15).

Amazon ES offers in-place Elasticsearch upgrades for domains that run versions 5.1 and later. If you use services like Amazon Kinesis Data Firehose or Amazon CloudWatch Logs to stream data to Amazon ES, check that these services support the newer version of Elasticsearch before migrating.

Currently, Amazon ES supports the following upgrade paths.

| From Version | To Version |
|---|---|
| 7.*x* | 7.*x* |
| 6.8 | 7.*x*<br><br>**Important**<br>Elasticsearch 7.0 includes numerous breaking changes. Before initiating an in-place upgrade, we recommend taking a manual snapshot (p. 44) of the 6.8 domain, restoring it on a test 7.*x* domain, and using that test domain to identify potential upgrade issues.<br>Like Elasticsearch 6.*x*, indices can only contain one mapping type, but that type must now be named `_doc`. As a result, certain APIs no longer require a mapping type in the request body (such as the `_bulk` API).<br>For new indices, self-hosted Elasticsearch 7.*x* has a default shard count of one. Amazon ES 7.*x* domains retain the previous default of five. |

| From Version | To Version |
|---|---|
| 6.*x* | 6.*x* |
| 5.6 | 6.*x*<br><br>**Important**<br>Indices created in version 6.*x* no longer support multiple mapping types. Indices created in version 5.*x* still support multiple mapping types when restored into a 6.*x* cluster. Check that your client code creates only a single mapping type per index.<br>To minimize downtime during the upgrade from Elasticsearch 5.6 to 6.*x*, Amazon ES reindexes the `.kibana` index to `.kibana-6`, deletes `.kibana`, creates an alias named `.kibana`, and maps the new index to the new alias. |
| 5.*x* | 5.6 |

The upgrade process consists of three steps:

1. **Pre-upgrade checks** – Amazon ES performs a series of checks for issues that can block an upgrade and doesn't proceed to the next step unless these checks succeed.
2. **Snapshot** – Amazon ES takes a snapshot of the Elasticsearch cluster and doesn't proceed to the next step unless the snapshot succeeds. If the upgrade fails, Amazon ES uses this snapshot to restore the cluster to its original state. For more information about this snapshot, see the section called “Can't Downgrade After Upgrade” (p. 233).
3. **Upgrade** – Amazon ES starts the upgrade, which can take from 15 minutes to several hours to complete. Kibana might be unavailable during some or all of the upgrade.

## Troubleshooting an Upgrade

In-place Elasticsearch upgrades require healthy domains. Your domain might be ineligible for an upgrade or fail to upgrade for a wide variety of reasons. The following table shows the most common issues.

| Issue | Description |
|---|---|
| Too many shards per node | The 7.*x* versions of Elasticsearch have a default setting of no more than 1,000 shards per node. If a node in your current cluster exceeds this setting, Amazon ES doesn't allow you to upgrade. See the section called “Maximum Shard Limit” (p. 232) for troubleshooting options. |
| Domain in processing | The domain is in the middle of a configuration change. Check upgrade eligibility after the operation completes. |
| Red cluster status | One or more indices in the cluster is red. For troubleshooting steps, see the section called “Red Cluster Status” (p. 228). |
| High error rate | The Elasticsearch cluster is returning a large number of 5*xx* errors when attempting to process requests. This problem is usually the result of too many simultaneous read or write requests. Consider reducing traffic to the cluster or scaling your domain. |
| Split brain | *Split brain* means that your Elasticsearch cluster has more than one master node and has split into two clusters that never will rejoin on their own. You can avoid split brain by using the recommended number of dedicated master nodes (p. 176). For help recovering from split brain, contact AWS Support. |

| Issue | Description |
|---|---|
| Master node not found | Amazon ES can't find the cluster's master node. If your domain uses multi-AZ (p. 17), an Availability Zone failure might have caused the cluster to lose quorum and be unable to elect a new master node (p. 176). If the issue does not self-resolve, contact AWS Support. |
| Too many pending tasks | The master node is under heavy load and has many pending tasks. Consider reducing traffic to the cluster or scaling your domain. |
| Impaired storage volume | The disk volume of one or more nodes isn't functioning properly. This issue often occurs alongside other issues, like a high error rate or too many pending tasks. If it occurs in isolation and doesn't self-resolve, contact AWS Support. |
| KMS key issue | The KMS key that is used to encrypt the domain is either inaccessible or missing. For more information, see the section called "Monitoring Domains That Encrypt Data at Rest" (p. 62). |
| Snapshot in progress | The domain is currently taking a snapshot. Check upgrade eligibility after the snapshot finishes. Also check that you can list manual snapshot repositories, list snapshots within those repositories, and take manual snapshots. If Amazon ES is unable to check whether a snapshot is in progress, upgrades can fail. |
| Snapshot timeout or failure | The pre-upgrade snapshot took too long to complete or failed. Check cluster health, and try again. If the problem persists, contact AWS Support. |
| Incompatible indices | One or more indices is incompatible with the target Elasticsearch version. This problem can occur if you migrated the indices from an older version of Elasticsearch, like 2.3. Reindex the indices, and try again. |
| High disk usage | Disk usage for the cluster is above 90%. Delete data or scale the domain, and try again. |
| High JVM usage | JVM memory pressure is above 75%. Reduce traffic to the cluster or scale the domain, and try again. |
| Kibana alias problem | `.kibana` is already configured as an alias and maps to an incompatible index, likely one from an earlier version of Kibana. Reindex, and try again. |
| Red Kibana status | Kibana status is red. Try using Kibana when the upgrade completes. If the red status persists, resolve it manually, and try again. |
| Other Amazon ES service issue | Issues with Amazon ES itself might cause your domain to display as ineligible for an upgrade. If none of the preceding conditions apply to your domain and the problem persists for more than a day, contact AWS Support. |

## Starting an Upgrade

The upgrade process is irreversible and can't be paused or canceled. During an upgrade, you can't make configuration changes to the domain. Before starting an upgrade, double-check that you want to proceed. You can use these same steps to perform the pre-upgrade check without actually starting an upgrade.

If the cluster has dedicated master nodes, upgrades complete without downtime. Otherwise, the cluster might be unresponsive for several seconds post-upgrade while it elects a master node.

**To upgrade a domain to a later version of Elasticsearch (console)**

1. Take a manual snapshot (p. 44) of your domain. This snapshot serves as a backup that you can restore on a new domain (p. 49) if you want to return to using the prior Elasticsearch version.
2. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
3. Under **Analytics**, choose **Elasticsearch Service**.
4. In the navigation pane, under **My domains**, choose the domain that you want to upgrade.
5. Choose **Actions** and **Upgrade domain**.
6. For **Operation**, choose **Upgrade**, **Submit**, and **Continue**.
7. Return to the **Overview** tab and choose **Upgrade status** to monitor the state of the upgrade.

**To upgrade a domain to a later version of Elasticsearch (AWS CLI and SDK)**

You can use the following operations to identify the right Elasticsearch version for your domain, start an in-place upgrade, perform the pre-upgrade check, and view progress:

- `get-compatible-elasticsearch-versions` (`GetCompatibleElasticsearchVersions`)
- `upgrade-elasticsearch-domain` (`UpgradeElasticsearchDomain`)
- `get-upgrade-status` (`GetUpgradeStatus`)
- `get-upgrade-history` (`GetUpgradeHistory`)

For more information, see the AWS CLI Command Reference and *Amazon ES Configuration API Reference* (p. 236).

## Using a Snapshot to Migrate Data

In-place upgrades are the easier, faster, and more reliable way to upgrade a domain to a later Elasticsearch version. Snapshots are a good option if you need to migrate from a pre-5.1 version of Elasticsearch or want to migrate to an entirely new cluster.

The following table shows how to use snapshots to migrate data to a domain that uses a different Elasticsearch version. For more information about taking and restoring snapshots, see the section called "Working with Index Snapshots" (p. 44).

| From Version | To Version | Migration Process |
|---|---|---|
| 6.*x* | 7.*x* | 1. Review breaking changes for 7.0 to see if you need to make adjustments to your indices or applications. For other considerations, see the table in the section called "Upgrading Elasticsearch" (p. 51).<br>2. Create a manual snapshot of the 6.*x* domain.<br>3. Create a 7.*x* domain.<br>4. Restore the snapshot from the original domain to the 7.*x* domain. During the operation, you likely need to restore the `.kibana` index under a new name:<br>`POST _snapshot/<repository-name>/<snapshot-name>/`<br>`_restore`<br>`{`<br>`  "indices": "*",`<br>`  "ignore_unavailable": true,`<br>`  "rename_pattern": ".kibana",`<br>`  "rename_replacement": ".backup-kibana"` |

| From Version | To Version | Migration Process |
|---|---|---|
| | | `}`<br><br>Then you can reindex `.backup-kibana` on the new domain and alias it to `.kibana`.<br>5. If you no longer need your original domain, delete it. Otherwise, you continue to incur charges for the domain. |
| 6.*x* | 6.8 | 1. Create a manual snapshot of the 6.*x* domain.<br>2. Create a 6.8 domain.<br>3. Restore the snapshot from the original domain to the 6.8 domain.<br>4. If you no longer need your original domain, delete it. Otherwise, you continue to incur charges for the domain. |
| 5.*x* | 6.*x* | 1. Review breaking changes for 6.0 to see if you need to make adjustments to your indices or applications. For other considerations, see the table in the section called “Upgrading Elasticsearch” (p. 51).<br>2. Create a manual snapshot of the 5.*x* domain.<br>3. Create a 6.*x* domain.<br>4. Restore the snapshot from the original domain to the 6.*x* domain.<br>5. If you no longer need your 5.*x* domain, delete it. Otherwise, you continue to incur charges for the domain. |
| 5.*x* | 5.6 | 1. Create a manual snapshot of the 5.*x* domain.<br>2. Create a 5.6 domain.<br>3. Restore the snapshot from the original domain to the 5.6 domain.<br>4. If you no longer need your original domain, delete it. Otherwise, you continue to incur charges for the domain. |
| 2.3 | 6.*x* | Elasticsearch 2.3 snapshots are not compatible with 6.*x*. To migrate your data directly from 2.3 to 6.*x*, you must manually recreate your indices in the new domain.<br><br>Alternately, you can follow the 2.3 to 5.*x* steps in this table, perform `_reindex` operations in the new 5.*x* domain to convert your 2.3 indices to 5.*x* indices, and then follow the 5.*x* to 6.*x* steps. |
| 2.3 | 5.*x* | 1. Review breaking changes for 5.0 to see if you need to make adjustments to your indices or applications.<br>2. Create a manual snapshot of the 2.3 domain.<br>3. Create a 5.*x* domain.<br>4. Restore the snapshot from the 2.3 domain to the 5.*x* domain.<br>5. If you no longer need your 2.3 domain, delete it. Otherwise, you continue to incur charges for the domain. |

| From Version | To Version | Migration Process |
|---|---|---|
| 1.5 | 5.*x* | Elasticsearch 1.5 snapshots are not compatible with 5.*x*. To migrate your data from 1.5 to 5.*x*, you must manually recreate your indices in the new domain. **Important** 1.5 snapshots *are* compatible with 2.3, but Amazon ES 2.3 domains do not support the `_reindex` operation. Because you cannot reindex them, indices that originated in a 1.5 domain still fail to restore from 2.3 snapshots to 5.*x* domains. |
| 1.5 | 2.3 | 1. Use the migration plugin to find out if you can directly upgrade to version 2.3. You might need to make changes to your data before migration. a. In a web browser, open `http://`*`domain_endpoint`*`/_plugin/migration/`. b. Choose **Run checks now**. c. Review the results and, if needed, follow the instructions to make changes to your data. 2. Create a manual snapshot of the 1.5 domain. 3. Create a 2.3 domain. 4. Restore the snapshot from the 1.5 domain to the 2.3 domain. 5. If you no longer need your 1.5 domain, delete it. Otherwise, you continue to incur charges for the domain. |

## Tagging Amazon Elasticsearch Service Domains

You can use Amazon ES tags to add metadata to your Amazon ES domains. AWS does not apply any semantic meaning to your tags. Tags are interpreted strictly as character strings. All tags have the following elements.

| Tag Element | Description |
|---|---|
| Tag key | The tag key is the required name of the tag. Tag keys must be unique for the Amazon ES domain to which they are attached. For a list of basic restrictions on tag keys and values, see User-Defined Tag Restrictions. |
| Tag value | The tag value is an optional string value of the tag. Tag values can be `null` and do not have to be unique in a tag set. For example, you can have a key-value pair in a tag set of project/Trinity and cost-center/Trinity. For a list of basic restrictions on tag keys and values, see User-Defined Tag Restrictions. |

Each Amazon ES domain has a tag set, which contains all the tags that are assigned to that Amazon ES domain. AWS does not automatically set any tags on Amazon ES domains. A tag set can contain up to 50 tags, or it can be empty. If you add a tag to an Amazon ES domain that has the same key as an existing tag for a resource, the new value overwrites the old value.

You can use these tags to track costs by grouping expenses for similarly tagged resources. An Amazon ES domain tag is a name-value pair that you define and associate with an Amazon ES domain. The name is referred to as the *key*. You can use tags to assign arbitrary information to an Amazon ES domain. A tag key could be used, for example, to define a category, and the tag value could be an item in that category. For example, you could define a tag key of “project” and a tag value of “Salix,” indicating that the Amazon ES domain is assigned to the Salix project. You could also use tags to designate Amazon ES domains as being used for test or production by using a key such as `environment=test` or `environment=production`. We recommend that you use a consistent set of tag keys to make it easier to track metadata that is associated with Amazon ES domains.

You also can use tags to organize your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. Then, organize your billing information according to resources with the same tag key values to see the cost of combined resources. For example, you can tag several Amazon ES domains with key-value pairs, and then organize your billing information to see the total cost for each domain across several services. For more information, see Using Cost Allocation Tags in the *AWS Billing and Cost Management* documentation.

> **Note**
> Tags are cached for authorization purposes. Because of this, additions and updates to tags on Amazon ES domains might take several minutes before they are available.

## Working with Tags (Console)

Use the following procedure to create a resource tag.

**To create a tag (console)**

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. In the navigation pane, choose your Amazon ES domain.
4. On the domain dashboard, choose **Manage tags**.
5. In the **Key** column, enter a tag key.
6. (Optional) In the **Value** column, enter a tag value.
7. Choose **Submit**.

**To delete a tag (console)**

Use the following procedure to delete a resource tag.

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. In the navigation pane, choose your Amazon ES domain.
4. On the domain dashboard, choose **Manage tags**.
5. Next to the tag that you want to delete, choose **Remove**.
6. Choose **Submit**.

For more information about using the console to work with tags, see Working with Tag Editor in the *AWS Management Console Getting Started Guide*.

## Working with Tags (AWS CLI)

You can create resource tags using the AWS CLI with the **--add-tags** command.

**Syntax**

```
add-tags --arn=<domain_arn> --tag-list Key=<key>,Value=<value>
```

| Parameter | Description |
|---|---|
| `--arn` | Amazon resource name for the Amazon ES domain to which the tag is attached. |
| `--tag-list` | Set of space-separated key-value pairs in the following format: `Key=<key>,Value=<value>` |

**Example**

The following example creates two tags for the *logs* domain:

```
aws es add-tags --arn arn:aws:es:us-east-1:379931976431:domain/logs --tag-list
 Key=service,Value=Elasticsearch Key=instances,Value=m3.2xlarge
```

You can remove tags from an Amazon ES domain using the **remove-tags** command.

**Syntax**

```
remove-tags --arn=<domain_arn> --tag-keys Key=<key>,Value=<value>
```

| Parameter | Description |
|---|---|
| `--arn` | Amazon Resource Name (ARN) for the Amazon ES domain to which the tag is attached. |
| `--tag-keys` | Set of space-separated key-value pairs that you want to remove from the Amazon ES domain. |

**Example**

The following example removes two tags from the *logs* domain that were created in the preceding example:

```
aws es remove-tags --arn arn:aws:es:us-east-1:379931976431:domain/logs --tag-keys service
 instances
```

You can view the existing tags for an Amazon ES domain with the **list-tags** command:

**Syntax**

```
list-tags --arn=<domain_arn>
```

| Parameter | Description |
|---|---|
| `--arn` | Amazon Resource Name (ARN) for the Amazon ES domain to which the tags are attached. |

**Example**

The following example lists all resource tags for the *logs* domain:

```
aws es list-tags --arn arn:aws:es:us-east-1:379931976431:domain/logs
```

## Working with Tags (AWS SDKs)

The AWS SDKs (except the Android and iOS SDKs) support all the actions defined in the Amazon ES Configuration API Reference (p. 236), including the `AddTags`, `ListTags`, and `RemoveTags` operations. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

# Security in Amazon Elasticsearch Service

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS 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 AWS compliance programs. To learn about the compliance programs that apply to Amazon Elasticsearch Service, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS 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 ES. The following topics show you how to configure Amazon ES to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon ES resources.

**Topics**

## Data Protection in Amazon Elasticsearch Service

Amazon Elasticsearch Service conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. Choose **Require HTTPS** when you create domains.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into domain names, index names, document types, or document IDs. Elasticsearch uses these names in its Uniform Resource Identifiers (URIs). Servers and applications often log HTTP requests, which can lead to unnecessary data exposure if URIs contain sensitive information.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

# Encryption of Data at Rest for Amazon Elasticsearch Service

Amazon ES domains offer encryption of data at rest, a security feature that helps prevent unauthorized access to your data. The feature uses AWS Key Management Service (AWS KMS) to store and manage your encryption keys and the Advanced Encryption Standard algorithm with 256-bit keys (AES-256) to perform the encryption. If enabled, the feature encrypts the following aspects of a domain:

- Indices
- Elasticsearch logs
- Swap files
- All other data in the application directory
- Automated snapshots

The following are *not* encrypted when you enable encryption of data at rest, but you can take additional steps to protect them:

- Manual snapshots: Currently, you can't use KMS master keys to encrypt manual snapshots. You can, however, use server-side encryption with S3-managed keys to encrypt the bucket that you use as a snapshot repository. For instructions, see the section called “Registering a Manual Snapshot Repository” (p. 46).
- Slow logs and error logs: If you publish logs (p. 40) and want to encrypt them, you can encrypt their CloudWatch Logs log group using the same AWS KMS master key as the Amazon ES domain. For more information, see Encrypt Log Data in CloudWatch Logs Using AWS KMS in the *Amazon CloudWatch Logs User Guide*.

Amazon ES supports only symmetric customer master keys, not asymmetric ones. To learn how to create symmetric customer master keys, see Creating Keys in the *AWS Key Management Service Developer Guide*.

Regardless of whether encryption at rest is enabled, all domains automatically encrypt custom packages (p. 148) using AES-256 and Amazon ES-managed keys.

## Enabling Encryption of Data at Rest

By default, domains don't encrypt data at rest, and you can't configure existing domains to use the feature. To enable the feature, you must create another domain (p. 9) and migrate your data. Encryption of data at rest requires Elasticsearch 5.1 or later.

To use the Amazon ES console to create a domain that encrypts data at rest, you must have read-only permissions to AWS KMS, such as the following identity-based policy:

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

If you want to use a key other than **(Default) aws/es**, you must also have permissions to create grants for the key. These permissions typically take the form of a resource-based policy that you specify when you create the key.

If you want to keep your key exclusive to Amazon ES, you can add the kms:ViaService condition to the key policy:

```
"Condition": {
  "StringEquals": {
    "kms:ViaService": "es.us-west-1.amazonaws.com"
  },
  "Bool": {
    "kms:GrantIsForAWSResource": "true"
  }
}
```

For more information, see Using Key Policies in AWS KMS in the *AWS Key Management Service Developer Guide*.

**Warning**
If you delete the key that you used to encrypt a domain, the domain becomes inaccessible. The Amazon ES team can't help you recover your data. AWS KMS deletes master keys only after a waiting period of at least seven days, so the Amazon ES team might contact you if they detect that your domain is at risk.

## Disabling Encryption of Data at Rest

After you configure a domain to encrypt data at rest, you can't disable the setting. Instead, you can take a manual snapshot (p. 44) of the existing domain, create another domain (p. 9), migrate your data, and delete the old domain.

## Monitoring Domains That Encrypt Data at Rest

Domains that encrypt data at rest have two additional metrics: `KMSKeyError` and `KMSKeyInaccessible`. These metrics appear only if the domain encounters a problem with your encryption key. For full descriptions of these metrics, see the section called "Cluster Metrics" (p. 29). You can view them using either the Amazon ES console or the Amazon CloudWatch console.

**Tip**
Each metric represents a significant problem for a domain, so we recommend that you create CloudWatch alarms for both. For more information, see the section called "Recommended CloudWatch Alarms" (p. 178).

## Other Considerations

- Automatic key rotation preserves the properties of your AWS KMS master keys, so the rotation has no effect on your ability to access your Elasticsearch data. Encrypted Amazon ES domains don't support manual key rotation, which involves creating a new master key and updating any references to the old key. To learn more, see Rotating Customer Master Keys in the *AWS Key Management Service Developer Guide*.
- Certain instance types don't support encryption of data at rest. For details, see the section called "Supported Instance Types" (p. 180).
- Domains that encrypt data at rest use a different repository name for their automated snapshots. For more information, see the section called "Restoring Snapshots" (p. 49).
- Encrypting an Amazon ES domain requires a grant, and each encryption key has a limit of 500 grants per principal. This limit means that the maximum number of Amazon ES domains that you can encrypt using a single key is 500. Currently, Amazon ES supports a maximum of 100 domains per account (per Region), so this grant limit is of no consequence. If the domain limit per account increases, however, the grant limit might become relevant.

  If you need to encrypt more than 500 domains at that time, you can create additional keys. Keys are regional, not global, so if you operate in more than one AWS Region, you already need multiple keys.

# Node-to-node Encryption for Amazon Elasticsearch Service

Node-to-node encryption provides an additional layer of security on top of the default features of Amazon ES.

Each Amazon ES domain—regardless of whether the domain uses VPC access—resides within its own, dedicated VPC. This architecture prevents potential attackers from intercepting traffic between Elasticsearch nodes and keeps the cluster secure. By default, however, traffic within the VPC is unencrypted. Node-to-node encryption enables TLS 1.2 encryption for all communications within the VPC.

If you send data to Amazon ES over HTTPS, node-to-node encryption helps ensure that your data remains encrypted as Elasticsearch distributes (and redistributes) it throughout the cluster. If data arrives unencrypted over HTTP, Amazon ES encrypts it after it reaches the cluster. You can require that all traffic to the domain arrive over HTTPS using the console, AWS CLI, or configuration API.

## Enabling Node-to-node Encryption

By default, domains do not use node-to-node encryption, and you can't configure existing domains to use the feature. To enable the feature, you must create another domain (p. 9) and migrate your data (p. 54). Node-to-node encryption requires Elasticsearch 6.0 or later.

## Disabling Node-to-node Encryption

After you configure a domain to use node-to-node encryption, you can't disable the setting. Instead, you can take a manual snapshot (p. 44) of the encrypted domain, create another domain (p. 9), migrate your data, and delete the old domain.

## Other Considerations

- Kibana still works on domains that use node-to-node encryption.

# Identity and Access Management in Amazon Elasticsearch Service

Amazon Elasticsearch Service offers several ways of controlling access to your domains. This section covers the various policy types, how they interact with each other, and how to create your own, custom policies.

**Important**
VPC support introduces some additional considerations to Amazon ES access control. For more information, see the section called "About Access Policies on VPC Domains" (p. 24).

## Types of Policies

Amazon ES supports three types of access policies:

- the section called "Resource-based Policies" (p. 64)
- the section called "Identity-based Policies" (p. 65)
- the section called "IP-based Policies" (p. 66)

### Resource-based Policies

You add a resource-based policy, sometimes called the domain access policy, when you create a domain. These policies specify which actions a principal can perform on the domain's *subresources*. Subresources include Elasticsearch indices and APIs.

The Principal element specifies the accounts, users, or roles that are allowed access. The Resource element specifies which subresources these principals can access. The following resource-based policy grants `test-user` full access (`es:*`) to `test-domain`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:user/test-user"
        ]
      },
      "Action": [
        "es:*"
      ],
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/*"
    }
  ]
}
```

Two important considerations apply to this policy:

- These privileges apply only to this domain. Unless you create additional policies, `test-user` can't access data from other domains.
- The trailing `/*` in the `Resource` element is significant. Resource-based policies only apply to the domain's subresources, not the domain itself.

  For example, `test-user` can make requests against an index (`GET https://search-test-domain.us-west-1.es.amazonaws.com/test-index`), but can't update the domain's configuration (`POST https://es.us-west-1.amazonaws.com/2015-01-01/es/domain/test-domain/config`). Note the difference between the two endpoints. Accessing the configuration API (p. 236) requires an identity-based policy (p. 65).

To further restrict `test-user`, you can apply the following policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:user/test-user"
        ]
      },
      "Action": [
        "es:ESHttpGet"
      ],
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/test-index/_search"
    }
  ]
}
```

Now `test-user` can perform only one operation: searches against `test-index`. All other indices within the domain are inaccessible, and without permissions to use the `es:ESHttpPut` or `es:ESHttpPost` actions, `test-user` can't add or modify documents.

Next, you might decide to configure a role for power users. This policy gives `power-user-role` access to the HTTP GET and PUT methods for all URIs in the index:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:role/power-user-role"
        ]
      },
      "Action": [
        "es:ESHttpGet",
        "es:ESHttpPut"
      ],
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/test-index/*"
    }
  ]
}
```

For information about all available actions, see the section called "Policy Element Reference" (p. 69).

## Identity-based Policies

Unlike resource-based policies, which are a part of each Amazon ES domain, you attach identity-based policies to users or roles using the AWS Identity and Access Management (IAM) service. Just like resource-based policies (p. 64), identity-based policies specify who can access a service, which actions they can perform, and if applicable, the resources on which they can perform those actions.

While they certainly don't have to be, identity-based policies tend to be more generic. They often govern only the configuration API actions a user can perform. After you have these policies in place, you can use resource-based policies in Amazon ES to offer users access to Elasticsearch indices and APIs.

Because identity-based policies attach to users or roles (principals), the JSON doesn't specify a principal. The following policy grants access to actions that begin with `Describe` and `List`. This combination of actions provides read-only access to domain configurations, but not to the data stored in the domain itself:

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

An administrator might have full access to Amazon ES and all data stored on all domains:

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

For more information about the differences between resource-based and identity-based policies, see IAM Policies in the *IAM User Guide*.

> **Note**
> Users with the AWS-managed `AmazonESReadOnlyAccess` policy can't see cluster health status on the console. To allow them to see cluster health status, add the `"es:ESHttpGet"` action to an access policy and attach it to their accounts or roles.

## IP-based Policies

IP-based policies restrict access to a domain to one or more IP addresses or CIDR blocks. Technically, IP-based policies are not a distinct type of policy. Instead, they are just resource-based policies that specify an anonymous principal and include a special Condition element.

The primary appeal of IP-based policies is that they allow unsigned requests to an Amazon ES domain, which lets you use clients like curl and *Kibana* (p. 156) or access the domain through a proxy server. To learn more, see the section called "Using a Proxy to Access Amazon ES from Kibana" (p. 156).

> **Note**
> If you enabled VPC access for your domain, you can't configure an IP-based policy. Instead, you can use security groups to control which IP addresses can access the domain. For more information, see the section called "About Access Policies on VPC Domains" (p. 24).

The following policy grants all HTTP requests that originate from the specified IP range access to `test-domain`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "es:ESHttp*"
      ],
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": [
            "192.0.2.0/24"
          ]
        }
      },
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/*"
    }
  ]
}
```

If your domain has a public endpoint and doesn't use fine-grained access control (p. 76), we recommend combining IAM principals and IP addresses. This policy grants `test-user` HTTP access only if the request originates from the specified IP range:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "arn:aws:iam::987654321098:user/test-user"
      ]
    },
    "Action": [
      "es:ESHttp*"
    ],
    "Condition": {
      "IpAddress": {
        "aws:SourceIp": [
          "192.0.2.0/24"
        ]
      }
    },
    "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/*"
  }]
}
```

## Making and Signing Amazon ES Requests

Even if you configure a completely open resource-based access policy, *all* requests to the Amazon ES configuration API must be signed. If your policies specify IAM users or roles, requests to the Elasticsearch APIs also must be signed using AWS Signature Version 4. The signing method differs by API:

- To make calls to the Amazon ES configuration API, we recommend that you use one of the AWS SDKs. The SDKs greatly simplify the process and can save you a significant amount of time compared to creating and signing your own requests. The configuration API endpoints use the following format:

  ```
  es.region.amazonaws.com/2015-01-01/
  ```

  For example, the following request makes a configuration change to the `movies` domain, but you have to sign it yourself (not recommended):

  ```
  POST https://es.us-east-1.amazonaws.com/2015-01-01/es/domain/movies/config
  {
    "ElasticsearchClusterConfig": {
      "InstanceType": "c5.xlarge.elasticsearch"
    }
  }
  ```

  If you use one of the SDKs, such as Boto 3, the SDK automatically handles the request signing:

  ```
  import boto3

  client = boto3.client('es')
  response = client.update_elasticsearch_domain_config(
    DomainName='movies',
    ElasticsearchClusterConfig={
      'InstanceType': 'c5.xlarge.elasticsearch'
    }
  )
  ```

  For a Java code sample, see the section called "Using the AWS SDKs" (p. 121).
- To make calls to the Elasticsearch APIs, you must sign your own requests. For sample code in a variety of languages, see the section called "Signing HTTP Requests" (p. 113). The Elasticsearch APIs use the following format:

  ```
  domain-id.region.es.amazonaws.com
  ```

  For example, the following request searches the `movies` index for *thor*:

  ```
  GET https://my-domain.us-east-1.es.amazonaws.com/movies/_search?q=thor
  ```

  **Note**
  The service ignores parameters passed in URLs for HTTP POST requests that are signed with Signature Version 4.

## When Policies Collide

Complexities arise when policies disagree or make no explicit mention of a user. Understanding How IAM Works in the *IAM User Guide* provides a concise summary of policy evaluation logic:

- By default, all requests are denied.
- An explicit allow overrides this default.
- An explicit deny overrides any allows.

For example, if a resource-based policy grants you access to a domain, but an identity-based policy denies you access, you are denied access. If an identity-based policy grants access and a resource-based policy does not specify whether or not you should have access, you are allowed access. See the following table of intersecting policies for a full summary of outcomes.

| | **Allowed in Resource-based Policy** | **Denied in Resource-based Policy** | **Neither Allowed nor Denied in Resource-based Policy** |
|---|---|---|---|
| **Allowed in Identity-based Policy** | Allow | Deny | Allow |
| **Denied in Identity-based Policy** | Deny | Deny | Deny |
| **Neither Allowed nor Denied in Identity-based Policy** | Allow | Deny | Deny |

## Policy Element Reference

Amazon ES supports most policy elements in the IAM Policy Elements Reference, with the exception of `NotPrincipal`. The following table shows the most common elements.

| JSON Policy Element | Summary |
|---|---|
| `Version` | The current version of the policy language is `2012-10-17`. All access policies should specify this value. |
| `Effect` | This element specifies whether the statement allows or denies access to the specified actions. Valid values are `Allow` or `Deny`. |
| `Principal` | This element specifies the AWS account or IAM user or role that is allowed or denied access to a resource and can take several forms:<br>• **AWS accounts**: `"Principal":{"AWS": ["123456789012"]}` or `"Principal":{"AWS": ["arn:aws:iam::123456789012:root"]}`<br>• **IAM users**: `"Principal":{"AWS": ["arn:aws:iam::123456789012:user/test-user"]}`<br>• **IAM roles**: `"Principal":{"AWS": ["arn:aws:iam::123456789012:role/test-role"]}`<br><br>Specifying the `*` wildcard enables anonymous access to the domain, which we don't recommend unless you add an IP-based condition (p. 66), use VPC support (p. 21), or enable fine-grained access control (p. 76). |
| `Action` | Amazon ES uses the following actions for HTTP methods:<br>• `es:ESHttpDelete`<br>• `es:ESHttpGet`<br>• `es:ESHttpHead`<br>• `es:ESHttpPost`<br>• `es:ESHttpPut`<br>• `es:ESHttpPatch` |

| JSON Policy Element | Summary |
|---|---|
| | Amazon ES uses the following actions for the configuration API (p. 236):<br><br>• `es:AddTags`<br>• `es:CreateElasticsearchDomain`<br>• `es:CreateElasticsearchServiceRole`<br>• `es:DeleteElasticsearchDomain`<br>• `es:DeleteElasticsearchServiceRole`<br>• `es:DescribeElasticsearchDomain`<br>• `es:DescribeElasticsearchDomainConfig`<br>• `es:DescribeElasticsearchDomains`<br>• `es:DescribeElasticsearchInstanceTypeLimits`<br>• `es:DescribeReservedElasticsearchInstanceOfferings`<br>• `es:DescribeReservedElasticsearchInstances`<br>• `es:GetCompatibleElasticsearchVersions`<br>• `es:ListDomainNames`<br>• `es:ListElasticsearchInstanceTypeDetails`<br>• `es:ListElasticsearchInstanceTypes`<br>• `es:ListElasticsearchVersions`<br>• `es:ListTags`<br>• `es:PurchaseReservedElasticsearchInstanceOffering`<br>• `es:RemoveTags`<br>• `es:UpdateElasticsearchDomainConfig`<br><br>**Tip**<br>You can use wildcards to specify a subset of actions, such as `"Action":"es:*"` or `"Action":"es:Describe*"`.<br><br>Certain `es:` actions support resource-level permissions. For example, you can give a user permissions to delete one particular domain without giving that user permissions to delete *any* domain. Other actions apply only to the service itself. For example, `es:ListDomainNames` makes no sense in the context of a single domain and thus requires a wildcard.<br><br>**Important**<br>Resource-based policies differ from resource-level permissions. Resource-based policies (p. 64) are full JSON policies that attach to domains. Resource-level permissions let you restrict actions to particular domains or subresources. In practice, you can think of resource-level permissions as an optional part of a resource- or identity-based policy.<br><br>The following identity-based policy (p. 65) lists all `es:` actions and groups them according to whether they apply to the domain subresources (`test-domain/*`), to the domain configuration (`test-domain`), or only to the service (`*`):<br><br>`{`<br>`  "Version": "2012-10-17",`<br>`  "Statement": [`<br>`    {`<br>`      "Effect": "Allow",` |

| JSON Policy Element | Summary |
|---|---|

```
      "Action": [
        "es:ESHttpDelete",
        "es:ESHttpGet",
        "es:ESHttpHead",
        "es:ESHttpPost",
        "es:ESHttpPut",
        "es:ESHttpPatch"
      ],
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-
domain/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "es:CreateElasticsearchDomain",
        "es:DeleteElasticsearchDomain",
        "es:DescribeElasticsearchDomain",
        "es:DescribeElasticsearchDomainConfig",
        "es:DescribeElasticsearchDomains",
        "es:GetCompatibleElasticsearchVersions",
        "es:UpdateElasticsearchDomainConfig"
      ],
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-
domain"
    },
    {
      "Effect": "Allow",
      "Action": [
        "es:AddTags",
        "es:CreateElasticsearchServiceRole",
        "es:DeleteElasticsearchServiceRole",
        "es:DescribeElasticsearchInstanceTypeLimits",
        "es:DescribeReservedElasticsearchInstanceOfferings",
        "es:DescribeReservedElasticsearchInstances",
        "es:ListDomainNames",
        "es:ListElasticsearchInstanceTypeDetails",
        "es:ListElasticsearchInstanceTypes",
        "es:ListElasticsearchVersions",
        "es:ListTags",
        "es:PurchaseReservedElasticsearchInstanceOffering",
        "es:RemoveTags"
      ],
      "Resource": "*"
    }
  ]
}
```

**Note**
While resource-level permissions for `es:CreateElasticsearchDomain` might seem unintuitive —after all, why give a user permissions to create a domain that already exists?—the use of a wildcard lets you enforce a simple naming scheme for your domains, such as `"Resource": "arn:aws:es:us-west-1:987654321098:domain/my-team-name-*"`.

Of course, nothing prevents you from including actions alongside less restrictive resource elements, such as the following:

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

| JSON Policy Element | Summary |
|---|---|
| | `  "Statement": [`<br>`    {`<br>`      "Effect": "Allow",`<br>`      "Action": [`<br>`        "es:ESHttpGet",`<br>`        "es:DescribeElasticsearchDomain"`<br>`      ],`<br>`      "Resource": "*"`<br>`    }`<br>`  ]`<br>`}`<br><br>To learn more about pairing actions and resources, see the `Resource` element in this table. |
| `Condition` | Amazon ES supports most conditions that are described in Available Global Condition Keys in the *IAM User Guide*. One notable exception is the `aws:SecureTransport` key, which Amazon ES does not support.<br><br>When configuring an IP-based policy (p. 66), you specify the IP addresses or CIDR block as a condition, such as the following:<br><br>`"Condition": {`<br>`  "IpAddress": {`<br>`    "aws:SourceIp": [`<br>`      "192.0.2.0/32"`<br>`    ]`<br>`  }`<br>`}` |

| JSON Policy Element | Summary |
|---|---|
| `Resource` | Amazon ES uses `Resource` elements in three basic ways:<br><br>• For actions that apply to Amazon ES itself, like `es:ListDomainNames`, or to allow full access, use the following syntax:<br>`"Resource": "*"`<br><br>• For actions that involve a domain's configuration, like `es:DescribeElasticsearchDomain`, you can use the following syntax:<br>`"Resource": "arn:aws:es:region:aws-account-id:domain/domain-name"`<br><br>• For actions that apply to a domain's subresources, like `es:ESHttpGet`, you can use the following syntax:<br>`"Resource": "arn:aws:es:region:aws-account-id:domain/domain-name/*"`<br><br>You don't have to use a wildcard. Amazon ES lets you define a different access policy for each Elasticsearch index or API. For example, you might limit a user's permissions to the `test-index` index:<br>`"Resource": "arn:aws:es:region:aws-account-id:domain/domain-name/test-index"`<br><br>Instead of full access to `test-index`, you might prefer to limit the policy to just the search API:<br>`"Resource": "arn:aws:es:region:aws-account-id:domain/domain-name/test-index/_search"`<br><br>You can even control access to individual documents:<br>`"Resource": "arn:aws:es:region:aws-account-id:domain/domain-name/test-index/test-type/1"`<br><br>Essentially, if Elasticsearch expresses the subresource as a URI, you can control access to it using an access policy. For even more control over which resources a user can access, see the section called "Fine-Grained Access Control" (p. 76).<br><br>For details about which actions support resource-level permissions, see the `Action` element in this table. |

## Advanced Options and API Considerations

Amazon ES has several advanced options, one of which has access control implications: `rest.action.multi.allow_explicit_index`. At its default setting of true, it allows users to bypass subresource permissions under certain circumstances.

For example, consider the following resource-based policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:user/test-user"
        ]
      },
      "Action": [
        "es:ESHttp*"
      ],
      "Resource": [
        "arn:aws:es:us-west-1:987654321098:domain/test-domain/test-index/*",
        "arn:aws:es:us-west-1:987654321098:domain/test-domain/_bulk"
      ]
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:user/test-user"
        ]
      },
      "Action": [
        "es:ESHttpGet"
      ],
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/restricted-index/*"
    }
  ]
}
```

This policy grants `test-user` full access to `test-index` and the Elasticsearch bulk API. It also allows `GET` requests to `restricted-index`.

The following indexing request, as you might expect, fails due to a permissions error:

```
PUT https://search-test-domain.us-west-1.es.amazonaws.com/restricted-index/movie/1
{
  "title": "Your Name",
  "director": "Makoto Shinkai",
  "year": "2016"
}
```

Unlike the index API, the bulk API lets you create, update, and delete many documents in a single call. You often specify these operations in the request body, however, rather than in the request URL. Because Amazon ES uses URLs to control access to domain subresources, `test-user` can, in fact, use the bulk API to make changes to `restricted-index`. Even though the user lacks `POST` permissions on the index, the following request **succeeds**:

```
POST https://search-test-domain.us-west-1.es.amazonaws.com/_bulk
{ "index" : { "_index": "restricted-index", "_type" : "movie", "_id" : "1" } }
{ "title": "Your Name", "director": "Makoto Shinkai", "year": "2016" }
```

In this situation, the access policy fails to fulfill its intent. To prevent users from bypassing these kinds of restrictions, you can change `rest.action.multi.allow_explicit_index` to false. If this value is false, all calls to the bulk, mget, and msearch APIs that specify index names in the request body stop working. In other words, calls to `_bulk` no longer work, but calls to `test-index/_bulk` do. This second endpoint contains an index name, so you don't need to specify one in the request body.

Kibana (p. 156) relies heavily on mget and msearch, so it is unlikely to work properly after this change. For partial remediation, you can leave `rest.action.multi.allow_explicit_index` as true and deny certain users access to one or more of these APIs.

For information about changing this setting, see the section called “Advanced Options” (p. 13).

Similarly, the following resource-based policy contains two subtle issues:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/test-user"
      },
      "Action": "es:ESHttp*",
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/*"
    },
    {
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/test-user"
      },
      "Action": "es:ESHTTP*",
      "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-domain/restricted-index/*"
    }
  ]
}
```

- Despite the explicit deny, `test-user` can still make calls such as `GET https://search-test-domain.us-west-1.es.amazonaws.com/_all/_search` and `GET https://search-test-domain.us-west-1.es.amazonaws.com/*/_search` to access the documents in `restricted-index`.
- Because the `Resource` element references `restricted-index/*`, `test-user` doesn't have permissions to directly access the index's documents. The user does, however, have permissions to *delete the entire index*. To prevent access and deletion, the policy instead must specify `restricted-index*`.

Rather than mixing broad allows and focused denies, the safest approach is to follow the principle of least privilege and grant only the permissions that are required to perform a task. For more information about controlling access to individual indices or Elasticsearch operations, see the section called “Fine-Grained Access Control” (p. 76).

## Configuring Access Policies

- For instructions on creating or modifying resource- and IP-based policies in Amazon ES, see the section called “Configuring Access Policies” (p. 13).
- For instructions on creating or modifying identity-based policies in IAM, see Creating IAM Policies in the *IAM User Guide*.

## Additional Sample Policies

Although this chapter includes many sample policies, AWS access control is a complex subject that is best understood through examples. For more, see Example Policies in the *IAM User Guide*.

# Fine-Grained Access Control in Amazon Elasticsearch Service

Fine-grained access control offers additional ways of controlling access to your data on Amazon Elasticsearch Service. For example, depending on who makes the request, you might want a search to return results from only one index. You might want to hide certain fields in your documents or exclude certain documents altogether. Fine-grained access control offers the following features:

- Role-based access control
- Security at the index, document, and field level
- Kibana multi-tenancy
- HTTP basic authentication for Elasticsearch and Kibana

**Topics**

## The Bigger Picture: Fine-Grained Access Control and Amazon ES Security

Amazon Elasticsearch Service security has three main layers:

Network

The first security layer is the network, which determines whether requests reach an Amazon ES domain. If you choose **Public access** when you create a domain, requests from any internet-connected client can reach the domain endpoint. If you choose **VPC access**, clients must connect to the VPC (and the associated security groups must permit it) for a request to reach the endpoint. For more information, see the section called “VPC Support” (p. 21).

Domain access policy

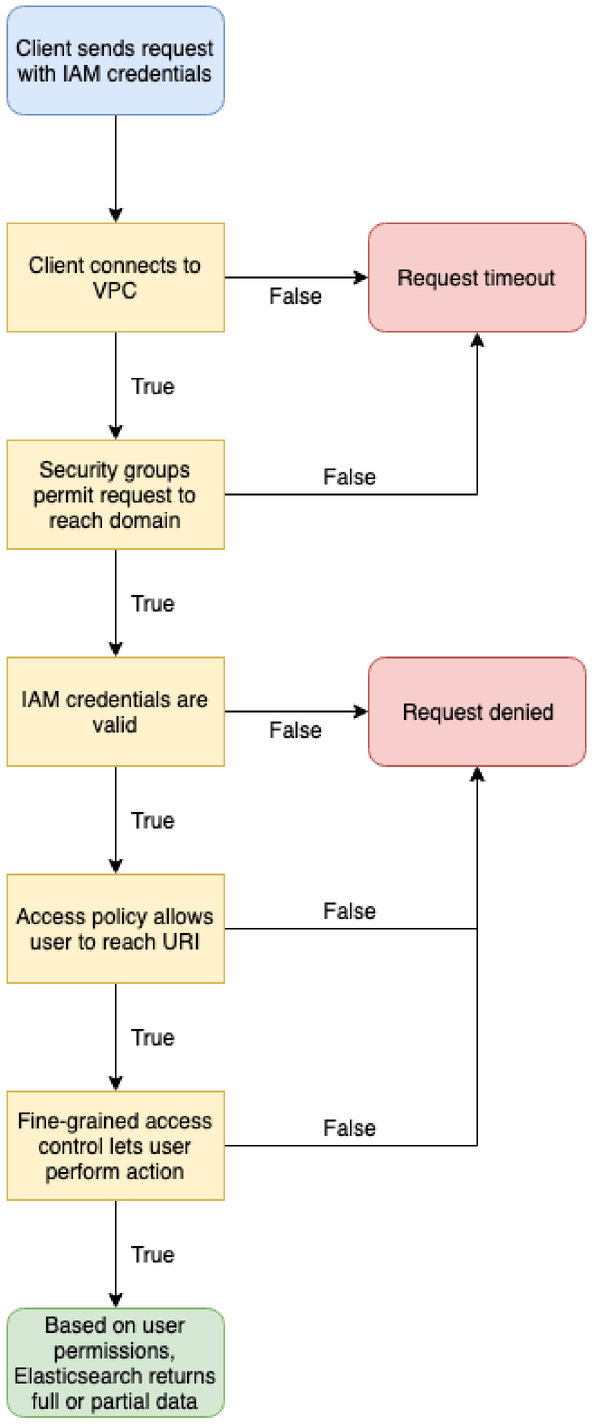
The second security layer is the domain access policy. After a request reaches a domain endpoint, the resource-based access policy (p. 64) allows or denies the request access to a given URI. The access policy accepts or rejects requests at the "edge" of the domain, before they reach Elasticsearch itself.

Fine-grained access control

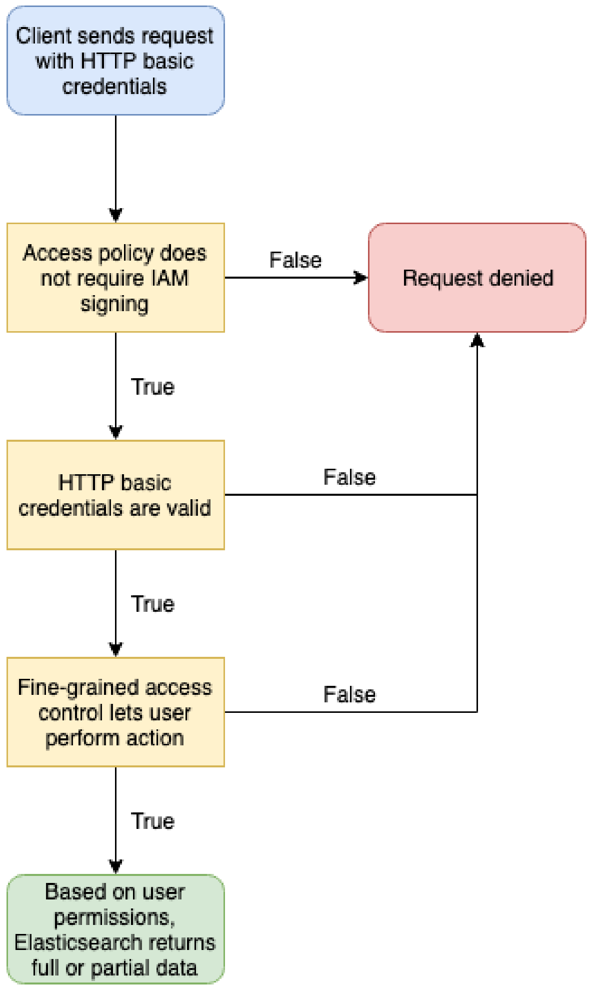
The third and final security layer is fine-grained access control. After a resource-based access policy allows a request to reach a domain endpoint, fine-grained access control evaluates the user credentials and either authenticates the user or denies the request. If fine-grained access control authenticates the user, it fetches all roles mapped to that user and uses the complete set of permissions to determine how to handle the request.

**Note**
If a resource-based access policy contains IAM users or roles, clients must send signed requests using AWS Signature Version 4. As such, access policies can conflict with fine-grained access control, especially if you use the internal user database and HTTP basic authentication. You can't sign a request with a user name and password *and* IAM credentials. In general, if you enable fine-grained access control, we recommend using a domain access policy that doesn't require signed requests.

This first diagram illustrates a common configuration: a VPC access domain with fine-grained access control enabled, an IAM-based access policy, and an IAM master user.


Client sends request with IAM credentials
Client connects to VPC
False
Request timeout
True
Security groups permit request to reach domain
False
True
IAM credentials are valid
False
Request denied
True
Access policy allows user to reach URI
False
True
Fine-grained access control lets user perform action
False
True
Based on user permissions, Elasticsearch returns full or partial data

This second diagram illustrates another common configuration: a public access domain with fine-grained access control enabled, an access policy that doesn't use IAM principals, and a master user in the internal user database.


Client sends request with HTTP basic credentials
Access policy does not require IAM signing
False
Request denied
True
HTTP basic credentials are valid
False
True
Fine-grained access control lets user perform action
False
True
Based on user permissions, Elasticsearch returns full or partial data

## Example

Consider a `GET` request to `movies/_search?q=thor`. Does the user have permissions to search the `movies` index? If so, does the user have permissions to see all documents within it? Should the response omit or anonymize any fields? For the master user, the response might look like this:

```
{
  "hits": {
    "total": 7,
    "max_score": 8.772789,
    "hits": [{
        "_index": "movies",
        "_type": "_doc",
        "_id": "tt0800369",
        "_score": 8.772789,
        "_source": {
          "directors": [
            "Kenneth Branagh",
            "Joss Whedon"
          ],
          "release_date": "2011-04-21T00:00:00Z",
          "genres": [
            "Action",
            "Adventure",
            "Fantasy"
          ],
          "plot": "The powerful but arrogant god Thor is cast out of Asgard to live amongst
 humans in Midgard (Earth), where he soon becomes one of their finest defenders.",
          "title": "Thor",
          "actors": [
            "Chris Hemsworth",
            "Anthony Hopkins",
            "Natalie Portman"
          ],
          "year": 2011
        }
      },
      ...
    ]
  }
}
```

If a user with more limited permissions issues the exact same request, the response might look like this:

```
{
  "hits": {
    "total": 2,
    "max_score": 8.772789,
    "hits": [{
        "_index": "movies",
        "_type": "_doc",
        "_id": "tt0800369",
        "_score": 8.772789,
        "_source": {
          "year": 2011,
          "release_date":
 "3812a72c6dd23eef3c750c2d99e205cbd260389461e19d610406847397ecb357",
          "plot": "The powerful but arrogant god Thor is cast out of Asgard to live amongst
 humans in Midgard (Earth), where he soon becomes one of their finest defenders.",
          "title": "Thor"
        }
      },
      ...
    ]
  }
}
```

The response has fewer hits and fewer fields for each hit. Also, the `release_date` field is anonymized. If a user with no permissions makes the same request, the cluster returns an error:

```
{
  "error": {
    "root_cause": [{
      "type": "security_exception",
      "reason": "no permissions for [indices:data/read/search] and User [name=limited-user,
 roles=[], requestedTenant=null]"
    }],
    "type": "security_exception",
    "reason": "no permissions for [indices:data/read/search] and User [name=limited-user,
 roles=[], requestedTenant=null]"
  },
  "status": 403
}
```

If a user provides invalid credentials, the cluster returns an `Unauthorized` exception.

## Key Concepts

*Roles* are the core way of using fine-grained access control. In this case, roles are distinct from IAM roles. Roles contain any combination of permissions: cluster-wide, index-specific, document level, and field level.

After configuring a role, you *map* it to one or more users. For example, you might map three roles to a single user: one role that provides access to Kibana, one that provides read-only access to `index1`, and one that provides write access to `index2`. Or you could include all of those permissions in a single role.

*Users* are people or applications that make requests to the Elasticsearch cluster. Users have credentials—either IAM access keys or a user name and password—that they specify when they make requests. With fine-grained access control on Amazon Elasticsearch Service, you choose one or the other for your *master user* when you configure your domain. The master user has full permissions to the cluster and manages roles and role mappings.

- If you choose IAM for your master user, all requests to the cluster must be signed using AWS Signature Version 4. For sample code, see the section called “Signing HTTP Requests” (p. 113).

  We recommend IAM if you want to use the same users on multiple clusters, if you want to use Amazon Cognito to access Kibana (with or without an external identity provider), or if you have Elasticsearch clients that support Signature Version 4 signing.
- If you choose the internal user database, you can use HTTP basic authentication (as well as IAM credentials) to make requests to the cluster. Most clients support basic authentication, including curl. The internal user database is stored in an Elasticsearch index, so you can't share it with other clusters.

  We recommend the internal user database if you don't need to reuse users across multiple clusters, if you want to use HTTP basic authentication to access Kibana (rather than Amazon Cognito), or if you have clients that only support basic authentication. The internal user database is the simplest way to get started with Amazon ES.

## Enabling Fine-Grained Access Control

Enable fine-grained access control using the console, AWS CLI, or configuration API. The console offers the simplest experience. For steps, see *Creating and Managing Amazon ES Domains* (p. 9). Here are the requirements for enabling fine-grained access control:

- Elasticsearch 6.7 or later
- Encryption of data at rest (p. 61) and node-to-node encryption (p. 63) enabled
- **Require HTTPS for all traffic to the domain** enabled

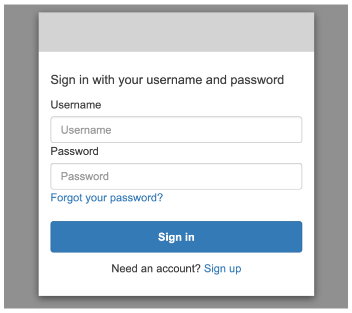
You can't enable fine-grained access control on existing domains, only new ones. After you enable fine-grained access control, you can't disable it.

## Accessing Kibana as the Master User

Fine-grained access control has a Kibana plugin that simplifies management tasks. You can use Kibana to manage users, roles, mappings, action groups, and tenants. The Kibana sign-in page and underlying authentication method differs, however, depending on how you configured your domain.

- If you choose to use IAM for user management, you must enable the section called “Authentication for Kibana” (p. 99) and sign in using credentials from your user pool to access Kibana. Otherwise, Kibana shows a nonfunctional sign-in page. See the section called “Limitations” (p. 90).

One of the assumed roles from the Amazon Cognito identity pool must match the IAM role that you specified for the master user. For more information about this configuration, see the section called "(Optional) Configuring Granular Access" (p. 105) and the section called "Tutorial: IAM Master User and Amazon Cognito" (p. 86).


Sign in with your username and password

Username

Username

Password

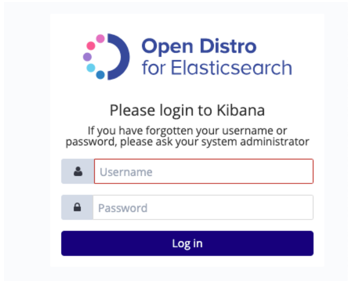
Password

Forgot your password?

Sign in

Need an account? Sign up

- If you choose to use the internal user database, you can sign in to Kibana with your master user name and password. You must access Kibana over HTTPS. For more information about this configuration, see the section called "Tutorial: Internal User Database and HTTP Basic Authentication" (p. 89).


Open Distro for Elasticsearch

Please login to Kibana

If you have forgotten your username or password, please ask your system administrator

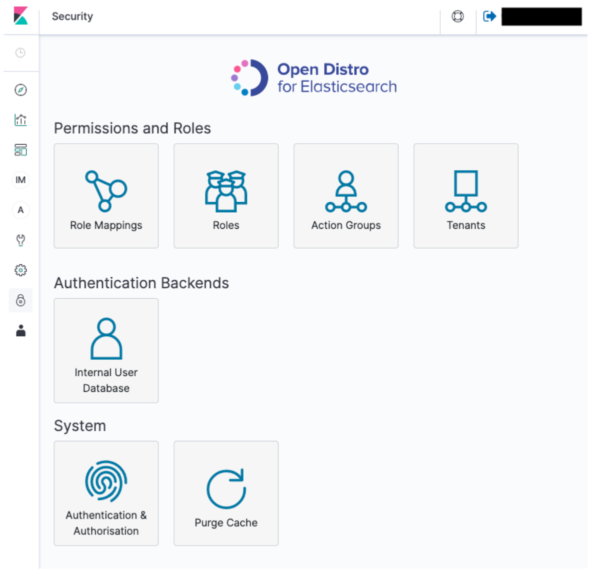
Username

Password

Log in

## Managing Permissions

As noted in the section called "Key Concepts" (p. 81), you manage fine-grained access control permissions using roles, users, and mappings. This section describes how to create and apply those resources. We recommend that you sign in to Kibana as the master user (p. 81) to perform these operations.



## Creating Roles

You can create new roles for fine-grained access control using Kibana or the `_opendistro/_security` operation in the REST API. For more information, see the Open Distro for Elasticsearch documentation.

Fine-grained access control also includes a number of predefined roles. Clients such as Kibana and Logstash make a wide variety of requests to Elasticsearch, which can make it hard to manually create roles with the minimum set of permissions. For example, the `kibana_user` role includes the permissions that a user needs to work with index patterns, visualizations, dashboards, and tenants. We recommend mapping it (p. 84) to any user or backend role that accesses Kibana, along with additional roles that allow access to other indices.

### Cluster-Level Security

Cluster-level permissions include the ability to execute broad requests such as `_mget`, `_msearch`, and `_bulk`, monitor health, take snapshots, and more. Manage these permissions using the **Cluster Permissions** tab when creating a role. For a list of cluster-level action groups, see the Open Distro for Elasticsearch documentation.

### Index-Level Security

Index-level permissions include the ability to create new indices, search indices, read and write documents, delete documents, manage aliases, and more. Manage these permissions using the **Index Permissions** tab when creating a role. For a list of index-level action groups, see the Open Distro for Elasticsearch documentation.

### Document-Level Security

Document-level security lets you restrict which documents in an index a user can see. When creating a role, specify an index pattern and an Elasticsearch query. Any users that you map to that role can see only the documents that match the query. Document-level security affects the number of hits that you receive when you search (p. 79).

For more information, see the Open Distro for Elasticsearch documentation.

### Field-Level Security

Field-level security lets you control which document fields a user can see. When creating a role, add a list of fields to either include or exclude. If you include fields, any users you map to that role can see only those fields. If you exclude fields, they can see all fields *except* the excluded ones. Field-level security affects the number of fields included in hits when you search (p. 79).

For more information, see the Open Distro for Elasticsearch documentation.

### Field Masking

Field masking is an alternative to field-level security that lets you anonymize the data in a field rather than remove it altogether. When creating a role, add a list of fields to mask. Field masking affects whether you can see the contents of a field when you search (p. 79).

## Creating Users

If you enabled the internal user database, you can create users using Kibana or the `_opendistro/_security` operation in the REST API. For more information, see the Open Distro for Elasticsearch documentation.

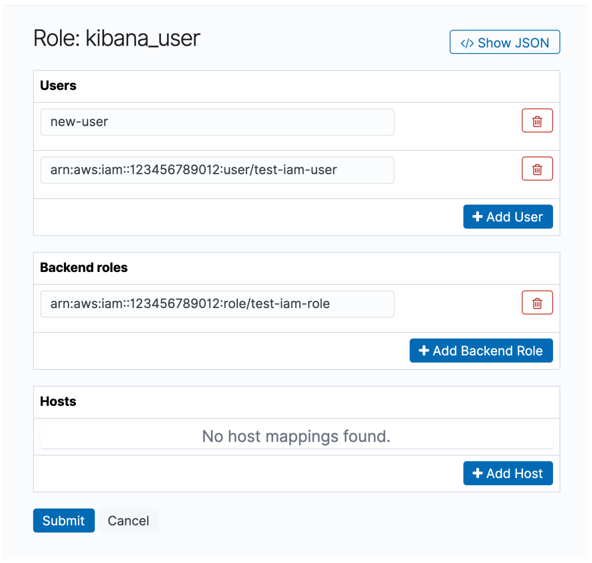
If you chose IAM for your master user, ignore this portion of Kibana. Create IAM users and IAM roles instead. For more information, see the IAM User Guide.

## Mapping Roles to Users

Role mapping is the most critical aspect of fine-grained access control. Fine-grained access control has some predefined roles to help you get started, but unless you map roles to users, every request to the cluster ends in a permissions error.

*Backend roles* offer another way of mapping roles to users. Rather than mapping the same role to dozens of different users, you can map the role to a single backend role, and then make sure that all users have that backend role. Backend roles can be IAM roles or arbitrary strings that you specify when you create users in the internal user database.

- Specify users, IAM user ARNs, and Amazon Cognito user strings in the **Users** section. Cognito user strings take the form of `Cognito/user-pool-id/username`.
- Specify backend roles and IAM role ARNs in the **Backend roles** section.



You can map roles to users using Kibana or the `_opendistro/_security` operation in the REST API. For more information, see the Open Distro for Elasticsearch documentation.

## Creating Action Groups

Action groups are sets of permissions that you can reuse across different resources. You can create new action groups using Kibana or the `_opendistro/_security` operation in the REST API, although the default action groups suffice for most use cases. For more information about the default action groups, see the Open Distro for Elasticsearch documentation.

## Kibana Multi-Tenancy

Tenants are spaces for saving index patterns, visualizations, dashboards, and other Kibana objects. Kibana multi-tenancy lets you safely share your work with other Kibana users (or keep it private). You can control which roles have access to a tenant and whether those roles have read or write access. To learn more, see the Open Distro for Elasticsearch documentation.

### To view your current tenant or change tenants

1. Navigate to Kibana and sign in.

2. Choose **Tenants**.
3. Verify your tenant before creating visualizations or dashboards. If you want to share your work with all other Kibana users, choose **Global**. To share your work with a subset of Kibana users, choose a different shared tenant. Otherwise, choose **Private**.

## Recommended Configurations

Due to how fine-grained access control interacts with other security features (p. 76), we recommend several fine-grained access control configurations that work well for most use cases.

| Description | Master User | Amazon Cognito Authentication for Kibana | Domain Access Policy |
|---|---|---|---|
| Use IAM credentials or basic authentication for calls to the Elasticsearch APIs, and use basic authentication to access Kibana. Manage fine-grained access control roles using Kibana or the REST API. | User name and password | Disabled | `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "AWS": "*" }, "Action": "es:ESHttp*", "Resource": "domain-arn/*" } ] }` |
| Use IAM credentials for calls to the Elasticsearch APIs, and use Amazon Cognito to access Kibana. Manage fine-grained access control roles using Kibana or the REST API. | IAM user or role | Enabled | `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "AWS": "*" }, "Action": "es:ESHttp*", "Resource": "domain-arn/*" } ] }` |
| Use IAM credentials for calls to the Elasticsearch APIs, and block most access to Kibana. Manage fine-grained access control | IAM user or role | Disabled | `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": {` |

| Description | Master User | Amazon Cognito Authentication for Kibana | Domain Access Policy |
|---|---|---|---|
| roles using the REST API. | | | `"AWS": "*"`<br>`},`<br>`"Action":`<br>`"es:ESHttp*",`<br>`"Resource":`<br>`"domain-arn/*"`<br>`},`<br>`{`<br>`"Effect":`<br>`"Deny",`<br>`"Principal": {`<br>`"AWS": "*"`<br>`},`<br>`"Action":`<br>`"es:ESHttp*",`<br>`"Resource":`<br>`"domain-arn/_plugin/`<br>`kibana*"`<br>`}`<br>`]`<br>`}` |

## Tutorial: IAM Master User and Amazon Cognito

This tutorial covers a popular use case: an IAM master user with Amazon Cognito authentication for Kibana. Although these steps use the Amazon Cognito user pool for authentication, this same basic process works for any Cognito authentication provider that lets you assign different IAM roles to different users (SAML, for example).

> **Note**
> This tutorial assumes you have two existing IAM roles, one for the master user and one for more limited users. If you don't have two roles, create them.

### To get started with fine-grained access control

1. Create a domain (p. 9) with the following settings:

   - Elasticsearch 7.4
   - Public access
   - Fine-grained access control enabled with an IAM role as the master user (`IAMMasterUserRole` for the rest of this tutorial)
   - Amazon Cognito authentication for Kibana (p. 99) enabled
   - The following access policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "*"
        ]
      },
      "Action": [
        "es:ESHttp*"
      ],
      "Resource": "arn:aws:es:region:account:domain/domain-name/*"
    }
  ]
}
```

- HTTPS required for all traffic to the domain
- Node-to-node encryption
- Encryption of data at rest

2. Navigate to the IAM console, and then choose **Roles**.
3. Choose `IAMMasterUserRole`, and then choose the **Trust relationships** tab.
4. Choose **Edit trust relationship**, and ensure that the Amazon Cognito identity pool can assume the role. You should see the following statement:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Federated": "cognito-identity.amazonaws.com"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {
        "cognito-identity.amazonaws.com:aud": "identity-pool-id"
      },
      "ForAnyValue:StringLike": {
        "cognito-identity.amazonaws.com:amr": "authenticated"
      }
    }
  }]
}
```

5. Choose **Update Trust Policy**.
6. Add the same trust policy to a second IAM role (`IAMLimitedUserRole` for the rest of this tutorial).
7. Navigate to the Amazon Cognito console, and then choose **Manage User Pools**.
8. Choose your user pool, and then choose **Users and groups**.
9. Choose **Create user**, specify a user name of `master-user` and a password, and then choose **Create user**.
10. Create another user named `limited-user`.
11. Choose the **Groups** tab, and then choose **Create group**.
12. Name the group `master-user-group`, choose `IAMMasterUserRole` in the **IAM role** dropdown list, and then choose **Create group**.
13. Create another group named `limited-user-group` that uses `IAMLimitedUserRole`.
14. Choose `master-user-group`, choose **Add users**, and then add `master-user`.
15. Choose `limited-user-group`, choose **Add users**, and then add `limited-user`.
16. Choose **App client settings** and note the app client ID for your domain.
17. Choose **Federated Identities**, choose your identity pool, and then choose **Edit identity pool**.
18. Expand **Authentication providers**, find your user pool ID and the app client ID for your domain, and then change **Use default role** to **Choose role from token**.
19. For **Role resolution**, choose **DENY**. With this setting, users must be in a group to receive an IAM role after authenticating.
20. Choose **Save Changes**.

21. Navigate to Kibana.
22. Sign in with `master-user`.
23. Choose **Try our sample data**.
24. Add the sample flight data.
25. Choose **Security**, **Roles**, **Add a new role**.
26. Name the role `new-role`, and then choose **Index Permissions**.
27. Choose **Add index permissions**, and then specify `kibana_sample_data_fli*` for the index pattern.
28. Choose **Add Action**, **read**.
29. For **Document Level Security Query**, specify the following query:

    ```
    {
      "match": {
        "FlightDelay": true
      }
    }
    ```

    Then choose **Test DLS query syntax**.
30. For **Include or exclude fields**, choose **Exclude fields**, and then choose **Add Field**. Specify `FlightNum`.
31. For **Anonymize fields**, choose **Add Field**. Specify `Dest`.
32. Choose **Save Role Definition**.
33. Choose **Back**, **Role Mappings**, **Add a new role mapping**.
34. For **Role**, choose **new-role**. Choose **Add Backend Role**, and specify the ARN for `IAMLimitedUserRole`. Then choose **Submit**.
35. Choose **Add a new role mapping** again.
36. For **Role**, choose **kibana_user**. Choose **Add Backend Role**, and specify the ARN for `IAMLimitedUserRole`. Then choose **Submit**.
37. In a new, private browser window, navigate to Kibana, sign in using `limited-user`, and then choose **Explore on my own**.
38. Choose **Dev Tools**, and then run the default search:

    ```
    GET _search
    {
      "query": {
        "match_all": {}
      }
    }
    ```

    Note the permissions error. `limited-user` doesn't have permissions to run cluster-wide searches.
39. Run another search:

    ```
    GET kibana_sample_data_flights/_search
    {
      "query": {
        "match_all": {}
      }
    }
    ```

    Note that all matching documents have a `FlightDelay` field of `true`, an anonymized `Dest` field, and no `FlightNum` field.

40. In your original browser window, signed in as `master-user`, choose **Dev Tools**, and then perform the same searches. Note the difference in permissions, number of hits, matching documents, and included fields.

## Tutorial: Internal User Database and HTTP Basic Authentication

This tutorial covers another popular use case: a master user in the internal user database and HTTP basic authentication for Kibana.

**To get started with fine-grained access control**

1. Create a domain (p. 9) with the following settings:

    - Elasticsearch 7.4
    - Public access
    - Fine-grained access control with a master user in the internal user database (`TheMasterUser` for the rest of this tutorial)
    - Amazon Cognito authentication for Kibana *disabled*
    - The following access policy:

    ```
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "AWS": [
              "*"
            ]
          },
          "Action": [
            "es:ESHttp*"
          ],
          "Resource": "arn:aws:es:region:account:domain/domain-name/*"
        }
      ]
    }
    ```

    - HTTPS required for all traffic to the domain
    - Node-to-node encryption
    - Encryption of data at rest
2. Navigate to Kibana.
3. Sign in using `TheMasterUser`.
4. Choose **Try our sample data**.
5. Add the sample flight data.
6. Choose **Security**, **Internal User Database**, **Add a new internal user**.
7. Name the user `new-user`, specify a password, and give the user the backend role of `new-backend-role`. Then choose **Submit**.
8. Choose **Back**, **Roles**, **Add a new role**.
9. Name the role `new-role`, and then choose **Index Permissions**.
10. Choose **Add index permissions**, and then specify `kibana_sample_data_fli*` for the index pattern.

11. Choose **Add Action Group**, **read**.
12. For **Document Level Security Query**, specify the following query:

    ```
    {
      "match": {
        "FlightDelay": true
      }
    }
    ```

    Then choose **Test DLS query syntax**.
13. For **Include or exclude fields**, choose **Exclude fields**, and then choose **Add Field**. Specify `FlightNum`.
14. For **Anonymize fields**, choose **Add Field**. Specify `Dest`.
15. Choose **Save Role Definition**.
16. Choose **Back**, **Role Mappings**, **Add a new role mapping**.
17. For **Role**, choose **new-role**. Choose **Add Backend Role**, and then specify `new-backend-role`. Then choose **Submit**.
18. Choose **Add a new role mapping** again.
19. For **Role**, choose `kibana_user`. Choose **Add User** and specify `new-user`. Then choose **Submit**.

    Only `new-user` has the `kibana_user` role, but all users with the `new-backend-role` backend role have the `new-role` role.
20. In a new, private browser window, navigate to Kibana, sign in using `new-user`, and then choose **Explore on my own**.
21. Choose **Dev Tools** and run the default search:

    ```
    GET _search
    {
      "query": {
        "match_all": {}
      }
    }
    ```

    Note the permissions error. `new-user` doesn't have permissions to run cluster-wide searches.
22. Run another search:

    ```
    GET kibana_sample_data_flights/_search
    {
      "query": {
        "match_all": {}
      }
    }
    ```

    Note that all matching documents have a `FlightDelay` field of `true`, an anonymized `Dest` field, and no `FlightNum` field.
23. In your original browser window, signed in as `TheMasterUser`, choose **Dev Tools** and perform the same searches. Note the difference in permissions, number of hits, matching documents, and included fields.

## Limitations

Fine-grained access control has several important limitations:

- The `hosts` aspect of role mappings, which maps roles to hostnames or IP addresses, doesn't work if the domain is within a VPC. You can still map roles to users and backend roles.
- Users in the internal user database can't change their own passwords. Master users (or users with equivalent permissions) must change their passwords for them.
- If you choose IAM for the master user and don't enable Amazon Cognito authentication, Kibana displays a nonfunctional sign-in page.
- If you choose IAM for the master user, you can still create users in the internal user database. Because HTTP basic authentication is not enabled under this configuration, however, any requests signed with those user credentials are rejected.
- If you use SQL (p. 153) to query an index that you don't have access to, you receive a "no permissions" error. If the index doesn't exist, you receive a "no such index" error. This difference in error messages means that you can confirm the existence of an index if you happen to guess its name.

  To minimize the issue, don't include sensitive information in index names (p. 130). To deny all access to SQL, add the following element to your domain access policy:

```
{
  "Effect": "Deny",
  "Principal": {
    "AWS": [
      "*"
    ]
  },
  "Action": [
    "es:*"
  ],
  "Resource": "arn:aws:es:us-east-1:123456789012:domain/my-domain/_opendistro/_sql"
}
```

## Modifying the Master User

If you forget the details of the master user, you can reconfigure it using the console, AWS CLI, or configuration API.

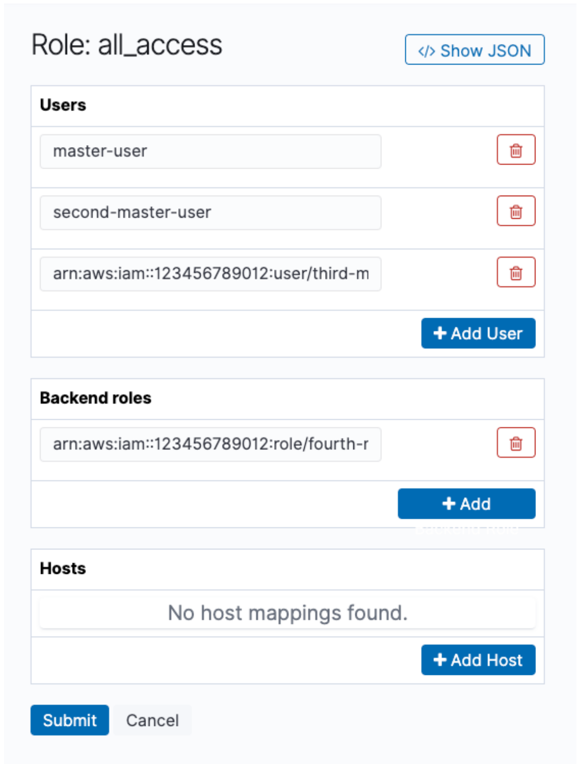
### To modify the master user (console)

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. Choose your domain.
4. Choose **Actions**, **Modify master user**.
5. Choose either **Set IAM role as master user** or **Create new master user**.
   - If you previously used an IAM master user, fine-grained access control re-maps the `all_access` role to the new IAM ARN that you specify.
   - If you previously used the internal user database, fine-grained access control creates a new master user. You can use the new master user to delete the old one.
6. Choose **Submit**.

## Additional Master Users

You designate a master user when you create a domain, but if you want, you can use this master user to create additional master users. You have two options: Kibana or the REST API.

- In Kibana, choose **Security**, **Role Mappings**, and then map the new master user to the `all_access` and `security_manager` roles.



Role: all_access

Users

master-user

second-master-user

arn:aws:iam::123456789012:user/third-m

Backend roles

arn:aws:iam::123456789012:role/fourth-r

Hosts

No host mappings found.

- To use the REST API, send the following requests:

```
PUT _opendistro/_security/api/rolesmapping/all_access
{
  "backend_roles": [
    "arn:aws:iam::123456789012:role/fourth-master-user"
  ],
  "hosts": [],
  "users": [
    "master-user",
    "second-master-user",
    "arn:aws:iam::123456789012:user/third-master-user"
  ]
}
```

```
PUT _opendistro/_security/api/rolesmapping/security_manager
{
  "backend_roles": [
    "arn:aws:iam::123456789012:role/fourth-master-user"
  ],
  "hosts": [],
  "users": [
    "master-user",
    "second-master-user",
    "arn:aws:iam::123456789012:user/third-master-user"
  ]
}
```

These requests *replace* the current role mappings, so perform `GET` requests first so that you can include all current roles in the `PUT` requests. The REST API is especially useful if you can't access Kibana and want to map an IAM role from Amazon Cognito to the `all_access` role.

## Manual Snapshots

Fine-grained access control introduces some additional complications with taking manual snapshots. To register a snapshot repository—even if you use HTTP basic authentication for all other purposes—you must map the `manage_snapshots` role to an IAM role that has `iam:PassRole` permissions to assume `TheSnapshotRole`, as defined in the section called “Manual Snapshot Prerequisites” (p. 45).

Then use that IAM role to send a signed request to the domain, as outlined in the section called “Registering a Manual Snapshot Repository” (p. 46).

## Integrations

If you use other AWS services (p. 132) with Amazon ES, you must provide the IAM roles for those services with appropriate permissions. For example, Kinesis Data Firehose delivery streams often use an IAM role called `firehose_delivery_role`. In Kibana, create a role for fine-grained access control (p. 83), and map the IAM role to it (p. 84). In this case, the new role needs the following permissions:

```
{
  "cluster_permissions": [
    "cluster_composite_ops",
    "cluster_monitor"
  ],
  "index_permissions": [{
    "index_patterns": [
      "firehose-index*"
    ],
    "allowed_actions": [
      "create_index",
      "manage",
      "crud"
    ]
  }]
}
```

Permissions vary based on the actions each service performs. An AWS IoT rule or AWS Lambda function that indexes data likely needs similar permissions to Kinesis Data Firehose, while a Lambda function that only performs searches can use a more limited set.

## REST API Differences

The fine-grained access control REST API differs slightly depending on your Elasticsearch version. Prior to making a `PUT` request, make a `GET` request to verify the expected request body. For example, a `GET` request to `_opendistro/_security/api/user` returns all users, which you can then modify and use to make valid `PUT` requests.

On Elasticsearch 6.*x*, requests to create users look like this:

```
PUT _opendistro/_security/api/user/new-user
{
  "password": "some-password",
  "roles": ["new-backend-role"]
}
```

On Elasticsearch 7.x, requests look like this:

```
PUT _opendistro/_security/api/user/new-user
{
  "password": "some-password",
  "backend_roles": ["new-backend-role"]
}
```

Further, tenants are properties of roles in Elasticsearch 6.*x*:

```
GET _opendistro/_security/api/roles/all_access

{
  "all_access": {
    "cluster": ["UNLIMITED"],
    "tenants": {
      "admin_tenant": "RW"
    },
    "indices": {
      "*": {
        "*": ["UNLIMITED"]
      }
    },
    "readonly": "true"
  }
}
```

In Elasticsearch 7.x, they are objects with their own URI:

```
GET _opendistro/_security/api/tenants

{
  "global_tenant": {
    "reserved": true,
    "hidden": false,
    "description": "Global tenant",
    "static": false
  }
}
```

For documentation on the 7.*x* REST API, see the Open Distro for Elasticsearch documentation.

**Tip**
If you use the internal user database, you can use curl to make requests and test your domain. Try the following sample commands:

```
curl -XGET -u master-user:master-user-password 'domain-endpoint/_search'
curl -XGET -u master-user:master-user-password 'domain-endpoint/_opendistro/
_security/api/user'
```

# Logging and Monitoring in Amazon Elasticsearch Service

Amazon Elasticsearch Service integrates with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon ES. CloudTrail captures all configuration API calls for Amazon ES as events.

**Note**
CloudTrail only captures calls to the configuration API (p. 236), such as `CreateElasticsearchDomain` and `GetUpgradeStatus`. CloudTrail doesn't capture calls to the Elasticsearch APIs (p. 183), such as `_search` and `_bulk`.

The captured calls include calls from the Amazon ES console, AWS CLI, or an AWS SDK. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon ES. If you don't configure a trail, you can still view the most recent events on the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon ES, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

## Amazon Elasticsearch Service Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon ES, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Amazon ES, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All Amazon ES configuration API actions are logged by CloudTrail and are documented in the *Amazon ES Configuration API Reference* (p. 236).

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 or AWS Identity and Access Management (IAM) user credentials
- Whether the request was made with temporary security credentials for a role or federated user
- Whether the request was made by another AWS service

For more information, see the CloudTrail userIdentity Element.

## Understanding Amazon Elasticsearch Service Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `CreateElasticsearchDomain` operation:

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/test-user",
    "accountId": "123456789012",
    "accessKeyId": "access-key",
    "userName": "test-user",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2018-08-21T21:59:11Z"
      }
    },
    "invokedBy": "signin.amazonaws.com"
  },
  "eventTime": "2018-08-21T22:00:05Z",
  "eventSource": "es.amazonaws.com",
  "eventName": "CreateElasticsearchDomain",
  "awsRegion": "us-west-1",
  "sourceIPAddress": "123.123.123.123",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
    "elasticsearchVersion": "6.3",
    "elasticsearchClusterConfig": {
      "instanceType": "m4.large.elasticsearch",
      "instanceCount": 1
    },
    "snapshotOptions": {
      "automatedSnapshotStartHour": 0
    },
    "domainName": "test-domain",
    "encryptionAtRestOptions": {},
    "eBSOptions": {
      "eBSEnabled": true,
      "volumeSize": 10,
      "volumeType": "gp2"
    },
    "accessPolicies": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow
\",\"Principal\":{\"AWS\":[\"123456789012\"]},\"Action\":[\"es:*\"],\"Resource\":
\"arn:aws:es:us-west-1:123456789012:domain/test-domain/*\"}]}",
    "advancedOptions": {
      "rest.action.multi.allow_explicit_index": "true"
    }
  },
  "responseElements": {
    "domainStatus": {
      "created": true,
      "elasticsearchClusterConfig": {
        "zoneAwarenessEnabled": false,
        "instanceType": "m4.large.elasticsearch",
        "dedicatedMasterEnabled": false,
        "instanceCount": 1
      },
      "cognitoOptions": {
        "enabled": false
      },
      "encryptionAtRestOptions": {
        "enabled": false
      },
      "advancedOptions": {
        "rest.action.multi.allow_explicit_index": "true"
      },
      "upgradeProcessing": false,
      "snapshotOptions": {
        "automatedSnapshotStartHour": 0
      },
      "eBSOptions": {
        "eBSEnabled": true,
        "volumeSize": 10,
        "volumeType": "gp2"
      },
      "elasticsearchVersion": "6.3",
      "processing": true,
      "aRN": "arn:aws:es:us-west-1:123456789012:domain/test-domain",
      "domainId": "123456789012/test-domain",
      "deleted": false,
      "domainName": "test-domain",
      "accessPolicies": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"AWS\":\"arn:aws:iam::123456789012:root\"},\"Action\":\"es:*\",\"Resource
\":\"arn:aws:es:us-west-1:123456789012:domain/test-domain/*\"}]}"
    }
  },
  "requestID": "12345678-1234-1234-1234-987654321098",
  "eventID": "87654321-4321-4321-4321-987654321098",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

# Compliance Validation for Amazon Elasticsearch Service

Third-party auditors assess the security and compliance of Amazon Elasticsearch Service as part of multiple AWS compliance programs. These include SOC, PCI, and HIPAA.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using Amazon ES is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Config – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

## Resilience in Amazon Elasticsearch Service

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

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

In addition to the AWS global infrastructure, Amazon ES offers several features to help support your data resiliency and backup needs:

- Multi-AZ domains and replica shards (p. 17)
- Automated and manual snapshots (p. 44)

## Infrastructure Security in Amazon Elasticsearch Service

As a managed service, Amazon Elasticsearch Service is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access the Amazon ES configuration API through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests to the configuration API 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 AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Depending on your domain configuration, you might also need to sign requests to the Elasticsearch APIs. For more information, see the section called “Making and Signing Amazon ES Requests” (p. 67).

Amazon ES supports public access domains, which can receive requests from any internet-connected device, and VPC access domains (p. 21), which are isolated from the public internet.

# Amazon Cognito Authentication for Kibana

Amazon Elasticsearch Service uses Amazon Cognito to offer user name and password protection for Kibana (p. 156). This authentication feature is optional and available only for domains using Elasticsearch 5.1 or later. If you don't configure Amazon Cognito authentication, you can still protect Kibana using an IP-based access policy (p. 66) and a proxy server (p. 156).

Much of the authentication process occurs in Amazon Cognito, but this section offers guidelines and requirements for configuring Amazon Cognito resources to work with Amazon ES domains. Standard pricing applies to all Amazon Cognito resources.

**Tip**
The first time that you configure a domain to use Amazon Cognito authentication for Kibana, we recommend using the console. Amazon Cognito resources are extremely customizable, and the console can help you identify and understand the features that matter to you.

**Topics**

## Prerequisites

Before you can configure Amazon Cognito authentication for Kibana, you must fulfill several prerequisites. The Amazon ES console helps streamline the creation of these resources, but understanding the purpose of each resource helps with configuration and troubleshooting. Amazon Cognito authentication for Kibana requires the following resources:

- Amazon Cognito user pool
- Amazon Cognito identity pool
- IAM role that has the `AmazonESCognitoAccess` policy attached (`CognitoAccessForAmazonES`)

**Note**
The user pool and identity pool must be in the same AWS Region. You can use the same user pool, identity pool, and IAM role to add Amazon Cognito authentication for Kibana to multiple Amazon ES domains. To learn more, see the section called "Limits" (p. 107).

### About the User Pool

User pools have two main features: create and manage a directory of users, and let users sign up and log in. For instructions about creating a user pool, see Create a User Pool in the *Amazon Cognito Developer Guide*.

When you create a user pool to use with Amazon ES, consider the following:

- Your Amazon Cognito user pool must have a domain name. Amazon ES uses this domain name to redirect users to a login page for accessing Kibana. Other than a domain name, the user pool doesn't require any non-default configuration.
- You must specify the pool's required standard attributes—attributes like name, birth date, email address, and phone number. You can't change these attributes after you create the user pool, so choose the ones that matter to you at this time.
- While creating your user pool, choose whether users can create their own accounts, the minimum password strength for accounts, and whether to enable multi-factor authentication. If you plan to use an external identity provider, these settings are inconsequential. Technically, you can enable the user pool as an identity provider *and* enable an external identity provider, but most people prefer one or the other.

User pool IDs take the form of `region_ID`. If you plan to use the AWS CLI or an AWS SDK to configure Amazon ES, make note of the ID.

## About the Identity Pool

Identity pools let you assign temporary, limited-privilege roles to users after they log in. For instructions about creating an identity pool, see Identity Pools in the *Amazon Cognito Developer Guide*. When you create an identity pool to use with Amazon ES, consider the following:

- If you use the Amazon Cognito console, you must select the **Enable access to unauthenticated identities** check box to create the identity pool. After you create the identity pool and configure the Amazon ES domain (p. 101), Amazon Cognito disables this setting.
- You don't need to add external identity providers to the identity pool. When you configure Amazon ES to use Amazon Cognito authentication, it configures the identity pool to use the user pool that you just created.
- After you create the identity pool, you must choose unauthenticated and authenticated IAM roles. These roles specify the access policies that users have before and after they log in. If you use the Amazon Cognito console, it can create these roles for you. After you create the authenticated role, make note of the ARN, which takes the form of `arn:aws:iam::123456789012:role/Cognito_identitypoolAuth_Role`.

Identity pool IDs take the form of `region:ID-ID-ID-ID-ID`. If you plan to use the AWS CLI or an AWS SDK to configure Amazon ES, make note of the ID.

## About the CognitoAccessForAmazonES Role

Amazon ES needs permissions to configure the Amazon Cognito user and identity pools and use them for authentication. You can use `AmazonESCognitoAccess`, which is an AWS managed policy, for this purpose. If you use the console to create or configure your Amazon ES domain, it creates an IAM role for you and attaches this policy to the role. The default name for this role is `CognitoAccessForAmazonES`.

If you use the AWS CLI or one of the AWS SDKs, you must create your own role, attach the policy, and specify the ARN for this role when you configure your Amazon ES domain. The role must have the following trust relationship:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "es.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

For instructions, see Creating a Role to Delegate Permissions to an AWS Service and Attaching and Detaching IAM Policies in the *IAM User Guide*.

# Configuring an Amazon ES Domain

After you complete the prerequisites, you can configure an Amazon ES domain to use Amazon Cognito for Kibana.

> **Note**
> Amazon Cognito is not available in all AWS Regions. For a list of supported regions, see AWS Regions and Endpoints. You don't need to use the same Region for Amazon Cognito that you use for Amazon ES.

## Configuring Amazon Cognito Authentication (Console)

Because it creates the CognitoAccessForAmazonES (p. 100) role for you, the console offers the simplest configuration experience. In addition to the standard Amazon ES permissions, you need the following set of permissions to use the console to create a domain that uses Amazon Cognito authentication for Kibana:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole",
        "iam:CreateRole",
        "iam:AttachRolePolicy",
        "ec2:DescribeVpcs",
        "cognito-identity:ListIdentityPools",
        "cognito-idp:ListUserPools"
      ],
      "Resource": "*"
    }
  ]
}
```

If CognitoAccessForAmazonES (p. 100) already exists, you need fewer permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcs",
        "cognito-identity:ListIdentityPools",
        "cognito-idp:ListUserPools"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/CognitoAccessForAmazonES"
    }
  ]
}
```

**To configure Amazon Cognito authentication for Kibana (console)**

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. In the navigation pane, under **My domains**, choose the domain that you want to configure.
4. Choose **Edit domain**.
5. For **Amazon Cognito authentication**, choose **Enable Amazon Cognito authentication**.
6. For **Region**, select the Region that contains your Amazon Cognito user pool and identity pool.
7. For **Cognito User Pool**, select a user pool or create one. For guidance, see the section called "About the User Pool" (p. 99).
8. For **Cognito Identity Pool**, select an identity pool or create one. For guidance, see the section called "About the Identity Pool" (p. 100).

   **Note**
   The **Create new user pool** and **Create new identity pool** links direct you to the Amazon Cognito console and require you to create these resources manually. The process is not automatic. To learn more, see the section called "Prerequisites" (p. 99).
9. For **IAM Role**, use the default value of `CognitoAccessForAmazonES` (recommended) or enter a new name. To learn more about the purpose of this role, see the section called "About the CognitoAccessForAmazonES Role" (p. 100).
10. Choose **Submit**.

After your domain finishes processing, see the section called "Allowing the Authenticated Role" (p. 103) and the section called "Configuring Identity Providers" (p. 103) for additional configuration steps.

## Configuring Amazon Cognito Authentication (AWS CLI)

Use the `--cognito-options` parameter to configure your Amazon ES domain. The following syntax is used by both the `create-elasticsearch-domain` and `update-elasticsearch-domain-config` commands:

```
--cognito-options Enabled=true,UserPoolId="user-pool-id",IdentityPoolId="identity-pool-
id",RoleArn="arn:aws:iam::123456789012:role/CognitoAccessForAmazonES"
```

**Example**

The following example creates a domain in the `us-east-1` Region that enables Amazon Cognito authentication for Kibana using the `CognitoAccessForAmazonES` role and provides domain access to `Cognito_Auth_Role`:

```
aws es create-elasticsearch-domain --domain-name my-domain --region us-east-1 --access-
policies '{ "Version":"2012-10-17", "Statement":[{"Effect":"Allow","Principal":{"AWS":
 ["arn:aws:iam::123456789012:role/
Cognito_Auth_Role"]},"Action":"es:ESHttp*","Resource":"arn:aws:es:us-
east-1:123456789012:domain/*" }]}' --elasticsearch-version "6.0" --elasticsearch-
cluster-config InstanceType=m4.xlarge.elasticsearch,InstanceCount=1
 --ebs-options EBSEnabled=true,VolumeSize=10 --cognito-options
 Enabled=true,UserPoolId="us-east-1_123456789",IdentityPoolId="us-
```

```
east-1:12345678-1234-1234-1234-123456789012",RoleArn="arn:aws:iam::123456789012:role/
CognitoAccessForAmazonES"
```

After your domain finishes processing, see the section called "Allowing the Authenticated Role" (p. 103) and the section called "Configuring Identity Providers" (p. 103) for additional configuration steps.

## Configuring Amazon Cognito Authentication (AWS SDKs)

The AWS SDKs (except the Android and iOS SDKs) support all the operations that are defined in the *Amazon ES Configuration API Reference* (p. 236), including the `CognitoOptions` parameter for the `CreateElasticsearchDomain` and `UpdateElasticsearchDomainConfig` operations. For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

After your domain finishes processing, see the section called "Allowing the Authenticated Role" (p. 103) and the section called "Configuring Identity Providers" (p. 103) for additional configuration steps.

## Allowing the Authenticated Role

By default, the authenticated IAM role that you configured by following the guidelines in the section called "About the Identity Pool" (p. 100) does not have the necessary privileges to access Kibana. You must provide the role with additional permissions.

> **Important**
> If you configured fine-grained access control (p. 76) and use an "open" or IP-based access policy, you can skip this step.

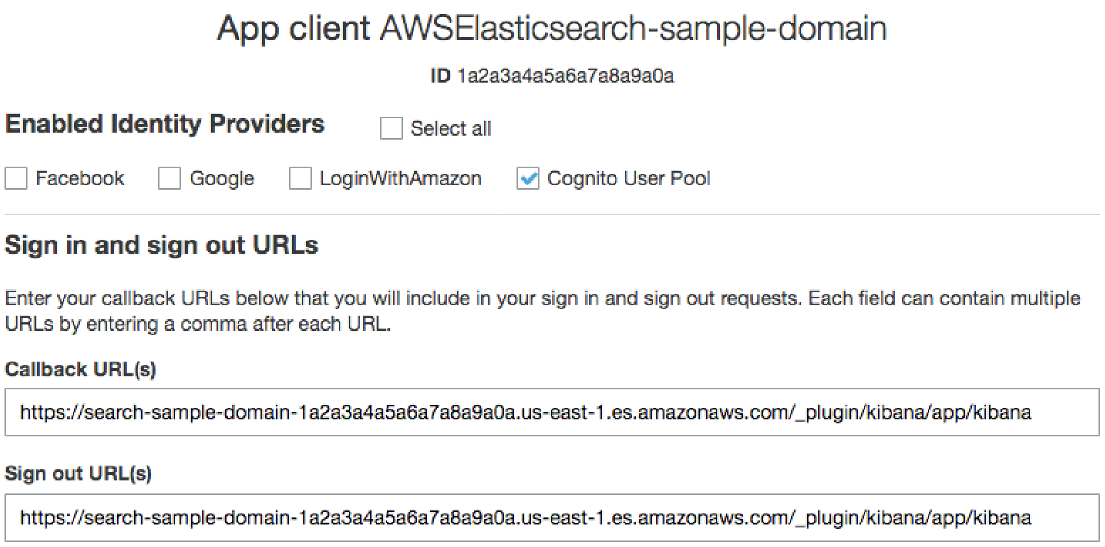
You can include these permissions in an identity-based (p. 65) policy, but unless you want authenticated users to have access to all Amazon ES domains, a resource-based (p. 64) policy attached to a single domain is the better approach:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:role/Cognito_identitypoolAuth_Role"
        ]
      },
      "Action": [
        "es:ESHttp*"
      ],
      "Resource": "arn:aws:es:region:123456789012:domain/domain-name/*"
    }
  ]
}
```

For instructions about adding a resource-based policy to an Amazon ES domain, see the section called "Configuring Access Policies" (p. 13).

## Configuring Identity Providers

When you configure a domain to use Amazon Cognito authentication for Kibana, Amazon ES adds an app client to the user pool and adds the user pool to the identity pool as an authentication provider. The following screenshot shows the **App client settings** page in the Amazon Cognito console.



App client AWSElasticsearch-sample-domain

**ID** 1a2a3a4a5a6a7a8a9a0a

**Enabled Identity Providers** ☐ Select all

☐ Facebook ☐ Google ☐ LoginWithAmazon ✓ Cognito User Pool

**Sign in and sign out URLs**

Enter your callback URLs below that you will include in your sign in and sign out requests. Each field can contain multiple URLs by entering a comma after each URL.

**Callback URL(s)**

https://search-sample-domain-1a2a3a4a5a6a7a8a9a0a.us-east-1.es.amazonaws.com/_plugin/kibana/app/kibana

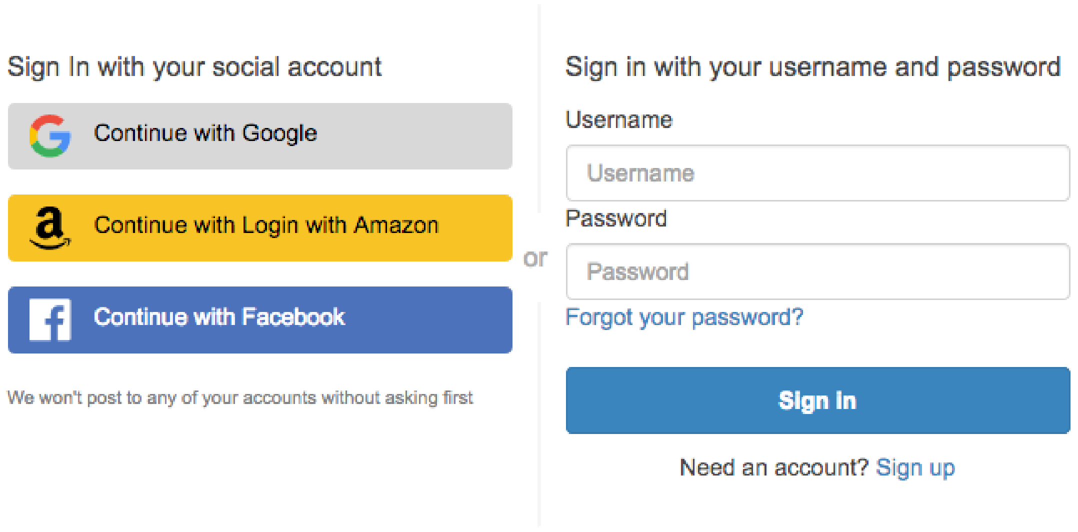
**Sign out URL(s)**

https://search-sample-domain-1a2a3a4a5a6a7a8a9a0a.us-east-1.es.amazonaws.com/_plugin/kibana/app/kibana

**Warning**
Don't rename or delete the app client.

Depending on how you configured your user pool, you might need to create user accounts manually, or users might be able to create their own. If these settings are acceptable, you don't need to take further action. Many people, however, prefer to use external identity providers.

To enable a SAML 2.0 identity provider, you must provide a SAML metadata document. To enable social identity providers like Login with Amazon, Facebook, and Google, you must have an app ID and app secret from those providers. You can enable any combination of identity providers. The sign-in page adds options as you add providers, as shown in the following screenshot.


Sign In with your social account
Continue with Google
Continue with Login with Amazon
Continue with Facebook
We won't post to any of your accounts without asking first
or
Sign in with your username and password
Username
Username
Password
Password
Forgot your password?
Sign In
Need an account? Sign up

The easiest way to configure your user pool is to use the Amazon Cognito console. Use the **Identity Providers** page to add external identity providers and the **App client settings** page to enable and disable identity providers for the Amazon ES domain's app client. For example, you might want to enable your own SAML identity provider and disable **Cognito User Pool** as an identity provider.

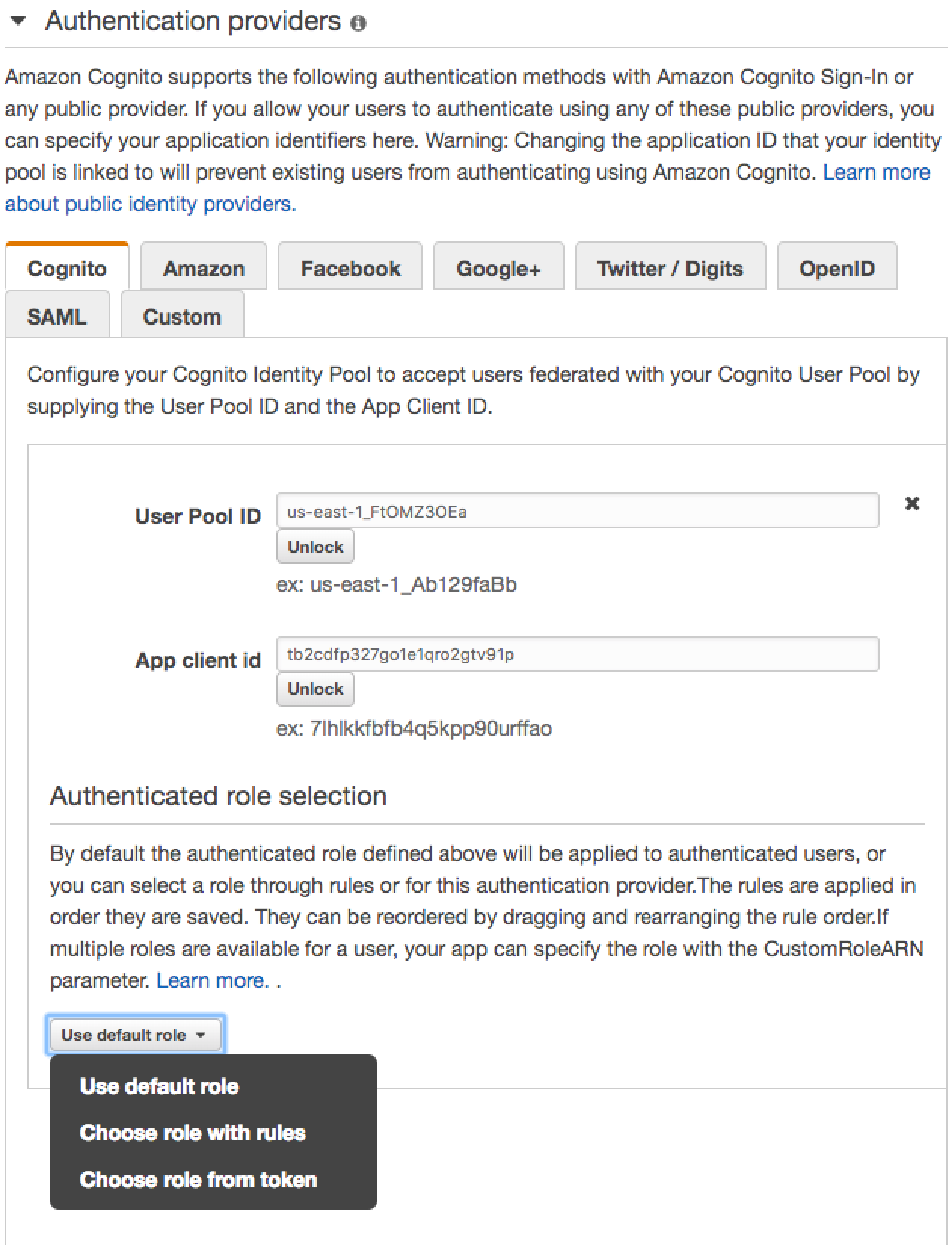
For instructions, see Using Federation from a User Pool and Specifying Identity Provider Settings for Your User Pool App in the *Amazon Cognito Developer Guide*.

## (Optional) Configuring Granular Access

You might have noticed that the default identity pool settings assign every user who logs in the same IAM role (`Cognito_identitypoolAuth_Role`), which means that every user can access the same AWS resources. If you want to use fine-grained access control (p. 76) with Amazon Cognito—for example, if you want your organization's analysts to have read-only access to several indices, but developers to have write access to all indices—you have two options:

- Create user groups and configure your identity provider to choose the IAM role based on the user's authentication token (recommended).
- Configure your identity provider to choose the IAM role based on one or more rules.

You configure these options using the **Edit identity pool** page of the Amazon Cognito console, as shown in the following screenshot. For a walkthrough that includes fine-grained access control, see the section called "Tutorial: IAM Master User and Amazon Cognito" (p. 86).



Authentication providers

Amazon Cognito supports the following authentication methods with Amazon Cognito Sign-In or any public provider. If you allow your users to authenticate using any of these public providers, you can specify your application identifiers here. Warning: Changing the application ID that your identity pool is linked to will prevent existing users from authenticating using Amazon Cognito. Learn more about public identity providers.

Configure your Cognito Identity Pool to accept users federated with your Cognito User Pool by supplying the User Pool ID and the App Client ID.

User Pool ID: us-east-1_FtOMZ3OEa

ex: us-east-1_Ab129faBb

App client id: tb2cdfp327go1e1qro2gtv91p

ex: 7lhlkkfbfb4q5kpp90urffao

Authenticated role selection

By default the authenticated role defined above will be applied to authenticated users, or you can select a role through rules or for this authentication provider.The rules are applied in order they are saved. They can be reordered by dragging and rearranging the rule order.If multiple roles are available for a user, your app can specify the role with the CustomRoleARN parameter. Learn more. .

**Important**
Just like the default role, Amazon Cognito must be part of each additional role's trust relationship. For details, see Creating Roles for Role Mapping in the *Amazon Cognito Developer Guide*.

### User Groups and Tokens

When you create a user group, you choose an IAM role for members of the group. For information about creating groups, see User Groups in the *Amazon Cognito Developer Guide*.

After you create one or more user groups, you can configure your authentication provider to assign users their groups' roles rather than the identity pool's default role. Choose the **Choose role from token** option. Then choose either **Use default Authenticated role** or **DENY** to specify how the identity pool should handle users who are not part of a group.

### Rules

Rules are essentially a series of `if` statements that Amazon Cognito evaluates sequentially. For example, if a user's email address contains `@corporate`, Amazon Cognito assigns that user `Role_A`. If a user's email address contains `@subsidiary`, it assigns that user `Role_B`. Otherwise, it assigns the user the default authenticated role.

To learn more, see Using Rule-Based Mapping to Assign Roles to Users in the *Amazon Cognito Developer Guide*.

## (Optional) Customizing the Sign-in Page

The **UI customization** page of the Amazon Cognito console lets you upload a custom logo and make CSS changes to the sign-in page. For instructions and a full list of CSS properties, see Specifying App UI Customization Settings for Your User Pool in the *Amazon Cognito Developer Guide*.

## (Optional) Configuring Advanced Security

Amazon Cognito user pools support advanced security features like multi-factor authentication, compromised credential checking, and adaptive authentication. To learn more, see Managing Security in the *Amazon Cognito Developer Guide*.

## Testing

After you are satisfied with your configuration, verify that the user experience meets your expectations.

**To access Kibana**

1. Navigate to `https://`*`elasticsearch-domain`*`/_plugin/kibana/` in a web browser.
2. Sign in using your preferred credentials.
3. After Kibana loads, configure at least one index pattern. Kibana uses these patterns to identity which indices that you want to analyze. Enter `*`, choose **Next step**, and then choose **Create index pattern**.
4. To search or explore your data, choose **Discover**.

If any step of this process fails, see the section called “Common Configuration Issues” (p. 108) for troubleshooting information.

## Limits

Amazon Cognito has soft limits on many of its resources. If you want to enable Kibana authentication for a large number of Amazon ES domains, review Limits in Amazon Cognito and request limit increases as necessary.

Each Amazon ES domain adds an app client to the user pool, which adds an authentication provider to the identity pool. If you enable Kibana authentication for more than 10 domains, you might encounter the "maximum Amazon Cognito user pool providers per identity pool" limit. If you exceed a limit, any Amazon ES domains that you try to configure to use Amazon Cognito authentication for Kibana can get stuck in a configuration state of **Processing**.

## Common Configuration Issues

The following tables list common configuration issues and solutions.

### Configuring Amazon ES

| Issue | Solution |
|---|---|
| `Amazon ES can't create the role` (console) | You don't have the correct IAM permissions. Add the permissions specified in the section called "Configuring Amazon Cognito Authentication (Console)" (p. 101). |
| `User is not authorized to perform: iam:PassRole on resource CognitoAccessForAmazonES` (console) | You don't have `iam:PassRole` permissions for the CognitoAccessForAmazonES (p. 100) role. Attach the following policy to your account:<br>`{`<br>`  "Version": "2012-10-17",`<br>`  "Statement": [`<br>`    {`<br>`      "Effect": "Allow",`<br>`      "Action": [`<br>`        "iam:PassRole"`<br>`      ],`<br>`      "Resource": "arn:aws:iam::123456789012:role/service-role/CognitoAccessForAmazonES"`<br>`    }`<br>`  ]`<br>`}`<br>Alternately, you can attach the `IAMFullAccess` policy. |
| `User is not authorized to perform: cognito-identity:ListIdentityPools on resource` | You don't have read permissions for Amazon Cognito. Attach the `AmazonCognitoReadOnly` policy to your account. |
| `An error occurred (ValidationException) when calling the CreateElasticsearchDomain operation: Amazon Elasticsearch must be allowed to use the passed role` | Amazon ES isn't specified in the trust relationship of the `CognitoAccessForAmazonES` role. Check that your role uses the trust relationship that is specified in the section called "About the CognitoAccessForAmazonES Role" (p. 100). Alternately, use the console to configure Amazon Cognito authentication. The console creates a role for you. |
| `An error occurred (ValidationException) when calling the CreateElasticsearchDomain operation: User is not authorized to perform: cognito-idp:`*`action`*` on resource: `*`user pool`* | The role specified in `--cognito-options` does not have permissions to access Amazon Cognito. Check that the role has the AWS managed `AmazonESCognitoAccess` policy attached. Alternately, use the console to configure Amazon Cognito authentication. The console creates a role for you. |

| Issue | Solution |
|---|---|
| `An error occurred (ValidationException) when calling the CreateElasticsearchDomain operation: User pool does not exist` | Amazon ES can't find the user pool. Confirm that you created one and have the correct ID. To find the ID, you can use the Amazon Cognito console or the following AWS CLI command:<br>`aws cognito-idp list-user-pools --max-results 60 --region region` |
| `An error occurred (ValidationException) when calling the CreateElasticsearchDomain operation: IdentityPool not found` | Amazon ES can't find the identity pool. Confirm that you created one and have the correct ID. To find the ID, you can use the Amazon Cognito console or the following AWS CLI command:<br>`aws cognito-identity list-identity-pools --max-results 60 --region region` |
| `An error occurred (ValidationException) when calling the CreateElasticsearchDomain operation: Domain needs to be specified for user pool` | The user pool does not have a domain name. You can configure one using the Amazon Cognito console or the following AWS CLI command:<br>`aws cognito-idp create-user-pool-domain --domain name --user-pool-id id` |

### Accessing Kibana

| Issue | Solution |
|---|---|
| The login page doesn't show my preferred identity providers. | Check that you enabled the identity provider for the Amazon ES app client as specified in the section called "Configuring Identity Providers" (p. 103). |
| The login page doesn't look as if it's associated with my organization. | See the section called "(Optional) Customizing the Sign-in Page" (p. 107). |
| My login credentials don't work. | Check that you have configured the identity provider as specified in the section called "Configuring Identity Providers" (p. 103).<br><br>If you use the user pool as your identity provider, check that the account exists and is confirmed on the **User and groups** page of the Amazon Cognito console. |
| Kibana either doesn't load at all or doesn't work properly. | The Amazon Cognito authenticated role needs `es:ESHttp*` permissions for the domain (`/*`) to access and use Kibana. Check that you added an access policy as specified in the section called "Allowing the Authenticated Role" (p. 103). |
| `Invalid identity pool configuration. Check assigned IAM roles for this pool.` | Amazon Cognito doesn't have permissions to assume the IAM role on behalf of the authenticated user. Modify the trust relationship for the role to include:<br>`{`<br>`  "Version": "2012-10-17",`<br>`  "Statement": [{` |

| Issue | Solution |
|---|---|
| | <pre>    "Effect": "Allow",<br>    "Principal": {<br>      "Federated": "cognito-identity.amazonaws.com"<br>    },<br>    "Action": "sts:AssumeRoleWithWebIdentity",<br>    "Condition": {<br>      "StringEquals": {<br>        "cognito-identity.amazonaws.com:aud":<br>"identity-pool-id"<br>      },<br>      "ForAnyValue:StringLike": {<br>        "cognito-identity.amazonaws.com:amr":<br>"authenticated"<br>      }<br>    }<br>  }]<br>}</pre> |
| `Token is not from a supported provider of this identity pool.` | This uncommon error can occur when you remove the app client from the user pool. Try opening Kibana in a new browser session. |

## Disabling Amazon Cognito Authentication for Kibana

Use the following procedure to disable Amazon Cognito authentication for Kibana.

**To disable Amazon Cognito authentication for Kibana (console)**

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. In the navigation pane, under **My domains**, choose the domain that you want to configure.
4. Choose **Edit domain**.
5. For **Amazon Cognito authentication**, clear the **Enable Amazon Cognito authentication** check box.
6. Choose **Submit**.

**Important**
If you no longer need the Amazon Cognito user pool and identity pool, delete them. Otherwise, you can continue to incur charges.

## Deleting Domains that Use Amazon Cognito Authentication for Kibana

To prevent domains that use Amazon Cognito authentication for Kibana from becoming stuck in a configuration state of **Processing**, delete Amazon ES domains *before* deleting their associated Amazon Cognito user pools and identity pools.

# Using Service-Linked Roles for Amazon ES

Amazon Elasticsearch Service uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Amazon ES. Service-linked roles are predefined by Amazon ES and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Amazon ES easier because you don’t have to manually add the necessary permissions. Amazon ES defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon ES can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting its related resources. This protects your Amazon ES resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

## Service-Linked Role Permissions for Amazon ES

Amazon ES uses the service-linked role named **AWSServiceRoleForAmazonElasticsearchService**.

The AWSServiceRoleForAmazonElasticsearchService service-linked role trusts the following services to assume the role:

- `es.amazonaws.com`

The role permissions policy allows Amazon ES to complete the following actions on the specified resources:

- Action: `ec2:CreateNetworkInterface` on `*`
- Action: `ec2:DeleteNetworkInterface` on `*`
- Action: `ec2:DescribeNetworkInterfaces` on `*`
- Action: `ec2:ModifyNetworkInterfaceAttribute` on `*`
- Action: `ec2:DescribeSecurityGroups` on `*`
- Action: `ec2:DescribeSubnets` on `*`

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-Linked Role Permissions in the *IAM User Guide*.

## Creating a Service-Linked Role for Amazon ES

You don't need to manually create a service-linked role. When you create a VPC access domain using the AWS Management Console, Amazon ES creates the service-linked role for you. In order for this automatic creation to succeed, you must have permissions for the `es:CreateElasticsearchServiceRole` and `iam:CreateServiceLinkedRole` actions.

If you delete this service-linked role and then need to create it again, you can use the same process to recreate the role in your account.

You can also use the IAM console, the IAM CLI, or the IAM API to create a service-linked role manually. For more information, see Creating a Service-Linked Role in the *IAM User Guide*.

## Editing a Service-Linked Role for Amazon ES

Amazon ES does not allow you to edit the AWSServiceRoleForAmazonElasticsearchService service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a Service-Linked Role in the *IAM User Guide*.

## Deleting a Service-Linked Role for Amazon ES

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up your service-linked role before you can manually delete it.

### Cleaning Up a Service-Linked Role

Before you can use IAM to delete a service-linked role, you must first confirm that the role has no active sessions and remove any resources used by the role.

**To check whether the service-linked role has an active session in the IAM console**

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the IAM console, choose **Roles**. Then choose the name (not the check box) of the AWSServiceRoleForAmazonElasticsearchService role.
3. On the **Summary** page for the selected role, choose the **Access Advisor** tab.
4. On the **Access Advisor** tab, review recent activity for the service-linked role.

   **Note**
   If you are unsure whether Amazon ES is using the AWSServiceRoleForAmazonElasticsearchService role, you can try to delete the role. If the service is using the role, then the deletion fails and you can view the regions where the role is being used. If the role is being used, then you must wait for the session to end before you can delete the role. You cannot revoke the session for a service-linked role.

**To remove Amazon ES resources used by the AWSServiceRoleForAmazonElasticsearchService**

1. Sign in to the AWS Management Console and open the Amazon ES console.
2. Delete any domains that list **VPC** under the **Endpoint** column.

### Manually Delete a Service-Linked Role

Use the Amazon ES configuration API to delete the AWSServiceRoleForAmazonElasticsearchService service-linked role. For more information, see the section called "DeleteElasticsearchServiceRole" (p. 243).

# Sample Code for Amazon Elasticsearch Service

This chapter contains common sample code for working with Amazon Elasticsearch Service: HTTP request signing in a variety of programming languages, using the AWS SDKs to create domains, and using Curator with AWS Lambda to perform periodic index and snapshot management tasks.

**Topics**

## Signing HTTP Requests to Amazon Elasticsearch Service

This section includes examples of how to send signed HTTP requests to Amazon Elasticsearch Service using Elasticsearch clients and other common libraries. These code samples are for interacting with the Elasticsearch APIs, such as `_index`, `_bulk`, and `_snapshot`. If your domain access policy includes IAM users or roles (or you use an IAM master user with fine-grained access control (p. 76)), you must sign requests to the Elasticsearch APIs with your IAM credentials.

**Important**
For examples of how to interact with the configuration API, including operations like creating, updating, and deleting Amazon ES domains, see the section called “Using the AWS SDKs” (p. 121).

**Topics**

### Java

The easiest way of sending a signed request is to use the AWS Request Signing Interceptor. The repository contains some samples to help you get started, or you can download a sample project for Amazon ES on GitHub.

The following example uses the Elasticsearch low-level Java REST client to perform two unrelated actions: registering a snapshot repository and indexing a document. You must provide values for `region` and `host`.

```
import org.apache.http.HttpEntity;
import org.apache.http.HttpHost;
import org.apache.http.HttpRequestInterceptor;
import org.apache.http.entity.ContentType;
import org.apache.http.nio.entity.NStringEntity;
import org.elasticsearch.client.Request;
import org.elasticsearch.client.Response;
import org.elasticsearch.client.RestClient;
import com.amazonaws.auth.AWS4Signer;
import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.http.AWSRequestSigningApacheInterceptor;
import java.io.IOException;

public class AmazonElasticsearchServiceSample {

    private static String serviceName = "es";
    private static String region = "us-west-1";
    private static String aesEndpoint = "https://domain.us-west-1.es.amazonaws.com";

    private static String payload = "{ \"type\": \"s3\", \"settings\":
 { \"bucket\": \"your-bucket\", \"region\": \"us-west-1\", \"role_arn\":
 \"arn:aws:iam::123456789012:role/TheServiceRole\" } }";
    private static String snapshotPath = "/_snapshot/my-snapshot-repo";

    private static String sampleDocument = "{" + "\"title\":\"Walk the Line\"," +
 "\"director\":\"James Mangold\"," + "\"year\":\"2005\"}";
    private static String indexingPath = "/my-index/_doc";

    static final AWSCredentialsProvider credentialsProvider = new
 DefaultAWSCredentialsProviderChain();

    public static void main(String[] args) throws IOException {
        RestClient esClient = esClient(serviceName, region);

        // Register a snapshot repository
        HttpEntity entity = new NStringEntity(payload, ContentType.APPLICATION_JSON);
        Request request = new Request("PUT", snapshotPath);
        request.setEntity(entity);
        // request.addParameter(name, value); // optional parameters
        Response response = esClient.performRequest(request);
        System.out.println(response.toString());

        // Index a document
        entity = new NStringEntity(sampleDocument, ContentType.APPLICATION_JSON);
        String id = "1";
        request = new Request("PUT", indexingPath + "/" + id);
        request.setEntity(entity);

        // Using a String instead of an HttpEntity sets Content-Type to application/json
 automatically.
        // request.setJsonEntity(sampleDocument);

        response = esClient.performRequest(request);
        System.out.println(response.toString());
    }

    // Adds the interceptor to the ES REST client
    public static RestClient esClient(String serviceName, String region) {
        AWS4Signer signer = new AWS4Signer();
        signer.setServiceName(serviceName);
        signer.setRegionName(region);
        HttpRequestInterceptor interceptor = new
 AWSRequestSigningApacheInterceptor(serviceName, signer, credentialsProvider);
        return
 RestClient.builder(HttpHost.create(aesEndpoint)).setHttpClientConfigCallback(hacb ->
 hacb.addInterceptorLast(interceptor)).build();
    }
}
```

If you prefer the high-level REST client, which offers most of the same features and simpler code, try the following sample, which also uses the AWS Request Signing Interceptor:

```
import org.apache.http.HttpHost;
import org.apache.http.HttpRequestInterceptor;
import org.elasticsearch.action.index.IndexRequest;
import org.elasticsearch.action.index.IndexResponse;
import org.elasticsearch.client.RequestOptions;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestHighLevelClient;
import com.amazonaws.auth.AWS4Signer;
import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.http.AWSRequestSigningApacheInterceptor;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class AmazonElasticsearchServiceSample {

    private static String serviceName = "es";
    private static String region = "us-west-1";
    private static String aesEndpoint = ""; // e.g. https://search-mydomain.us-
west-1.es.amazonaws.com
    private static String index = "my-index";
    private static String type = "_doc";
    private static String id = "1";

    static final AWSCredentialsProvider credentialsProvider = new
 DefaultAWSCredentialsProviderChain();

    public static void main(String[] args) throws IOException {
        RestHighLevelClient esClient = esClient(serviceName, region);

        // Create the document as a hash map
        Map<String, Object> document = new HashMap<>();
        document.put("title", "Walk the Line");
        document.put("director", "James Mangold");
        document.put("year", "2005");

        // Form the indexing request, send it, and print the response
        IndexRequest request = new IndexRequest(index, type, id).source(document);
        IndexResponse response = esClient.index(request, RequestOptions.DEFAULT);
        System.out.println(response.toString());
    }

    // Adds the interceptor to the ES REST client
    public static RestHighLevelClient esClient(String serviceName, String region) {
        AWS4Signer signer = new AWS4Signer();
        signer.setServiceName(serviceName);
        signer.setRegionName(region);
        HttpRequestInterceptor interceptor = new
 AWSRequestSigningApacheInterceptor(serviceName, signer, credentialsProvider);
        return new
 RestHighLevelClient(RestClient.builder(HttpHost.create(aesEndpoint)).setHttpClientConfigCallback(hacb
 -> hacb.addInterceptorLast(interceptor)));
    }
}
```

**Tip**

Both signed samples use the default credential chain. Run `aws configure` using the AWS CLI to set your credentials.

## Python

You can install elasticsearch-py, an Elasticsearch client for Python, using pip. Instead of the client, you might prefer requests. The requests-aws4auth and SDK for Python (Boto 3) packages simplify the authentication process, but are not strictly required. From the terminal, run the following commands:

```
pip install boto3
pip install elasticsearch
pip install requests
pip install requests-aws4auth
```

The following sample code establishes a secure connection to the specified Amazon ES domain and indexes a single document. You must provide values for `region` and `host`.

```
from elasticsearch import Elasticsearch, RequestsHttpConnection
from requests_aws4auth import AWS4Auth
import boto3

host = '' # For example, my-test-domain.us-east-1.es.amazonaws.com
region = '' # e.g. us-west-1

service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

es = Elasticsearch(
    hosts = [{'host': host, 'port': 443}],
    http_auth = awsauth,
    use_ssl = True,
    verify_certs = True,
    connection_class = RequestsHttpConnection
)

document = {
    "title": "Moneyball",
    "director": "Bennett Miller",
    "year": "2011"
}

es.index(index="movies", doc_type="_doc", id="5", body=document)

print(es.get(index="movies", doc_type="_doc", id="5"))
```

If you don't want to use elasticsearch-py, you can just make standard HTTP requests. This sample creates a new index with seven shards and two replicas:

```
from requests_aws4auth import AWS4Auth
import boto3
import requests

host = '' # The domain with https:// and trailing slash. For example, https://my-test-
domain.us-east-1.es.amazonaws.com/
path = 'my-index' # the Elasticsearch API endpoint
region = '' # For example, us-west-1

service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)
```

```
url = host + path

# The JSON body to accompany the request (if necessary)
payload = {
    "settings" : {
        "number_of_shards" : 7,
        "number_of_replicas" : 2
    }
}

r = requests.put(url, auth=awsauth, json=payload) # requests.get, post, and delete have
 similar syntax

print(r.text)
```

This next example uses the Beautiful Soup library to help build a bulk file from a local directory of HTML files. Using the same client as the first example, you can send the file to the `_bulk` API for indexing. You could use this code as the basis for adding search functionality to a website:

```
from bs4 import BeautifulSoup
from elasticsearch import Elasticsearch, RequestsHttpConnection
from requests_aws4auth import AWS4Auth
import boto3
import glob
import json

bulk_file = ''
id = 1

# This loop iterates through all HTML files in the current directory and
# indexes two things: the contents of the first h1 tag and all other text.

for html_file in glob.glob('*.htm'):

    with open(html_file) as f:
        soup = BeautifulSoup(f, 'html.parser')

    title = soup.h1.string
    body = soup.get_text(" ", strip=True)
    # If get_text() is too noisy, you can do further processing on the string.

    index = { 'title': title, 'body': body, 'link': html_file }
    # If running this script on a website, you probably need to prepend the URL and path to
 html_file.

    # The action_and_metadata portion of the bulk file
    bulk_file += '{ "index" : { "_index" : "site", "_type" : "_doc", "_id" : "' + str(id) +
 '" } }\n'

    # The optional_document portion of the bulk file
    bulk_file += json.dumps(index) + '\n'

    id += 1

host = '' # For example, my-test-domain.us-east-1.es.amazonaws.com
region = '' # e.g. us-west-1

service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service)

es = Elasticsearch(
    hosts = [{'host': host, 'port': 443}],
    http_auth = awsauth,
    use_ssl = True,
    verify_certs = True,
    connection_class = RequestsHttpConnection
)

es.bulk(bulk_file)

print(es.search(q='some test query'))
```

## Ruby

This first example uses the Elasticsearch Ruby client and Faraday middleware to perform the request signing. From the terminal, run the following commands:

```
gem install elasticsearch
gem install faraday_middleware-aws-sigv4
```

This sample code creates a new Elasticsearch client, configures Faraday middleware to sign requests, and indexes a single document. You must provide values for `host` and `region`.

```
require 'elasticsearch'
require 'faraday_middleware/aws_sigv4'

host = '' # e.g. https://my-domain.region.es.amazonaws.com
index = 'ruby-index'
type = '_doc'
id = '1'
document = {
  year: 2007,
  title: '5 Centimeters per Second',
  info: {
    plot: 'Told in three interconnected segments, we follow a young man named Takaki
 through his life.',
    rating: 7.7
  }
}

region = '' # e.g. us-west-1
service = 'es'

client = Elasticsearch::Client.new(url: host) do |f|
  f.request :aws_sigv4,
    service: service,
    region: region,
    access_key_id: ENV['AWS_ACCESS_KEY_ID'],
    secret_access_key: ENV['AWS_SECRET_ACCESS_KEY'],
    session_token: ENV['AWS_SESSION_TOKEN'] # optional
end

puts client.index index: index, type: type, id: id, body: document
```

If your credentials don't work, export them at the terminal using the following commands:

```
export AWS_ACCESS_KEY_ID="your-access-key"
export AWS_SECRET_ACCESS_KEY="your-secret-key"
export AWS_SESSION_TOKEN="your-session-token"
```

This next example uses the AWS SDK for Ruby and standard Ruby libraries to send a signed HTTP request. Like the first example, it indexes a single document. You must provide values for host and region.

```
require 'aws-sdk-elasticsearchservice'

host = '' # e.g. https://my-domain.region.es.amazonaws.com
index = 'ruby-index'
type = '_doc'
id = '2'
document = {
  year: 2007,
  title: '5 Centimeters per Second',
  info: {
    plot: 'Told in three interconnected segments, we follow a young man named Takaki
 through his life.',
    rating: 7.7
  }
}

service = 'es'
region = '' # e.g. us-west-1

signer = Aws::Sigv4::Signer.new(
  service: service,
  region: region,
  access_key_id: ENV['AWS_ACCESS_KEY_ID'],
  secret_access_key: ENV['AWS_SECRET_ACCESS_KEY'],
  session_token: ENV['AWS_SESSION_TOKEN']
)

signature = signer.sign_request(
  http_method: 'PUT',
  url: host + '/' + index + '/' + type + '/' + id,
  body: document.to_json
)

uri = URI(host + '/' + index + '/' + type + '/' + id)

Net::HTTP.start(uri.host, uri.port, :use_ssl => true) do |http|
  request = Net::HTTP::Put.new uri
  request.body = document.to_json
  request['Host'] = signature.headers['host']
  request['X-Amz-Date'] = signature.headers['x-amz-date']
  request['X-Amz-Security-Token'] = signature.headers['x-amz-security-token']
  request['X-Amz-Content-Sha256']= signature.headers['x-amz-content-sha256']
  request['Authorization'] = signature.headers['authorization']
  request['Content-Type'] = 'application/json'
  response = http.request request
  puts response.body
end
```

## Node

This example uses the SDK for JavaScript in Node.js. From the terminal, run the following commands:

```
npm install aws-sdk
```

This sample code indexes a single document. You must provide values for `region` and `domain`.

```
var AWS = require('aws-sdk');

var region = ''; // e.g. us-west-1
var domain = ''; // e.g. search-domain.region.es.amazonaws.com
var index = 'node-test';
var type = '_doc';
var id = '1';
var json = {
  "title": "Moneyball",
  "director": "Bennett Miller",
  "year": "2011"
}

indexDocument(json);

function indexDocument(document) {
  var endpoint = new AWS.Endpoint(domain);
  var request = new AWS.HttpRequest(endpoint, region);

  request.method = 'PUT';
  request.path += index + '/' + type + '/' + id;
  request.body = JSON.stringify(document);
  request.headers['host'] = domain;
  request.headers['Content-Type'] = 'application/json';
  // Content-Length is only needed for DELETE requests that include a request
  // body, but including it for all requests doesn't seem to hurt anything.
  request.headers['Content-Length'] = Buffer.byteLength(request.body);

  var credentials = new AWS.EnvironmentCredentials('AWS');
  var signer = new AWS.Signers.V4(request, 'es');
  signer.addAuthorization(credentials, new Date());

  var client = new AWS.HttpClient();
  client.handleRequest(request, null, function(response) {
    console.log(response.statusCode + ' ' + response.statusMessage);
    var responseBody = '';
    response.on('data', function (chunk) {
      responseBody += chunk;
    });
    response.on('end', function (chunk) {
      console.log('Response body: ' + responseBody);
    });
  }, function(error) {
    console.log('Error: ' + error);
  });
}
```

If your credentials don't work, export them at the terminal using the following commands:

```
export AWS_ACCESS_KEY_ID="your-access-key"
export AWS_SECRET_ACCESS_KEY="your-secret-key"
export AWS_SESSION_TOKEN="your-session-token"
```

## Go

This example uses the AWS SDK for Go and indexes a single document. You must provide values for `domain` and `region`.

```
package main

import (
  "fmt"
  "net/http"
  "strings"
  "time"
  "github.com/aws/aws-sdk-go/aws/credentials"
  "github.com/aws/aws-sdk-go/aws/signer/v4"
)

func main() {

  // Basic information for the Amazon Elasticsearch Service domain
  domain := "" // e.g. https://my-domain.region.es.amazonaws.com
  index := "my-index"
  id := "1"
  endpoint := domain + "/" + index + "/" + "_doc" + "/" + id
  region := "" // e.g. us-east-1
  service := "es"

  // Sample JSON document to be included as the request body
  json := `{ "title": "Thor: Ragnarok", "director": "Taika Waititi", "year": "2017" }`
  body := strings.NewReader(json)

  // Get credentials from environment variables and create the AWS Signature Version 4
 signer
  credentials := credentials.NewEnvCredentials()
  signer := v4.NewSigner(credentials)

  // An HTTP client for sending the request
  client := &http.Client{}

  // Form the HTTP request
  req, err := http.NewRequest(http.MethodPut, endpoint, body)
  if err != nil {
    fmt.Print(err)
  }

  // You can probably infer Content-Type programmatically, but here, we just say that it's
 JSON
  req.Header.Add("Content-Type", "application/json")

  // Sign the request, send it, and print the response
  signer.Sign(req, body, service, region, time.Now())
  resp, err := client.Do(req)
  if err != nil {
    fmt.Print(err)
  }
  fmt.Print(resp.Status + "\n")
}
```

If your credentials don't work, export them at the terminal using the following commands:

```
export AWS_ACCESS_KEY_ID="your-access-key"
export AWS_SECRET_ACCESS_KEY="your-secret-key"
export AWS_SESSION_TOKEN="your-session-token"
```

## Using the AWS SDKs with Amazon Elasticsearch Service

This section includes examples of how to use the AWS SDKs to interact with the Amazon Elasticsearch Service configuration API. These code samples show how to create, update, and delete Amazon ES domains.

**Important**
For examples of how to interact with the Elasticsearch APIs, such as `_index`, `_bulk`, `_search`, and `_snapshot`, see the section called "Signing HTTP Requests" (p. 113).

## Java

This example uses the AWS SDK for Java to create a domain, update its configuration, and delete it. Uncomment the calls to `waitForDomainProcessing` (and comment the call to `deleteDomain`) to allow the domain to come online and be useable.

```
package com.amazonaws.samples;

import java.util.concurrent.TimeUnit;

import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.elasticsearch.AWSElasticsearch;
import com.amazonaws.services.elasticsearch.AWSElasticsearchClientBuilder;
import com.amazonaws.services.elasticsearch.model.CreateElasticsearchDomainRequest;
import com.amazonaws.services.elasticsearch.model.CreateElasticsearchDomainResult;
import com.amazonaws.services.elasticsearch.model.DeleteElasticsearchDomainRequest;
import com.amazonaws.services.elasticsearch.model.DeleteElasticsearchDomainResult;
import com.amazonaws.services.elasticsearch.model.DescribeElasticsearchDomainRequest;
import com.amazonaws.services.elasticsearch.model.DescribeElasticsearchDomainResult;
import com.amazonaws.services.elasticsearch.model.EBSOptions;
import com.amazonaws.services.elasticsearch.model.ElasticsearchClusterConfig;
import com.amazonaws.services.elasticsearch.model.NodeToNodeEncryptionOptions;
import com.amazonaws.services.elasticsearch.model.ResourceNotFoundException;
import com.amazonaws.services.elasticsearch.model.SnapshotOptions;
import com.amazonaws.services.elasticsearch.model.UpdateElasticsearchDomainConfigRequest;
import com.amazonaws.services.elasticsearch.model.UpdateElasticsearchDomainConfigResult;
import com.amazonaws.services.elasticsearch.model.VolumeType;

/**
 * Sample class demonstrating how to use the AWS SDK for Java to create, update,
 * and delete Amazon Elasticsearch Service domains.
 */

public class AESSample {

    public static void main(String[] args) {

        final String domainName = "my-test-domain";

        // Build the client using the default credentials chain.
        // You can use the AWS CLI and run `aws configure` to set access key, secret
        // key, and default region.
        final AWSElasticsearch client = AWSElasticsearchClientBuilder
                .standard()
                // Unnecessary, but lets you use a region different than your default.
                .withRegion(Regions.US_WEST_2)
                // Unnecessary, but if desired, you can use a different provider chain.
                .withCredentials(new DefaultAWSCredentialsProviderChain())
                .build();

        // Create a new domain, update its configuration, and delete it.
        createDomain(client, domainName);
        // waitForDomainProcessing(client, domainName);
        updateDomain(client, domainName);
        // waitForDomainProcessing(client, domainName);
        deleteDomain(client, domainName);
    }

    /**
     * Creates an Amazon Elasticsearch Service domain with the specified options.
     * Some options require other AWS resources, such as an Amazon Cognito user pool
     * and identity pool, whereas others require just an instance type or instance
     * count.
     *
     * @param client
     *            The AWSElasticsearch client to use for the requests to Amazon
     *            Elasticsearch Service
     * @param domainName
     *            The name of the domain you want to create
     */
    private static void createDomain(final AWSElasticsearch client, final String
 domainName) {

        // Create the request and set the desired configuration options
        CreateElasticsearchDomainRequest createRequest = new
 CreateElasticsearchDomainRequest()
                .withDomainName(domainName)
                .withElasticsearchVersion("6.3")
                .withElasticsearchClusterConfig(new ElasticsearchClusterConfig()
                        .withDedicatedMasterEnabled(true)
                        .withDedicatedMasterCount(3)
                        // Small, inexpensive instance types for testing. Not recommended
 for production
                        // domains.
                        .withDedicatedMasterType("t2.small.elasticsearch")
                        .withInstanceType("t2.small.elasticsearch")
                        .withInstanceCount(5))
                // Many instance types require EBS storage.
                .withEBSOptions(new EBSOptions()
                        .withEBSEnabled(true)
                        .withVolumeSize(10)
                        .withVolumeType(VolumeType.Gp2))
                // You can uncomment this line and add your account ID, a user name, and
 the
                // domain name to add an access policy.
                // .withAccessPolicies("{\"Version\":\"2012-10-17\",\"Statement\":
[{\"Effect\":\"Allow\",\"Principal\":{\"AWS\":[\"arn:aws:iam::123456789012:user/user-name
\"]},\"Action\":[\"es:*\"],\"Resource\":\"arn:aws:es:region:123456789012:domain/domain-
name/*\"}]}")
                .withNodeToNodeEncryptionOptions(new NodeToNodeEncryptionOptions()
                        .withEnabled(true));

        // Make the request.
        System.out.println("Sending domain creation request...");
        CreateElasticsearchDomainResult createResponse =
 client.createElasticsearchDomain(createRequest);
        System.out.println("Domain creation response from Amazon Elasticsearch Service:");
        System.out.println(createResponse.getDomainStatus().toString());
    }

    /**
     * Updates the configuration of an Amazon Elasticsearch Service domain with the
     * specified options. Some options require other AWS resources, such as an
     * Amazon Cognito user pool and identity pool, whereas others require just an
     * instance type or instance count.
     *
     * @param client
     *            The AWSElasticsearch client to use for the requests to Amazon
     *            Elasticsearch Service
     * @param domainName
     *            The name of the domain to update
     */
    private static void updateDomain(final AWSElasticsearch client, final String
 domainName) {
        try {
            // Updates the domain to use three data instances instead of five.
            // You can uncomment the Cognito lines and fill in the strings to enable
 Cognito
            // authentication for Kibana.
```

```
           final UpdateElasticsearchDomainConfigRequest updateRequest = new
UpdateElasticsearchDomainConfigRequest()
                   .withDomainName(domainName)
                   // .withCognitoOptions(new CognitoOptions()
                           // .withEnabled(true)
                           // .withUserPoolId("user-pool-id")
                           // .withIdentityPoolId("identity-pool-id")
                           // .withRoleArn("role-arn"))
                   .withElasticsearchClusterConfig(new ElasticsearchClusterConfig()
                           .withInstanceCount(3));

           System.out.println("Sending domain update request...");
           final UpdateElasticsearchDomainConfigResult updateResponse = client
                   .updateElasticsearchDomainConfig(updateRequest);
           System.out.println("Domain update response from Amazon Elasticsearch
Service:");
           System.out.println(updateResponse.toString());
       } catch (ResourceNotFoundException e) {
           System.out.println("Domain not found. Please check the domain name.");
       }
   }

   /**
    * Deletes an Amazon Elasticsearch Service domain. Deleting a domain can take
    * several minutes.
    *
    * @param client
    *            The AWSElasticsearch client to use for the requests to Amazon
    *            Elasticsearch Service
    * @param domainName
    *            The name of the domain that you want to delete
    */
   private static void deleteDomain(final AWSElasticsearch client, final String
domainName) {
       try {
           final DeleteElasticsearchDomainRequest deleteRequest = new
DeleteElasticsearchDomainRequest()
                   .withDomainName(domainName);

           System.out.println("Sending domain deletion request...");
           final DeleteElasticsearchDomainResult deleteResponse =
client.deleteElasticsearchDomain(deleteRequest);
           System.out.println("Domain deletion response from Amazon Elasticsearch
Service:");
           System.out.println(deleteResponse.toString());
       } catch (ResourceNotFoundException e) {
           System.out.println("Domain not found. Please check the domain name.");
       }
   }

   /**
    * Waits for the domain to finish processing changes. New domains typically take 15-30
minutes
    * to initialize, but can take longer depending on the configuration. Most updates to
existing domains
    * take a similar amount of time. This method checks every 15 seconds and finishes only
when
    * the domain's processing status changes to false.
    *
    * @param client
    *            The AWSElasticsearch client to use for the requests to Amazon
    *            Elasticsearch Service
    * @param domainName
    *            The name of the domain that you want to check
    */
    private static void waitForDomainProcessing(final AWSElasticsearch client, final String
 domainName) {
        // Create a new request to check the domain status.
        final DescribeElasticsearchDomainRequest describeRequest = new
 DescribeElasticsearchDomainRequest()
                .withDomainName(domainName);

        // Every 15 seconds, check whether the domain is processing.
        DescribeElasticsearchDomainResult describeResponse =
 client.describeElasticsearchDomain(describeRequest);
        while (describeResponse.getDomainStatus().isProcessing()) {
            try {
                System.out.println("Domain still processing...");
                TimeUnit.SECONDS.sleep(15);
                describeResponse = client.describeElasticsearchDomain(describeRequest);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        // Once we exit that loop, the domain is available
        System.out.println("Amazon Elasticsearch Service has finished processing changes
 for your domain.");
        System.out.println("Domain description response from Amazon Elasticsearch
 Service:");
        System.out.println(describeResponse.toString());
    }
}
```

## Using Curator to Rotate Data in Amazon Elasticsearch Service

This section contains sample code for using AWS Lambda and Curator to manage indices and snapshots. Curator offers numerous filters to help you identify indices and snapshots that meet certain criteria, such as indices created more than 60 days ago or snapshots that failed to complete. Index State Management (p. 166) has some similar features and doesn't require Lambda or a separate EC2 instance. Depending on your use case, it might be a better choice.

Although Curator is often used as a command line interface (CLI), it also features a Python API, which means that you can use it within Lambda functions.

For information about configuring Lambda functions and creating deployment packages, see the section called “Loading Streaming Data into Amazon ES from Amazon S3” (p. 132). For even more information, see the AWS Lambda Developer Guide. This section contains only sample code, basic settings, triggers, and permissions.

**Topics**

### Sample Code

The following sample code uses Curator and elasticsearch-py to delete any index whose name contains a time stamp indicating that the data is more than 30 days old. For example, if an index name is `my-`

`logs-2014.03.02`, the index is deleted. Deletion occurs even if you create the index today, because this filter uses the name of the index to determine its age.

The code also contains some commented-out examples of other common filters, including one that determines age by creation date. The AWS SDK for Python (Boto 3) and requests-aws4auth library sign the requests to Amazon ES.

**Warning**
Both code samples in this section delete data—potentially a lot of data. Modify and test each sample on a non-critical domain until you're satisfied with its behavior.

**Index Deletion**

```
import boto3
from requests_aws4auth import AWS4Auth
from elasticsearch import Elasticsearch, RequestsHttpConnection
import curator

host = '' # For example, search-my-domain.region.es.amazonaws.com
region = '' # For example, us-west-1
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

# Lambda execution starts here.
def lambda_handler(event, context):

    # Build the Elasticsearch client.
    es = Elasticsearch(
        hosts = [{'host': host, 'port': 443}],
        http_auth = awsauth,
        use_ssl = True,
        verify_certs = True,
        connection_class = RequestsHttpConnection
    )

    # A test document.
    document = {
        "title": "Moneyball",
        "director": "Bennett Miller",
        "year": "2011"
    }

    # Index the test document so that we have an index that matches the timestring pattern.
    # You can delete this line and the test document if you already created some test
 indices.
    es.index(index="movies-2017.01.31", doc_type="movie", id="1", body=document)

    index_list = curator.IndexList(es)

    # Filters by age, anything with a time stamp older than 30 days in the index name.
    index_list.filter_by_age(source='name', direction='older', timestring='%Y.%m.%d',
 unit='days', unit_count=30)

    # Filters by naming prefix.
    # index_list.filter_by_regex(kind='prefix', value='my-logs-2017')

    # Filters by age, anything created more than one month ago.
    # index_list.filter_by_age(source='creation_date', direction='older', unit='months',
 unit_count=1)

    print("Found %s indices to delete" % len(index_list.indices))
    # If our filtered list contains any indices, delete them.
    if index_list.indices:
        curator.DeleteIndices(index_list).do_action()
```

You must update the values for `host` and `region`.

The next code sample deletes any snapshot that is more than two weeks old. It also takes a new snapshot.

**Snapshot Deletion**

```
import boto3
from datetime import datetime
from requests_aws4auth import AWS4Auth
from elasticsearch import Elasticsearch, RequestsHttpConnection
import logging
import curator

# Adding a logger isn't strictly required, but helps with understanding Curator's requests
 and debugging.
logger = logging.getLogger('curator')
logger.addHandler(logging.StreamHandler())
logger.setLevel(logging.INFO)

host = '' # For example, search-my-domain.region.es.amazonaws.com
region = '' # For example, us-west-1
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)
repository_name = 'my-repo'

# Lambda execution starts here.
def lambda_handler(event, context):

    now = datetime.now()

    # Clunky, but this approach keeps colons out of the URL.
    date_string = '-'.join((str(now.year), str(now.month), str(now.day), str(now.hour),
 str(now.minute)))
    snapshot_name = 'my-snapshot-prefix-' + date_string

    # Build the Elasticsearch client.
    es = Elasticsearch(
        hosts = [{'host': host, 'port': 443}],
        http_auth = awsauth,
        use_ssl = True,
        verify_certs = True,
        connection_class = RequestsHttpConnection,
        timeout = 120 # Deleting snapshots can take a while, so keep the connection open
 for long enough to get a response.
    )

    try:
        # Get all snapshots in the repository.
        snapshot_list = curator.SnapshotList(es, repository=repository_name)

        # Filter by age, any snapshot older than two weeks.
        # snapshot_list.filter_by_age(source='creation_date', direction='older',
 unit='weeks', unit_count=2)

        # Delete the old snapshots.
        curator.DeleteSnapshots(snapshot_list, retry_interval=30,
 retry_count=3).do_action()
```

```
    except (curator.exceptions.SnapshotInProgress, curator.exceptions.NoSnapshots,
curator.exceptions.FailedExecution) as e:
        print(e)

    # Split into two try blocks. We still want to try and take a snapshot if deletion
failed.
    try:
        # Get the list of indices.
        # You can filter this list if you didn't want to snapshot all indices.
        index_list = curator.IndexList(es)

        # Take a new snapshot. This operation can take a while, so we don't want to wait
for it to complete.
        curator.Snapshot(index_list, repository=repository_name, name=snapshot_name,
wait_for_completion=False).do_action()
    except (curator.exceptions.SnapshotInProgress, curator.exceptions.FailedExecution) as
e:
        print(e)
```

You must update the values for `host`, `region`, `snapshot_name`, and `repository_name`. If the output is too verbose for your taste, you can change `logging.INFO` to `logging.WARN`.

Because taking and deleting snapshots can take a while, this code is more sensitive to connection and Lambda timeouts—hence the extra logging code. In the Elasticsearch client, you can see that we set the timeout to 120 seconds. If the `DeleteSnapshots` function takes longer to get a response from the Amazon ES domain, you might need to increase this value. You must also increase the Lambda function timeout from its default value of three seconds. For a recommended value, see the section called “Basic Settings” (p. 128).

## Basic Settings

We recommend the following settings for these code samples.

| Sample Code | Memory | Timeout |
|---|---|---|
| Index Deletion | 128 MB | 10 seconds |
| Snapshot Deletion | 128 MB | 3 minutes |

## Triggers

Rather than reacting to some event (such as a file upload to Amazon S3), these functions are meant to be scheduled. You might prefer to run these functions more or less frequently.

| Sample Code | Service | Rule Type | Example Expression |
|---|---|---|---|
| Index Deletion | CloudWatch Events | Schedule expression | rate(1 day) |
| Snapshot Deletion | CloudWatch Events | Schedule expression | rate(4 hours) |

## Permissions

Both Lambda functions in this section need the basic logging permissions that all Lambda functions need, plus HTTP method permissions for the Amazon ES domain:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:CreateLogGroup",
      "Resource": "arn:aws:logs:us-west-1:123456789012:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-west-1:123456789012:log-group:/aws/lambda/your-lambda-function:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "es:ESHttpPost",
        "es:ESHttpGet",
        "es:ESHttpPut",
        "es:ESHttpDelete"
      ],
      "Resource": "arn:aws:es:us-west-1:123456789012:domain/my-domain/*"
    }
  ]
}
```

# Introduction to Indexing Data in Amazon Elasticsearch Service

Because Elasticsearch uses a REST API, numerous methods exist for indexing documents. You can use standard clients like curl or any programming language that can send HTTP requests. To further simplify the process of interacting with it, Elasticsearch has clients for many programming languages. Advanced users can skip directly to the section called "Signing HTTP Requests" (p. 113) or the section called "Loading Streaming Data into Amazon ES" (p. 132).

For an introduction to indexing, see the Open Distro for Elasticsearch documentation.

## Naming Restrictions for Indices

Elasticsearch indices have the following naming restrictions:

- All letters must be lowercase.
- Index names cannot begin with _ or -.
- Index names can't contain spaces, commas, `:`, ", *, +, /, \, |, ?, #, >, or <.

Don't include sensitive information in index, type, or document ID names. Elasticsearch uses these names in its Uniform Resource Identifiers (URIs). Servers and applications often log HTTP requests, which can lead to unnecessary data exposure if URIs contain sensitive information:

```
2018-10-03T23:39:43 198.51.100.14 200 "GET https://elasticsearch_domain/dr-jane-doe/flu-
patients-2018/202-555-0100/ HTTP/1.1"
```

Even if you don't have permissions (p. 64) to view the associated JSON document, you could infer from this fake log line that one of Dr. Doe's patients with a phone number of 202-555-0100 had the flu in 2018.

## Reducing Response Size

Responses from the `_index` and `_bulk` APIs contain quite a bit of information. This information can be useful for troubleshooting requests or for implementing retry logic, but can use considerable bandwidth. In this example, indexing a 32 byte document results in a 339 byte response (including headers):

```
PUT elasticsearch_domain/more-movies/_doc/1
{"title": "Back to the Future"}
```

**Response**

```
{
  "_index": "more-movies",
  "_type": "_doc",
  "_id": "1",
  "_version": 4,
  "result": "updated",
  "_shards": {
    "total": 2,
    "successful": 2,
    "failed": 0
  },
  "_seq_no": 3,
  "_primary_term": 1
}
```

This response size might seem minimal, but if you index 1,000,000 documents per day—approximately 11.5 documents per second—339 bytes per response works out to 10.17 GB of download traffic per month.

If data transfer costs are a concern, use the `filter_path` parameter to reduce the size of the Elasticsearch response, but be careful not to filter out fields that you need in order to identify or retry failed requests. These fields vary by client. The `filter_path` parameter works for all Elasticsearch REST APIs, but is especially useful with APIs that you call frequently, such as the `_index` and `_bulk` APIs:

```
PUT elasticsearch_domain/more-movies/_doc/1?filter_path=result,_shards.total
{"title": "Back to the Future"}
```

**Response**

```
{
  "result": "updated",
  "_shards": {
    "total": 2
  }
}
```

Instead of including fields, you can exclude fields with a - prefix. `filter_path` also supports wildcards:

```
POST elasticsearch_domain/_bulk?filter_path=-took,-items.index._*
{ "index": { "_index": "more-movies", "_id": "1" } }
{"title": "Back to the Future"}
{ "index": { "_index": "more-movies", "_id": "2" } }
{"title": "Spirited Away"}
```

**Response**

```
{
  "errors": false,
  "items": [
    {
      "index": {
        "result": "updated",
        "status": 200
      }
    },
    {
      "index": {
        "result": "updated",
        "status": 200
      }
    }
  ]
}
```

# Loading Streaming Data into Amazon Elasticsearch Service

You can load streaming data into your Amazon Elasticsearch Service domain from many different sources. Some sources, like Amazon Kinesis Data Firehose and Amazon CloudWatch Logs, have built-in support for Amazon ES. Others, like Amazon S3, Amazon Kinesis Data Streams, and Amazon DynamoDB, use AWS Lambda functions as event handlers. The Lambda functions respond to new data by processing it and streaming it to your domain.

**Note**
Lambda supports several popular programming languages and is available in most AWS Regions. For more information, see Building Lambda Functions in the *AWS Lambda Developer Guide* and AWS Lambda Regions in the *AWS General Reference*.

**Topics**

## Loading Streaming Data into Amazon ES from Amazon S3

You can use Lambda to send data to your Amazon ES domain from Amazon S3. New data that arrives in an S3 bucket triggers an event notification to Lambda, which then runs your custom code to perform the indexing.

This method of streaming data is extremely flexible. You can index object metadata, or if the object is plaintext, parse and index some elements of the object body. This section includes some unsophisticated Python sample code that uses regular expressions to parse a log file and index the matches.

**Tip**
For more robust code in Node.js, see amazon-elasticsearch-lambda-samples on GitHub. Some Lambda blueprints also contain useful parsing examples.

### Prerequisites

Before proceeding, you must have the following resources.

| Prerequisite | Description |
|---|---|
| Amazon S3 Bucket | For more information, see Creating a Bucket in the *Amazon Simple Storage Service Getting Started Guide*. The bucket must reside in the same region as your Amazon ES domain. |
| Amazon ES Domain | The destination for data after your Lambda function processes it. For more information, see Creating Amazon ES Domains (p. 9). |

## Creating the Lambda Deployment Package

Deployment packages are ZIP or JAR files that contain your code and its dependencies. This section includes Python sample code. For other programming languages, see Creating a Deployment Package in the *AWS Lambda Developer Guide*.

1. Create a directory. In this sample, we use the name `s3-to-es`.
2. Create a file in the directory named `sample.py`:

```
import boto3
import re
import requests
from requests_aws4auth import AWS4Auth

region = '' # e.g. us-west-1
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

host = '' # the Amazon ES domain, including https://
index = 'lambda-s3-index'
type = 'lambda-type'
url = host + '/' + index + '/' + type

headers = { "Content-Type": "application/json" }

s3 = boto3.client('s3')

# Regular expressions used to parse some simple log lines
ip_pattern = re.compile('(\d+\.\d+\.\d+\.\d+)')
time_pattern = re.compile('\[(\d+\/\w\w\w\/\d\d\d\d:\d\d:\d\d:\d\d\s-\d\d\d\d)\]')
message_pattern = re.compile('\"(.+)\"')

# Lambda execution starts here
def handler(event, context):
    for record in event['Records']:

        # Get the bucket name and key for the new file
        bucket = record['s3']['bucket']['name']
        key = record['s3']['object']['key']

        # Get, read, and split the file into lines
        obj = s3.get_object(Bucket=bucket, Key=key)
        body = obj['Body'].read()
        lines = body.splitlines()

        # Match the regular expressions to each line and index the JSON
        for line in lines:
            ip = ip_pattern.search(line).group(1)
            timestamp = time_pattern.search(line).group(1)
            message = message_pattern.search(line).group(1)

            document = { "ip": ip, "timestamp": timestamp, "message": message }
            r = requests.post(url, auth=awsauth, json=document, headers=headers)
```

   Edit the variables for `region` and `host`.

3. Install dependencies:

```
cd s3-to-es
pip install requests -t .
pip install requests_aws4auth -t .
```

All Lambda execution environments have Boto3 installed, so you don't need to include it in your deployment package.

**Tip**
If you use macOS, these commands might not work properly. As a workaround, add a file named `setup.cfg` to the `s3-to-es` directory:

```
[install]
prefix=
```

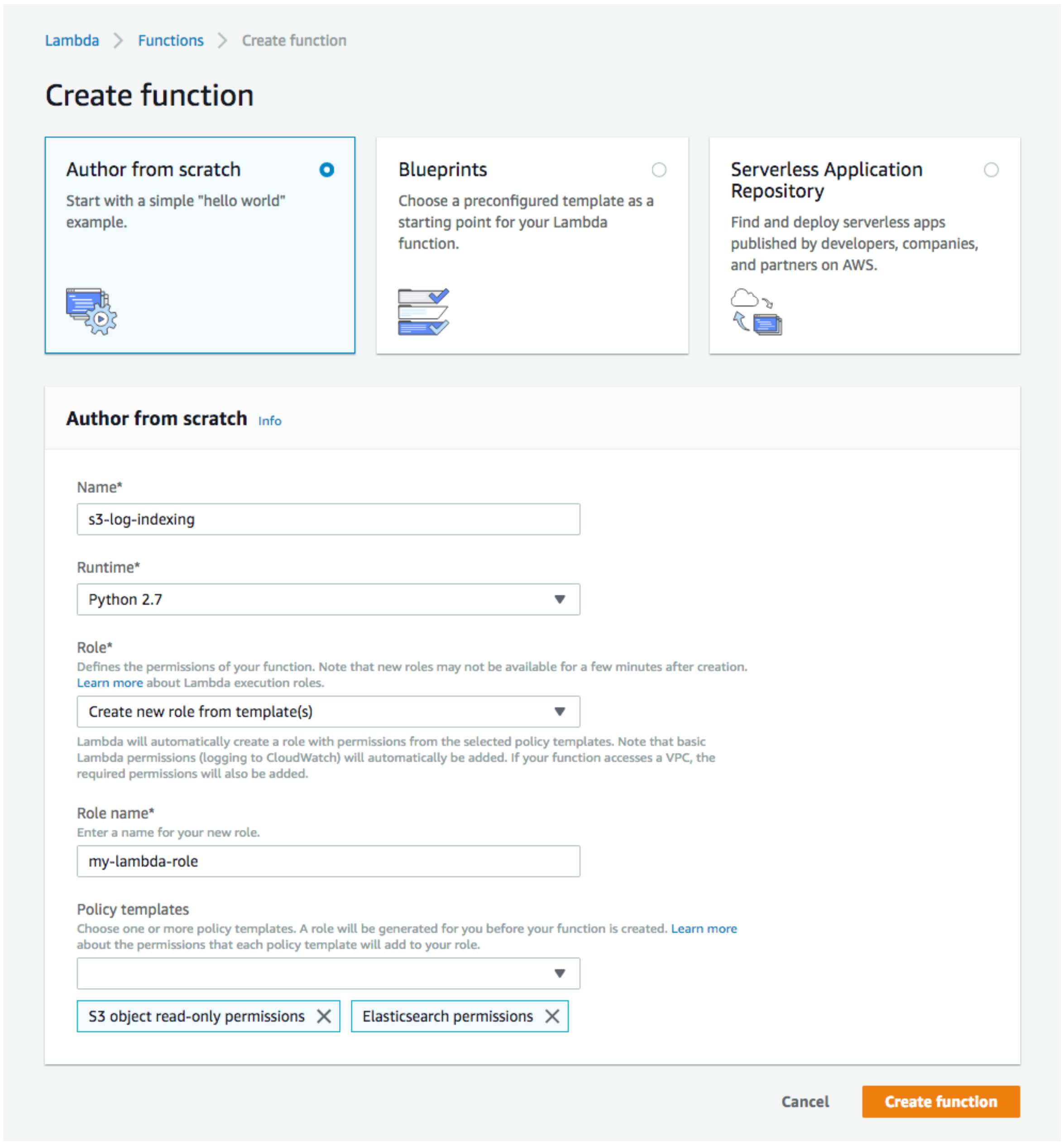
4. Package the application code and dependencies:

```
zip -r lambda.zip *
```

## Creating the Lambda Function

After you create the deployment package, you can create the Lambda function. When you create a function, choose a name, runtime (for example, Python 2.7), and IAM role. The IAM role defines the permissions for your function. For detailed instructions, see Create a Simple Lambda Function in the *AWS Lambda Developer Guide*.

This example assumes that you are using the console. Choose Python 2.7 and a role that has S3 read permissions and Amazon ES write permissions, as shown in the following screenshot.


Lambda > Functions > Create function
Create function
Author from scratch
Start with a simple "hello world" example.
Blueprints
Choose a preconfigured template as a starting point for your Lambda function.
Serverless Application Repository
Find and deploy serverless apps published by developers, companies, and partners on AWS.
Author from scratch Info

Name*
s3-log-indexing

Runtime*
Python 2.7

Role*
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. Learn more about Lambda execution roles.
Create new role from template(s)
Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name*
Enter a name for your new role.
my-lambda-role

Policy templates
Choose one or more policy templates. A role will be generated for you before your function is created. Learn more about the permissions that each policy template will add to your role.
S3 object read-only permissions
Elasticsearch permissions

After you create the function, you must add a trigger. For this example, we want the code to execute whenever a log file arrives in an S3 bucket:

1. Choose S3.
2. Choose your bucket.
3. For **Event type**, choose **PUT**.
4. For **Prefix**, type `logs/`.
5. For **Filter pattern**, type `.log`.
6. Select **Enable trigger**.
7. Choose **Add**.

Finally, you can upload your deployment package:

1. For **Handler**, type `sample.handler`. This setting tells Lambda the file (`sample.py`) and method (`handler`) that it should execute after a trigger.
2. For **Code entry type**, choose **Upload a .ZIP file**, and then follow the prompts to upload your deployment package.
3. Choose **Save**.

At this point, you have a complete set of resources: a bucket for log files, a function that executes whenever a log file is added to the bucket, code that performs the parsing and indexing, and an Amazon ES domain for searching and visualization.

## Testing the Lambda Function

After you create the function, you can test it by uploading a file to the Amazon S3 bucket. Create a file named `sample.log` using following sample log lines:

```
12.345.678.90 - [10/Oct/2000:13:55:36 -0700] "PUT /some-file.jpg"
12.345.678.91 - [10/Oct/2000:14:56:14 -0700] "GET /some-file.jpg"
```

Upload the file to the `logs` folder of your S3 bucket. For instructions, see Add an Object to a Bucket in the *Amazon Simple Storage Service Getting Started Guide*.

Then use the Amazon ES console or Kibana to verify that the `lambda-s3-index` index contains two documents. You can also make a standard search request:

```
GET https://es-domain/lambda-index/_search?pretty
{
  "hits" : {
    "total" : 2,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "lambda-s3-index",
        "_type" : "lambda-type",
        "_id" : "vTYXaWIBJWV_TTkEuSDg",
        "_score" : 1.0,
        "_source" : {
          "ip" : "12.345.678.91",
          "message" : "GET /some-file.jpg",
          "timestamp" : "10/Oct/2000:14:56:14 -0700"
        }
      },
      {
        "_index" : "lambda-s3-index",
        "_type" : "lambda-type",
        "_id" : "vjYmaWIBJWV_TTkEuCAB",
        "_score" : 1.0,
        "_source" : {
          "ip" : "12.345.678.90",
          "message" : "PUT /some-file.jpg",
          "timestamp" : "10/Oct/2000:13:55:36 -0700"
        }
      }
    ]
  }
}
```

# Loading Streaming Data into Amazon ES from Amazon Kinesis Data Streams

You can load streaming data from Kinesis Data Streams to Amazon ES. New data that arrives in the data stream triggers an event notification to Lambda, which then runs your custom code to perform the indexing. This section includes some unsophisticated Python sample code. For more robust code in Node.js, see amazon-elasticsearch-lambda-samples on GitHub.

## Prerequisites

Before proceeding, you must have the following resources.

| Prerequisite | Description |
|---|---|
| Amazon Kinesis Data Stream | The event source for your Lambda function. To learn more, see Kinesis Data Streams. |
| Amazon ES Domain | The destination for data after your Lambda function processes it. For more information, see Creating Amazon ES Domains (p. 9). |
| IAM Role | This role must have basic Amazon ES, Kinesis, and Lambda permissions, such as the following:<br>`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "es:ESHttpPost", "es:ESHttpPut", "logs:CreateLogGroup", "logs:CreateLogStream", "logs:PutLogEvents", "kinesis:GetShardIterator", "kinesis:GetRecords", "kinesis:DescribeStream", "kinesis:ListStreams" ], "Resource": "*" } ] }`<br>The role must have the following trust relationship:<br>`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "lambda.amazonaws.com" }, "Action": "sts:AssumeRole" } ] }` |

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "es:ESHttpPost",
        "es:ESHttpPut",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "kinesis:GetShardIterator",
        "kinesis:GetRecords",
        "kinesis:DescribeStream",
        "kinesis:ListStreams"
      ],
      "Resource": "*"
    }
  ]
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

| Prerequisite | Description |
|---|---|
| | To learn more, see Creating IAM Roles in the *IAM User Guide*. |

## Creating the Lambda Function

Follow the instructions in the section called “Creating the Lambda Deployment Package” (p. 133), but create a directory named `kinesis-to-es` and use the following code for `sample.py`:

```
import base64
import boto3
import json
import requests
from requests_aws4auth import AWS4Auth

region = '' # e.g. us-west-1
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

host = '' # the Amazon ES domain, including https://
index = 'lambda-kine-index'
type = 'lambda-kine-type'
url = host + '/' + index + '/' + type + '/'

headers = { "Content-Type": "application/json" }

def handler(event, context):
    count = 0
    for record in event['Records']:
        id = record['eventID']
        timestamp = record['kinesis']['approximateArrivalTimestamp']

        # Kinesis data is base64-encoded, so decode here
        message = base64.b64decode(record['kinesis']['data'])

        # Create the JSON document
        document = { "id": id, "timestamp": timestamp, "message": message }
        # Index the document
        r = requests.put(url + id, auth=awsauth, json=document, headers=headers)
        count += 1
    return 'Processed ' + str(count) + ' items.'
```

Edit the variables for `region` and `host`.

Use the following commands to install your dependencies:

```
cd kinesis-to-es
pip install requests -t .
pip install requests_aws4auth -t .
```

Then follow the instructions in the section called “Creating the Lambda Function” (p. 134), but specify the IAM role from the section called “Prerequisites” (p. 137) and the following settings for the trigger:

- **Kinesis stream**: your Kinesis stream
- **Batch size**: 100
- **Starting position**: Trim horizon

To learn more, see Working with Amazon Kinesis Data Streams in the *Amazon Kinesis Data Streams Developer Guide*.

At this point, you have a complete set of resources: a Kinesis data stream, a function that executes after the stream receives new data and indexes that data, and an Amazon ES domain for searching and visualization.

### Testing the Lambda Function

After you create the function, you can test it by adding a new record to the data stream using the AWS CLI:

```
aws kinesis put-record --stream-name es-test --data "My test data." --partition-key
 partitionKey1 --region us-west-1
```

Then use the Amazon ES console or Kibana to verify that `lambda-kine-index` contains a document. You can also use the following request:

```
GET https://es-domain/lambda-kine-index/_search
{
  "hits" : [
    {
      "_index": "lambda-kine-index",
      "_type": "lambda-kine-type",
      "_id":
 "shardId-000000000000:49583511615762699495012960821421456686529436680496087042",
      "_score": 1,
      "_source": {
        "timestamp": 1523648740.051,
        "message": "My test data.",
        "id":
 "shardId-000000000000:49583511615762699495012960821421456686529436680496087042"
      }
    }
  ]
}
```

## Loading Streaming Data into Amazon ES from Amazon DynamoDB

You can use AWS Lambda to send data to your Amazon ES domain from Amazon DynamoDB. New data that arrives in the database table triggers an event notification to Lambda, which then runs your custom code to perform the indexing.

### Prerequisites

Before proceeding, you must have the following resources.

| Prerequisite | Description |
|---|---|
| DynamoDB Table | The table contains your source data. For more information, see Basic Operations for Tables in the *Amazon DynamoDB Developer Guide*.<br><br>The table must reside in the same region as your Amazon ES domain and have a stream set to **New image**. To learn more, see Enabling a Stream. |

| Prerequisite | Description |
|---|---|
| Amazon ES Domain | The destination for data after your Lambda function processes it. For more information, see Creating Amazon ES Domains (p. 9). |
| IAM Role | This role must have basic Amazon ES, DynamoDB, and Lambda execution permissions, such as the following: <br>`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "es:ESHttpPost", "es:ESHttpPut", "dynamodb:DescribeStream", "dynamodb:GetRecords", "dynamodb:GetShardIterator", "dynamodb:ListStreams", "logs:CreateLogGroup", "logs:CreateLogStream", "logs:PutLogEvents" ], "Resource": "*" } ] }` <br>The role must have the following trust relationship: <br>`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "lambda.amazonaws.com" }, "Action": "sts:AssumeRole" } ] }` <br>To learn more, see Creating IAM Roles in the *IAM User Guide*. |

## Creating the Lambda Function

Follow the instructions in the section called “Creating the Lambda Deployment Package” (p. 133), but create a directory named `ddb-to-es` and use the following code for `sample.py`:

```
import boto3
import requests
from requests_aws4auth import AWS4Auth

region = '' # e.g. us-east-1
service = 'es'
credentials = boto3.Session().get_credentials()
```

```
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

host = '' # the Amazon ES domain, with https://
index = 'lambda-index'
type = 'lambda-type'
url = host + '/' + index + '/' + type + '/'

headers = { "Content-Type": "application/json" }

def handler(event, context):
    count = 0
    for record in event['Records']:
        # Get the primary key for use as the Elasticsearch ID
        id = record['dynamodb']['Keys']['id']['S']

        if record['eventName'] == 'REMOVE':
            r = requests.delete(url + id, auth=awsauth)
        else:
            document = record['dynamodb']['NewImage']
            r = requests.put(url + id, auth=awsauth, json=document, headers=headers)
        count += 1
    return str(count) + ' records processed.'
```

Edit the variables for `region` and `host`.

Use the following commands to install your dependencies:

```
cd ddb-to-es
pip install requests -t .
pip install requests_aws4auth -t .
```

Then follow the instructions in the section called “Creating the Lambda Function” (p. 134), but specify the IAM role from the section called “Prerequisites” (p. 139) and the following settings for the trigger:

- **Table**: your DynamoDB table
- **Batch size**: 100
- **Starting position**: Trim horizon

To learn more, see Processing New Items in a DynamoDB Table in the *Amazon DynamoDB Developer Guide*.

At this point, you have a complete set of resources: a DynamoDB table for your source data, a DynamoDB stream of changes to the table, a function that executes after your source data changes and indexes those changes, and an Amazon ES domain for searching and visualization.

## Testing the Lambda Function

After you create the function, you can test it by adding a new item to the DynamoDB table using the AWS CLI:

```
aws dynamodb put-item --table-name es-test --item '{"director": {"S": "Kevin
 Costner"},"id": {"S": "00001"},"title": {"S": "The Postman"}}' --region us-west-1
```

Then use the Amazon ES console or Kibana to verify that `lambda-index` contains a document. You can also use the following request:

```
GET https://es-domain/lambda-index/lambda-type/00001
```

```
{
    "_index": "lambda-index",
    "_type": "lambda-type",
    "_id": "00001",
    "_version": 1,
    "found": true,
    "_source": {
        "director": {
            "S": "Kevin Costner"
        },
        "id": {
            "S": "00001"
        },
        "title": {
            "S": "The Postman"
        }
    }
}
```

### Loading Streaming Data into Amazon ES from Amazon Kinesis Data Firehose

Kinesis Data Firehose supports Amazon ES as a delivery destination. For instructions about how to load streaming data into Amazon ES, see Creating a Kinesis Data Firehose Delivery Stream and Choose Amazon ES for Your Destination in the *Amazon Kinesis Data Firehose Developer Guide*.

Before you load data into Amazon ES, you might need to perform transforms on the data. To learn more about using Lambda functions to perform this task, see Data Transformation in the same guide.

As you configure a delivery stream, Kinesis Data Firehose features a "one-click" IAM role that gives it the resource access it needs to send data to Amazon ES, back up data on Amazon S3, and transform data using Lambda. Because of the complexity involved in creating such a role manually, we recommend using the provided role.

### Loading Streaming Data into Amazon ES from Amazon CloudWatch

You can load streaming data from CloudWatch Logs to your Amazon ES domain by using a CloudWatch Logs subscription. For information about Amazon CloudWatch subscriptions, see Real-time Processing of Log Data with Subscriptions. For configuration information, see Streaming CloudWatch Logs Data to Amazon Elasticsearch Service in the *Amazon CloudWatch Developer Guide*.

### Loading Data into Amazon ES from AWS IoT

You can send data from AWS IoT using rules. To learn more, see Amazon ES Action in the *AWS IoT Developer Guide*.

## Loading Data with Logstash

The open source version of Logstash (Logstash OSS) provides a convenient way to use the bulk API to upload data into your Amazon ES domain. The service supports all standard Logstash input plugins, including the Amazon S3 input plugin. Amazon ES supports two Logstash output plugins: the standard Elasticsearch plugin and the logstash-output-amazon-es plugin, which uses IAM credentials to sign and export Logstash events to Amazon ES.

If your Amazon ES domain uses fine-grained access control (p. 76) with HTTP basic authentication, configuration is similar to any other Elasticsearch cluster. This example configuration file takes its input from the open source version of Filebeat (Filebeat OSS).

```
input {
  beats {
    port => 5044
  }
}

output {
  elasticsearch {
    hosts => ["https://domain-endpoint:443"]
    ssl => true
    index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
    user => "some-user"
    password => "some-user-password"
    ilm_enabled => false
  }
}
```

If your domain uses an IAM-based domain access policy or fine-grained access control with an IAM master user, you must sign all requests to Amazon ES using IAM credentials. In this case, the simplest solution to sign requests from Logstash is to use the logstash-output-amazon-es plugin. First, install the plugin.

```
bin/logstash-plugin install logstash-output-amazon_es
```

Then export your IAM credentials (or run `aws configure`).

```
export AWS_ACCESS_KEY_ID="your-access-key"
export AWS_SECRET_ACCESS_KEY="your-secret-key"
export AWS_SESSION_TOKEN="your-session-token"
```

Finally, change your configuration file to use the plugin for its output. This example configuration file takes its input from files in an S3 bucket.

```
input {
  s3 {
    bucket => "my-s3-bucket"
    region => "us-east-1"
  }
}

output {
  amazon_es {
    hosts => ["domain-endpoint"]
    ssl => true
    region => "us-east-1"
    index => "production-logs-%{+YYYY.MM.dd}"
  }
}
```

If your Amazon ES domain is in a VPC, Logstash must run on a machine in that same VPC and have access to the domain through the VPC security groups. For more information, see the section called “About Access Policies on VPC Domains” (p. 24).

# Searching Data in Amazon Elasticsearch Service

This chapter introduces search and covers several Amazon Elasticsearch Service features that improve the experience. For more comprehensive information on the Elasticsearch search API, see the Open Distro for Elasticsearch documentation.

**Note**
All example requests in this chapter work with the Elasticsearch 6.*x* and 7.*x* APIs. Some requests might not work with older Elasticsearch versions.

## URI Searches

Universal Resource Identifier (URI) searches are the simplest form of search. In a URI search, you specify the query as an HTTP request parameter:

```
GET https://search-my-domain.us-west-1.es.amazonaws.com/_search?q=house
```

A sample response might look like the following:

```
{
  "took": 25,
  "timed_out": false,
  "_shards": {
    "total": 10,
    "successful": 10,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 85,
    "max_score": 6.6137657,
    "hits": [
      {
        "_index": "movies",
        "_type": "movie",
        "_id": "tt0077975",
        "_score": 6.6137657,
        "_source": {
          "directors": [
            "John Landis"
          ],
          "release_date": "1978-07-27T00:00:00Z",
          "rating": 7.5,
          "genres": [
            "Comedy",
            "Romance"
          ],
          "image_url": "http://ia.media-imdb.com/images/M/
MV5BMTY2OTQxNTc1OF5BMl5BanBnXkFtZTYwNjA3NjI5._V1_SX400_.jpg",
          "plot": "At a 1962 College, Dean Vernon Wormer is determined to expel the entire
 Delta Tau Chi Fraternity, but those troublemakers have other plans for him.",
          "title": "Animal House",
          "rank": 527,
          "running_time_secs": 6540,
          "actors": [
            "John Belushi",
            "Karen Allen",
            "Tom Hulce"
          ],
          "year": 1978,
          "id": "tt0077975"
        }
      },
      ...
    ]
  }
}
```

By default, this query searches all fields of all indices for the term *house*. To narrow the search, specify an index (`movies`) and a document field (`title`) in the URI:

```
GET https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search?q=title:house
```

You can include additional parameters in the request, but the supported parameters provide only a small subset of the Elasticsearch search options. The following request returns 20 results (instead of the default of 10) and sorts by year (rather than by `_score`):

```
GET https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search?
q=title:house&size=20&sort=year:desc
```

## Request Body Searches

To perform more complex searches, use the HTTP request body and the Elasticsearch domain-specific language (DSL) for queries. The query DSL lets you specify the full range of Elasticsearch search options. The following `match` query is similar to the final URI search (p. 144) example:

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search
{
  "size": 20,
  "sort": {
    "year": {
      "order": "desc"
    }
  },
  "query": {
    "query_string": {
      "default_field": "title",
      "query": "house"
    }
  }
}
```

**Note**
The `_search` API accepts HTTP `GET` and `POST` for request body searches, but not all HTTP clients support adding a request body to a `GET` request. `POST` is the more universal choice.

In many cases, you might want to search several fields, but not all fields. Use the `multi_match` query:

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search
{
  "size": 20,
  "query": {
    "multi_match": {
      "query": "house",
      "fields": ["title", "plot", "actors", "directors"]
    }
  }
}
```

## Boosting Fields

You can improve search relevancy by "boosting" certain fields. Boosts are multipliers that weigh matches in one field more heavily than matches in other fields. In the following example, a match for *john* in the `title` field influences `_score` twice as much as a match in the `plot` field and four times as much as a match in the `actors` or `directors` fields. The result is that films like *John Wick* and *John Carter* are near the top of the search results, and films starring John Travolta are near the bottom.

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search
{
  "size": 20,
  "query": {
    "multi_match": {
      "query": "john",
      "fields": ["title^4", "plot^2", "actors", "directors"]
    }
  }
}
```

## Paginating Search Results

If you need to display a large number of search results, you can implement pagination using the `from` parameter. The following request returns results 20–39 of the zero-indexed list of search results:

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search
{
  "from": 20,
  "size": 20,
  "query": {
    "multi_match": {
      "query": "house",
      "fields": ["title^4", "plot^2", "actors", "directors"]
    }
  }
}
```

## Search Result Highlighting

The `highlight` option tells Elasticsearch to return an additional object inside of the `hits` array if the query matched one or more fields:

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search
{
  "size": 20,
  "query": {
    "multi_match": {
      "query": "house",
      "fields": ["title^4", "plot^2", "actors", "directors"]
    }
  },
  "highlight": {
    "fields": {
      "plot": {}
    }
  }
}
```

If the query matched the content of the `plot` field, a hit might look like the following:

```
{
  "_index": "movies",
  "_type": "movie",
  "_id": "tt0091541",
  "_score": 11.276199,
  "_source": {
    "directors": [
      "Richard Benjamin"
    ],
    "release_date": "1986-03-26T00:00:00Z",
    "rating": 6,
    "genres": [
      "Comedy",
      "Music"
    ],
    "image_url": "http://ia.media-imdb.com/images/M/
MV5BMTIzODEzODE2OF5BMl5BanBnXkFtZTcwNjQ3ODcyMQ@@._V1_SX400_.jpg",
    "plot": "A young couple struggles to repair a hopelessly dilapidated house.",
    "title": "The Money Pit",
    "rank": 4095,
    "running_time_secs": 5460,
    "actors": [
      "Tom Hanks",
      "Shelley Long",
      "Alexander Godunov"
    ],
    "year": 1986,
    "id": "tt0091541"
  },
  "highlight": {
    "plot": [
      "A young couple struggles to repair a hopelessly dilapidated <em>house</em>."
    ]
  }
}
```

By default, Elasticsearch wraps the matching string in `<em>` tags, provides up to 100 characters of context around the match, and breaks content into sentences by identifying punctuation marks, spaces, tabs, and line breaks. All of these settings are customizable:

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_search
{
  "size": 20,
  "query": {
    "multi_match": {
      "query": "house",
      "fields": ["title^4", "plot^2", "actors", "directors"]
    }
  },
  "highlight": {
    "fields": {
      "plot": {}
    },
    "pre_tags": "<strong>",
    "post_tags": "</strong>",
    "fragment_size": 200,
    "boundary_chars": ".,!? "
  }
}
```

## Count API

If you're not interested in the contents of your documents and just want to know the number of matches, you can use the `_count` API instead of the `_search` API. The following request uses the `query_string` query to identify romantic comedies:

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/movies/_count
{
  "query": {
    "query_string": {
      "default_field": "genres",
      "query": "romance AND comedy"
    }
  }
}
```

A sample response might look like the following:

```
{
  "count": 564,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  }
}
```

# Custom Packages for Amazon Elasticsearch Service

Amazon Elasticsearch Service lets you upload custom dictionary files (for example, stop words and synonyms) for use with your cluster. The generic term for these types of files is *packages*. Dictionary files improve your search results by telling Elasticsearch to ignore certain high-frequency words or to treat terms like "frozen custard," "gelato," and "ice cream" as equivalent. They can also improve stemming, such as in the Japanese (kuromoji) Analysis plugin.

**Topics**

## Uploading Packages to Amazon S3

Before you can associate a package with your domain, you must upload it to an Amazon S3 bucket. For instructions, see Uploading S3 Objects in the *Amazon Simple Storage Service Getting Started Guide*.

If your package contains sensitive information, specify server-side encryption with S3-managed keys when you upload it. Amazon ES can't access files on S3 that you protect using an AWS KMS master key.

After you upload the file, make note of its S3 path. The path format is `s3://bucket-name/file-path/file-name`.

You can use the following synonyms file for testing purposes. Save it as `synonyms.txt`.

```
danish, croissant, pastry
ice cream, gelato, frozen custard
sneaker, tennis shoe, running shoe
basketball shoe, hightop
```

## Importing and Associating Packages

The console is the simplest way to import a package into Amazon ES and associate the package with a domain. When you import a package from Amazon S3, Amazon ES stores its own copy of the package and automatically encrypts that copy using AES-256 with Amazon ES-managed keys.

### To import and associate a package with a domain (console)

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. In the navigation pane, choose **Packages**.
4. Choose **Import**.
5. Give the package a descriptive name.
6. Provide the S3 path to the file, and then choose **Import**.
7. Return to the **Packages** screen.
8. When the package status is **Available**, select it. Then choose **Associate to a domain**.
9. Choose a domain, and then choose **Associate**.
10. In the navigation pane, choose your domain, and then choose the **Packages** tab.
11. When the package status is **Available**, note its ID. Use `analyzers/id` as the file path in requests to Elasticsearch (p. 149).

Alternately, use the AWS CLI, SDKs, or configuration API to import and associate packages. For more information, see the AWS CLI Command Reference and *Amazon ES Configuration API Reference* (p. 236).

## Using Custom Packages with Elasticsearch

After you associate a file with a domain, you can use it in parameters such as `synonyms_path`, `stopwords_path`, and `user_dictionary` when you create tokenizers and token filters. The exact parameter varies by object. The following example request adds a synonym file to a new index:

```
PUT my-index
{
  "settings": {
    "index": {
      "analysis": {
        "analyzer": {
          "synonym_analyzer": {
            "type": "custom",
            "tokenizer": "standard",
            "filter": ["synonym_filter"]
          }
        },
        "filter": {
          "synonym_filter": {
            "type": "synonym",
            "synonyms_path": "analyzers/F111111111"
          }
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "description": {
        "type": "text",
        "analyzer": "synonym_analyzer"
      }
    }
  }
}
```

This request creates a custom analyzer for the index that uses the standard tokenizer and a synonym token filter.

- Tokenizers break streams of characters into *tokens* (typically words) based on some set of rules. The simplest example is the whitespace tokenizer, which breaks the preceding characters into a token each time it encounters a whitespace character. A more complex example is the standard tokenizer, which uses a set of grammar-based rules to work across many languages.
- Token filters add, modify, or delete tokens. For example, a synonym token filter adds tokens when it finds a word in the synonyms list. The stop token filter removes tokens when finds a word in the stop words list.

This request also adds a text field (`description`) to the mapping and tells Elasticsearch to use the new analyzer for that field.

For testing purposes, add some documents to the index:

```
POST _bulk
{ "index": { "_index": "my-index", "_id": "1" } }
{ "description": "ice cream" }
{ "index": { "_index": "my-index", "_id": "2" } }
{ "description": "croissant" }
{ "index": { "_index": "my-index", "_id": "3" } }
{ "description": "tennis shoe" }
{ "index": { "_index": "my-index", "_id": "4" } }
{ "description": "hightop" }
```

Then search them using a synonym:

```
GET my-index/_search
{
  "query": {
    "match": {
      "description": "gelato"
    }
  }
}
```

In this case, Elasticsearch returns the following response:

```
{
  "hits": {
    "total": {
      "value": 1,
      "relation": "eq"
    },
    "max_score": 0.99463606,
    "hits": [{
      "_index": "my-index",
      "_type": "_doc",
      "_id": "1",
      "_score": 0.99463606,
      "_source": {
        "description": "ice cream"
      }
    }]
  }
}
```

## Updating Custom Packages

Uploading a new version of a package to Amazon S3 does *not* automatically update the package on Amazon Elasticsearch Service. Amazon ES stores its own copy of the file, so if you upload a new version to S3, you must import the file into Amazon ES again and associate it with your domains (p. 149).

After you associate the updated file with your domain, you can use it with new indices by using the requests in the section called “Using Custom Packages with Elasticsearch” (p. 149).

If you want to use the updated file with existing indices though, you must reindex them. First, create an index that uses the updated synonyms file:

```
PUT my-new-index
{
  "settings": {
    "index": {
      "analysis": {
        "analyzer": {
          "synonym_analyzer": {
            "type": "custom",
            "tokenizer": "standard",
            "filter": ["synonym_filter"]
          }
        },
        "filter": {
          "synonym_filter": {
            "type": "synonym",
            "synonyms_path": "analyzers/F222222222"
          }
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "description": {
        "type": "text",
        "analyzer": "synonym_analyzer"
      }
    }
  }
}
```

Then reindex the old index to that new index:

```
POST _reindex
{
  "source": {
    "index": "my-index"
  },
  "dest": {
    "index": "my-new-index"
  }
}
```

If you frequently update synonym files, use index aliases to maintain a consistent path to the latest index:

```
POST _aliases
{
  "actions": [
    {
      "remove": {
        "index": "my-index",
        "alias": "latest-index"
      }
    },
    {
      "add": {
        "index": "my-new-index",
        "alias": "latest-index"
      }
    }
  ]
}
```

If you don't need the old index, delete it. If you no longer need the older version of the package, dissociate and remove it (p. 152).

## Dissociating and Removing Packages

Dissociating a package from a domain means that you can no longer use that file when you create new indices. Any indices that already use the file can continue using it.

The console is the simplest way to dissociate a package from a domain and remove it from Amazon ES. Removing a package from Amazon ES does *not* remove it from its original location on Amazon S3.

### To dissociate a package from a domain and remove it from Amazon ES (console)

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. In the navigation pane, choose your domain, and then choose the **Packages** tab.
4. Choose a package, **Actions**, and then choose **Dissociate**. Confirm your choice.
5. Wait for the package to disappear from the list. You might need to refresh your browser.
6. If you want to use the package with other domains, stop here. To continue with removing the package, choose **Packages** in the navigation pane.
7. Select the package and choose **Delete**.

Alternately, use the AWS CLI, SDKs, or configuration API to dissociate and remove packages. For more information, see the AWS CLI Command Reference and *Amazon ES Configuration API Reference* (p. 236).

# SQL Support for Amazon Elasticsearch Service

SQL support for Amazon Elasticsearch Service lets you query your data using SQL rather than the JSON-based Elasticsearch query DSL. This feature is useful if you're already familiar with SQL or want to integrate your domain with an application that uses SQL.

SQL support is available on domains running Elasticsearch 6.5 or higher. Full documentation is available in the Open Distro for Elasticsearch documentation.

## Sample Call

To query your data using SQL, send HTTP requests to `_opendistro/_sql` using the following format:

```
POST elasticsearch_domain/_opendistro/_sql
{
  "query": "SELECT * FROM my-index LIMIT 50"
}
```

## Notes and Differences

Calls to `_opendistro/_sql` include index names in the request body and thus have the same access policy considerations (p. 73) as the bulk, mget, and msearch APIs. As always, follow the principle of least privilege when granting permissions to APIs.

For a security consideration regarding using SQL with fine-grained access control, see Fine-Grained Access Control in Amazon Elasticsearch Service (p. 90).

## JDBC Driver

The Java Database Connectivity (JDBC) driver lets you integrate Amazon ES domains with your favorite business intelligence (BI) applications. To get started, see the GitHub repository. The following table summarizes version compatibility for the driver.

| Elasticsearch Version | JDBC Driver Version |
|---|---|
| 7.4 | 1.4.0 |
| 7.1 | 1.0.0 |
| 6.8 | 0.9.0 |
| 6.7 | 0.9.0 |
| 6.5 | 0.9.0 |

# KNN

Short for its associated *k-nearest neighbors* algorithm, KNN for Amazon Elasticsearch Service lets you search for points in a vector space and find the "nearest neighbors" for those points by Euclidean distance. Use cases include recommendations (for example, an "other songs you might like" feature in a music application), image recognition, and fraud detection.

KNN requires Elasticsearch 7.1 or later. Full documentation for the Elasticsearch feature, including descriptions of settings and statistics, is available in the Open Distro for Elasticsearch documentation. For background information about the k-nearest neighbors algorithm, see Wikipedia.

## Getting Started with KNN

To use KNN, you must create an index with the `index.knn` setting and add one or more fields of the `knn_vector` data type.

```
PUT my-index
{
  "settings": {
    "index.knn": true
  },
  "mappings": {
    "properties": {
      "my_vector1": {
        "type": "knn_vector",
        "dimension": 2
      },
      "my_vector2": {
        "type": "knn_vector",
        "dimension": 4
      }
    }
  }
}
```

The `knn_vector` data type supports a single list of up to 10,000 floats, with the number of floats defined by the required `dimension` parameter. After you create the index, add some data to it.

```
POST _bulk
{ "index": { "_index": "my-index", "_id": "1" } }
{ "my_vector1": [1.5, 2.5], "price": 12.2 }
{ "index": { "_index": "my-index", "_id": "2" } }
{ "my_vector1": [2.5, 3.5], "price": 7.1 }
{ "index": { "_index": "my-index", "_id": "3" } }
{ "my_vector1": [3.5, 4.5], "price": 12.9 }
{ "index": { "_index": "my-index", "_id": "4" } }
{ "my_vector1": [5.5, 6.5], "price": 1.2 }
{ "index": { "_index": "my-index", "_id": "5" } }
{ "my_vector1": [4.5, 5.5], "price": 3.7 }
{ "index": { "_index": "my-index", "_id": "6" } }
{ "my_vector2": [1.5, 5.5, 4.5, 6.4], "price": 10.3 }
{ "index": { "_index": "my-index", "_id": "7" } }
{ "my_vector2": [2.5, 3.5, 5.6, 6.7], "price": 5.5 }
{ "index": { "_index": "my-index", "_id": "8" } }
{ "my_vector2": [4.5, 5.5, 6.7, 3.7], "price": 4.4 }
{ "index": { "_index": "my-index", "_id": "9" } }
{ "my_vector2": [1.5, 5.5, 4.5, 6.4], "price": 8.9 }
```

Then you can search the data using the `knn` query type.

```
GET my-index/_search
{
  "size": 2,
  "query": {
    "knn": {
      "my_vector2": {
        "vector": [2, 3, 5, 6],
        "k": 2
      }
    }
  }
}
```

In this case, `k` is the number of neighbors you want the query to return, but you must also include the `size` option. Otherwise, you get `k` results for each shard (and each segment) rather than `k` results for the entire query. KNN supports a maximum `k` value of 10,000.

If you mix the `knn` query with other clauses, you might receive fewer than `k` results. In this example, the `post_filter` clause reduces the number of results from 2 to 1.

```
GET my-index/_search
{
  "size": 2,
  "query": {
    "knn": {
      "my_vector2": {
        "vector": [2, 3, 5, 6],
        "k": 2
      }
    }
  },
  "post_filter": {
    "range": {
      "price": {
        "gte": 6,
        "lte": 10
      }
    }
  }
}
```

## KNN Differences and Tuning

Open Distro for Elasticsearch lets you modify all KNN settings using the `_cluster/settings` API. On Amazon ES, you can change all settings except `knn.memory.circuit_breaker.enabled` and `knn.circuit_breaker.triggered`. KNN statistics are included as Amazon CloudWatch metrics (p. 28).

In particular, check the `KNNGraphMemoryUsage` metric on each data node against the `knn.memory.circuit_breaker.limit` statistic and the available RAM for the instance type. Amazon ES uses half of an instance's RAM for the Java heap (up to a heap size of 32 GiB). By default, KNN uses up to 60% of the remaining half, so an instance type with 32 GiB of RAM can accommodate 9.6 GiB of graphs (32 * 0.5 * 0.6). Performance can suffer if graph memory usage exceeds this value.

# Kibana

Kibana is a popular open source visualization tool designed to work with Elasticsearch. Amazon ES provides an installation of Kibana with every Amazon ES domain. You can find a link to Kibana on your domain dashboard on the Amazon ES console. The URL is `domain-endpoint/_plugin/kibana/`. Queries using this default Kibana installation have a 300-second timeout.

The following sections address some common Kibana use cases:

- the section called "Controlling Access to Kibana" (p. 156)
- the section called "Configuring Kibana to Use a WMS Map Server" (p. 159)
- the section called "Connecting a Local Kibana Server to Amazon ES" (p. 159)

## Controlling Access to Kibana

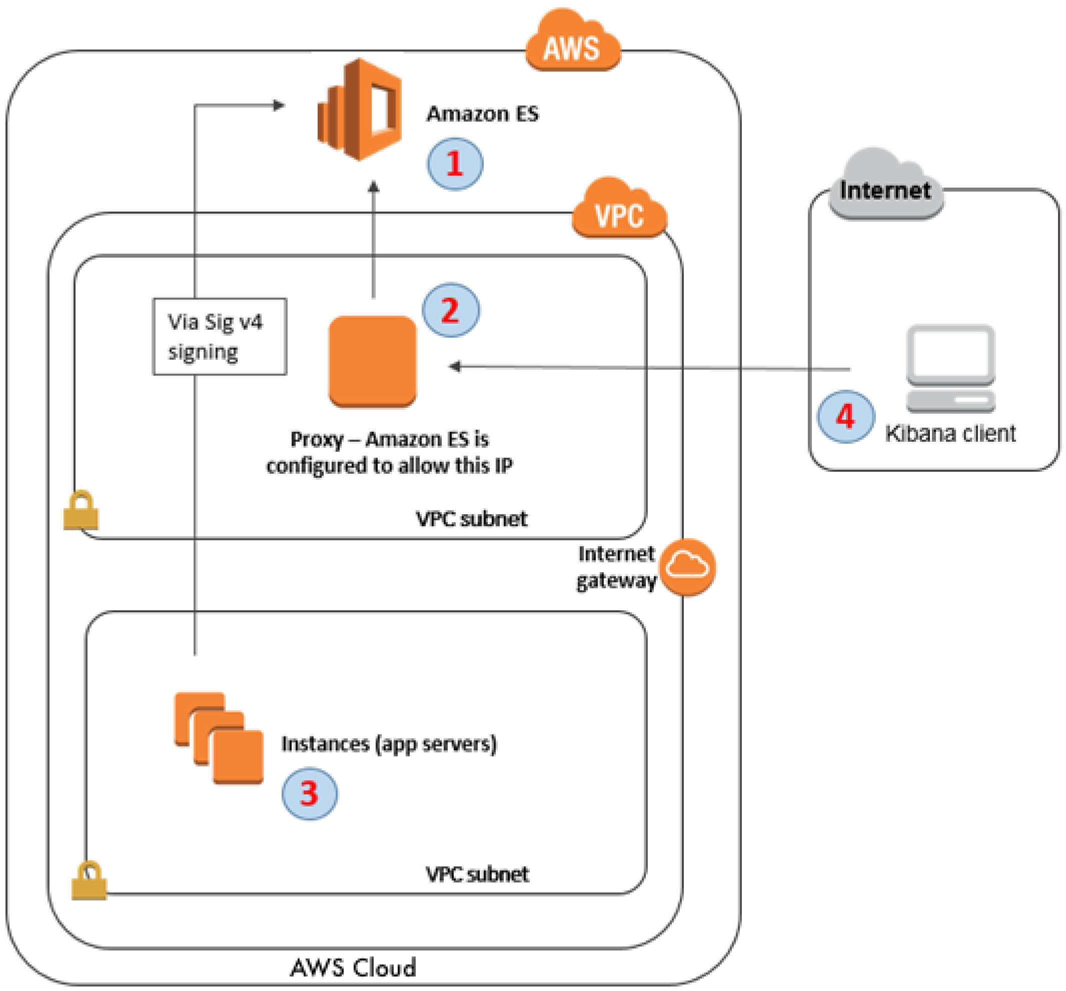
Kibana does not natively support IAM users and roles, but Amazon ES offers several solutions for controlling access to Kibana:

| Domain Configuration | Access Control Options |
|---|---|
| Public access | • Configure the section called "Authentication for Kibana" (p. 99).<br>• Configure an IP-based access policy (p. 66), with or without a proxy server (p. 156). |
| VPC access | • Configure the section called "Authentication for Kibana" (p. 99).<br>• Configure an open access policy, with or without a proxy server, and use security groups to control access. To learn more, see the section called "About Access Policies on VPC Domains" (p. 24). |

### Using a Proxy to Access Amazon ES from Kibana

**Note**
This process is only applicable if your domain uses public access and you don't want to use the section called "Authentication for Kibana" (p. 99). See the section called "Controlling Access to Kibana" (p. 156).

Because Kibana is a JavaScript application, requests originate from the user's IP address. IP-based access control might be impractical due to the sheer number of IP addresses you would need to whitelist in order for each user to have access to Kibana. One workaround is to place a proxy server between Kibana and Amazon ES. Then you can add an IP-based access policy that allows requests from only one IP address, the proxy's. The following diagram shows this configuration.



1. This is your Amazon ES domain. IAM provides authorized access to this domain. An additional, IP-based access policy provides access to the proxy server.
2. This is the proxy server, running on an Amazon EC2 instance.
3. Other applications can use the Signature Version 4 signing process to send authenticated requests to Amazon ES.
4. Kibana clients connect to your Amazon ES domain through the proxy.

To enable this sort of configuration, you need a resource-based policy that specifies roles and IP addresses. Here's a sample policy:

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Resource": "arn:aws:es:us-west-2:111111111111:domain/my-domain/*",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:role/allowedrole1"
      },
      "Action": [
        "es:ESHttpGet"
      ],
      "Effect": "Allow"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "es:*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": [
            "123.456.789.123"
          ]
        }
      },
      "Resource": "arn:aws:es:us-west-2:111111111111:domain/my-domain/*"
    }
  ]
}
```

We recommend that you configure the EC2 instance running the proxy server with an Elastic IP address. This way, you can replace the instance when necessary and still attach the same public IP address to it. To learn more, see Elastic IP Addresses in the *Amazon EC2 User Guide for Linux Instances*.

If you use a proxy server *and* the section called "Authentication for Kibana" (p. 99), you might need to add settings for Kibana and Amazon Cognito to avoid `redirect_mismatch` errors. See the following `nginx.conf` example:

```
server {
    listen 443;
    server_name $host;
    rewrite ^/$ https://$host/_plugin/kibana redirect;

    ssl_certificate           /etc/nginx/cert.crt;
    ssl_certificate_key       /etc/nginx/cert.key;

    ssl on;
    ssl_session_cache  builtin:1000  shared:SSL:10m;
    ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers HIGH:!aNULL:!eNULL:!EXPORT:!CAMELLIA:!DES:!MD5:!PSK:!RC4;
    ssl_prefer_server_ciphers on;

    location /_plugin/kibana {
        # Forward requests to Kibana
        proxy_pass https://$kibana_host/_plugin/kibana;

        # Handle redirects to Cognito
        proxy_redirect https://$cognito_host https://$host;

        # Update cookie domain and path
        proxy_cookie_domain $kibana_host $host;
        proxy_cookie_path / /_plugin/kibana/;

        # Response buffer settings
        proxy_buffer_size 128k;
        proxy_buffers 4 256k;
        proxy_busy_buffers_size 256k;
    }

    location ~ \/(log|sign|fav|forgot|change|saml|oauth2) {
        # Forward requests to Cognito
        proxy_pass https://$cognito_host;

        # Handle redirects to Kibana
        proxy_redirect https://$kibana_host https://$host;
        # Update cookie domain
        proxy_cookie_domain $cognito_host $host;
    }
}
```

## Configuring Kibana to Use a WMS Map Server

Due to licensing restrictions, the default installation of Kibana on Amazon ES domains that use Elasticsearch 5.*x* or greater does *not* include a map server for tile map visualizations. Use the following procedure to configure Kibana to use a Web Map Service (WMS) map server.

**To configure Kibana to use a WMS map server:**

1. Open Kibana. You can find a link to Kibana in the domain summary at https://console.aws.amazon.com/es/.
2. Choose **Management**.
3. Choose **Advanced Settings**.
4. Locate **visualization:tileMap:WMSdefaults**, and then choose the **edit** button to modify the default value.
5. Change `enabled` to `true` and `url` to the URL of a valid WMS map server.
6. (Optional) Locate **visualization:tileMap:WMSdefaults**, and then choose the **edit** button to modify the default value.
7. (Optional) Change `"layers": "0"` to a comma-separated list of map layers that you want to display. Layers vary by map service. The default value of `0` is often appropriate.
8. Choose the **save** button.

To apply the new default value to visualizations, you might need to reload Kibana.

**Note**
Map services often have licensing fees or restrictions. You are responsible for all such considerations on any map server that you specify. You might find the map services from the U.S. Geological Survey useful for testing.

## Connecting a Local Kibana Server to Amazon ES

If you have invested significant time into configuring your own Kibana instance, you can use it instead of (or in addition to) the default Kibana instance that Amazon ES provides.

**To connect a local Kibana server to Amazon ES:**

- Make the following changes to `config/kibana.yml`:

```
kibana.index: ".kibana_1"
# Use elasticsearch.url for versions older than 6.6
# elasticsearch.url: "https://domain-endpoint:443"
# Use elasticsearch.hosts for versions 6.6 and later
elasticsearch.hosts: "https://domain-endpoint:443"
```

Older versions of Elasticsearch might only work over HTTP. In all cases, add the `http` or `https` prefix. For older versions, you must explicitly specify port 80 or 443. For newer versions, you can omit the port.

# Managing Indices

After you add data to Amazon Elasticsearch Service, you often need to reindex that data, work with index aliases, move an index to more cost-effective storage, or delete it altogether. This chapter covers UltraWarm storage and Index State Management. For information on the Elasticsearch index APIs, see the Open Distro for Elasticsearch documentation.

**Topics**

## UltraWarm for Amazon Elasticsearch Service

UltraWarm provides a cost-effective way to store large amounts of read-only data on Amazon Elasticsearch Service. Standard data nodes use "hot" storage, which takes the form of instance stores or Amazon EBS volumes attached to each node. Hot storage provides the fastest possible performance for indexing and searching new data.

Rather than attached storage, UltraWarm nodes use Amazon S3 and a sophisticated caching solution to improve performance. For indices that you are not actively writing to and query less frequently, UltraWarm offers significantly lower costs per GiB of data. In Elasticsearch, these warm indices behave just like any other index. You can query them using the same APIs or use them to create dashboards in Kibana.

**Topics**

### Prerequisites

UltraWarm has a few important prerequisites:

- UltraWarm requires Elasticsearch 6.8 or higher.
- To use warm storage, domains must have dedicated master nodes (p. 176).
- If your domain uses a T2 instance type for your data nodes, you can't use warm storage.

## Calculating UltraWarm Storage Requirements

As covered in the section called "Calculating Storage Requirements" (p. 172), data in hot storage incurs significant overhead: replicas, Linux reserved space, and Amazon ES reserved space. For example, a 10 GiB primary shard with one replica shard requires roughly 26 GiB of hot storage.

Because it uses Amazon S3, UltraWarm incurs none of this overhead. When calculating UltraWarm storage requirements, you consider only the size of the primary shards. The durability of data in S3 removes the need for replicas, and S3 abstracts away any operating system or service considerations. That same 10 GiB shard requires 10 GiB of warm storage. If you provision an `ultrawarm1.large.elasticsearch` instance, you can use all 20 TiB of its maximum storage for primary shards. See the section called "UltraWarm Storage Limits" (p. 198) for a summary of instance types and the maximum amount of storage that each can address.

**Tip**
With UltraWarm, we still recommend a maximum shard size of 50 GiB.

## UltraWarm Pricing

With hot storage, you pay for what you provision. Some instances require an attached Amazon EBS volume, while others include an instance store. Whether that storage is empty or full, you pay the same price.

With UltraWarm storage, you pay for what you use. An `ultrawarm1.large.elasticsearch` instance can address up to 20 TiB of storage on S3, but if you store only 1 TiB of data, you're only billed for 1 TiB of data. Like all other node types, you also pay an hourly rate for each UltraWarm node. For more information, see the section called "Pricing for Amazon ES" (p. 3).

## Enabling UltraWarm

The console is the simplest way to create a domain that uses warm storage. While creating the domain, choose **Enable UltraWarm data nodes** and the number of warm nodes that you want. The same basic process works on existing domains, provided they meet the prerequisites (p. 160). Even after the domain state changes from **Processing** to **Active**, UltraWarm might not be available to use for several hours.

You can also use the AWS CLI or configuration API (p. 236) to enable UltraWarm, specifically the `WarmEnabled`, `WarmCount`, and `WarmType` options in `ElasticsearchClusterConfig`.

**Note**
Domains support a maximum number of warm nodes. For details, see the section called "Limits" (p. 197).

### Sample CLI Command

The following AWS CLI command creates a domain with three data nodes, three dedicated master nodes, and six warm nodes with a restrictive access policy:

```
aws es create-elasticsearch-domain --domain-name my-domain --elasticsearch-cluster-config
 InstanceCount=3,InstanceType=r5.large.elasticsearch,DedicatedMasterEnabled=true,DedicatedMasterType=c5
 --elasticsearch-version 6.8 --ebs-options EBSEnabled=true,VolumeType=gp2,VolumeSize=11
 --access-policies '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"AWS":["123456789012"]},"Action":["es:*"],"Resource":"arn:aws:es:us-
east-1:123456789012:domain/my-domain/*"}]}' --region us-east-1
```

For detailed information, see the AWS CLI Command Reference.

## Sample Configuration API Request

The following request to the configuration API creates a domain with three data nodes, three dedicated master nodes, and six warm nodes with all encryption features enabled and a restrictive access policy:

```
POST https://es.us-east-2.amazonaws.com/2015-01-01/es/domain
{
  "ElasticsearchClusterConfig": {
    "InstanceCount": 3,
    "InstanceType": "r5.large.elasticsearch",
    "DedicatedMasterEnabled": true,
    "DedicatedMasterType": "c5.large.elasticsearch",
    "DedicatedMasterCount": 3,
    "ZoneAwarenessEnabled": true,
    "ZoneAwarenessConfig": {
      "AvailabilityZoneCount": 3
    },
    "WarmEnabled": true,
    "WarmCount": 6,
    "WarmType": "ultrawarm1.medium.elasticsearch"
  },
  "EBSOptions": {
    "EBSEnabled": true,
    "VolumeType": "gp2",
    "VolumeSize": 11
  },
  "EncryptionAtRestOptions": {
    "Enabled": true
  },
  "NodeToNodeEncryptionOptions": {
    "Enabled": true
  },
  "DomainEndpointOptions": {
    "EnforceHTTPS": true,
    "TLSSecurityPolicy": "Policy-Min-TLS-1-2-2019-07"
  },
  "ElasticsearchVersion": "6.8",
  "DomainName": "my-domain",
  "AccessPolicies": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow
\",\"Principal\":{\"AWS\":[\"123456789012\"]},\"Action\":[\"es:*\"],\"Resource\":
\"arn:aws:es:us-east-1:123456789012:domain/my-domain/*\"}]}"
}
```

For detailed information, see *Amazon ES Configuration API Reference* (p. 236).

# Migrating Indices to UltraWarm Storage

If you finish writing to an index and no longer need the fastest possible search performance, migrate it from hot to warm:

```
POST _ultrawarm/migration/my-index/_warm
```

Then check the status of the migration:

```
GET _ultrawarm/migration/my-index/_status

{
  "migration_status": {
    "index": "my-index",
    "state": "RUNNING_SHARD_RELOCATION",
    "migration_type": "HOT_TO_WARM",
    "shard_level_status": {
      "running": 0,
      "total": 5,
      "pending": 3,
      "failed": 0,
      "succeeded": 2
    }
  }
}
```

If you migrate several indices in quick succession, you can get a summary of all migrations in plaintext, similar to the `_cat` API:

```
GET _ultrawarm/migration/_status?v

index    migration_type state
my-index HOT_TO_WARM    RUNNING_SHARD_RELOCATION
```

You can have up to 25 simultaneous migrations from hot to warm storage. To check the current number, monitor the `HotToWarmMigrationQueueSize` metric (p. 37).

The migration process has the following states:

```
PENDING_INCREMENTAL_SNAPSHOT
RUNNING_INCREMENTAL_SNAPSHOT
FAILED_INCREMENTAL_SNAPSHOT
PENDING_FORCE_MERGE
RUNNING_FORCE_MERGE
FAILED_FORCE_MERGE
PENDING_FULL_SNAPSHOT
RUNNING_FULL_SNAPSHOT
FAILED_FULL_SNAPSHOT
PENDING_SHARD_RELOCATION
RUNNING_SHARD_RELOCATION
FINISHED_SHARD_RELOCATION
```

As these states indicate, migrations might fail during snapshots, shard relocations, or force merges. Failures during snapshots or shard relocation are typically due to node failures or S3 connectivity issues. Lack of disk space is usually the underlying cause of force merge failures.

After a migration finishes, the same `_status` request returns an error. If you check the index at that time, you can see some settings that are unique to warm indices:

```
GET my-index/_settings

{
  "my-index": {
    "settings": {
      "index": {
        "refresh_interval": "-1",
        "auto_expand_replicas": "false",
        "blocks": {
          "ultrawarm_allow_delete": "true"
        },
        "provided_name": "my-index",
        "creation_date": "1572886951679",
        "unassigned": {
          "node_left": {
            "delayed_timeout": "5m"
          }
        },
        "number_of_replicas": "1",
        "uuid": "3iyTkhXvR8Cytc6sWKBirg",
        "version": {
          "created": "6080099"
        },
        "routing": {
          "allocation": {
            "require": {
              "box_type": "warm"
            }
          },
          "search_preference": "_primary_first"
        },
        "number_of_shards": "5"
      }
    }
  }
}
```

- `blocks.ultrawarm_allow_delete` specifies whether to block most `_settings` updates to the index (`true`) or allow them (`false`).
- `number_of_replicas`, in this case, is the number of passive replicas, which don't consume disk space.
- `routing.allocation.require.box_type` specifies that the index should use warm nodes rather than standard data nodes.
- `routing.search_preference` instructs Amazon ES to query primary shards first and only use passive replicas if the query fails. This setting reduces disk usage.

Indices in warm storage are read-only unless you return them to hot storage (p. 165). You can query the indices, but you can't add data to them. If you try, you encounter the following error:

```
{
  "error": {
    "root_cause": [{
      "type": "cluster_block_exception",
      "reason": "blocked by: [FORBIDDEN/12/index read-only / allow delete (api)];"
    }],
    "type": "cluster_block_exception",
    "reason": "blocked by: [FORBIDDEN/12/index read-only / allow delete (api)];"
  },
  "status": 403
}
```

## Automating Migrations

We recommend using the section called “Index State Management” (p. 166) to automate the migration process after an index reaches a certain age or meets other conditions. The sample policy here (p. 166) demonstrates that workflow.

## Listing Hot and Warm Indices

UltraWarm adds two additional options, similar to `_all`, to help manage hot and warm indices. For a list of all warm or hot indices, make the following requests:

```
GET _warm
GET _hot
```

You can use these options in other requests that specify indices, such as:

```
_cat/indices/_warm
_cluster/state/_all/_hot
```

## Returning Warm Indices to Hot Storage

If you need to write to an index again, migrate it back to hot storage:

```
POST _ultrawarm/migration/my-index/_hot
```

You can have up to 10 simultaneous migrations from warm to hot storage. To check the current number, monitor the `WarmToHotMigrationQueueSize` metric (p. 37).

After the migration finishes, check the index settings to make sure they meet your needs. Indices return to hot storage with one replica.

## Restoring Warm Indices from Snapshots

In addition to the standard repository for automated snapshots, UltraWarm adds a second repository, `cs-ultrawarm`. Snapshots in `cs-ultrawarm` have the same 14-day retention period as other automated snapshots.

Unlike other automated snapshots, each snapshot in this repository contains only one index. When you restore a snapshot from `cs-ultrawarm`, it restores to warm storage, not hot storage. Snapshots in the `cs-automated` and `cs-automated-enc` repositories restore to hot storage.

**To restore an UltraWarm snapshot to warm storage**

1. Identify the latest snapshot that contains the index that you want to restore:

   ```
   GET _snapshot/cs-ultrawarm/_all

   {
     "snapshots": [{
       "snapshot": "snapshot-name",
       "version": "6.8.0",
       "indices": [
         "my-index"
       ]
     }]
   }
   ```
2. If the index already exists, delete it:

   ```
   DELETE my-index
   ```

   If you don't want to delete the index, reindex it.
3. Restore the snapshot:

   ```
   POST _snapshot/cs-ultrawarm/snapshot-name/_restore
   ```

   UltraWarm ignores any settings you specify in this restore request, so you can omit the request body.

## Disabling UltraWarm

The console is the simplest way to disable UltraWarm. Choose the domain, **Edit domain**, uncheck **Enable UltraWarm data nodes**, and **Submit**. You can also use the `WarmEnabled` option in the AWS CLI and configuration API.

Before you disable UltraWarm, you must either delete all warm indices or migrate them back to hot storage. After warm storage is empty, wait five minutes before attempting to disable the feature.

# Index State Management

Index State Management (ISM) lets you define custom management policies to automate routine tasks and apply them to indices and index patterns. You no longer need to set up and manage external processes to run your index operations.

A policy contains a default state and a list of states for the index to transition between. Within each state, you can define a list of actions to perform and conditions that trigger these transitions. A typical use case is to periodically delete old indices after a certain period of time.

For example, you can define a policy that moves your index into a `read_only` state after 30 days and then ultimately deletes it after 90 days.

ISM requires Elasticsearch 6.8 or later. Full documentation for the feature is available in the Open Distro for Elasticsearch documentation.

> **Note**
> After you attach a policy to an index, ISM creates a job that runs every 30 to 48 minutes to perform policy actions, check conditions, and transition the index into different states. The base time for this job to run is every 30 minutes, plus a random 0-60% jitter is added to it to make sure you do not see a surge of activity from all your indices at the same time.

## Sample Policies

This first sample policy moves an index from hot storage to UltraWarm (p. 160) storage after seven days and deletes the index after 90 days.

In this case, an index is initially in the `hot` state. After seven days, ISM moves it to the `warm` state. 83 days later, the service sends a notification to an Amazon Chime room that the index is being deleted and then permanently deletes it.

```
{
  "policy": {
    "description": "Demonstrate a hot-warm-delete workflow.",
    "default_state": "hot",
    "schema_version": 1,
    "states": [{
        "name": "hot",
        "actions": [],
        "transitions": [{
          "state_name": "warm",
          "conditions": {
            "min_index_age": "7d"
          }
        }]
      },
      {
        "name": "warm",
        "actions": [{
          "warm_migration": {},
          "retry": {
            "count": 5,
            "delay": "1h"
          }
        }],
        "transitions": [{
          "state_name": "delete",
          "conditions": {
            "min_index_age": "90d"
          }
        }]
      },
      {
        "name": "delete",
        "actions": [{
            "notification": {
              "destination": {
                "chime": {
                  "url": "<URL>"
                }
              },
              "message_template": {
                "source": "The index {{ctx.index}} is being deleted."
              }
            }
          },
          {
            "delete": {}
          }
        ]
      }
    ]
  }
}
```

This second, simpler sample policy reduces replica count to zero after seven days to conserve disk space and then deletes the index after 21 days. This policy assumes your index is non-critical and no longer receiving write requests; having zero replicas carries some risk of data loss.

```
{
  "policy": {
    "description": "Changes replica count and deletes.",
    "schema_version": 1,
    "default_state": "current",
    "states": [{
        "name": "current",
        "actions": [],
        "transitions": [{
          "state_name": "old",
          "conditions": {
            "min_index_age": "7d"
          }
        }]
      },
      {
        "name": "old",
        "actions": [{
          "replica_count": {
            "number_of_replicas": 0
          }
        }],
        "transitions": [{
          "state_name": "delete",
          "conditions": {
            "min_index_age": "21d"
          }
        }]
      },
      {
        "name": "delete",
        "actions": [{
          "delete": {}
        }],
        "transitions": []
      }
    ]
  }
}
```

## Differences

Compared to Open Distro for Elasticsearch, ISM for Amazon Elasticsearch Service has several differences.

### ISM Operations

Amazon ES supports a unique ISM operation, `warm_migration`. If your domain has UltraWarm (p. 160) enabled, this action transitions the index to warm storage.

Amazon ES does not support the following ISM operations:

- `open`
- `close`

### ISM Settings

Open Distro for Elasticsearch lets you change all available ISM settings using the `_cluster/settings` API. On Amazon ES, you can only change the following settings:

- **Cluster-level settings:**
  - `enabled`
  - `history.enabled`
- **Index-level settings:**
  - `roll_over`
  - `policy_id`

# Alerting

The alerting feature notifies you when data from one or more Elasticsearch indices meets certain conditions. For example, you might want to receive an email if your application logs more than five HTTP 503 errors in one hour, or you might want to page a developer if no new documents have been indexed in the past 20 minutes. To get started, open Kibana and choose **Alerting**.

Alerting requires Elasticsearch 6.2 or higher. Full documentation for the feature is available in the Open Distro for Elasticsearch documentation.

## Differences

Compared to Open Distro for Elasticsearch, the Amazon Elasticsearch Service alerting feature has two notable differences: Amazon SNS support and fixed settings.

### Amazon SNS Support

Amazon ES supports Amazon SNS for notifications. This integration with Amazon SNS means that, in addition to standard destinations (Slack, custom webhooks, and Amazon Chime), the alerting feature can send emails, text messages, and even execute AWS Lambda functions using SNS topics. For more information about Amazon SNS, see the Amazon Simple Notification Service Developer Guide.

**To add Amazon SNS as a destination**

1. Open Kibana.
2. Choose **Alerting**.
3. Choose the **Destinations** tab and then **Add Destination**.
4. Provide a unique name for the destination.
5. For **Type**, choose **Amazon SNS**.
6. Provide the SNS topic ARN.
7. Provide the ARN for an IAM role within your account that has the following trust relationship and permissions (at minimum):

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "es.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
  }]
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "sns-topic-arn"
  }]
}
```

For more information, see Adding IAM Identity Permissions in the *IAM User Guide*.

8. Choose **Create**.

## Alerting Settings

Open Distro for Elasticsearch lets you modify certain alerting settings using the `_cluster/settings` API (for example, `opendistro.alerting.monitor.max_monitors`). Amazon ES uses the default values, and you can't change them.

You can, however, disable the alerting feature. Send the following request:

```
PUT _cluster/settings
{
  "persistent" : {
    "opendistro.scheduled_jobs.enabled" : false
  }
}
```

If you previously created monitors and want to stop the creation of daily alerting indices, delete all alert history indices:

```
DELETE .opendistro-alerting-alert-history-*
```

# Amazon Elasticsearch Service Best Practices

This chapter addresses some best practices for operating Amazon Elasticsearch Service domains and provides general guidelines that apply to many use cases. Production domains should adhere to the following standards:

- Apply a restrictive resource-based access policy (p. 64) to the domain (or enable fine-grained access control), and follow the principle of least privilege when granting access to the configuration API and the Elasticsearch APIs.
- Configure at least one replica, the Elasticsearch default, for each index.
- Use three dedicated master nodes (p. 176).
- Deploy the domain across three Availability Zones. This configuration lets Amazon ES distribute replica shards to different Availability Zones than their corresponding primary shards. For a list of Regions that have three Availability Zones and some other considerations, see the section called "Configuring a Multi-AZ Domain" (p. 17).
- Upgrade to the latest Elasticsearch versions (p. 51) as they become available on Amazon Elasticsearch Service.
- Update to the latest service software (p. 15) as it becomes available.
- Size the domain appropriately for your workload. For storage volume, shard size, and data node recommendations, see the section called "Sizing Amazon ES Domains" (p. 171) and the section called "Petabyte Scale" (p. 175). For dedicated master node recommendations, see the section called "Dedicated Master Nodes" (p. 176).
- Have no more than 1,000 shards on any data node. This limit is the default in Elasticsearch 7.*x* and later. For a more nuanced guideline, see the section called "Choosing the Number of Shards" (p. 173).
- Use the latest-generation instances available on the service. For example, use I3 instances rather than I2 instances.
- Don't use burstable instances for production domains. For example, don't use T2 instances as data nodes or dedicated master nodes.
- If appropriate for your network configuration, create the domain within a VPC (p. 21).
- If your domain stores sensitive data, enable encryption of data at rest (p. 61) and node-to-node encryption (p. 63).

For more information, see the remaining topics in this chapter.

**Topics**

## Sizing Amazon ES Domains

No surefire method of sizing Amazon ES domains exists, but by starting with an understanding of your storage needs, the service, and Elasticsearch itself, you can make an educated initial estimate on your hardware needs. This estimate can serve as a useful starting point for the most critical aspect of sizing domains: testing them with representative workloads and monitoring their performance.

**Topics**

## Calculating Storage Requirements

Most Elasticsearch workloads fall into one of two broad categories:

- Long-lived index: You write code that processes data into one or more Elasticsearch indices and then updates those indices periodically as the source data changes. Some common examples are website, document, and ecommerce search.
- Rolling indices: Data continuously flows into a set of temporary indices, with an indexing period and retention window, such as a set of daily indices that is retained for two weeks. Some common examples are log analytics, time-series processing, and clickstream analytics.

For long-lived index workloads, you can examine the source data on disk and easily determine how much storage space it consumes. If the data comes from multiple sources, just add those sources together.

For rolling indices, you can multiply the amount of data generated during a representative time period by the retention period. For example, if you generate 200 MiB of log data per hour, that's 4.7 GiB per day, which is 66 GiB of data at any given time if you have a two-week retention period.

The size of your source data, however, is just one aspect of your storage requirements. You also have to consider the following:

1. Number of replicas: Each replica is a full copy of an index and needs the same amount of disk space. By default, each Elasticsearch index has one replica. We recommend at least one to prevent data loss. Replicas also improve search performance, so you might want more if you have a read-heavy workload.
2. Elasticsearch indexing overhead: The on-disk size of an index varies, but is often 10% larger than the source data. After indexing your data, you can use the `_cat/indices?v` API and `pri.store.size` value to calculate the exact overhead. `_cat/allocation?v` also provides a useful summary.
3. Operating system reserved space: By default, Linux reserves 5% of the file system for the `root` user for critical processes, system recovery, and to safeguard against disk fragmentation problems.
4. Amazon ES overhead: Amazon ES reserves 20% of the storage space of each instance (up to 20 GiB) for segment merges, logs, and other internal operations.

   Because of this 20 GiB maximum, the total amount of reserved space can vary dramatically depending on the number of instances in your domain. For example, a domain might have three `m4.xlarge.elasticsearch` instances, each with 500 GiB of storage space, for a total of 1.46 TiB. In this case, the total reserved space is only 60 GiB. Another domain might have 10 `m3.medium.elasticsearch` instances, each with 100 GiB of storage space, for a total of 0.98 TiB. Here, the total reserved space is 200 GiB, even though the first domain is 50% larger.

   In the following formula, we apply a "worst-case" estimate for overhead that includes additional free space to help minimize the impact of node failures and Availability Zone outages.

In summary, if you have 66 GiB of data at any given time and want one replica, your *minimum* storage requirement is closer to 66 * 2 * 1.1 / 0.95 / 0.8 = 191 GiB. You can generalize this calculation as follows:

**Source Data * (1 + Number of Replicas) * (1 + Indexing Overhead) / (1 - Linux Reserved Space) / (1 - Amazon ES Overhead) = Minimum Storage Requirement**

Or you can use this simplified version:

**Source Data * (1 + Number of Replicas) * 1.45 = Minimum Storage Requirement**

Insufficient storage space is one of the most common causes of cluster instability, so you should cross-check the numbers when you choose instance types, instance counts, and storage volumes (p. 174).

Other storage considerations exist:

- If your minimum storage requirement exceeds 1 PB, see the section called “Petabyte Scale” (p. 175).
- If you have rolling indices and want to use a hot-warm architecture, see the section called “UltraWarm” (p. 160).

## Choosing the Number of Shards

After you understand your storage requirements, you can investigate your indexing strategy. Each Elasticsearch index is split into some number of shards. Because you can't easily change the number of primary shards for an existing index, you should decide about shard count *before* indexing your first document.

The overarching goal of choosing a number of shards is to distribute an index evenly across all data nodes in the cluster. However, these shards shouldn't be too large or too numerous. A good rule of thumb is to try to keep shard size between 10–50 GiB. Large shards can make it difficult for Elasticsearch to recover from failure, but because each shard uses some amount of CPU and memory, having too many small shards can cause performance issues and out of memory errors. In other words, shards should be small enough that the underlying Amazon ES instance can handle them, but not so small that they place needless strain on the hardware.

For example, suppose you have 66 GiB of data. You don't expect that number to increase over time, and you want to keep your shards around 30 GiB each. Your number of shards therefore should be approximately 66 * 1.1 / 30 = 3. You can generalize this calculation as follows:

**(Source Data + Room to Grow) * (1 + Indexing Overhead) / Desired Shard Size = Approximate Number of Primary Shards**

This equation helps compensate for growth over time. If you expect those same 67 GiB of data to quadruple over the next year, the approximate number of shards is (66 + 198) * 1.1 / 30 = 10. Remember, though, you don't have those extra 198 GiB of data *yet*. Check to make sure that this preparation for the future doesn't create unnecessarily tiny shards that consume huge amounts of CPU and memory in the present. In this case, 66 * 1.1 / 10 shards = 7.26 GiB per shard, which will consume extra resources and is below the recommended size range. You might consider the more middle-of-the-road approach of six shards, which leaves you with 12 GiB shards today and 48 GiB shards in the future. Then again, you might prefer to start with three shards and reindex your data when the shards exceed 50 GiB.

A far less common issue involves limiting the number of shards per node. If you size your shards appropriately, you typically run out of disk space long before encountering this limit. For example, an `m5.large.elasticsearch` instance has a maximum disk size of 512 GiB. If you stay below 80% disk usage and size your shards at 20 GiB, it can accommodate approximately 20 shards. Elasticsearch 7.*x* and later have a limit of *1,000* shards per node, adjustable using the `cluster.max_shards_per_node` setting.

Sizing shards appropriately almost always keeps you below this limit, but you can also consider the number of shards for each GiB of Java heap. On a given node, have no more than 20 shards per GiB of

Java heap. For example, an `m5.large.elasticsearch` instance has a 4 GiB heap, so each node should have no more than 80 shards. At that shard count, each shard is roughly 5 GiB in size, which is well below our recommendation.

## Choosing Instance Types and Testing

After you calculate your storage requirements and choose the number of shards that you need, you can start to make hardware decisions. Hardware requirements vary dramatically by workload, but we can still offer some basic recommendations.

In general, the storage limits (p. 197) for each instance type map to the amount of CPU and memory that you might need for light workloads. For example, an `m4.large.elasticsearch` instance has a maximum EBS volume size of 512 GiB, 2 vCPU cores, and 8 GiB of memory. If your cluster has many shards, performs taxing aggregations, updates documents frequently, or processes a large number of queries, those resources might be insufficient for your needs. If you believe your cluster falls into one of these categories, try starting with a configuration closer to 2 vCPU cores and 8 GiB of memory for every 100 GiB of your storage requirement.

**Tip**
For a summary of the hardware resources that are allocated to each instance type, see Amazon Elasticsearch Service Pricing.

Still, even those resources might be insufficient. Some Elasticsearch users report that they need many times those resources to fulfill their requirements. Finding the right hardware for your workload means making an educated initial estimate, testing with representative workloads, adjusting, and testing again:

1. To start, we recommend a minimum of three nodes to avoid potential Elasticsearch issues, such as split brain. If you have three dedicated master nodes (p. 176), we still recommend a minimum of two data nodes for replication.
2. If you have a 184 GiB storage requirement and the recommended minimum number of three nodes, use the equation 184 / 3 = 61 GiB to find the amount of storage that each node needs. In this example, you might select three `m5.large.elasticsearch` instances, each using a 90 GiB EBS storage volume so that you have a safety net and some room for growth over time. This configuration provides 6 vCPU cores and 24 GiB of memory, so it's suited to lighter workloads.

   For a more substantial example, consider a 14 TiB (14,336 GiB) storage requirement and a heavy workload. In this case, you might choose to begin testing with 2 * 144 = 288 vCPU cores and 8 * 144 = 1152 GiB of memory. These numbers work out to approximately 18 `i3.4xlarge.elasticsearch` instances. If you don't need the fast, local storage, you could also test 18 `r5.4xlarge.elasticsearch` instances, each using a 1 TiB EBS storage volume.

   If your cluster includes hundreds of terabytes of data, see the section called “Petabyte Scale” (p. 175).
3. After configuring the cluster, you can add your indices (p. 130) using the number of shards you calculated earlier, perform some representative client testing using a realistic dataset, and monitor CloudWatch metrics (p. 28) to see how the cluster handles the workload.
4. If performance satisfies your needs, tests succeed, and CloudWatch metrics are normal, the cluster is ready to use. Remember to set CloudWatch alarms (p. 178) to detect unhealthy resource usage.

   If performance isn't acceptable, tests fail, or `CPUUtilization` or `JVMMemoryPressure` are high, you might need to choose a different instance type (or add instances) and continue testing. As you add instances, Elasticsearch automatically rebalances the distribution of shards throughout the cluster.

   Because it is easier to measure the excess capacity in an overpowered cluster than the deficit in an underpowered one, we recommend starting with a larger cluster than you think you need. Next, test and scale down to an efficient cluster that has the extra resources to ensure stable operations during periods of increased activity.

Production clusters or clusters with complex states benefit from dedicated master nodes (p. 176), which improve performance and cluster reliability.

## Petabyte Scale for Amazon Elasticsearch Service

Amazon Elasticsearch Service domains offer attached storage of up to 3 PB. You can configure a domain with 200 `i3.16xlarge.elasticsearch` instance types, each with 15 TB of storage. Because of the sheer difference in scale, recommendations for domains of this size differ from our general recommendations (p. 171). This section discusses considerations for creating domains, costs, storage, and shard size.

While this section frequently references the `i3.16xlarge.elasticsearch` instance types, you can use several other instance types to reach 1 PB of total domain storage.

**Creating domains**

Domains of this size exceed the default limit of 40 instances per domain. To request a service limit increase of up to 200 instances per domain, open a case at the AWS Support Center.

**Pricing**

Before creating a domain of this size, check the Amazon Elasticsearch Service Pricing page to ensure that the associated costs match your expectations. Examine the section called “UltraWarm” (p. 160) to see if a hot-warm architecture fits your use case.

**Storage**

The `i3` instance types are designed to provide fast, local non-volatile memory express (NVMe) storage. Because this local storage tends to offer considerable performance benefits when compared to Amazon Elastic Block Store, EBS volumes are not an option when you select these instance types in Amazon ES. If you prefer EBS storage, use another instance type, such as `r5.12xlarge.elasticsearch`.

**Shard size and count**

A common Elasticsearch guideline is not to exceed 50 GB per shard. Given the number of shards necessary to accommodate large domains and the resources available to `i3.16xlarge.elasticsearch` instances, we recommend a shard size of 100 GB.

For example, if you have 450 TB of source data and want one replica, your *minimum* storage requirement is closer to 450 TB * 2 * 1.1 / 0.95 = 1.04 PB. For an explanation of this calculation, see the section called “Calculating Storage Requirements” (p. 172). Although 1.04 PB / 15 TB = 70 instances, you might select 80 or more `i3.16xlarge.elasticsearch` instances to give yourself a storage safety net and account for some variance in the amount of data over time. Each instance adds another 20 GiB to your minimum storage requirement, but for disks of this size, those 20 GiB are almost negligible.

Controlling the number of shards is tricky. Elasticsearch users often rotate indices on a daily basis and retain data for a week or two. In this situation, you might find it useful to distinguish between "active" and "inactive" shards. Active shards are, well, actively being written to or read from. Inactive shards might service the occasional read request, but are largely idle. In general, you should keep the number of active shards below a few thousand. As the number of active shards approaches 10,000, considerable performance and stability risks emerge.

To calculate the number of primary shards, use this formula: 450,000 GB * 1.1 / 100 GB per shard = 4,950 shards. Doubling that number to account for replicas is 9,900 shards, which represents a major concern if all shards are active. But if you rotate indices and only $1/7^{th}$ or $1/14^{th}$ of the shards are active on any given day (1,414 or 707 shards, respectively), the cluster might work well. As always, the most important step of sizing and configuring your domain is to perform representative client testing using a realistic dataset.

# Dedicated Master Nodes

Amazon Elasticsearch Service uses *dedicated master nodes* to increase cluster stability. A dedicated master node performs cluster management tasks, but does not hold data or respond to data upload requests. This offloading of cluster management tasks increases the stability of your domain.

We recommend that you allocate **three** dedicated master nodes for each production Amazon ES domain:

1. One dedicated master node means that you have no backup in the event of a failure.
2. Two dedicated master nodes means that your cluster does not have the necessary quorum of nodes to elect a new master node in the event of a failure.

   A quorum is the number of dedicated master nodes / 2 + 1 (rounded down to the nearest whole number), which Amazon ES sets to `discovery.zen.minimum_master_nodes` when you create your domain.

   In this case, 2 / 2 + 1 = 2. Because one dedicated master node has failed and only one backup exists, the cluster doesn't have a quorum and can't elect a new master.
3. Three dedicated master nodes, the recommended number, provides two backup nodes in the event of a master node failure and the necessary quorum (2) to elect a new master.
4. Four dedicated master nodes are no better than three and can cause issues if you use multiple Availability Zones (p. 17) in a Region that has only two zones.
   - If one master node fails, you have the quorum (3) to elect a new master. If two nodes fail, you lose that quorum, just as you do with three dedicated master nodes.
   - If each Availability Zone has two dedicated master nodes and the zones are unable to communicate with each other, neither zone has the quorum to elect a new master.
5. Having five dedicated master nodes works as well as three and allows you to lose two nodes while maintaining a quorum. But because only one dedicated master node is active at any given time, this configuration means paying for four idle nodes. Many users find this level of failover protection excessive.

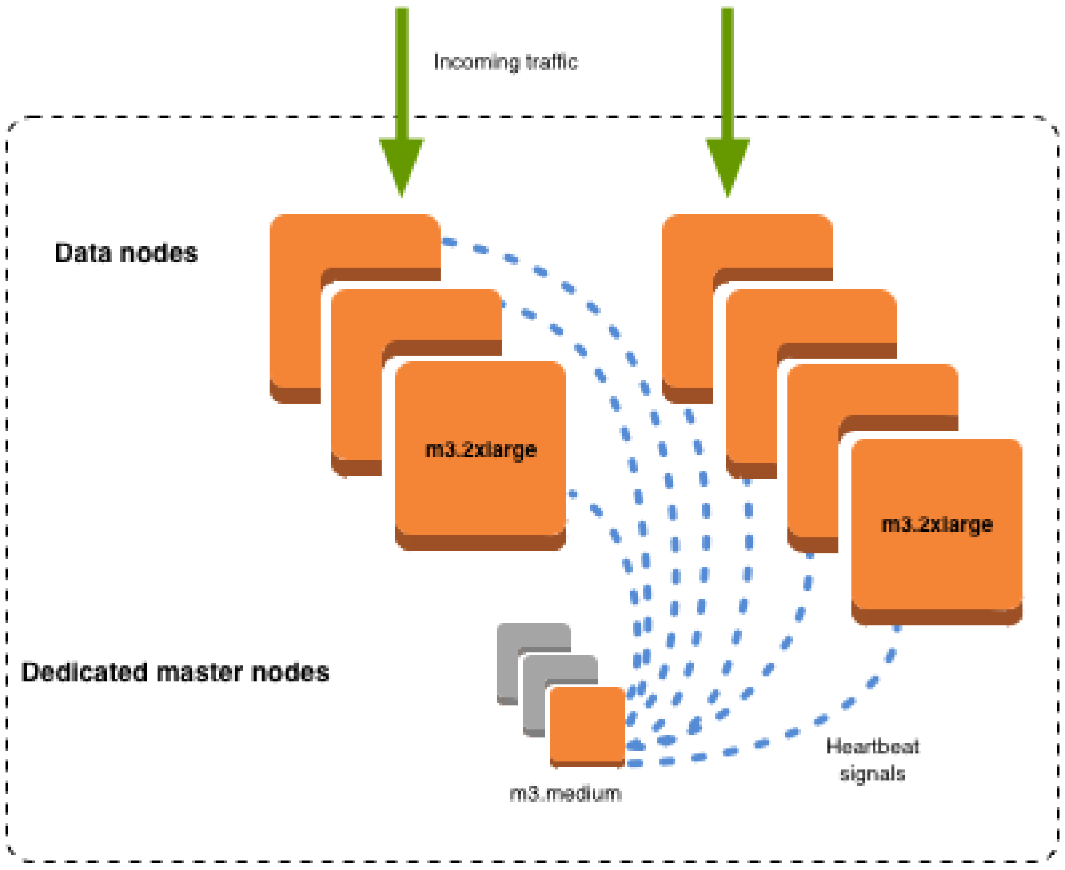
If a cluster has an even number of master-eligible nodes, Elasticsearch versions 7.*x* and later ignore one node so that the voting configuration is always an odd number. In this case, four dedicated master nodes are essentially equivalent to three (and two to one).

**Note**
If your cluster doesn't have the necessary quorum to elect a new master node, write *and* read requests to the cluster both fail. This behavior differs from the Elasticsearch default.

Dedicated master nodes perform the following cluster management tasks:

- Track all nodes in the cluster
- Track the number of indices in the cluster
- Track the number of shards belonging to each index
- Maintain routing information for nodes in the cluster
- Update the cluster state after state changes, such as creating an index and adding or removing nodes in the cluster
- Replicate changes to the cluster state across all nodes in the cluster
- Monitor the health of all cluster nodes by sending *heartbeat signals*, periodic signals that monitor the availability of the data nodes in the cluster

The following illustration shows an Amazon ES domain with ten instances. Seven of the instances are data nodes and three are dedicated master nodes. Only one of the dedicated master nodes is active; the two gray dedicated master nodes wait as backup in case the active dedicated master node fails. All data upload requests are served by the seven data nodes, and all cluster management tasks are offloaded to the active dedicated master node.



Although dedicated master nodes don't process search and query requests, their size is highly correlated with the number of instances, indices, and shards that they can manage. For production clusters, we recommend the following instance types for dedicated master nodes. These recommendations are based on typical workloads and can vary based on your needs. Clusters with many shards or field mappings can benefit from larger instance types. Monitor the dedicated master node metrics (p. 178) to see if you need to use a larger instance type.

| Instance Count | Recommended Minimum Dedicated Master Instance Type |
|---|---|
| 1–10 | `c5.large.elasticsearch` |
| 10–30 | `c5.xlarge.elasticsearch` |
| 30–75 | `c5.2xlarge.elasticsearch` |
| 75–200 | `r5.4xlarge.elasticsearch` |

- For information about how certain configuration changes can affect dedicated master nodes, see the section called "Configuration Changes" (p. 14).

- For clarification on instance count limits, see the section called "Cluster and Instance Limits" (p. 197).
- For more information about specific instance types, including vCPU, memory, and pricing, see Amazon Elasticsearch Instance Prices.

## Recommended CloudWatch Alarms

CloudWatch alarms perform an action when a CloudWatch metric exceeds a specified value for some amount of time. For example, you might want AWS to email you if your cluster health status is `red` for longer than one minute. This section includes some recommended alarms and how to respond to them.

For more information about setting alarms, see Creating Amazon CloudWatch Alarms in the *Amazon CloudWatch User Guide*.

| Alarm | Issue |
|---|---|
| `ClusterStatus.red` maximum is >= 1 for 1 minute, 1 consecutive time | At least one primary shard and its replicas are not allocated to a node. See the section called "Red Cluster Status" (p. 228). |
| `ClusterStatus.yellow` maximum is >= 1 for 1 minute, 1 consecutive time | At least one replica shard is not allocated to a node. See the section called "Yellow Cluster Status" (p. 230). |
| `FreeStorageSpace` minimum is <= 20480 for 1 minute, 1 consecutive time | A node in your cluster is down to 20 GiB of free storage space. See the section called "Lack of Available Storage Space" (p. 230). This value is in MiB, so rather than 20480, we recommend setting it to 25% of the storage space for each node. |
| `ClusterIndexWritesBlocked` is >= 1 for 5 minutes, 1 consecutive time | Your cluster is blocking write requests. See the section called "ClusterBlockException" (p. 230). |
| `Nodes` minimum is < *x* for 1 day, 1 consecutive time | *x* is the number of nodes in your cluster. This alarm indicates that at least one node in your cluster has been unreachable for one day. See the section called "Failed Cluster Nodes" (p. 231). |
| `AutomatedSnapshotFailure` maximum is >= 1 for 1 minute, 1 consecutive time | An automated snapshot failed. This failure is often the result of a red cluster health status. See the section called "Red Cluster Status" (p. 228).<br><br>For a summary of all automated snapshots and some information about failures, you can also try the following:<br><br>`GET domain_endpoint/_snapshot/cs-automated/_all` |
| `CPUUtilization` maximum is >= 80% for 15 minutes, 3 consecutive times | 100% CPU utilization isn't uncommon, but *sustained* high usage is problematic. Consider using larger instance types or adding instances. |
| `JVMMemoryPressure` maximum is >= 80% for 5 minutes, 3 consecutive times | The cluster could encounter out of memory errors if usage increases. Consider scaling vertically. Amazon ES uses half of an instance's RAM for the Java heap, up to a heap size of 32 GiB. You can scale instances vertically up to 64 GiB of RAM, at which point you can scale horizontally by adding instances. |

| Alarm | Issue |
|---|---|
| `MasterCPUUtilization` maximum is >= 50% for 15 minutes, 3 consecutive times | Consider using larger instance types for your dedicated master nodes (p. 176). Because of their role in cluster stability and blue/green deployments (p. 14), dedicated master nodes should have lower CPU usage than data nodes. |
| `MasterJVMMemoryPressure` maximum is >= 80% for 15 minutes, 1 consecutive time | |
| `KMSKeyError` is >= 1 for 1 minute, 1 consecutive time | The KMS encryption key that is used to encrypt data at rest in your domain is disabled. Re-enable it to restore normal operations. For more information, see the section called “Encryption at Rest” (p. 61). |
| `KMSKeyInaccessible` is >= 1 for 1 minute, 1 consecutive time | The KMS encryption key that is used to encrypt data at rest in your domain has been deleted or has revoked its grants to Amazon ES. You can't recover domains that are in this state, but if you have a manual snapshot, you can use it to migrate to a new domain. To learn more, see the section called “Encryption at Rest” (p. 61). |

**Note**
If you just want to *view* metrics, see Monitoring CloudWatch Metrics (p. 28).

# Amazon Elasticsearch Service General Reference

Amazon Elasticsearch Service (Amazon ES) supports a variety of instances, operations, plugins, and other resources.

**Topics**

## Supported Instance Types

Amazon ES supports the following instance types. Not all Regions support all instance types. For availability details, see Amazon Elasticsearch Service Pricing.

For information about which instance type is appropriate for your use case, see the section called "Sizing Amazon ES Domains" (p. 171), the section called "EBS Volume Size Limits" (p. 198), and the section called "Network Limits" (p. 200).

| Instance Type | Restrictions |
|---|---|
| C4 | |
| C5 | The C5 instance types require Elasticsearch version 5.1 or later. |
| I2 | |
| I3 | The I3 instance types require Elasticsearch version 5.1 or later and do not support EBS storage volumes. |
| M3 | The M3 instance types do not support encryption of data at rest or fine-grained access control. |
| M4 | |
| M5 | The M5 instance types require Elasticsearch version 5.1 or later. |
| R3 | The R3 instance types do not support encryption of data at rest or fine-grained access control. |
| R4 | |

| Instance Type | Restrictions |
|---|---|
| R5 | The R5 instance types require Elasticsearch version 5.1 or later. |
| T2 | • You can use the T2 instance types only if the instance count for your domain is 10 or fewer.<br>• The `t2.micro.elasticsearch` instance type supports only Elasticsearch 1.5 and 2.3.<br>• The T2 instance types do not support encryption of data at rest, fine-grained access control, or UltraWarm storage. |

**Tip**
You can use different instance types for dedicated master nodes (p. 176) and data nodes.

## Features by Elasticsearch Version

Many Amazon ES features have a minimum Elasticsearch version requirement.

| Feature | Minimum Elasticsearch Version |
|---|---|
| VPC support | Included on all domains |
| Require HTTPS for all traffic to the domain | |
| Multi-AZ support | |
| Dedicated master nodes | |
| Custom packages | |
| Error and slow logs publishing | 5.1 |
| Curator CLI support | |
| Encryption of data at rest | |
| Cognito authentication for Kibana | |
| In-place Elasticsearch upgrades | |
| Hourly automated snapshots | 5.3 |

| Feature | Minimum Elasticsearch Version |
|---|---|
| Node-to-node encryption | 6.0 |
| Java high-level REST client support | |
| Alerting | 6.2 |
| SQL | 6.5 |
| Fine-grained access control | 6.7 |
| UltraWarm | 6.8 |
| Index State Management | 6.8 |
| KNN | 7.1 |

For information about plugins, which enable some of these features and additional functionality, see the section called "Plugins by Elasticsearch Version" (p. 182). For information about the Elasticsearch API for each version, see the section called "Supported Elasticsearch Operations" (p. 183).

## Plugins by Elasticsearch Version

Amazon Elasticsearch Service domains come prepackaged with plugins from the Elasticsearch community. The service automatically deploys and manages plugins for you, but deploys different plugins depending on the version of Elasticsearch you choose for your domain.

| Plugins | Minimum Elasticsearch Version |
|---|---|
| ICU Analysis | Included on all domains |
| Japanese (kuromoji) Analysis | |
| Phonetic Analysis | 2.3 |
| Seunjeon Korean Analysis | 5.1 |
| Smart Chinese Analysis | |
| Stempel Polish Analysis | |
| Ingest Attachment Processor | |

| Plugins | Minimum Elasticsearch Version |
|---|---|
| Ingest User Agent Processor | |
| Mapper Murmur3 | |
| Mapper Size | 5.3 |
| Ukrainian Analysis | |
| Open Distro for Elasticsearch Alerting | 6.2 |
| Open Distro for Elasticsearch SQL | 6.5 |
| Open Distro for Elasticsearch Security | 6.7 |
| Open Distro for Elasticsearch Index State Management | 6.8 |
| Open Distro for Elasticsearch KNN | 7.1 |

**Note**
This table is not comprehensive. Amazon ES uses additional plugins to enable core service functionality, such as the S3 Repository plugin for snapshots and the Open Distro for Elasticsearch Performance Analyzer plugin for optimization and monitoring.

## Supported Elasticsearch Operations

Amazon ES supports many versions of Elasticsearch. The following topics show the operations that Amazon ES supports for each version.

**Topics**

# Notable API Differences

## Settings and Statistics

Amazon ES only accepts PUT requests to the `_cluster/settings` API that use the "flat" settings form. It rejects requests that use the expanded settings form.

```
// Accepted
PUT _cluster/settings
{
  "persistent" : {
    "action.auto_create_index" : false
  }
}

// Rejected
PUT _cluster/settings
{
  "persistent": {
    "action": {
      "auto_create_index": false
    }
  }
}
```

The high-level Java REST client uses the expanded form, so if you need to send settings requests, use the low-level client.

Prior to Elasticsearch 5.3, the `_cluster/settings` API on Amazon ES domains supported only the HTTP `PUT` method, not the `GET` method. Later versions support the `GET` method, as shown in the following example:

```
GET https://domain.region.es.amazonaws.com/_cluster/settings?pretty
```

Here is a return example:

```
{
  "persistent": {
    "cluster": {
      "routing": {
        "allocation": {
          "cluster_concurrent_rebalance": "2",
          "node_concurrent_recoveries": "2",
          "disk": {
            "watermark": {
              "low": "1.35gb",
              "flood_stage": "0.45gb",
              "high": "0.9gb"
            }
          },
          "node_initial_primaries_recoveries": "4"
        }
      }
    },
    "indices": {
      "recovery": {
        "max_bytes_per_sec": "40mb"
      }
    }
  }
}
```

If you compare responses from an open source Elasticsearch cluster and Amazon ES for certain settings and statistics APIs, you might notice missing fields. Amazon ES redacts certain information that exposes service internals, such as the file system data path from `_nodes/stats` or the operating system name and version from `_nodes`.

## Shrink

The `_shrink` API can cause upgrades, configuration changes, and domain deletions to fail. We don't recommend using it on domains that run Elasticsearch versions 5.3 or 5.1. These versions have a bug that can cause snapshot restoration of shrunken indices to fail.

If you use the `_shrink` API on other Elasticsearch versions, make the following request before starting the shrink operation:

```
PUT https://domain.region.es.amazonaws.com/source-index/_settings
{
  "settings": {
    "index.routing.allocation.require._name": "name-of-the-node-to-shrink-to",
    "index.blocks.read_only": true
  }
}
```

Then make the following requests after completing the shrink operation:

```
PUT https://domain.region.es.amazonaws.com/source-index/_settings
{
  "settings": {
    "index.routing.allocation.require._name": null,
    "index.blocks.read_only": false
  }
}

PUT https://domain.region.es.amazonaws.com/shrunken-index/_settings
{
  "settings": {
    "index.routing.allocation.require._name": null,
    "index.blocks.read_only": false
  }
}
```

## Version 7.4

For Elasticsearch 7.4, Amazon ES supports the following operations.

| | | |
|---|---|---|
| • All operations in the index path (such as `/index-name/_forcemerge` and `/index-name/update/id`) **except** `/index-name/_close`<br>• `/_alias`<br>• `/_aliases`<br>• `/_all`<br>• `/_analyze`<br>• `/_bulk`<br>• `/_cat` (except `/_cat/nodeattrs`)<br>• `/_cluster/allocation/explain`<br>• `/_cluster/health`<br>• `/_cluster/pending_tasks`<br>• `/_cluster/settings` for several properties[4]:<br>  • `action.auto_create_index`<br>  • `action.search.shard_count.limit`<br>  • `indices.breaker.fielddata.limit`<br>  • `indices.breaker.request.limit`<br>  • `indices.breaker.total.limit`<br>  • `cluster.max_shards_per_node` | • `/_cluster/state`<br>• `/_cluster/stats`<br>• `/_count`<br>• `/_delete_by_query`[1]<br>• `/_explain`<br>• `/_field_caps`<br>• `/_field_stats`<br>• `/_flush`<br>• `/_ingest/pipeline`<br>• `/_mapping`<br>• `/_mget`<br>• `/_msearch`<br>• `/_mtermvectors`<br>• `/_nodes`<br>• `/_opendistro/_alerting`<br>• `/_opendistro/_ism`<br>• `/_opendistro/_security`<br>• `/_opendistro/_sql`<br>• `/_percolate`<br>• `/_plugin/kibana`<br>• `/_rank_eval` | • `/_refresh`<br>• `/_reindex`[1]<br>• `/_render`<br>• `/_rollover`<br>• `/_scripts`[3]<br>• `/_search`[2]<br>• `/_search profile`<br>• `/_shard_stores`<br>• `/_shrink`[5]<br>• `/_snapshot`<br>• `/_split`<br>• `/_stats`<br>• `/_status`<br>• `/_tasks`<br>• `/_template`<br>• `/_update_by_query`[1]<br>• `/_validate` |

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the `/_tasks` operation along with these operations to verify that the requests completed successfully.
2. DELETE requests to `/_search/scroll` with a message body must specify `"Content-Length"` in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in `scroll_id` values, use the request body, not the query string, to pass `scroll_id` values to Amazon ES.
3. For considerations about using scripts, see the section called "Other Supported Resources" (p. 205).
4. Refers to the `PUT` method. For information about the `GET` method, see the section called "Notable API Differences" (p. 184).
5. See the section called "Shrink" (p. 185).

## Version 7.1

For Elasticsearch 7.1, Amazon ES supports the following operations.

| | | |
|---|---|---|
| • All operations in the index path (such as `/index-name/_forcemerge` and `/index-name/update/id`) **except** `/index-name/_close`<br>• `/_alias` | • `/_cluster/state`<br>• `/_cluster/stats`<br>• `/_count`<br>• `/_delete_by_query`[1]<br>• `/_explain`<br>• `/_field_caps` | • `/_refresh`<br>• `/_reindex`[1]<br>• `/_render`<br>• `/_rollover`<br>• `/_scripts`[3]<br>• `/_search`[2] |

- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (except /_cat/nodeattrs)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings for several properties[4]:
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
  - cluster.max_shards_per_node

- /_field_stats
- /_flush
- /_ingest/pipeline
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_opendistro/_alerting
- /_opendistro/_ism
- /_opendistro/_security
- /_opendistro/_sql
- /_percolate
- /_plugin/kibana
- /_rank_eval

- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query[1]
- /_validate

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the /_tasks operation along with these operations to verify that the requests completed successfully.
2. DELETE requests to /_search/scroll with a message body must specify "Content-Length" in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in scroll_id values, use the request body, not the query string, to pass scroll_id values to Amazon ES.
3. For considerations about using scripts, see the section called "Other Supported Resources" (p. 205).
4. Refers to the PUT method. For information about the GET method, see the section called "Notable API Differences" (p. 184).
5. See the section called "Shrink" (p. 185).

## Version 6.8

For Elasticsearch 6.8, Amazon ES supports the following operations.

- All operations in the index path (such as /*index-name*/_forcemerge and /*index-name*/update/*id*) **except** /*index-name*/_close
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (except /_cat/nodeattrs)
- /_cluster/allocation/explain

- /_cluster/state
- /_cluster/stats
- /_count
- /_delete_by_query[1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_mapping
- /_mget
- /_msearch

- /_refresh
- /_reindex[1]
- /_render
- /_rollover
- /_scripts[3]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[5]
- /_snapshot
- /_split
- /_stats

| | | |
|---|---|---|
| • `/_cluster/health`<br>• `/_cluster/pending_tasks`<br>• `/_cluster/settings` for several properties[4]:<br>  • `action.auto_create_index`<br>  • `action.search.shard_count.limit`<br>  • `indices.breaker.fielddata.limit`<br>  • `indices.breaker.request.limit`<br>  • `indices.breaker.total.limit`<br>  • `cluster.max_shards_per_node`<br>  • `cluster.blocks.read_only` | • `/_mtermvectors`<br>• `/_nodes`<br>• `/_opendistro/_alerting`<br>• `/_opendistro/_security`<br>• `/_opendistro/_sql`<br>• `/_percolate`<br>• `/_plugin/kibana`<br>• `/_rank_eval` | • `/_status`<br>• `/_tasks`<br>• `/_template`<br>• `/_update_by_query`[1]<br>• `/_validate` |

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the `/_tasks` operation along with these operations to verify that the requests completed successfully.
2. DELETE requests to `/_search/scroll` with a message body must specify `"Content-Length"` in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in `scroll_id` values, use the request body, not the query string, to pass `scroll_id` values to Amazon ES.
3. For considerations about using scripts, see the section called “Other Supported Resources” (p. 205).
4. Refers to the `PUT` method. For information about the `GET` method, see the section called “Notable API Differences” (p. 184).
5. See the section called “Shrink” (p. 185).

## Version 6.7

For Elasticsearch 6.7, Amazon ES supports the following operations.

| | | |
|---|---|---|
| • All operations in the index path (such as `/`*`index-name`*`/_forcemerge` and `/`*`index-name`*`/update/`*`id`*) **except** `/`*`index-name`*`/_close`<br>• `/_alias`<br>• `/_aliases`<br>• `/_all`<br>• `/_analyze`<br>• `/_bulk`<br>• `/_cat` (except `/_cat/nodeattrs`)<br>• `/_cluster/allocation/explain`<br>• `/_cluster/health`<br>• `/_cluster/pending_tasks`<br>• `/_cluster/settings` for several properties[4]:<br>  • `action.auto_create_index`<br>  • `action.search.shard_count.limit` | • `/_cluster/state`<br>• `/_cluster/stats`<br>• `/_count`<br>• `/_delete_by_query`[1]<br>• `/_explain`<br>• `/_field_caps`<br>• `/_field_stats`<br>• `/_flush`<br>• `/_ingest/pipeline`<br>• `/_mapping`<br>• `/_mget`<br>• `/_msearch`<br>• `/_mtermvectors`<br>• `/_nodes`<br>• `/_opendistro/_alerting`<br>• `/_opendistro/_security`<br>• `/_opendistro/_sql` | • `/_refresh`<br>• `/_reindex`[1]<br>• `/_render`<br>• `/_rollover`<br>• `/_scripts`[3]<br>• `/_search`[2]<br>• `/_search profile`<br>• `/_shard_stores`<br>• `/_shrink`[5]<br>• `/_snapshot`<br>• `/_split`<br>• `/_stats`<br>• `/_status`<br>• `/_tasks`<br>• `/_template`<br>• `/_update_by_query`[1]<br>• `/_validate` |

| | | |
|---|---|---|
| • `indices.breaker.fielddata.limit`<br>• `indices.breaker.request.limit`<br>• `indices.breaker.total.limit`<br>• `cluster.max_shards_per_node` | • `/_percolate`<br>• `/_plugin/kibana`<br>• `/_rank_eval` | |

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the `/_tasks` operation along with these operations to verify that the requests completed successfully.
2. DELETE requests to `/_search/scroll` with a message body must specify `"Content-Length"` in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in `scroll_id` values, use the request body, not the query string, to pass `scroll_id` values to Amazon ES.
3. For considerations about using scripts, see the section called "Other Supported Resources" (p. 205).
4. Refers to the `PUT` method. For information about the `GET` method, see the section called "Notable API Differences" (p. 184).
5. See the section called "Shrink" (p. 185).

## Version 6.5

For Elasticsearch 6.5, Amazon ES supports the following operations.

| | | |
|---|---|---|
| • All operations in the index path (such as `/index-name/_forcemerge` and `/index-name/update/id`) **except** `/index-name/_close`<br>• `/_alias`<br>• `/_aliases`<br>• `/_all`<br>• `/_analyze`<br>• `/_bulk`<br>• `/_cat` (except `/_cat/nodeattrs`)<br>• `/_cluster/allocation/explain`<br>• `/_cluster/health`<br>• `/_cluster/pending_tasks`<br>• `/_cluster/settings` for several properties[4]:<br>  • `action.auto_create_index`<br>  • `action.search.shard_count.limit`<br>  • `indices.breaker.fielddata.limit`<br>  • `indices.breaker.request.limit`<br>  • `indices.breaker.total.limit` | • `/_cluster/state`<br>• `/_cluster/stats`<br>• `/_count`<br>• `/_delete_by_query`[1]<br>• `/_explain`<br>• `/_field_caps`<br>• `/_field_stats`<br>• `/_flush`<br>• `/_ingest/pipeline`<br>• `/_mapping`<br>• `/_mget`<br>• `/_msearch`<br>• `/_mtermvectors`<br>• `/_nodes`<br>• `/_opendistro/_alerting`<br>• `/_opendistro/_sql`<br>• `/_percolate`<br>• `/_plugin/kibana`<br>• `/_rank_eval` | • `/_refresh`<br>• `/_reindex`[1]<br>• `/_render`<br>• `/_rollover`<br>• `/_scripts`[3]<br>• `/_search`[2]<br>• `/_search profile`<br>• `/_shard_stores`<br>• `/_shrink`[5]<br>• `/_snapshot`<br>• `/_split`<br>• `/_stats`<br>• `/_status`<br>• `/_tasks`<br>• `/_template`<br>• `/_update_by_query`[1]<br>• `/_validate` |

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the `/_tasks` operation along with these operations to verify that the requests completed successfully.

2. DELETE requests to `/_search/scroll` with a message body must specify `"Content-Length"` in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in `scroll_id` values, use the request body, not the query string, to pass `scroll_id` values to Amazon ES.
3. For considerations about using scripts, see the section called “Other Supported Resources” (p. 205).
4. Refers to the `PUT` method. For information about the `GET` method, see the section called “Notable API Differences” (p. 184).
5. See the section called “Shrink” (p. 185).

## Version 6.4

For Elasticsearch 6.4, Amazon ES supports the following operations.

| | | |
|---|---|---|
| • All operations in the index path (such as `/index-name/_forcemerge` and `/index-name/update/id`) **except** `/index-name/_close`<br>• `/_alias`<br>• `/_aliases`<br>• `/_all`<br>• `/_analyze`<br>• `/_bulk`<br>• `/_cat` (except `/_cat/nodeattrs`)<br>• `/_cluster/allocation/explain`<br>• `/_cluster/health`<br>• `/_cluster/pending_tasks`<br>• `/_cluster/settings` for several properties[4]:<br>  • `action.auto_create_index`<br>  • `action.search.shard_count.limit`<br>  • `indices.breaker.fielddata.limit`<br>  • `indices.breaker.request.limit`<br>  • `indices.breaker.total.limit` | • `/_cluster/state`<br>• `/_cluster/stats`<br>• `/_count`<br>• `/_delete_by_query`[1]<br>• `/_explain`<br>• `/_field_caps`<br>• `/_field_stats`<br>• `/_flush`<br>• `/_ingest/pipeline`<br>• `/_mapping`<br>• `/_mget`<br>• `/_msearch`<br>• `/_mtermvectors`<br>• `/_nodes`<br>• `/_opendistro/_alerting`<br>• `/_percolate`<br>• `/_plugin/kibana`<br>• `/_rank_eval` | • `/_refresh`<br>• `/_reindex`[1]<br>• `/_render`<br>• `/_rollover`<br>• `/_scripts`[3]<br>• `/_search`[2]<br>• `/_search profile`<br>• `/_shard_stores`<br>• `/_shrink`[5]<br>• `/_snapshot`<br>• `/_split`<br>• `/_stats`<br>• `/_status`<br>• `/_tasks`<br>• `/_template`<br>• `/_update_by_query`[1]<br>• `/_validate` |

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the `/_tasks` operation along with these operations to verify that the requests completed successfully.
2. DELETE requests to `/_search/scroll` with a message body must specify `"Content-Length"` in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in `scroll_id` values, use the request body, not the query string, to pass `scroll_id` values to Amazon ES.
3. For considerations about using scripts, see the section called “Other Supported Resources” (p. 205).
4. Refers to the `PUT` method. For information about the `GET` method, see the section called “Notable API Differences” (p. 184).
5. See the section called “Shrink” (p. 185).

## Version 6.3

For Elasticsearch 6.3, Amazon ES supports the following operations.

| | | |
|---|---|---|
| • All operations in the index path (such as `/`*`index-name`*`/_forcemerge` and `/`*`index-name`*`/update/`*`id`*) **except** `/`*`index-name`*`/_close`<br>• `/_alias`<br>• `/_aliases`<br>• `/_all`<br>• `/_analyze`<br>• `/_bulk`<br>• `/_cat` (except `/_cat/nodeattrs`)<br>• `/_cluster/allocation/explain`<br>• `/_cluster/health`<br>• `/_cluster/pending_tasks`<br>• `/_cluster/settings` for several properties[4]:<br>  • `action.auto_create_index`<br>  • `action.search.shard_count.limit`<br>  • `indices.breaker.fielddata.limit`<br>  • `indices.breaker.request.limit`<br>  • `indices.breaker.total.limit` | • `/_cluster/state`<br>• `/_cluster/stats`<br>• `/_count`<br>• `/_delete_by_query`[1]<br>• `/_explain`<br>• `/_field_caps`<br>• `/_field_stats`<br>• `/_flush`<br>• `/_ingest/pipeline`<br>• `/_mapping`<br>• `/_mget`<br>• `/_msearch`<br>• `/_mtermvectors`<br>• `/_nodes`<br>• `/_opendistro/_alerting`<br>• `/_percolate`<br>• `/_plugin/kibana`<br>• `/_rank_eval` | • `/_refresh`<br>• `/_reindex`[1]<br>• `/_render`<br>• `/_rollover`<br>• `/_scripts`[3]<br>• `/_search`[2]<br>• `/_search profile`<br>• `/_shard_stores`<br>• `/_shrink`[5]<br>• `/_snapshot`<br>• `/_split`<br>• `/_stats`<br>• `/_status`<br>• `/_tasks`<br>• `/_template`<br>• `/_update_by_query`[1]<br>• `/_validate` |

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the `/_tasks` operation along with these operations to verify that the requests completed successfully.
2. DELETE requests to `/_search/scroll` with a message body must specify `"Content-Length"` in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in `scroll_id` values, use the request body, not the query string, to pass `scroll_id` values to Amazon ES.
3. For considerations about using scripts, see the section called "Other Supported Resources" (p. 205).
4. Refers to the `PUT` method. For information about the `GET` method, see the section called "Notable API Differences" (p. 184).
5. See the section called "Shrink" (p. 185).

## Version 6.2

For Elasticsearch 6.2, Amazon ES supports the following operations.

| | | |
|---|---|---|
| • All operations in the index path (such as `/`*`index-name`*`/_forcemerge` and `/`*`index-name`*`/update/`*`id`*) **except** `/`*`index-name`*`/_close` | • `/_cluster/state`<br>• `/_cluster/stats`<br>• `/_count`<br>• `/_delete_by_query`[1] | • `/_refresh`<br>• `/_reindex`[1]<br>• `/_render`<br>• `/_rollover` |

- `/_alias`
- `/_aliases`
- `/_all`
- `/_analyze`
- `/_bulk`
- `/_cat` (except `/_cat/nodeattrs`)
- `/_cluster/allocation/explain`
- `/_cluster/health`
- `/_cluster/pending_tasks`
- `/_cluster/settings` for several properties[4]:
  - `action.auto_create_index`
  - `action.search.shard_count.limit`
  - `indices.breaker.fielddata.limit`
  - `indices.breaker.request.limit`
  - `indices.breaker.total.limit`
- `/_explain`
- `/_field_caps`
- `/_field_stats`
- `/_flush`
- `/_ingest/pipeline`
- `/_mapping`
- `/_mget`
- `/_msearch`
- `/_mtermvectors`
- `/_nodes`
- `/_opendistro/_alerting`
- `/_percolate`
- `/_plugin/kibana`
- `/_rank_eval`
- `/_scripts`[3]
- `/_search`[2]
- `/_search profile`
- `/_shard_stores`
- `/_shrink`[5]
- `/_snapshot`
- `/_split`
- `/_stats`
- `/_status`
- `/_tasks`
- `/_template`
- `/_update_by_query`[1]
- `/_validate`

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the `/_tasks` operation along with these operations to verify that the requests completed successfully.
2. DELETE requests to `/_search/scroll` with a message body must specify `"Content-Length"` in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in `scroll_id` values, use the request body, not the query string, to pass `scroll_id` values to Amazon ES.
3. For considerations about using scripts, see the section called "Other Supported Resources" (p. 205).
4. Refers to the `PUT` method. For information about the `GET` method, see the section called "Notable API Differences" (p. 184).
5. See the section called "Shrink" (p. 185).

## Version 6.0

For Elasticsearch 6.0, Amazon ES supports the following operations.

- All operations in the index path (such as `/index-name/_forcemerge` and `/index-name/update/id`) **except** `/index-name/_close`
- `/_alias`
- `/_aliases`
- `/_all`
- `/_analyze`
- `/_bulk`
- `/_cat` (except `/_cat/nodeattrs`)
- `/_cluster/allocation/explain`
- `/_cluster/health`
- `/_cluster/state`
- `/_cluster/stats`
- `/_count`
- `/_delete_by_query`[1]
- `/_explain`
- `/_field_caps`
- `/_field_stats`
- `/_flush`
- `/_ingest/pipeline`
- `/_mapping`
- `/_mget`
- `/_msearch`
- `/_mtermvectors`
- `/_render`
- `/_rollover`
- `/_scripts`[3]
- `/_search`[2]
- `/_search profile`
- `/_shard_stores`
- `/_shrink`[5]
- `/_snapshot`
- `/_stats`
- `/_status`
- `/_tasks`
- `/_template`
- `/_update_by_query`[1]

| | | |
|---|---|---|
| • /_cluster/pending_tasks<br>• /_cluster/settings for several properties[4]:<br>  • action.auto_create_index<br>  • action.search.shard_count.limit<br>  • indices.breaker.fielddata.limit<br>  • indices.breaker.request.limit<br>  • indices.breaker.total.limit | • /_nodes<br>• /_percolate<br>• /_plugin/kibana<br>• /_refresh<br>• /_reindex[1] | • /_validate |

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the `/_tasks` operation along with these operations to verify that the requests completed successfully.
2. DELETE requests to `/_search/scroll` with a message body must specify `"Content-Length"` in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in `scroll_id` values, use the request body, not the query string, to pass `scroll_id` values to Amazon ES.
3. For considerations about using scripts, see the section called “Other Supported Resources” (p. 205).
4. Refers to the `PUT` method. For information about the `GET` method, see the section called “Notable API Differences” (p. 184).
5. See the section called “Shrink” (p. 185).

## Version 5.6

For Elasticsearch 5.6, Amazon ES supports the following operations.

| | | |
|---|---|---|
| • All operations in the index path (such as /*index-name*/_forcemerge and /*index-name*/update/*id*) **except** /*index-name*/_close<br>• /_alias<br>• /_aliases<br>• /_all<br>• /_analyze<br>• /_bulk<br>• /_cat (except /_cat/nodeattrs)<br>• /_cluster/allocation/explain<br>• /_cluster/health<br>• /_cluster/pending_tasks<br>• /_cluster/settings for several properties[4]:<br>  • action.auto_create_index<br>  • action.search.shard_count.limit<br>  • indices.breaker.fielddata.limit<br>  • indices.breaker.request.limit<br>  • indices.breaker.total.limit | • /_cluster/state<br>• /_cluster/stats<br>• /_count<br>• /_delete_by_query[1]<br>• /_explain<br>• /_field_caps<br>• /_field_stats<br>• /_flush<br>• /_ingest/pipeline<br>• /_mapping<br>• /_mget<br>• /_msearch<br>• /_mtermvectors<br>• /_nodes<br>• /_percolate<br>• /_plugin/kibana<br>• /_refresh<br>• /_reindex[1] | • /_render<br>• /_rollover<br>• /_scripts[3]<br>• /_search[2]<br>• /_search profile<br>• /_shard_stores<br>• /_shrink[5]<br>• /_snapshot<br>• /_stats<br>• /_status<br>• /_tasks<br>• /_template<br>• /_update_by_query[1]<br>• /_validate |

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the `/_tasks` operation along with these operations to verify that the requests completed successfully.
2. DELETE requests to `/_search/scroll` with a message body must specify `"Content-Length"` in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in `scroll_id` values, use the request body, not the query string, to pass `scroll_id` values to Amazon ES.
3. For considerations about using scripts, see the section called "Other Supported Resources" (p. 205).
4. Refers to the `PUT` method. For information about the `GET` method, see the section called "Notable API Differences" (p. 184).
5. See the section called "Shrink" (p. 185).

## Version 5.5

For Elasticsearch 5.5, Amazon ES supports the following operations.

- All operations in the index path (such as /*index-name*/_forcemerge and /*index-name*/update/*id*) **except** /*index-name*/_close
- `/_alias`
- `/_aliases`
- `/_all`
- `/_analyze`
- `/_bulk`
- `/_cat` (except `/_cat/nodeattrs`)
- `/_cluster/allocation/explain`
- `/_cluster/health`
- `/_cluster/pending_tasks`
- `/_cluster/settings` for several properties[4]:
  - `action.auto_create_index`
  - `action.search.shard_count.limit`
  - `indices.breaker.fielddata.limit`
  - `indices.breaker.request.limit`
  - `indices.breaker.total.limit`
- `/_cluster/state`
- `/_cluster/stats`
- `/_count`
- `/_delete_by_query`[1]
- `/_explain`
- `/_field_caps`
- `/_field_stats`
- `/_flush`
- `/_ingest/pipeline`
- `/_mapping`
- `/_mget`
- `/_msearch`
- `/_mtermvectors`
- `/_nodes`
- `/_percolate`
- `/_plugin/kibana`
- `/_refresh`
- `/_reindex`[1]
- `/_render`
- `/_rollover`
- `/_scripts`[3]
- `/_search`[2]
- `/_search profile`
- `/_shard_stores`
- `/_shrink`[5]
- `/_snapshot`
- `/_stats`
- `/_status`
- `/_tasks`
- `/_template`
- `/_update_by_query`[1]
- `/_validate`

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the `/_tasks` operation along with these operations to verify that the requests completed successfully.
2. DELETE requests to `/_search/scroll` with a message body must specify `"Content-Length"` in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in `scroll_id` values, use the request body, not the query string, to pass `scroll_id` values to Amazon ES.
3. For considerations about using scripts, see the section called "Other Supported Resources" (p. 205).
4. Refers to the `PUT` method. For information about the `GET` method, see the section called "Notable API Differences" (p. 184).

5. See the section called “Shrink” (p. 185).

## Version 5.3

For Elasticsearch 5.3, Amazon ES supports the following operations.

| | | |
|---|---|---|
| • All operations in the index path (such as `/index-name/_forcemerge` and `/index-name/update/id`) **except** `/index-name/_close`<br>• `/_alias`<br>• `/_aliases`<br>• `/_all`<br>• `/_analyze`<br>• `/_bulk`<br>• `/_cat` (except `/_cat/nodeattrs`)<br>• `/_cluster/allocation/explain`<br>• `/_cluster/health`<br>• `/_cluster/pending_tasks`<br>• `/_cluster/settings` for several properties[3]:<br>  • `action.auto_create_index`<br>  • `action.search.shard_count.limit`<br>  • `indices.breaker.fielddata.limit`<br>  • `indices.breaker.request.limit`<br>  • `indices.breaker.total.limit` | • `/_cluster/state`<br>• `/_cluster/stats`<br>• `/_count`<br>• `/_delete_by_query`[1]<br>• `/_explain`<br>• `/_field_caps`<br>• `/_field_stats`<br>• `/_flush`<br>• `/_ingest/pipeline`<br>• `/_mapping`<br>• `/_mget`<br>• `/_msearch`<br>• `/_mtermvectors`<br>• `/_nodes`<br>• `/_percolate`<br>• `/_plugin/kibana`<br>• `/_refresh`<br>• `/_reindex`[1] | • `/_render`<br>• `/_rollover`<br>• `/_search`[2]<br>• `/_search profile`<br>• `/_shard_stores`<br>• `/_shrink`[4]<br>• `/_snapshot`<br>• `/_stats`<br>• `/_status`<br>• `/_tasks`<br>• `/_template`<br>• `/_update_by_query`[1]<br>• `/_validate` |

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the `/_tasks` operation along with these operations to verify that the requests completed successfully.
2. DELETE requests to `/_search/scroll` with a message body must specify `"Content-Length"` in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in `scroll_id` values, use the request body, not the query string, to pass `scroll_id` values to Amazon ES.
3. Refers to the `PUT` method. For information about the `GET` method, see the section called “Notable API Differences” (p. 184).
4. See the section called “Shrink” (p. 185).

## Version 5.1

For Elasticsearch 5.1, Amazon ES supports the following operations.

| | | |
|---|---|---|
| • All operations in the index path (such as `/index-name/_forcemerge` and `/index-name/` | • `/_cluster/state`<br>• `/_cluster/stats` | • `/_render`<br>• `/_rollover` |

update/*id*) **except** /*index-name*/_close

- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cat (except /_cat/nodeattrs)
- /_cluster/allocation/explain
- /_cluster/health
- /_cluster/pending_tasks
- /_cluster/settings for several properties (PUT only):
  - action.auto_create_index
  - action.search.shard_count.limit
  - indices.breaker.fielddata.limit
  - indices.breaker.request.limit
  - indices.breaker.total.limit
- /_count
- /_delete_by_query[1]
- /_explain
- /_field_caps
- /_field_stats
- /_flush
- /_ingest/pipeline
- /_mapping
- /_mget
- /_msearch
- /_mtermvectors
- /_nodes
- /_percolate
- /_plugin/kibana
- /_refresh
- /_reindex[1]
- /_search[2]
- /_search profile
- /_shard_stores
- /_shrink[3]
- /_snapshot
- /_stats
- /_status
- /_tasks
- /_template
- /_update_by_query[1]
- /_validate

1. Cluster configuration changes might interrupt these operations before completion. We recommend that you use the /_tasks operation along with these operations to verify that the requests completed successfully.
2. DELETE requests to /_search/scroll with a message body must specify "Content-Length" in the HTTP header. Most clients add this header by default. To avoid a problem with = characters in scroll_id values, use the request body, not the query string, to pass scroll_id values to Amazon ES.
3. See the section called “Shrink” (p. 185).

## Version 2.3

For Elasticsearch 2.3, Amazon ES supports the following operations.

- All operations in the index path (such as /*index-name*/_forcemerge and /*index-name*/_recovery) **except** /*index-name*/_close
- /_alias
- /_aliases
- /_all
- /_analyze
- /_bulk
- /_cache/clear (index only)
- /_cat (except /_cat/nodeattrs)
- /_cluster/health
- /_cluster/settings for several properties (PUT only):
- /_cluster/stats
- /_count
- /_flush
- /_mapping
- /_mget
- /_msearch
- /_nodes
- /_percolate
- /_plugin/kibana
- /_refresh
- /_render
- /_search
- /_snapshot

  - `indices.breaker.fielddata.limit`
  - `indices.breaker.request.limit`
  - `indices.breaker.total.limit`
  - `threadpool.get.queue_size`
  - `threadpool.bulk.queue_size`
  - `threadpool.index.queue_size`
  - `threadpool.percolate.queue_size`
  - `threadpool.search.queue_size`
  - `threadpool.suggest.queue_size`

- `/_stats`
- `/_status`
- `/_template`

## Version 1.5

For Elasticsearch 1.5, Amazon ES supports the following operations.

- All operations in the index path, such as `/index-name/_optimize` and `/index-name/_warmer`, **except** `/index-name/_close`
- `/_alias`
- `/_aliases`
- `/_all`
- `/_analyze`
- `/_bulk`
- `/_cat`
- `/_cluster/health`
- `/_cluster/settings` for several properties (PUT only):
  - `indices.breaker.fielddata.limit`
  - `indices.breaker.request.limit`
  - `indices.breaker.total.limit`
  - `threadpool.get.queue_size`
  - `threadpool.bulk.queue_size`
  - `threadpool.index.queue_size`
  - `threadpool.percolate.queue_size`
  - `threadpool.search.queue_size`
  - `threadpool.suggest.queue_size`
- `/_cluster/stats`
- `/_count`
- `/_flush`
- `/_mapping`
- `/_mget`
- `/_msearch`
- `/_nodes`
- `/_percolate`
- `/_plugin/kibana`
- `/_plugin/kibana3`
- `/_plugin/migration`
- `/_refresh`
- `/_search`
- `/_snapshot`
- `/_stats`
- `/_status`
- `/_template`

# Amazon Elasticsearch Service Limits

The following tables show limits for Amazon ES resources, including the number of nodes per cluster, the minimum and maximum sizes for EBS volumes, and network limits.

## Cluster and Instance Limits

The following table shows Amazon ES limits for clusters and instances.

| Clusters and Instances | Limit |
|---|---|
| Maximum number of data nodes (including warm nodes) per cluster | 40 (except for the T2 instance types, which have a maximum of 10)<br>**Note**<br>The default limit is 40 data nodes per cluster. To request an increase up to 200 (for Elasticsearch 2.3 or later), create a case with the AWS Support Center. For more information about requesting an increase, see AWS Service Limits. |
| Maximum number of warm nodes per cluster | 45 |
| Maximum number of dedicated master nodes | 5<br>**Note**<br>You can use the T2 instance types for dedicated master nodes (not recommended for production domains) only if the number of data nodes is 10 or fewer. |
| Smallest supported instance type | `t2.micro.elasticsearch` (versions 1.5 and 2.3) and `t2.small.elasticsearch` (version 5.*x* and 6.*x*). |
| Maximum number of domains per account (per Region) | 100 |

For a list of the instance types that Amazon ES supports, see Supported Instance Types (p. 180).

## UltraWarm Storage Limits

The following table lists the UltraWarm instance types and the maximum amount of storage that each type can use. For more information about UltraWarm, see the section called "UltraWarm" (p. 160).

| Instance Type | Maximum Storage |
|---|---|
| `ultrawarm1.medium.elasticsearch` | 1.5 TiB |
| `ultrawarm1.large.elasticsearch` | 20 TiB |

## EBS Volume Size Limits

The following table shows the minimum and maximum sizes for EBS volumes for each instance type that Amazon ES supports. For information about which instance types include instance storage and additional hardware details, see Amazon Elasticsearch Service Pricing.

- If you choose magnetic storage under **EBS volume type** when creating your domain, the maximum volume size is 100 GiB for all instance types except `t2.micro`, `t2.small`, and `t2.medium`. For the maximum sizes listed in the following table, choose one of the SSD options.
- 512 GiB is the maximum volume size that is supported with Elasticsearch version 1.5.
- Some older-generation instance types include instance storage, but also support EBS storage. If you choose EBS storage for one of these instance types, the storage volumes are *not* additive. You can use either an EBS volume or the instance storage, not both.

| Instance Type | Minimum EBS Size | Maximum EBS Size |
|---|---|---|
| `t2.micro.elasticsearch` | 10 GiB | 35 GiB |
| `t2.small.elasticsearch` | 10 GiB | 35 GiB |
| `t2.medium.elasticsearch` | 10 GiB | 35 GiB |
| `m3.medium.elasticsearch` | 10 GiB | 100 GiB |
| `m3.large.elasticsearch` | 10 GiB | 512 GiB |
| `m3.xlarge.elasticsearch` | 10 GiB | 512 GiB |
| `m3.2xlarge.elasticsearch` | 10 GiB | 512 GiB |
| `m4.large.elasticsearch` | 10 GiB | 512 GiB |
| `m4.xlarge.elasticsearch` | 10 GiB | 1 TiB |
| `m4.2xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `m4.4xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `m4.10xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `m5.large.elasticsearch` | 10 GiB | 512 GiB |
| `m5.xlarge.elasticsearch` | 10 GiB | 1 TiB |
| `m5.2xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `m5.4xlarge.elasticsearch` | 10 GiB | 3 TiB |
| `m5.12xlarge.elasticsearch` | 10 GiB | 9 TiB |
| `c4.large.elasticsearch` | 10 GiB | 100 GiB |
| `c4.xlarge.elasticsearch` | 10 GiB | 512 GiB |
| `c4.2xlarge.elasticsearch` | 10 GiB | 1 TiB |
| `c4.4xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `c4.8xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `c5.large.elasticsearch` | 10 GiB | 256 GiB |
| `c5.xlarge.elasticsearch` | 10 GiB | 512 GiB |
| `c5.2xlarge.elasticsearch` | 10 GiB | 1 TiB |
| `c5.4xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `c5.9xlarge.elasticsearch` | 10 GiB | 3.5 TiB |
| `c5.18xlarge.elasticsearch` | 10 GiB | 7 TiB |
| `r3.large.elasticsearch` | 10 GiB | 512 GiB |
| `r3.xlarge.elasticsearch` | 10 GiB | 512 GiB |
| `r3.2xlarge.elasticsearch` | 10 GiB | 512 GiB |

| Instance Type | Minimum EBS Size | Maximum EBS Size |
|---|---|---|
| `r3.4xlarge.elasticsearch` | 10 GiB | 512 GiB |
| `r3.8xlarge.elasticsearch` | 10 GiB | 512 GiB |
| `r4.large.elasticsearch` | 10 GiB | 1 TiB |
| `r4.xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `r4.2xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `r4.4xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `r4.8xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `r4.16xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `r5.large.elasticsearch` | 10 GiB | 1 TiB |
| `r5.xlarge.elasticsearch` | 10 GiB | 1.5 TiB |
| `r5.2xlarge.elasticsearch` | 10 GiB | 3 TiB |
| `r5.4xlarge.elasticsearch` | 10 GiB | 6 TiB |
| `r5.12xlarge.elasticsearch` | 10 GiB | 12 TiB |
| `i2.xlarge.elasticsearch` | 10 GiB | 512 GiB |
| `i2.2xlarge.elasticsearch` | 10 GiB | 512 GiB |
| `i3.large.elasticsearch` | N/A | N/A |
| `i3.xlarge.elasticsearch` | N/A | N/A |
| `i3.2xlarge.elasticsearch` | N/A | N/A |
| `i3.4xlarge.elasticsearch` | N/A | N/A |
| `i3.8xlarge.elasticsearch` | N/A | N/A |
| `i3.16xlarge.elasticsearch` | N/A | N/A |

## Network Limits

The following table shows the maximum size of HTTP request payloads.

| Instance Type | Maximum Size of HTTP Request Payloads |
|---|---|
| `t2.micro.elasticsearch` | 10 MiB |
| `t2.small.elasticsearch` | 10 MiB |
| `t2.medium.elasticsearch` | 10 MiB |
| `m3.medium.elasticsearch` | 10 MiB |
| `m3.large.elasticsearch` | 10 MiB |

| Instance Type | Maximum Size of HTTP Request Payloads |
|---|---|
| `m3.xlarge.elasticsearch` | 100 MiB |
| `m3.2xlarge.elasticsearch` | 100 MiB |
| `m4.large.elasticsearch` | 10 MiB |
| `m4.xlarge.elasticsearch` | 100 MiB |
| `m4.2xlarge.elasticsearch` | 100 MiB |
| `m4.4xlarge.elasticsearch` | 100 MiB |
| `m4.10xlarge.elasticsearch` | 100 MiB |
| `m5.large.elasticsearch` | 10 MiB |
| `m5.xlarge.elasticsearch` | 100 MiB |
| `m5.2xlarge.elasticsearch` | 100 MiB |
| `m5.4xlarge.elasticsearch` | 100 MiB |
| `m5.12xlarge.elasticsearch` | 100 MiB |
| `c4.large.elasticsearch` | 10 MiB |
| `c4.xlarge.elasticsearch` | 100 MiB |
| `c4.2xlarge.elasticsearch` | 100 MiB |
| `c4.4xlarge.elasticsearch` | 100 MiB |
| `c4.8xlarge.elasticsearch` | 100 MiB |
| `c5.large.elasticsearch` | 10 MiB |
| `c5.xlarge.elasticsearch` | 100 MiB |
| `c5.2xlarge.elasticsearch` | 100 MiB |
| `c5.4xlarge.elasticsearch` | 100 MiB |
| `c5.9xlarge.elasticsearch` | 100 MiB |
| `c5.18xlarge.elasticsearch` | 100 MiB |
| `r3.large.elasticsearch` | 10 MiB |
| `r3.xlarge.elasticsearch` | 100 MiB |
| `r3.2xlarge.elasticsearch` | 100 MiB |
| `r3.4xlarge.elasticsearch` | 100 MiB |
| `r3.8xlarge.elasticsearch` | 100 MiB |
| `r4.large.elasticsearch` | 100 MiB |
| `r4.xlarge.elasticsearch` | 100 MiB |
| `r4.2xlarge.elasticsearch` | 100 MiB |

| Instance Type | Maximum Size of HTTP Request Payloads |
|---|---|
| `r4.4xlarge.elasticsearch` | 100 MiB |
| `r4.8xlarge.elasticsearch` | 100 MiB |
| `r4.16xlarge.elasticsearch` | 100 MiB |
| `r5.large.elasticsearch` | 100 MiB |
| `r5.xlarge.elasticsearch` | 100 MiB |
| `r5.2xlarge.elasticsearch` | 100 MiB |
| `r5.4xlarge.elasticsearch` | 100 MiB |
| `r5.12xlarge.elasticsearch` | 100 MiB |
| `i2.xlarge.elasticsearch` | 100 MiB |
| `i2.2xlarge.elasticsearch` | 100 MiB |
| `i3.large.elasticsearch` | 100 MiB |
| `i3.xlarge.elasticsearch` | 100 MiB |
| `i3.2xlarge.elasticsearch` | 100 MiB |
| `i3.4xlarge.elasticsearch` | 100 MiB |
| `i3.8xlarge.elasticsearch` | 100 MiB |
| `i3.16xlarge.elasticsearch` | 100 MiB |

### Java Process Limit

Amazon ES limits Java processes to a heap size of 32 GiB. Advanced users can specify the percentage of the heap used for field data. For more information, see the section called "Advanced Options" (p. 13) and the section called "JVM OutOfMemoryError" (p. 231).

### Domain Policy Limit

Amazon ES limits access policies on domains (p. 64) to 100 KiB.

## Amazon Elasticsearch Service Reserved Instances

Amazon Elasticsearch Service Reserved Instances (RIs) offer significant discounts compared to standard On-Demand Instances. The instances themselves are identical; RIs are just a billing discount applied to On-Demand Instances in your account. For long-lived applications with predictable usage, RIs can provide considerable savings over time.

Amazon ES RIs require one- or three-year terms and have three payment options that affect the discount rate:

- **No Upfront** – You pay nothing upfront. You pay a discounted hourly rate for every hour within the term.

- **Partial Upfront** – You pay a portion of the cost upfront, and you pay a discounted hourly rate for every hour within the term.
- **All Upfront** – You pay the entirety of the cost upfront. You don't pay an hourly rate for the term.

Generally speaking, a larger upfront payment means a larger discount. You can't cancel Reserved Instances—when you reserve them, you commit to paying for the entire term—and upfront payments are nonrefundable. For full details, see Amazon Elasticsearch Service Pricing and FAQ.

**Topics**

## Purchasing Reserved Instances (Console)

The console lets you view your existing Reserved Instances and purchase new ones.

### To purchase a reservation

1. Go to https://aws.amazon.com, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Elasticsearch Service**.
3. Choose **Reserved Instances**.

   On this page, you can view your existing reservations. If you have many reservations, you can filter them to more easily identify and view a particular reservation.

   **Tip**
   If you don't see the **Reserved Instances** link, create a domain (p. 9) in the region.
4. Choose **Purchase Reserved Instance**.
5. For **Reservation Name**, type a unique, descriptive name.
6. Choose an instance type, size, and number of instances. For guidance, see the section called "Sizing Amazon ES Domains" (p. 171).
7. Choose a term length and payment option.
8. Review the payment details carefully.
9. Choose **Submit**.
10. Review the purchase summary carefully. Purchases of Reserved Instances are non-refundable.
11. Choose **Purchase**.

## Purchasing Reserved Instances (AWS CLI)

The AWS CLI has commands for viewing offerings, purchasing a reservation, and viewing your reservations. The following command and sample response show the offerings for a given AWS Region:

```
aws es describe-reserved-elasticsearch-instance-offerings --region us-east-1
{
  "ReservedElasticsearchInstanceOfferings": [
    {
      "FixedPrice": x,
      "ReservedElasticsearchInstanceOfferingId": "1a2a3a4a5-1a2a-3a4a-5a6a-1a2a3a4a5a6a",
      "RecurringCharges": [
        {
          "RecurringChargeAmount": y,
          "RecurringChargeFrequency": "Hourly"
        }
      ],
      "UsagePrice": 0.0,
      "PaymentOption": "PARTIAL_UPFRONT",
      "Duration": 31536000,
      "ElasticsearchInstanceType": "m4.2xlarge.elasticsearch",
      "CurrencyCode": "USD"
    }
  ]
}
```

For an explanation of each return value, see the following table.

| Field | Description |
|---|---|
| `FixedPrice` | The upfront cost of the reservation. |
| `ReservedElasticsearchInstanceOfferingId` | The offering ID. Make note of this value if you want to reserve the offering. |
| `RecurringCharges` | The hourly rate for the reservation. |
| `UsagePrice` | A legacy field. For Amazon ES, this value is always 0. |
| `PaymentOption` | No Upfront, Partial Upfront, or All Upfront. |
| `Duration` | Length of the term in seconds:<br>• 31536000 seconds is one year.<br>• 94608000 seconds is three years. |
| `ElasticsearchInstanceType` | The instance type for the reservation. For information about the hardware resources that are allocated to each instance type, see Amazon Elasticsearch Service Pricing. |
| `CurrencyCode` | The currency for `FixedPrice` and `RecurringChargeAmount`. |

This next example purchases a reservation:

```
aws es purchase-reserved-elasticsearch-instance-offering --reserved-elasticsearch-instance-
offering-id 1a2a3a4a5-1a2a-3a4a-5a6a-1a2a3a4a5a6a --reservation-name my-reservation --
instance-count 3 --region us-east-1
{
  "ReservationName": "my-reservation",
  "ReservedElasticsearchInstanceId": "9a8a7a6a-5a4a-3a2a-1a0a-9a8a7a6a5a4a"
}
```

Finally, you can list your reservations for a given region using the following example:

```
aws es describe-reserved-elasticsearch-instances --region us-east-1
{
  "ReservedElasticsearchInstances": [
    {
      "FixedPrice": x,
      "ReservedElasticsearchInstanceOfferingId": "1a2a3a4a5-1a2a-3a4a-5a6a-1a2a3a4a5a6a",
      "ReservationName": "my-reservation",
      "PaymentOption": "PARTIAL_UPFRONT",
      "UsagePrice": 0.0,
      "ReservedElasticsearchInstanceId": "9a8a7a6a-5a4a-3a2a-1a0a-9a8a7a6a5a4a",
      "RecurringCharges": [
        {
          "RecurringChargeAmount": y,
          "RecurringChargeFrequency": "Hourly"
        }
      ],
      "State": "payment-pending",
      "StartTime": 1522872571.229,
      "ElasticsearchInstanceCount": 3,
      "Duration": 31536000,
      "ElasticsearchInstanceType": "m4.2xlarge.elasticsearch",
      "CurrencyCode": "USD"
    }
  ]
}
```

**Note**
`StartTime` is Unix epoch time, which is the number of seconds that have passed since midnight UTC of 1 January 1970. For example, 1522872571 epoch time is 20:09:31 UTC of 4 April 2018. You can use online converters.

To learn more about the commands used in the preceding examples, see the AWS CLI Command Reference.

## Purchasing Reserved Instances (AWS SDKs)

The AWS SDKs (except the Android and iOS SDKs) support all the operations that are defined in the Amazon ES Configuration API Reference (p. 236), including the following:

- `DescribeReservedElasticsearchInstanceOfferings`
- `PurchaseReservedElasticsearchInstanceOffering`
- `DescribeReservedElasticsearchInstances`

For more information about installing and using the AWS SDKs, see AWS Software Development Kits.

## Examining Costs

Cost Explorer is a free tool that you can use to view your spending data for the past 13 months. Analyzing this data helps you identify trends and understand if RIs fit your use case. If you already have RIs, you can group by **Purchase Option** and show amortized costs to compare that spending to your spending for On-Demand Instances. You can also set usage budgets to make sure you are taking full advantage of your reservations. For more information, see Analyzing Your Costs with Cost Explorer in the *AWS Billing and Cost Management User Guide*.

# Other Supported Resources

**bootstrap.mlockall**

The service enables `bootstrap.mlockall` in `elasticsearch.yml`, which locks JVM memory and prevents the operating system from swapping it to disk. This applies to all supported instance types except for the following:

- `t2.micro.elasticsearch`
- `t2.small.elasticsearch`
- `t2.medium.elasticsearch`

**Scripting module**

The service supports scripting for Elasticsearch 5.*x* and later domains. The service does not support scripting for 1.5 or 2.3.

Supported scripting options include the following:

- Painless
- Lucene Expressions
- Mustache

For Elasticsearch 5.5 and later domains, Amazon ES supports stored scripts using the `_scripts` endpoint. Elasticsearch 5.3 and 5.1 domains support inline scripts only.

**TCP transport**

The service supports HTTP on port 80 and HTTPS over port 443, but does not support TCP transport.

# Amazon Elasticsearch Service Tutorials

This chapter includes several start-to-finish tutorials for working with Amazon Elasticsearch Service, including how to migrate to the service, build a simple search application, and create a Kibana dashboard.

**Topics**

## Migrating to Amazon Elasticsearch Service

Index snapshots are a popular way to migrate from a self-managed Elasticsearch cluster to Amazon Elasticsearch Service. Broadly, the process consists of the following steps:

1. Take a snapshot of the existing cluster, and upload the snapshot to an Amazon S3 bucket.
2. Create an Amazon ES domain.
3. Give Amazon ES permissions to access the bucket, and give your user account permissions to work with snapshots.
4. Restore the snapshot on the Amazon ES domain.

This walkthrough provides more detailed steps and alternate options, where applicable.

### Take and Upload the Snapshot

Although you can use the repository-s3 plugin to take snapshots directly to S3, you have to install the plugin on every node, tweak `elasticsearch.yml`, restart each node, add your AWS credentials, and finally take the snapshot. The plugin is a great option for ongoing use or for migrating larger clusters.

For smaller clusters, a one-time approach is to take a shared file system snapshot and then use the AWS CLI to upload it to S3. If you already have a snapshot, skip to step 4.

**To take a snapshot and upload it to Amazon S3**

1. Add the `path.repo` setting to `elasticsearch.yml` on all nodes, and then restart each node.

```
path.repo: ["/my/shared/directory/snapshots"]
```

2. Register the snapshot repository:

```
PUT _snapshot/migration-repository
{
  "type": "fs",
  "settings": {
    "location": "/my/shared/directory/snapshots"
  }
}
```

3. Take the snapshot:

```
PUT _snapshot/migration-repository/migration-snapshot
{
  "indices": "migration-index1,migration-index2,other-indices-*",
  "include_global_state": false
}
```

4. Install the AWS CLI, and run `aws configure` to add your credentials.
5. Navigate to the snapshot directory. Then run the following commands to create a new S3 bucket and upload the contents of the snapshot directory to that bucket:

```
aws s3 mb s3://migration-bucket --region us-west-2
aws s3 sync . s3://migration-bucket --sse AES256
```

   Depending on the size of the snapshot and the speed of your internet connection, this operation can take a while.

## Create the Domain

Although the console is the easiest way to create a domain, in this case, you already have the terminal open and the AWS CLI installed. Modify the following command to create a domain that fits your needs:

```
aws es create-elasticsearch-domain \
  --domain-name migration-domain \
  --elasticsearch-version 7.4 \
  --elasticsearch-cluster-config InstanceType=c5.large.elasticsearch,InstanceCount=2 \
  --ebs-options EBSEnabled=true,VolumeType=gp2,VolumeSize=100 \
  --node-to-node-encryption-options Enabled=true \
  --encryption-at-rest-options Enabled=true \
  --domain-endpoint-options EnforceHTTPS=true,TLSSecurityPolicy=Policy-Min-TLS-1-2-2019-07
 \
  --advanced-security-options
 Enabled=true,InternalUserDatabaseEnabled=true,MasterUserOptions='{MasterUserName=master-
user,MasterUserPassword=master-user-password}' \
  --access-policies '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"AWS":["*"]},"Action":["es:ESHttp*"],"Resource":"arn:aws:es:us-
west-2:123456789012:domain/migration-domain/*"}]}' \
  --region us-west-2
```

As is, the command creates an internet-accessible domain with two data nodes, each with 100 GiB of storage. It also enables fine-grained access control (p. 76) with HTTP basic authentication and all encryption settings. Use the Amazon ES console if you need a more advanced security configuration, such as a VPC.

Before issuing the command, change the domain name, master user credentials, and account number. Specify the same region that you used for the S3 bucket and an Elasticsearch version that is compatible with your snapshot.

**Important**
Snapshots are only forward-compatible, and only by one major version. For example, you can't restore a snapshot from a 2.*x* cluster on a 1.*x* cluster or a 6.*x* cluster, only a 2.*x* or 5.*x* cluster. Minor version matters, too. You can't restore a snapshot from a self-managed 5.3.3 cluster on a 5.3.2 Amazon ES domain. We recommend choosing the most-recent version of Elasticsearch that your snapshot supports.

## Provide Permissions

In the AWS Identity and Access Management (IAM) console, create a role with the following permissions and trust relationship. Name the role `AmazonESSnapshotRole` so that it's easy to find.

**Permissions**

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Action": [
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::migration-bucket"
      ]
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::migration-bucket/*"
      ]
    }
  ]
}
```

**Trust Relationship**

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Principal": {
        "Service": "es.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

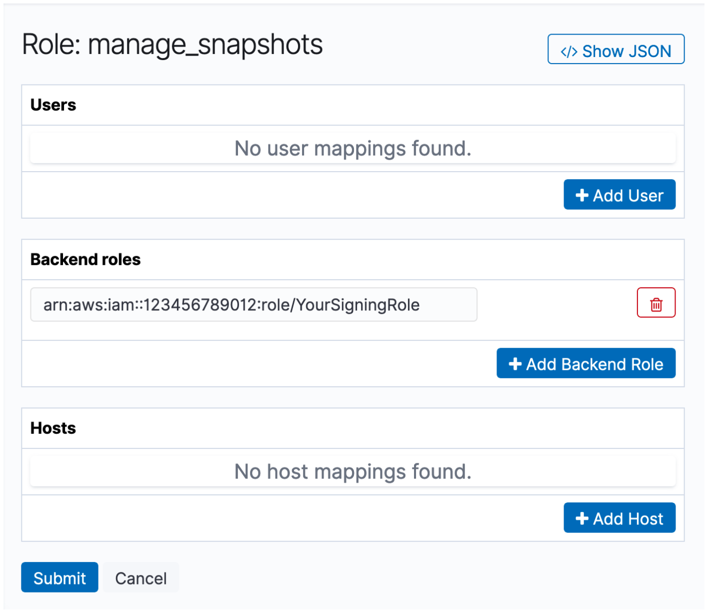
Then give your personal IAM user or role—whatever you used to configure the AWS CLI earlier—permissions to assume `AmazonESSnapshotRole`. Create the following policy and attach it to your identity.

**Permissions**

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/AmazonESSnapshotRole"
    }
  ]
}
```

Then log in to Kibana using the master user credentials you specified when you created the Amazon ES domain. You can find the Kibana URL in the Amazon ES console. It takes the form of `https://domain-endpoint/_plugin/kibana/`.

In Kibana, choose **Security**, **Role Mappings**, and **Add**. For **Role,** choose **manage_snapshots**. Then specify the ARN for your personal IAM user or role in the appropriate field. User ARNs go in the **Users** section. Role ARNs go in the **Backend roles** section. This step uses fine-grained access control (p. 84) to give your identity permissions to work with snapshots.



Role: manage_snapshots

Users: No user mappings found.

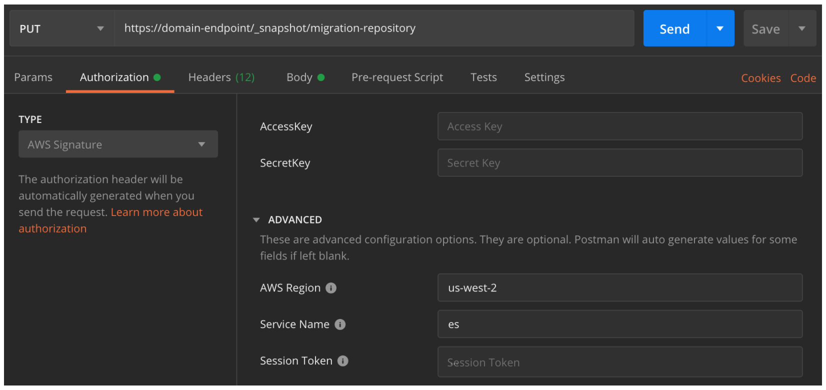
Backend roles: arn:aws:iam::123456789012:role/YourSigningRole

Hosts: No host mappings found.

## Restore the Snapshot

At this point, you have two ways to access your Amazon ES domain: HTTP basic authentication with your master user credentials or AWS authentication using your IAM credentials. Because snapshots use Amazon S3, which has no concept of the master user, you must use your IAM credentials to register the snapshot repository with your Amazon ES domain.

Most programming languages have libraries to assist with signing requests (p. 113), but the simpler approach is to use a tool like Postman and put your IAM credentials into the **Authorization** section.



PUT https://domain-endpoint/_snapshot/migration-repository

TYPE: AWS Signature

The authorization header will be automatically generated when you send the request. Learn more about authorization

AccessKey: Access Key

SecretKey: Secret Key

ADVANCED

These are advanced configuration options. They are optional. Postman will auto generate values for some fields if left blank.

AWS Region: us-west-2

Service Name: es

Session Token: Session Token

### To restore the snapshot

1. Regardless of how you choose to sign your requests, the first step is to register the repository:

```
PUT _snapshot/migration-repository
{
  "type": "s3",
  "settings": {
    "bucket": "migration-bucket",
    "region": "us-west-2",
    "role_arn": "arn:aws:iam::123456789012:role/AmazonESSnapshotRole"
  }
}
```

2. Then list the snapshots in the repository, and find the one you want to restore. At this point, you can continue using Postman or switch to a tool like curl.

   **Shorthand**

   ```
   GET _snapshot/migration-repository/_all
   ```

   **curl**

   ```
   curl -XGET -u master-user:master-user-password https://domain-endpoint/
   _snapshot/migration-repository/_all
   ```

3. Restore the snapshot.

   **Shorthand**

   ```
   POST _snapshot/migration-repository/migration-snapshot/_restore
   {
     "indices": "migration-index1,migration-index2,other-indices-*",
     "include_global_state": false
   }
   ```

   **curl**

   ```
   curl -XPOST -u master-user:master-user-password https://domain-endpoint/
   _snapshot/migration-repository/migration-snapshot/_restore \
     -H 'Content-Type: application/json' \
     -d '{"indices":"migration-index1,migration-index2,other-indices-
   *","include_global_state":false}'
   ```

4. Finally, verify that your indices restored as expected.

   **Shorthand**

   ```
   GET _cat/indices?v
   ```

   **curl**

   ```
   curl -XGET -u master-user:master-user-password https://domain-endpoint/_cat/indices?v
   ```

At this point, the migration is complete. You might configure your clients to use the new Amazon ES endpoint, resize the domain (p. 171) to suit your workload, check the shard count for your indices, switch to an IAM master user (p. 81), or start building Kibana dashboards.

# Creating a Search Application with Amazon Elasticsearch Service

A common way to create a search application with Amazon ES is to use web forms to send user queries to a server. Then you can authorize the server to call the Elasticsearch APIs directly and have the server send requests to Amazon ES.

If you want to write client-side code that doesn't rely on a server, however, you should compensate for the security and performance risks. Allowing unsigned, public access to the Elasticsearch APIs is inadvisable. Users might access unsecured endpoints or impact cluster performance through overly broad queries (or too many queries).

This chapter presents a solution: use Amazon API Gateway to restrict users to a subset of the Elasticsearch APIs and AWS Lambda to sign requests from API Gateway to Amazon ES.

**Note**
Standard API Gateway and Lambda pricing applies, but within the limited usage of this tutorial, costs should be negligible.

## Step 1: Index Sample Data

A prerequisite for these steps is an Amazon ES domain. Download sample-movies.zip, unzip it, and use the `_bulk` API to add the 5,000 documents to the `movies` index:

```
POST https://search-my-domain.us-west-1.es.amazonaws.com/_bulk
{ "index": { "_index": "movies", "_type": "movie", "_id": "tt1979320" } }
{"fields":{"directors":["Ron
 Howard"],"release_date":"2013-09-02T00:00:00Z","rating":8.3,"genres":
["Action","Biography","Drama","Sport"],"image_url":"http://ia.media-imdb.com/images/
M/MV5BMTQyMDE0MTY0OV5BMl5BanBnXkFtZTcwMjI2OTI0OQ@@._V1_SX400_.jpg","plot":"A re-
creation of the merciless 1970s rivalry between Formula One rivals James Hunt and Niki
 Lauda.","title":"Rush","rank":2,"running_time_secs":7380,"actors":["Daniel Brühl","Chris
 Hemsworth","Olivia Wilde"],"year":2013},"id":"tt1979320","type":"add"}
{ "index": { "_index": "movies", "_type": "movie", "_id": "tt1951264" } }
{"fields":{"directors":["Francis Lawrence"],"release_date":"2013-11-11T00:00:00Z","genres":
["Action","Adventure","Sci-Fi","Thriller"],"image_url":"http://ia.media-imdb.com/images/
M/MV5BMTAyMjQ3OTAxMzNeQTJeQWpwZ15BbWU4MDU0NzA1MzAx._V1_SX400_.jpg","plot":"Katniss
 Everdeen and Peeta Mellark become targets of the Capitol after their victory in the 74th
 Hunger Games sparks a rebellion in the Districts of Panem.","title":"The Hunger Games:
 Catching Fire","rank":4,"running_time_secs":8760,"actors":["Jennifer Lawrence","Josh
 Hutcherson","Liam Hemsworth"],"year":2013},"id":"tt1951264","type":"add"}
...
```

To learn more, see *Indexing Data* (p. 130).

## Step 2: Create the API

Using API Gateway to create a more limited API simplifies the process of interacting with the Elasticsearch `_search` API. It also lets you enable security features like Amazon Cognito authentication and request throttling. Create and deploy an API according to the following table.

| Setting | Values |
|---|---|
| API | Type: New API<br><br>**Settings** |

| **Setting** | **Values** |
|---|---|
| | API name: search-es-api<br><br>Description: Public API for searching an Amazon Elasticsearch Service domain<br><br>Endpoint type: Regional |
| Resource | / |
| HTTP Method | `GET` |
| Method Request | **Settings**<br><br>Authorization: none<br><br>Request validator: Validate query string parameters and headers<br><br>API key required: false<br><br>**URL Query String Parameters**<br><br>Name: q<br><br>Required: Yes |
| Integration Request | Integration type: Lambda function<br><br>Use Lambda proxy integration: Yes<br><br>Lambda Region: *`us-west-1`*<br><br>Lambda function: search-es-lambda<br><br>Invoke with caller credentials: No<br><br>Credentials cache: Do not add caller credentials to cache key<br><br>Use default timeout: Yes |
| Stage | Name: search-es-api-test<br><br>**Default Method Throttling**<br><br>Enable throttling: Yes<br><br>Rate: 1000<br><br>Burst: 500 |

These settings configure an API that has only one method: a `GET` request to the endpoint root (`https://`*`some-id`*`.execute-api.`*`us-west-1`*`.amazonaws.com/search-es-api-test`). The request requires a single parameter (`q`), the query string to search for. When called, the method passes the request to Lambda, which executes the `search-es-lambda` function. For more information, see Creating an API in Amazon API Gateway and Deploying an API in Amazon API Gateway.

## Step 3: Create the Lambda Function

In this solution, API Gateway passes requests to the following Python 2.7 Lambda function, which queries Amazon ES and returns results:

```
import boto3
import json
import requests
from requests_aws4auth import AWS4Auth

region = '' # For example, us-west-1
service = 'es'
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

host = '' # For example, search-mydomain-id.us-west-1.es.amazonaws.com
index = 'movies'
url = 'https://' + host + '/' + index + '/_search'

# Lambda execution starts here
def handler(event, context):

    # Put the user query into the query DSL for more accurate search results.
    # Note that certain fields are boosted (^).
    query = {
        "size": 25,
        "query": {
            "multi_match": {
                "query": event['queryStringParameters']['q'],
                "fields": ["fields.title^4", "fields.plot^2", "fields.actors",
 "fields.directors"]
            }
        }
    }

    # ES 6.x requires an explicit Content-Type header
    headers = { "Content-Type": "application/json" }

    # Make the signed HTTP request
    r = requests.get(url, auth=awsauth, headers=headers, data=json.dumps(query))

    # Create the response and add some extra content to support CORS
    response = {
        "statusCode": 200,
        "headers": {
            "Access-Control-Allow-Origin": '*'
        },
        "isBase64Encoded": False
    }

    # Add the search results to the response
    response['body'] = r.text
    return response
```

The function must have the following trigger.

| Trigger | API | Deployment Stage | Security |
|---|---|---|---|
| API Gateway | search-es-api | search-es-api-test | Open |

Because this sample function uses external libraries, you must create a deployment package and upload it to Lambda for the code to work. For more information about creating Lambda functions and deployment packages, see Creating a Deployment Package (Python) in the *AWS Lambda Developer Guide* and the section called "Creating the Lambda Deployment Package" (p. 133) in this guide.

## Step 4: Modify the Domain Access Policy

Your Amazon ES domain must allow the Lambda function to make `GET` requests to the `movies` index. The following policy provides `search-es-role` (created through Lambda) access to the `movies` index:

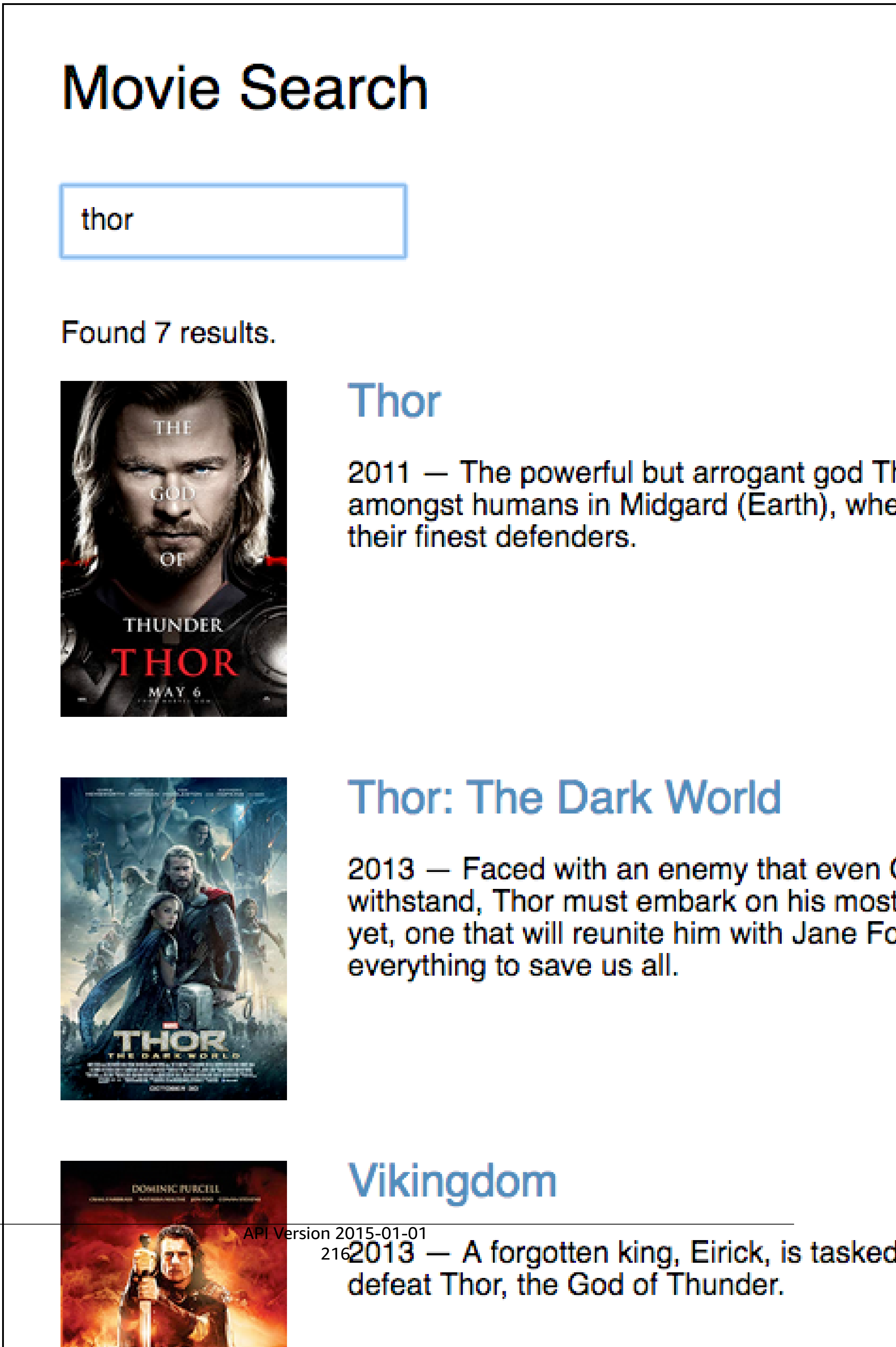
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/service-role/search-es-role"
      },
      "Action": "es:ESHttpGet",
      "Resource": "arn:aws:es:us-west-1:123456789012:domain/web/movies/_search"
    }
  ]
}
```

For more information, see the section called “Configuring Access Policies” (p. 13).

## Step 5: Test the Web Application

**To test the web application**

1. Download sample-site.zip, unzip it, and open `scripts/search.js` in your favorite text editor.
2. Update the `apigatewayendpoint` variable to point to your API Gateway endpoint. The endpoint takes the form of `https://`*`some-id`*`.execute-api.`*`us-west-1`*`.amazonaws.com/search-es-api-test`.
3. Open `index.html` and try running searches for *thor*, *house*, and a few other terms.



Movie Search

thor

Found 7 results.

Thor

2011 — The powerful but arrogant god Th
amongst humans in Midgard (Earth), whe
their finest defenders.

Thor: The Dark World

2013 — Faced with an enemy that even C
withstand, Thor must embark on his most
yet, one that will reunite him with Jane Fo
everything to save us all.

Vikingdom

2013 — A forgotten king, Eirick, is tasked
defeat Thor, the God of Thunder.

## Next Steps

This chapter is just a starting point to demonstrate a concept. You might consider the following modifications:

- Add your own data to the Amazon ES domain.
- Add methods to your API.
- In the Lambda function, modify the search query or boost different fields.
- Style the results differently or modify `search.js` to display different fields to the user.

# Visualizing Customer Support Calls with Amazon Elasticsearch Service and Kibana

This chapter is a full walkthrough of the following situation: a business receives some number of customer support calls and wants to analyze them. What is the subject of each call? How many were positive? How many were negative? How can managers search or review the the transcripts of these calls?

A manual workflow might involve employees listening to recordings, noting the subject of each call, and deciding whether or not the customer interaction was positive.

Such a process would be extremely labor-intensive. Assuming an average time of 10 minutes per call, each employee could listen to only 48 calls per day. Barring human bias, the data they generate would be highly accurate, but the *amount* of data would be minimal: just the subject of the call and a boolean for whether or not the customer was satisfied. Anything more involved, such as a full transcript, would take a huge amount of time.

Using Amazon S3, Amazon Transcribe, Amazon Comprehend, and Amazon Elasticsearch Service (Amazon ES), you can automate a similar process with very little code and end up with much more data. For example, you can get a full transcript of the call, keywords from the transcript, and an overall "sentiment" of the call (positive, negative, neutral, or mixed). Then you can use Elasticsearch and Kibana to search and visualize the data.

While you can use this walkthrough as-is, the intent is to spark ideas about how to enrich your JSON documents before you index them in Amazon ES.

**Estimated Costs**

In general, performing the steps in this walkthrough should cost less than $2. The walkthrough uses the following resources:

- S3 bucket with less than 100 MB transferred and stored

  To learn more, see Amazon S3 Pricing.
- Amazon ES domain with one `t2.medium` instance and 10 GiB of EBS storage for several hours

  To learn more, see Amazon Elasticsearch Service Pricing.
- Several calls to Amazon Transcribe

  To learn more, see Amazon Transcribe Pricing.
- Several natural language processing calls to Amazon Comprehend

  To learn more, see Amazon Comprehend Pricing.

**Topics**

## Step 1: Configure Prerequisites

Before proceeding, you must have the following resources.

| Prerequisite | Description |
|---|---|
| Amazon S3 Bucket | For more information, see Creating a Bucket in the *Amazon Simple Storage Service Getting Started Guide*. |
| Amazon ES Domain | The destination for data. For more information, see Creating Amazon ES Domains (p. 9). |

If you don't already have these resources, you can create them using the following AWS CLI commands:

```
aws s3 mb s3://my-transcribe-test --region us-west-2
```

```
aws es create-elasticsearch-domain --domain-name my-transcribe-test --elasticsearch-version
 6.2 --elasticsearch-cluster-config  InstanceType=t2.medium.elasticsearch,InstanceCount=1
 --ebs-options EBSEnabled=true,VolumeType=standard,VolumeSize=10 --access-
policies '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"AWS":"arn:aws:iam::123456789012:root"},"Action":"es:*","Resource":"arn:aws:es:us-
west-2:123456789012:domain/my-transcribe-test/*"}]}' --region us-west-2
```

**Note**
These commands use the `us-west-2` region, but you can use any region that Amazon Comprehend supports. To learn more, see the AWS General Reference.

## Step 2: Copy Sample Code

1. Copy and paste the following sample code into a new file named `call-center.py`:

```
import boto3
import datetime
import json
import requests
from requests_aws4auth import AWS4Auth
import time
import urllib2

# Variables to update
audio_file_name = '' # For example, 000001.mp3
bucket_name = '' # For example, my-transcribe-test
domain = '' # For example, https://search-my-transcribe-test-12345.us-
west-2.es.amazonaws.com
index = 'support-calls'
type = 'call'
es_region = 'us-west-2'
```

```
# Upload audio file to S3.
s3_client = boto3.client('s3')

audio_file = open(audio_file_name, 'r')

print('Uploading ' + audio_file_name + '...')
response = s3_client.put_object(
    Body=audio_file,
    Bucket=bucket_name,
    Key=audio_file_name,
)

response = s3_client.get_bucket_location(
    Bucket=bucket_name
)

bucket_region = response['LocationConstraint']

# Build the URL to the audio file on S3.
mp3_uri = 'https://s3-' + bucket_region + '.amazonaws.com/' + bucket_name + '/' +
 audio_file_name

# Start transcription job.
transcribe_client = boto3.client('transcribe')

print('Starting transcription job...')
response = transcribe_client.start_transcription_job(
    TranscriptionJobName=audio_file_name,
    LanguageCode='en-US',
    MediaFormat='mp3',
    Media={
        'MediaFileUri': mp3_uri
    },
    Settings={
        'ShowSpeakerLabels': True,
        'MaxSpeakerLabels': 2 # assumes two people on a phone call
    }
)

# Wait for the transcription job to finish.
print('Waiting for job to complete...')
while True:
    response =
 transcribe_client.get_transcription_job(TranscriptionJobName=audio_file_name)
    if response['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
 'FAILED']:
        break
    else:
        print('Still waiting...')
    time.sleep(10)

transcript_uri = response['TranscriptionJob']['Transcript']['TranscriptFileUri']

# Open the JSON file, read it, and get the transcript.
response = urllib2.urlopen(transcript_uri)
raw_json = response.read()
loaded_json = json.loads(raw_json)
transcript = loaded_json['results']['transcripts'][0]['transcript']

# Send transcript to Comprehend for key phrases and sentiment.
comprehend_client = boto3.client('comprehend')

# If necessary, trim the transcript.
# If the transcript is more than 5 KB, the Comprehend calls fail.
if len(transcript) > 5000:
    trimmed_transcript = transcript[:5000]
else:
    trimmed_transcript = transcript

print('Detecting key phrases...')
response = comprehend_client.detect_key_phrases(
    Text=trimmed_transcript,
    LanguageCode='en'
)

keywords = []
for keyword in response['KeyPhrases']:
    keywords.append(keyword['Text'])

print('Detecting sentiment...')
response = comprehend_client.detect_sentiment(
    Text=trimmed_transcript,
    LanguageCode='en'
)

sentiment = response['Sentiment']

# Build the Amazon Elasticsearch Service URL.
id = audio_file_name.strip('.mp3')
url = domain + '/' + index + '/' + type + '/' + id

# Create the JSON document.
json_document = {'transcript': transcript, 'keywords': keywords, 'sentiment':
 sentiment, 'timestamp': datetime.datetime.now().isoformat()}

# Provide all details necessary to sign the indexing request.
credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, es_region, 'es',
 session_token=credentials.token)

# Add explicit header for Elasticsearch 6.x.
headers = {'Content-Type': 'application/json'}

# Index the document.
print('Indexing document...')
response = requests.put(url, auth=awsauth, json=json_document, headers=headers)

print(response)
print(response.json())
```

2. Update the initial six variables.
3. Install the required packages using the following commands:

```
pip install boto3
pip install requests
pip install requests_aws4auth
```

4. Place your MP3 in the same directory as `call-center.py` and run the script. A sample output follows:

```
$ python call-center.py
Uploading 000001.mp3...
Starting transcription job...
Waiting for job to complete...
Still waiting...
Still waiting...
Still waiting...
Still waiting...
Still waiting...
Still waiting...
Still waiting...
Detecting key phrases...
Detecting sentiment...
Indexing document...
<Response [201]>
{u'_type': u'call', u'_seq_no': 0, u'_shards': {u'successful': 1, u'failed': 0,
 u'total': 2}, u'_index': u'support-calls4', u'_version': 1, u'_primary_term': 1,
 u'result': u'created', u'_id': u'000001'}
```

`call-center.py` performs a number of operations:

1. The script uploads an audio file (in this case, an MP3, but Amazon Transcribe supports several formats) to your S3 bucket.
2. It sends the audio file's URL to Amazon Transcribe and waits for the transcription job to finish.

   The time to finish the transcription job depends on the length of the audio file. Assume minutes, not seconds.

   **Tip**
   To improve the quality of the transcription, you can configure a custom vocabulary for Amazon Transcribe.
3. After the transcription job finishes, the script extracts the transcript, trims it to 5,000 characters, and sends it to Amazon Comprehend for keyword and sentiment analysis.
4. Finally, the script adds the full transcript, keywords, sentiment, and current time stamp to a JSON document and indexes it in Amazon ES.

**Tip**
LibriVox has public domain audiobooks that you can use for testing.

## (Optional) Step 3: Add Sample Data

If you don't have a bunch of call recordings handy—and who does?—you can index (p. 130) the sample documents in sample-calls.zip, which are comparable to what `call-center.py` produces.

1. Create a file named `bulk-helper.py`:

```
import boto3
from elasticsearch import Elasticsearch, RequestsHttpConnection
import json
from requests_aws4auth import AWS4Auth

host = '' # For example, my-test-domain.us-west-2.es.amazonaws.com
region = '' # For example, us-west-2
service = 'es'

bulk_file = open('sample-calls.bulk', 'r').read()

credentials = boto3.Session().get_credentials()
awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service,
 session_token=credentials.token)

es = Elasticsearch(
    hosts = [{'host': host, 'port': 443}],
    http_auth = awsauth,
    use_ssl = True,
    verify_certs = True,
    connection_class = RequestsHttpConnection
)
```

```
response = es.bulk(bulk_file)
print(json.dumps(response, indent=2, sort_keys=True))
```

**Note**
This code sample uses Python 2 and might require modifications for Python 3.

2. Update the initial two variables for `host` and `region`.
3. Install the required package using the following command:

```
pip install elasticsearch
```

4. Download and unzip sample-calls.zip.
5. Place `sample-calls.bulk` in the same directory as `bulk-helper.py` and run the helper. A sample output follows:

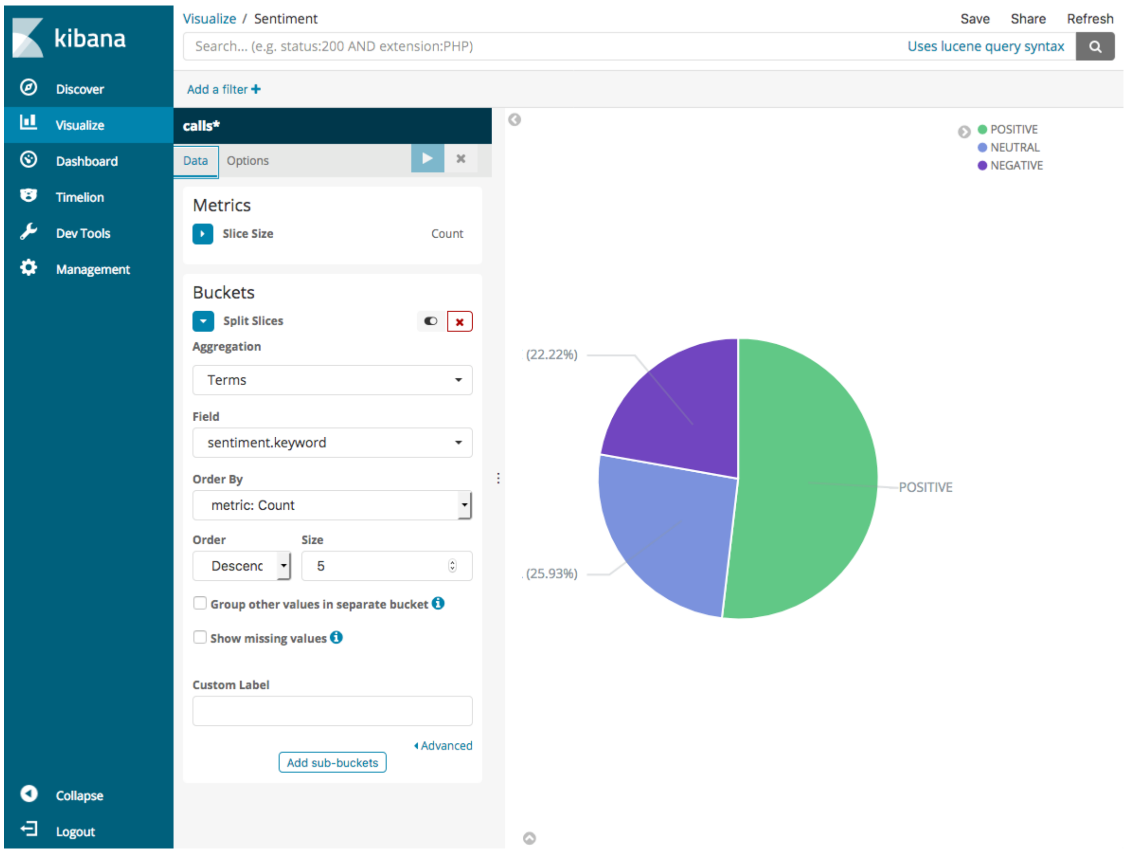
```
$ python bulk-helper.py
{
  "errors": false,
  "items": [
    {
      "index": {
        "_id": "1",
        "_index": "test-data",
        "_primary_term": 1,
        "_seq_no": 42,
        "_shards": {
          "failed": 0,
          "successful": 1,
          "total": 2
        },
        "_type": "call",
        "_version": 9,
        "result": "updated",
        "status": 200
      }
    },
    ...
  ],
  "took": 27
}
```

## Step 4: Analyze and Visualize Your Data

Now that you have some data in Amazon ES, you can visualize it using Kibana.

1. Navigate to `https://search-`*`domain`*`.`*`region`*`.es.amazonaws.com/_plugin/kibana`.
2. Before you can use Kibana, you need an index pattern. Kibana uses index patterns to narrow your analysis to one or more indices. To match the `support-calls` index that `call-center.py` created, define an index pattern of `support*`, and then choose **Next step**.
3. For **Time Filter field name**, choose **timestamp**.
4. Now you can start creating visualizations. Choose **Visualize**, and then add a new visualization.
5. Choose the pie chart and the `support*` index pattern.
6. The default visualization is basic, so choose **Split Slices** to create a more interesting visualization.

   For **Aggregation**, choose **Terms**. For **Field**, choose **sentiment.keyword**. Then choose **Apply changes** and **Save**.



Metrics

Slice Size: Count

Buckets

Split Slices

Aggregation: Terms

Field: sentiment.keyword

Order By: metric: Count

Order: Descenc

Size: 5

Group other values in separate bucket

Show missing values

Custom Label

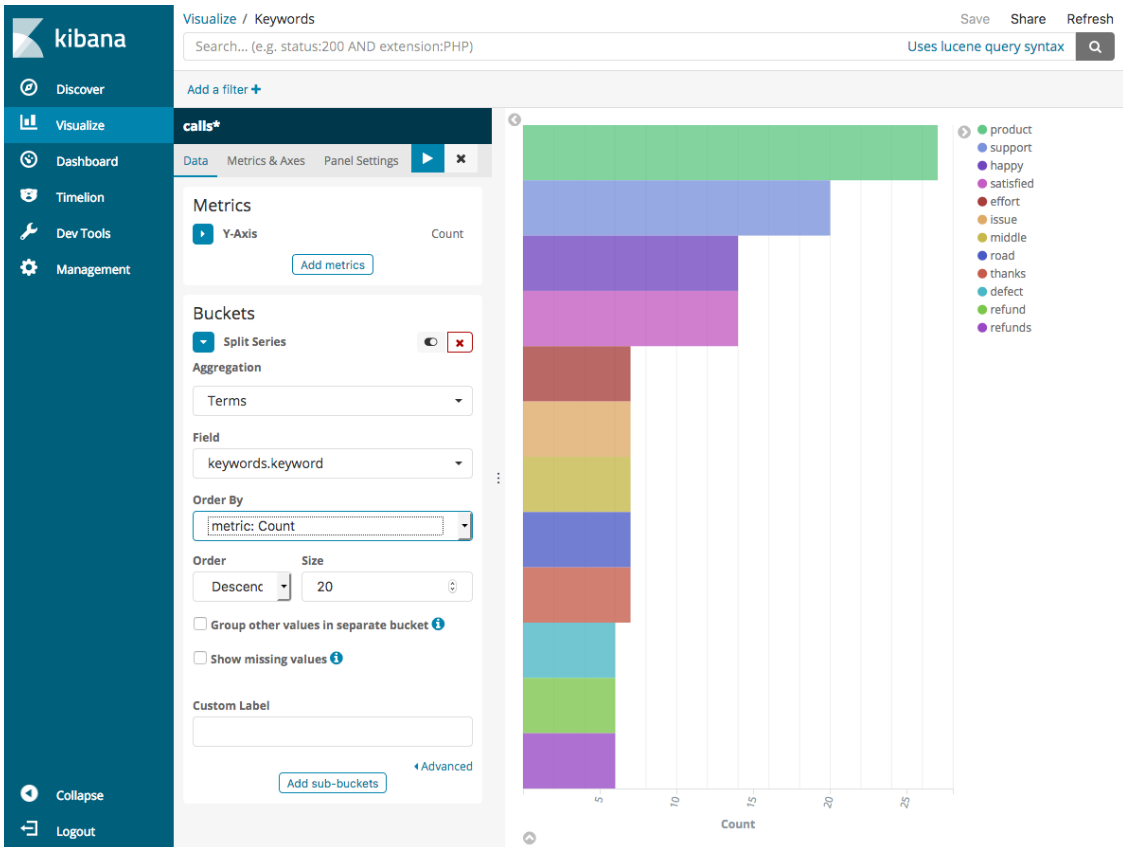
Advanced

Add sub-buckets

7. Return to the **Visualize** page, and add another visualization. This time, choose the horizontal bar chart.

8. Choose **Split Series**.

   For **Aggregation**, choose **Terms**. For **Field**, choose **keywords.keyword** and change **Size** to 20. Then choose **Apply Changes** and **Save**.



Metrics

Y-Axis Count

Buckets

Split Series

Aggregation: Terms

Field: keywords.keyword

Order By: metric: Count

Order: Descenc

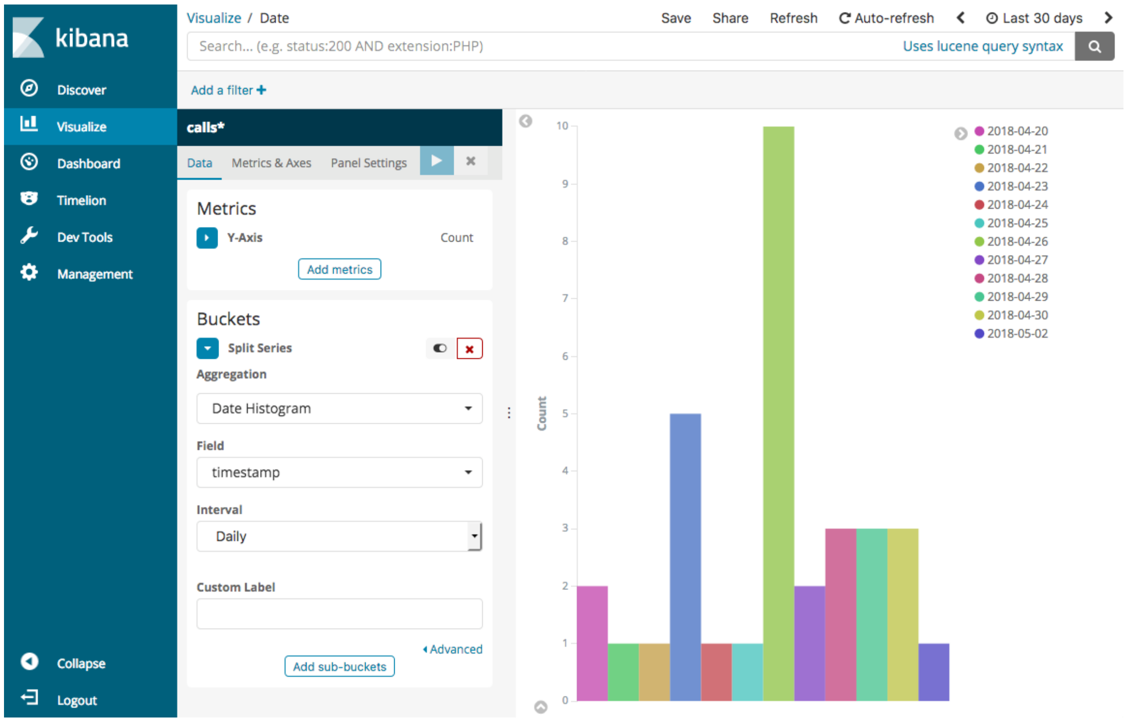
Size: 20

Group other values in separate bucket

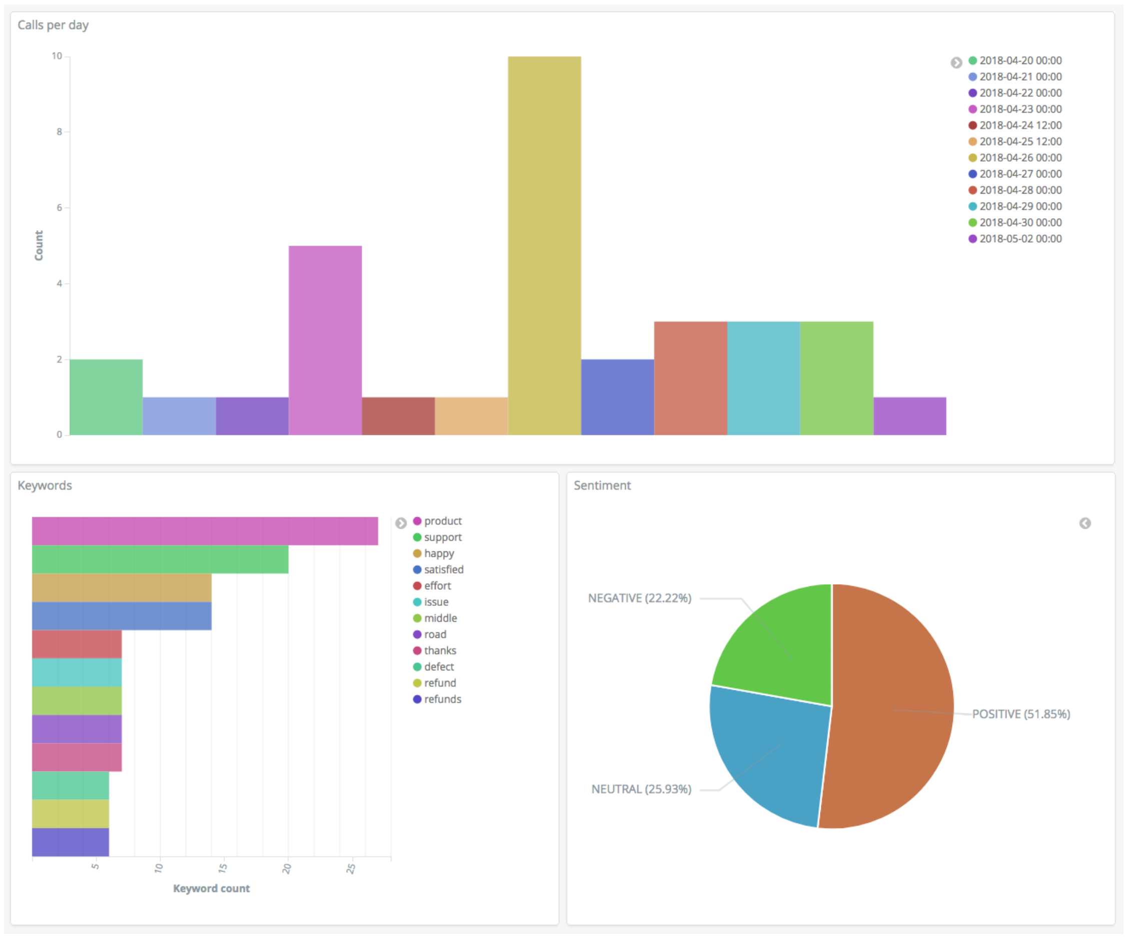
Show missing values

Custom Label

9. Return to the **Visualize** page and add one final visualization, a vertical bar chart.
10. Choose **Split Series**. For **Aggregation**, choose **Date Histogram**. For **Field**, choose **timestamp** and change **Interval** to **Daily**.
11. Choose **Metrics & Axes** and change **Mode** to **normal**.
12. Choose **Apply Changes** and **Save**.



13. Now that you have three visualizations, you can add them to a Kibana dashboard. Choose **Dashboard**, create a dashboard, and add your visualizations.



## Step 5: Clean Up Resources and Next Steps

To avoid unnecessary charges, delete the S3 bucket and Amazon ES domain. To learn more, see Delete a Bucket in the *Amazon Simple Storage Service Developer Guide* and Delete an Amazon ES Domain (p. 8) in this guide.

Transcripts require much less disk space than MP3 files. You might be able to shorten your MP3 retention window—for example, from three months of call recordings to one month—retain years of transcripts, and still save on storage costs.

You could also automate the transcription process using AWS Step Functions and Lambda, add additional metadata before indexing, or craft more complex visualizations to fit your exact use case.

# Amazon Elasticsearch Service Troubleshooting

This section describes how to identify and solve common Amazon Elasticsearch Service issues. Consult the information in this section before contacting AWS Support.

## Can't Access Kibana

The Kibana endpoint doesn't support signed requests. If the access control policy for your domain only grants access to certain IAM users or roles and you haven't configured the section called "Authentication for Kibana" (p. 99), you might receive the following error when you attempt to access Kibana:

```
"User: anonymous is not authorized to perform: es:ESHttpGet"
```

If your Amazon ES domain uses VPC access, you might not receive this error. Instead, the request might time out. To learn more about correcting this issue and the various configuration options available to you, see the section called "Controlling Access to Kibana" (p. 156), the section called "About Access Policies on VPC Domains" (p. 24), and the section called "Identity and Access Management" (p. 64).

## Can't Access VPC Domain

See the section called "About Access Policies on VPC Domains" (p. 24) and the section called "Testing VPC Domains" (p. 24).

## Cluster in Read-Only State

Compared to earlier versions, Elasticsearch 7.*x* uses a different system for cluster coordination. In this new system, when the cluster loses quorum, the cluster is unavailable until you take action. Loss of quorum can take two forms:

- If your cluster uses dedicated master nodes, quorum loss occurs when half or more are unavailable.
- If your cluster does not use dedicated master nodes, quorum loss occurs when half or more of your data nodes are unavailable.

If quorum loss occurs and your cluster has more than one node, Amazon ES restores quorum and places the cluster into a read-only state. You have two options:

- Remove the read-only state and use the cluster as-is.
- Restore the cluster or individual indices from a snapshot (p. 49).

If you prefer to use the cluster as-is, verify that cluster health is green using the following request:

```
GET _cat/health?v
```

If cluster health is red, we recommend restoring the cluster from a snapshot. You can also see the section called "Red Cluster Status" (p. 228) for troubleshooting steps. If cluster health is green, check that all expected indices are present using the following request:

```
GET _cat/indices?v
```

Then run some searches to verify that the expected data is present. If it is, you can remove the read-only state using the following request:

```
PUT _cluster/settings
{
  "persistent": {
    "cluster.blocks.read_only": false
  }
}
```

If quorum loss occurs and your cluster has only one node, Amazon ES replaces the node and does *not* place the cluster into a read-only state. Otherwise, your options are the same: use the cluster as-is or restore from a snapshot.

In both situations, Amazon ES sends two events to your Personal Health Dashboard. The first informs you of the loss of quorum. The second occurs after Amazon ES successfully restores quorum. For more information about using the Personal Health Dashboard, see the AWS Health User Guide.

## Red Cluster Status

A red cluster status means that at least one primary shard and its replicas are not allocated to a node. Amazon ES stops taking automatic snapshots, even of healthy indices, while the red cluster status persists.

The most common causes of a red cluster status are failed cluster nodes (p. 231) and the Elasticsearch process crashing due to a continuous heavy processing load.

**Note**
Amazon ES stores automatic snapshots for 14 days, so if the red cluster status persists for more than two weeks, you can permanently lose your cluster's data. If your Amazon ES domain enters a red cluster status, AWS Support might contact you to ask whether you want to address the problem yourself or you want the support team to assist. You can set a CloudWatch alarm (p. 178) to notify you when a red cluster status occurs.

Ultimately, red shards cause red clusters, and red indices cause red shards. To identity the indices causing the red cluster status, Elasticsearch has some helpful APIs.

- `GET /_cluster/allocation/explain` chooses the first unassigned shard that it finds and explains why it cannot be allocated to a node:

```
{
    "index": "test4",
    "shard": 0,
    "primary": true,
    "current_state": "unassigned",
    "can_allocate": "no",
    "allocate_explanation": "cannot allocate because allocation is not permitted to any
 of the nodes"
}
```

- `GET /_cat/indices?v` shows the health status, number of documents, and disk usage for each index:

```
health status index           uuid                   pri rep docs.count docs.deleted
 store.size pri.store.size
green  open   test1           30h1EiMvS5uAFr2t5CEVoQ   5   0        820            0
   14mb           14mb
green  open   test2           sdIxs_WDT56afFGu5KPbFQ   1   0          0            0
   233b           233b
green  open   test3           GGRZp_TBRZuSaZpAGk2pmw   1   1          2            0
 14.7kb          7.3kb
red    open   test4           BJxfAErbTtu5HBjIXJV_7A   1   0
green  open   test5           _8C6MIXOSxCqVYicH3jsEA   1   0          7            0
 24.3kb         24.3kb
```

Deleting red indices is the fastest way to fix a red cluster status. Depending on the reason for the red cluster status, you might then scale your Amazon ES domain to use larger instance types, more instances, or more EBS-based storage and try to recreate the problematic indices.

If deleting a problematic index isn't feasible, you can restore a snapshot (p. 49), delete documents from the index, change the index settings, reduce the number of replicas, or delete other indices to free up disk space. The important step is to resolve the red cluster status *before* reconfiguring your Amazon ES domain. Reconfiguring a domain with a red cluster status can compound the problem and lead to the domain being stuck in a configuration state of **Processing** until you resolve the status.

## Recovering from a Continuous Heavy Processing Load

To determine if a red cluster status is due to a continuous heavy processing load on a data node, monitor the following cluster metrics.

| Relevant Metric | Description | Recovery |
|---|---|---|
| **JVMMemoryPressure** | Specifies the percentage of the Java heap used for all data nodes in a cluster. View the **Maximum** statistic for this metric, and look for smaller and smaller drops in memory pressure as the Java garbage collector fails to reclaim sufficient memory. This pattern likely is due to complex queries or large data fields.<br><br>The Concurrent Mark Sweep (CMS) garbage collector triggers when 75% of the “old generation” object space is full. This collector runs alongside other threads to keep pauses to a minimum. If CMS is unable to reclaim enough memory during these normal collections, Elasticsearch triggers a different garbage collection algorithm that halts all threads. Nodes are unresponsive during these stop-the- | Set memory circuit breakers for the JVM. For more information, see the section called “JVM OutOfMemoryError” (p. 231).<br><br>If the problem persists, delete unnecessary indices, reduce the number or complexity of requests to the domain, add instances, or use larger instance types. |

| Relevant Metric | Description | Recovery |
|---|---|---|
| | world collections, which can affect cluster stability.<br><br>If memory usage continues to grow, Elasticsearch eventually crashes due to an out of memory error. A good rule of thumb is to keep usage below 80%.<br><br>The `_nodes/stats/jvm` API offers a useful summary of JVM statistics, memory pool usage, and garbage collection information:<br><br>`GET elasticsearch_domain/_nodes/stats/jvm?pretty` | |
| **CPUUtilization** | Specifies the percentage of CPU resources used for data nodes in a cluster. View the **Maximum** statistic for this metric, and look for a continuous pattern of high usage. | Add data nodes or increase the size of the instance types of existing data nodes. |
| **Nodes** | Specifies the number of nodes in a cluster. View the **Minimum** statistic for this metric. This value fluctuates when the service deploys a new fleet of instances for a cluster. | Add data nodes. |

## Yellow Cluster Status

A yellow cluster status means that the primary shards for all indices are allocated to nodes in a cluster, but the replica shards for at least one index are not. Single-node clusters always initialize with a yellow cluster status because there is no other node to which Amazon ES can assign a replica. To achieve green cluster status, increase your node count. For more information, see the section called "Sizing Amazon ES Domains" (p. 171).

## ClusterBlockException

You might receive a `ClusterBlockException` error for the following reasons.

### Lack of Available Storage Space

If no nodes have enough storage space to accommodate shard relocation, basic write operations like adding documents and creating indices can begin to fail. the section called "Calculating Storage Requirements" (p. 172) provides a summary of how Amazon ES uses disk space.

To avoid issues, monitor the `FreeStorageSpace` metric in the Amazon ES console and create CloudWatch alarms (p. 178) to trigger when `FreeStorageSpace` drops below a certain threshold. `GET /_cat/allocation?v` also provides a useful summary of shard allocation and disk usage. To resolve issues associated with a lack of storage space, scale your Amazon ES domain to use larger instance types, more instances, or more EBS-based storage.

### Block Disks Due to Low Memory

When the **JVMMemoryPressure** metric exceeds 92% for 30 minutes, Amazon ES triggers a protection mechanism and blocks all write operations to prevent the cluster from reaching red status. When the protection is on, write operations fail with a `ClusterBlockException` error, new indices can't be created, and the `IndexCreateBlockException` error is thrown.

When the **JVMMemoryPressure** metric returns to 88% or lower for five minutes, the protection is disabled, and write operations to the cluster are unblocked.

## JVM OutOfMemoryError

A JVM `OutOfMemoryError` typically means that one of the following JVM circuit breakers was reached.

| Circuit Breaker | Description | Cluster Setting Property |
|---|---|---|
| Parent Breaker | Total percentage of JVM heap memory allowed for all circuit breakers. The default value is 70%. | `indices.breaker.total.limit` |
| Field Data Breaker | Percentage of JVM heap memory allowed to load a single data field into memory. The default value is 60%. If you upload data with large fields, we recommend raising this limit. | `indices.breaker.fielddata.limit` |
| Request Breaker | Percentage of JVM heap memory allowed for data structures used to respond to a service request. The default value is 40%. If your service requests involve calculating aggregations, we recommend raising this limit. | `indices.breaker.request.limit` |

## Failed Cluster Nodes

Amazon EC2 instances might experience unexpected terminations and restarts. Typically, Amazon ES restarts the nodes for you. However, it's possible for one or more nodes in an Elasticsearch cluster to remain in a failed condition.

To check for this condition, open your domain dashboard on the Amazon ES console. Choose the **Cluster health** tab, and then choose the **Nodes** metric. See if the reported number of nodes is fewer than the number that you configured for your cluster. If the metric shows that one or more nodes is down for more than one day, contact AWS Support.

You can also set a CloudWatch alarm (p. 178) to notify you when this issue occurs.

**Note**
The **Nodes** metric is not accurate during changes to your cluster configuration and during routine maintenance for the service. This behavior is expected. The metric will report the

correct number of cluster nodes soon. To learn more, see the section called "Configuration Changes" (p. 14).

To protect your clusters from unexpected node terminations and restarts, create at least one replica for each index in your Amazon ES domain.

## Maximum Shard Limit

The 7.*x* versions of Elasticsearch have a default setting of no more than 1,000 shards per node. Elasticsearch throws an error if a request, such as creating a new index, would cause you to exceed this limit. If you encounter this error, you have several options:

- Add more data nodes to the cluster.
- Increase the `_cluster/settings/cluster.max_shards_per_node` setting.
- Use the _shrink API (p. 185) to reduce the number of shards on the node.

## Can't Close Index

Amazon ES doesn't support the `_close` API. If you are restoring an index from a snapshot, you can delete the existing index (before or after reindexing it). The other option is to use the `rename_pattern` and `rename_replacement` fields to rename the index as you restore it:

```
POST /_snapshot/my-repository/my-snapshot/_restore
{
  "indices": "my-index-1,myindex-2",
  "include_global_state": true,
  "rename_pattern": "my-index-(\\d)",
  "rename_replacement": "restored-my-index-$1"
}
```

If you plan to reindex, shrink, or split an index, you likely want to stop writing to it before performing the operation.

## Can't SSH into Node

You can't use SSH to access any of the nodes in your Elasticsearch cluster, and you can't directly modify `elasticsearch.yml`. Instead, use the console, AWS CLI, or SDKs to configure your domain. You can specify a few cluster-level settings using the Elasticsearch REST APIs, as well. To learn more, see *Amazon ES Configuration API Reference* (p. 236) and the section called "Supported Elasticsearch Operations" (p. 183).

If you need more insight into the performance of the cluster, you can publish error logs and slow logs to CloudWatch (p. 40).

## "Not Valid for the Object's Storage Class" Snapshot Error

Amazon ES snapshots do not support the S3 Glacier storage class. You might encounter this error when you attempt to list snapshots if your S3 bucket includes a lifecycle rule that transitions objects to the S3 Glacier storage class.

If you need to restore a snapshot from the bucket, restore the objects from S3 Glacier, copy the objects to a new bucket, and register the new bucket (p. 46) as a snapshot repository.

## Invalid Host Header

Amazon ES requires that clients specify `Host` in the request headers. A valid `Host` value is the domain endpoint without `https://`, such as:

```
Host: search-my-sample-domain-ih2lhn2ew2scurji.us-west-2.es.amazonaws.com
```

If you receive an `Invalid Host Header` error when making a request, check that your client includes the Amazon ES domain endpoint (and not, for example, its IP address) in the `Host` header.

## Can't Downgrade After Upgrade

In-place upgrades (p. 51) are irreversible, but if you contact AWS Support, they can help you restore the automatic, pre-upgrade snapshot on a new domain. For example, if you upgrade a domain from Elasticsearch 5.6 to 6.4, AWS Support can help you restore the pre-upgrade snapshot on a new Elasticsearch 5.6 domain. If you took a manual snapshot of the original domain, you can perform that step yourself (p. 44).

## Need Summary of Domains for All Regions

The following script uses the Amazon EC2 describe-regions AWS CLI command to create a list of all regions in which Amazon ES could be available. Then it calls list-domain-names for each region:

```
for region in `aws ec2 describe-regions --output text | cut -f4`
do
    echo "\nListing domains in region '$region':"
    aws es list-domain-names --region $region --query 'DomainNames'
done
```

You receive the following output for each region:

```
Listing domains in region:'us-west-2'...
[
  {
    "DomainName": "sample-domain"
  }
]
```

Regions in which Amazon ES is not available return "Could not connect to the endpoint URL."

## Browser Error When Using Kibana

Your browser wraps service error messages in HTTP response objects when you use Kibana to view data in your Amazon ES domain. You can use developer tools commonly available in web browsers, such as Developer Mode in Chrome, to view the underlying service errors and assist your debugging efforts.

**To view service errors in Chrome**

1. From the menu, choose **View**, **Developer**, **Developer Tools**.

2. Choose the **Network** tab.
3. In the **Status** column, choose any HTTP session with a status of 500.

**To view service errors in Firefox**

1. From the menu, choose **Tools**, **Web Developer**, **Network**.
2. Choose any HTTP session with a status of 500.
3. Choose the **Response** tab to view the service response.

## Unauthorized Operation After Selecting VPC Access

When you create a new domain using the Amazon ES console, you have the option to select VPC or public access. If you select **VPC access**, Amazon ES queries for VPC information and fails if you don't have the proper permissions:

```
You are not authorized to perform this operation. (Service: AmazonEC2; Status Code: 403;
 Error Code: UnauthorizedOperation
```

To enable this query, you must have access to the `ec2:DescribeVpcs`, `ec2:DescribeSubnets`, and `ec2:DescribeSecurityGroups` operations. This requirement is only for the console. If you use the AWS CLI to create and configure a domain with a VPC endpoint, you don't need access to those operations.

## Stuck at Loading After Creating VPC Domain

After creating a new domain that uses VPC access, the domain's **Configuration state** might never progress beyond **Loading**. If this issue occurs, you likely have AWS Security Token Service (AWS STS) *disabled* for your region.

To add VPC endpoints to your VPC, Amazon ES needs to assume the `AWSServiceRoleForAmazonElasticsearchService` role. Thus, AWS STS must be enabled to create new domains that use VPC access in a given region. To learn more about enabling and disabling AWS STS, see the IAM User Guide.

## Can't Connect from Alpine Linux

Alpine Linux limits DNS response size to 512 bytes. If you try to connect to your Amazon ES domain from Alpine Linux, DNS resolution can fail if the domain is in a VPC and has more than 20 nodes. If your domain is in a VPC, we recommend using other Linux distributions, such as Debian, Ubuntu, CentOS, Red Hat Enterprise Linux, or Amazon Linux 2, to connect to it.

## Certificate Error When Using SDK

Because AWS SDKs use the CA certificates from your computer, changes to the certificates on the AWS servers can cause connection failures when you attempt to use an SDK. Error messages vary, but typically contain the following text:

```
Failed to query Elasticsearch
...
SSL3_GET_SERVER_CERTIFICATE:certificate verify failed
```

You can prevent these failures by keeping your computer's CA certificates and operating system up-to-date. If you encounter this issue in a corporate environment and do not manage your own computer, you might need to ask an administrator to assist with the update process.

The following list shows minimum operating system and Java versions:

- Microsoft Windows versions that have updates from January 2005 or later installed contain at least one of the required CAs in their trust list.
- Mac OS X 10.4 with Java for Mac OS X 10.4 Release 5 (February 2007), Mac OS X 10.5 (October 2007), and later versions contain at least one of the required CAs in their trust list.
- Red Hat Enterprise Linux 5 (March 2007), 6, and 7 and CentOS 5, 6, and 7 all contain at least one of the required CAs in their default trusted CA list.
- Java 1.4.2_12 (May 2006), 5 Update 2 (March 2005), and all later versions, including Java 6 (December 2006), 7, and 8, contain at least one of the required CAs in their default trusted CA list.

The three certificate authorities are:

- Amazon Root CA 1
- Starfield Services Root Certificate Authority - G2
- Starfield Class 2 Certification Authority

Root certificates from the first two authorities are available from Amazon Trust Services, but keeping your computer up-to-date is the more straightforward solution. To learn more about ACM-provided certificates, see AWS Certificate Manager FAQs.

**Note**
Currently, Amazon ES domains in the us-east-1 region use certificates from a different authority. We plan to update the region to use these new certificate authorities in the near future.

# Amazon Elasticsearch Service Configuration API Reference

This reference describes the actions, data types, and errors in the Amazon Elasticsearch Service Configuration API. The configuration API is a REST API that you can use to create and configure Amazon ES domains over HTTP. You also can use the AWS CLI and the console to configure Amazon ES domains. For more information, see Creating and Configuring Amazon ES Domains (p. 9).

- Actions (p. 236)
- Data Types (p. 264)
- Errors (p. 279)

## Actions

The following table provides a quick reference to the HTTP method required for each operation for the REST interface to the Amazon Elasticsearch Service configuration API. The description of each operation also includes the required HTTP method.

**Note**
All configuration service requests must be signed. For more information, see Signing Amazon Elasticsearch Service Requests (p. 67) in this guide and Signature Version 4 Signing Process in the *AWS General Reference*.

| Action | HTTP Method |
|---|---|
| AddTags (p. 237) | `POST` |
| the section called “AssociatePackage” (p. 238) | `POST` |
| CreateElasticsearchDomain (p. 239) | `POST` |
| the section called “CreatePackage” (p. 242) | `POST` |
| DeleteElasticsearchDomain (p. 243) | `DELETE` |
| DeleteElasticsearchServiceRole (p. 243) | `DELETE` |
| the section called “DeletePackage” (p. 244) | `DELETE` |
| DescribeElasticsearchDomain (p. 244) | `GET` |
| DescribeElasticsearchDomainConfig (p. 245) | `GET` |
| DescribeElasticsearchDomains (p. 245) | `POST` |
| DescribeElasticsearchInstanceTypeLimits (p. 246) | `GET` |
| the section called “DescribePackages” (p. 247) | `POST` |
| DescribeReservedElasticsearchInstanceOfferings (p. 248) | `GET` |

| Action | HTTP Method |
|---|---|
| DescribeReservedElasticsearchInstances (p. 249) | GET |
| the section called “DissociatePackage” (p. 250) | POST |
| GetCompatibleElasticsearchVersions (p. 250) | GET |
| GetUpgradeHistory (p. 251) | GET |
| GetUpgradeStatus (p. 252) | GET |
| ListDomainNames (p. 253) | GET |
| the section called “ListDomainsForPackage” (p. 253) | GET |
| ListElasticsearchInstanceTypeDetails (p. 254) | GET |
| ListElasticsearchInstanceTypes (p. 255) | GET |
| ListElasticsearchVersions (p. 256) | GET |
| the section called “ListPackagesForDomain” (p. 257) | GET |
| ListTags (p. 258) | GET |
| `PurchaseReservedElasticsearchInstanceOffering` (p. 258) | POST |
| RemoveTags (p. 259) | POST |
| StartElasticsearchServiceSoftwareUpdate (p. 260) | POST |
| StopElasticsearchServiceSoftwareUpdate (p. 260) | POST |
| UpdateElasticsearchDomainConfig (p. 261) | POST |
| UpgradeElasticsearchDomain (p. 263) | POST |

## AddTags

Attaches resource tags to an Amazon ES domain. For more information, see Tagging Amazon ES Domains (p. 56).

### Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/tags
{
  "ARN": "domain-arn",
  "TagList": [{
    "Key": "tag-key",
    "Value": "tag-value"
  }]
}
```

### Request Parameters

This operation does not use request parameters.

## Request Body

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `TagList` | the section called "TagList" (p. 278) | Yes | List of resource tags. |
| `ARN` | the section called "ARN" (p. 265) | Yes | Amazon Resource Name (ARN) for the Amazon ES domain to which you want to attach resource tags. |

## Response Elements

The `AddTags` operation does not return a data structure.

# AssociatePackage

Associates a package with an Amazon ES domain.

## Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/packages/associate/package-id/domain-
name
```

## Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `PackageID` | String | Yes | Internal ID of the package that you want to associate with a domain. Use the section called "DescribePackages" (p. 247) to find this value. |
| `DomainName` | the section called "DomainName" (p. 267) | Yes | Name of the domain that you want to associate the package with. |

## Request Body

This operation does not use the HTTP request body.

## Response Elements

| Field | Data Type |
|---|---|
| `DomainPackageDetails` | the section called "DomainPackageDetails" (p. 268) |

## CreateElasticsearchDomain

Creates an Amazon ES domain. For more information, see the section called " Creating Amazon ES Domains" (p. 9).

**Note**
If you attempt to create an Amazon ES domain and a domain with the same name already exists, the API does not report an error. Instead, it returns details for the existing domain.

### Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/es/domain
{
  "ElasticsearchClusterConfig": {
    "ZoneAwarenessConfig": {
      "AvailabilityZoneCount": 3
    },
    "ZoneAwarenessEnabled": true|false,
    "InstanceCount": 3,
    "DedicatedMasterEnabled": true|false,
    "DedicatedMasterType": "c5.large.elasticsearch",
    "DedicatedMasterCount": 3,
    "InstanceType": "r5.large.elasticsearch",
    "WarmCount": 3,
    "WarmEnabled": true|false,
    "WarmType": "ultrawarm1.large.elasticsearch"
  },
  "EBSOptions": {
    "EBSEnabled": true|false,
    "VolumeType": "io1|gp2|standard",
    "Iops": 1000,
    "VolumeSize": 35
  },
  "EncryptionAtRestOptions": {
    "Enabled": true|false,
    "KmsKeyId":"arn:aws:kms:us-east-1:123456789012:alias/my-key"
  },
  "SnapshotOptions": {
    "AutomatedSnapshotStartHour": 3
  },
  "VPCOptions": {
    "VPCId": "vpc-12345678",
    "SubnetIds": ["subnet-abcdefg1", "subnet-abcdefg2", "subnet-abcdefg3"],
    "SecurityGroupIds": ["sg-12345678"]
  },
  "AdvancedOptions": {
    "rest.action.multi.allow_explicit_index": "true|false",
    "indices.fielddata.cache.size": "40",
    "indices.query.bool.max_clause_count": "1024"
  },
  "CognitoOptions": {
    "Enabled": true|false,
    "UserPoolId": "us-east-1_121234567",
    "IdentityPoolId": "us-east-1:12345678-1234-1234-1234-123456789012",
    "RoleArn": "arn:aws:iam::123456789012:role/service-role/CognitoAccessForAmazonES"
  },
  "NodeToNodeEncryptionOptions": {
    "Enabled": true|false
  },
  "DomainEndpointOptions": {
    "EnforceHTTPS": true|false,
    "TLSSecurityPolicy": "Policy-Min-TLS-1-2-2019-07|Policy-Min-TLS-1-0-2019-07"
  },
  "LogPublishingOptions": {
    "SEARCH_SLOW_LOGS": {
      "CloudWatchLogsLogGroupArn":"arn:aws:logs:us-east-1:264071961897:log-group1:sample-
domain",
      "Enabled":true|false
    },
    "INDEX_SLOW_LOGS": {
      "CloudWatchLogsLogGroupArn":"arn:aws:logs:us-east-1:264071961897:log-group2:sample-
domain",
      "Enabled":true|false
    },
    "ES_APPLICATION_LOGS": {
      "CloudWatchLogsLogGroupArn":"arn:aws:logs:us-east-1:264071961897:log-group3:sample-
domain",
      "Enabled":true|false
    }
  },
  "AdvancedSecurityOptions": {
    "Enabled": true|false,
    "InternalUserDatabaseEnabled": true|false,
    "MasterUserOptions": {
      "MasterUserARN": "arn:aws:iam::123456789012:role/my-master-user-role"
      "MasterUserName": "my-master-username",
      "MasterUserPassword": "my-master-password"
    }
  },
  "ElasticsearchVersion": "7.1",
  "DomainName": "my-domain",
  "AccessPolicies": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"AWS\":[\"123456789012\"]},\"Action\":[\"es:es:ESHttp*\"],\"Resource\":
\"arn:aws:es:us-east-1:123456789012:domain/my-domain/*\"}]}"
}
```

## Request Parameters

This operation does not use HTTP request parameters.

## Request Body

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `DomainName` | the section called “DomainName” (p. 267) | Yes | Name of the Amazon ES domain to create. |
| `ElasticsearchVersion` | String | No | Version of Elasticsearch. If not specified, 1.5 is used as the default. For the full list of supported versions, see the section called “Supported Elasticsearch Versions” (p. 2). |
| `ElasticsearchClusterConfig` | the section called “ElasticsearchClusterConfig” (p. 269) | No | Container for the cluster configuration of an Amazon ES domain. |
| `EBSOptions` | the section called “EBSOptions” (p. 268) | No | Container for the parameters required to enable EBS-based storage for an Amazon ES domain. |

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `VPCOptions` | the section called "VPCOptions" (p. 278) | No | Container for the values required to configure VPC access domains. If you don't specify these values, Amazon ES creates the domain with a public endpoint. To learn more, see VPC Support for Amazon Elasticsearch Service Domains (p. 21). |
| `CognitoOptions` | the section called "CognitoOptions" (p. 266) | No | Key-value pairs to configure Amazon ES to use Amazon Cognito authentication for Kibana. |
| `AccessPolicies` | String | No | IAM policy document specifying the access policies for the new Amazon ES domain. For more information, see the section called "Identity and Access Management" (p. 64). |
| `SnapshotOptions` | the section called "SnapshotOptions" (p. 277) | No | **DEPRECATED**. For domains running Elasticsearch 5.3 and later, Amazon ES takes hourly automated snapshots, making this setting irrelevant.<br><br>For domains running earlier versions of Elasticsearch, Amazon ES takes daily automated snapshots. This value acts as a container for the hour of the day at which you want the service to take the snapshot. |
| `AdvancedOptions` | the section called "AdvancedOptions" (p. 264) | No | Key-value pairs to specify advanced configuration options. For more information, see Configuring Advanced Options (p. 13). |
| `LogPublishingOptions` | the section called "LogPublishingOptions" (p. 273) | No | Key-value pairs to configure slow log publishing. |
| `EncryptionAtRestOptions` | the section called "EncryptionAtRestOptions" (p. 273) | No | Key-value pairs to enable encryption at rest. |
| `NodeToNodeEncryptionOptions` | the section called "NodeToNodeEncryptionOptions" (p. 275) | No | Enables node-to-node encryption. |
| `DomainEndpointOptions` | the section called "DomainEndpointOptions" (p. 267) | No | Additional options for the domain endpoint, such as whether to require HTTPS for all traffic. |
| `AdvancedSecurityOptions` | the section called "AdvancedSecurityOptions" (p. 266) | No | Options for fine-grained access control. |

### Response Elements

| Field | Data Type |
|---|---|
| `DomainStatus` | the section called "ElasticsearchDomainStatus" (p. 270) |

## CreatePackage

Add a package for use with Amazon ES domains.

### Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/packages
{
  "PackageName": "my-package-name",
  "PackageType": "TXT-DICTIONARY",
  "PackageDescription": "My synonym file.",
  "PackageSource": {
    "S3BucketName": "my-s3-bucket",
    "S3Key": "synonyms.txt"
  }
}
```

### Request Parameters

This operation does not use request parameters.

### Request Body

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `PackageName` | String | Yes | Unique name for the package. |
| `PackageType` | String | Yes | Type of package. Currently supports only `TXT-DICTIONARY`. |
| `PackageDescription` | String | No | Description of the package. |
| `PackageSource` | the section called "PackageSource" (p. 276) | Yes | S3 bucket and key for the package. |

### Response Elements

| Field | Data Type |
|---|---|
| `PackageDetails` | the section called "PackageDetails" (p. 275) |

## DeleteElasticsearchDomain

Deletes an Amazon ES domain and all of its data. A domain cannot be recovered after it is deleted.

### Syntax

```
DELETE https://es.us-east-1.amazonaws.com/2015-01-01/es/domain/domain-name
```

### Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `DomainName` | the section called "DomainName" (p. 267) | Yes | Name of the Amazon ES domain that you want to delete. |

### Request Body

This operation does not use the HTTP request body.

### Response Elements

| Field | Data Type |
|---|---|
| `DomainStatus` | the section called "ElasticsearchDomainStatus" (p. 270) |

## DeleteElasticsearchServiceRole

Deletes the service-linked role between Amazon ES and Amazon EC2. This role gives Amazon ES permissions to place VPC endpoints into your VPC. A service-linked role must be in place for domains with VPC endpoints to be created or function properly.

**Note**
This action succeeds only if no domains are using the service-linked role.

### Syntax

```
DELETE https://es.us-east-1.amazonaws.com/2015-01-01/es/role
```

### Request Parameters

This operation does not use request parameters.

### Request Body

This operation does not use the HTTP request body.

### Response Elements

The `DeleteElasticsearchServiceRole` operation does not return a data structure.

## DeletePackage

Deletes a package from Amazon ES. The package must not be associated with any Amazon ES domain.

### Syntax

```
DELETE https://es.us-east-1.amazonaws.com/2015-01-01/packages/package-id
```

### Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `PackageID` | String | Yes | Internal ID of the package that you want to delete. Use the section called "DescribePackages" (p. 247) to find this value. |

### Request Body

This operation does not use the HTTP request body.

### Response Elements

| Field | Data Type |
|---|---|
| `PackageDetails` | the section called "PackageDetails" (p. 275) |

## DescribeElasticsearchDomain

Describes the domain configuration for the specified Amazon ES domain, including the domain ID, domain service endpoint, and domain ARN.

### Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/es/domain/domain-name
```

### Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `DomainName` | the section called "DomainName" (p. 267) | Yes | Name of the Amazon ES domain that you want to describe. |

### Request Body

This operation does not use the HTTP request body.

### Response Elements

| Field | Data Type |
|---|---|
| `DomainStatus` | the section called "ElasticsearchDomainStatus" (p. 270) |

## DescribeElasticsearchDomainConfig

Displays the configuration of an Amazon ES domain.

### Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/es/domain/domain-name/config
```

### Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `DomainName` | the section called "DomainName" (p. 267) | Yes | Name of the Amazon ES domain configuration that you want to describe. |

### Request Body

This operation does not use the HTTP request body.

### Response Elements

| Field | Data Type |
|---|---|
| `DomainConfig` | the section called "ElasticsearchDomainConfig" (p. 269) |

## DescribeElasticsearchDomains

Describes the domain configuration for up to five specified Amazon ES domains. Information includes the domain ID, domain service endpoint, and domain ARN.

### Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/es/domain-info
{
  "DomainNames": [
    "domain-name1",
    "domain-name2",
  ]
}
```

### Request Parameters

This operation does not use HTTP request parameters.

### Request Body

| Field | Data Type | Required | Description |
|---|---|---|---|
| `DomainNames` | the section called "DomainNameList" (p. 268) | Yes | Array of Amazon ES domain names. |

### Response Elements

| Field | Data Type |
|---|---|
| `DomainStatusList` | the section called "ElasticsearchDomainStatusList" (p. 272) |

## DescribeElasticsearchInstanceTypeLimits

Describes the instance count, storage, and master node limits for a given Elasticsearch version and instance type.

### Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/es/instanceTypeLimits/elasticsearch-
version/instance-type?domainName=domain-name
```

### Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ElasticsearchVersion` | String | Yes | Elasticsearch version. For a list of supported versions, see the section called "Supported Elasticsearch Versions" (p. 2). |
| `InstanceType` | String | Yes | Instance type. To view instance types by Region, see Amazon Elasticsearch Service Pricing. |

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `DomainName` | the section called "DomainName" (p. 267) | No | The name of an existing domain. Only specify if you need the limits for an existing domain. |

### Request Body

This operation does not use the HTTP request body.

### Response Elements

| Field | Data Type | Description |
|---|---|---|
| `LimitsByRole` | Map | Map that contains all applicable instance type limits. `"data"` refers to data nodes. `"master"` refers to dedicated master nodes. |

## DescribePackages

Describes all packages available to Amazon ES. Includes options for filtering, limiting the number of results, and pagination.

### Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/packages/describe
{
  "Filters": [{
    "Name": "PackageStatus",
    "Value": [
      "DELETING", "AVAILABLE"
    ]
  }],
  "MaxResults": 5,
  "NextToken": "next-token",
}
```

### Request Parameters

This operation does not use request parameters.

### Request Body

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `Filters` | the section called "Filters" (p. 273) | No | Only returns packages that match the provided values. |
| `MaxResults` | Integer | No | Limits results to a maximum number of packages. |

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `NextToken` | String | No | Used for pagination. Only necessary if a previous API call includes a non-null `NextToken` value. If provided, returns results for the next page. |

## Response Elements

| Field | Data Type | Description |
|---|---|---|
| `PackageDetailsList` | List | List of the section called "PackageDetails" (p. 275) objects. |

## DescribeReservedElasticsearchInstanceOfferings

Describes the available Reserved Instance offerings for a given Region.

### Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/es/reservedInstanceOfferings?
offeringId=offering-id&maxResults=max-results&nextToken=next-token
```

### Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `OfferingId` | String | No | The offering ID. |
| `MaxResults` | Integer | No | Limits the number of results. Must be between 30 and 100. |
| `NextToken` | String | No | Used for pagination. Only necessary if a previous API call produced a result that contains `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token output in the response, which clients can use to retrieve more results. |

### Request Body

This operation does not use the HTTP request body.

## Response Elements

| Field | Data Type | Description |
|---|---|---|
| `ReservedElasticsearchInstanceOfferings` | ReservedElasticsearchInstanceOfferings | Container for all information about a Reserved Instance offering. For more information, see the section called "Purchasing Reserved Instances (AWS CLI)" (p. 203). |

## DescribeReservedElasticsearchInstances

Describes the instance that you have reserved in a given Region.

## Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/es/reservedInstances?
reservationId=reservation-id&maxResults=max-results&nextToken=next-token
```

## Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ReservationId` | String | No | The reservation ID, assigned after you purchase a reservation. |
| `MaxResults` | Integer | No | Limits the number of results. Must be between 30 and 100. |
| `NextToken` | String | No | Used for pagination. Only necessary if a previous API call produced a result that contains `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token output in the response, which clients can use to retrieve more results. |

## Request Body

This operation does not use the HTTP request body.

## Response Elements

| Field | Data Type | Description |
|---|---|---|
| `ReservedElasticsearchInstances` | ReservedElasticsearchInstances | Container for all information about the instance that you have reserved. For |

| Field | Data Type | Description |
|---|---|---|
| | | more information, see the section called "Purchasing Reserved Instances (AWS CLI)" (p. 203). |

## DissociatePackage

Removes the package from the specified Amazon ES domain. The package must not be in use with any ES index for dissociate to succeed. The package will still be available in the Amazon ES service for associating later.

### Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/packages/dissociate/package-id/domain-
name
```

### Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `PackageID` | String | Yes | Internal ID of the package that you want to dissociate from the domain. Use the section called "ListPackagesForDomain" (p. 257) to find this value. |
| `DomainName` | the section called "DomainName" (p. 267) | Yes | Name of the domain that you want to dissociate the package from. |

### Request Body

This operation does not use the HTTP request body.

### Response Elements

| Field | Data Type |
|---|---|
| `DomainPackageDetails` | the section called "DomainPackageDetails" (p. 268) |

## GetCompatibleElasticsearchVersions

Returns a map of Elasticsearch versions and the versions you can upgrade them to.

### Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/es/compatibleVersions?domainName=domain-
name
```

## Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `DomainName` | the section called "DomainName" (p. 267) | No | The name of an existing domain. |

## Request Body

This operation does not use the HTTP request body.

## Response Elements

| Field | Data Type | Description |
|---|---|---|
| `CompatibleElasticsearchVersions` | Map | A map of Elasticsearch versions and the versions that you can upgrade them to: <br> `{` <br> `"CompatibleElasticsearchVersions": [{` <br> `"SourceVersion": "6.7",` <br> `"TargetVersions": ["6.8"]` <br> `}]` <br> `}` |

# GetUpgradeHistory

Returns a list of the domain's 10 most-recent upgrade operations.

## Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/es/upgradeDomain/domain-name/history?
maxResults=max-results&amp;nextToken=next-token
```

## Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `MaxResults` | Integer | No | Limits the number of results. Must be between 30 and 100. |
| `DomainName` | the section called "DomainName" (p. 267) | Yes | The name of an existing domain. |
| `NextToken` | String | No | Used for pagination. Only necessary if a previous API call |

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| | | | produced a result containing `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token output in the response, which clients can use to retrieve more results. |

## Request Body

This operation does not use the HTTP request body.

## Response Elements

| Field | Data Type | Description |
|---|---|---|
| `UpgradeHistoryList` | UpgradeHistoryList | Container for result logs of the past 10 upgrade operations. |

# GetUpgradeStatus

Returns the most-recent status of a domain's Elasticsearch version upgrade.

## Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/es/upgradeDomain/domain-name/status
```

## Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `DomainName` | the section called “DomainName” (p. 267) | Yes | The name of an existing domain. |

## Request Body

This operation does not use the HTTP request body.

## Response Elements

| Field | Data Type | Description |
|---|---|---|
| `UpgradeStepItem` | UpgradeStepItem | Container for the most-recent status of a domain's version upgrade. |

## ListDomainNames

Displays the names of all Amazon ES domains owned by the current user *in the active Region*.

### Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/domain
```

### Request Parameters

This operation does not use request parameters.

### Request Body

This operation does not use the HTTP request body.

### Response Elements

| Field | Data Type | Description |
|---|---|---|
| `DomainNameList` | DomainNameList (p. 268) | The names of all Amazon ES domains owned by the current user. |

## ListDomainsForPackage

Lists all Amazon ES domains that a package is associated with.

### Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/packages/package-id/domains?
maxResults=max-results&amp;nextToken=next-token
```

### Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `PackageID` | String | Yes | The package for which to list domains. |
| `MaxResults` | Integer | No | Limits the number of results. |
| `NextToken` | String | No | Used for pagination. Only necessary if a previous API call produced a result that contains `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token output in the response, which |

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| | | | clients can use to retrieve more results. |

### Request Body

This operation does not use the HTTP request body.

### Response Elements

| Field | Data Type | Description |
|---|---|---|
| `DomainPackageDetailsList` | List | List of the section called "DomainPackageDetails" (p. 268) objects. |
| `NextToken` | String | Used for pagination. Only necessary if a previous API call produced a result containing `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token output in the response, which clients can use to retrieve more results. |

## ListElasticsearchInstanceTypeDetails

Lists all Elasticsearch instance types that are supported for a given Elasticsearch version and the features that these instance types support.

### Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/es/instanceTypeDetails/elasticsearch-
version?domainName=domain-name&maxResults=max-results&nextToken=next-token
```

### Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ElasticsearchVersion` | String | Yes | The Elasticsearch version. |
| `DomainName` | String | No | The Amazon ES domain name. |
| `MaxResults` | Integer | No | Limits the number of results. Must be between 30 and 100. |
| `NextToken` | String | No | Used for pagination. Only necessary if a previous API call produced a result containing `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token output in the response, which |

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| | | | clients can use to retrieve more results. |

## Request Body

This operation does not use the HTTP request body.

## Response Elements

| Field | Data Type | Description |
|---|---|---|
| `ElasticsearchInstanceTypes` | List | List of supported instance types for the given Elasticsearch version and the features that these instance types support. |
| `NextToken` | String | Used for pagination. Only necessary if a previous API call produced a result containing `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token output in the response, which clients can use to retrieve more results. |

# ListElasticsearchInstanceTypes (Deprecated)

Lists all Elasticsearch instance types that are supported for a given Elasticsearch version. This action is deprecated. Use ListElasticsearchInstanceTypeDetails (p. 254) instead.

## Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/es/instanceTypes/elasticsearch-version?
domainName=domain-name&maxResults=max-results&nextToken=next-token
```

## Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ElasticsearchVersion` | String | Yes | The Elasticsearch version. |
| `DomainName` | String | No | The Amazon ES domain name. |
| `MaxResults` | Integer | No | Limits the number of results. Must be between 30 and 100. |
| `NextToken` | String | No | Used for pagination. Only necessary if a previous API call produced a result containing `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token |

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| | | | output in the response, which clients can use to retrieve more results. |

## Request Body

This operation does not use the HTTP request body.

## Response Elements

| Field | Data Type | Description |
|---|---|---|
| `ElasticsearchInstanceTypes` | List | List of supported instance types for the given Elasticsearch version. |
| `NextToken` | String | Used for pagination. Only necessary if a previous API call produced a result containing `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token output in the response, which clients can use to retrieve more results. |

## ListElasticsearchVersions

Lists all supported Elasticsearch versions on Amazon ES.

## Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/es/versions?maxResults=max-
results&nextToken=next-token
```

## Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `MaxResults` | Integer | No | Limits the number of results. Must be between 30 and 100. |
| `NextToken` | String | No | Used for pagination. Only necessary if a previous API call produced a result containing `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token output in the response, which clients can use to retrieve more results. |

### Request Body

This operation does not use the HTTP request body.

### Response Elements

| Field | Data Type | Description |
|---|---|---|
| `ElasticsearchVersions` | List | Lists all supported Elasticsearch versions. |
| `NextToken` | String | Used for pagination. Only necessary if a previous API call produced a result containing `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token output in the response, which clients can use to retrieve more results. |

## ListPackagesForDomain

Lists all packages associated with the Amazon ES domain.

### Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/domain/domain-name/packages?
maxResults=max-results&amp;nextToken=next-token
```

### Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `DomainName` | String | Yes | The name of the domain for which you want to list associated packages. |
| `MaxResults` | Integer | No | Limits the number of results. |
| `NextToken` | String | No | Used for pagination. Only necessary if a previous API call produced a result that contains `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token output in the response, which clients can use to retrieve more results. |

### Request Body

This operation does not use the HTTP request body.

### Response Elements

| Field | Data Type | Description |
|---|---|---|
| `DomainPackageDetailsList` | List | List of the section called "DomainPackageDetails" (p. 268) objects. |
| `NextToken` | `String` | Used for pagination. Only necessary if a previous API call produced a result containing `NextToken`. Accepts a next-token input to return results for the next page, and provides a next-token output in the response, which clients can use to retrieve more results. |

## ListTags

Displays all resource tags for an Amazon ES domain.

### Syntax

```
GET https://es.us-east-1.amazonaws.com/2015-01-01/tags?arn=domain-arn
```

### Request Parameters

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ARN` | `ARN` (p. 265) | Yes | Amazon Resource Name (ARN) for the Amazon ES domain. |

### Request Body

This operation does not use the HTTP request body.

### Response Elements

| Field | Data Type | Description |
|---|---|---|
| `TagList` | `TagList` (p. 278) | List of resource tags. For more information, see Tagging Amazon Elasticsearch Service Domains (p. 56). |

## PurchaseReservedElasticsearchInstanceOffering

Purchases a Reserved Instance.

### Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/es/purchaseReservedInstanceOffering
{
  "ReservationName" : "my-reservation",
  "ReservedElasticsearchInstanceOfferingId" : "1a2a3a4a5-1a2a-3a4a-5a6a-1a2a3a4a5a6a",
  "InstanceCount" : 3
}
```

## Request Parameters

This operation does not use HTTP request parameters.

## Request Body

| Name | Data Type | Required? | Description |
|---|---|---|---|
| `ReservationName` | String | Yes | A descriptive name for your reservation. |
| `ReservedElasticsearchInstanceOfferingId` | String | Yes | The offering ID. |
| `InstanceCount` | Integer | Yes | The number of instances that you want to reserve. |

## Response Elements

| Field | Data Type | Description |
|---|---|---|
| `ReservationName` | String | The name of your reservation. |
| `ReservedElasticsearchInstanceId` | String | The reservation ID. |

## RemoveTags

Removes the specified resource tags from an Amazon ES domain.

## Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/tags-removal
{
  "ARN": "arn:aws:es:us-east-1:123456789012:domain/my-domain",
  "TagKeys": [
    "tag-key1",
    "tag-key2"
  ]
}
```

## Request Parameters

This operation does not use HTTP request parameters.

## Request Body

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ARN` | `ARN` (p. 265) | Yes | Amazon Resource Name (ARN) of an Amazon ES domain. For more information, |

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| | | | see Identifiers for IAM Entities in *Using AWS Identity and Access Management*. |
| TagKeys | TagKey (p. 277) | Yes | List of tag keys for resource tags that you want to remove from an Amazon ES domain. |

### Response Elements

The `RemoveTags` operation does not return a response element.

## StartElasticsearchServiceSoftwareUpdate

Schedules a service software update for an Amazon ES domain.

### Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/es/serviceSoftwareUpdate/start
{
  "DomainName": "domain-name"
}
```

### Request Parameters

This operation does not use HTTP request parameters.

### Request Body

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| DomainName | DomainName (p. 267) | Yes | Name of the Amazon ES domain that you want to update to the latest service software. |

### Response Elements

| Field | Data Type | Description |
|---|---|---|
| ServiceSoftwareOptions | ServiceSoftwareOptions | Container for the state of your domain relative to the latest service software. |

## StopElasticsearchServiceSoftwareUpdate

Stops a scheduled service software update for an Amazon ES domain. Only works if the domain's `UpdateStatus` is `PENDING_UPDATE`.

### Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/es/serviceSoftwareUpdate/stop
```

```
{
  "DomainName": "domain-name"
}
```

### Request Parameters

This operation does not use HTTP request parameters.

### Request Body

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `DomainName` | `DomainName` (p. 267) | Yes | Name of the Amazon ES domain that you want to update to the latest service software. |

### Response Elements

| Field | Data Type | Description |
|---|---|---|
| `ServiceSoftwareOptions` | `ServiceSoftwareOptions` (p. 276) | Container for the state of your domain relative to the latest service software. |

## UpdateElasticsearchDomainConfig

Modifies the configuration of an Amazon ES domain, such as the instance type and the number of instances. You need to specify only the values that you want to update.

### Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/es/domain/<DOMAIN_NAME>/config
{
  "ElasticsearchClusterConfig": {
    "ZoneAwarenessConfig": {
      "AvailabilityZoneCount": 3
    },
    "ZoneAwarenessEnabled": true|false,
    "InstanceCount": 3,
    "DedicatedMasterEnabled": true|false,
    "DedicatedMasterType": "c5.large.elasticsearch",
    "DedicatedMasterCount": 3,
    "InstanceType": "r5.large.elasticsearch",
    "WarmCount": 6,
    "WarmType": "ultrawarm1.medium.elasticsearch"
  },
  "EBSOptions": {
    "EBSEnabled": true|false,
    "VolumeType": "io1|gp2|standard",
    "Iops": 1000,
    "VolumeSize": 35
  },
  "SnapshotOptions": {
    "AutomatedSnapshotStartHour": 3
  },
  "VPCOptions": {
    "SubnetIds": ["subnet-abcdefg1", "subnet-abcdefg2", "subnet-abcdefg3"],
    "SecurityGroupIds": ["sg-12345678"]
  },
  "AdvancedOptions": {
    "rest.action.multi.allow_explicit_index": "true|false",
    "indices.fielddata.cache.size": "40",
    "indices.query.bool.max_clause_count": "1024"
  },
  "CognitoOptions": {
    "Enabled": true|false,
    "UserPoolId": "us-east-1_121234567",
    "IdentityPoolId": "us-east-1:12345678-1234-1234-1234-123456789012",
    "RoleArn": "arn:aws:iam::123456789012:role/service-role/CognitoAccessForAmazonES"
  },
  "DomainEndpointOptions": {
    "EnforceHTTPS": true|false,
    "TLSSecurityPolicy": "Policy-Min-TLS-1-2-2019-07|Policy-Min-TLS-1-0-2019-07"
  },
  "LogPublishingOptions": {
    "SEARCH_SLOW_LOGS": {
      "CloudWatchLogsLogGroupArn":"arn:aws:logs:us-east-1:264071961897:log-group1:sample-
domain",
      "Enabled":true|false
    },
    "INDEX_SLOW_LOGS": {
      "CloudWatchLogsLogGroupArn":"arn:aws:logs:us-east-1:264071961897:log-group2:sample-
domain",
      "Enabled":true|false
    },
    "ES_APPLICATION_LOGS": {
      "CloudWatchLogsLogGroupArn":"arn:aws:logs:us-east-1:264071961897:log-group3:sample-
domain",
      "Enabled":true|false
    }
  },
  "AdvancedSecurityOptions": {
    "InternalUserDatabaseEnabled": true|false,
    "MasterUserOptions": {
      "MasterUserARN": "arn:aws:iam::123456789012:role/my-master-user-role"
      "MasterUserName": "my-master-username",
      "MasterUserPassword": "my-master-password"
    }
  },
  "DomainName": "my-domain",
  "AccessPolicies": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow
\",\"Principal\":{\"AWS\":[\"*\"]},\"Action\":[\"es:*\"],\"Resource\":\"arn:aws:es:us-
east-1:123456789012:domain/my-domain/*\"}]}"
}
```

## Request Parameters

This operation does not use HTTP request parameters.

## Request Body

| Parameter | Data Type | Required | Description |
|---|---|---|---|
| `DomainName` | `DomainName` (p. 267) | Yes | Name of the Amazon ES domain for which you want to update the configuration. |

| Parameter | Data Type | Required | Description |
|---|---|---|---|
| `ElasticsearchClusterConfig` | `ElasticsearchClusterConfig` (p. 269) | No | Changes that you want to make to the cluster configuration, such as the instance type and number of EC2 instances. |
| `EBSOptions` | `EBSOptions` (p. 268) | No | Type and size of EBS volumes attached to data nodes. |
| `VPCOptions` | `VPCOptions` (p. 278) | No | Container for the values required to configure Amazon ES to work with a VPC. To learn more, see VPC Support for Amazon Elasticsearch Service Domains (p. 21). |
| `SnapshotOptions` | `SnapshotOptions` (p. 277) | No | **DEPRECATED**. Hour during which the service takes an automated daily snapshot of the indices in the Amazon ES domain. |
| `AdvancedOptions` | `AdvancedOptions` (p. 264) | No | Key-value pairs to specify advanced configuration options. For more information, see Configuring Advanced Options (p. 13). |
| `AccessPolicies` | String | No | Specifies the access policies for the Amazon ES domain. For more information, see Configuring Access Policies (p. 13). |
| `LogPublishingOptions` | `LogPublishingOptions` (p. 275) | No | Key-value string pairs to configure slow log publishing. |
| `CognitoOptions` | `CognitoOptions` (p. 266) | No | Key-value pairs to configure Amazon ES to use Amazon Cognito authentication for Kibana. |
| `DomainEndpointOptions` | the section called "DomainEndpointOptions" (p. 267) | No | Additional options for the domain endpoint, such as whether to require HTTPS for all traffic. |
| `AdvancedSecurityOptions` | the section called "AdvancedSecurityOptions" (p. 266) | No | Options for fine-grained access control. |

## Response Elements

| Field | Data Type |
|---|---|
| `DomainConfig` | the section called "ElasticsearchDomainConfig" (p. 269) |

# UpgradeElasticsearchDomain

Upgrades an Amazon ES domain to a new version of Elasticsearch. Alternately, checks upgrade eligibility.

### Syntax

```
POST https://es.us-east-1.amazonaws.com/2015-01-01/es/upgradeDomain
{
  "DomainName": "domain-name",
  "TargetVersion": "7.4",
  "PerformCheckOnly": true|false
}
```

### Request Parameters

This operation does not use HTTP request parameters.

### Request Body

| Parameter | Data Type | Required | Description |
|---|---|---|---|
| `DomainName` | String | Yes | Name of the Amazon ES domain that you want to upgrade. |
| `TargetVersion` | String | Yes | Elasticsearch version to which you want to upgrade. See the section called "GetCompatibleElasticsearchVersions" (p. 250). |
| `PerformCheckOnly` | Boolean | No | Defaults to `false`. If `true`, Amazon ES checks the eligibility of the domain, but does not perform the upgrade. |

### Response Elements

| Field | Data Type | Description |
|---|---|---|
| `UpgradeElasticsearchDomainResponse` | Map | Basic response confirming operation details. |

## Data Types

This section describes the data types used by the configuration API.

### AdvancedOptions

Key-value pairs to specify advanced Elasticsearch configuration options.

| Field | Data Type | Description |
|---|---|---|
| `rest.action.multi.allow_explicit_index` | Key-value pair: `"rest.action.multi.allow_explicit_index":"true"` | Note the use of a string rather than a boolean. |

| Field | Data Type | Description |
|---|---|---|
| | | Specifies whether explicit references to indices are allowed inside the body of HTTP requests. If you want to configure access policies for domain sub-resources, such as specific indices and domain APIs, you must disable this property. For more information about access policies for sub-resources, see Configuring Access Policies (p. 13). |
| `indices.fielddata.cache.size` | Key-value pair:<br><br>`"indices.fielddata.cache.size":"80"` | Note the use of a string rather than an integer. Specifies the percentage of Java heap space that is allocated to field data. By default, this setting is unbounded. |
| `indices.query.bool.max_clause_count` | Key-value pair:<br><br>`"indices.query.bool.max_clause_count":"1024"` | Note the use of a string rather than an integer. Specifies the maximum number of clauses allowed in a Lucene boolean query. 1,024 is the default. Queries with more than the permitted number of clauses that result in a `TooManyClauses` error. To learn more, see the Lucene documentation. |

## ARN

| Field | Data Type | Description |
|---|---|---|
| `ARN` | String | Amazon Resource Name (ARN) of an Amazon ES domain. For more information, see IAM ARNs in the AWS Identity and Access Management documentation. |

## AdvancedSecurityOptions

| Field | Data Type | Description |
|---|---|---|
| `Enabled` | Boolean | True to enable fine-grained access control (p. 76). |
| `InternalUserDatabaseEnabled` | Boolean | True to enable the internal user database. |
| `MasterUserOptions` | the section called "MasterUserOptions" (p. 274) | Container for information about the master user. |

## CognitoOptions

| Field | Data Type | Description |
|---|---|---|
| `Enabled` | Boolean | Whether to enable or disable Amazon Cognito authentication for Kibana. See the section called "Authentication for Kibana" (p. 99). |
| `UserPoolId` | String | The Amazon Cognito user pool ID that you want Amazon ES to use for Kibana authentication. |
| `IdentityPoolId` | String | The Amazon Cognito identity pool ID that you want Amazon ES to use for Kibana authentication. |
| `RoleArn` | String | The `AmazonESCognitoAccess` role that allows Amazon ES to configure your user pool and identity pool. |

## CreateElasticsearchDomainRequest

Container for the parameters required by the `CreateElasticsearchDomain` service operation.

| Field | Data Type | Description |
|---|---|---|
| `DomainName` | `DomainName` (p. 267) | Name of the Amazon ES domain to create. |
| `ElasticsearchClusterConfig` | `ElasticsearchClusterConfig` (p. 269) | Container for the cluster configuration of an Amazon ES domain. |
| `EBSOptions` | `EBSOptions` (p. 268) | Container for the parameters required to enable EBS-based storage for an Amazon ES domain. |
| `AccessPolicies` | String | IAM policy document that specifies the access policies for the new Amazon ES domain. For more information, see Configuring Access Policies (p. 13). |

| Field | Data Type | Description |
|---|---|---|
| `DomainEndpointOptions` | `DomainEndpointOptions` (p. 267) | Additional options for the domain endpoint, such as whether to require HTTPS for all traffic. |
| `SnapshotOptions` | `SnapshotOptions` (p. 277) | **DEPRECATED**. Container for parameters required to configure automated snapshots of domain indices. |
| `VPCOptions` | `VPCOptions` (p. 278) | Container for the values required to configure Amazon ES to work with a VPC. |
| `LogPublishingOptions` | `LogPublishingOptions` (p. 273) | Key-value string pairs to configure slow log publishing. |
| `AdvancedOptions` | the section called "AdvancedOptions" (p. 264) | Key-value pairs to specify advanced configuration options. |
| `CognitoOptions` | `CognitoOptions` (p. 266) | Key-value pairs to configure Amazon ES to use Amazon Cognito authentication for Kibana. |
| `NodeToNodeEncryptionOptions` | `NodeToNodeEncryptionOptions` (p. 275) | Specify `true` to enable node-to-node encryption. |

## DomainEndpointOptions

| Field | Data Type | Description |
|---|---|---|
| `EnforceHTTPS` | Boolean | `true` to require that all traffic to the domain arrive over HTTPS. |
| `TLSSecurityPolicy` | String | The minimum TLS version required for traffic to the domain. Valid values are TLS 1.0 (default) or 1.2: <br>• `Policy-Min-TLS-1-0-2019-07` <br>• `Policy-Min-TLS-1-2-2019-07` |

## DomainID

| Data Type | Description |
|---|---|
| String | Unique identifier for an Amazon ES domain. |

## DomainName

Name of an Amazon ES domain.

| Data Type | Description |
|---|---|
| String | Name of an Amazon ES domain. Domain names are unique across all domains owned by the same account within an AWS Region. Domain names must start with a lowercase letter and must be between 3 and 28 characters. Valid characters are a-z (lowercase only), 0-9, and – (hyphen). |

## DomainNameList

String of Amazon ES domain names.

| Data Type | Description |
|---|---|
| String Array | Array of Amazon ES domains in the following format: `["<Domain_Name>","<Domain_Name>"...]` |

## DomainPackageDetails

Information on a package that is associated with a domain.

| Field | Data Type | Description |
|---|---|---|
| DomainName | String | Name of the domain you've associated a package with. |
| DomainPackageStatus | String | State of the association. Values are `ASSOCIATING`, `ASSOCIATION_FAILED`, `ACTIVE`, `DISSOCIATING`, and `DISSOCIATION_FAILED`. |
| ErrorDetails | String | Additional information if the package is in an error state. Null otherwise. |
| LastUpdated | Timestamp | Timestamp of the most-recent update to the association status. |
| PackageID | String | Internal ID of the package. |
| PackageName | String | User-specified name of the package. |
| PackageType | String | Currently supports only `TXT-DICTIONARY`. |
| ReferencePath | String | Denotes the location of the package on the Amazon ES cluster nodes. It's the same as `synonym_path` for dictionary files. |

## EBSOptions

Container for the parameters required to enable EBS-based storage for an Amazon ES domain.

| Field | Data Type | Description |
|---|---|---|
| `EBSEnabled` | Boolean | Indicates whether EBS volumes are attached to data nodes in an Amazon ES domain. |
| `VolumeType` | String | Specifies the type of EBS volumes attached to data nodes. |
| `VolumeSize` | String | Specifies the size (in GiB) of EBS volumes attached to data nodes. |
| `Iops` | String | Specifies the baseline input/output (I/O) performance of EBS volumes attached to data nodes. Applicable only for the Provisioned IOPS EBS volume type. |

## ElasticsearchClusterConfig

Container for the cluster configuration of an Amazon ES domain.

| Field | Data Type | Description |
|---|---|---|
| `InstanceType` | String | Instance type of data nodes in the cluster. |
| `InstanceCount` | Integer | Number of instances in the cluster. |
| `DedicatedMasterEnabled` | Boolean | Indicates whether dedicated master nodes are enabled for the cluster. `True` if the cluster will use a dedicated master node. `False` if the cluster will not. For more information, see About Dedicated Master Nodes (p. 176). |
| `DedicatedMasterType` | String | Amazon ES instance type of the dedicated master nodes in the cluster. |
| `DedicatedMasterCount` | Integer | Number of dedicated master nodes in the cluster. |
| `ZoneAwarenessEnabled` | Boolean | Indicates whether multiple Availability Zones are enabled. For more information, see the section called “Configuring a Multi-AZ Domain” (p. 17). |
| `ZoneAwarenessConfig` | `ZoneAwarenessConfig` (p. 279) | Container for zone awareness configuration options. Only required if `ZoneAwarenessEnabled` is `true`. |
| `WarmEnabled` | Boolean | Whether to enable warm storage for the cluster. |
| `WarmCount` | Integer | The number of warm nodes in the cluster. |
| `WarmType` | String | The instance type for the cluster's warm nodes. |
| `WarmStorage` | Integer | The total provisioned amount of warm storage in GiB. |

## ElasticsearchDomainConfig

Container for the configuration of an Amazon ES domain.

| Field | Data Type | Description |
|---|---|---|
| `ElasticsearchVersion` | String | Elasticsearch version. |
| `ElasticsearchClusterConfig` | `ElasticsearchClusterConfig` (p. 269) | Container for the cluster configuration of an Amazon ES domain. |
| `EBSOptions` | `EBSOptions` (p. 268) | Container for EBS options configured for an Amazon ES domain. |
| `AccessPolicies` | String | Specifies the access policies for the Amazon ES domain. For more information, see Configuring Access Policies (p. 13). |
| `SnapshotOptions` | `SnapshotOptions` (p. 277) | **DEPRECATED**. Hour during which the service takes an automated daily snapshot of the indices in the Amazon ES domain. |
| `DomainEndpointOptions` | `DomainEndpointOptions` (p. 267) | Additional options for the domain endpoint, such as whether to require HTTPS for all traffic. |
| `VPCOptions` | `VPCDerivedInfo` (p. 278) | The current VPCOptions (p. 278) for the domain and the status of any updates to their configuration. |
| `LogPublishingOptions` | `LogPublishingOptions` (p. 273) | Key-value pairs to configure slow log publishing. |
| `AdvancedOptions` | `AdvancedOptions` (p. 264) | Key-value pairs to specify advanced configuration options. |
| `EncryptionAtRestOptions` | `EncryptionAtRestOptions` (p. 273) | Key-value pairs to enable encryption at rest. |
| `NodeToNodeEncryptionOptions` | `NodeToNodeEncryptionOptions` (p. 275) | Whether node-to-node encryption is enabled or disabled. |

## ElasticsearchDomainStatus

Container for the contents of a `DomainStatus` data structure.

| Field | Data Type | Description |
|---|---|---|
| `DomainID` | `DomainID` (p. 267) | Unique identifier for an Amazon ES domain. |

| Field | Data Type | Description |
|---|---|---|
| `DomainName` | `DomainName` (p. 267) | Name of an Amazon ES domain. Domain names are unique across all domains owned by the same account within an AWS Region. Domain names must start with a lowercase letter and must be between 3 and 28 characters. Valid characters are a-z (lowercase only), 0-9, and – (hyphen). |
| `ARN` | `ARN` (p. 265) | Amazon Resource Name (ARN) of an Amazon ES domain. For more information, see Identifiers for IAM Entities in *Using AWS Identity and Access Management*. |
| `Created` | Boolean | Status of the creation of an Amazon ES domain. `True` if creation of the domain is complete. `False` if domain creation is still in progress. |
| `Deleted` | Boolean | Status of the deletion of an Amazon ES domain. `True` if deletion of the domain is complete. `False` if domain deletion is still in progress. |
| `Endpoint` | `ServiceUrl` (p. 277) | Domain-specific endpoint used to submit index, search, and data upload requests to an Amazon ES domain. |
| `Endpoints` | `EndpointsMap` (p. 273) | The key-value pair that exists if the Amazon ES domain uses VPC endpoints. |
| `Processing` | Boolean | Status of a change in the configuration of an Amazon ES domain. `True` if the service is still processing the configuration changes. `False` if the configuration change is active. You must wait for a domain to reach active status before submitting index, search, and data upload requests. |
| `ElasticsearchVersion` | String | Elasticsearch version. |
| `ElasticsearchClusterConfig` | `ElasticsearchClusterConfig` (p. 269) | Container for the cluster configuration of an Amazon ES domain. |

| Field | Data Type | Description |
|---|---|---|
| `EBSOptions` | `EBSOptions` (p. 268) | Container for the parameters required to enable EBS-based storage for an Amazon ES domain. |
| `AccessPolicies` | String | IAM policy document specifying the access policies for the new Amazon ES domain. For more information, see Configuring Access Policies (p. 13). |
| `SnapshotOptions` | `SnapshotOptions` (p. 277) | **DEPRECATED**. Container for parameters required to configure the time of daily automated snapshots of Amazon ES domain indices. |
| `VPCOptions` | `VPCDerivedInfo` (p. 278) | Information that Amazon ES derives based on VPCOptions (p. 278) for the domain. |
| `LogPublishingOptions` | `LogPublishingOptions` (p. 273) | Key-value pairs to configure slow log publishing. |
| `AdvancedOptions` | `AdvancedOptions` (p. 264) | Key-value pairs to specify advanced configuration options. |
| `EncryptionAtRestOptions` | `EncryptionAtRestOptions` (p. 273) | Key-value pairs to enable encryption at rest. |
| `CognitoOptions` | `CognitoOptions` (p. 266) | Key-value pairs to configure Amazon ES to use Amazon Cognito authentication for Kibana. |
| `NodeToNodeEncryptionOptions` | `NodeToNodeEncryptionOptions` (p. 275) | Whether node-to-node encryption is enabled or disabled. |
| `UpgradeProcessing` | Boolean | True if an upgrade to a new Elasticsearch version is in progress. |
| `ServiceSoftwareOptions` | the section called “ServiceSoftwareOptions” (p. 276) | The status of the domain's service software. |

## ElasticsearchDomainStatusList

List that contains the status of each specified Amazon ES domain.

| Field | Data Type | Description |
|---|---|---|
| `DomainStatusList` | `ElasticsearchDomainStatus` (p. 270) | List that contains the status of each specified Amazon ES domain. |

## EncryptionAtRestOptions

Specifies whether the domain should encrypt data at rest, and if so, the AWS Key Management Service (KMS) key to use. Can be used only to create a new domain, not update an existing one. To learn more, see the section called "Enabling Encryption of Data at Rest" (p. 61).

| Field | Data Type | Description |
|---|---|---|
| `Enabled` | Boolean | Specify `true` to enable encryption at rest. |
| `KmsKeyId` | String | The KMS key ID. Takes the form `1a2a3a4-1a2a-3a4a-5a6a-1a2a3a4a5a` |

## EndpointsMap

The key-value pair that contains the VPC endpoint. Only exists if the Amazon ES domain resides in a VPC.

| Field | Data Type | Description |
|---|---|---|
| `Endpoints` | Key-value string pair: `"vpc": "<VPC_ENDPOINT>"` | The VPC endpoint for the domain. |

## Filters

Filters the packages included in a the section called "DescribePackages" (p. 247) response.

| Field | Data Type | Description |
|---|---|---|
| `Name` | String | Any field from the section called "PackageDetails" (p. 275). |
| `Value` | List | A list of values for the specified field. |

## LogPublishingOptions

Specifies whether the Amazon ES domain publishes the Elasticsearch application and slow logs to Amazon CloudWatch. You still have to enable the *collection* of slow logs using the Elasticsearch REST API. To learn more, see the section called "Setting Elasticsearch Logging Thresholds for Slow Logs" (p. 43).

| Field | Data Type | Description |
|---|---|---|
| `INDEX_SLOW_LOGS` | Key-value | Two key-value pairs that define the CloudWatch log group and whether the Elasticsearch index slow log should be published there: `"CloudWatchLogsLogGroupArn":"arn:aws:logs:us-east-1:264071961897:log-group:sample-domain", "Enabled":true` |
| `SEARCH_SLOW_LOGS` | Key-value | Two key-value pairs that define the CloudWatch log group and whether the Elasticsearch search slow log should be published there: `"CloudWatchLogsLogGroupArn":"arn:aws:logs:us-east-1:264071961897:log-group:sample-domain", "Enabled":true` |
| `ES_APPLICATION_LOGS` | Key-value | Two key-value pairs that define the CloudWatch log group and whether the Elasticsearch error logs should be published there: `"CloudWatchLogsLogGroupArn":"arn:aws:logs:us-east-1:264071961897:log-group:sample-domain", "Enabled":true` |

## MasterUserOptions

| Field | Data Type | Description |
|---|---|---|
| `MasterUserARN` | String | ARN for the master user. Only specify if `InternalUserDatabaseEnabled` is `false` in the section called "AdvancedSecurityOptions" (p. 266). |
| `MasterUserName` | String | Username for the master user. Only specify if `InternalUserDatabaseEnabled` is `true` in the section called "AdvancedSecurityOptions" (p. 266). |
| `MasterUserPassword` | String | Password for the master user. Only specify if `InternalUserDatabaseEnabled` is `true` in the section called "AdvancedSecurityOptions" (p. 266). |

## NodeToNodeEncryptionOptions

Enables or disables node-to-node encryption.

| **Field** | **Data Type** | **Description** |
|---|---|---|
| `Enabled` | Boolean | Enable with `true`. |

## OptionState

State of an update to advanced options for an Amazon ES domain.

| **Field** | **Data Type** | **Description** |
|---|---|---|
| `OptionStatus` | String | One of three valid values:<br>• RequiresIndexDocuments<br>• Processing<br>• Active |

## OptionStatus

Status of an update to configuration options for an Amazon ES domain.

| **Field** | **Data Type** | **Description** |
|---|---|---|
| `CreationDate` | Timestamp | Date and time when the Amazon ES domain was created. |
| `UpdateDate` | Timestamp | Date and time when the Amazon ES domain was updated. |
| `UpdateVersion` | Integer | Whole number that specifies the latest version for the entity. |
| `State` | `OptionState` (p. 275) | State of an update to configuration options for an Amazon ES domain. |
| `PendingDeletion` | Boolean | Indicates whether the service is processing a request to permanently delete the Amazon ES domain and all of its resources. |

## PackageDetails

Basic information about a package.

| **Field** | **Data Type** | **Description** |
|---|---|---|
| `CreatedAt` | Timestamp | Name of the bucket containing the package. |

| Field | Data Type | Description |
|---|---|---|
| `ErrorDetails` | String | Additional information if the package is in an error state. Null otherwise. |
| `PackageDescription` | String | User-specified description of the package. |
| `PackageID` | String | Internal ID of the package. |
| `PackageName` | String | User-specified name of the package. |
| `PackageStatus` | String | Values are `COPYING`, `COPY_FAILED`, `AVAILABLE`, `DELETING`, or `DELETE_FAILED` . |
| `PackageType` | String | Currently supports only `TXT-DICTIONARY`. |

## PackageSource

Bucket and key for the package you want to add to Amazon ES.

| Field | Data Type | Description |
|---|---|---|
| `S3BucketName` | String | Name of the bucket containing the package. |
| `S3Key` | String | Key (file name) of the package. |

## ServiceSoftwareOptions

Container for the state of your domain relative to the latest service software.

| Field | Data Type | Description |
|---|---|---|
| `UpdateAvailable` | Boolean | Whether a service software update is available for your domain. |
| `Cancellable` | Boolean | If you have requested a domain update, whether or not you can cancel the update. |
| `AutomatedUpdateDate` | Timestamp | The Epoch time that the deployment window closes for required updates. After this time, Amazon ES schedules the software upgrade automatically. |
| `UpdateStatus` | String | The status of the update. Values are `ELIGIBLE`, `PENDING_UPDATE`, `IN_PROGRESS`, `COMPLETED`, and `NOT_ELIGIBLE`. |
| `Description` | String | More detailed description of the status. |
| `CurrentVersion` | String | Your current service software version. |
| `NewVersion` | String | The latest service software version. |

| Field | Data Type | Description |
|---|---|---|
| `OptionalDeployment` | Boolean | Whether the service software update is optional. |

## ServiceURL

Domain-specific endpoint used to submit index, search, and data upload requests to an Amazon ES domain.

| Field | Data Type | Description |
|---|---|---|
| `ServiceURL` | String | Domain-specific endpoint used to submit index, search, and data upload requests to an Amazon ES domain. |

## SnapshotOptions

**DEPRECATED**. See the section called "Working with Index Snapshots" (p. 44). Container for parameters required to configure the time of daily automated snapshots of the indices in an Amazon ES domain.

| Field | Data Type | Description |
|---|---|---|
| `AutomatedSnapshotStartHour` | Integer | **DEPRECATED**. Hour during which the service takes an automated daily snapshot of the indices in the Amazon ES domain. |

## Tag

| Field | Data Type | Description |
|---|---|---|
| `Key` | `TagKey` (p. 277) | Required name of the tag. Tag keys must be unique for the Amazon ES domain to which they are attached. For more information, see Tagging Amazon Elasticsearch Service Domains (p. 56). |
| `Value` | `TagValue` (p. 278) | Optional string value of the tag. Tag values can be `null` and do not have to be unique in a tag set. For example, you can have a key-value pair in a tag set of project/Trinity and cost-center/Trinity. |

## TagKey

| Field | Data Type | Description |
|---|---|---|
| `Key` | String | Name of the tag. String can have up to 128 characters. |

## TagList

| Field | Data Type | Description |
|---|---|---|
| `Tag` | `Tag` (p. 277) | Resource tag attached to an Amazon ES domain. |

## TagValue

| Field | Data Type | Description |
|---|---|---|
| `Value` | String | Holds the value for a `TagKey`. String can have up to 256 characters. |

## VPCDerivedInfo

| Field | Data Type | Description |
|---|---|---|
| `VPCId` | String | The ID for your VPC. Amazon VPC generates this value when you create a VPC. |
| `SubnetIds` | StringList | A list of subnet IDs associated with the VPC endpoints for the domain. For more information, see VPCs and Subnets in the *Amazon VPC User Guide*. |
| `AvailabilityZones` | StringList | The list of Availability Zones associated with the VPC subnets. For more information, see VPC and Subnet Basics in the *Amazon VPC User Guide*. |
| `SecurityGroupIds` | StringList | The list of security group IDs associated with the VPC endpoints for the domain. For more information, see Security Groups for your VPC in the *Amazon VPC User Guide*. |

## VPCOptions

| Field | Data Type | Description |
|---|---|---|
| `SubnetIds` | StringList | A list of subnet IDs associated with the VPC endpoints for the domain. If your domain uses multiple Availability Zones, you need to provide two subnet IDs, one per zone. Otherwise, provide only one. To learn more, see VPCs and Subnets in the *Amazon VPC User Guide*. |
| `SecurityGroupIds` | StringList | The list of security group IDs associated with the VPC endpoints for the domain. If you do not provide a security group ID, Amazon ES uses the default security group for the VPC. To learn more, |

| Field | Data Type | Description |
|---|---|---|
| | | see Security Groups for your VPC in the *Amazon VPC User Guide*. |
| `VPCId` | String | ID for the VPC. |

## ZoneAwarenessConfig

| Field | Data Type | Description |
|---|---|---|
| `AvailabilityZoneCount` | Integer | If you enabled multiple Availability Zones, this field is the number of zones that you want the domain to use. Valid values are `2` and `3`. |

## Errors

Amazon ES throws the following errors:

| Exception | Description |
|---|---|
| `BaseException` | Thrown for all service errors. Contains the HTTP status code of the error. |
| `ValidationException` | Thrown when the HTTP request contains invalid input or is missing required input. Returns HTTP status code 400. |
| `DisabledOperationException` | Thrown when the client attempts to perform an unsupported operation. Returns HTTP status code 409. |
| `InternalException` | Thrown when an error internal to the service occurs while processing a request. Returns HTTP status code 500. |
| `InvalidTypeException` | Thrown when trying to create or access an Amazon ES domain sub-resource that is either invalid or not supported. Returns HTTP status code 409. |
| `LimitExceededException` | Thrown when trying to create more than the allowed number and type of Amazon ES domain resources and sub-resources. Returns HTTP status code 409. |
| `ResourceNotFoundException` | Thrown when accessing or deleting a resource that does not exist. Returns HTTP status code 400. |
| `ResourceAlreadyExistsException` | Thrown when a client attempts to create a resource that already exists in an Amazon ES domain. Returns HTTP status code 400. |
| `AccessDeniedException` | |
| `ConflictException` | |

# Document History for Amazon Elasticsearch Service

This topic describes important changes to Amazon Elasticsearch Service.

**Relevant Dates to this History:**

- **Current product version—**2015-01-01
- **Latest product release—**May 5, 2020
- **Latest documentation update—**May 5, 2020

## Release Notes

The following table describes important changes to Amazon ES. For notifications about updates, you can subscribe to the RSS feed.

**Important**
Service software updates add support for new features, security patches, bug fixes, and other improvements. To use new features, you might need to update the service software on your domain. For more information, see the section called "Service Software Updates" (p. 15).

| update-history-change | update-history-description | update-history-date |
|---|---|---|
| UltraWarm (p. 280) | UltraWarm storage for Amazon Elasticsearch Service has left public preview and is now generally available. The feature now supports a wider range of Elasticsearch versions and AWS Regions. For more information, see the documentation. | May 5, 2020 |
| Custom Dictionaries (p. 280) | Amazon Elasticsearch Service lets you upload custom dictionary files for use with your cluster. These files improve your search results by telling Elasticsearch to ignore certain high-frequency words or to treat terms as equivalent. For more information, see Custom Packages. | April 21, 2020 |
| Elasticsearch 7.4 Support (p. 280) | Amazon Elasticsearch Service now supports Elasticsearch version 7.4. To learn more, see Supported Elasticsearch Versions and Upgrading Elasticsearch. | March 12, 2020 |
| KNN (p. 280) | Amazon Elasticsearch Service adds support for k-nearest | March 3, 2020 |

| | | |
|---|---|---|
| | neighbor (KNN) search. This feature requires service software R20200302 or later. For more information, see the documentation. | |
| Index State Management (p. 280) | Amazon Elasticsearch Service adds Index State Management (ISM), which lets you automate routine tasks, such as deleting indices when they reach a certain age. This feature requires service software R20200302 or later. For more information, see the documentation. | March 3, 2020 |
| Elasticsearch 5.6.16 Support (p. 280) | Amazon Elasticsearch Service now supports the latest patch release for version 5.6, which adds bug fixes and improves security. Amazon ES will automatically upgrade existing 5.6 domains to this release. Note that this Elasticsearch release incorrectly reports its version as 5.6.17. | March 2, 2020 |
| Fine-Grained Access Control (p. 280) | Amazon Elasticsearch Service now supports fine-grained access control, which offers security at the index, document, and field level, Kibana multi-tenancy, and optional HTTP basic authentication for your cluster. | February 11, 2020 |
| UltraWarm Storage (Preview) (p. 280) | Amazon Elasticsearch Service adds UltraWarm, a new warm storage tier that uses Amazon S3 and a sophisticated caching solution to improve performance. For indices that you are not actively writing to and query less frequently, UltraWarm storage offers significantly lower costs per GiB. For more information, see the documentation. | December 3, 2019 |
| Encryption Features for China Regions (p. 280) | Encryption of data at rest and node-to-node encryption are now available in the `cn-north-1` (Beijing) and `cn-northwest-1` (Ningxia) Regions. | November 20, 2019 |

| | | |
|---|---|---|
| Require HTTPS (p. 280) | You can now require that all traffic to your Amazon ES domains arrive over HTTPS. When configuring your domain, check the **Require HTTPS** box. This feature requires service software R20190808 or later. | October 3, 2019 |
| Elasticsearch 7.1 and 6.8 Support (p. 280) | Amazon Elasticsearch Service now supports Elasticsearch version 7.1 and 6.8. To learn more, see Supported Elasticsearch Versions and Upgrading Elasticsearch. | August 13, 2019 |
| Hourly Snapshots (p. 280) | Rather than daily snapshots, Amazon Elasticsearch Service now takes hourly snapshots of domains running Elasticsearch 5.3 and later so that you have more frequent backups from which to restore your data. To learn more, see Working with Amazon Elasticsearch Service Index Snapshots. | July 8, 2019 |
| Elasticsearch 6.7 Support (p. 280) | Amazon Elasticsearch Service now supports Elasticsearch version 6.7. To learn more, see Supported Elasticsearch Versions. | May 29, 2019 |
| SQL Support (p. 280) | Amazon Elasticsearch Service now lets you query your data using SQL. For more information, see SQL Support. This feature requires service software R20190418 or later. | May 15, 2019 |
| 5-series Instance Types (p. 280) | Amazon Elasticsearch Service now supports M5, C5, and R5 instance types. Compared to previous-generation instance types, these new types offer better performance at lower prices. For more information, see Supported Instance Types and Limits. | April 24, 2019 |
| Elasticsearch 6.5 Support (p. 280) | Amazon Elasticsearch Service now supports Elasticsearch version 6.5. To learn more, see Supported Elasticsearch Versions. | April 8, 2019 |

| | | |
|---|---|---|
| Alerting (p. 280) | The alerting feature notifies you when data from one or more Elasticsearch indices meets certain conditions. To learn more, see Alerting. This feature requires service software R20190221 or later. | March 25, 2019 |
| Three Availability Zone Support (p. 280) | Amazon Elasticsearch Service now supports three Availability Zones in many regions. This release also includes a streamlined console experience. To learn more, see Configuring a Multi-AZ Domain. This feature requires service software R20181023 or later. | February 7, 2019 |
| Elasticsearch 6.4 Support (p. 280) | Amazon Elasticsearch Service now supports Elasticsearch version 6.4. To learn more, see Supported Elasticsearch Versions. | January 23, 2019 |
| 200-Node Clusters (p. 280) | Amazon ES now lets you create clusters with up to 200 data nodes for a total of 3 PB of storage. To learn more, see Petabyte Scale. | January 22, 2019 |
| Service Software Updates (p. 280) | Amazon ES now lets you manually update the service software for your domain in order to benefit from new features more quickly or update at a low traffic time. To learn more, see Service Software Updates. | November 20, 2018 |
| New CloudWatch Metrics (p. 280) | Amazon ES now offers node-level metrics and new **Cluster health** and **Instance health** tabs in the Amazon ES console. To learn more, see Monitoring Cluster Metrics. | November 20, 2018 |
| China (Beijing) Support (p. 280) | Amazon Elasticsearch Service is now available in the cn-north-1 region, where it supports the M4, C4, and R4 instance types. | October 17, 2018 |
| Node-to-node Encryption (p. 280) | Amazon Elasticsearch Service now supports node-to-node encryption, which keeps your data encrypted as Elasticsearch distributes it throughout your cluster. To learn more, see Node-to-node Encryption. | September 18, 2018 |

| | | |
|---|---|---|
| In-place version upgrades (p. 280) | Amazon Elasticsearch Service now supports in-place version upgrades for Elasticsearch. To learn more, see Upgrading Elasticsearch. | August 14, 2018 |
| Elasticsearch 6.3 and 5.6 Support (p. 280) | Amazon Elasticsearch Service now supports Elasticsearch version 6.3 and 5.6. To learn more, see Supported Elasticsearch Versions. | August 14, 2018 |
| Error Logs (p. 280) | Amazon ES now lets you publish Elasticsearch error logs to Amazon CloudWatch. To learn more, see Configuring Logs. | July 31, 2018 |
| China (Ningxia) Reserved Instances (p. 280) | Amazon ES now offers Reserved Instances in the China (Ningxia) region. | May 29, 2018 |
| Reserved Instances (p. 280) | Amazon ES now offers Reserved Instances. To learn more, see Amazon ES Reserved Instances. | May 7, 2018 |

## Earlier Updates

The following table describes important changes Amazon ES before May 2018.

| Change | Description | Date |
|---|---|---|
| Amazon Cognito Authentication for Kibana | Amazon ES now offers login page protection for Kibana. To learn more, see the section called "Authentication for Kibana" (p. 99). | April 2, 2018 |
| Elasticsearch 6.2 Support | Amazon Elasticsearch Service now supports Elasticsearch version 6.2. | March 14, 2018 |
| Korean Analysis Plugin | Amazon ES now supports a memory-optimized version of the Seunjeon Korean analysis plugin. | March 13, 2018 |
| Instant Access Control Updates | Changes to the access control policies on Amazon ES domains now take effect instantly. | March 7, 2018 |
| Petabyte Scale | Amazon ES now supports I3 instance types and total domain storage of up to 1.5 PB. To learn more, see the section called "Petabyte Scale" (p. 175). | 19 December 2017 |
| Encryption of Data at Rest | Amazon ES now supports encryption of data at rest. To learn more, see the section called "Encryption at Rest" (p. 61). | December 7, 2017 |
| Elasticsearch 6.0 Support | Amazon ES now supports Elasticsearch version 6.0. For migration considerations and instructions, see the section called "Upgrading Elasticsearch" (p. 51). | December 6, 2017 |
| VPC Support | Amazon ES now lets you launch domains within an Amazon Virtual Private Cloud. VPC support provides an additional layer | October 17, 2017 |

| Change | Description | Date |
|---|---|---|
| | of security and simplifies communications between Amazon ES and other services within a VPC. To learn more, see the section called "VPC Support" (p. 21). | |
| Slow Logs Publishing | Amazon ES now supports the publishing of slow logs to CloudWatch Logs. To learn more, see the section called "Configuring Logs" (p. 40). | October 16, 2017 |
| Elasticsearch 5.5 Support | Amazon ES now supports Elasticsearch version 5.5. For new feature summaries, see the Amazon announcement of availability.<br><br>You can now restore automated snapshots without contacting AWS Support and store scripts using the Elasticsearch `_scripts` API. | September 7, 2017 |
| Elasticsearch 5.3 Support | Amazon ES added support for Elasticsearch version 5.3. | June 1, 2017 |
| More Instances and EBS Capacity per Cluster | Amazon ES now supports up to 100 nodes and 150 TB EBS capacity per cluster. | April 5, 2017 |
| Canada (Central) and EU (London) Support | Amazon ES added support for the following regions: Canada (Central), ca-central-1, and EU (London), eu-west-2. | March 20, 2017 |
| More Instances and Larger EBS Volumes | Amazon ES added support for more instances and larger EBS volumes. | February 21, 2017 |
| Elasticsearch 5.1 Support | Amazon ES added support for Elasticsearch version 5.1. | January 30, 2017 |
| Support for the Phonetic Analysis Plugin | Amazon ES now provides built-in integration with the Phonetic Analysis plugin, which allows you to run "sounds-like" queries on your data. | December 22, 2016 |
| US East (Ohio) Support | Amazon ES added support for the following region: US East (Ohio), us-east-2. | October 17, 2016 |
| New Performance Metric | Amazon ES added a performance metric, `ClusterUsedSpace`. | July 29, 2016 |
| Elasticsearch 2.3 Support | Amazon ES added support for Elasticsearch version 2.3. | July 27, 2016 |
| Asia Pacific (Mumbai) Support | Amazon ES added support for the following region: Asia Pacific (Mumbai), ap-south-1. | June 27, 2016 |
| More Instances per Cluster | Amazon ES increased the maximum number of instances (instance count) per cluster from 10 to 20. | May 18, 2016 |
| Asia Pacific (Seoul) Support | Amazon ES added support for the following region: Asia Pacific (Seoul), ap-northeast-2. | January 28, 2016 |
| Amazon ES | Initial release. | October 1, 2015 |

# AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.