
Amazon OpenSearch Service

Developer Guide

API Version 2015-01-01



Amazon OpenSearch Service : Developer Guide

Copyright © 2022 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 OpenSearch Service	1
Features of Amazon OpenSearch Service	1
Supported versions of OpenSearch and Elasticsearch	2
Pricing for Amazon OpenSearch Service	2
Getting started with Amazon OpenSearch Service	3
Related services	3
Amazon OpenSearch Service rename	5
New API version	5
Renamed instance types	5
Access policy changes	6
IAM policies	6
SCP policies	6
New resource types	6
Kibana renamed to OpenSearch Dashboards	7
Renamed CloudWatch metrics	7
Billing and Cost Management console changes	8
New event format	9
What's staying the same?	9
Get started: Upgrade your domains to OpenSearch 1.x	9
Getting started	11
Step 1: Create a domain	11
Step 2: Upload data for indexing	12
Option 1: Upload a single document	12
Option 2: Upload multiple documents	12
Step 3: Search documents	13
Search documents from the command line	13
Search documents using OpenSearch Dashboards	14
Step 4: Delete a domain	15
Next steps	15
Creating and managing domains	16
Creating OpenSearch Service domains	16
Creating OpenSearch Service domains (console)	16
Creating OpenSearch Service domains (AWS CLI)	19
Creating OpenSearch Service domains (AWS SDKs)	21
Creating OpenSearch Service domains (AWS CloudFormation)	21
Configuring access policies	21
Advanced cluster settings	21
Configuration changes	22
Changes that usually cause blue/green deployments	22
Changes that usually don't cause blue/green deployments	23
Determine whether a change will cause a blue/green deployment	23
Initiating a configuration change	24
Stages of a configuration change	24
Troubleshooting validation errors	26
Charges for configuration changes	28
Service software updates	29
Domain update considerations	29
Patch releases	29
Request a service software update (console)	30
Request a service software update (AWS CLI)	30
Request a service software update (SDK)	30
Monitoring service software update events	31
When domains are ineligible for an update	31
Notifications	32

Getting started with notifications	32
Notification severities	33
Sample EventBridge event	33
Configuring a multi-AZ domain	34
Shard distribution	34
Dedicated master node distribution	35
Availability zone disruptions	36
VPC support	37
VPC versus public domains	37
Limitations	37
Architecture	38
Creating index snapshots	42
Prerequisites	43
Registering a manual snapshot repository	45
Taking manual snapshots	48
Restoring snapshots	49
Deleting manual snapshots	51
Automating snapshots with Index State Management	51
Using Curator for snapshots	51
Upgrading Amazon OpenSearch Service domains	51
Supported upgrade paths	52
Starting an upgrade (console)	53
Starting an upgrade (CLI)	53
Starting an upgrade (SDK)	54
Troubleshooting validation failures	55
Troubleshooting an upgrade	55
Using a snapshot to migrate data	56
Creating a custom endpoint	59
Custom endpoints for new domains	59
Custom endpoints for existing domains	60
Next steps	60
Auto-Tune	60
Types of changes	60
Enabling or disabling Auto-Tune	61
Scheduling changes	62
Cron expressions	62
Tagging domains	63
Tagging examples	63
Working with tags (console)	64
Working with tags (AWS CLI)	64
Working with tags (AWS SDKs)	65
Monitoring domains	67
Monitoring cluster metrics	67
Viewing metrics in CloudWatch	68
Interpreting health charts in OpenSearch Service	68
Cluster metrics	69
Dedicated master node metrics	73
EBS volume metrics	74
Instance metrics	75
UltraWarm metrics	81
Cold storage metrics	84
Alerting metrics	84
Anomaly detection metrics	85
Asynchronous search metrics	86
SQL metrics	87
k-NN metrics	88
Cross-cluster search metrics	90

Cross-cluster replication metrics	90
Learning to Rank metrics	91
Piped Processing Language metrics	91
Monitoring logs	92
Enabling log publishing (console)	92
Enabling log publishing (AWS CLI)	94
Enabling log publishing (AWS SDKs)	95
Enabling log publishing (CloudFormation)	95
Setting OpenSearch logging thresholds for slow logs	96
Viewing logs	97
Monitoring audit logs	97
Limitations	98
Enabling audit logs	98
Enable audit logging using the AWS CLI	99
Enable audit logging using the configuration API	99
Audit log layers and categories	100
Audit log settings	101
Audit log example	103
Configuring audit logs using the REST API	105
Monitoring events	106
Service software update events	106
Auto-Tune events	108
Cluster health events	112
VPC endpoint events	117
Domain error events	119
Tutorial: Listening for OpenSearch Service events	120
Tutorial: Sending SNS alerts for available updates	122
Monitoring with CloudTrail	123
Amazon OpenSearch Service information in CloudTrail	123
Understanding Amazon OpenSearch Service log file entries	124
Security	126
Data protection	126
Encryption at rest	127
Node-to-node encryption	129
Identity and Access Management	130
Types of policies	130
Making and signing OpenSearch Service requests	135
When policies collide	136
Policy element reference	137
Advanced options and API considerations	139
Configuring access policies	141
Additional sample policies	142
AWS managed policies	142
Cross-service confused deputy prevention	145
Fine-grained access control	146
The bigger picture: fine-grained access control and OpenSearch Service security	146
Key concepts	149
Enabling fine-grained access control	149
Accessing OpenSearch Dashboards as the master user	151
Managing permissions	152
Recommended configurations	156
Limitations	157
Modifying the master user	158
Additional master users	158
Manual snapshots	159
Integrations	159
REST API differences	160

Tutorial: IAM master user and Amazon Cognito	161
Tutorial: Internal user database and HTTP basic authentication	163
Compliance validation	165
Resilience	166
Infrastructure security	166
Working with OpenSearch Service-managed VPC endpoints	167
SAML authentication for OpenSearch Dashboards	169
SAML configuration overview	169
SAML authentication for domains running in a VPC	170
Enabling SAML authentication	170
SAML troubleshooting	174
Disabling SAML authentication	175
Amazon Cognito authentication for OpenSearch Dashboards	175
Prerequisites	176
Configuring a domain to use Amazon Cognito	178
Allowing the authenticated role	180
Configuring identity providers	181
(Optional) Configuring granular access	182
(Optional) Customizing the sign-in page	184
(Optional) Configuring advanced security	184
Testing	184
Limits	184
Common configuration issues	185
Disabling Amazon Cognito authentication for OpenSearch Dashboards	187
Deleting domains that use Amazon Cognito authentication for OpenSearch Dashboards	187
Service-linked roles	188
Legacy Elasticsearch service-linked role	188
Permissions	188
Creating a service-linked role	189
Editing a service-linked role	189
Deleting a service-linked role	189
Sample code	191
Elasticsearch client compatibility	191
Signing HTTP requests	191
Java	192
Python	193
Ruby	196
Node	198
Go	202
Compressing HTTP requests	203
Enabling gzip compression	203
Required headers	204
Sample code (Python 3)	204
Using the AWS SDKs	205
Java	205
Python	212
Node	214
Indexing data	217
Naming restrictions for indexes	217
Reducing response size	217
Loading streaming data into OpenSearch Service	219
Loading streaming data from Amazon S3	219
Loading streaming data from Amazon Kinesis Data Streams	223
Loading streaming data from Amazon DynamoDB	226
Loading streaming data from Amazon Kinesis Data Firehose	229
Loading streaming data from Amazon CloudWatch	229
Loading streaming data from AWS IoT	229

Loading data with Logstash	229
Configuration	229
Searching data	232
URI searches	232
Request body searches	233
Boosting fields	234
Paginating search results	234
Search result highlighting	235
Count API	236
Dashboards Query Language	237
Custom packages	238
Package permissions requirements	238
Uploading packages to Amazon S3	239
Importing and associating packages	239
Using custom packages with OpenSearch	239
Updating custom packages (console)	241
Updating custom packages (AWS SDK)	242
Manual index updates	243
Dissociating and removing packages	245
SQL support	245
Sample call	246
Notes and differences	246
SQL Workbench	247
SQL CLI	247
JDBC driver	247
ODBC driver	248
k-NN search	248
Getting started with k-NN	249
k-NN differences, tuning, and limitations	251
Cross-cluster search	251
Limitations	251
Cross-cluster search prerequisites	252
Cross-cluster search pricing	252
Setting up a connection	252
Removing a connection	253
Setting up security and sample walkthrough	253
OpenSearch Dashboards	257
Learning to Rank	257
Getting started with Learning to Rank	258
Learning to Rank API	272
Asynchronous search	276
Sample search call	276
Asynchronous search permissions	277
Asynchronous search settings	278
Cross-cluster search	278
UltraWarm	279
OpenSearch Dashboards	280
Controlling access to OpenSearch Dashboards	280
Using a proxy to access OpenSearch Service from Dashboards	280
Configuring OpenSearch Dashboards to use a WMS map server	282
Connecting a local Dashboards server to OpenSearch Service	283
Managing indexes in OpenSearch Dashboards	284
Additional features	285
Managing indexes	286
UltraWarm storage	286
Prerequisites	287
UltraWarm storage requirements and performance considerations	288

UltraWarm pricing	288
Enabling UltraWarm	288
Migrating indexes to UltraWarm storage	290
Automating migrations	292
Migration tuning	292
Cancelling migrations	293
Listing hot and warm indexes	293
Returning warm indexes to hot storage	293
Restoring warm indexes from automated snapshots	293
Manual snapshots of warm indexes	294
Migrating warm indexes to cold storage	295
Disabling UltraWarm	295
Cold storage	295
Prerequisites	296
Cold storage requirements and performance considerations	297
Cold storage pricing	297
Enabling cold storage	297
Managing cold indexes in OpenSearch Dashboards	298
Migrating indexes to cold storage	299
Automating migrations to cold storage	300
Canceling migrations to cold storage	300
Listing cold indices	300
Migrating cold indexes to warm storage	303
Restoring cold indexes from snapshots	304
Canceling migrations from cold to warm storage	304
Updating cold index metadata	304
Deleting cold indices	305
Disabling cold storage	305
Index State Management	305
Create an ISM policy	305
Sample policies	306
ISM templates	309
Differences	309
Tutorial: Automating ISM processes	310
Index rollups	313
Creating an index rollup job	313
Index transforms	314
Creating an index transform job	314
Cross-cluster replication	315
Limitations	316
Prerequisites	316
Permissions requirements	316
Set up a cross-cluster connection	317
Start replication	317
Confirm replication	318
Pause and resume replication	319
Stop replication	319
Auto-follow	320
Remote reindex	321
Prerequisites	321
Reindex data between OpenSearch Service domains	321
Reindex data between OpenSearch Service domains in a VPC	323
Reindex data between non-OpenSearch Service domains	323
Reindex large datasets	324
Remote reindex settings	325
Data streams	325
Getting started with data streams	326

Monitoring data	328
Alerting	328
Differences	328
Anomaly detection	331
.....	331
Tutorial: Detect high CPU usage with anomaly detection	334
Observability	336
Explore your data with event analytics	336
Create visualizations	338
Dive deeper with Trace Analytics	338
Trace Analytics	338
Prerequisites	339
OpenTelemetry Collector sample configuration	339
Data Prepper sample configuration	340
Exploring trace data	342
Piped Processing Language	343
.....	343
Best practices	345
Monitoring and alerting	345
Configure CloudWatch alarms	345
Enable log publishing	345
Shard strategy	346
Determine shard and data node counts	346
Avoid storage skew	347
Stability	347
Keep current with OpenSearch	347
Back up your data	347
Enable dedicated master nodes	348
Deploy across multiple Availability Zones	348
Control ingest flow and buffering	348
Create mappings for search workloads	349
Use index templates	349
Manage indexes with Index State Management	350
Remove unused indexes	350
Use multiple domains for high availability	350
Performance	350
Optimize bulk request size and compression	350
Reduce the size of bulk request responses	351
Tune refresh intervals	351
Enable Auto-Tune	351
Security	351
Enable fine-grained access control	351
Deploy domains within a VPC	352
Apply a restrictive access policy	352
Enable encryption at rest	352
Enable node-to-node encryption	352
Cost optimization	352
Use the latest generation instance types	352
Use UltraWarm and cold storage for time-series log data	353
Review recommendations for Reserved Instances	353
Sizing domains	353
Calculating storage requirements	353
Choosing the number of shards	355
Choosing instance types and testing	355
Petabyte scale	357
Dedicated master nodes	358
Choosing the number of dedicated master nodes	359

Choosing instance types for dedicated master nodes	360
Recommended CloudWatch alarms	361
Other alarms you might consider	363
General reference	365
Supported instance types	365
Current generation instance types	365
Previous generation instance types	367
Features by engine version	369
Plugins by engine version	371
Supported operations	373
Notable API differences	374
OpenSearch version 2.3	375
OpenSearch version 1.3	376
OpenSearch version 1.2	377
OpenSearch version 1.1	378
OpenSearch version 1.0	379
Elasticsearch version 7.10	380
Elasticsearch version 7.9	381
Elasticsearch version 7.8	382
Elasticsearch version 7.7	383
Elasticsearch version 7.4	384
Elasticsearch version 7.1	385
Elasticsearch version 6.8	386
Elasticsearch version 6.7	387
Elasticsearch version 6.5	388
Elasticsearch version 6.4	389
Elasticsearch version 6.3	390
Elasticsearch version 6.2	391
Elasticsearch version 6.0	392
Elasticsearch version 5.6	392
Elasticsearch version 5.5	393
Elasticsearch version 5.3	394
Elasticsearch version 5.1	395
Elasticsearch version 2.3	396
Elasticsearch version 1.5	396
Quotas	397
Domain and instance quotas	397
UltraWarm storage quotas	398
EBS volume size quotas	399
Network quotas	402
Java process quota	404
Domain policy quota	404
Reserved Instances	404
Purchasing Reserved Instances (console)	405
Purchasing Reserved Instances (AWS CLI)	406
Purchasing Reserved Instances (AWS SDKs)	407
Examining costs	408
Other supported resources	409
Tutorials	410
Creating and searching for documents	410
Prerequisites	410
Adding a document to an index	410
Creating automatically generated IDs	411
Updating a document with a POST command	412
Performing bulk actions	412
Searching for documents	413
Related resources	414

Migrating to OpenSearch Service	415
Take and upload the snapshot	415
Create a domain	416
Provide permissions to the S3 bucket	416
Restore the snapshot	418
Creating a search application	419
Prerequisites	420
Step 1: Index sample data	420
Step 2: Create the API in API Gateway	420
Step 3: Create and deploy the Lambda function	422
Step 4: (Optional) Modify the domain access policy	423
Map the Lambda role (if using fine-grained access control)	424
Step 5: Test the web application	424
Next steps	426
Visualizing support calls	426
Step 1: Configure prerequisites	427
Step 2: Copy sample code	428
(Optional) Step 3: Index sample data	430
Step 4: Analyze and visualize your data	432
Step 5: Clean up resources and next steps	435
Troubleshooting	436
Can't access OpenSearch Dashboards	436
Can't access VPC domain	436
Cluster in read-only state	436
Red cluster status	437
Automatic remediation of red clusters	438
Recovering from a continuous heavy processing load	438
Yellow cluster status	439
ClusterBlockException	440
Lack of available storage space	440
High JVM memory pressure	440
JVM OutOfMemoryError	441
Failed cluster nodes	441
Exceeded maximum shard limit	442
Domain stuck in processing state	442
Low EBS burst balance	442
Can't enable audit logs	442
Can't close index	443
Client license checks	443
Request throttling	443
Can't SSH into node	443
"Not Valid for the Object's Storage Class" snapshot error	444
Invalid host header	444
Invalid M3 instance type	444
Hot queries stop working after enabling UltraWarm	444
Can't downgrade after upgrade	445
Need summary of domains for all AWS Regions	445
Browser error when using OpenSearch Dashboards	445
Node shard and storage skew	446
Index shard and storage skew	446
Unauthorized operation after selecting VPC access	447
Stuck at loading after creating VPC domain	447
Denied requests to the OpenSearch API	447
Can't connect from Alpine Linux	448
Certificate error when using SDK	448
Document history	450
Earlier updates	465

AWS glossary	468
--------------------	-----

What is Amazon OpenSearch Service?

Amazon OpenSearch Service is a managed service that makes it easy to deploy, operate, and scale OpenSearch clusters in the AWS Cloud. Amazon OpenSearch Service supports OpenSearch and legacy Elasticsearch OSS (up to 7.10, the final open source version of the software). When you create a cluster, you have the option of which search engine to use.

OpenSearch is a fully open-source search and analytics engine for use cases such as log analytics, real-time application monitoring, and clickstream analysis. For more information, see the [OpenSearch documentation](#).

Amazon OpenSearch Service provisions all the resources for your cluster and launches it. It also automatically detects and replaces failed OpenSearch Service 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 OpenSearch Service, you create an OpenSearch Service *domain*, which is equivalent to an OpenSearch *cluster*. Each EC2 instance in the cluster acts as one OpenSearch Service node.

You can use the OpenSearch Service 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](#).

Features of Amazon OpenSearch Service

OpenSearch Service includes the following features:

Scale

- Numerous configurations of CPU, memory, and storage capacity known as *instance types*, including cost-effective Graviton instances
- Up to 3 PB of attached storage
- Cost-effective [UltraWarm](#) (p. 286) and [cold storage](#) (p. 295) 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, HTTP basic, or SAML authentication for OpenSearch Dashboards
- Index-level, document-level, and field-level security
- Audit logs
- Dashboards 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 OpenSearch Service domains

Flexibility

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

Integration with popular services

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

Supported versions of OpenSearch and Elasticsearch

OpenSearch Service currently supports the following OpenSearch versions:

- 2.3, 1.3, 1.2, 1.1, 1.0

OpenSearch Service also supports the following legacy Elasticsearch OSS versions:

- 7.10, 7.9, 7.8, 7.7, 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

For more information, see [the section called “Supported operations” \(p. 373\)](#), [the section called “Features by engine version” \(p. 369\)](#), and [the section called “Plugins by engine version” \(p. 371\)](#).

If you start a new OpenSearch Service project, we strongly recommend that you choose the latest supported OpenSearch 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 Amazon OpenSearch Service domains” \(p. 51\)](#).

Pricing for Amazon OpenSearch Service

For OpenSearch Service, 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 \(p. 34\)](#), OpenSearch Service does not bill for traffic between the Availability Zones. Significant

data transfer occurs within a domain during shard allocation and rebalancing. OpenSearch Service neither meters nor bills for this traffic. Similarly, OpenSearch Service does not bill for data transfer between [UltraWarm \(p. 286\)](#)/[cold \(p. 295\)](#) nodes and Amazon S3.

For full pricing details, see [Amazon OpenSearch Service pricing](#). For information about charges incurred during configuration changes, see [the section called "Charges for configuration changes" \(p. 28\)](#).

Getting started with Amazon OpenSearch 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 \(p. 11\)](#) tutorial for Amazon OpenSearch Service. Consult the following introductory topics if you need more information while learning about the service:

- [Create a domain \(p. 16\)](#)
- [Size the domain \(p. 353\)](#) appropriately for your workload
- Control access to your domain using a [domain access policy \(p. 130\)](#) or [fine-grained access control \(p. 146\)](#)
- Index data [manually \(p. 217\)](#) or from [other AWS services \(p. 219\)](#)
- Use [OpenSearch Dashboards \(p. 280\)](#) to search your data and create visualizations

For information on migrating to OpenSearch Service from a self-managed OpenSearch cluster, see [the section called "Migrating to OpenSearch Service" \(p. 415\)](#).

Related services

OpenSearch Service commonly is used with the following services:

[Amazon CloudWatch](#)

OpenSearch Service domains automatically send metrics to CloudWatch so that you can monitor domain health and performance. For more information, see [Monitoring OpenSearch cluster metrics with Amazon CloudWatch \(p. 67\)](#).

CloudWatch Logs can also go the other direction. You might configure CloudWatch Logs to stream data to OpenSearch Service for analysis. To learn more, see [the section called "Loading streaming data from Amazon CloudWatch" \(p. 229\)](#).

[AWS CloudTrail](#)

Use AWS CloudTrail to get a history of the OpenSearch Service configuration API calls and related events for your account. For more information, see [Monitoring Amazon OpenSearch Service API calls with AWS CloudTrail \(p. 123\)](#).

[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 from Amazon Kinesis Data Streams" \(p. 223\)](#) and [the section called "Loading streaming data from Amazon Kinesis Data Firehose" \(p. 229\)](#).

[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 from Amazon S3" \(p. 219\)](#).

AWS IAM

AWS Identity and Access Management (IAM) is a web service that you can use to manage access to your OpenSearch Service domains. For more information, see [the section called "Identity and Access Management" \(p. 130\)](#).

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 OpenSearch Service" \(p. 219\)](#).

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 OpenSearch Service, see [the section called "Loading streaming data from Amazon DynamoDB" \(p. 226\)](#).

Amazon QuickSight

You can visualize data from OpenSearch Service using Amazon QuickSight dashboards. For more information, see [Using Amazon OpenSearch Service with Amazon QuickSight](#) in the *Amazon QuickSight User Guide*.

Note

OpenSearch includes certain Apache-licensed Elasticsearch code from Elasticsearch B.V. and other source code. Elasticsearch B.V. is not the source of that other source code. ELASTICSEARCH is a registered trademark of Elasticsearch B.V.

Amazon OpenSearch Service - Summary of changes

On September 8, 2021, Amazon Elasticsearch Service was renamed to Amazon OpenSearch Service. OpenSearch Service supports OpenSearch as well as legacy Elasticsearch OSS. The following sections describe the different parts of the service that changed with the rename, and what actions you need to take to ensure that your domains continue to function properly.

Some of these changes only apply when you upgrade your domains from Elasticsearch to OpenSearch. In other cases, such as in the Billing and Cost Management console, the experience changes immediately.

Note that this list is not exhaustive. While other parts of the product also changed, these updates are the most relevant.

Topics

- [New API version \(p. 5\)](#)
- [Renamed instance types \(p. 5\)](#)
- [Access policy changes \(p. 6\)](#)
- [New resource types \(p. 6\)](#)
- [Kibana renamed to OpenSearch Dashboards \(p. 7\)](#)
- [Renamed CloudWatch metrics \(p. 7\)](#)
- [Billing and Cost Management console changes \(p. 8\)](#)
- [New event format \(p. 9\)](#)
- [What's staying the same? \(p. 9\)](#)
- [Get started: Upgrade your domains to OpenSearch 1.x \(p. 9\)](#)

New API version

The new version of the OpenSearch Service configuration API (2021-01-01) works with OpenSearch as well as legacy Elasticsearch OSS. 21 API operations were replaced with more concise and engine-agnostic names (for example, `CreateElasticsearchDomain` changed to `CreateDomain`), but OpenSearch Service continues to support both API versions.

We recommend that you use the new API operations to create and manage domains going forward. Note that when you use the new API operations to create a domain, you need to specify the `EngineVersion` parameter in the format `Elasticsearch_X.Y` or `OpenSearch_X.Y`, rather than just the version number. If you don't specify a version, it defaults to the latest version of OpenSearch.

Upgrade your AWS CLI to version 1.20.40 or later in order to use `aws opensearch ...` to create and manage your domains. For the new CLI format, see the [OpenSearch CLI reference](#).

Renamed instance types

Instance types in Amazon OpenSearch Service are now in the format `<type>. <size>. search`—for example, `m6g.large.search` rather than `m6g.large.elasticsearch`. You don't need to take any action. Existing domains will start automatically referring to the new instance types within the API and in the Billing and Cost Management console.

If you have Reserved Instances (RIs), your contract won't be impacted by the change. The old configuration API version is still compatible with the old naming format, but if you want to use the new API version, you need to use the new format.

Access policy changes

The following sections describe what actions you need to take to update your access policies.

IAM policies

We recommend that you update your [IAM policies \(p. 130\)](#) to use the renamed API operations. However, OpenSearch Service will continue to respect existing policies by internally replicating the old API permissions. For example, if you currently have permission to perform the `CreateElasticsearchDomain` operation, you can now make calls to both `CreateElasticsearchDomain` (old API operation) and `CreateDomain` (new API operation). The same applies to explicit denies. For a list of updated API operations, see the [policy element reference \(p. 137\)](#).

SCP policies

[Service control policies \(SCPs\)](#) introduce an additional layer of complexity compared to standard IAM. To prevent your SCP policies from breaking, you need to add both the old *and* the new API operations to each of your SCP policies. For example, if a user currently has allow permissions for `CreateElasticsearchDomain`, you also need to grant them allow permissions for `CreateDomain` so they can retain the ability to create domains. The same applies to explicit denies.

For example:

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "es:CreateElasticsearchDomain",
            "es:CreateDomain"
            ...
        ],
        "Effect": "Deny",
        "Action": [
            "es:DeleteElasticsearchDomain",
            "es:DeleteDomain"
            ...
        ]
    }
]
```

New resource types

OpenSearch Service introduces the following new resource types:

Resource	Description
<code>AWS::OpenSearchService::Domain</code>	Represents an Amazon OpenSearch Service domain. This resource exists at the service level and isn't specific to the software running on the domain. It applies to services like AWS

Resource	Description
	<p>CloudFormation and AWS Resource Groups, in which you create and manage resources for the service as a whole.</p> <p>For instructions to upgrade domains defined within CloudFormation from Elasticsearch to OpenSearch, see Remarks in the CloudFormation User Guide.</p>
AWS::OpenSearch::Domain	<p>Represents OpenSearch/Elasticsearch software running on a domain. This resource applies to services like AWS CloudTrail and AWS Config, which reference the software running on the domain rather than OpenSearch Service as a whole. These services now contain separate resource types for domains running Elasticsearch (AWS::Elasticsearch::Domain) versus domains running OpenSearch (AWS::OpenSearch::Domain).</p>

Note

In [AWS Config](#), you'll continue to see your data under the existing AWS::Elasticsearch::Domain resource type for several weeks, even if you upgrade one or more domains to OpenSearch.

Kibana renamed to OpenSearch Dashboards

[OpenSearch Dashboards \(p. 280\)](#), the AWS alternative to Kibana, is an open-source visualization tool designed to work with OpenSearch. After you upgrade a domain from Elasticsearch to OpenSearch, the /_plugin/kibana endpoint changes to /_dashboards. OpenSearch Service will redirect all requests to the new endpoint, but if you use the Kibana endpoint in any of your IAM policies, update those policies to include the new /_dashboards endpoint as well.

If you're using [the section called "SAML authentication for OpenSearch Dashboards" \(p. 169\)](#), before you upgrade your domain to OpenSearch, you need to change all Kibana URLs configured in your identity provider (IdP) from /_plugin/kibana to /_dashboards. The most common URLs are assertion consumer service (ACS) URLs and recipient URLs.

The default kibana_read_only role for OpenSearch Dashboards was renamed to opensearch_dashboards_read_only, and the kibana_user role was renamed to opensearch_dashboards_user. The change applies to all *newly-created* OpenSearch 1.x domains running service software R20211203 or later. If you upgrade an existing domain to service software R20211203, the role names remain the same.

Renamed CloudWatch metrics

Several CloudWatch metrics change for domains running OpenSearch. When you upgrade a domain to OpenSearch, the metrics change automatically and your current CloudWatch alarms will break. Before upgrading your cluster from an Elasticsearch version to an OpenSearch version, make sure to update your CloudWatch alarms to use the new metrics.

The following metrics changed:

Original metric name	New name
KibanaHealthyNodes	OpenSearchDashboardsHealthyNodes
KibanaConcurrentConnections	OpenSearchDashboardsConcurrentConnections
KibanaHeapTotal	OpenSearchDashboardsHeapTotal
KibanaHeapUsed	OpenSearchDashboardsHeapUsed
KibanaHeapUtilization	OpenSearchDashboardsHeapUtilization
KibanaOS1MinuteLoad	OpenSearchDashboardsOS1MinuteLoad
KibanaRequestTotal	OpenSearchDashboardsRequestTotal
KibanaResponseTimesMaxInMillis	OpenSearchDashboardsResponseTimesMaxInMillis
ESReportingFailedRequestSysErrCount	KibanaReportingFailedRequestSysErrCount
ESReportingRequestCount	KibanaReportingRequestCount
ESReportingFailedRequestUserErrCount	KibanaReportingFailedRequestUserErrCount
ESReportingSuccessCount	KibanaReportingSuccessCount
ElasticsearchRequests	OpenSearchRequests

For a full list of metrics that OpenSearch Service sends to Amazon CloudWatch, see [the section called "Monitoring cluster metrics" \(p. 67\)](#).

Billing and Cost Management console changes

Historic data in the [Billing and Cost Management](#) console and in [Cost and Usage Reports](#) will continue to use the old service name, so you need to start using filters for both **Amazon Elasticsearch Service** and **Amazon OpenSearch Service** when searching for data. If you have existing saved reports, update the filters to make sure they also include OpenSearch Service. You might initially receive an alert when your usage decreases for Elasticsearch and increases for OpenSearch, but it disappears within several days.

The following fields will change for all reports, bills, and price list API operations:

Field	Old format	New format
Instance type	m5.large.elasticsearch	m5.large.search
Product name	Amazon Elasticsearch Service	Amazon OpenSearch Service
Product family	Elasticsearch Instance Elasticsearch Volume	Amazon OpenSearch Service Instance Amazon OpenSearch Service Volume
Pricing description	\$5.098 per c5.18xlarge.elasticsearch instance hour (or partial hour) - EU	\$5.098 per c5.18xlarge.search instance hour (or partial hour) - EU

Field	Old format	New format
Service name	Amazon Elasticsearch Service	Amazon OpenSearch Service
Instance family	ultrawarm.elasticsearch	ultrawarm.search

New event format

The format of events that OpenSearch Service sends to Amazon EventBridge and Amazon CloudWatch has changed, specifically the detail-type field. The source field (aws.es) remains the same. For the complete format for each event type, see [the section called "Monitoring events" \(p. 106\)](#). If you have existing event rules that depend on the old format, make sure to update them to conform to the new format.

What's staying the same?

The following features and functionality, among others not listed, will remain the same:

- Service principal (es.amazonaws.com)
- Vendor code
- Domain ARNs
- Domain endpoints

Get started: Upgrade your domains to OpenSearch 1.x

OpenSearch 1.x supports upgrades from Elasticsearch versions 6.8 and 7.x. For instructions to upgrade your domain, see [the section called "Starting an upgrade \(console\)" \(p. 53\)](#). If you're using the AWS CLI or configuration API to upgrade your domain, you need to specify the TargetVersion as OpenSearch_1.x.

OpenSearch 1.x introduces an additional domain setting called **Enable compatibility mode**. Because certain Elasticsearch OSS clients and plugins check the cluster version before connecting, compatibility mode sets OpenSearch to report its version as 7.10 so these clients continue to work.

You can enable compatibility mode when you create OpenSearch domains for the first time, or when you upgrade to OpenSearch from an Elasticsearch version. If it's not set, the parameter defaults to false when you create a domain, and true when you upgrade a domain.

To enable compatibility mode using the [configuration API](#), set override_main_response_version to true:

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/upgradeDomain
{
    "DomainName": "domain-name",
    "TargetVersion": "OpenSearch_1.0",
    "AdvancedOptions": {
        "override_main_response_version": "true"
    }
}
```

To enable or disable compatibility mode on *existing* OpenSearch domains, you need to use the OpenSearch [_cluster/settings](#) API operation:

```
PUT /_cluster/settings
{
  "persistent" : {
    "compatibility.override_main_response_version" : true
  }
}
```

Getting started with Amazon OpenSearch Service

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

This tutorial walks you through the basic steps to get an OpenSearch Service domain up and running quickly. For more detailed information, see [Creating and managing domains \(p. 16\)](#) and the other topics within this guide. For information on migrating to OpenSearch Service from a self-managed OpenSearch cluster, see [the section called “Migrating to OpenSearch Service” \(p. 415\)](#).

You can complete the following steps by using the OpenSearch Service console, the AWS CLI, or the AWS SDK:

1. [Create a domain \(p. 11\)](#)
2. [Upload data for indexing \(p. 12\)](#)
3. [Search documents \(p. 13\)](#)
4. [Delete a domain \(p. 15\)](#)

For information about installing and setting up the AWS CLI, see the [AWS Command Line Interface User Guide](#).

Step 1: Create an Amazon OpenSearch Service domain

Important

This is a concise tutorial for configuring a *test* Amazon OpenSearch Service domain. Do not use this process to create production domains. For a comprehensive version of the same process, see [Creating and managing domains \(p. 16\)](#).

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

To create an OpenSearch Service domain using the console

1. Go to <https://aws.amazon.com> and choose **Sign In to the Console**.
2. Under **Analytics**, choose **Amazon OpenSearch Service**.
3. Choose **Create domain**.
4. Provide a name for the domain. The examples in this tutorial use the name *movies*.
5. Ignore the **Custom endpoint** setting.
6. For the deployment type, choose **Development and testing**.
7. For **Version**, choose the latest version.
8. Under **Data nodes**, change the instance type to *t3.small.search* and keep the default value of three nodes.
9. For simplicity in this tutorial, use a public access domain. Under **Network**, choose **Public access**.

10. In the fine-grained access control settings, choose **Create master user**. Provide a user name and password.
11. For now, ignore the **SAML authentication** and **Amazon Cognito authentication** sections.
12. For **Access policy**, choose **Only use fine-grained access control**. In this tutorial, fine-grained access control handles authentication, not the domain access policy.
13. Ignore the rest of the settings and choose **Create**. New domains typically take 15–30 minutes to initialize, but can take longer depending on the configuration. After your domain initializes, select it to open its configuration pane. Note the domain endpoint under **General information** (for example, <https://search-my-domain.us-east-1.es.amazonaws.com>), which you'll use in the next step.

Next: [Upload data to an OpenSearch Service domain for indexing \(p. 12\)](#)

Step 2: Upload data to Amazon OpenSearch Service for indexing

Important

This is a concise tutorial for uploading a small amount of test data to Amazon OpenSearch Service. For more about uploading data in a production domain, see [Indexing data \(p. 217\)](#).

You can upload data to an OpenSearch 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's required if your access policies specify IAM users or roles. To successfully complete this process, you must use fine-grained access control with a primary user name and password like you configured in [Step 1 \(p. 11\)](#).

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 preinstalled.

Option 1: Upload a single document

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'
```

In the command, provide the user name and password that you created in [Step 1 \(p. 11\)](#).

For a detailed explanation of this command and how to make signed requests to OpenSearch Service, see [Indexing data \(p. 217\)](#).

Option 2: Upload multiple documents

To upload a JSON file that contains multiple documents to an OpenSearch Service domain

1. Create a local file called `bulk_movies.json`. Paste the following content into the file 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"}
```

- Run the following command in the local directory where the file is stored to upload it 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 \(p. 217\)](#).

Next: [Search documents \(p. 13\)](#)

Step 3: Search documents in Amazon OpenSearch Service

To search documents in an Amazon OpenSearch Service domain, use the OpenSearch search API. Alternatively, you can use [OpenSearch Dashboards \(p. 280\)](#) to search documents in the domain.

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.

You should see a response similar to the following:

```
{
  "took" : 5,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
```

```
    "skipped" : 0,
    "failed" : 0
},
"hits" : {
  "total" : {
    "value" : 1,
    "relation" : "eq"
  },
  "max_score" : 0.2876821,
  "hits" : [
    {
      "_index" : "movies",
      "_type" : "_doc",
      "_id" : "1",
      "_score" : 0.2876821,
      "_source" : {
        "director" : "Burton, Tim",
        "genre" : [
          "Comedy",
          "Sci-Fi"
        ],
        "year" : 1996,
        "actor" : [
          "Jack Nicholson",
          "Pierce Brosnan",
          "Sarah Jessica Parker"
        ],
        "title" : "Mars Attacks!"
      }
    }
  ]
}
```

Search documents using OpenSearch Dashboards

OpenSearch Dashboards is a popular open source visualization tool designed to work with OpenSearch. It provides a helpful user interface for you to search and monitor your indices.

To search documents from an OpenSearch Service domain using Dashboards

1. Navigate to the OpenSearch Dashboards URL for your domain. You can find the URL on the domain's dashboard in the OpenSearch Service console. The URL follows this format:

```
domain-endpoint/_dashboards/
```

2. Log in using your primary user name and password.
3. To use Dashboards, you need to create at least one index pattern. Dashboards uses these patterns to identify which indexes you want to analyze. Open the left navigation panel, choose **Stack Management**, choose **Index Patterns**, and then choose **Create index pattern**. For this tutorial, enter **movies**.
4. Choose **Next step** and then choose **Create index pattern**. After the pattern is created, you can view the various document fields such as **actor** and **director**.
5. Go back to the **Index Patterns** page and make sure that **movies** is set as the default. If it's not, select the pattern and choose the star icon to make it the default.
6. To begin searching your data, open the left navigation panel again and choose **Discover**.
7. In the search bar, enter *mars* if you uploaded a single document, or *rebel* if you uploaded multiple documents, and then press **Enter**. You can try searching other terms, such as **actor** or **director** names.

Next: [Delete a domain \(p. 15\)](#)

Step 4: Delete an Amazon OpenSearch Service domain

Because the **movies** domain from this tutorial is for test purposes, make sure to delete it when you're done experimenting to avoid incurring charges.

To delete an OpenSearch Service domain from the console

1. Sign in to the **Amazon OpenSearch Service** console.
2. Under **Domains**, select the **movies** domain.
3. Choose **Delete** and confirm deletion.

Next steps

Now that you know how to create a domain and index data, you might want to try some of the following exercises:

- Learn about more advanced options for creating a domain. For more information, see [Creating and managing domains \(p. 16\)](#).
- Discover how to manage the indices in your domain. For more information, see [Managing indexes \(p. 286\)](#).
- Try out one of the tutorials for working with Amazon OpenSearch Service. For more information, see [Tutorials \(p. 410\)](#).

Creating and managing Amazon OpenSearch Service domains

This chapter describes how to create and manage Amazon OpenSearch Service domains. An OpenSearch Service domain is synonymous with an OpenSearch 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 tutorial \(p. 11\)](#), this chapter describes all options and provides relevant reference information. You can complete each procedure by using instructions for the OpenSearch Service console, the AWS Command Line Interface (AWS CLI), or the AWS SDKs.

Creating OpenSearch Service domains

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

Creating OpenSearch Service domains (console)

Use the following procedure to create an OpenSearch Service domain by using the console.

To create an OpenSearch Service domain (console)

1. Go to <https://aws.amazon.com> and choose **Sign In to the Console**.
2. Under **Analytics**, choose **Amazon OpenSearch Service**.
3. Choose **Create domain**.
4. For **Domain name**, enter a domain name. The name must meet the following criteria:
 - Unique to your account and AWS 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 (-)
5. If you want to use a custom endpoint rather than the standard one of `https://search-mydomain-1a2a3a4a5a6a7a8a9a0a9a8a7a.us-east-1.es.amazonaws.com`, choose **Enable custom endpoint** and provide a name and certificate. For more information, see [the section called "Creating a custom endpoint" \(p. 59\)](#).
6. For **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).

7. For **Version**, choose the version of OpenSearch or legacy Elasticsearch OSS to use. We recommend that you choose the latest version of OpenSearch. For more information, see [the section called "Supported versions of OpenSearch and Elasticsearch" \(p. 2\)](#).

(Optional) If you chose an OpenSearch version for your domain, select **Enable compatibility mode** to make OpenSearch report its version as 7.10, which allows certain Elasticsearch OSS clients and plugins that check the version before connecting to continue working with the service.

8. For **Auto-Tune**, choose whether to allow OpenSearch Service to suggest memory-related configuration changes to your domain to improve speed and stability. For more information, see [the section called "Auto-Tune" \(p. 60\)](#).
- (Optional) Select **Add maintenance window** to schedule a recurring window during which Auto-Tune updates the domain.
9. Under **Data nodes**, choose the number of availability zones. For more information, see [the section called "Configuring a multi-AZ domain" \(p. 34\)](#).

Note

The OpenSearch Service console doesn't support moving from multiple availability zones to a single availability zone after the domain is created. If you choose 2 or 3 availability zones and later want to move to 1, you must disable the `ZoneAwarenessEnabled` parameter using the AWS CLI or configuration API.

10. For **Instance type**, choose an instance type for your data nodes. For more information, see [the section called "Supported instance types" \(p. 365\)](#).

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.

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

For maximum values, see [the section called "Domain and instance quotas" \(p. 397\)](#). 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 domains" \(p. 353\)](#) and [the section called "Configuring a multi-AZ domain" \(p. 34\)](#).

12. For **Storage type**, select Amazon EBS or instance store volumes to associate with your instance. The volume types available in the list depend on the instance type that you've chosen. For guidance on creating especially large domains, see [the section called "Petabyte scale" \(p. 357\)](#).
13. If you chose **EBS** as the storage type, configure the following additional settings. Some settings might not appear depending on the type of volume you choose.

Setting	Description
EBS volume type	Choose between General Purpose (SSD) - gp3 and General Purpose (SSD) - gp2 , or the previous generation Provisioned IOPS (SSD) , and Magnetic (standard) .
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 OpenSearch Service 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. 399) .
Provisioned IOPS	If you selected a Provisioned IOPS SSD volume type, enter the number of I/O operations per second (IOPS) that the volume can support.

14. (Optional) If you selected a gp3 volume type, expand **Advanced settings** and specify additional IOPS (up to 1,000 MiB/s) and throughput (up to 16,000) to provision for each node, beyond what is included with the price of storage, for an additional cost. For more information, see the [Amazon OpenSearch Service pricing](#).
 15. Choose the type and number of [dedicated master nodes \(p. 358\)](#). 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.
16. (Optional) To enable [UltraWarm storage \(p. 286\)](#), choose **Enable UltraWarm data nodes**. Each instance type has a [maximum amount of storage \(p. 398\)](#) that it can address. Multiply that amount by the number of warm data nodes for the total addressable warm storage.
 17. (Optional) To enable [cold storage \(p. 295\)](#), choose **Enable cold storage**. You must enable UltraWarm to enable cold storage.
 18. (Optional) For domains running OpenSearch or Elasticsearch 5.3 and later, the **Snapshot configuration** is irrelevant. For more information about automated snapshots, see [the section called "Creating index snapshots" \(p. 42\)](#).
 19. Under **Network**, choose either **VPC access** or **Public access**. If you choose **Public access**, skip to the next step. If you choose **VPC access**, make sure you meet the [prerequisites \(p. 40\)](#), then configure the following settings:

Setting	Description
VPC	Choose the ID of the virtual private cloud (VPC) that you want to use. The VPC and domain must be in the same AWS Region, and you must select a VPC with tenancy set to Default . OpenSearch Service does not yet support VPCs that use dedicated tenancy.
Subnet	Choose a subnet. If you enabled Multi-AZ, you must choose two or three subnets. OpenSearch Service will place a VPC endpoint and <i>elastic network interfaces</i> in the subnets. You must reserve sufficient IP addresses for the network interfaces in the subnet(s). For more information, see Reserving IP addresses in a VPC subnet (p. 41) .
Security groups	Choose one or more VPC security groups that allow your required application to reach the OpenSearch Service domain on the ports (80 or 443) and protocols (HTTP or HTTPS) exposed by the domain. For more information, see the section called "VPC support" (p. 37) .
IAM Role	Keep the default role. OpenSearch Service uses this predefined role (also known as a <i>service-linked role</i>) 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. 42) .

20. Enable or disable fine-grained access control:
 - If you want to use IAM for user management, choose **Set IAM ARN as master user** and specify the ARN for an IAM role.
 - If you want to use the internal user database, choose **Create master user** and specify a user name and password.

Whichever option you choose, the master user can access all indexes in the cluster and all OpenSearch APIs. For guidance on which option to choose, see [the section called "Key concepts" \(p. 149\)](#).

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.

Note

We *strongly* recommend enabling fine-grained access control to protect the data on your domain. Fine-grained access control provides security at the cluster, index, document, and field levels.

21. (Optional) If you want to use SAML authentication for OpenSearch Dashboards, choose **Prepare SAML authentication**. After the domain is available, see [the section called "SAML authentication for OpenSearch Dashboards" \(p. 169\)](#) for additional steps.
22. (Optional) If you want to use Amazon Cognito authentication for OpenSearch Dashboards, choose **Enable Amazon Cognito authentication**. Then choose the Amazon Cognito user pool and identity pool that you want to use for OpenSearch Dashboards authentication. For guidance on creating these resources, see [the section called "Amazon Cognito authentication for OpenSearch Dashboards" \(p. 175\)](#).
23. For **Domain access policy**, choose an access policy or configure one of your own. If you choose to create a custom policy, you can configure it yourself or import one from another domain. For more information, see [the section called "Identity and Access Management" \(p. 130\)](#).

Note

If you enabled VPC access, you can't use IP-based policies. 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" \(p. 39\)](#).

24. (Optional) To require that all requests to the domain arrive over HTTPS, select **Require HTTPS for all traffic to the domain**.
25. (Optional) To enable node-to-node encryption, select **Node-to-node encryption**. For more information, see [the section called "Node-to-node encryption" \(p. 129\)](#).
26. (Optional) To enable encryption of data at rest, select **Enable encryption of data at rest**.

Select **Use AWS owned key** to have OpenSearch Service create an AWS KMS encryption key on your behalf (or use the one that it already created). Otherwise, choose your own KMS key. For more information, see [the section called "Encryption at rest" \(p. 127\)](#).

27. (Optional) Add tags to describe your domain so you can categorize and filter on that information. For more information, see [the section called "Tagging domains" \(p. 63\)](#).
28. (Optional) Expand and configure **Advanced cluster settings**. For a summary of these options, see [the section called "Advanced cluster settings" \(p. 21\)](#).
29. Choose **Create**.

Creating OpenSearch Service domains (AWS CLI)

Instead of creating an OpenSearch Service domain by using the console, you can use the AWS CLI. For syntax, see Amazon OpenSearch Service in the [AWS CLI command reference](#).

Example commands

This first example demonstrates the following OpenSearch Service domain configuration:

- Creates an OpenSearch Service domain named *mylogs* with OpenSearch version 1.2

- Populates the domain with two instances of the `r6g.large.search` instance type
- Uses a 100 GiB General Purpose (SSD) gp3 EBS volume for storage for each data node
- Allows anonymous access, but only from a single IP address: 192.0.2.0/32

```
aws opensearch create-domain --domain-name mylogs --engine-version OpenSearch_1.2
--cluster-config InstanceType=r6g.large.search,InstanceCount=2 --ebs-options
EBSEnabled=true,VolumeType=gp3,VolumeSize=100,Iops=3500,Throughput=125 --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 OpenSearch Service domain configuration:

- Creates an OpenSearch Service domain named `mylogs` with Elasticsearch version 7.10
- Populates the domain with six instances of the `r6g.large.search` instance type
- Uses a 100 GiB General Purpose (SSD) gp2 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 opensearch create-domain --domain-name mylogs --
engine-version Elasticsearch_7.10 --cluster-config
InstanceType=r6g.large.search,InstanceCount=6,ZoneAwarenessEnabled=true,ZoneAwarenessConfig={Availability
--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 OpenSearch Service domain configuration:

- Creates an OpenSearch Service domain named `mylogs` with OpenSearch version 1.0
- Populates the domain with ten instances of the `r6g.xlarge.search` instance type
- Populates the domain with three instances of the `r6g.large.search` 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 opensearch create-domain --domain-name mylogs --engine-version OpenSearch_1.0 --
cluster-config
InstanceType=r6g.xlarge.search,InstanceCount=10,DedicatedMasterEnabled=true,DedicatedMasterType=r6g.la
--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 OpenSearch Service 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 OpenSearch Service domains (AWS SDKs)

The AWS SDKs (except the Android and iOS SDKs) support all the actions defined in the [Amazon OpenSearch Service API Reference](#), including `CreateDomain`. For sample code, see the section called “Using the AWS SDKs” (p. 205). For more information about installing and using the AWS SDKs, see [AWS Software Development Kits](#).

Creating OpenSearch Service domains (AWS CloudFormation)

OpenSearch Service is integrated with AWS CloudFormation, a service that helps you to model and set up your AWS resources so that you can spend less time creating and managing your resources and infrastructure. You create a template that describes the OpenSearch domain you want to create, and CloudFormation provisions and configures the domain for you. For more information, including examples of JSON and YAML templates for OpenSearch domains, see the [Amazon OpenSearch Service resource type reference](#) in the [AWS CloudFormation User Guide](#).

Configuring access policies

Amazon OpenSearch Service offers several ways to configure access to your OpenSearch Service domains. For more information, see the section called “Identity and Access Management” (p. 130) and the section called “Fine-grained access control” (p. 146).

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 OpenSearch Service domains. For information about how these access policies interact with VPC access, see the section called “About access policies on VPC domains” (p. 39).

To configure access policies (console)

1. Go to <https://aws.amazon.com>, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Amazon OpenSearch Service**.
3. In the navigation pane, under **Domains**, choose the domain you want to update.
4. Choose **Actions** and **Edit security configuration**.
5. Edit the access policy JSON, or import a preconfigured option.
6. Choose **Save changes**.

Advanced cluster settings

Use advanced options to configure the following:

Indices in request bodies

Specifies whether explicit references to indexes 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. 139).

Fielddata cache allocation

Specifies the percentage of Java heap space that is allocated to field data. By default, this setting is 20% of the JVM heap.

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 of these use cases. For very large indices, you might need a large field data cache.

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](#).

Making configuration changes in Amazon OpenSearch Service

Amazon OpenSearch Service 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 OpenSearch Service, 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.

Changes that usually cause blue/green deployments

The following operations cause blue/green deployments:

- Changing instance type
- Enabling fine-grained access control
- Performing service software updates
- If your domain *doesn't* have dedicated master nodes, changing data instance count
- Enabling or disabling dedicated master nodes
- Changing dedicated master node count or instance type
- 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 OpenSearch Dashboards
- Choosing a different Amazon Cognito user pool or identity pool
- Modifying advanced settings
- Enabling or disabling the publication of error logs, audit logs, or slow logs to CloudWatch
- Upgrading to a new OpenSearch version
- Enabling or disabling **Require HTTPS**
- Enabling encryption of data at rest or node-to-node encryption
- Enabling or disabling UltraWarm or cold storage
- Disabling Auto-Tune and rolling back its changes
- Modifying the custom endpoint

Changes that usually don't cause blue/green deployments

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

- Changing access policy
- Changing the automated snapshot hour
- Enabling Auto-Tune or disabling it without rolling back its changes
- If your domain has dedicated master nodes, changing data node or UltraWarm node count

There are some exceptions depending on your service software version. If you want to be absolutely sure that a change will not cause a blue/green deployment, [perform a dry run \(p. 23\)](#) before updating your domain.

Determine whether a change will cause a blue/green deployment

You can conduct a dry run of a planned configuration change to determine whether it will cause a blue/green deployment. When making a change in the console, you're prompted to choose **Run analysis**, and OpenSearch Service calculates the type of deployment the change will cause.

You can also perform a dry run analysis through the configuration API. For example, this [UpdateDomainConfig](#) request tests the deployment type caused by enabling UltraWarm:

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/my-domain/config
{
  "ClusterConfig": {
    "WarmCount": 3,
    "WarmEnabled": true,
    "WarmType": "ultrawarm1.large.search"
  },
  "DryRun": true
}
```

The request returns the type of deployment the change will cause but doesn't actually perform the update:

```
{
  "ClusterConfig": {
    ...
  },
  "DryRunResults": {
    "DeploymentType": "Blue/Green",
    "Message": "This change will require a blue/green deployment."
  }
}
```

Possible deployment types are:

- Blue/Green - The change will cause a blue/green deployment.
- DynamicUpdate - The change won't cause a blue/green deployment.
- Undetermined - The domain is still in a processing state, so the deployment type can't be determined.
- None - No configuration change.

Initiating a configuration change

When you initiate a configuration change, the domain state changes to **Processing** until OpenSearch Service has created a new environment with the latest [service software \(p. 29\)](#), at which point it changes back to **Active**. During certain service software updates, the state remains **Active** the whole time. 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 \(p. 358\)](#), which suddenly might have many more nodes to manage. It can also increase search and indexing latencies as OpenSearch Service copies data from the old cluster to the new one. It's important to maintain sufficient capacity on the cluster 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're billed only for the number of nodes that you request for your cluster. For specifics, see [the section called "Charges for configuration changes" \(p. 28\)](#).

To prevent overloading dedicated master nodes, you can [monitor usage with the Amazon CloudWatch metrics \(p. 67\)](#). For recommended maximum values, see [the section called "Recommended CloudWatch alarms" \(p. 361\)](#).

Stages of a configuration change

After you initiate a configuration change, OpenSearch Service goes through a series of steps to update your domain. You can view the progress of the configuration change under **Domain status** in the console. The exact steps that an update goes through depends on the type of change you're making. You can also monitor a configuration change using the [DescribeDomainChangeProgress>](#) API operation.

In some cases, such as during service software updates, you won't see progress information until the blue/green deployment actually starts. During this time the domain status is **Pending Updates**.

The following are possible stages an update can go through during a configuration change:

Stage name	Description
Validation	Validating that the domain is eligible for an update, and surfacing validation issues (p. 26) if necessary.
Creating a new environment	Completing the necessary

Stage name	Description
	prerequisites and creating required resources to start the blue/green deployment.
Provisioning new nodes	Creating a new set of instances in the new environment.
Traffic routing on new nodes	Redirecting traffic to the newly created data nodes.
Traffic routing on old nodes	Disabling traffic on the old data nodes.
Preparing nodes for removal	Preparing to remove nodes. This step only happens when you're downscaling your domain (for example, from 8 nodes to 6 nodes).
Copying shards to new nodes	Moving shards from the old nodes to the new nodes.
Terminating nodes	Terminating and deleting old nodes after shards are removed.
Deleting older resources	Deleting resources associated with the old environment (e.g. load balancer).

Stage name	Description
Dynamic update	Displayed when the update does not require a blue/green deployment and can be dynamically applied.

Troubleshooting validation errors

When you initiate a configuration change or perform an OpenSearch or Elasticsearch version upgrade, OpenSearch Service first performs a series of validation checks to ensure that your domain is eligible for an update. If any of these checks fail, you receive a notification in the console containing the specific issues that you must fix before updating your domain. The following table lists the possible domain issues that OpenSearch Service might surface, and steps to resolve them.

Issue	Error code	Troubleshooting steps
Security group not found	SecurityGroupNotFound	The security group associated with your OpenSearch Service domain does not exist. To resolve this issue, create a security group with the specified name.
Subnet not found	SubnetNotFound	The subnet associated with your OpenSearch Service domain does not exist. To resolve this issue, create a subnet in your VPC.
Service-linked role not configured	SLRNotConfigured	The service-linked role (p. 188) for OpenSearch Service is not configured. The service-linked role is predefined by OpenSearch Service and includes all the permissions the service requires to call other AWS services on your behalf. If the role doesn't exist, you might need to create it manually (p. 189).
Not enough IP addresses	InsufficientIPAddresses	Check IP address subnet ABC subnets don't have enough IP addresses to update your domain. To calculate how many IP addresses you need, see the section called "Reserving IP addresses in a VPC subnet" (p. 41) .
Cognito user pool doesn't exist	CognitoUserPoolNotFound	OpenSearch Service can't find the Amazon Cognito 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:
		<pre>aws cognito-idp list-user-pools --max-results 60 --region us-east-1</pre>
Cognito identity pool doesn't exist	CognitoIdentityPoolNotFound	OpenSearch Service can't find the Cognito 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:
		<pre>aws cognito-identity list-identity-pools --max-results 60 --region us-east-1</pre>
Cognito domain not	CognitoDomainNameNotConfigured	The Cognito user pool does not have a domain name. You can configure one using the Amazon Cognito console or the following AWS CLI command:

Issue	Error code	Troubleshooting steps
found for user pool		<pre>aws cognito-identity create-user-pool-domain --domain <i>my-domain</i> --user-pool-id <i>id</i></pre>
Cognito role not configured	CognitoRoleNotConfigured	<p>The IAM role that grants OpenSearch Service permission to configure the Amazon Cognito user and identity pools, and use them for authentication, is not configured. Configure the role with an appropriate permission set and trust relationship. You can use the console, which creates the default CognitoAccessForAmazonOpenSearch (p. 177) role for you, or you can manually configure a role using the AWS CLI or the AWS SDK.</p>
Unable to describe user pool	UserPoolNotDescribed	<p>The specified Amazon Cognito role doesn't have permission to describe the user pool associated with your domain. Make sure the role permissions policy allows the <code>cognito-identity:DescribeUserPool</code> action. See the section called "About the CognitoAccessForAmazonOpenSearch role" (p. 177) for the full permissions policy.</p>
Unable to describe identity pool	IdentityPoolNotDescribed	<p>The specified Amazon Cognito role doesn't have permission to describe the identity pool associated with your domain. Make sure the role permissions policy allows the <code>cognito-identity:DescribeIdentityPool</code> action. See the section called "About the CognitoAccessForAmazonOpenSearch role" (p. 177) for the full permissions policy.</p>
Unable to describe user and identity pool	CognitoPoolNotDescribed	<p>The specified Amazon Cognito role doesn't have permission to describe the user and identity pools associated with your domain. Make sure the role permissions policy allows the <code>cognito-identity:DescribeIdentityPool</code> and <code>cognito-identity:DescribeUserPool</code> actions. See the section called "About the CognitoAccessForAmazonOpenSearch role" (p. 177) for the full permissions policy.</p>
KMS key not enabled	KMSKeyNotEnabled	<p>The AWS Key Management Service (AWS KMS) key used to encrypt your domain is disabled. Re-enable the key immediately.</p>
Custom certificate not in ISSUED state	InvalidCertificateStatus	<p>If your domain uses a custom endpoint, you secure it by either generating an SSL certificate in AWS Certificate Manager (ACM) or importing one of your own. The certificate status must be Issued. If you receive this error, check the status of your certificate in the ACM console. If the status is Expired, Failed, Inactive, or Pending validation, see the ACM troubleshooting documentation to resolve the issue.</p>
Not enough capacity to launch chosen instance type	InsufficientCapacity	<p>The requested instance type capacity is not available. For example, you might have requested five <code>i3.16xlarge.search</code> nodes, but OpenSearch Service doesn't have enough <code>i3.16xlarge.search</code> hosts available, so the request can't be fulfilled. Check the supported instance types (p. 365) in OpenSearch Service and choose a different instance type.</p>
Red indexes in cluster	RedCluster	<p>One or more indexes in your cluster have a red status, leading to an overall red cluster status. To troubleshoot and remediate this issue, see the section called "Red cluster status" (p. 437).</p>

Issue	Error code	Troubleshooting steps
Memory circuit breaker, too many requests	TooManyRequestsException	There are too many search and write requests to your domain, so OpenSearch Service can't update its configuration. You can reduce the number of requests, scale instances vertically up to 64 GiB of RAM, or scale horizontally by adding instances.
New configuration can't hold data (low disk space)	InsufficientStorageException	The storage capacity size can't hold all of the data on your domain. To resolve this issue, choose a larger volume (p. 399) , delete unused indexes , or increase the number of nodes in the cluster to immediately free up disk space.
Shards pinned to specific nodes	ShareMovementBlockedException	One or more indexes in your domain are attached to specific nodes and can't be reassigned. This most likely happened because you configured shard allocation filtering, which lets you specify which nodes are allowed to host the shards of a particular index. To resolve this issue, remove shard allocation filters from all affected indexes: <pre>PUT my-index/_settings { "settings": { "index.routing.allocation.require._name": null } }</pre>
New configuration can't hold all shards (shard count)	TooManyShardsException	The shard count on your domain is too high, which prevents OpenSearch Service from moving them to the new configuration. To resolve this issue, scale your domain horizontally by adding nodes of the same configuration type as your current cluster nodes. Note that the maximum EBS volume size (p. 399) depends on the node's instance type. To prevent this issue in the future, see the section called "Choosing the number of shards" (p. 355) and define a sharding strategy that is appropriate for your use case.

Charges for configuration changes

If you change the configuration for a domain, OpenSearch Service creates a new cluster as described in [the section called "Configuration changes" \(p. 22\)](#). During the migration of old to new, you incur the following charges:

- If you change the instance type, you're charged for both clusters for the first hour. After the first hour, you're only charged for the new cluster. EBS volumes aren't charged twice because they're part of your cluster, so their billing follows instance billing.

Example: You change the configuration from three m3.xlarge instances to four m4.large instances. For the first hour, you're charged for both clusters ($3 * \text{m3.xlarge} + 4 * \text{m4.large}$). After the first hour, you're charged only for the new cluster ($4 * \text{m4.large}$).

- If you don't change the instance type, you're charged only for the largest cluster for the first hour. After the first hour, you're 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're charged for the largest cluster (6 * m3.xlarge). After the first hour, you're charged only for the new cluster (3 * m3.xlarge).

Service software updates in Amazon OpenSearch Service

Note

For descriptions of the changes and additions made in each major service software release, see the [release notes \(p. 450\)](#).

Amazon OpenSearch Service regularly releases service software updates that add features or otherwise improve your domains. The **Notifications** panel in the console is the easiest way to see if an update is available or to check the status of an update. Each notification includes details about the service software update. The notification severity is **Informational** if the update is optional and **High** if it's required.

Service software updates differ from OpenSearch version upgrades. For information about upgrading to a later version of OpenSearch, see [the section called "Upgrading Amazon OpenSearch Service domains" \(p. 51\)](#).

Domain update considerations

Consider the following when deciding whether to update your domain:

- If you take no action on available updates, OpenSearch Service eventually updates your domain for you. If an update is *optional*, OpenSearch Service updates your domain the next time you make a change that causes a [blue/green deployment \(p. 22\)](#). If the update is *required*, OpenSearch Service initiates a blue/green deployment after a certain timeframe (typically two weeks) if the domain hasn't already been updated. You receive notifications when the update starts and when it's complete.
- If you start an update manually, OpenSearch Service doesn't send a notification when it starts the update, only when the update is complete.
- Software updates use blue/green deployments to minimize downtime. Updates can temporarily strain a cluster's dedicated master nodes, so make sure to maintain sufficient capacity to handle the associated overhead.

Manually updating your domain lets you take advantage of new features more quickly. When you choose **Update**, OpenSearch Service places the request in a queue and begins the update when it has time. Updates typically complete within minutes, but can also take several hours or even days if your system is experiencing heavy load. Consider updating your domain at a low traffic time to avoid long update periods.

Patch releases

Service software versions that end in "-P" and a number, such as R20211203-P4, are patch releases. Patches are likely to include performance improvements, minor bug fixes, and security fixes or posture improvements. Patch releases do not include new features or breaking changes, and they generally do not have a direct or noticeable impact on users.

Request a service software update (console)

To request a service software update (console)

1. Go to <https://aws.amazon.com> and choose **Sign In to the Console**.
2. Under **Analytics**, choose **Amazon OpenSearch Service**.
3. In the navigation pane, under **Domains**, choose the domain name to open its settings.
4. Choose **Actions, Update** and confirm the update.

Request a service software update (AWS CLI)

Send the following AWS CLI command to request a service software update:

```
aws opensearch start-service-software-update --domain-name my-domain
```

For more information, see [start-service-software-update](#) in the AWS CLI command reference and [StartServiceSoftwareUpdate](#) in the configuration API reference.

Tip

After requesting an update, you might have a narrow window of time in which you can cancel it. The duration of this PENDING_UPDATE state can vary greatly and depends on your AWS Region and the number of concurrent updates OpenSearch Service is performing. To cancel, use the console or `cancel-service-software-update` (`CancelServiceSoftwareUpdate`) command.

Request a service software update (SDK)

This sample Python script uses the `describe_domain` and `start_service_software_update` methods from the AWS SDK for Python (Boto3) to check whether a domain is eligible for a service software update and if so, starts the update. You must provide a value for `domain_name`.

```
import boto3
from botocore.config import Config
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default region.

my_config = Config(
    # Optionally lets you specify a Region other than your default.
    region_name='us-east-1'
)

domain_name = '' # The name of the domain to check and update

client = boto3.client('opensearch', config=my_config)

def getUpdateStatus(client):
    """Determines whether the domain is eligible for an update"""
    response = client.describe_domain(
        DomainName=domain_name
    )
    sso = response['DomainStatus']['ServiceSoftwareOptions']
    if sso['UpdateStatus'] == 'ELIGIBLE':
```

```

        print('Domain [' + domain_name + '] is eligible for a service software update from
version ' +
              sso['CurrentVersion'] + ' to version ' + sso['NewVersion'])
        updateDomain(client)
    else:
        print('Domain is not eligible for an update at this time.')

def updateDomain(client):
    """Starts a service software update for the eligible domain"""
    response = client.start_service_software_update(
        DomainName=domain_name
    )
    print('Updating domain [' + domain_name + '] to version ' +
          response['ServiceSoftwareOptions']['NewVersion'] + '...')
    waitForUpdate(client)

def waitForUpdate(client):
    """Waits for the domain to finish updating"""
    response = client.describe_domain(
        DomainName=domain_name
    )
    status = response['DomainStatus']['ServiceSoftwareOptions']['UpdateStatus']
    if status == 'PENDING_UPDATE' or status == 'IN_PROGRESS':
        time.sleep(30)
        waitForUpdate(client)
    elif status == 'COMPLETED':
        print('Domain [' + domain_name +
              '] successfully updated to the latest software version')
    else:
        print('Domain is not currently being updated.')

def main():
    getUpdateStatus(client)

```

Monitoring service software update events

OpenSearch Service sends a [notification \(p. 32\)](#) when a service software update is available, required, started, completed, or failed. You can view these notifications on the **Notifications** panel of the OpenSearch Service console. It also sends these notifications to Amazon EventBridge. You can use EventBridge to configure rules that send an email or perform a specific action when an event is received. For an example walkthrough, see [the section called “Tutorial: Sending SNS alerts for available updates” \(p. 122\)](#).

To see the format of each service software event sent to Amazon EventBridge, see the section called [“Service software update events” \(p. 106\)](#).

When domains are ineligible for an update

Your domain is ineligible for a service software update if it's in any of the following states:

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 indexes in the cluster is red. For troubleshooting steps, see the section called “Red cluster status” (p. 437) .

State	Description
High error rate	The OpenSearch cluster is returning a large number of 5xx 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	<i>Split brain</i> means your OpenSearch 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. 358) . For help recovering from split brain, contact AWS Support .
Amazon Cognito integration issue	Your domain uses authentication for OpenSearch Dashboards (p. 175) , and OpenSearch Service 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 OpenSearch Service domain to use it.
Other OpenSearch Service service issue	Issues with OpenSearch Service 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 .

Notifications in Amazon OpenSearch Service

Notifications in Amazon OpenSearch Service currently contain information about available software updates and Auto-Tune events for your domains. In the future, they might also include performance optimization recommendations such as moving to the correct instance type for a domain or rebalancing shards to reduce performance bottlenecks.

You can view notifications in the **Notifications** panel of the OpenSearch Service console or in [Amazon EventBridge](#). Some notifications are also available in the [AWS Personal Health Dashboard](#). They're available for all versions of OpenSearch and Elasticsearch OSS, with some minor exceptions. For the format of each event sent to EventBridge, see [the section called "Monitoring events" \(p. 106\)](#).

Getting started with notifications

Notifications are enabled automatically when you create a domain. Go to the **Notifications** panel of the OpenSearch Service console to monitor and acknowledge notifications. Each notification includes information such as the time it was posted, the domain it relates to, a severity and status level, and a brief explanation. You can view historical notifications for up to 90 days in the console.

After accessing the **Notifications** panel or acknowledging a notification, you might receive an error message about not having permissions to perform `es>ListNotifications` or `esUpdateNotificationStatus`. To resolve this problem, give your user or role the following permissions in IAM:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "es:UpdateNotificationStatus",
        "es>ListNotifications"
      ],
      "Resource": "arn:aws:es:*:123456789012:domain/*"
    }
  ]
}
```

The IAM console throws an error ("IAM does not recognize one or more actions.") that you can safely ignore. You can also restrict the es:UpdateNotificationStatus action to certain domains. To learn more, see [the section called "Policy element reference" \(p. 137\)](#).

Notification severities

Notifications in OpenSearch Service can be *informational*, which relate to any action you've already taken or the operations of your domain, or *actionable*, which require you to take specific actions such as applying a mandatory security patch. Each notification has a severity associated with it, which can be Informational, Low, Medium, High, or Critical. The following table summarizes each severity:

Severity	Description	Examples
Informational	Information related to the operation of your domain.	<ul style="list-style-type: none"> Service software update available Auto-Tune started
Low	A recommended action, but has no adverse impact on domain availability or performance if no action is taken.	<ul style="list-style-type: none"> Auto-Tune cancelled High shard count warning
Medium	There might be an impact if the recommended action is not taken, but comes with an extended time window for the action to be taken.	<ul style="list-style-type: none"> Service software update failed Shard count limit exceeded
High	Urgent action is required to avoid adverse impact.	<ul style="list-style-type: none"> Service software update required KMS key inaccessible
Critical	Immediate action is required to avoid adverse impact, or to recover from it.	None currently available

Sample EventBridge event

The following example shows an OpenSearch Service notification event sent to Amazon EventBridge. The notification has a severity of Informational because the update is optional:

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Amazon OpenSearch Service Software Update Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2016-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
  "detail": {
    "event": "Service Software Update",
    "status": "Available",
    "severity": "Informational",
    "description": "Service software update [R20200330-p1] available."
  }
}
```

Configuring a multi-AZ domain in Amazon OpenSearch Service

To prevent data loss and minimize Amazon OpenSearch Service 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. Availability Zones are isolated locations within each AWS Region.

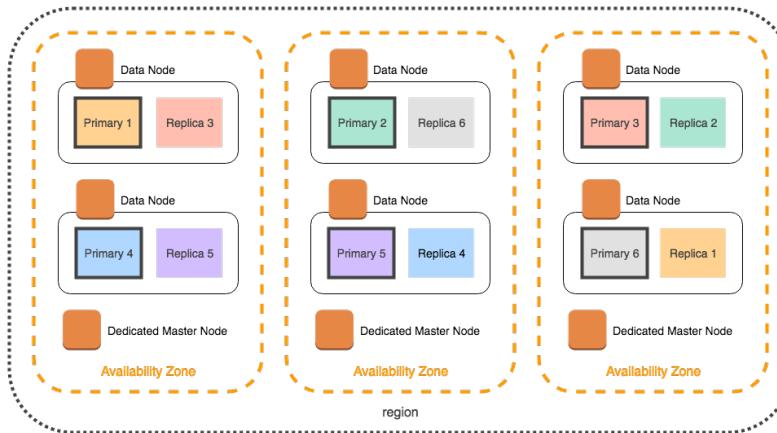
For domains that run production workloads, we recommend the following configuration:

- Choose a Region that supports three Availability Zones with OpenSearch Service.
- 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

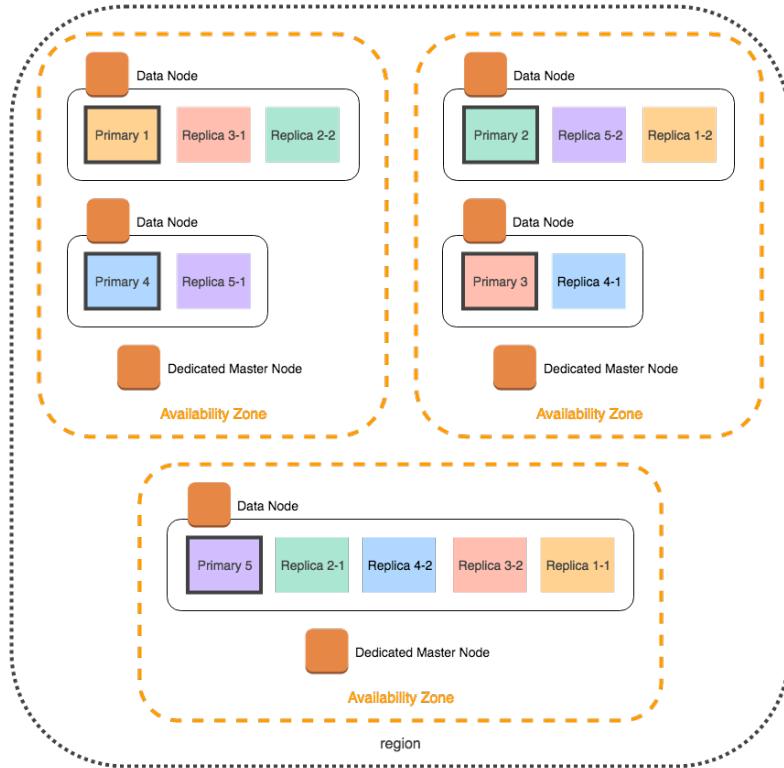
If you enable Multi-AZ, you should create at least one replica for each index in your cluster. Without replicas, OpenSearch Service can't distribute copies of your data to other Availability Zones, which largely defeats the purpose of Multi-AZ. Fortunately, the default configuration for any index is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



In addition to distributing shards by Availability Zone, OpenSearch Service 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

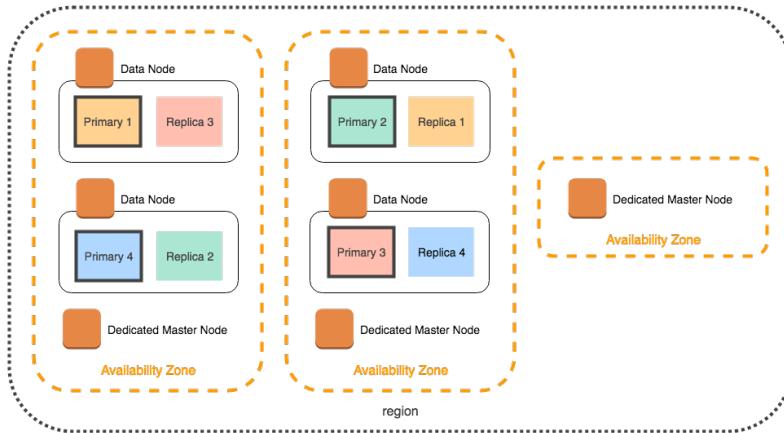
In this situation, OpenSearch Service has to overload one node in order to distribute the primary and replica shards across the zones, as shown in the following diagram.



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, OpenSearch Service automatically distributes [dedicated master nodes \(p. 358\)](#) 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, OpenSearch Service throws an error. Choose a different instance type, and try again.
 - If you chose two Availability Zones for the domain, OpenSearch Service 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 OpenSearch Service can only distribute dedicated master nodes across two zones).

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 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. OpenSearch Service distributes two dedicated master nodes into one Availability Zone and one into the other: <ul style="list-style-type: none"> • 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. • If the Availability Zone with two dedicated master nodes experiences a disruption, the cluster is unavailable until the remaining Availability Zone recovers.
3 or more	2	3	No downtime. OpenSearch Service 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 OpenSearch Service 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

OpenSearch Service manages Multi-AZ domains transparently, so you can't manually simulate Availability Zone disruptions.

Launching your Amazon OpenSearch Service domains within a VPC

You can launch AWS resources, such as Amazon OpenSearch Service domains, into a *virtual private cloud* (VPC). A VPC is a virtual network that's dedicated to your AWS account. It's logically isolated from other virtual networks in the AWS Cloud. Placing an OpenSearch Service domain within a VPC enables secure communication between OpenSearch Service 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.

Note

If you place your OpenSearch Service domain within a VPC, your computer must be able to connect to the VPC. This connection often takes the form of a VPN, transit gateway, managed network, or proxy server. You can't directly access your domains from outside the VPC.

VPC versus public domains

The following are some of the ways VPC domains differ from public domains. Each difference is described later in more detail.

- Because of their logical isolation, domains that reside within a VPC have an extra layer of security compared to domains that use public endpoints.
- While public domains are accessible from any internet-connected device, VPC domains require some form of VPN or proxy.
- Compared to public domains, VPC domains display less information in the console. Specifically, the **Cluster health** tab does not include shard information, and the **Indices** tab isn't present.
- The domain endpoints take different forms (<https://search-domain-name> vs. <https://vpc-domain-name>).
- You can't apply IP-based access policies to domains that reside within a VPC because security groups already enforce IP-based access policies.

Limitations

Operating an OpenSearch Service domain within a VPC has the following limitations:

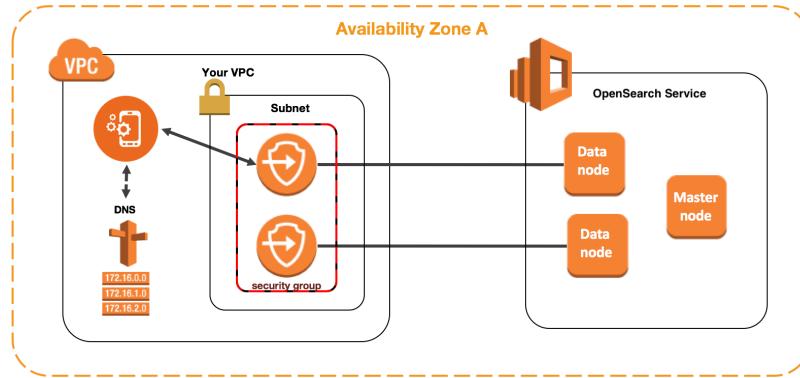
- If you launch a new domain within a VPC, you can't later switch it to use a public endpoint. The reverse is also true: If you create a domain with a public endpoint, you can't later place it within a VPC. Instead, you must create a new domain and migrate your data.
- You can either launch your domain 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.
- 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, but you can change the subnets and security group settings.
- To access the default installation of OpenSearch Dashboards 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 or transit gateway. To learn more, see the section called "About access policies on VPC domains" (p. 39), the [Amazon VPC User Guide](#), and the section called "Controlling access to OpenSearch Dashboards" (p. 280).

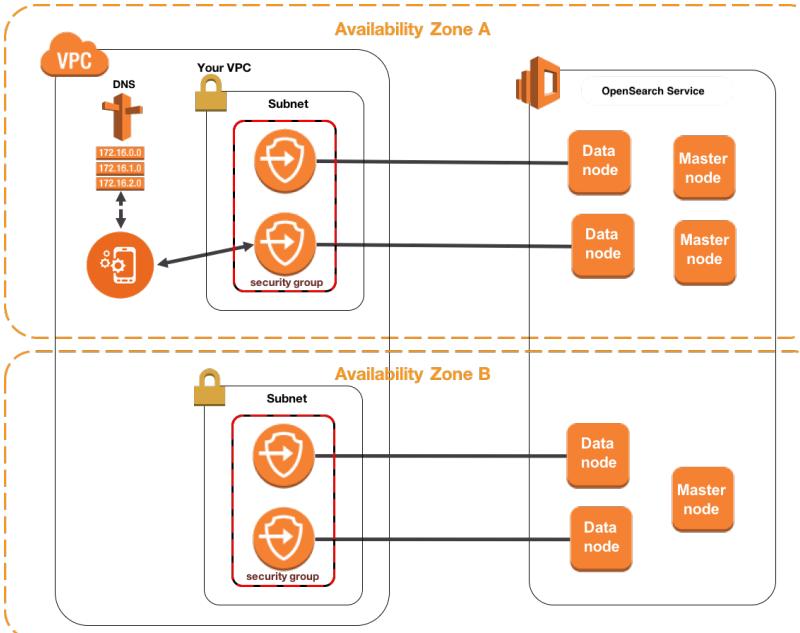
Architecture

To support VPCs, OpenSearch Service places an endpoint into one, two, or three subnets of your VPC. If you enable [multiple Availability Zones \(p. 34\)](#) for your domain, each subnet must be in a different Availability Zone in the same region. If you only use one Availability Zone, OpenSearch Service 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:



OpenSearch Service also places an *elastic network interface* (ENI) in the VPC for each of your data nodes. OpenSearch Service 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 OpenSearch Service 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 OpenSearch Service 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

OpenSearch Service doesn't support IPv6 addresses with a VPC. You can use a VPC that has IPv6 enabled, but the domain will use IPv4 addresses.

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 "Creating index snapshots" \(p. 42\)](#).

About access policies on VPC domains

Placing your OpenSearch Service 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 \(p. 130\)](#). 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, transit gateway, managed network, or proxy server. For details on the various forms it can take, see [Examples for VPC](#) in the *Amazon VPC User Guide*. For a development-focused example, see [the section called "Testing VPC domains" \(p. 40\)](#).

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 OpenSearch Service domain. Rather, it means that if a request reaches the OpenSearch Service domain and the associated security groups permit it, the domain accepts the request. The only exception is if you're using fine-grained access control or an access policy that specifies IAM users or roles. In these

situations, 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 OpenSearch Service domains that reside within a VPC. If you use public access, IP-based policies are still available.

Before you begin: prerequisites for VPC access

Before you can enable a connection between a VPC and your new OpenSearch Service 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 [Working with VPCs](#) in the *Amazon VPC User Guide*. If you already have a VPC, you can skip this step.

- **Reserve IP addresses**

OpenSearch Service 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 \(p. 41\)](#).

Testing VPC domains

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

1. For your domain's access policy, choose **Only use fine-grained access control**. 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 OpenSearch Service 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	<code>your-cidr-block</code>
HTTPS (443)	TCP (6)	443	<code>your-security-group-id</code>

The first rule lets you SSH into your EC2 instance. The second allows the EC2 instance to communicate with the OpenSearch Service 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-domain-name.region.es.amazonaws.com:443
```

This command creates an SSH tunnel that forwards requests to <https://localhost:9200> to your OpenSearch Service domain through the EC2 instance. Specifying port 9200 in the command simulates a local OpenSearch install, but use whichever port you'd like. OpenSearch Service only accepts connections over port 80 (HTTP) or 443 (HTTPS).

The command provides no feedback and runs indefinitely. To stop it, press **Ctrl + C**.

7. Navigate to https://localhost:9200/_dashboards/ 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.

Reserving IP addresses in a VPC subnet

OpenSearch Service 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 \(p. 34\)](#)). Each network interface is associated with an IP address. Before you create your OpenSearch Service domain, you must have a sufficient number of IP addresses available in each subnet to accommodate the network interfaces.

Here's the basic formula: The number of IP addresses that OpenSearch Service reserves in each subnet is three times the number of data nodes, divided by the number of Availability Zones.

Examples

- If a domain has nine data nodes across three Availability Zones, the IP count per subnet is $9 * 3 / 3 = 9$.
- If a domain has eight data nodes across two Availability Zones, the IP count per subnet is $8 * 3 / 2 = 12$.
- If a domain has six data nodes in one Availability Zone, the IP count per subnet is $6 * 3 / 1 = 18$.

When you create the domain, OpenSearch Service reserves the IP addresses, uses some for the domain, and reserves the rest for [blue/green deployments \(p. 22\)](#). You can see the network interfaces and their associated IP addresses in the **Network Interfaces** section of the Amazon EC2 console. The **Description** column shows which OpenSearch Service domain the network interface is associated with.

Tip

We recommend that you create dedicated subnets for the OpenSearch Service 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. OpenSearch Service requires a service-linked role to access your VPC, create the domain endpoint, and place network interfaces in a subnet of your VPC.

OpenSearch Service automatically creates the role when you use the OpenSearch Service console to create a domain within a VPC. For this automatic creation to succeed, you must have permissions for the `iam:CreateServiceLinkedRole` action. To learn more, see [Service-linked role permissions](#) in the *IAM User Guide*.

After OpenSearch Service creates the role, you can view it (`AWSServiceRoleForAmazonOpenSearchService`) using the IAM console.

For full information on this role's permissions and how to delete it, see [the section called "Service-linked roles" \(p. 188\)](#).

Creating index snapshots in Amazon OpenSearch Service

Snapshots in Amazon OpenSearch Service are backups of a cluster's indexes and state. *State* includes cluster settings, node information, index settings, and shard allocation.

OpenSearch Service snapshots come in the following forms:

- **Automated snapshots** are only for cluster recovery. You can use them to restore your domain in the event of red cluster status or data loss. For more information, see [Restoring snapshots \(p. 49\)](#) below. OpenSearch Service stores automated snapshots in a preconfigured Amazon S3 bucket at no additional charge.
- **Manual snapshots** are for cluster recovery *or* for moving data from one cluster to another. 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 OpenSearch cluster, you can use that snapshot to migrate to an OpenSearch Service domain. For more information, see [Migrating to Amazon OpenSearch Service \(p. 415\)](#).

All OpenSearch Service domains take automated snapshots, but the frequency differs in the following ways:

- For domains running OpenSearch or Elasticsearch 5.3 and later, OpenSearch Service takes hourly automated snapshots and retains up to 336 of them for 14 days. Hourly snapshots are less disruptive because of their incremental nature. They also provide a more recent recovery point in case of domain problems.
- For domains running Elasticsearch 5.1 and earlier, OpenSearch Service takes daily automated snapshots during the hour you specify, retains up to 14 of them, and doesn't retain any snapshot data for more than 30 days.

If your cluster enters red status, all automated snapshots fail while the cluster status persists. If you don't correct the problem within two weeks, you can permanently lose the data in your cluster. For troubleshooting steps, see [the section called "Red cluster status" \(p. 437\)](#).

Topics

- [Prerequisites \(p. 43\)](#)
- [Registering a manual snapshot repository \(p. 45\)](#)

- [Taking manual snapshots \(p. 48\)](#)
- [Restoring snapshots \(p. 49\)](#)
- [Deleting manual snapshots \(p. 51\)](#)
- [Automating snapshots with Index State Management \(p. 51\)](#)
- [Using Curator for snapshots \(p. 51\)](#)

Prerequisites

To create snapshots manually, you need to work with IAM and Amazon S3. Make sure you meet the following prerequisites before you attempt to take a snapshot:

Prerequisite	Description
S3 bucket	<p>Create an S3 bucket to store manual snapshots for your OpenSearch Service domain. For instructions, see Create a Bucket in the <i>Amazon Simple Storage Service User Guide</i>.</p> <p>Remember the name of the bucket to use it in the following places:</p> <ul style="list-style-type: none"> • The Resource statement of the IAM policy attached to your IAM role • The Python client used to register a snapshot repository (if you use this method) <p>Important Do not apply an S3 Glacier lifecycle rule to this bucket. Manual snapshots don't support the S3 Glacier storage class.</p>
IAM role	<p>Create an IAM role to delegate permissions to OpenSearch Service. For instructions, see Creating an IAM role (console) in the <i>IAM User Guide</i>. The rest of this chapter refers to this role as TheSnapshotRole.</p> <p>Attach an IAM policy</p> <p>Attach the following policy to TheSnapshotRole to allow access to the S3 bucket:</p> <pre>{ "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": ["arn:aws:s3:::s3-bucket-name/*"] }] }</pre>

Prerequisite	Description
	<p>}</p> <p>For instructions to attach a policy to a role, see Adding IAM Identity Permissions in the <i>IAM User Guide</i>.</p> <p>Edit the trust relationship</p> <p>Edit the trust relationship of TheSnapshotRole to specify OpenSearch Service in the Principal statement as shown in the following example:</p> <pre>{ "Version": "2012-10-17", "Statement": [{ "Sid": "", "Effect": "Allow", "Principal": { "Service": "opensearchservice.amazonaws.com" }, "Action": "sts:AssumeRole" }] }</pre> <p>We recommend that you use the aws:SourceAccount and aws:SourceArn condition keys to protect yourself against the confused deputy problem. The source account is the owner of the domain and the source ARN is the ARN of the domain. Your domain must be on service software R20211203 or later in order to add these condition keys.</p> <p>For example, you could add the following condition block to the trust policy:</p> <pre>"Condition": { "StringEquals": { "aws:SourceAccount": "account-id" }, "ArnLike": { "aws:SourceArn": "arn:aws:es:region:account-id:domain/domain-name" } }</pre> <p>For instructions to edit the trust relationship, see Modifying a role trust policy in the <i>IAM User Guide</i>.</p>

Prerequisite	Description
Permissions	<p>In order to register the snapshot repository, you need to be able to pass TheSnapshotRole to OpenSearch Service. You also need access to the es:ESHttpPut action. To grant both of these permissions, attach the following policy to the IAM user or role whose credentials are being used to sign the request:</p> <pre>{ "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/domain-name/*" }] }</pre> <p>If your user or role doesn't have iam:PassRole permissions to pass TheSnapshotRole you might encounter the following common error when you try to register a repository in the next step:</p> <pre>\$ 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"}</pre>

Registering a manual snapshot repository

You need to register a snapshot repository with OpenSearch 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 “Prerequisites” \(p. 43\)](#).

Step 1: Map the snapshot role in OpenSearch Dashboards (if using fine-grained access control)

Fine-grained access control introduces an additional step when registering a repository. Even if you use HTTP basic authentication for all other purposes, you need to map the manage_snapshots role to your IAM user or role that has iam:PassRole permissions to pass TheSnapshotRole.

1. Navigate to the OpenSearch Dashboards plugin for your OpenSearch Service domain. You can find the Dashboards endpoint on your domain dashboard on the OpenSearch Service console.
2. From the main menu choose **Security, Roles**, and select the **manage_snapshots** role.
3. Choose **Mapped users, Manage mapping**.
4. Add the ARN of the user or role that has permissions to pass TheSnapshotRole. Put user ARNs under **Users** and role ARNs under **Backend roles**.

```
arn:aws:iam::123456789123:user/user-name
```

```
arn:aws:iam::123456789123:role/role-name
```

5. Select **Map** and confirm the user or role shows up under **Mapped users**.

Step 2: Register a repository

To register a snapshot repository, send a PUT request to the OpenSearch Service domain endpoint. 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 \(p. 191\)](#) to register the snapshot repository.

The request takes the following format:

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

Note

Repository names cannot start with "cs-".

If your domain resides within a virtual private cloud (VPC), your computer must be connected to the VPC 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 OpenSearch Service domain, navigate to [https://*your-vpc-domain.region.es.amazonaws.com*](https://<i>your-vpc-domain.region.es.amazonaws.com) in a web browser and verify that you receive the default JSON response.

Encrypting snapshot repositories

You currently can't use AWS Key Management Service (KMS) keys to encrypt manual snapshots, but you can protect them using server-side encryption (SSE).

To enable SSE with S3-managed keys for the bucket you use as a snapshot repository, add "server_side_encryption": true to the "settings" block of the PUT request. For more information, see [Protecting data using server-side encryption with Amazon S3-managed encryption keys in the Amazon Simple Storage Service User Guide](#).

Alternatively, you can use AWS KMS keys for server-side encryption on the S3 bucket that you use as a snapshot repository. If you use this approach, make sure to provide TheSnapshotRole permission to the AWS KMS key used to encrypt the S3 bucket. For more information, see [Key policies in AWS KMS](#).

Migrating data to a different domain

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

Consider the following guidelines when migrating to a new domain or registering the same repository with multiple domains for another reason:

- When registering the repository on the new domain, add "readonly": true to the "settings" block of the PUT request. This setting prevents you from accidentally overwriting data from the old domain.
- If you're migrating data to a domain in a different region, (for example, from an old domain and bucket located in us-east-2 to a new domain in us-west-2), you might see this 500 error when sending the PUT request:

The bucket is in this region: us-east-2. Please use this region to retry the request.

If you encounter this error, try replacing "region": "us-east-2" with "endpoint": "s3.amazonaws.com" in the PUT statement and retry the request.

Using the sample Python client

The Python client is easier to automate than a simple HTTP request and has better reusability. If you choose to use this method to register a snapshot repository, save the following sample Python code as a Python file, such as `register-repo.py`. The client requires the [AWS SDK for Python \(Boto3\)](#), `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 \(p. 192\)](#).

Update the following variables in the sample code: 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 OpenSearch API endpoint
url = host + path

payload = {
    "type": "s3",
    "settings": {
        "bucket": "s3-bucket-name",
        "region": "us-west-1",
        "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-name/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 indexes except Dashboards and fine-grained access control)  
#  
# path = '_snapshot/my-snapshot-repo-name/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)  
#  
# print(r.text)  
#  
# # Restore snapshot (one index)  
#  
# path = '_snapshot/my-snapshot-repo-name/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 don't 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 OpenSearch initiated the snapshot. Depending on the size of your snapshot thread pool, different shards might be included in the snapshot at slightly different times.

Snapshot storage and performance

OpenSearch snapshots are incremental, meaning they only store data that changed since the last successful snapshot. This incremental nature means 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. For example, daily snapshots can take 20-30 minutes to complete, whereas hourly snapshots might complete within a few minutes. Some OpenSearch users take snapshots as often as every hour.

Take a snapshot

You specify the following information when you create a snapshot:

- The name of your snapshot repository
- A name for the snapshot

The examples in this chapter use `curl`, a common HTTP client, for convenience and brevity. However, if your access policies specify IAM users or roles, 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 take a manual snapshot, perform the following steps:

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

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

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

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

To include or exclude certain indexes and specify other settings, add a request body. For the request structure, see [Take snapshots](#) in the OpenSearch documentation.

Note

The time required to take a snapshot increases with the size of the OpenSearch Service domain. Long-running snapshot operations sometimes encounter the following error: 504 GATEWAY_TIMEOUT. You can typically ignore these errors and wait for the operation to complete successfully. Run the following command to verify the state of all snapshots of your domain:

```
curl -XGET 'domain-endpoint/_snapshot/repository-name/_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, perform the following steps:

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

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

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

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

Note

Most automated snapshots are stored in the cs-automated repository. If your domain encrypts data at rest, they're stored in the cs-automated-enc repository. If you don't see the manual snapshot repository you're looking for, make sure you [registered it \(p. 45\)](#) to the domain.

2. (Optional) Delete or rename one or more indexes in the OpenSearch Service domain if you have naming conflicts between indexes on the cluster and indexes in the snapshot. You can't restore a snapshot of your indexes to an OpenSearch cluster that already contains indexes with the same names.

You have the following options if you have index naming conflicts:

- Delete the indexes on the existing OpenSearch Service domain and then restore the snapshot.
- [Rename the indexes as you restore them from the snapshot \(p. 443\)](#) and reindex them later.
- Restore the snapshot to a different OpenSearch Service domain (only possible with manual snapshots).

The following command deletes all existing indexes in a domain:

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

However, if you don't plan to restore all indexes, you can just delete one:

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

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

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

Due to special permissions on the OpenSearch Dashboards and fine-grained access control indexes, attempts to restore all indexes might fail, especially if you try to restore from an automated snapshot. The following example restores just one index, my-index, from 2020-snapshot in the cs-automated snapshot repository:

```
curl -XPOST 'domain-endpoint/_snapshot/cs-automated/2020-snapshot/_restore' -d  
'{"indices": "my-index"}' -H 'Content-Type: application/json'
```

Alternately, you might want to restore all indexes *except* the Dashboards and fine-grained access control indexes:

```
curl -XPOST 'domain-endpoint/_snapshot/cs-automated/2020-snapshot/_restore' -d  
'{"indices": "-.kibana*,-.opendistro*"}' -H 'Content-Type: application/json'
```

Note

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

Deleting manual snapshots

To delete a manual snapshot, run the following command:

```
DELETE _snapshot/repository-name/snapshot-name
```

Automating snapshots with Index State Management

You can use the Index State Management (ISM) [snapshot](#) operation to automatically trigger snapshots of indexes based on changes in their age, size, or number of documents. For an example ISM policy using the snapshot operation, see [Sample Policies \(p. 306\)](#).

Using Curator for snapshots

If ISM doesn't work for index and snapshot management, you can use Curator instead. It offers advanced filtering functionality that can help simplify management tasks on complex clusters. Use [pip](#) to install Curator:

```
pip install elasticsearch-curator
```

You can use Curator as a command line interface (CLI) or Python API. If you use the Python API, you must use version 7.13.4 or earlier of the legacy [elasticsearch-py](#) client. It doesn't support the opensearch-py client.

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
```

Upgrading Amazon OpenSearch Service domains

Note

OpenSearch and Elasticsearch version upgrades differ from service software updates. For information on updating the service software for your OpenSearch Service domain, see [the section called "Service software updates" \(p. 29\)](#).

Amazon OpenSearch Service offers in-place upgrades for domains that run OpenSearch 1.0 or later, or Elasticsearch 5.1 or later. If you use services like Amazon Kinesis Data Firehose or Amazon CloudWatch Logs to stream data to OpenSearch Service, check that these services support the newer version of OpenSearch before migrating.

Supported upgrade paths

Currently, OpenSearch Service supports the following upgrade paths:

From version	To version
OpenSearch 1.3	<p>OpenSearch 2.3</p> <p>Important If your domain contains UltraWarm or cold indexes that were originally created in Elasticsearch 6.8, those indexes are not compatible with OpenSearch 2.3. Before you upgrade to version 2.3, you must migrate incompatible indexes to hot storage, reindex the data, and then migrate them back to warm or cold storage. Alternately, you can delete the indexes if you no longer need them. If you accidentally upgrade your domain to version 2.3 without performing these steps first, you won't be able to migrate the incompatible indexes out of their current storage tier. Your only option is to delete them.</p>
OpenSearch 1.x	OpenSearch 1.x
Elasticsearch 7.x	<p>Elasticsearch 7.x or OpenSearch 1.x</p> <p>Important OpenSearch 1.x introduces numerous breaking changes. For details, see Amazon OpenSearch Service rename (p. 5).</p>
Elasticsearch 6.8	<p>Elasticsearch 7.x or OpenSearch 1.x</p> <p>Important Elasticsearch 7.0 and OpenSearch 1.0 include numerous breaking changes. Before initiating an in-place upgrade, we recommend taking a manual snapshot (p. 42) of the 6.x domain, restoring it on a test 7.x or OpenSearch 1.x domain, and using that test domain to identify potential upgrade issues. For breaking changes in OpenSearch 1.0, see Amazon OpenSearch Service rename (p. 5). Like Elasticsearch 6.x, indexes 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). For new indexes, self-hosted Elasticsearch 7.x and OpenSearch 1.x have a default shard count of one. OpenSearch Service domains on Elasticsearch 7.x and later retain the previous default of five.</p>
Elasticsearch 6.x	Elasticsearch 6.x
Elasticsearch 5.6	<p>Elasticsearch 6.x</p> <p>Important Indexes created in version 6.x no longer support multiple mapping types. Indexes 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. To minimize downtime during the upgrade from Elasticsearch 5.6 to 6.x, OpenSearch Service reindexes the .kibana index to .kibana-6, deletes .kibana, creates an alias named .kibana, and maps the new index to the new alias.</p>

From version	To version
Elasticsearch 5.x	Elasticsearch 5.x

The upgrade process consists of three steps:

1. **Pre-upgrade checks** – OpenSearch Service checks for issues that can block an upgrade and doesn't proceed to the next step unless these checks succeed.
2. **Snapshot** – OpenSearch Service takes a snapshot of the OpenSearch or Elasticsearch cluster and doesn't proceed to the next step unless the snapshot succeeds. If the upgrade fails, OpenSearch Service uses this snapshot to restore the cluster to its original state. For more information see [the section called "Can't downgrade after upgrade" \(p. 445\)](#).
3. **Upgrade** – OpenSearch Service starts the upgrade, which can take from 15 minutes to several hours to complete. OpenSearch Dashboards might be unavailable during some or all of the upgrade.

Starting an upgrade (console)

The upgrade process is irreversible and can't be paused or cancelled. 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, OpenSearch 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 OpenSearch or Elasticsearch

1. Take a manual snapshot (p. 42) 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 OpenSearch version.
2. Go to <https://aws.amazon.com> and choose **Sign In to the Console**.
3. Under **Analytics**, choose **Amazon OpenSearch Service**.
4. In the navigation pane, under **Domains**, choose the domain that you want to upgrade.
5. Choose **Actions** and **Upgrade**.
6. Select the version to upgrade to. If you're upgrading to an OpenSearch version, the **Enable compatibility mode** option appears. If you enable this setting, OpenSearch reports its version as 7.10 to allow Elasticsearch OSS clients and plugins like Logstash to continue working with Amazon OpenSearch Service. You can disable this setting later
7. Choose **Upgrade**.
8. Check the **Status** on the domain dashboard to monitor the status of the upgrade.

Starting an upgrade (CLI)

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

- `get-compatible-versions` (`GetCompatibleVersions`)
- `upgrade-domain` (`UpgradeDomain`)
- `get-upgrade-status` (`GetUpgradeStatus`)
- `get-upgrade-history` (`GetUpgradeHistory`)

For more information, see the [AWS CLI command reference](#) and [Amazon OpenSearch Service API Reference](#).

Starting an upgrade (SDK)

This sample uses the [OpenSearchService](#) low-level Python client from the AWS SDK for Python (Boto) to check if a domain is eligible for upgrade to a specific version, upgrades it, and continuously checks the upgrade status.

```
import boto3
from botocore.config import Config
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default Region.

DOMAIN_NAME = '' # The name of the domain to upgrade
TARGET_VERSION = '' # The version you want to upgrade the domain to. For example,
                    OpenSearch_1.1

my_config = Config(
    # Optionally lets you specify a Region other than your default.
    region_name='us-east-1'
)
client = boto3.client('opensearch', config=my_config)

def check_versions():
    """Determine whether domain is eligible for upgrade"""
    response = client.get_compatible_versions(
        DomainName=DOMAIN_NAME
    )
    compatible_versions = response['CompatibleVersions']
    for i in range(len(compatible_versions)):
        if TARGET_VERSION in compatible_versions[i]["TargetVersions"]:
            print('Domain is eligible for upgrade to ' + TARGET_VERSION)
            upgrade_domain()
            print(response)
        else:
            print('Domain not eligible for upgrade to ' + TARGET_VERSION)

def upgrade_domain():
    """Upgrades the domain"""
    response = client.upgrade_domain(
        DomainName=DOMAIN_NAME,
        TargetVersion=TARGET_VERSION
    )
    print('Upgrading domain to ' + TARGET_VERSION + '...' + response)
    time.sleep(5)
    wait_for_upgrade()

def wait_for_upgrade():
    """Get the status of the upgrade"""
    response = client.get_upgrade_status(
        DomainName=DOMAIN_NAME
    )
    if (response['UpgradeStep']) == 'UPGRADE' and (response['StepStatus']) == 'SUCCEEDED':
        print('Domain successfully upgraded to ' + TARGET_VERSION)
    elif (response['StepStatus']) == 'FAILED':
        print('Upgrade failed. Please try again.')
    elif (response['StepStatus']) == 'SUCCEEDED_WITH_ISSUES':
```

```

        print('Upgrade succeeded with issues')
    elif (response['StepStatus']) == 'IN_PROGRESS':
        time.sleep(30)
        wait_for_upgrade()

def main():
    check_versions()

if __name__ == "__main__":
    main()

```

Troubleshooting validation failures

When you initiate an OpenSearch or Elasticsearch version upgrade, OpenSearch Service first performs a series of validation checks to ensure that your domain is eligible for an upgrade. If any of these checks fail, you receive a notification containing the specific issues that you must fix before upgrading your domain. For a list of potential issues and steps to resolve them, see [the section called “Troubleshooting validation errors” \(p. 26\)](#).

Troubleshooting an upgrade

In-place 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	OpenSearch, as well as 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, OpenSearch Service doesn't allow you to upgrade. See the section called “Exceeded maximum shard limit” (p. 442) 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 indexes in the cluster is red. For troubleshooting steps, see the section called “Red cluster status” (p. 437) .
High error rate	The cluster is returning a large number of 5xx 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	<i>Split brain</i> means that your 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. 358) . For help recovering from split brain, contact AWS Support .
Master node not found	OpenSearch Service can't find the cluster's master node. If your domain uses multi-AZ (p. 34) , an Availability Zone failure might have caused the cluster to lose quorum and be unable to elect a new master node (p. 358) . 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.

Issue	Description
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. 129) .
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 OpenSearch Service 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 indexes	One or more indexes is incompatible with the target version. This problem can occur if you migrated the indexes from an older version of OpenSearch or Elasticsearch. Reindex the indexes 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.
OpenSearch Dashboards alias problem	.kibana is already configured as an alias and maps to an incompatible index, likely one from an earlier version of OpenSearch Dashboards. Reindex, and try again.
Red Dashboards status	OpenSearch Dashboards status is red. Try using Dashboards when the upgrade completes. If the red status persists, resolve it manually, and try again.
Cross-cluster compatibility	You can only upgrade if cross-cluster compatibility is maintained between the source and destination domains after the upgrade. During the upgrade process, any incompatible connections are identified. To proceed, either upgrade the remote domain or delete the incompatible connections. Note that if replication is active on the domain, you can't resume it once you delete the connection.
Other OpenSearch Service service issue	Issues with OpenSearch Service 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 .

Using a snapshot to migrate data

In-place upgrades are the easier, faster, and more reliable way to upgrade a domain to a later OpenSearch or 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 OpenSearch or Elasticsearch version. For more information about taking and restoring snapshots, see [the section called "Creating index snapshots" \(p. 42\)](#).

From version	To version	Migration process
Elasticsearch 6.x or 7.x	OpenSearch 1.x	<ol style="list-style-type: none"> Review breaking changes for OpenSearch 1.0 to see if you need to make adjustments to your indexes or applications. For other considerations, see the table in the section called "Upgrading Amazon OpenSearch Service domains" (p. 51). Create a manual snapshot of the Elasticsearch 7.x or 6.x domain. Create an OpenSearch 1.x domain. Restore the snapshot from the Elasticsearch domain to the OpenSearch domain. During the operation, you might need to restore the .kibana index under a new name: <pre>POST _snapshot/<repository-name>/<snapshot-name>/_restore { "indices": "*", "ignore_unavailable": true, "rename_pattern": ".kibana", "rename_replacement": ".backup-kibana" }</pre> <p>Then you can reindex .backup-kibana on the new domain and alias it to .kibana.</p> <ol style="list-style-type: none"> If you no longer need your original domain, delete it. Otherwise, you continue to incur charges for the domain.
Elasticsearch 6.x	Elasticsearch 7.x	<ol style="list-style-type: none"> Review breaking changes for 7.0 to see if you need to make adjustments to your indexes or applications. For other considerations, see the table in the section called "Upgrading Amazon OpenSearch Service domains" (p. 51). Create a manual snapshot of the 6.x domain. Create a 7.x domain. 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: <pre>POST _snapshot/<repository-name>/<snapshot-name>/_restore { "indices": "*", "ignore_unavailable": true, "rename_pattern": ".kibana", "rename_replacement": ".backup-kibana" }</pre> <p>Then you can reindex .backup-kibana on the new domain and alias it to .kibana.</p> <ol style="list-style-type: none"> If you no longer need your original domain, delete it. Otherwise, you continue to incur charges for the domain.
Elasticsearch 6.x	Elasticsearch 6.8	<ol style="list-style-type: none"> Create a manual snapshot of the 6.x domain. Create a 6.8 domain.

From version	To version	Migration process
		3. Restore the snapshot from the original domain to the 6.8 domain. 4. If you no longer need your original domain, delete it. Otherwise, you continue to incur charges for the domain.
Elasticsearch 5.x	Elasticsearch 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 Amazon OpenSearch Service domains" (p. 51) . 2. Create a manual snapshot of the 5.x domain. 3. Create a 6.x domain. 4. Restore the snapshot from the original domain to the 6.x domain. 5. If you no longer need your 5.x domain, delete it. Otherwise, you continue to incur charges for the domain.
Elasticsearch 5.x	Elasticsearch 5.6	1. Create a manual snapshot of the 5.x domain. 2. Create a 5.6 domain. 3. Restore the snapshot from the original domain to the 5.6 domain. 4. If you no longer need your original domain, delete it. Otherwise, you continue to incur charges for the domain.
Elasticsearch 2.3	Elasticsearch 6.x	<p>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 indexes in the new domain.</p> <p>Alternately, you can follow the 2.3 to 5.x steps in this table, perform <code>_reindex</code> operations in the new 5.x domain to convert your 2.3 indexes to 5.x indexes, and then follow the 5.x to 6.x steps.</p>
Elasticsearch 2.3	Elasticsearch 5.x	1. Review breaking changes for 5.0 to see if you need to make adjustments to your indexes or applications. 2. Create a manual snapshot of the 2.3 domain. 3. Create a 5.x domain. 4. Restore the snapshot from the 2.3 domain to the 5.x domain. 5. If you no longer need your 2.3 domain, delete it. Otherwise, you continue to incur charges for the domain.
Elasticsearch 1.5	Elasticsearch 5.x	<p>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 indexes in the new domain.</p> <p>Important 1.5 snapshots <i>are</i> compatible with 2.3, but OpenSearch Service 2.3 domains do not support the <code>_reindex</code> operation. Because you cannot reindex them, indexes that originated in a 1.5 domain still fail to restore from 2.3 snapshots to 5.x domains.</p>

From version	To version	Migration process
Elasticsearch 1.5	Elasticsearch 2.3	<ol style="list-style-type: none"> 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. <ol style="list-style-type: none"> 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.

Creating a custom endpoint for Amazon OpenSearch Service

Creating a custom endpoint for your Amazon OpenSearch Service domain makes it easier for you to refer to your OpenSearch and OpenSearch Dashboards URLs. You can include your company's branding or just use a shorter, easier-to-remember endpoint than the standard one.

If you ever need to switch to a new domain, just update your DNS to point to the new URL and continue using the same endpoint as before.

You secure custom endpoints by either generating a certificate in AWS Certificate Manager (ACM) or importing one of your own.

Custom endpoints for new domains

You can enable a custom endpoint for a new OpenSearch Service domain using the OpenSearch Service console, AWS CLI, or configuration API.

To customize your endpoint (console)

1. From the OpenSearch Service console, choose **Create domain** and provide a name for the domain.
2. Under **Custom endpoint**, select **Enable custom endpoint**.
3. For **Custom hostname**, enter your preferred custom endpoint hostname. The hostname should be a fully qualified domain name (FQDN), such as `www.yourdomain.com` or `example.yourdomain.com`.

Note

If you don't have a [wildcard certificate](#) you must obtain a new certificate for your custom endpoint's subdomains.

4. For **AWS certificate**, choose the SSL certificate to use for your domain. If no certificates are available, you can import one into ACM or use ACM to provision one. For more information, see [Issuing and Managing Certificates](#) in the *AWS Certificate Manager User Guide*.

Note

The certificate must have the custom endpoint name and be in the same account as your OpenSearch Service domain. The certificate status should be **ISSUED**.

- Follow the rest of the steps to create your domain and choose **Create**.
- Select the domain when it's finished processing to view your custom endpoint.

To use the CLI or configuration API, use the `CreateDomain` and `UpdateDomainConfig` operations. For more information, see the [AWS CLI Command Reference](#) and [Amazon OpenSearch Service API Reference](#).

Custom endpoints for existing domains

To add a custom endpoint to an existing OpenSearch Service domain, choose **Edit** and perform steps 2–4 above. Editing a domain's custom endpoint triggers a [blue/green deployment](#) (p. 22).

Next steps

After you enable a custom endpoint for your OpenSearch Service domain, you must create a CNAME mapping in Amazon Route 53 (or your preferred DNS service provider) to route traffic to the custom endpoint and its subdomains. Create the CNAME record pointing the custom endpoint to the auto-generated domain endpoint. The custom endpoint hostname is the *name* of the CNAME record, and the domain endpoint hostname is the *value* of the CNAME record. Without this mapping, your custom endpoint won't work. For steps to create this mapping in Route 53, see [Configuring DNS routing for a new domain](#) and [Creating a hosted zone for a subdomain](#). For other providers, consult their documentation.

If you use [SAML authentication for OpenSearch Dashboards](#) (p. 169), you must update your IdP with the new SSO URL.

Auto-Tune for Amazon OpenSearch Service

Auto-Tune in Amazon OpenSearch Service uses performance and usage metrics from your OpenSearch cluster to suggest memory-related configuration changes, including queue and cache sizes and Java virtual machine (JVM) settings on your nodes. These optional changes improve cluster speed and stability.

Some changes deploy immediately, while others require you to schedule a maintenance window. You can revert to the default OpenSearch Service settings at any time.

As Auto-Tune gathers and analyzes performance metrics for your domain, you can view its recommendations in the OpenSearch Service console on the **Notifications** page.

Auto-Tune is available in commercial AWS Regions on domains running any OpenSearch version, or Elasticsearch 6.7 or later, with a [supported instance type](#) (p. 365).

Types of changes

Auto-Tune has two broad categories of changes:

- Nondisruptive changes that it applies as the cluster runs
- Changes that require a [blue/green deployment](#) (p. 22)

Based on your domain's performance metrics, Auto-Tune can suggest adjustments to the following settings:

Change type	Category	Description
JVM heap size	Blue/green	<p>By default, OpenSearch Service uses 50% of an instance's RAM for the JVM heap, up to a heap size of 32 GiB.</p> <p>Increasing this percentage gives OpenSearch more memory, but leaves less for the operating system and other processes. Larger values can decrease the number of garbage collection pauses, but increase the length of those pauses.</p>
JVM young generation settings	Blue/green	JVM "young generation" settings affect the frequency of minor garbage collections. More frequent minor collections can decrease the number of major collections and pauses.
Queue size	Nondisruptive	<p>By default, the search queue size is 1000 and the write queue size is 10000. Auto-Tune automatically scales the search and write queues if additional heap is available to handle requests.</p>
Cache size	Nondisruptive	<p>The <i>field cache</i> monitors on-heap data structures, so it's important to monitor the cache's use. Auto-Tune scales the field data cache size to avoid out of memory and circuit breaker issues.</p> <p>The <i>shard request cache</i> is managed at the node level and has a default maximum size of 1% of the heap. Auto-Tune scales the shard request cache size to accept more search and index requests than what the configured cluster can handle.</p>
Request size	Nondisruptive	<p>By default, when the aggregated size of in-flight requests surpasses 10% of total JVM (2% for t2 instance types and 1% for t3.small), OpenSearch throttles all new _search and _bulk requests until the existing requests complete.</p> <p>Auto-Tune automatically tunes this threshold, typically between 5-15%, based on the amount of JVM that is currently occupied on the system. For example, if JVM memory pressure is high, Auto-Tune might reduce the threshold to 5%, at which point you might see more rejections until the cluster stabilizes and the threshold increases.</p>

If you enable Auto-Tune without setting a maintenance window, Auto-Tune only applies nondisruptive changes. The performance benefits over time are generally smaller, but you avoid the overhead associated with blue/green deployments.

For guidance on configuring maintenance windows, see [the section called “Scheduling changes” \(p. 62\)](#).

Enabling or disabling Auto-Tune

OpenSearch Service enables Auto-Tune by default on new domains. To enable or disable Auto-Tune on existing domains, we recommend using the console, which greatly simplifies the process. In the console, choose your domain and go to the **Auto-Tune** tab, then choose **Edit**. Enabling Auto-Tune doesn't cause a blue/green deployment.

AWS CLI

To use the [AWS CLI](#), configure the auto-tune-options parameters. The following sample command enables Auto-Tune on an existing domain with a maintenance schedule that repeats every day at 12:00pm UTC:

```
aws opensearch update-domain-config \
--domain-name mylogs \
--auto-tune-options '[{"DesiredState": "ENABLED", "MaintenanceSchedules": [{"StartAt": "2021-12-19", "Duration": {"Value": 2, "Unit": "HOURS"}, "CronExpressionForRecurrence": "cron(0 12 * * ? *)"}]]'
```

Configuration API

To use the [configuration API](#), configure the AutoTuneOptions settings:

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/domain-name/config
{
    "AutoTuneOptions": {
        "DesiredState": "ENABLED",
        "MaintenanceSchedules": [
            {
                "StartAt": 4104152288000,
                "Duration": {
                    "Value": 2,
                    "Unit": "HOURS"
                },
                "CronExpressionForRecurrence": "cron(0 12 * * ? *)"
            }
        ]
    }
}
```

CloudFormation

You currently can't enable or disable Auto-Tune using AWS CloudFormation.

Scheduling changes

To apply changes that require a blue/green deployment, you schedule a maintenance window for your domain—for example, between 6:00 and 9:00 AM on a Friday morning. We recommend scheduling maintenance windows for low-traffic times.

- To review all changes before deploying them, wait for Auto-Tune to notify you of a suggested optimization. Then schedule a one-time maintenance window to deploy the changes.
- For a more automated experience, set a weekly maintenance window, such as every Saturday at 2:00 AM, or use a custom [cron expression \(p. 62\)](#) for more complex schedules.

To schedule changes in the console, choose your domain, go to the **Auto-Tune** tab, choose **Edit**, and then select **Add maintenance window**. This tab also shows your current maintenance window and whether Auto-Tune will make any changes during the next window.

Cron expressions

Cron expressions for Auto-Tune use the same six-field syntax as [Amazon CloudWatch Events](#):

```
minute hour day-of-month month day-of-week year
```

For example, the following expression translates to "every Tuesday and Friday at 1:15 AM from 2021 through 2024":

```
15 1 ? * 2,5 2021-2024
```

The following table includes valid values for each field.

Field	Valid values
Minute	0–59
Hour	0–23
Day of month	1–31
Month	1–12 or JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC
Day of week	1–7 or SUN, MON, TUE, WED, THU, FRI, SAT
Year	1970–2199

Day of month and day of week overlap, so you can specify one, but not both. You must mark the other as ?. For a full summary of wildcard options, see the [Amazon CloudWatch Events User Guide](#).

Tagging Amazon OpenSearch Service domains

Tags let you assign arbitrary information to an Amazon OpenSearch Service domain so you can categorize and filter on that information. A tag is a key-value pair that you define and associate with an OpenSearch Service domain. You can use these tags to track costs by grouping expenses for similarly tagged resources. AWS doesn't apply any semantic meaning to your tags. Tags are interpreted strictly as character strings. All tags have the following elements:

Tag Element	Description	Required
Tag key	The tag key is the name of the tag. Key must be unique to the OpenSearch Service domain to which they're attached. For a list of basic restrictions on tag keys and values, see User-Defined Tag Restrictions .	Yes
Tag value	The tag value is the string value of the tag. Tag values can be null and don't 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 .	No

Each OpenSearch Service domain has a tag set, which contains all the tags assigned to that OpenSearch Service domain. AWS doesn't automatically assign any tags to OpenSearch Service domains. A tag set can contain between 0 and 50 tags. If you add a tag to a domain with the same key as an existing tag, the new value overwrites the old value.

Tagging examples

You can use a key to define a category, and the 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 OpenSearch Service domain is assigned to the `Salix` project. You could also use tags to designate OpenSearch Service domains as being used for test or production by using a key such as `environment=test` or `environment=production`. Try to use a consistent set of tag keys to make it easier to track metadata that is associated with OpenSearch Service 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 OpenSearch Service 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 OpenSearch Service domains might take several minutes before they're available.

Working with tags (console)

The console is the simplest way to tag a domain.

To create a tag (console)

1. Go to <https://aws.amazon.com>, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Amazon OpenSearch Service**.
3. Select the domain you want to add tags to and go to the **Tags** tab.
4. Choose **Manage** and **Add new tag**.
5. Enter a tag key and an optional value.
6. Choose **Save**.

To delete a tag, follow the same steps and choose **Remove** on the **Manage tags** page.

For more information about using the console to work with tags, see [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 OpenSearch Service 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 opensearch add-tags --arn arn:aws:es:us-east-1:379931976431:domain/logs --tag-list  
Key=service,Value=OpenSearch Key=instances,Value=m3.2xlarge
```

You can remove tags from an OpenSearch Service 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 OpenSearch Service domain to which the tag is attached.
--tag-keys	Set of space-separated key-value pairs that you want to remove from the OpenSearch Service domain.

Example

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

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

You can view the existing tags for an OpenSearch Service domain with the **list-tags** command:

Syntax

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

Parameter	Description
--arn	Amazon Resource Name (ARN) for the OpenSearch Service domain to which the tags are attached.

Example

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

```
aws opensearch 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 OpenSearch Service API Reference](#), including the AddTags, ListTags, and RemoveTags operations. For more information about installing and using the AWS SDKs, see [AWS Software Development Kits](#).

Python

This example uses the [OpenSearchService](#) low-level Python client from the AWS SDK for Python (Boto) to add a tag to a domain, list the tag attached to the domain, and remove a tag from the domain. You must provide values for DOMAIN_ARN, TAG_KEY, and TAG_VALUE.

```
import boto3
from botocore.config import Config # import configuration

DOMAIN_ARN = '' # ARN for the domain. i.e "arn:aws:es:us-east-1:123456789012:domain/my-domain
```

```
TAG_KEY = '' # The name of the tag key. i.e 'Smileyface'  
TAG_VALUE = '' # The value assigned to the tag. i.e 'Practicetag'  
  
# defines the configurations parameters such as region  
  
my_config = Config(region_name='us-east-1')  
client = boto3.client('opensearch', config=my_config)  
  
# defines the client variable  
  
def addTags():  
    """Adds tags to the domain"""  
  
    response = client.add_tags(ARN=DOMAIN_ARN,  
                               TagList=[{'Key': TAG_KEY,  
                                         'Value': TAG_VALUE}])  
  
    print(response)  
  
def listTags():  
    """List tags that have been added to the domain"""  
  
    response = client.list_tags(ARN=DOMAIN_ARN)  
    print(response)  
  
def removeTags():  
    """Remove tags that have been added to the domain"""  
  
    response = client.remove_tags(ARN=DOMAIN_ARN, TagKeys=[TAG_KEY])  
  
    print('Tag removed')  
    return response
```

Monitoring Amazon OpenSearch Service domains

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon OpenSearch Service and your other AWS solutions. AWS provides the following tools to monitor your OpenSearch Service resources, report issues, and take automatic actions when appropriate:

Amazon CloudWatch

Amazon CloudWatch monitors your OpenSearch Service resources in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a metric reaches a certain threshold. For more information, see the [Amazon CloudWatch User Guide](#).

Amazon CloudWatch Logs

Amazon CloudWatch Logs lets you monitor, store, and access your OpenSearch log files. CloudWatch Logs monitors the information in log files and can notify you when certain thresholds are met. For more information, see the [Amazon CloudWatch Logs User Guide](#).

Amazon EventBridge

Amazon EventBridge delivers a near real-time stream of system events that describe changes in your OpenSearch Service domains. You can create rules that watch for certain events, and trigger automated actions in other AWS services when these events occur. For more information, see the [Amazon EventBridge User Guide](#).

AWS CloudTrail

AWS CloudTrail captures configuration API calls made to OpenSearch Service as events. It can deliver these events to an Amazon S3 bucket that you specify. Using this information, you can identify which users and accounts made requests, the source IP address from which the requests were made, and when the requests occurred. For more information, see the [AWS CloudTrail User Guide](#).

Topics

- [Monitoring OpenSearch cluster metrics with Amazon CloudWatch \(p. 67\)](#)
- [Monitoring OpenSearch logs with Amazon CloudWatch Logs \(p. 92\)](#)
- [Monitoring audit logs in Amazon OpenSearch Service \(p. 97\)](#)
- [Monitoring OpenSearch Service events with Amazon EventBridge \(p. 106\)](#)
- [Monitoring Amazon OpenSearch Service API calls with AWS CloudTrail \(p. 123\)](#)

Monitoring OpenSearch cluster metrics with Amazon CloudWatch

Amazon OpenSearch Service publishes data from your domains to Amazon CloudWatch. CloudWatch lets you retrieve statistics about those data points as an ordered set of time-series data, known as *metrics*. OpenSearch Service sends metrics to CloudWatch in 60-second intervals. If you use General Purpose or Magnetic EBS volumes, the EBS volume metrics update only every five minutes. For more information about Amazon CloudWatch, see the [Amazon CloudWatch User Guide](#).

The OpenSearch Service console displays a series of charts based on the raw data from CloudWatch. Depending on your needs, you might prefer to view cluster data in CloudWatch instead of the graphs in the console. The service archives metrics for two weeks before discarding them. The metrics are provided at no extra charge, but CloudWatch still charges for creating dashboards and alarms. For more information, see [Amazon CloudWatch pricing](#).

OpenSearch Service publishes the following metrics to CloudWatch:

- the section called “Cluster metrics” (p. 69)
- the section called “Dedicated master node metrics” (p. 73)
- the section called “EBS volume metrics” (p. 74)
- the section called “Instance metrics” (p. 75)
- the section called “UltraWarm metrics” (p. 81)
- the section called “Cold storage metrics” (p. 84)
- the section called “Alerting metrics” (p. 84)
- the section called “Anomaly detection metrics” (p. 85)
- the section called “Asynchronous search metrics” (p. 86)
- the section called “SQL metrics” (p. 87)
- the section called “k-NN metrics” (p. 88)
- the section called “Cross-cluster search metrics” (p. 90)
- the section called “Cross-cluster replication metrics” (p. 90)
- the section called “Learning to Rank metrics” (p. 91)
- the section called “Piped Processing Language metrics” (p. 91)

Viewing metrics in CloudWatch

CloudWatch metrics are grouped first by the service namespace, and then by the various dimension combinations within each namespace.

To view metrics using the CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **All metrics** and select the **AWS/ES** namespace.
3. Choose a dimension to view the corresponding metrics. Metrics for individual nodes are in the **ClientId**, **DomainName**, **NodeId** dimension. Cluster metrics are in the **Per-Domain**, **Per-Client Metrics** dimension. Some node metrics are aggregated at the cluster level and thus included in both dimensions. Shard metrics are in the **ClientId**, **DomainName**, **NodeId**, **ShardRole** dimension.

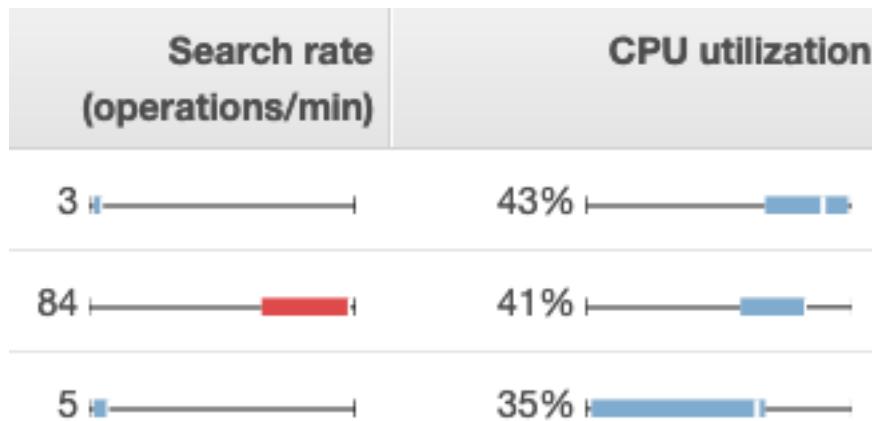
To view a list of metrics using the AWS CLI

Run the following command:

```
aws cloudwatch list-metrics --namespace "AWS/ES"
```

Interpreting health charts in OpenSearch Service

To view metrics in OpenSearch Service, use the **Cluster health** and **Instance health** tabs. The **Instance health** tab uses box charts to provide at-a-glance visibility into the health of each OpenSearch 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.

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” \(p. 22\)](#).

Cluster metrics

Amazon OpenSearch Service provides 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. Relevant statistics: Maximum
ClusterStatus.yellow	A value of 1 indicates that the primary shards for all indexes 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. 439) . 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. 437) . Relevant statistics: Maximum
Shards.active	The total number of active primary and replica shards. Relevant statistics: Maximum, Sum
Shards.unassigned	The number of shards that are not allocated to nodes in the cluster.

Metric	Description
	Relevant statistics: Maximum, Sum
Shards.delayedUnassigned	The number of shards whose node allocation has been delayed by the timeout settings. Relevant statistics: Maximum, Sum
Shards.activePrimary	The number of active primary shards. Relevant statistics: Maximum, Sum
Shards.initializing	The number of shards that are under initialization. Relevant statistics: Sum
Shards.relocating	The number of shards that are under relocation. Relevant statistics: Sum
Nodes	The number of nodes in the OpenSearch Service cluster, including dedicated master nodes and UltraWarm nodes. For more information, see the section called "Configuration changes" (p. 22) . Relevant statistics: Maximum
SearchableDocuments	The total number of searchable documents across all data nodes in the cluster. 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 OpenSearch only removes deleted documents from disk during segment merges. This metric increases after delete requests and decreases after segment merges. 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. Relevant statistics: Maximum, Average

Metric	Description
FreeStorageSpace	<p>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 least and most free space, respectively. This metric is also available for individual nodes. OpenSearch Service throws a <code>ClusterBlockException</code> when this metric reaches 0. To recover, you must either delete indexes, add larger instances, or add EBS-based storage to existing instances. To learn more, see the section called "Lack of available storage space" (p. 440).</p> <p>The OpenSearch Service console displays this value in GiB. The Amazon CloudWatch console displays it in MiB.</p> <p>Note FreeStorageSpace will always be lower than the values that the OpenSearch _cluster/stats and _cat/allocation APIs provide. OpenSearch Service reserves a percentage of the storage space on each instance for internal operations. For more information, see Calculating storage requirements (p. 353).</p> <p>Relevant statistics: Minimum, Maximum, Average, Sum</p>
ClusterUsedSpace	<p>The total used space for the cluster. You must leave the period at one minute to get an accurate value.</p> <p>The OpenSearch Service console displays this value in GiB. The Amazon CloudWatch console displays it in MiB.</p> <p>Relevant statistics: Minimum, Maximum</p>
ClusterIndexWritesBlocked	<p>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.</p> <p>Some common factors include the following: FreeStorageSpace is too low or JVMMemoryPressure is too high. To alleviate this issue, consider adding more disk space or scaling your cluster.</p> <p>Relevant statistics: Maximum</p>
JVMMemoryPressure	<p>The maximum percentage of the Java heap used for all data nodes in the cluster. OpenSearch Service 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. 361).</p> <p>Relevant statistics: Maximum</p> <p>Note The logic for this metric changed in service software R20220323. For more information, see the release notes (p. 450).</p>

Metric	Description
OldGenJVMMemoryPressure	The maximum percentage of the Java heap used for the "old generation" on all data nodes in the cluster. This metric is also available at the node level. 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. Relevant statistics: Minimum, Maximum
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 <i>Amazon EC2 Developer Guide</i> . This metric is available only for the T2 instance types. Relevant statistics: Minimum
OpenSearchDashboardsHealthyNodes (Previously KibanaHealthyNodes)	The HealthyNodes check for OpenSearch Dashboards. If the minimum, maximum, and average are all equal to 1, Dashboards is behaving normally. If you have 10 nodes with a maximum of 1, minimum of 0, and average of 0.7, this means 7 nodes (70%) are healthy and 3 nodes (30%) are unhealthy. Relevant statistics: Minimum, Maximum, Average
KibanaReportingFailedRequests	The Number of requests to generate OpenSearch Dashboards reports that failed due to server problems or feature limitations. Relevant statistics: Sum
KibanaReportingFailedUserRequests	The User number of requests to generate OpenSearch Dashboards reports that failed due to client issues. Relevant statistics: Sum
KibanaReportingRequestCount	The total number of requests to generate OpenSearch Dashboards reports. Relevant statistics: Sum
KibanaReportingSuccessCount	The number of successful requests to generate OpenSearch Dashboards reports. Relevant statistics: Sum
KMSKeyError	A value of 1 indicates that the AWS KMS 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. Relevant statistics: Minimum, Maximum

Metric	Description
KMSKeyInaccessible	<p>A value of 1 indicates that the AWS KMS key used to encrypt data at rest has been deleted or revoked its grants to OpenSearch Service. 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.</p> <p>Relevant statistics: Minimum, Maximum</p>
InvalidHostHeaderRequest	<p>The number of HTTP requests made to the OpenSearch cluster that included an invalid (or missing) host header. Valid requests include the domain hostname as the host header value. OpenSearch Service 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.</p> <p>If you see large values for this metric, confirm that your OpenSearch clients include the domain hostname (and not, for example, its IP address) in their requests.</p> <p>Relevant statistics: Sum</p>
OpenSearchRequests (previous name: ElasticsearchRequests)	<p>The number of requests made to the OpenSearch cluster.</p> <p>Relevant statistics: Sum</p>
2xx, 3xx, 4xx, 5xx	<p>The number of requests to the domain that resulted in the given HTTP response code (2xx, 3xx, 4xx, 5xx).</p> <p>Relevant statistics: Sum</p>
ThroughputThrottle	<p>Indicates whether requests are being throttled due to the throughput limitations of your EBS volumes. A value of 1 indicates that some requests were throttled within the selected timeframe. A value of 0 indicates normal behavior.</p> <p>If you consistently see a value of 1 for this metric, you can scale up your instances by following AWS recommended best practices.</p> <p>Relevant statistics: Minimum, Maximum</p>

Dedicated master node metrics

Amazon OpenSearch Service provides the following metrics for [dedicated master nodes \(p. 358\)](#).

Metric	Description
MasterCPUUtilization	<p>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.</p> <p>Relevant statistics: Maximum</p>
MasterFreeStorageSpace	This metric is not relevant and can be ignored. The service does not use master nodes as data nodes.

Metric	Description
MasterJVMMemoryPressure	<p>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.</p> <p>Relevant statistics: Maximum</p> <p>Note The logic for this metric changed in service software R20220323. For more information, see the release notes (p. 450).</p>
MasterOldGenJVMMemoryPressure	<p>The maximum percentage of the Java heap used for the "old generation" per master node.</p> <p>Relevant statistics: Maximum</p>
MasterCPUCreditBalance	<p>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 <i>Amazon EC2 Developer Guide</i>. This metric is available only for the T2 instance types.</p> <p>Relevant statistics: Minimum</p>
MasterReachableFromNode	<p>A health check for <code>MasterNotDiscovered</code> exceptions. A value of 1 indicates normal behavior. A value of 0 indicates that <code>/_cluster/_health/</code> is failing.</p> <p>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.</p> <p>Relevant statistics: Minimum</p>
MasterSysMemoryUtilization	<p>The percentage of the master node's memory that is in use.</p> <p>Relevant statistics: Maximum</p>

EBS volume metrics

Amazon OpenSearch Service provides the following metrics for EBS volumes.

Metric	Description
ReadLatency	<p>The latency, in seconds, for read operations on EBS volumes. This metric is also available for individual nodes.</p> <p>Relevant statistics: Minimum, Maximum, Average</p>
WriteLatency	<p>The latency, in seconds, for write operations on EBS volumes. This metric is also available for individual nodes.</p> <p>Relevant statistics: Minimum, Maximum, Average</p>
ReadThroughput	<p>The throughput, in bytes per second, for read operations on EBS volumes. This metric is also available for individual nodes.</p>

Metric	Description
	Relevant statistics: Minimum, Maximum, Average
WriteThroughput	The throughput, in bytes per second, for write operations on EBS volumes. This metric is also available for individual nodes. Relevant statistics: Minimum, Maximum, Average
DiskQueueDepth	The number of pending input and output (I/O) requests for an EBS volume. Relevant statistics: Minimum, Maximum, Average
ReadIOPS	The number of input and output (I/O) operations per second for read operations on EBS volumes. This metric is also available for individual nodes. Relevant statistics: Minimum, Maximum, Average
WriteIOPS	The number of input and output (I/O) operations per second for write operations on EBS volumes. This metric is also available for individual nodes. Relevant statistics: Minimum, Maximum, Average
BurstBalance	The percentage of input and output (I/O) credits remaining in the burst bucket for an EBS volume. A value of 100 means that the volume has accumulated the maximum number of credits. If this percentage falls below 70%, see the section called "Low EBS burst balance" (p. 442) . Relevant statistics: Minimum, Maximum, Average

Instance metrics

Amazon OpenSearch Service provides the following metrics for each instance in a domain. OpenSearch Service also aggregates these instance metrics to provide insight into overall cluster health. You can verify this behavior using the **Sample Count** 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. OpenSearch and Elasticsearch 6.3 and later use the write thread pool. Currently, the OpenSearch Service console doesn't include a graph for the bulk thread pool.

Use `GET _cluster/settings?include_defaults=true` to check thread pool and queue sizes for your cluster.

Metric	Description
IndexingLatency	The average time, in milliseconds, that it takes a shard to complete an indexing operation. Relevant node statistics: Average Relevant cluster statistics: Average, Maximum
IndexingRate	The number of indexing operations per minute. A single call to the <code>_bulk</code> 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

Metric	Description
	<p>a total of four indexing operations. Document deletions do not count towards this metric.</p> <p>Relevant node statistics: Average</p> <p>Relevant cluster statistics: Average, Maximum, Sum</p>
SearchLatency	<p>The average time, in milliseconds, that it takes a shard on a data node to complete a search operation.</p> <p>Relevant node statistics: Average</p> <p>Relevant cluster statistics: Average, Maximum</p>
SearchRate	<p>The total number of search requests per minute for all shards on a data node. A single call to the <code>_search</code> 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.</p> <p>Relevant node statistics: Average</p> <p>Relevant cluster statistics: Average, Maximum, Sum</p>
SegmentCount	<p>The number of segments on a data node. The more segments you have, the longer each search takes. OpenSearch occasionally merges smaller segments into a larger one.</p> <p>Relevant node statistics: Maximum, Average</p> <p>Relevant cluster statistics: Sum, Maximum, Average</p>
SysMemoryUtilization	<p>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.</p> <p>Relevant node statistics: Minimum, Maximum, Average</p> <p>Relevant cluster statistics: Minimum, Maximum, Average</p>
JVMGCYoungCollectionCount	<p>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.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Sum, Maximum, Average</p>
JVMGCYoungCollectionTime	<p>The amount of time, in milliseconds, that the cluster has spent performing "young generation" garbage collection.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Sum, Maximum, Average</p>

Metric	Description
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. Relevant node statistics: Maximum Relevant cluster statistics: Sum, Maximum, Average
JVMGCOldCollectionTime	The amount of time, in milliseconds, that the cluster has spent performing "old generation" garbage collection. Relevant node statistics: Maximum Relevant cluster statistics: Sum, Maximum, Average
OpenSearchDashboardsConcurrentConnections (previously KibanaConcurrentConnections)	The number of active concurrent connections to OpenSearch Dashboards. If this number is consistently high, consider scaling your cluster. Relevant node statistics: Maximum Relevant cluster statistics: Sum, Maximum, Average
OpenSearchDashboardsHealthy (previously KibanaHealthyNode)	The healthy check for the individual OpenSearch Dashboards node. A value of 1 indicates normal behavior. A value of 0 indicates that Dashboards is inaccessible. Relevant node statistics: Minimum Relevant cluster statistics: Minimum, Maximum, Average
OpenSearchDashboardsHeap (previously KibanaHeapTotal)	The total amount of heap memory allocated to OpenSearch Dashboards in MiB. Different EC2 instance types can impact the exact memory allocation. Relevant node statistics: Maximum Relevant cluster statistics: Sum, Maximum, Average
OpenSearchDashboardsHeapUsed (previously KibanaHeapUsed)	The absolute amount of heap memory used by OpenSearch Dashboards in MiB. Relevant node statistics: Maximum Relevant cluster statistics: Sum, Maximum, Average
OpenSearchDashboardsHeapUtilization (previously KibanaHeapUtilization)	The maximum percentage of available heap memory used by OpenSearch Dashboards. If this value increases above 80%, consider scaling your cluster. Relevant node statistics: Maximum Relevant cluster statistics: Minimum, Maximum, Average

Metric	Description
OpenSearchDashboardsOS1M (previously KibanaOS1MinuteLoad)	The total CPU load average for OpenSearch Dashboards. The CPU load should ideally stay below 1.00. While temporary spikes are fine, we recommend increasing the size of the instance type if this metric is consistently above 1.00. Relevant node statistics: Average Relevant cluster statistics: Average, Maximum
OpenSearchDashboardsRequestTotal (previously KibanaRequestTotal)	The total count of HTTP requests made to OpenSearch Dashboards. If your system is slow or you see high numbers of Dashboards requests, consider increasing the size of the instance type. Relevant node statistics: Sum Relevant cluster statistics: Sum
OpenSearchDashboardsResponseTimesMaxInMilliseconds (previously KibanaResponseTimesMaxInMilliseconds)	The maximum time, in milliseconds, that it takes for OpenSearch Dashboards to respond to a request. If requests continually take a long time to return results, consider increasing the size of the instance type. Relevant node statistics: Maximum Relevant cluster statistics: 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. Relevant node statistics: Maximum Relevant cluster statistics: Sum, Maximum, Average
ThreadpoolForce_mergeRejected	The number of rejected tasks in the force merge thread pool. If this number continually grows, consider scaling your cluster. Relevant node statistics: Maximum Relevant cluster statistics: Sum
ThreadpoolForce_mergeThreads	The size of the force merge thread pool. Relevant node statistics: Maximum 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. Relevant node statistics: Maximum Relevant cluster statistics: Sum, Maximum, Average

Metric	Description
ThreadpoolIndexRejected	<p>The number of rejected tasks in the index thread pool. If this number continually grows, consider scaling your cluster.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Sum</p>
ThreadpoolIndexThreads	<p>The size of the index thread pool.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Average, Sum</p>
ThreadpoolSearchQueue	<p>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.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Sum, Maximum, Average</p>
ThreadpoolSearchRejected	<p>The number of rejected tasks in the search thread pool. If this number continually grows, consider scaling your cluster.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Sum</p>
ThreadpoolSearchThreads	<p>The size of the search thread pool.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Average, Sum</p>
Threadpoolsql-workerQueue	<p>The number of queued tasks in the SQL search thread pool. If the queue size is consistently high, consider scaling your cluster.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Sum, Maximum, Average</p>
Threadpoolsql-workerRejected	<p>The number of rejected tasks in the SQL search thread pool. If this number continually grows, consider scaling your cluster.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Sum</p>
Threadpoolsql-workerThreads	<p>The size of the SQL search thread pool.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Average, Sum</p>

Metric	Description
ThreadpoolBulkQueue	<p>The number of queued tasks in the bulk thread pool. If the queue size is consistently high, consider scaling your cluster.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Sum, Maximum, Average</p>
ThreadpoolBulkRejected	<p>The number of rejected tasks in the bulk thread pool. If this number continually grows, consider scaling your cluster.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Sum</p>
ThreadpoolBulkThreads	<p>The size of the bulk thread pool.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Average, Sum</p>
ThreadpoolWriteThreads	<p>The size of the write thread pool.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Average, Sum</p>
ThreadpoolWriteQueue	<p>The number of queued tasks in the write thread pool.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Average, Sum</p>
ThreadpoolWriteRejected	<p>The number of rejected tasks in the write thread pool.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Average, Sum</p> <p>Note Because the default write queue size was increased from 200 to 10000 in version 7.9, this metric is no longer the only indicator of rejections from OpenSearch Service. Use the <code>CoordinatingWriteRejected</code>, <code>PrimaryWriteRejected</code>, and <code>ReplicaWriteRejected</code> metrics to monitor rejections in versions 7.9 and later.</p>
CoordinatingWriteRejected	<p>The total number of rejections happened on the coordinating node due to indexing pressure since the last OpenSearch Service process startup.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Average, Sum</p> <p>This metric is available in version 7.9 and above.</p>

Metric	Description
PrimaryWriteRejected	<p>The total number of rejections happened on the primary shards due to indexing pressure since the last OpenSearch Service process startup.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Average, Sum</p> <p>This metric is available in version 7.9 and above.</p>
ReplicaWriteRejected	<p>The total number of rejections happened on the replica shards due to indexing pressure since the last OpenSearch Service process startup.</p> <p>Relevant node statistics: Maximum</p> <p>Relevant cluster statistics: Average, Sum</p> <p>This metric is available in version 7.9 and above.</p>

UltraWarm metrics

Amazon OpenSearch Service provides the following metrics for [UltraWarm \(p. 286\)](#) nodes.

Metric	Description
WarmCPUUtilization	<p>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.</p> <p>Relevant statistics: Maximum, Average</p>
WarmFreeStorageSpace	<p>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.</p> <p>Relevant statistics: Sum</p>
WarmSearchableDocuments	<p>The total number of searchable documents across all warm indexes in the cluster. You must leave the period at one minute to get an accurate value.</p> <p>Relevant statistics: Sum</p>
WarmSearchLatency	<p>The average time, in milliseconds, that it takes a shard on an UltraWarm node to complete a search operation.</p> <p>Relevant node statistics: Average</p> <p>Relevant cluster statistics: Average, Maximum</p>
WarmSearchRate	<p>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.</p> <p>Relevant node statistics: Average</p>

Metric	Description
	Relevant cluster statistics: Average, Maximum, Sum
WarmStorageSpaceUtil	The total amount of warm storage space, in MiB, that the cluster is using. Relevant statistics: Maximum
HotStorageSpaceUtil	The total amount of hot storage space that the cluster is using. Relevant statistics: Maximum
WarmSysMemoryUtilizat	The percentage of the warm node's memory that is in use. Relevant statistics: Maximum
HotToWarmMigrationQu	The number of indexes currently waiting to migrate from hot to warm storage. Relevant statistics: Maximum
WarmToHotMigrationQu	The number of indexes currently waiting to migrate from warm to hot storage. Relevant statistics: Maximum
HotToWarmMigrationFa	The total number of failed hot to warm migrations. Relevant statistics: Sum
HotToWarmMigrationFo	The average latency of the force merge stage of the migration process. If this stage consistently takes too long, consider increasing <code>index.ultrawarm.migration.force_merge.max_num_segments</code> . Relevant statistics: Average
HotToWarmMigrationSn	The average latency of the snapshot stage of the migration process. If this stage consistently takes too long, ensure that your shards are appropriately sized and distributed throughout the cluster. Relevant statistics: Average
HotToWarmMigrationPr	The average latency of successful hot to warm migrations, <i>not</i> including time spent in the queue. This value is the sum of the amount of time it takes to complete the force merge, snapshot, and shard relocation stages of the migration process. Relevant statistics: Average
HotToWarmMigrationSu	The total number of successful hot to warm migrations. Relevant statistics: Sum
HotToWarmMigrationSu	The average latency of successful hot to warm migrations, including time spent in the queue. Relevant statistics: Average

Metric	Description
WarmThreadpoolSearchRejected	The count of the UltraWarm search thread pool. Relevant node statistics: Maximum Relevant cluster statistics: Average, Sum
WarmThreadpoolSearchQueue	The count of rejected tasks in the UltraWarm search thread pool. If this number continually grows, consider adding more UltraWarm nodes. Relevant node statistics: Maximum Relevant cluster statistics: Sum
WarmThreadpoolSearchQueueSize	The number of queued tasks in the UltraWarm search thread pool. If the queue size is consistently high, consider adding more UltraWarm nodes. Relevant node statistics: Maximum Relevant cluster statistics: Sum, Maximum, Average
WarmJVMMemoryPressure	The maximum percentage of the Java heap used for the UltraWarm nodes. Relevant statistics: Maximum Note The logic for this metric changed in service software R20220323. For more information, see the release notes (p. 450) .
WarmOldGenJVMMemoryPressure	The maximum percentage of the Java heap used for the "old generation" per UltraWarm node. Relevant statistics: Maximum
WarmJVMGCGYoungCollection	The count of times that "young generation" garbage collection has run on UltraWarm nodes. A large, ever-growing number of runs is a normal part of cluster operations. Relevant node statistics: Maximum Relevant cluster statistics: Sum, Maximum, Average
WarmJVMGCGYoungCollectionTime	The time of time, in milliseconds, that the cluster has spent performing "young generation" garbage collection on UltraWarm nodes. Relevant node statistics: Maximum Relevant cluster statistics: Sum, Maximum, Average
WarmJVMGCOldCollection	The count of times that "old generation" garbage collection has run on UltraWarm nodes. In a cluster with sufficient resources, this number should remain small and grow infrequently. Relevant node statistics: Maximum Relevant cluster statistics: Sum, Maximum, Average

Cold storage metrics

Amazon OpenSearch Service provides the following metrics for [cold storage \(p. 295\)](#).

Metric	Description
ColdStorageSpaceUtilization	The total amount of cold storage space, in MiB, that the cluster is using. Relevant statistics: Max
ColdToWarmMigrationFailureCount	The total number of failed cold to warm migrations. Relevant statistics: Sum
ColdToWarmMigrationLatency	The amount of time for successful cold to warm migrations to complete. Relevant statistics: Average
ColdToWarmMigrationQueueSize	The number of indexes currently waiting to migrate from cold to warm storage. Relevant statistics: Maximum
ColdToWarmMigrationSuccessCount	The total number of successful cold to warm migrations. Relevant statistics: Sum
WarmToColdMigrationFailureCount	The total number of failed warm to cold migrations. Relevant statistics: Sum
WarmToColdMigrationLatency	The amount of time for successful warm to cold migrations to complete. Relevant statistics: Average
WarmToColdMigrationQueueSize	The number of indexes currently waiting to migrate from warm to cold storage. Relevant statistics: Maximum
WarmToColdMigrationSuccessCount	The total number of successful warm to cold migrations. Relevant statistics: Sum

Alerting metrics

Amazon OpenSearch Service provides the following metrics for [alerting \(p. 328\)](#).

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. Relevant statistics: Maximum

Metric	Description
AlertingIndexExists	A value of 1 means the <code>.opensearch-alerting-config</code> index exists. A value of 0 means it does not. Until you use the alerting feature for the first time, this value remains 0. Relevant statistics: Maximum
AlertingIndexStatus	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. Relevant statistics: Maximum
AlertingIndexStatus	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. Relevant statistics: Maximum
AlertingIndexStatus	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. 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 OpenSearch Service console or make a <code>_nodes/stats</code> request to see if any nodes show high resource usage. 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. Relevant statistics: Maximum
AlertingScheduledJobsEnabled	A value of 1 means that the <code>opensearch.scheduled_jobs.enabled</code> cluster setting is true. A value of 0 means it is false, and scheduled jobs are disabled. Relevant statistics: Maximum

Anomaly detection metrics

Amazon OpenSearch Service provides the following metrics for [anomaly detection \(p. 331\)](#).

Metric	Description
ADPluginUnhealthy	A value of 1 means that the anomaly detection plugin is not functioning properly, either because of a high number of failures or because one of the indexes that it uses is red. A value of 0 indicates the plugin is working as expected. Relevant statistics: Maximum
ADExecuteRequestCount	The number of requests to detect anomalies. Relevant statistics: Sum

Metric	Description
ADExecuteFailureCount	The number of failed requests to detect anomalies. Relevant statistics: Sum
ADHExecuteFailureCount	The number of failed requests to detect anomalies for high cardinality detectors. Relevant statistics: Sum
ADHExecuteRequestCount	The number of requests to detect anomalies for high cardinality detectors. Relevant statistics: Sum
ADAnomalyResultsIndexStatus	A status of index that the .opensearch-anomaly-results alias points to exists. Until you use anomaly detection for the first time, this value remains 0. Relevant statistics: Maximum
ADAnomalyResultsIndexStatus	A status of index that the .opensearch-anomaly-results alias points to is red. A value of 0 means it is not. Until you use anomaly detection for the first time, this value remains 0. Relevant statistics: Maximum
ADAnomalyDetectorsIndexStatus	A status of index that the .opensearch-anomaly-detectors index exists. A value of 0 means it does not. Until you use anomaly detection for the first time, this value remains 0. Relevant statistics: Maximum
ADAnomalyDetectorsIndexStatus	A status of index that the .opensearch-anomaly-detectors index is red. A value of 0 means it is not. Until you use anomaly detection for the first time, this value remains 0. Relevant statistics: Maximum
ADModelsCheckpointsIndexStatus	A status of index that the .opensearch-anomaly-checkpoints index exists. A value of 0 means it does not. Until you use anomaly detection for the first time, this value remains 0. Relevant statistics: Maximum
ADModelsCheckpointsIndexStatus	A status of index that the .opensearch-anomaly-checkpoints index is red. A value of 0 means it is not. Until you use anomaly detection for the first time, this value remains 0. Relevant statistics: Maximum

Asynchronous search metrics

Amazon OpenSearch Service provides the following metrics for [asynchronous search \(p. 276\)](#).

Asynchronous search coordinator node statistics (per coordinator node)

Metric	Description
AsynchronousSearchSubmitted	The sum of asynchronous searches submitted in the last minute.
AsynchronousSearchInitialized	The sum of asynchronous searches initialized in the last minute.
AsynchronousSearchRunning	The number of asynchronous searches currently running.
AsynchronousSearchCompleted	The sum of asynchronous searches successfully completed in the last minute.
AsynchronousSearchFailed	The rate of asynchronous searches that completed and failed in the last minute.
AsynchronousSearchPersisted	The rate of asynchronous searches that persisted in the last minute.
AsynchronousSearchPersistFailure	The rate of asynchronous searches that failed to persist in the last minute.
AsynchronousSearchRejected	The total number of asynchronous searches rejected since the node up time.
AsynchronousSearchCancelled	The total number of asynchronous searches cancelled since the node up time.
AsynchronousSearchMaxDuration	The duration of longest running asynchronous search on a node in the last minute.

Asynchronous search cluster statistics

Metric	Description
AsynchronousSearchStoreHealth	The Health of the store in the persisted index (RED/non-RED) in the last minute.
AsynchronousSearchStoreSize	The Size of the system index across all shards in the last minute.
AsynchronousSearchStoreResponseTime	The Response Time of responses in the system index in the last minute.

SQL metrics

Amazon OpenSearch Service provides the following metrics for [SQL support \(p. 245\)](#).

Metric	Description
SQLFailedRequestCountClient	The sum of requests to the _sql API that failed due to a client issue. For example, a request might return HTTP status code 400 due to an IndexNotFoundException. Relevant statistics: Sum
SQLFailedRequestCountSystem	The sum of requests to the _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. Relevant statistics: Sum

Metric	Description
SQLRequestCount	The number of requests to the <code>_sql</code> API. Relevant statistics: Sum
SQLDefaultCursorRequestCount	Similar to <code>SQLRequestCount</code> but only counts pagination requests. Relevant statistics: Sum
SQLUnhealthy	A value of 1 indicates that, in response to certain requests, the SQL plugin is returning 5xx response codes or passing invalid query DSL to OpenSearch. 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. Relevant statistics: Maximum

k-NN metrics

Amazon OpenSearch Service includes the following metrics for the k-nearest neighbor ([k-NN \(p. 248\)](#)) plugin.

Metric	Description
KNNCacheCapacityReached	Per-node metric for whether cache capacity has been reached. This metric is only relevant to approximate k-NN search. Relevant statistics: Maximum
KNNCircuitBreakerTriggered	Per-cluster metric for whether the circuit breaker is triggered. If any nodes return a value of 1 for <code>KNNCacheCapacityReached</code> , this value will also return 1. This metric is only relevant to approximate k-NN search. Relevant statistics: Maximum
KNNEvictionCount	Per-node metric for the number of graphs that have been evicted from the cache due to memory constraints or idle time. Explicit evictions that occur because of index deletion are not counted. This metric is only relevant to approximate k-NN search. Relevant statistics: Sum
KNNGraphIndexErrors	Per-node metric for the number of requests to add the <code>knn_vector</code> field of a document to a graph that produced an error. Relevant statistics: Sum
KNNGraphIndexRequests	Per-node metric for the number of requests to add the <code>knn_vector</code> field of a document to a graph. Relevant statistics: Sum
KNNGraphMemoryUsage	Per-node metric for the current cache size (total size of all graphs in memory) in kilobytes. This metric is only relevant to approximate k-NN search.

Metric	Description
	Relevant statistics: Average
KNNGraphQueryErrors	Per-node metric for the number of graph queries that produced an error. Relevant statistics: Sum
KNNGraphQueryRequests	Per-node metric for the number of graph queries. Relevant statistics: Sum
KNNHitCount	Per-node metric for the number of cache hits. A cache hit occurs when a user queries a graph that is already loaded into memory. This metric is only relevant to approximate k-NN search. Relevant statistics: Sum
KNNLoadExceptionCount	Per-node metric for the number of times an exception occurred while trying to load a graph into the cache. This metric is only relevant to approximate k-NN search. Relevant statistics: Sum
KNNLoadSuccessCount	Per-node metric for the number of times the plugin successfully loaded a graph into the cache. This metric is only relevant to approximate k-NN search. Relevant statistics: Sum
KNNMissCount	Per-node metric for the number of cache misses. A cache miss occurs when a user queries a graph that is not yet loaded into memory. This metric is only relevant to approximate k-NN search. Relevant statistics: Sum
KNNQueryRequests	Per-node metric for the number of query requests the k-NN plugin received. Relevant statistics: Sum
KNNScriptCompilationErrors	Per-node metric for the number of errors during script compilation. This statistic is only relevant to k-NN score script search. Relevant statistics: Sum
KNNScriptCompilations	Per-node metric for the number of times the k-NN script has been compiled. This value should usually be 1 or 0, but if the cache containing the compiled scripts is filled, the k-NN script might be recompiled. This statistic is only relevant to k-NN score script search. Relevant statistics: Sum
KNNScriptQueryErrors	Per-node metric for the number of errors during script queries. This statistic is only relevant to k-NN score script search. Relevant statistics: Sum

Metric	Description
KNNScriptQueryRequests	Per-node metric for the total number of script queries. This statistic is only relevant to k-NN score script search. Relevant statistics: Sum
KNNTotalLoadTime	The time in nanoseconds that k-NN has taken to load graphs into the cache. This metric is only relevant to approximate k-NN search. Relevant statistics: Sum

Cross-cluster search metrics

Amazon OpenSearch Service provides the following metrics for [cross-cluster search \(p. 251\)](#).

Source domain metrics

Metric	Dimension	Description
CrossClusterOutboundConnections	Connections	Number of connected nodes. If your response includes one or more skipped domains, use this metric to trace any unhealthy connections. If this number drops to 0, then the connection is unhealthy.
CrossClusterOutboundRequests	Destinations	Number of search requests sent to the destination domain. Use to check if the load of cross-cluster search requests are overwhelming your domain, correlate any spike in this metric with any JVM/CPU spike.

Destination domain metric

Metric	Dimension	Description
CrossClusterInboundRequests	Requests	Number of incoming connection requests received from the source domain.

Add a CloudWatch alarm in the event that you lose a connection unexpectedly. For steps to create an alarm, see [Create a CloudWatch Alarm Based on a Static Threshold](#).

Cross-cluster replication metrics

Amazon OpenSearch Service provides the following metrics for [cross-cluster replication \(p. 315\)](#).

Metric	Description
ReplicationRate	The average rate of replication operations per second. This metric is similar to the IndexingRate metric.
LeaderCheckPoint	For a specific connection, the sum of leader checkpoint values across all replicating indexes. You can use this metric to measure replication latency.

Metric	Description
FollowerCheckPoint	For a specific connection, the sum of follower checkpoint values across all replicating indexes. You can use this metric to measure replication latency.
ReplicationNumSyncingIndexes	The number of indexes that have a replication status of SYNCING.
ReplicationNumBootstrappingIndexes	The number of indexes that have a replication status of BOOTSTRAPPING.
ReplicationNumPausedIndexes	The number of indexes that have a replication status of PAUSED.
ReplicationNumFailedIndexes	The number of indexes that have a replication status of FAILED.
AutoFollowNumSuccesses	The number of follower indexes that have been successfully created by a replication rule for a specific connection.
AutoFollowNumFailedStarts	The number of follower indexes that failed to be created by a replication rule when there was a matching pattern. This problem might arise due to a network issue on the remote cluster, or a security issue (i.e. the associated role doesn't have permission to start replication).
AutoFollowLeaderCallsFailed	Whether there have been any failed queries from the follower index to the leader index to pull new data. A value of 1 means that there have been 1 or more failed calls in the last minute.

Learning to Rank metrics

Amazon OpenSearch Service provides the following metrics for [Learning to Rank \(p. 257\)](#).

Metric	Description
LTRRequestTotalCount	Total count of ranking requests.
LTRRequestErrorCount	Total count of unsuccessful requests.
LTRStatus.red	Tracks if one of the indexes needed to run the plugin is red.
LTRMemoryUsage	Total memory used by the plugin.
LTRFeatureMemoryUsageInBytes	The amount of memory, in bytes, used by Learning to Rank feature fields.
LTRFeaturesetMemoryUsageInBytes	The amount of memory, in bytes, used by all Learning to Rank feature sets.
LTRModelMemoryUsageInBytes	The amount of memory, in bytes, used by all Learning to Rank models.

Piped Processing Language metrics

Amazon OpenSearch Service provides the following metrics for [Piped Processing Language \(p. 343\)](#).

Metric	Description
PPLFailedRequestCount	The number of requests to the _ppl API that failed due to a client issue. For example, a request might return HTTP status code 400 due to an IndexNotFoundException.

Metric	Description
PPLFailedRequestCount	The number of requests to the _ppl 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.
PPLRequestCount	The number of requests to the _ppl API.

Monitoring OpenSearch logs with Amazon CloudWatch Logs

Amazon OpenSearch Service exposes the following OpenSearch logs through Amazon CloudWatch Logs:

- Error logs
- [Search slow logs](#)
- [Indexing slow logs](#)
- [Audit logs \(p. 97\)](#)

Search slow logs, indexing slow logs, and error logs are useful for troubleshooting performance and stability issues. Audit logs track user activity for compliance purposes. All the logs are *disabled* by default. If enabled, [standard CloudWatch pricing](#) applies.

Note

Error logs are available only for OpenSearch and Elasticsearch versions 5.1 and later. Slow logs are available for all OpenSearch and Elasticsearch versions.

For its logs, OpenSearch 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, OpenSearch Service publishes log lines of WARN, ERROR, and FATAL to CloudWatch. OpenSearch Service also publishes several exceptions from the DEBUG level, including the following:

- org.opensearch.index.mapper.MapperParsingException
- org.opensearch.index.query.QueryShardException
- org.opensearch.action.search.SearchPhaseExecutionException
- org.opensearch.common.util.concurrent.OpenSearchRejectedExecutionException
- 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
- Index State Management migration failures

Enabling log publishing (console)

The OpenSearch Service 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 **Amazon OpenSearch Service**.
3. Select the domain you want to update.
4. On the **Logs** tab, select a log type and choose **Enable**.
5. Create a new 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:/*"  
        }  
    ]  
}
```

We recommend that you add the `aws:SourceAccount` and `aws:SourceArn` condition keys to the policy to protect yourself against the [confused deputy problem](#). The source account is the owner of the domain and the source ARN is the ARN of the domain. Your domain must be on service software R20211203 or later in order to add these condition keys.

For example, you could add the following condition block to the policy:

```
"Condition": {  
    "StringEquals": {  
        "aws:SourceAccount": "account-id"  
    },  
    "ArnLike": {  
        "aws:SourceArn": "arn:aws:es:region:account-id:domain/domain-name"  
    }  
}
```

Important

CloudWatch Logs supports [10 resource policies per Region](#). If you plan to enable logs for several OpenSearch Service domains, you should create and reuse a broader policy that includes multiple log groups to avoid reaching this limit. For steps on updating your policy, see [the section called “Enabling log publishing \(AWS CLI\)” \(p. 94\)](#).

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 OpenSearch logging thresholds for slow logs” \(p. 96\)](#). If you enabled audit logs, see [the section called “Step 2: Turn on audit logs in OpenSearch Dashboards” \(p. 99\)](#). 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 OpenSearch Service 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>CreateLogStream"], "Resource": "cw_log_group_arn:*"}]}'
```

Important

CloudWatch Logs supports [10 resource policies per Region](#). If you plan to enable slow logs for several OpenSearch Service domains, you should create and reuse a broader policy that includes multiple log groups to avoid reaching this limit.

If you need to review this policy at a later time, use the `aws logs describe-resource-policies` command. To update the policy, issue the same `aws logs put-resource-policy` command with a new policy document.

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

Parameter	Valid Values
--log-publishing-options	SEARCH_SLOW_LOGS={CloudWatchLogsLogGroupArn= <i>cw_log_group_arn</i> ,Enabled=false}
	INDEX_SLOW_LOGS={CloudWatchLogsLogGroupArn= <i>cw_log_group_arn</i> ,Enabled=false}
	ES_APPLICATION_LOGS={CloudWatchLogsLogGroupArn= <i>cw_log_group_arn</i> ,Enabled=false}
	AUDIT_LOGS={CloudWatchLogsLogGroupArn= <i>cw_log_group_arn</i> ,Enabled=true}

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 indexing slow logs for the specified domain:

```
aws opensearch update-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 OpenSearch logging thresholds for slow logs" \(p. 96\)](#). If you enabled audit logs, see [the section called "Step 2: Turn on audit logs in OpenSearch Dashboards" \(p. 99\)](#). 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 OpenSearch Service 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 the [Amazon OpenSearch Service API Reference](#), including the --log-publishing-options option for `CreateDomain` and `UpdateDomainConfig`.

If you enabled one of the slow logs, see [the section called "Setting OpenSearch logging thresholds for slow logs" \(p. 96\)](#). If you enabled only error logs, you don't need to perform any additional configuration steps.

Enabling log publishing (CloudFormation)

In this example, we use CloudFormation to create a log group called `opensearch-logs`, assign the appropriate permissions, and then create a domain with log publishing enabled for application logs, search slow logs, and indexing slow logs.

Before you can enable log publishing, you need to create a CloudWatch log group:

```
Resources:
  OpenSearchLogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: opensearch-logs
Outputs:
  Arn:
    Value:
      'Fn::GetAtt':
        - OpenSearchLogGroup
        - Arn
```

The template outputs the ARN of the log group. In this case, the ARN is `arn:aws:logs:us-east-1:123456789012:log-group:opensearch-logs`.

Using the ARN, create a resource policy that gives OpenSearch Service permissions to write to the log group:

```
Resources:
  OpenSearchLogPolicy:
    Type: AWS::Logs::ResourcePolicy
    Properties:
      PolicyName: my-policy
      PolicyDocument: "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Sid\": \"\", \"Effect\": \"Allow\", \"Principal\": { \"Service\": \"es.amazonaws.com\"}, \"Action\": [ \"logs:PutLogEvents\", \"logs>CreateLogStream\"], \"Resource\": \"arn:aws:logs:us-east-1:123456789012:log-group:opensearch-logs:*\" } ] }"
```

Finally, create the following CloudFormation stack which generates an OpenSearch Service domain with log publishing enabled. The access policy permits the root user for the AWS account to make all HTTP requests to the domain:

```
Resources:
  OpenSearchServiceDomain:
    Type: "AWS::OpenSearchService::Domain"
    Properties:
      DomainName: my-domain
      EngineVersion: "OpenSearch_1.0"
      ClusterConfig:
        InstanceCount: 2
        InstanceType: "r6g.xlarge.search"
        DedicatedMasterEnabled: true
        DedicatedMasterCount: 3
        DedicatedMasterType: "r6g.xlarge.search"
      EBSOptions:
        EBSEnabled: true
        VolumeSize: 10
        VolumeType: "gp2"
      AccessPolicies:
        Version: "2012-10-17"
        Statement:
          Effect: "Allow"
          Principal:
            AWS: "arn:aws:iam::123456789012:user/es-user"
            Action: "es:/*"
            Resource: "arn:aws:es:us-east-1:123456789012:domain/my-domain/*"
      LogPublishingOptions:
        ES_APPLICATION_LOGS:
          CloudWatchLogsLogGroupArn: "arn:aws:logs:us-east-1:123456789012:log-group:opensearch-logs"
          Enabled: true
        SEARCH_SLOW_LOGS:
          CloudWatchLogsLogGroupArn: "arn:aws:logs:us-east-1:123456789012:log-group:opensearch-logs"
          Enabled: true
        INDEX_SLOW_LOGS:
          CloudWatchLogsLogGroupArn: "arn:aws:logs:us-east-1:123456789012:log-group:opensearch-logs"
          Enabled: true
```

For detailed syntax information, see the [log publishing options](#) in the *AWS CloudFormation User Guide*.

Setting OpenSearch logging thresholds for slow logs

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

You specify these settings through the OpenSearch REST API:

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

```
GET 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 OpenSearch Service console or AWS CLI does *not* stop OpenSearch 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:

- OpenSearch Service publishes only the first 255,000 characters of each line to CloudWatch. Any remaining content is truncated. For audit logs, it's 10,000 characters per message.
- In CloudWatch, the log stream names have suffixes of `-index-slow-logs`, `-search-slow-logs`, `-application-logs`, and `-audit-logs` to help identify their contents.

Monitoring audit logs in Amazon OpenSearch Service

If your Amazon OpenSearch Service domain uses fine-grained access control, you can enable audit logs for your data. Audit logs are highly customizable and let you track user activity on your OpenSearch clusters, including authentication success and failures, requests to OpenSearch, index changes, and incoming search queries. The default configuration tracks a popular set of user actions, but we recommend tailoring the settings to your exact needs.

Just like [OpenSearch application logs and slow logs \(p. 92\)](#), OpenSearch Service publishes audit logs to CloudWatch Logs. If enabled, [standard CloudWatch pricing](#) applies.

Note

To enable audit logs, your user role must be mapped to the `security_manager` role, which gives you access to the OpenSearch plugins/_security REST API. To learn more, see the section called “[Modifying the master user](#)” (p. 158).

Limitations

Audit logs have the following limitations:

- Audit logs don't include cross-cluster search requests that were rejected by the destination's domain access policy.
- The maximum size of each audit log message is 10,000 characters. The audit log message is truncated if it exceeds this limit.

Enabling audit logs

Enabling audit logs is a two-step process. First, you configure your domain to publish audit logs to CloudWatch Logs. Then, you enable audit logs in OpenSearch Dashboards and configure them to meet your needs.

Important

If you encounter an error while following these steps, see the section called “[Can't enable audit logs](#)” (p. 442) for troubleshooting information.

Step 1: Enable audit logs and configure an access policy

These steps describe how to enable audit logs using the console. You can also [enable them using the AWS CLI \(p. 99\)](#), or the [configuration API \(p. 99\)](#).

To enable audit logs for an OpenSearch Service domain (console)

1. Choose the domain to open its configuration, then go to the **Logs** tab.
2. Select **Audit logs** and then **Enable**.
3. Create a CloudWatch log group, or choose an existing one.
4. 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"  
        }  
    ]  
}
```

We recommend that you add the `aws:SourceAccount` and `aws:SourceArn` condition keys to the policy to protect yourself against the [confused deputy problem](#). The source account is the owner of the domain and the source ARN is the ARN of the domain. Your domain must be on service software R20211203 or later in order to add these condition keys.

For example, you could add the following condition block to the policy:

```
"Condition": {  
    "StringEquals": {  
        "aws:SourceAccount": "account-id"  
    },  
    "ArnLike": {  
        "aws:SourceArn": "arn:aws:es:region:account-id:domain/domain-name"  
    }  
}
```

5. Choose **Enable**.

Step 2: Turn on audit logs in OpenSearch Dashboards

After you enable audit logs in the OpenSearch Service console, you *must* also enable them in OpenSearch Dashboards and configure them to match your needs.

1. Open OpenSearch Dashboards and choose **Security** from the left side menu.
2. Choose **Audit logs**.
3. Choose **Enable audit logging**.

The Dashboards UI offers full control of audit log settings under **General settings** and **Compliance settings**. For a description of all configuration options, see [Audit log settings \(p. 101\)](#).

Enable audit logging using the AWS CLI

The following AWS CLI command enables audit logs on an existing domain:

```
aws opensearch update-domain-config --domain-name my-domain --log-publishing-options  
"AUDIT_LOGS={CloudWatchLogsLogGroupArn=arn:aws:logs:us-east-1:123456789012:log-group:my-log-group,Enabled=true}"
```

You can also enable audit logs when you create a domain. For detailed information, see the [AWS CLI Command Reference](#).

Enable audit logging using the configuration API

The following request to the configuration API enables audit logs on an existing domain:

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/my-domain/config  
{  
    "LogPublishingOptions": {  
        "AUDIT_LOGS": {  
            "CloudWatchLogsLogGroupArn": "arn:aws:logs:us-east-1:123456789012:log-group1:sample-domain",  
            "Enabled": true|false  
        }  
    }  
}
```

}

For more information, see the [Amazon OpenSearch Service API Reference](#).

Audit log layers and categories

Cluster communication occurs over two separate *layers*: the REST layer and the transport layer.

- The REST layer covers communication with HTTP clients such as curl, Logstash, OpenSearch Dashboards, the [Java high-level REST client \(p. 192\)](#), the Python [Requests](#) library—all HTTP requests that arrive at the cluster.
- The transport layer covers communication between nodes. For example, after a search request arrives at the cluster (over the REST layer), the coordinating node serving the request sends the query to other nodes, receives their responses, gathers the necessary documents, and collates them into the final response. Operations such as shard allocation and rebalancing also occur over the transport layer.

You can enable or disable audit logs for entire layers, as well as individual audit categories for a layer. The following table contains a summary of audit categories and the layers for which they are available.

Category	Description	Available for REST	Available for transport
FAILED_LOGIN	A request contained invalid credentials, and authentication failed.	Yes	Yes
MISSING_PRIVILEGES	A user did not have the privileges to make the request.	Yes	Yes
GRANTED_PRIVILEGES	A user had the privileges to make the request.	Yes	Yes
OPENSEARCH_SECURITY_INDEX_ATTEMPTED	A request attempted to modify the .opendistro_security index.	No	Yes
AUTHENTICATED	A request contained valid credentials, and authentication succeeded.	Yes	Yes
INDEX_EVENT	A request performed an administrative operation on an index, such as creating one, setting an alias, or performing a force merge. The full list of indices:admin/actions that this category includes are available in the OpenSearch documentation .	No	Yes

In addition to these standard categories, fine-grained access control offers several additional categories designed to meet data compliance requirements.

Category	Description
COMPLIANCE_DOC_READ	A request performed a read event on a document in an index.
COMPLIANCE_DOC_WRITE	A request performed a write event on a document in an index.
COMPLIANCE_INTERNAL_CONFIG_READ	Configured a read event on the .opendistro_security index.
COMPLIANCE_INTERNAL_CONFIG_WRITE	Configured a write event on the .opendistro_security index.

You can have any combination of categories and message attributes. For example, if you send a REST request to index a document, you might see the following lines in the audit logs:

- AUTHENTICATED on REST layer (authentication)
- GRANTED_PRIVILEGE on transport layer (authorization)
- COMPLIANCE_DOC_WRITE (document written to an index)

Audit log settings

Audit logs have numerous configuration options.

General settings

General settings let you enable or disable individual categories or entire layers. We highly recommend leaving GRANTED_PRIVILEGES and AUTHENTICATED as excluded categories. Otherwise, these categories are logged for every valid request to the cluster.

Name	Backend setting	Description
REST layer	enable_rest	Enable or disable events that occur on the REST layer.
REST disabled categories	disabled_rest_categories	Specify audit categories to ignore on the REST layer. Modifying these categories can dramatically increase the size of the audit logs.
Transport layer	enable_transport	Enable or disable events that happen on the transport layer.
Transport disabled categories	disabled_transport_categories	Specify audit categories which must be ignored on the transport layer. Modifying these categories can dramatically increase the size of the audit logs.

Attribute settings let you customize the amount of detail in each log line.

Name	Backend setting	Description
Bulk requests	resolve_bulk_requests	Enabling this setting generates a log for each document in a bulk request, which can dramatically increase the size of the audit logs.
Request body	log_request_body	Include the request body of the requests.
Resolve indices	resolve_indices	Resolve aliases to indices.

Use ignore settings to exclude a set of users or API paths:

Name	Backend setting	Description
Ignored users	ignore_users	Specify users that you want to exclude.
Ignored requests	ignore_requests	Specify request patterns that you want to exclude.

Compliance settings

Compliance settings let you tune for index, document, or field-level access.

Name	Backend setting	Description
Compliance logging	enable_compliance	Enable or disable compliance logging.

You can specify the following settings for read and write event logging.

Name	Backend setting	Description
Internal config logging	internal_config	Enable or disable logging of events on the <code>.opendistro_security</code> index.
External config logging	external_config	Enable or disable logging of external configuration events.

You can specify the following settings for read events.

Name	Backend setting	Description
Read metadata	read_metadata_only	Include only metadata for read events. Do not include any document fields.
Ignored users	read_ignore_users	Do not include certain users for read events.
Watched fields	read_watched_fields	Specify the indices and fields to watch for read events. Adding watched fields generates one log per document access, which can dramatically increase the size of the audit logs. Watched fields support index patterns and field patterns: <pre>{ "index-name-pattern": ["field-name-pattern"], "logs*": ["message"], "twitter": ["id", "user*"] }</pre>

You can specify the following settings for write events.

Name	Backend setting	Description
Write metadata	write_metadata_only	Include only metadata for write events. Do not include any document fields.
Log diffs	write_log_diffs	If write_metadata_only is false, include only the differences between write events.
Ignored users	write_ignore_users	Do not include certain users for write events.
Watch indices	write_watched_indices	Specify the indices or index patterns to watch for write events. Adding watched fields generates one log per document access, which can dramatically increase the size of the audit logs.

Audit log example

This section includes an example configuration, search request, and the resulting audit log for all read and write events of an index.

Step 1: Configure audit logs

After you enable the publishing of audit logs to a CloudWatch Logs group, navigate to the OpenSearch Dashboards audit logging page and choose **Enable audit logging**.

1. In **General Settings**, choose **Configure** and make sure that the **REST layer** is enabled.
2. In **Compliance Settings**, choose **Configure**.
3. Under **Write**, in **Watched Fields**, add accounts for all write events to this index.
4. Under **Read**, in **Watched Fields**, add ssn and id- fields of the accounts index:

```
{
  "accounts-": [
    "ssn",
    "id-"
  ]
}
```

Step 2: Perform read and write events

1. Navigate to OpenSearch Dashboards, choose **Dev Tools**, and index a sample document:

```
PUT accounts/_doc/0
{
  "ssn": "123",
  "id-": "456"
}
```

2. To test a read event, send the following request:

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

```
}
```

Step 3: Observe the logs

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Log groups**.
3. Choose the log group that you specified while enabling audit logs. Within the log group, OpenSearch Service creates a log stream for each node in your domain.
4. In **Log streams**, choose **Search all**.
5. For the read and write events, see the corresponding logs. You can expect a delay of 5 seconds before the log appears.

Sample write audit log

```
{
  "audit_compliance_operation": "CREATE",
  "audit_cluster_name": "824471164578:audit-test",
  "audit_node_name": "be217225a0b77c2bd76147d3ed3ff83c",
  "audit_category": "COMPLIANCE_DOC_WRITE",
  "audit_request_origin": "REST",
  "audit_compliance_doc_version": 1,
  "audit_node_id": "3xNJhm4XS_yTzEgDWcGRjA",
  "@timestamp": "2020-08-23T05:28:02.285+00:00",
  "audit_format_version": 4,
  "audit_request_remote_address": "3.236.145.227",
  "audit_trace_doc_id": "lxnJGXQBqZSlDB91r_uZ",
  "audit_request_effective_user": "admin",
  "audit_trace_shard_id": 8,
  "audit_trace_indices": [
    "accounts"
  ],
  "audit_trace_resolved_indices": [
    "accounts"
  ]
}
```

Sample read audit log

```
{
  "audit_cluster_name": "824471164578:audit-docs",
  "audit_node_name": "806f6050cb45437e2401b07534a1452f",
  "audit_category": "COMPLIANCE_DOC_READ",
  "audit_request_origin": "REST",
  "audit_node_id": "saSevm9ASte0-pjAtYi2UA",
  "@timestamp": "2020-08-31T17:57:05.015+00:00",
  "audit_format_version": 4,
  "audit_request_remote_address": "54.240.197.228",
  "audit_trace_doc_id": "config:7.7.0",
  "audit_request_effective_user": "admin",
  "audit_trace_shard_id": 0,
  "audit_trace_indices": [
    "accounts"
  ],
  "audit_trace_resolved_indices": [
    "accounts"
  ]
}
```

To include the request body, return to **Compliance settings** in OpenSearch Dashboards and disable **Write metadata**. To exclude events by a specific user, add the user to **Ignored Users**.

For a description of each audit log field, see [Audit log field reference](#). For information on searching and analyzing your audit log data, see [Analyzing Log Data with CloudWatch Logs Insights](#) in the *Amazon CloudWatch Logs User Guide*.

Configuring audit logs using the REST API

We recommend using OpenSearch Dashboards to configure audit logs, but you can also use the fine-grained access control REST API. This section contains a sample request. Full documentation on the REST API is available in the [OpenSearch documentation](#).

```
PUT _plugins/_security/api/audit/config
{
  "enabled": true,
  "audit": {
    "enable_rest": true,
    "disabled_rest_categories": [
      "GRANTED_PRIVILEGES",
      "AUTENTICATED"
    ],
    "enable_transport": true,
    "disabled_transport_categories": [
      "GRANTED_PRIVILEGES",
      "AUTENTICATED"
    ],
    "resolve_bulk_requests": true,
    "log_request_body": true,
    "resolve_indices": true,
    "exclude_sensitive_headers": true,
    "ignore_users": [
      "kibanaserver"
    ],
    "ignore_requests": [
      "SearchRequest",
      "indices:data/read/*",
      "/_cluster/health"
    ]
  },
  "compliance": {
    "enabled": true,
    "internal_config": true,
    "external_config": false,
    "read_metadata_only": true,
    "read_watched_fields": {
      "read-index-1": [
        "field-1",
        "field-2"
      ],
      "read-index-2": [
        "field-3"
      ]
    },
    "read_ignore_users": [
      "read-ignore-1"
    ],
    "write_metadata_only": true,
    "write_log_diffs": false,
    "write_watched_indices": [
      "write-index-1",
      "write-index-2",
      "log-*",
      "log-1"
    ]
  }
}
```

```
    "*"
],
"write_ignore_users": [
    "write-ignore-1"
]
}
```

Monitoring OpenSearch Service events with Amazon EventBridge

Amazon OpenSearch Service integrates with Amazon EventBridge to notify you of certain events that affect your domains. Events from AWS services are delivered to EventBridge in near real time. The same events are also sent to [Amazon CloudWatch Events](#), the predecessor of Amazon EventBridge. You can write simple rules to indicate which events are of interest to you, and what automated actions to take when an event matches a rule. The actions that can be automatically triggered include the following:

- Invoking an AWS Lambda function
- Invoking an Amazon EC2 Run Command
- Relaying the event to Amazon Kinesis Data Streams
- Activating an AWS Step Functions state machine
- Notifying an Amazon SNS topic or an Amazon SQS queue

For more information, see [Get started with Amazon EventBridge](#) in the *Amazon EventBridge User Guide*.

Service software update events

OpenSearch Service sends events to EventBridge when one of the following [service software update \(p. 29\)](#) events occur.

Service software update available

OpenSearch Service sends this event when a service software update is available.

Example

The following is an example event of this type:

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-012345678901",
    "detail-type": "Amazon OpenSearch Service Software Update Notification",
    "source": "aws.es",
    "account": "123456789012",
    "time": "2016-11-01T13:12:22Z",
    "region": "us-east-1",
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
    "detail": {
        "event": "Service Software Update",
        "status": "Available",
        "severity": "Informational",
        "description": "Service software update [R20200330-p1] available."
    }
}
```

Service software update started

OpenSearch Service sends this event when a service software update has started.

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service Software Update Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2016-11-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],  
    "detail": {  
        "event": "Service Software Update",  
        "status": "Started",  
        "severity": "Informational",  
        "description": "Service software update [R20200330-p1] started."  
    }  
}
```

Service software update completed

OpenSearch Service sends this event when a service software update has completed.

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service Software Update Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2016-11-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],  
    "detail": {  
        "event": "Service Software Update",  
        "status": "Completed",  
        "severity": "Informational",  
        "description": "Service software update [R20200330-p1] completed."  
    }  
}
```

Service software update failed

OpenSearch Service sends this event when a service software update failed.

Example

The following is an example event of this type:

```
{  
    "version": "0",  
}
```

```
"id": "01234567-0123-0123-0123-012345678901",
"detail-type": "Amazon OpenSearch Service Software Update Notification",
"source": "aws.es",
"account": "123456789012",
"time": "2016-11-01T13:12:22Z",
"region": "us-east-1",
"resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
"detail": {
    "event": "Service Software Update",
    "status": "Failed",
    "severity": "Medium",
    "description": "Service software update [R20200330-p1] failed."
}
}
```

Service software update required

OpenSearch Service sends this event when a service software update is required.

Example

The following is an example event of this type:

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-012345678901",
    "detail-type": "Amazon OpenSearch Service Software Update Notification",
    "source": "aws.es",
    "account": "123456789012",
    "time": "2016-11-01T13:12:22Z",
    "region": "us-east-1",
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
    "detail": {
        "event": "Service Software Update",
        "status": "Required",
        "severity": "High",
        "description": "Service software update [R20200330-p1] available. Update will be automatically installed after [30/04/2020] if no action is taken."
    }
}
```

Auto-Tune events

OpenSearch Service sends events to EventBridge when one of the following [Auto-Tune \(p. 60\)](#) events occur.

Auto-Tune pending

OpenSearch Service sends this event when Auto-Tune has identified tuning recommendations for improved cluster performance and availability. You'll only see this event for domains with Auto-Tune disabled.

Example

The following is an example event of this type:

```
{
    "version": "0",
    "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
```

```
"detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
"source": "aws.es",
"account": "123456789012",
"time": "2020-10-30T22:06:31Z",
"region": "us-east-1",
"resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
"detail": {
    "event": "Auto-Tune Event",
    "severity": "Informational",
    "status": "Pending",
    "description": "Auto-Tune recommends new settings for your domain. Enable Auto-Tune to improve cluster stability and performance.",
    "scheduleTime": "{iso8601-timestamp}"
}
}
```

Auto-Tune started

OpenSearch Service sends this event when Auto-Tune begins to apply new settings to your domain.

Example

The following is an example event of this type:

```
{
    "version": "0",
    "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
    "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
    "source": "aws.es",
    "account": "123456789012",
    "time": "2020-10-30T22:06:31Z",
    "region": "us-east-1",
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
    "detail": {
        "event": "Auto-Tune Events",
        "severity": "Informational",
        "status": "Started",
        "scheduleTime": "{iso8601-timestamp}",
        "startTime": "{iso8601-timestamp}",
        "description": "Auto-Tune is applying new settings to your domain."
    }
}
```

Auto-Tune requires a scheduled blue/green deployment

OpenSearch Service sends this event when Auto-Tune has identified tuning recommendations that require a scheduled blue/green deployment.

Example

The following is an example event of this type:

```
{
    "version": "0",
    "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
    "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
    "source": "aws.es",
    "account": "123456789012",
    "time": "2020-10-30T22:06:31Z",
    "region": "us-east-1",
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
    "detail": {
```

```
        "event": "Auto-Tune Event",
        "severity": "Low",
        "status": "Pending",
        "startTime": "{iso8601-timestamp}",
        "description": "Auto-Tune has identified new settings for your domain that require a blue/green deployment.
                        You can schedule the deployment for your preferred time."
    }
}
```

Auto-Tune cancelled

OpenSearch Service sends this event when Auto-Tune schedule has been cancelled because there is no pending tuning recommendations.

Example

The following is an example event of this type:

```
{
    "version": "0",
    "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
    "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
    "source": "aws.es",
    "account": "123456789012",
    "time": "2020-10-30T22:06:31Z",
    "region": "us-east-1",
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
    "detail": {
        "event": "Auto-Tune Event",
        "severity": "Low",
        "status": "Cancelled",
        "scheduleTime": "{iso8601-timestamp}",
        "description": "Auto-Tune has cancelled the upcoming blue/green deployment."
    }
}
```

Auto-Tune completed

OpenSearch Service sends this event when Auto-Tune has completed the blue/green deployment and the cluster is operational with new JVM settings in place.

Example

The following is an example event of this type:

```
{
    "version": "0",
    "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
    "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
    "source": "aws.es",
    "account": "123456789012",
    "time": "2020-10-30T22:06:31Z",
    "region": "us-east-1",
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],
    "detail": {
        "event": "Auto-Tune Event",
        "severity": "Informational",
        "status": "Completed",
        "completionTime": "{iso8601-timestamp}",
        "description": "Auto-Tune has completed the blue/green deployment and successfully applied the updated settings."
    }
}
```

```
}
```

Auto-Tune disabled and changes reverted

OpenSearch Service sends this event when Auto-Tune has been disabled and the applied changes were rolled back.

Example

The following is an example event of this type:

```
{
  "version": "0",
  "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
  "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2020-10-30T22:06:31Z",
  "region": "us-east-1",
  "resources": [ "arn:aws:es:us-east-1:123456789012:domain/test-domain" ],
  "detail": {
    "event": "Auto-Tune Event",
    "severity": "Informational",
    "status": "Completed",
    "description": "Auto-Tune is now disabled. All settings have been reverted. Auto-Tune will continue to evaluate cluster performance and provide recommendations.",
    "completionTime": "{iso8601-timestamp}"
  }
}
```

Auto-Tune disabled and changes retained

OpenSearch Service sends this event when Auto-Tune has been disabled and the applied changes were retained.

Example

The following is an example event of this type:

```
{
  "version": "0",
  "id": "3acb26c8-397c-4c89-a80a-ce672a864c55",
  "detail-type": "Amazon OpenSearch Service Auto-Tune Notification",
  "source": "aws.es",
  "account": "123456789012",
  "time": "2020-10-30T22:06:31Z",
  "region": "us-east-1",
  "resources": [ "arn:aws:es:us-east-1:123456789012:domain/test-domain" ],
  "detail": {
    "event": "Auto-Tune Event",
    "severity": "Informational",
    "status": "Completed",
    "description": "Auto-Tune is now disabled. The most-recent settings by Auto-Tune have been retained. Auto-Tune will continue to evaluate cluster performance and provide recommendations.",
    "completionTime": "{iso8601-timestamp}"
  }
}
```

Cluster health events

OpenSearch Service sends certain events to EventBridge when your cluster's health is compromised.

Red cluster recovery started

OpenSearch Service sends this event after your cluster status has been continuously red for more than an hour. It attempts to automatically restore one or more red indexes from a snapshot in order to fix the cluster status.

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service Cluster Status Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2016-11-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": [  
        "arn:aws:es:us-east-1:123456789012:domain/test-domain"  
    ],  
    "detail": {  
        "event": "Automatic Snapshot Restore for Red Indices",  
        "status": "Started",  
        "severity": "High",  
        "description": "Your cluster status is red. We have started automatic snapshot restore  
for the red indices.  
No action is needed from your side. Red indices [red-index-0, red-  
index-1]"  
    }  
}
```

Red cluster recovery partially completed

OpenSearch Service sends this event when it was only able to restore a subset of red indexes from a snapshot while attempting to fix a red cluster status.

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service Cluster Status Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2016-11-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": [  
        "arn:aws:es:us-east-1:123456789012:domain/test-domain"  
    ],  
    "detail": {  
        "event": "Automatic Snapshot Restore for Red Indices",  
        "status": "Partially Restored",  
        "severity": "High",  
    }  
}
```

```
    "description":"Your cluster status is red. We were able to restore the following Red indices from
                  snapshot: [red-index-0]. Indices not restored: [red-index-1]. Please
                  refer https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-
                  errors.html#handling-errors-red-cluster-status for troubleshooting steps."
    }
}
```

Red cluster recovery failed

OpenSearch Service sends this event when it fails to restore any indexes while attempting to fix a red cluster status.

Example

The following is an example event of this type:

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Cluster Status Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2016-11-01T13:12:22Z",
  "region":"us-east-1",
  "resources":[
    "arn:aws:es:us-east-1:123456789012:domain/test-domain"
  ],
  "detail":{
    "event":"Automatic Snapshot Restore for Red Indices",
    "status":"Failed",
    "severity":"High",
    "description":"Your cluster status is red. We were unable to restore the Red indices
automatically.
Indices not restored: [red-index-0, red-index-1]. Please refer https://
docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-errors.html#handling-
errors-red-cluster-status for troubleshooting steps."
  }
}
```

Shards to be deleted

OpenSearch Service sends this event when it has attempted to automatically fix your red cluster status after it was continuously red for 14 days, but one or more indexes remains red. After 7 more days (21 total days of being continuously red), OpenSearch Service proceeds to [delete unassigned shards \(p. 114\)](#) on all red indexes.

Example

The following is an example event of this type:

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-012345678901",
  "detail-type":"Amazon OpenSearch Service Cluster Status Notification",
  "source":"aws.es",
  "account":"123456789012",
  "time":"2022-04-09T10:36:48Z",
  "region":"us-east-1",
  "resources":[
```

```
        "arn:aws:es:us-east-1:123456789012:domain/test-domain"
    ],
    "detail": {
        "severity": "Medium",
        "description": "Your cluster status is red. Please fix the red indices as soon as
possible.
If not fixed by 2022-04-12 01:51:47+00:00, we will delete all
unassigned shards,
the unit of storage and compute, for these red indices to recover your
domain and make it green.
Please refer to https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-errors.html#handling-errors-red-cluster-status for troubleshooting
steps.
        "test_data, test_data1",
        "event": "Automatic Snapshot Restore for Red Indices",
        "status": "Shard(s) to be deleted"
    }
}
```

Shards deleted

OpenSearch Service sends this event after your cluster status has been continuously red for 21 days. It proceeds to delete the unassigned shards (storage and compute) on all red indexes. For details, see [the section called “Automatic remediation of red clusters” \(p. 438\)](#).

Example

The following is an example event of this type:

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-012345678901",
    "detail-type": "Amazon OpenSearch Service Cluster Status Notification",
    "source": "aws.es",
    "account": "123456789012",
    "time": "2022-04-09T10:54:48Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:es:us-east-1:123456789012:domain/test-domain"
    ],
    "detail": {
        "severity": "High",
        "description": "We have deleted unassigned shards, the unit of storage and compute,
in
red indices: index-1, index-2 because these indices were red for more
than
21 days and could not be restored with the automated restore process.
Please refer to https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-errors.html#handling-errors-red-cluster-status for troubleshooting
steps.",
        "event": "Automatic Snapshot Restore for Red Indices",
        "status": "Shard(s) deleted"
    }
}
```

High shard count warning

OpenSearch Service sends this event when the average shard count across your hot data nodes has exceeded 90% of the recommended default limit of 1,000. Although later versions of Elasticsearch and OpenSearch support a configurable max shard count per node limit, we recommend you have no more than 1,000 shards per node. See [Choosing the number of shards \(p. 355\)](#).

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2016-11-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],  
    "detail": {  
        "event": "High Shard Count",  
        "status": "Warning",  
        "severity": "Low",  
        "description": "One or more data nodes have close to 1000 shards. To ensure optimum performance and stability of your cluster, please refer to the best practice guidelines - https://docs.aws.amazon.com/opensearch-service/latest/developerguide/sizing-domains.html#bp-sharding."  
    }  
}
```

Shard count limit exceeded

OpenSearch Service sends this event when the average shard count across your hot data nodes has exceeded the recommended default limit of 1,000. Although later versions of Elasticsearch and OpenSearch support a configurable max shard count per node limit, we recommend you have no more than 1,000 shards per node. See [Choosing the number of shards \(p. 355\)](#).

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2016-11-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],  
    "detail": {  
        "event": "High Shard Count",  
        "status": "Warning",  
        "severity": "Medium",  
        "description": "One or more data nodes have more than 1000 shards. To ensure optimum performance and stability of your cluster, please refer to the best practice guidelines - https://docs.aws.amazon.com/opensearch-service/latest/developerguide/sizing-domains.html#bp-sharding."  
    }  
}
```

Low disk space

OpenSearch Service sends this event when one or more nodes in your cluster has less than 25% of available storage space, or less than 25 GB.

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2017-12-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],  
    "detail": {  
        "event": "Low Disk Space",  
        "status": "Warning",  
        "severity": "Medium",  
        "description": "One or more data nodes in your cluster has less than 25% of storage space or less than 25GB.  
Your cluster will be blocked for writes at 20% or 20GB. Please refer to the documentation for more information - https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-errors.html#troubleshooting-cluster-block"  
    }  
}
```

EBS burst balance below 70%

OpenSearch Service sends this event when the EBS burst balance on one or more data nodes falls below 70%. EBS burst balance depletion can cause widespread cluster unavailability and throttling of I/O requests, which can lead to high latencies and timeouts on indexing and search requests. For steps to fix this issue, see [the section called “Low EBS burst balance” \(p. 442\)](#).

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2017-12-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],  
    "detail": {  
        "event": "EBS Burst Balance",  
        "status": "Warning",  
        "severity": "Medium",  
        "description": "EBS burst balance on one or more data nodes is below 70%.  
Follow https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-errors.html#handling-errors-low-ebs-burst  
to fix this issue."  
    }  
}
```

EBS burst balance below 20%

OpenSearch Service sends this event when the EBS burst balance on one or more data nodes falls below 20%. EBS burst balance depletion can cause widespread cluster unavailability and throttling of I/O

requests, which can lead to high latencies and timeouts on indexing and search requests. For steps to fix this issue, see [the section called “Low EBS burst balance” \(p. 442\)](#).

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2017-12-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],  
    "detail": {  
        "event": "EBS Burst Balance",  
        "status": "Warning",  
        "severity": "High",  
        "description": "EBS burst balance on one or more data nodes is below 20%.  
Follow https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-errors.html#handling-errors-low-ebs-burst  
to fix this issue."  
    }  
}
```

Disk throughput throttle

OpenSearch Service sends this event when read and write requests to your domain are being throttled due to the throughput limitations of your EBS volumes. If you receive this notification, consider scaling up your instances following AWS recommended best practices.

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2017-12-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],  
    "detail": {  
        "event": "Disk Throughput Throttle",  
        "status": "Warning",  
        "severity": "Medium",  
        "description": "Your domain is experiencing throttling as you have hit disk throughout limits.  
Please consider scaling your domain to suit your throughput needs.  
Please refer to the documentation for more information."  
    }  
}
```

VPC endpoint events

OpenSearch Service sends certain events to EventBridge related to [AWS PrivateLink interface endpoints \(p. 167\)](#).

VPC endpoint creation failed

OpenSearch Service sends this event when it's unable to create a requested VPC endpoint. This error might occur because you've reached the limit on the number of VPC endpoints allowed within a Region. You will also see this error if a specified subnet or security group doesn't exist.

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service VPC Endpoint Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2016-11-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": [  
        "arn:aws:es:us-east-1:123456789012:domain/test-domain"  
    ],  
    "detail": {  
        "event": "VPC Endpoint Create Validation",  
        "status": "Failed",  
        "severity": "High",  
        "description": "Unable to create VPC endpoint aos-0d4c74c0342343 for domain  
                        arn:aws:es:eu-south-1:123456789012:domain/my-domain due to the  
                        following validation failures: You've reached the limit on the  
                        number of VPC endpoints that you can create in the AWS Region."  
    }  
}
```

VPC endpoint update failed

OpenSearch Service sends this event when it's unable to delete a requested VPC endpoint.

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service VPC Endpoint Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2016-11-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": [  
        "arn:aws:es:us-east-1:123456789012:domain/test-domain"  
    ],  
    "detail": {  
        "event": "VPC Endpoint Update Validation",  
        "status": "Failed",  
        "severity": "High",  
        "description": "Unable to update VPC endpoint aos-0d4c74c0342343 for domain  
                        arn:aws:es:eu-south-1:123456789012:domain/my-domain due to the  
                        following validation failures: <failure message>."  
    }  
}
```

VPC endpoint deletion failed

OpenSearch Service sends this event when it's unable to delete a requested VPC endpoint.

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service VPC Endpoint Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2016-11-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": [  
        "arn:aws:es:us-east-1:123456789012:domain/test-domain"  
    ],  
    "detail": {  
        "event": "VPC Endpoint Delete Validation",  
        "status": "Failed",  
        "severity": "High",  
        "description": "Unable to delete VPC endpoint aos-0d4c74c0342343 for domain  
                    arn:aws:es:eu-south-1:123456789012:domain/my-domain due to the  
                    following validation failures: Specified subnet doesn't exist."  
    }  
}
```

Domain error events

OpenSearch Service sends events to EventBridge when one of the following domain errors occur.

Domain update validation failure

OpenSearch Service sends this event if it encounters one or more validation failures when attempting to update or perform a configuration change on a domain. For steps to resolve these failures, see [the section called “Troubleshooting validation errors” \(p. 26\)](#).

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Amazon OpenSearch Service Domain Update Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2016-11-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": [  
        "arn:aws:es:us-east-1:123456789012:domain/test-domain"  
    ],  
    "detail": {  
        "event": "Domain Update Validation",  
        "status": "Failed",  
        "severity": "High",  
        "description": "Unable to perform updates to your domain due to the following  
                    validation failures: <failures>"  
    }  
}
```

```
Please see the documentation for more information https://  
docs.aws.amazon.com/opensearch-service/latest/developerguide/managedomains-configuration-  
changes.html#validation"  
}  
}
```

KMS key inaccessible

OpenSearch Service sends this event when it [can't access your AWS KMS key \(p. 129\)](#).

Example

The following is an example event of this type:

```
{  
    "version": "0",  
    "id": "01234567-0123-0123-0123-012345678901",  
    "detail-type": "Domain Error Notification",  
    "source": "aws.es",  
    "account": "123456789012",  
    "time": "2016-11-01T13:12:22Z",  
    "region": "us-east-1",  
    "resources": ["arn:aws:es:us-east-1:123456789012:domain/test-domain"],  
    "detail": {  
        "event": "KMS Key Inaccessible",  
        "status": "Error",  
        "severity": "High",  
        "description": "The KMS key associated with this domain is inaccessible. You are at  
risk of losing access to your domain.  
For more information, please refer https://docs.aws.amazon.com/  
opensearch-service/latest/developerguide/encryption-at-rest.html#disabled-key."  
    }  
}
```

Tutorial: Listening for Amazon OpenSearch Service EventBridge events

In this tutorial, you set up a simple AWS Lambda function that listens for Amazon OpenSearch Service events and writes them to a CloudWatch Logs log stream.

Prerequisites

This tutorial assumes that you have an existing OpenSearch Service domain. If you haven't created a domain, follow the steps in [Creating and managing domains \(p. 16\)](#) to create one.

Step 1: Create the Lambda function

In this procedure, you create a simple Lambda function to serve as a target for OpenSearch Service event messages.

To create a target Lambda function

1. Open the AWS Lambda console at <https://console.aws.amazon.com/lambda/>.
2. Choose **Create function** and **Author from scratch**.
3. For **Function name**, enter **event-handler**.
4. For **Runtime**, choose **Python 3.8**.

5. Choose **Create function**.
6. In the **Function code** section, edit the sample code to match the following example:

```
import json

def lambda_handler(event, context):
    if event["source"] != "aws.es":
        raise ValueError("Function only supports input from events with a source type
of: aws.es")

    print(json.dumps(event))
```

This is a simple Python 3.8 function that prints the events sent by OpenSearch Service. If everything is configured correctly, at the end of this tutorial, the event details appear in the CloudWatch Logs log stream that's associated with this Lambda function.

7. Choose **Deploy**.

Step 2: Register an event rule

In this step, you create an EventBridge rule that captures events from your OpenSearch Service domains. This rule captures all events within the account where it's defined. The event messages themselves contain information about the event source, including the domain from which it originated. You can use this information to filter and sort events programmatically.

To create an EventBridge rule

1. Open the EventBridge console at <https://console.aws.amazon.com/events/>.
2. Choose **Create rule**.
3. Name the rule **event-rule**.
4. Choose **Next**.
5. For the event pattern, select **AWS services**, **Amazon OpenSearch Service**, and **All Events**. This pattern applies across all of your OpenSearch Service domains and to every OpenSearch Service event. Alternatively, you can create a more specific pattern to filter out some results.
6. Press **Next**.
7. For the target, choose **Lambda function**. In the function dropdown, choose **event-handler**.
8. Press **Next**.
9. Skip the tags and press **Next** again.
10. Review the configuration and choose **Create rule**.

Step 3: Test your configuration

The next time you receive a notification in the **Notifications** section of the OpenSearch Service console, if everything is configured properly, your Lambda function is triggered and it writes the event data to a CloudWatch Logs log stream for the function.

To test your configuration

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. On the navigation pane, choose **Logs** and select the log group for your Lambda function (for example, **/aws/lambda/event-handler**).
3. Select a log stream to view the event data.

Tutorial: Sending Amazon SNS alerts for available software updates

In this tutorial, you configure an Amazon EventBridge event rule that captures notifications for available service software updates in Amazon OpenSearch Service and sends you an email notification through Amazon Simple Notification Service (Amazon SNS).

Prerequisites

This tutorial assumes that you have an existing OpenSearch Service domain. If you haven't created a domain, follow the steps in [Creating and managing domains \(p. 16\)](#) to create one.

Step 1: Create and subscribe to an Amazon SNS topic

Configure an Amazon SNS topic to serve as an event target for your new event rule.

To create an Amazon SNS target

1. Open the Amazon SNS console at <https://console.aws.amazon.com/sns/v3/home>.
2. Choose **Topics** and **Create topic**.
3. For the job type, choose **Standard**, and name the job **software-update**.
4. Choose **Create topic**.
5. After the topic is created, choose **Create subscription**.
6. For **Protocol**, choose **Email**. For **Endpoint**, enter an email address that you currently have access to and choose **Create subscription**.
7. Check your email account and wait to receive a subscription confirmation email message. When you receive it, choose **Confirm subscription**.

Step 2: Register an event rule

Next, register an event rule that captures only service software update events.

To create an event rule

1. Open the EventBridge console at <https://console.aws.amazon.com/events/>.
2. Choose **Create rule**.
3. Name the rule **softwareupdate-rule**.
4. Choose **Next**.
5. For the event pattern, select **AWS services**, **Amazon OpenSearch Service**, and **Amazon OpenSearch Service Software Update Notification**. This pattern matches any service software update event from OpenSearch Service. For more information about event patterns, see [Amazon EventBridge event patterns](#) in the [Amazon EventBridge User Guide](#).
6. Optionally, you can filter to only specific severities. For the severities of each event, see the section [called "Service software update events" \(p. 106\)](#).
7. Choose **Next**.
8. For the target, choose **SNS topic** and select **software-update**.
9. Choose **Next**.
10. Skip the tags and choose **Next**.
11. Review the rule configuration and choose **Create rule**.

The next time you receive a notification from OpenSearch Service about an available service software update, if everything is configured properly, Amazon SNS should send you an email alert about the update.

Monitoring Amazon OpenSearch Service API calls with AWS CloudTrail

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

Note

CloudTrail only captures calls to the [Configuration API](#), such as `CreateDomain` and `GetUpgradeStatus`. CloudTrail doesn't capture calls to the [OpenSearch APIs \(p. 373\)](#), such as `_search` and `_bulk`. For these calls, see the section called "Monitoring audit logs" (p. 97).

The captured calls include calls from the OpenSearch Service 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 OpenSearch Service. 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 OpenSearch Service, 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 OpenSearch Service information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in OpenSearch Service, 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 account. For more information, see [Viewing events with CloudTrail event history](#).

For an ongoing record of events in your AWS account account, including events for OpenSearch Service, 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:

- [Creating a trail for your AWS account](#)
- [AWS service integrations with CloudTrail Logs](#)
- [Configuring Amazon SNS notifications for CloudTrail](#)
- [Receiving CloudTrail log files from multiple Regions](#) and [Receiving CloudTrail log files from multiple accounts](#)

All OpenSearch Service configuration API actions are logged by CloudTrail and are documented in the [Amazon OpenSearch Service API Reference](#).

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 OpenSearch 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 `CreateDomain` 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": "CreateDomain",  
    "awsRegion": "us-west-1",  
    "sourceIPAddress": "123.123.123.123",  
    "userAgent": "signin.amazonaws.com",  
    "requestParameters": {  
        "engineVersion": "OpenSearch_1.0",  
        "clusterConfig": {  
            "instanceType": "m4.large.search",  
            "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,
            "clusterConfig": {
                "zoneAwarenessEnabled": false,
                "instanceType": "m4.large.search",
                "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"
            },
            "engineVersion": "OpenSearch_1.0",
            "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"
}
```

Security in Amazon OpenSearch 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 OpenSearch 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 OpenSearch Service. The following topics show you how to configure OpenSearch Service to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your OpenSearch Service resources.

Topics

- [Data protection in Amazon OpenSearch Service \(p. 126\)](#)
- [Identity and Access Management in Amazon OpenSearch Service \(p. 130\)](#)
- [Cross-service confused deputy prevention \(p. 145\)](#)
- [Fine-grained access control in Amazon OpenSearch Service \(p. 146\)](#)
- [Compliance validation for Amazon OpenSearch Service \(p. 165\)](#)
- [Resilience in Amazon OpenSearch Service \(p. 166\)](#)
- [Infrastructure security in Amazon OpenSearch Service \(p. 166\)](#)
- [SAML authentication for OpenSearch Dashboards \(p. 169\)](#)
- [Configuring Amazon Cognito authentication for OpenSearch Dashboards \(p. 175\)](#)
- [Using service-linked roles for Amazon OpenSearch Service \(p. 188\)](#)

Data protection in Amazon OpenSearch Service

The AWS [shared responsibility model](#) applies to data protection in Amazon OpenSearch Service. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the [AWS Security Blog](#).

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). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- 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.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-2](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with OpenSearch Service or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption of data at rest for Amazon OpenSearch Service

OpenSearch Service 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:

- All indexes (including those in UltraWarm storage)
- OpenSearch 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: You currently can't use AWS KMS keys to encrypt manual snapshots. You can, however, use server-side encryption with S3-managed keys or KMS keys to encrypt the bucket you use as a snapshot repository. For instructions, see the section called ["Registering a manual snapshot repository" \(p. 45\)](#).
- Slow logs and error logs: If you [publish logs \(p. 92\)](#) and want to encrypt them, you can encrypt their CloudWatch Logs log group using the same AWS KMS key as the OpenSearch Service domain. For more information, see [Encrypt log data in CloudWatch Logs using AWS KMS](#) in the [Amazon CloudWatch Logs User Guide](#).

Note

You can't enable encryption at rest on an existing domain if UltraWarm is enabled on the domain. You must first disable UltraWarm storage, enable encryption at rest, and then re-enable UltraWarm.

OpenSearch Service supports only symmetric encryption KMS keys, not asymmetric ones. To learn how to create symmetric 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 \(p. 238\)](#) using AES-256 and OpenSearch Service-managed keys.

Permissions

To use the OpenSearch Service console to configure encryption of data at rest, you must have read permissions to AWS KMS, such as the following identity-based policy:

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

If you want to use a key other than the AWS owned key, 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 OpenSearch Service, you can add the `kms:ViaService` condition to that 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*.

Enabling encryption of data at rest

Encryption of data at rest on new domains requires either OpenSearch or Elasticsearch 5.1 or later. Enabling it on existing domains requires either OpenSearch or Elasticsearch 6.7 or later.

To enable encryption of data at rest (console)

1. Open the domain in the AWS console, then choose **Actions** and **Edit security configuration**.
2. Under **Encryption**, select **Enable encryption of data at rest**.
3. Choose an AWS KMS key to use, then choose **Save changes**.

You can also enable encryption through the configuration API. The following request enables encryption of data at rest on an existing domain:

```
{  
    "ClusterConfig":{  
        "EncryptionAtRestOptions":{  
            "Enabled": true,  
        }  
    }  
}
```

```
        "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:alias/my-key"
    }
}
```

Disabled or deleted KMS key

If you disable or delete the key that you used to encrypt a domain, the domain becomes inaccessible. OpenSearch Service sends you a [notification \(p. 32\)](#) informing you that it can't access the KMS key. Re-enable the key immediately to access your domain.

The OpenSearch Service team can't help you recover your data if your key is deleted. AWS KMS deletes keys only after a waiting period of at least seven days. If your key is pending deletion, either cancel deletion or take a [manual snapshot \(p. 42\)](#) of the domain to prevent loss of data.

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 \(p. 42\)](#) of the existing domain, [create another domain \(p. 16\)](#), 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” \(p. 69\)](#). You can view them using either the OpenSearch Service 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” \(p. 361\)](#).

Other considerations

- Automatic key rotation preserves the properties of your AWS KMS keys, so the rotation has no effect on your ability to access your OpenSearch data. Encrypted OpenSearch Service domains don't support manual key rotation, which involves creating a new key and updating any references to the old key. To learn more, see [Rotating 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” \(p. 365\)](#).
- Domains that encrypt data at rest use a different repository name for their automated snapshots. For more information, see [the section called “Restoring snapshots” \(p. 49\)](#).
- While we highly recommend enabling encryption at rest, it can add additional CPU overhead and a few milliseconds of latency. Most use cases aren't sensitive to these differences, however, and the magnitude of impact depends on the configuration of your cluster, clients, and usage profile.

Node-to-node encryption for Amazon OpenSearch Service

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

Each OpenSearch Service domain—regardless of whether the domain uses VPC access—resides within its own, dedicated VPC. This architecture prevents potential attackers from intercepting traffic

between OpenSearch 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 OpenSearch Service over HTTPS, node-to-node encryption helps ensure that your data remains encrypted as OpenSearch distributes (and redistributes) it throughout the cluster. If data arrives unencrypted over HTTP, OpenSearch Service 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

Node-to-node encryption on new domains requires any version of OpenSearch, or Elasticsearch 6.0 or later. Enabling node-to-node encryption on existing domains requires any version of OpenSearch, or Elasticsearch 6.7 or later. Choose the existing domain in the AWS console, **Actions**, and **Edit security configuration**.

Alternatively, you can use the AWS CLI or configuration API. For more information, see the [AWS CLI Command Reference](#) and [OpenSearch Service API reference](#).

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 \(p. 42\)](#) of the encrypted domain, [create another domain \(p. 16\)](#), migrate your data, and delete the old domain.

Identity and Access Management in Amazon OpenSearch Service

Amazon OpenSearch Service offers several ways to control access to your domains. This topic 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 OpenSearch Service access control. For more information, see [the section called "About access policies on VPC domains" \(p. 39\)](#).

Types of policies

OpenSearch Service supports three types of access policies:

- [the section called "Resource-based policies" \(p. 130\)](#)
- [the section called "Identity-based policies" \(p. 132\)](#)
- [the section called "IP-based policies" \(p. 134\)](#)

Resource-based policies

You add a resource-based policy, often called the domain access policy, when you create a domain. These policies specify which actions a principal can perform on the domain's *subresources* (with the exception of [cross-cluster search \(p. 253\)](#)). Subresources include OpenSearch indexes 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 the subresources on `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 similar policies on other domains, test-user can only access test-domain.
- The trailing `/*` in the Resource element is significant and indicates that resource-based policies only apply to the domain's subresources, not the domain itself. In resource-based policies, the `es:*` action is equivalent to `es:ESHttpGet*`.

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/2021-01-01/opensearch/domain/test-domain/config`). Note the difference between the two endpoints. Accessing the [configuration API](#) requires an [identity-based policy \(p. 132\)](#).

You can specify a partial index name by adding a wildcard. This one identifies any indexes beginning with `commerce`:

```
arn:aws:es:us-west-1:987654321098:domain/test-domain/commerce*
```

In this case, specifying using this wildcard means that test-user can make requests to indexes in the test-domain domain that have names that begin with `commerce`.

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/commerce-data/_search"  
        }  
    ]  
}
```

```
}
```

Now `test-user` can perform only one operation: searches against the `commerce-data` index. All other indexes 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 URLs 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/commerce-data/*"
    }
  ]
}
```

If your domain is in a VPC or uses fine-grained access control, you can use an open domain access policy. Otherwise, your domain access policy must contain some restriction, either by principal or IP address.

For information about all available actions, see [the section called “Policy element reference” \(p. 137\)](#). For far more granular control over your data, use an open domain access policy with [fine-grained access control \(p. 146\)](#).

Identity-based policies

Unlike resource-based policies, which are a part of each OpenSearch Service domain, you attach identity-based policies to users or roles using the AWS Identity and Access Management (IAM) service. Just like [resource-based policies \(p. 130\)](#), 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](#) (or [fine-grained access control \(p. 146\)](#)) in OpenSearch Service to offer users access to OpenSearch indexes and APIs.

Note

Users with the AWS managed `AmazonOpenSearchServiceReadOnlyAccess` policy can't see cluster health status on the console. To allow them to see cluster health status (and other OpenSearch data), add the `es:ESHttpGet` action to an access policy and attach it to their accounts or roles.

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 OpenSearch Service and all data stored on all domains:

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

Identity-based policies let you use tags to control access to the configuration API. The following policy, for example, lets attached principals view and update a domain's configuration if the domain has the team:devops tag:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "es:UpdateDomainConfig",
        "es:DescribeDomain",
        "es:DescribeDomainConfig"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": [
          "aws:ResourceTag/team": [
            "devops"
          ]
        ]
      }
    }
  ]
}
```

You can also use tags to control access to the OpenSearch API. Tag-based policies for the OpenSearch API only apply to HTTP methods. For example, the following policy lets attached principals send GET and PUT requests to the OpenSearch API if the domain has the environment:production tag:

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

```
"Effect": "Allow",
"Resource": "*",
"Condition": {
    "ForAnyValue:StringEquals": {
        "aws:ResourceTag/environment": [
            "production"
        ]
    }
}
}
```

For more granular control of the OpenSearch API, consider using [fine-grained access control \(p. 146\)](#).

Note

After you add one or more OpenSearch APIs to any tag-based policy, you must perform a single [tag operation \(p. 63\)](#) (such as adding, removing, or modifying a tag) in order for the changes to take effect on a domain. You must be on service software R20211203 or later to include OpenSearch API operations in tag-based policies.

OpenSearch Service supports the RequestTag and TagKeys global condition keys for the configuration API, not the OpenSearch API. These conditions only apply to API calls that include tags within the request, such as CreateDomain, AddTags, and RemoveTags. The following policy lets attached principals create domains, but only if they include the team:it tag in the request:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "es>CreateDomain",
            "es>AddTags"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/team": [
                    "it"
                ]
            }
        }
    }
}
```

For more details on using tags for access control and the differences between resource-based and identity-based policies, see the [IAM User Guide](#).

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 OpenSearch Service domain, which lets you use clients like [curl](#) and [OpenSearch Dashboards \(p. 280\)](#) or access the domain through a proxy server. To learn more, see the section called [“Using a proxy to access OpenSearch Service from Dashboards” \(p. 280\)](#).

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” \(p. 39\)](#).

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 \(p. 146\)](#), 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 OpenSearch Service requests

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

- To make calls to the OpenSearch Service 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/2021-01-01/
```

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/2021-01-01/opensearch/domain/movies/config
{
    "ClusterConfig": {
        "InstanceType": "c5.xlarge.search"
    }
}
```

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_domain_config(
    DomainName='movies',
    ClusterConfig={
        'InstanceType': 'c5.xlarge.search'
    }
)
```

For a Java code sample, see [the section called “Using the AWS SDKs” \(p. 205\)](#).

- To make calls to the OpenSearch APIs, you must sign your own requests. For sample code in a variety of languages, see [the section called “Signing HTTP requests” \(p. 191\)](#). The OpenSearch 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 subresource (an OpenSearch index or API), 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 for domain subresources.

	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

OpenSearch Service 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	<p>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:</p> <ul style="list-style-type: none"> AWS accounts: "Principal": {"AWS": ["123456789012"]} or "Principal": {"AWS": ["arn:aws:iam::123456789012:root"]} IAM users: "Principal": {"AWS": ["arn:aws:iam::123456789012:user/test-user"]} IAM roles: "Principal": {"AWS": ["arn:aws:iam::123456789012:role/test-role"]} <p>Specifying the * wildcard enables anonymous access to the domain, which we don't recommend unless you add an IP-based condition (p. 134), use VPC support (p. 37), or enable fine-grained access control (p. 146).</p>
Action	<p>OpenSearch Service uses ESHtp* actions for OpenSearch HTTP methods. The rest of the actions apply to the configuration API.</p> <p>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 <i>any</i> 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.</p>

JSON policy element	Summary
	<p>For a list of all available actions and whether they apply to the domain subresources (test-domain/*), to the domain configuration (test-domain), or only to the service (*), see Actions, resources, and condition keys for Amazon OpenSearch Service in the <i>Service Authorization Reference</i>.</p> <p>Resource-based policies differ from resource-level permissions. Resource-based policies (p. 130) 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.</p> <p>While resource-level permissions for es:CreateDomain 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-*".</p> <p>Of course, nothing prevents you from including actions alongside less restrictive resource elements, such as the following:</p> <pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["es:ESHttpGet", "es:DescribeDomain"], "Resource": "*" }] }</pre> <p>To learn more about pairing actions and resources, see the Resource element in this table.</p>
Condition	<p>OpenSearch Service supports most conditions that are described in AWS global condition context keys in the <i>IAM User Guide</i>. Notable exceptions include the aws:SecureTransport and aws:PrincipalTag keys, which OpenSearch Service does not support.</p> <p>When configuring an IP-based policy (p. 134), you specify the IP addresses or CIDR block as a condition, such as the following:</p> <pre>"Condition": { "IpAddress": { "aws:SourceIp": ["192.0.2.0/32"] } }</pre> <p>As noted in the section called “Identity-based policies” (p. 132), the aws:ResourceTag, aws:RequestTag, and aws:TagKeys condition keys apply to the configuration API as well as the OpenSearch APIs.</p>

JSON policy element	Summary
Resource	<p>OpenSearch Service uses Resource elements in three basic ways:</p> <ul style="list-style-type: none"> For actions that apply to OpenSearch Service itself, like <code>es>ListDomainNames</code>, or to allow full access, use the following syntax: <div style="border: 1px solid black; padding: 5px;"><pre>"Resource": "*"</pre></div> <ul style="list-style-type: none"> For actions that involve a domain's configuration, like <code>esDescribeDomain</code>, you can use the following syntax: <div style="border: 1px solid black; padding: 5px;"><pre>"Resource": "arn:aws:es:<i>region:aws-account-id</i>:domain/<i>domain-name</i>"</pre></div> <ul style="list-style-type: none"> For actions that apply to a domain's subresources, like <code>esESHttpGet</code>, you can use the following syntax: <div style="border: 1px solid black; padding: 5px;"><pre>"Resource": "arn:aws:es:<i>region:aws-account-id</i>:domain/<i>domain-name</i>/*"</pre></div> <p>You don't have to use a wildcard. OpenSearch Service lets you define a different access policy for each OpenSearch index or API. For example, you might limit a user's permissions to the <code>test-index</code> index:</p> <div style="border: 1px solid black; padding: 5px;"><pre>"Resource": "arn:aws:es:<i>region:aws-account-id</i>:domain/<i>domain-name</i>/test-index"</pre></div> <p>Instead of full access to <code>test-index</code>, you might prefer to limit the policy to just the search API:</p> <div style="border: 1px solid black; padding: 5px;"><pre>"Resource": "arn:aws:es:<i>region:aws-account-id</i>:domain/<i>domain-name</i>/test-index/_search"</pre></div> <p>You can even control access to individual documents:</p> <div style="border: 1px solid black; padding: 5px;"><pre>"Resource": "arn:aws:es:<i>region:aws-account-id</i>:domain/<i>domain-name</i>/test-index/test-type/1"</pre></div> <p>Essentially, if OpenSearch 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. 146).</p> <p>For details about which actions support resource-level permissions, see the Action element in this table.</p>

Advanced options and API considerations

OpenSearch Service 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 OpenSearch 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 OpenSearch Service 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.

[OpenSearch Dashboards \(p. 280\)](#) 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 cluster settings" \(p. 21\)](#).

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 indexes or OpenSearch operations, see [the section called "Fine-grained access control" \(p. 146\)](#).

Configuring access policies

- For instructions on creating or modifying resource- and IP-based policies in OpenSearch Service, see [the section called "Configuring access policies" \(p. 21\)](#).
- 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 IAM identity-based policies](#) in the *IAM User Guide*.

AWS managed policies for Amazon OpenSearch Service

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to [create IAM customer managed policies](#) that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see [AWS managed policies](#) in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the `ViewOnlyAccess` AWS managed policy provides read-only access to many AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see [AWS managed policies for job functions](#) in the *IAM User Guide*.

AmazonOpenSearchServiceFullAccess

Grants full access to the OpenSearch Service configuration API operations and resources for an AWS account.

You can find the [AmazonOpenSearchServiceFullAccess](#) policy in the IAM console.

AmazonOpenSearchServiceReadOnlyAccess

Grants read-only access to all OpenSearch Service resources for an AWS account.

You can find the [AmazonOpenSearchServiceReadOnlyAccess](#) policy in the IAM console.

AmazonOpenSearchServiceRolePolicy

You can't attach `AmazonOpenSearchServiceRolePolicy` to your IAM entities. This policy is attached to a service-linked role that allows OpenSearch Service to access account resources. For more information, see [the section called "Service-linked roles" \(p. 188\)](#).

You can find the [AmazonOpenSearchServiceRolePolicy](#) policy in the IAM console.

AmazonOpenSearchServiceCognitoAccess

Provides the minimum Amazon Cognito permissions necessary to enable [Cognito authentication](#) (p. 175).

You can find the [AmazonOpenSearchServiceCognitoAccess](#) policy in the IAM console.

OpenSearch Service updates to AWS managed policies

View details about updates to AWS managed policies for OpenSearch Service since this service began tracking changes.

Change	Description	Date
Updated AmazonOpenSearchServiceRolePolicy and AmazonElasticsearchServiceRolePolicy	<p>Added the permissions necessary for the service-linked role (p. 188) to create OpenSearch Service-managed VPC endpoints (p. 188). Some actions can only be performed when the request contains the tag <code>OpenSearchManaged=true</code>.</p> <p>The deprecated Elasticsearch policy has also been updated to ensure backwards compatibility.</p>	7 November 2022
Updated AmazonOpenSearchServiceRolePolicy and AmazonElasticsearchServiceRolePolicy	<p>Added support for the <code>PutMetricData</code> action, which is required to publish OpenSearch cluster metrics to Amazon CloudWatch.</p> <p>The deprecated Elasticsearch policy has also been updated to ensure backwards compatibility.</p> <p>For the policy JSON, see the IAM console.</p>	12 September 2022
Updated AmazonOpenSearchServiceRolePolicy and AmazonElasticsearchServiceRolePolicy	<p>Added support for the <code>acm</code> resource type. The policy provides the minimum AWS Certificate Manager (ACM) read-only permission necessary for the service-linked role (p. 188) to verify and validate ACM resources in order to create and update custom endpoint (p. 59) enabled domains.</p> <p>The deprecated Elasticsearch policy has also been updated to ensure backwards compatibility.</p>	28 July 2022
Updated AmazonOpenSearchServiceCognitoAccess and AmazonESCognitoAccess	Added support for the <code>UpdateUserPoolClient</code> action, which is required	20 December 2021

Change	Description	Date
	<p>to set Cognito user pool configuration during upgrade from Elasticsearch to OpenSearch.</p> <p>Corrected permissions for the <code>SetIdentityPoolRoles</code> action to allow access to all resources.</p> <p>The deprecated Elasticsearch policy has also been updated to ensure backwards compatibility.</p>	
Updated <code>AmazonOpenSearchServiceRolePolicy</code>	<p>Added support for the <code>Security-group</code> resource type. The policy provides the minimum Amazon EC2 and Elastic Load Balancing permissions necessary for the service-linked role (p. 188) to enable VPC access (p. 175).</p>	9 September 2021
<ul style="list-style-type: none"> Added <code>AmazonOpenSearchServiceFullAccess</code> Deprecated <code>AmazonESFullAccess</code> 	<p>This new policy is meant to replace the old policy. Both policies provide full access to the OpenSearch Service configuration API and all HTTP methods for the OpenSearch APIs. Fine-grained access control (p. 146) and resource-based policies (p. 130) can still restrict access.</p>	7 September 2021
<ul style="list-style-type: none"> Added <code>AmazonOpenSearchServiceReadOnlyAccess</code> Deprecated <code>AmazonESReadOnlyAccess</code> 	<p>This new policy is meant to replace the old policy. Both policies provide read-only access to the OpenSearch Service configuration API (<code>es:Describe*</code>, <code>es>List*</code>, and <code>es:Get*</code>) and <i>no</i> access to the HTTP methods for the OpenSearch APIs.</p>	7 September 2021
<ul style="list-style-type: none"> Added <code>AmazonOpenSearchServiceCognitoAccess</code> Deprecated <code>AmazonESCognitoAccess</code> 	<p>This new policy is meant to replace the old policy. Both policies provide the minimum Amazon Cognito permissions necessary to enable Cognito authentication (p. 175).</p>	7 September 2021

Change	Description	Date
<ul style="list-style-type: none"> Added AmazonOpenSearchServiceRolePolicy (p. 188) Deprecated AmazonElasticsearchServiceRolePolicy 	This new policy is meant to replace the old policy . Both policies provide the minimum Amazon EC2 and Elastic Load Balancing permissions necessary for the service-linked role (p. 188) to enable VPC access (p. 175) .	7 September 2021
Started tracking changes	Amazon OpenSearch Service now tracks changes to AWS-managed policies.	7 September 2021

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit the permissions that Amazon OpenSearch Service gives another service to the resource. If the `aws:SourceArn` value does not contain the account ID, such as an Amazon S3 bucket ARN, you must use both global condition context keys to limit permissions. If you use both global condition context keys and the `aws:SourceArn` value contains the account ID, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement. Use `aws:SourceArn` if you want only one resource to be associated with the cross-service access. Use `aws:SourceAccount` if you want to allow any resource in that account to be associated with the cross-service use.

The value of `aws:SourceArn` must be the ARN of the OpenSearch Service domain.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full ARN of the resource. If you don't know the full ARN of the resource or if you are specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (*) for the unknown portions of the ARN. For example, `arn:aws:es:*:123456789012:*`.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in OpenSearch Service to prevent the confused deputy problem.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ConfusedDeputyPreventionExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "es.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:es:*:123456789012:*
```

```
        "aws:SourceAccount":"123456789012"
    },
    "ArnLike":{
        "aws:SourceArn":"arn:aws:es:region:123456789012:domain/my-domain"
    }
}
}
```

Fine-grained access control in Amazon OpenSearch Service

Fine-grained access control offers additional ways of controlling access to your data on Amazon OpenSearch 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 benefits:

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

Topics

- [The bigger picture: fine-grained access control and OpenSearch Service security \(p. 146\)](#)
- [Key concepts \(p. 149\)](#)
- [Enabling fine-grained access control \(p. 149\)](#)
- [Accessing OpenSearch Dashboards as the master user \(p. 151\)](#)
- [Managing permissions \(p. 152\)](#)
- [Recommended configurations \(p. 156\)](#)
- [Limitations \(p. 157\)](#)
- [Modifying the master user \(p. 158\)](#)
- [Additional master users \(p. 158\)](#)
- [Manual snapshots \(p. 159\)](#)
- [Integrations \(p. 159\)](#)
- [REST API differences \(p. 160\)](#)
- [Tutorial: IAM master user and Amazon Cognito \(p. 161\)](#)
- [Tutorial: Internal user database and HTTP basic authentication \(p. 163\)](#)

The bigger picture: fine-grained access control and OpenSearch Service security

Amazon OpenSearch Service security has three main layers:

Network

The first security layer is the network, which determines whether requests reach an OpenSearch Service 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" \(p. 37\)](#).

Domain access policy

The second security layer is the domain access policy. After a request reaches a domain endpoint, the [resource-based access policy \(p. 130\)](#) 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 OpenSearch itself.

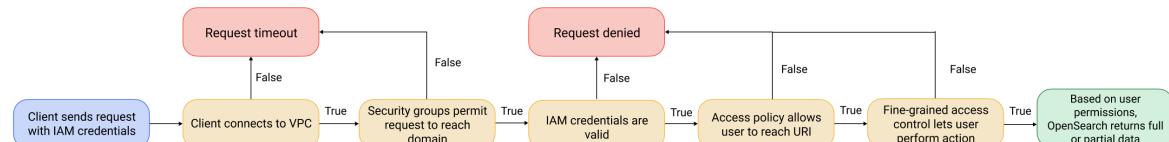
Fine-grained access control

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.

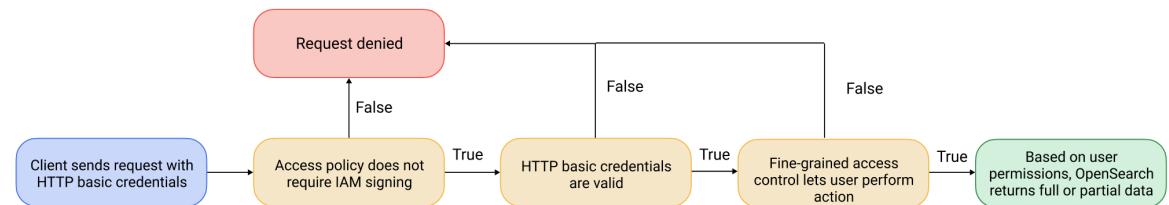
Note

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.

The following 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.



The following 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.



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": [
      {
        "id": "12345",
        "name": "Thor"
      }
    ]
  }
}
```

```
"_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": "User [user] is not authorized to perform [search] operation on index [movies]."
      }
    ],
    "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 Dashboards, 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 OpenSearch 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 OpenSearch 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" \(p. 191\)](#).

We recommend IAM if you want to use the same users on multiple clusters, if you want to use Amazon Cognito to access Dashboards, or if you have OpenSearch 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`, which also supports AWS Signature Version 4 with the [--aws-sigv4 option](#). The internal user database is stored in an OpenSearch 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 Dashboards (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 OpenSearch Service.

Enabling fine-grained access control

Enable fine-grained access control using the console, AWS CLI, or configuration API. For steps, see [Creating and managing domains \(p. 16\)](#).

Fine-grained access control requires OpenSearch or Elasticsearch 6.7 or later. It also requires HTTPS for all traffic to the domain, [Encryption of data at rest \(p. 127\)](#), and [node-to-node encryption \(p. 129\)](#). After you enable fine-grained access control, you can't disable it.

Enabling fine-grained access control on existing domains

You can enable fine-grained access control on existing domains running OpenSearch or Elasticsearch 6.7 or later.

To enable fine-grained access control on an existing domain (console)

1. Select your domain and choose **Actions** and **Edit security configuration**.
2. Select **Enable fine-grained access control**.
3. Choose how to create the master user:
 - If you want to use IAM for user management, choose **Set IAM ARN as master user** and specify the ARN for an IAM role.
 - If you want to use the internal user database, choose **Create master user** and specify a user name and password.
4. (Optional) Select **Enable migration period for open/IP-based access policy**. This setting enables a 30-day transition period during which your existing users can continue to access the domain without disruptions, and existing open and [IP-based access policies \(p. 134\)](#) will continue to work with your domain. During this migration period, we recommend that administrators [create the necessary roles and map them to users \(p. 152\)](#) for the domain. If you use identity-based policies instead of an open or IP-based access policy, you can disable this setting.

You also need to update your clients to work with fine-grained access control during the migration period. For example, if you map IAM users with fine-grained access control, you must update your clients to start signing requests with AWS Signature Version 4. If you configure HTTP basic authentication with fine-grained access control, you must update your clients to provide appropriate basic authentication credentials in requests.

During the migration period, users who access the OpenSearch Dashboards endpoint for the domain will land directly on the **Discover** page rather than the login page. Administrators and master users can choose **Login** to log in with admin credentials and configure role mappings.

Important

OpenSearch Service automatically disables the migration period after 30 days. We recommend ending it as soon as you create the necessary roles and map them to users. After the migration period ends, you can't re-enable it.

5. Choose **Save changes**.

The change triggers a [blue/green deployment \(p. 22\)](#) during which the cluster health becomes red, but all cluster operations remain unaffected.

To enable fine-grained access control on an existing domain (CLI)

Set `AnonymousAuthEnabled` to `true` to enable the migration period with fine-grained access control:

```
aws opensearch update-domain-config --domain-name test-domain --region us-east-1 \
    --advanced-security-options '{ "Enabled": true, "InternalUserDatabaseEnabled":true,
    "MasterUserOptions": { "MasterUserName": "master-username", "MasterUserPassword": "master-
    password" }, "AnonymousAuthEnabled": true}'
```

About the `default_role`

Fine-grained access control requires [role mapping \(p. 154\)](#). If your domain uses [identity-based access policies \(p. 132\)](#), OpenSearch Service automatically maps your IAM users to a new role called `default_role` in order to help you properly migrate existing users. This temporary mapping ensures that

your users can still successfully send IAM-signed GET and PUT requests until you create your own role mappings.

The role does not add any security vulnerabilities or flaws to your OpenSearch Service domain. We recommend deleting the default role as soon as you set up your own roles and map them accordingly.

Migration scenarios

The following table describes the behavior for each authentication method before and after enabling fine-grained access control on an existing domain, and the steps administrators must take to properly map their users to roles:

Authentication method	Before enabling fine-grained access control	After enabling fine-grained access control	Administrator tasks
Identity-based policies	All IAM users satisfying the IAM policy can access the domain.	You don't need to enable the migration period. OpenSearch Service automatically maps all IAM users satisfying the IAM policy to the default_role (p. 150) so they can continue to access the domain.	<ol style="list-style-type: none"> 1. Create custom role mappings on the domain. 2. Delete the default_role.
IP-based policies	All users from the allowed IP addresses or CIDR blocks can access the domain.	During the 30-day migration period, all users from the allowed IP addresses or CIDR blocks can continue to access the domain.	<ol style="list-style-type: none"> 1. Create custom role mappings on the domain. 2. Update your clients to either provide basic authentication credentials or IAM credentials, depending on your role mapping configuration. 3. Disable the migration period. Users from the allowed IP addresses or CIDR blocks sending requests without basic authentication or IAM credentials will lose access to the domain.
Open access policies	All users over internet can access the domain.	During the 30-day migration period, all users over the internet can continue to access to domain.	<ol style="list-style-type: none"> 1. Create role mappings on the domain. 2. Update your clients to either provide basic authentication credentials or IAM credentials, depending on your role mapping configuration. 3. Disable the migration period. Users sending requests without basic authentication or IAM credentials will lose access to the domain.

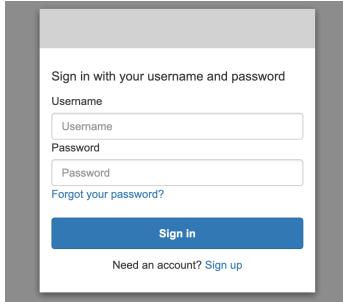
Accessing OpenSearch Dashboards as the master user

Fine-grained access control has an OpenSearch Dashboards plugin that simplifies management tasks. You can use Dashboards to manage users, roles, mappings, action groups, and tenants. The OpenSearch

Dashboards sign-in page and underlying authentication method differs, however, depending on how you manage users and configured your domain.

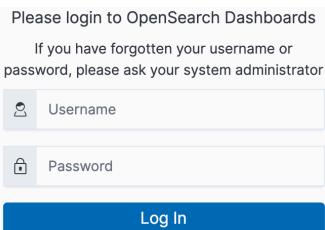
- If you want to use IAM for user management, use [the section called “Amazon Cognito authentication for OpenSearch Dashboards” \(p. 175\)](#) to access Dashboards. Otherwise, Dashboards shows a nonfunctional sign-in page. See [the section called “Limitations” \(p. 157\)](#).

With Amazon Cognito authentication, one of the assumed roles from the 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” \(p. 182\)](#) and [the section called “Tutorial: IAM master user and Amazon Cognito” \(p. 161\)](#).



- If you choose to use the internal user database, you can sign in to Dashboards with your master user name and password. You must access Dashboards over HTTPS. Amazon Cognito and SAML authentication for Dashboards both replace this login screen.

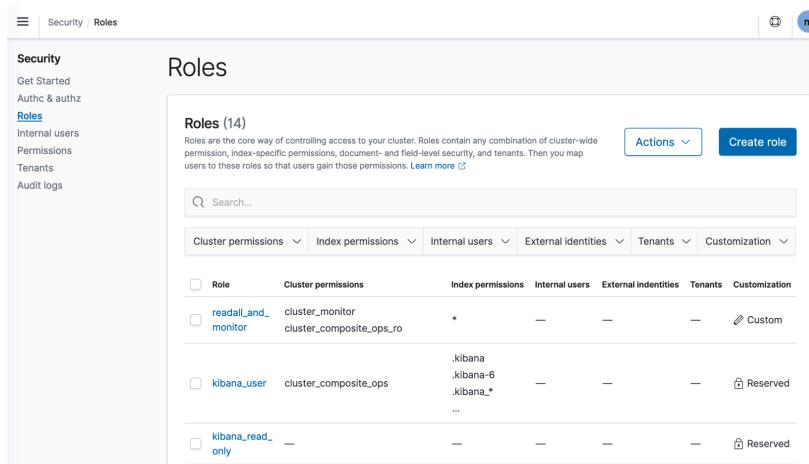
For more information about this configuration, see [the section called “Tutorial: Internal user database and HTTP basic authentication” \(p. 163\)](#).



- If you choose to use SAML authentication, you can sign in using credentials from an external identity provider. For more information, see [the section called “SAML authentication for OpenSearch Dashboards” \(p. 169\)](#).

Managing permissions

As noted in [the section called “Key concepts” \(p. 149\)](#), 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 Dashboards as the master user \(p. 151\)](#) to perform these operations.



The screenshot shows the 'Roles' page in the Amazon OpenSearch Service console. The left sidebar has a 'Security' section with links for 'Get Started', 'Auth & authz', 'Roles', 'Internal users', 'Permissions', 'Tenants', and 'Audit logs'. The main area is titled 'Roles (14)' and contains a table with columns: Role, Cluster permissions, Index permissions, Internal users, External identities, Tenants, and Customization. The table lists three roles:

Role	Cluster permissions	Index permissions	Internal users	External identities	Tenants	Customization
readall_and_monitor	cluster_monitor cluster_composite_ops_ro	*	—	—	—	Custom
kibana_user	cluster_composite_ops	.kibana .kibana-6 .kibana_*	—	—	—	Reserved
kibana_read_only	—	—	—	—	—	Reserved

Note

The permissions that you choose to grant to your users vary widely based on use case. We cannot feasibly cover all scenarios in this documentation. As you're determining which permissions to grant your users, make sure to reference the OpenSearch cluster and index permissions mentioned in the following sections, and always follow the [principle of least privilege](#).

Creating roles

You can create new roles for fine-grained access control using OpenSearch Dashboards or the `_plugins/_security` operation in the REST API. For more information, see [Create roles](#).

Fine-grained access control also includes a number of [predefined roles](#). Clients such as OpenSearch Dashboards and Logstash make a wide variety of requests to OpenSearch, which can make it hard to manually create roles with the minimum set of permissions. For example, the `opensearch_dashboards_user` role includes the permissions that a user needs to work with index patterns, visualizations, dashboards, and tenants. We recommend [mapping it \(p. 154\)](#) to any user or backend role that accesses Dashboards, along with additional roles that allow access to other indices.

Cluster-level security

Cluster-level permissions include the ability to make broad requests such as `_mget`, `_msearch`, and `_bulk`, monitor health, take snapshots, and more. Manage these permissions using the **Cluster Permissions** section when creating a role. For a full list of cluster-level permissions, see [Cluster permissions](#).

Rather than individual permissions, you can often achieve your desired security posture using a combination of the default action groups. For a list of cluster-level action groups, see [Cluster-level](#).

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** section when creating a role. For a full list of index-level permissions, see [Index permissions](#).

Rather than individual permissions, you can often achieve your desired security posture using a combination of the default action groups. For a list of index-level action groups, see [Index-level](#).

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 OpenSearch 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 \(p. 147\)](#).

For more information, see [Document-level security](#).

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 \(p. 147\)](#).

For more information, see [Field-level security](#).

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 \(p. 147\)](#).

Tip

If you apply the standard masking to a field, OpenSearch Service uses a secure, random hash that can cause inaccurate aggregation results. To perform aggregations on masked fields, use pattern-based masking instead.

Creating users

If you enabled the internal user database, you can create users using OpenSearch Dashboards or the `_plugins/_security` operation in the REST API. For more information, see [Create users](#).

If you chose IAM for your master user, ignore this portion of Dashboards. 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.

- 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.

The screenshot shows the 'Map user' page. At the top, there's a breadcrumb navigation: Security / Roles / kibana_user / Map user. The main title is 'Map user'. Below it, a sub-instruction says 'Map users to this role to inherit role permissions. Two types of users are supported: user, and backend role. Learn more'.

Users: A section for internal users. It shows a list with 'new-user' and 'arn:aws:iam::123456789012:user/test-iam-user'. There's a 'Create new internal user' button and a note to look up by user name or enter external users.

Backend roles: A section for mapping external roles. It shows a list with 'arn:aws:iam::123456789012:role/test-iam-role'. There are 'Remove' and 'Add another backend role' buttons.

At the bottom right are 'Cancel' and 'Map' buttons.

You can map roles to users using OpenSearch Dashboards or the `_plugins/_security` operation in the REST API. For more information, see [Map users to roles](#).

Creating action groups

Action groups are sets of permissions that you can reuse across different resources. You can create new action groups using OpenSearch Dashboards or the `_plugins/_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 [Default action groups](#).

OpenSearch Dashboards multi-tenancy

Tenants are spaces for saving index patterns, visualizations, dashboards, and other Dashboards objects. Dashboards multi-tenancy lets you safely share your work with other Dashboards users (or keep it private). You can control which roles have access to a tenant and whether those roles have read or write access. The Global tenant is the default. To learn more, see [OpenSearch Dashboards multi-tenancy](#).

To view your current tenant or change tenants

1. Navigate to OpenSearch Dashboards and sign in.
2. Select your user icon in the upper-right and choose **Switch tenants**.
3. Verify your tenant before creating visualizations or dashboards. If you want to share your work with all other Dashboards users, choose **Global**. To share your work with a subset of Dashboards users, choose a different shared tenant. Otherwise, choose **Private**.

Note

OpenSearch Dashboards maintains a separate index for each tenant, and creates an index template called `tenant_template`. Do not delete or modify the `tenant_template` index, as it could cause OpenSearch Dashboards to malfunction if the tenant index mapping is misconfigured.

Recommended configurations

Due to how fine-grained access control [interacts with other security features \(p. 146\)](#), we recommend several fine-grained access control configurations that work well for most use cases.

Description	Master user	Domain access policy
Use IAM credentials for calls to the OpenSearch APIs, and use SAML authentication (p. 169) to access Dashboards. Manage fine-grained access control roles using Dashboards or the REST API.	IAM user or role	<pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Principal": { "AWS": "*" }, "Action": "es:ESHttp*", "Resource": "domain-arn/*" }] }</pre>
Use IAM credentials or basic authentication for calls to the OpenSearch APIs. Manage fine-grained access control roles using Dashboards or the REST API. This configuration offers a lot of flexibility, especially if you have OpenSearch clients that only support basic authentication. If you have an existing identity provider, use SAML authentication (p. 169) to access Dashboards. Otherwise, manage Dashboards users in the internal user database.	User name and password	<pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Principal": { "AWS": "*" }, "Action": "es:ESHttp*", "Resource": "domain-arn/*" }] }</pre>
Use IAM credentials for calls to the OpenSearch APIs, and use Amazon Cognito to access Dashboards. Manage fine-grained access control roles using Dashboards or the REST API.	IAM user or role	<pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Principal": { "AWS": "*" }, "Action": "es:ESHttp*", "Resource": "domain-arn/*" }] }</pre>

Description	Master user	Domain access policy
Use IAM credentials for calls to the OpenSearch APIs, and block most access to Dashboards. Manage fine-grained access control roles using the REST API.	IAM user or role	<pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Principal": { "AWS": "*" }, "Action": "es:ESHttp*", "Resource": "domain-arn/*" }, { "Effect": "Deny", "Principal": { "AWS": "*" }, "Action": "es:ESHttp*", "Resource": "domain-arn/_dashboards*" }] }</pre>

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.
- If you choose IAM for the master user and don't enable Amazon Cognito or SAML authentication, Dashboards 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 \(p. 245\)](#) 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 \(p. 217\)](#). To deny all access to SQL, add the following element to your domain access policy:

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

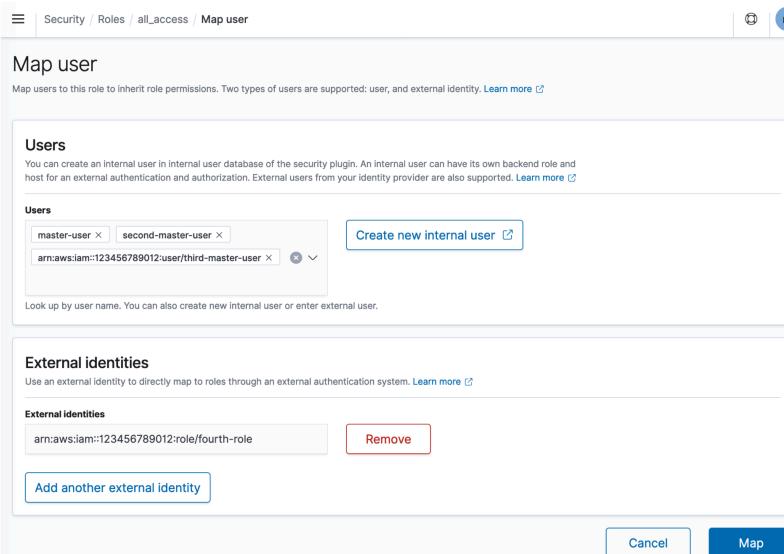
To modify the master user (console)

1. Go to <https://aws.amazon.com>, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Amazon OpenSearch Service**.
3. Choose your domain.
4. Choose **Actions, Edit security configuration**.
5. Choose either **Set IAM ARN as master user** or **Create 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.
 - Switching from the internal user database to an IAM master user does *not* delete any users from the internal user database. Instead, it just disables HTTP basic authentication. Manually delete users from the internal user database, or keep them in case you ever need to reenable HTTP basic authentication.
6. Choose **Save changes**.

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: OpenSearch Dashboards or the REST API.

- In Dashboards, choose **Security, Roles**, and then map the new master user to the `all_access` and `security_manager` roles.



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

```
PUT _plugins/_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 _plugins/_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 Dashboards 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 “Prerequisites” \(p. 43\)](#).

Then use that IAM role to send a signed request to the domain, as outlined in [the section called “Registering a manual snapshot repository” \(p. 45\)](#).

Integrations

If you use [other AWS services \(p. 219\)](#) with OpenSearch Service, 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 Dashboards, [create a role for fine-grained access control \(p. 153\)](#), and [map the IAM role to it \(p. 154\)](#). 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",
                "put"
            ]
        }
    ]
}

```

```
        "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 OpenSearch/Elasticsearch version. Prior to making a PUT request, make a GET request to verify the expected request body. For example, a GET request to `_plugins/_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 OpenSearch or Elasticsearch 7.x, requests look like this (change `_plugins` to `_opendistro` if using Elasticsearch):

```
PUT _plugins/_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 OpenSearch and Elasticsearch 7.x, they're objects with their own URI (change `_plugins` to `_opendistro` if using Elasticsearch)::

```
GET _plugins/_security/api/tenants
```

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

For documentation on the OpenSearch REST API, see the [Security plugin API reference](#).

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/_plugins/_security/api/user'
```

Tutorial: IAM master user and Amazon Cognito

This tutorial covers a popular [fine-grained access control \(p. 146\)](#) use case: an IAM master user with Amazon Cognito authentication for OpenSearch Dashboards. 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.

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 \(p. 16\)](#) with the following settings:

- OpenSearch 1.0 or later, or Elasticsearch 7.8 or later
- 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 enabled for OpenSearch Dashboards. For instructions to enable Cognito authentication and select a user and identity pool, see [the section called "Configuring a domain to use Amazon Cognito" \(p. 178\)](#).
- 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 choose **Roles**.
 3. Choose **IAMMasterUserRole** and go to 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 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. Go to 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 OpenSearch Dashboards.
22. Sign in with **master-user**.
23. Choose **Add sample data** and add some sample flight data.

24. Choose **Security, Roles, Create role**.
25. Name the role new-role.
26. For index permissions, specify opensearch_dashboards_sample_data_fli* for the index pattern (kibana_sample_data_fli* on Elasticsearch domains).
27. For the action group, choose **read**.
28. For **Document level security**, specify the following query:

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

29. For field-level security, choose **Exclude** and specify FlightNum.
30. For **Anonymization**, specify Dest.
31. Choose **Create**.
32. Choose **Mapped users, Manage mapping**. Then add the ARN for IAMLimitedUserRole as an external identity and choose **Map**.
33. Return to the list of roles and choose **opensearch_dashboards_user**. Choose **Mapped users, Manage mapping**. Add the ARN for IAMLimitedUserRole as a backend role and choose **Map**.
34. In a new, private browser window, navigate to Dashboards, sign in using limited-user, and then choose **Explore on my own**.
35. Go to **Dev Tools** and run the default search:

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

Note the permissions error. limited-user doesn't have permissions to run cluster-wide searches.

36. Run another search:

```
GET opensearch_dashboards_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.

37. 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 [fine-grained access control \(p. 146\)](#) use case: a master user in the internal user database and HTTP basic authentication for OpenSearch Dashboards.

To get started with fine-grained access control

1. [Create a domain \(p. 16\)](#) with the following settings:

- OpenSearch 1.0 or later, or Elasticsearch 7.9 or later
- 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 Dashboards *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 OpenSearch Dashboards.
 3. Sign in using TheMasterUser.
 4. Choose Try our sample data.
 5. Add the sample flight data.
 6. Choose Security, Internal users, Create internal user.
 7. Name the user new-user and specify a password. Then choose Create.
 8. Choose Roles, Create role.
 9. Name the role new-role.
 10. For index permissions, specify dashboards_sample_data_fli* for the index pattern.
 11. For the action group, choose read.
 12. For Document level security, specify the following query:

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

13. For field-level security, choose Exclude and specify FlightNum.
14. For Anonymization, specify Dest.
15. Choose Create.
16. Choose Mapped users, Manage mapping. Then add new-user to Users and choose Map.

17. Return to the list of roles and choose **opensearch_dashboards_user**. Choose **Mapped users**, **Manage mapping**. Then add new-user to **Users** and choose **Map**.
18. In a new, private browser window, navigate to Dashboards, sign in using new-user, and then choose **Explore on my own**.
19. Go to **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.

20. Run another search:

```
GET dashboards_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.

21. 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.

Compliance validation for Amazon OpenSearch Service

Third-party auditors assess the security and compliance of Amazon OpenSearch Service as part of multiple AWS compliance programs. These programs 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](#).

If you have compliance requirements, consider using any version of OpenSearch or Elasticsearch 6.0 or later. Earlier versions of Elasticsearch don't offer a combination of [encryption of data at rest \(p. 127\)](#) and [node-to-node encryption \(p. 129\)](#) and are unlikely to meet your needs. You might also consider using any version of OpenSearch or Elasticsearch 6.7 or later if [fine-grained access control \(p. 146\)](#) is important to your use case.

Regardless, choosing a particular OpenSearch or Elasticsearch version when you create a domain does not guarantee compliance. Your compliance responsibility when using OpenSearch Service 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 OpenSearch 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, OpenSearch Service offers several features to help support your data resiliency and backup needs:

- [Multi-AZ domains and replica shards \(p. 34\)](#)
- [Automated and manual snapshots \(p. 42\)](#)

Infrastructure security in Amazon OpenSearch Service

As a managed service, Amazon OpenSearch Service is protected by the AWS global network security procedures that are described in [Amazon Web Services: Overview of Security Processes](#)

You use AWS published API calls to access the OpenSearch Service configuration API through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. To configure the minimum required TLS version to accept, specify the `TLSecurityPolicy` value in the domain endpoint options. For details, see the [AWS CLI Command Reference](#).

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.

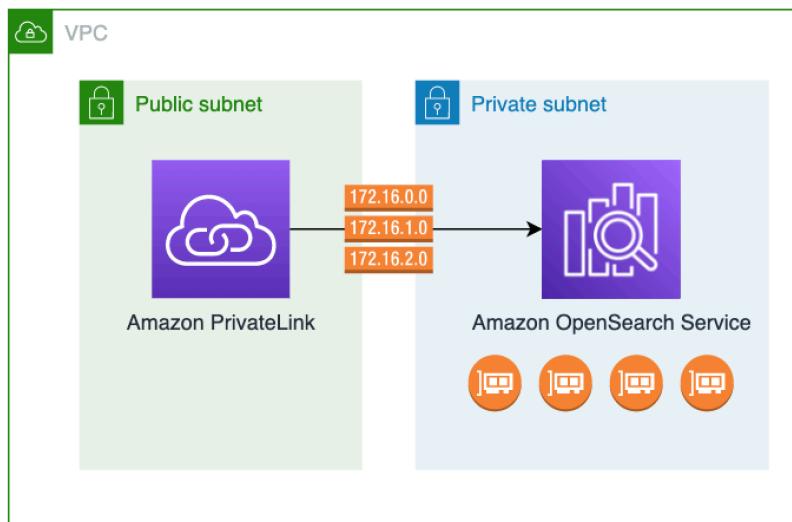
Depending on your domain configuration, you might also need to sign requests to the OpenSearch APIs. For more information, see the section called “[Making and signing OpenSearch Service requests](#)” (p. 135).

OpenSearch Service supports public access domains, which can receive requests from any internet-connected device, and [VPC access domains \(p. 37\)](#), which are isolated from the public internet.

Access Amazon OpenSearch Service using an OpenSearch Service-managed VPC endpoint (AWS PrivateLink)

You can access an Amazon OpenSearch Service domain by setting up an OpenSearch Service-managed VPC endpoint (powered by AWS PrivateLink). These endpoints create a private connection between your VPC and Amazon OpenSearch Service. You can access OpenSearch Service VPC domains as if they were in your VPC, without the use of an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to access OpenSearch Service.

You can configure OpenSearch Service domains to expose additional endpoints running on public or private subnets within the same VPC, different VPC, or different AWS accounts. This enables you to add an additional layer of security to access your domains regardless of where they run, with no infrastructure to manage. The following diagram illustrates OpenSearch Service-managed VPC endpoints within the same VPC:



You establish this private connection by creating an OpenSearch Service-managed *interface VPC endpoint*, powered by AWS PrivateLink. We create an endpoint network interface in each subnet that you enable for the interface VPC endpoint. These are service-managed network interfaces that serve as the entry point for traffic destined for OpenSearch Service. Standard [AWS PrivateLink interface endpoint pricing](#) applies for OpenSearch Service-managed VPC endpoints billed under AWS PrivateLink.

You can create VPC endpoints for domains running all versions of OpenSearch and legacy Elasticsearch. For more information, see [Access AWS services through AWS PrivateLink](#) in the [AWS PrivateLink Guide](#).

Considerations for OpenSearch Service

Before you set up an interface VPC endpoint for OpenSearch Service, review [Considerations](#) in the [AWS PrivateLink Guide](#).

When using OpenSearch Service-managed VPC endpoints, consider the following:

- You can only use interface VPC endpoints to connect to [VPC domains \(p. 37\)](#). Public domains aren't supported.
- HTTPS is the only supported protocol for VPC endpoints. HTTP is not allowed.
- OpenSearch Service supports making calls to all of the [supported OpenSearch API operations \(p. 373\)](#) through an interface VPC endpoint.

- You can configure a maximum of 50 endpoints per account, and a maximum of 10 endpoints per domain. A single domain can have a maximum of 10 [authorized principals \(p. 168\)](#).
- You currently can't use AWS CloudFormation to create interface VPC endpoints.
- You can only create interface VPC endpoints through the OpenSearch Service console or using the [OpenSearch Service configuration API](#). You can't create interface VPC endpoints for OpenSearch Service using the Amazon VPC console.
- OpenSearch Service-managed VPC endpoints aren't accessible from the internet. An OpenSearch Service-managed VPC endpoint is accessible only within the VPC where the endpoint is provisioned or any VPCs peered with the VPC where the endpoint is provisioned, as permitted by the route tables and security groups.
- VPC endpoint policies are not supported for OpenSearch Service. You can associate a security group with the endpoint network interfaces to control traffic to OpenSearch Service through the interface VPC endpoint.

Provide access to a domain

If the VPC that you want to access your domain is in another AWS account, you need to authorize it from the owner's account before you can create an interface VPC endpoint.

To allow a VPC in another AWS account to access your domain

1. Open the Amazon OpenSearch Service console at <https://console.aws.amazon.com/esv3/>.
2. In the navigation pane, choose **Domains** and open the domain that you want to provide access to.
3. Go to the **VPC endpoints** tab, which shows the accounts and corresponding VPCs that have access to your domain.
4. Choose **Authorize principal**.
5. Enter the AWS account ID of the account that will access your domain. This step authorizes the specified account to create VPC endpoints against the domain.
6. Choose **Authorize**.

Create an interface VPC endpoint for a VPC domain

You can create an interface VPC endpoint for OpenSearch Service using either the OpenSearch Service console or the AWS Command Line Interface (AWS CLI).

To create an interface VPC endpoint for an OpenSearch Service domain

1. Open the Amazon OpenSearch Service console at <https://console.aws.amazon.com/esv3/>.
2. In the left navigation pane, choose **VPC endpoints**.
3. Choose **Create endpoint**.
4. Select whether to connect a domain in the current AWS account or another AWS account.
5. Select the domain that you connect to with this endpoint. If the domain is in the current AWS account, use the dropdown to choose the domain. If the domain is in a different account, enter the Amazon Resource Name (ARN) of the domain to connect to. To choose a domain in a different account, the owner needs to [provide you access \(p. 168\)](#) to the domain.
6. For **VPC**, select the VPC from which you'll access OpenSearch Service.
7. For **Subnets**, select one or more subnets from which you'll access OpenSearch Service.
8. For **Security groups**, select the security groups to associate with the endpoint network interfaces. This is a critical step in which you limit what ports, protocols, and sources for inbound traffic that you're authorizing into your endpoint. The security group rules must allow the resources that will use the VPC endpoint to communicate with OpenSearch Service to communicate with the endpoint network interface.

9. Choose **Create endpoint**. The endpoint should be active within 2-5 minutes.

Working with OpenSearch Service-managed VPC endpoints using the configuration API

Use the following API operations to create and manage OpenSearch Service-managed VPC endpoints.

- [CreateVpcEndpoint](#)
- [ListVpcEndpoints](#)
- [UpdateVpcEndpoint](#)
- [DeleteVpcEndpoint](#)

Use the following API operations to manage endpoint access to VPC domains:

- [AuthorizeVpcEndpointAccess](#)
- [ListVpcEndpointAccess](#)
- [ListVpcEndpointsForDomain](#)
- [RevokeVpcEndpointAccess](#)

SAML authentication for OpenSearch Dashboards

SAML authentication for OpenSearch Dashboards lets you use your existing identity provider to offer single sign-on (SSO) for Dashboards on Amazon OpenSearch Service domains running OpenSearch or Elasticsearch 6.7 or later. To use SAML authentication, you must enable [fine-grained access control \(p. 146\)](#).

Rather than authenticating through [Amazon Cognito \(p. 175\)](#) or the [internal user database \(p. 151\)](#), SAML authentication for OpenSearch Dashboards lets you use third-party identity providers to log in to Dashboards, manage fine-grained access control, search your data, and build visualizations. OpenSearch Service supports providers that use the SAML 2.0 standard, such as Okta, Keycloak, Active Directory Federation Services (ADFS), and Auth0.

SAML authentication for Dashboards is only for accessing OpenSearch Dashboards through a web browser. Your SAML credentials do *not* let you make direct HTTP requests to the OpenSearch or Dashboards APIs.

SAML configuration overview

This documentation assumes you have an existing identity provider and some familiarity with it. We can't provide detailed configuration steps for your exact provider, only for your OpenSearch Service domain.

The Dashboards login flow can take one of two forms:

- **Service provider (SP) initiated:** You navigate to Dashboards (for example, `https://my-domain.us-east-1.es.amazonaws.com/_dashboards`), which redirects you to the login screen. After you log in, the identity provider redirects you to Dashboards.
- **Identity provider (IdP) initiated:** You navigate to your identity provider, log in, and choose OpenSearch Dashboards from an application directory.

OpenSearch Service provides two single sign-on URLs, SP-initiated and IdP-initiated, but you only need the one that matches your desired OpenSearch Dashboards login flow.

Regardless of which authentication type you use, the goal is to log in through your identity provider and receive a SAML assertion that contains your user name (required) and any [backend roles \(p. 149\)](#) (optional, but recommended). This information allows [fine-grained access control \(p. 146\)](#) to assign permissions to SAML users. In external identity providers, backend roles are typically called "roles" or "groups."

Note

You can't change the SSO URL from its service-provided value, so SAML authentication for OpenSearch Dashboards does not support proxy servers.

SAML authentication for domains running in a VPC

SAML does not require direct communication between your identity provider and your service provider. Therefore, even if your OpenSearch domain is hosted within a private VPC, you can still use SAML as long as your browser can communicate with both your OpenSearch cluster and your identity provider. Your browser essentially acts as the intermediary between your identity provider and your service provider. For a useful diagram that explains the SAML authentication flow, see the [Okta documentation](#).

Enabling SAML authentication

You can only enable SAML authentication for OpenSearch Dashboards on existing domains, not during the creation of new ones. Due to the size of the IdP metadata file, we highly recommend using the AWS console.

Domains only support one Dashboards authentication method at a time. If you have [Amazon Cognito authentication for OpenSearch Dashboards \(p. 175\)](#) enabled, you must disable it before you can enable SAML.

Enable either SP-initiated or IdP-initiated authentication

These steps explain how to enable SAML authentication with SP-initiated or IdP-initiated authentication for OpenSearch Dashboards. For the extra step required to enable both, see [Enable both SP- and IdP-initiated authentication \(p. 173\)](#).

To enable SAML authentication for Dashboards

1. In the OpenSearch Service console, select the domain, then choose **Actions** and **Edit security configuration**.
2. Select **Enable SAML authentication**.
3. Note the service provider entity ID and the two SSO URLs. You only need one of the SSO URLs. For guidance, see [the section called "SAML configuration overview" \(p. 169\)](#).

Tip

These URLs change if you later enable a [custom endpoint \(p. 59\)](#) for your domain. In that situation, you must update your IdP.

4. Use these values to configure your identity provider. This is the most complex part of the process, and unfortunately, terminology and steps vary wildly by provider. Consult your provider's documentation.

In Okta, for example, you create a "SAML 2.0 web application." For **Single sign on URL**, specify the SSO URL that you chose in step 3. For **Audience URI (SP Entity ID)**, specify the SP entity ID.

Rather than users and backend roles, Okta has users and groups. For **Group Attribute Statements**, we recommend adding `role` to the **Name** field and the regular expression `.+` to the **Filter** field. This statement tells the Okta identity provider to include all user groups under the `role` field of the SAML assertion after a user authenticates.

In Auth0, you create a "regular web application" and then enable the SAML 2.0 add-on. In Keycloak, you create a "client."

5. After you configure your identity provider, it generates an IdP metadata file. This XML file contains information on the provider, such as a TLS certificate, single sign-on endpoints, and the identity provider's entity ID.

Copy the contents of the IdP metadata file and paste it into the **Metadata from IdP** field in the OpenSearch Service console. Alternately, choose **Import from XML file** and upload the file. The metadata file should look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor entityID="entity-id">
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
    <md:IDPSSODescriptor WantAuthnRequestsSigned="false">
      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <md:KeyDescriptor use="signing">
          <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:X509Data>
              <ds:X509Certificate>tls-certificate</ds:X509Certificate>
            </ds:X509Data>
          </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</md:NameIDFormat>
        <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
        <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="idp-sso-url"/>
        <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="idp-sso-url"/>
      </md:IDPSSODescriptor>
    </md:EntityDescriptor>
```

6. Copy the value of the entityID property from your metadata file and paste it into the **IdP entity ID** field in the OpenSearch Service console. Many identity providers also display this value as part of a post-configuration summary. Some providers call it the "issuer".
7. Provide a **SAML master username** and/or a **SAML master backend role**. This username and/or backend role receives full permissions to the cluster, equivalent to a [new master user \(p. 158\)](#), but can only use those permissions within Dashboards.

In Okta, for example, you might have a user jdoe who belongs to the group admins. If you add jdoe to the **SAML master username** field, only that user receives full permissions. If you add admins to the **SAML master backend role** field, any user who belongs to the admins group receives full permissions.

The contents of the SAML assertion must exactly match the strings that you use for the SAML master username and/or SAML master role. Some identity providers add a prefix before their usernames, which can cause a hard-to-diagnose mismatch. In the identity provider user interface, you might see jdoe, but the SAML assertion might contain auth0|jdoe. Always use the string from the SAML assertion.

Many identity providers let you view a sample assertion during the configuration process, and tools like [SAML-tracer](#) can help you examine and troubleshoot the contents of real assertions. Assertions look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2 Assertion ID="id67229299299259351343340162" IssueInstant="2020-09-22T22:03:08.633Z" Version="2.0" xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
```

```

<saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">idp-issuer</saml2:Issuer>
  <saml2:Subject>
    <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">username</saml2:NameID>
    <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
      <saml2:SubjectConfirmationData NotOnOrAfter="2020-09-22T22:08:08.816Z"
        Recipient="domain-endpoint/_dashboards/_opendistro/_security/saml/acs">
    </saml2:SubjectConfirmation>
  </saml2:Subject>
  <saml2:Conditions NotBefore="2020-09-22T21:58:08.816Z"
    NotOnOrAfter="2020-09-22T22:08:08.816Z">
    <saml2:AudienceRestriction>
      <saml2:Audience>domain-endpoint</saml2:Audience>
    </saml2:AudienceRestriction>
  </saml2:Conditions>
  <saml2:AuthnStatement AuthnInstant="2020-09-22T19:54:37.274Z">
    <saml2:AuthnContext>

      <saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</saml2:AuthnContextClassRef>
      </saml2:AuthnContext>
    </saml2:AuthnStatement>
    <saml2:AttributeStatement>
      <saml2:Attribute Name="role" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="xs:string">GroupName Match Matches regex ".+" (case-sensitive)
        </saml2:AttributeValue>
      </saml2:Attribute>
    </saml2:AttributeStatement>
  </saml2:Assertion>

```

8. (Optional) Expand **Additional settings**.
 9. Leave the **Subject key** field empty to use the NameID element of the SAML assertion for the username. If your assertion doesn't use this standard element and instead includes the username as a custom attribute, specify that attribute here.
- If you want to use backend roles (recommended), specify an attribute from the assertion in the **Role key** field, such as `role` or `group`. This is another situation in which tools like [SAML-tracer](#) can help.
10. By default, OpenSearch Dashboards logs users out after 24 hours. You can configure this value to any number between 60 and 1,440 (24 hours) by specifying the **Session time to live**.
 11. Choose **Save changes**. The domain enters a processing state for approximately one minute, during which time Dashboards is unavailable.
 12. After the domain finishes processing, open OpenSearch Dashboards.
 - If you chose the SP-initiated URL, navigate to *domain-endpoint/_dashboards*. To log in to a specific tenant directly, append `?security_tenant=tenant-name` to the URL.
 - If you chose the IdP-initiated URL, navigate to your identity provider's application directory.

In both cases, log in as either the SAML master user or a user who belongs to the SAML master backend role. To continue the example from step 7, log in as either `jdoe` or a member of the `admins` group.

13. After Dashboards loads, choose **Security and Roles**.
14. [Map roles \(p. 154\)](#) to allow other users to access Dashboards.

For example, you might map your trusted colleague `jroe` to the `all_access` and `security_manager` roles. You might also map the backend role `analysts` to the `readall` and `kibana_user` roles.

If you prefer to use the API rather than Dashboards, see the following sample request:

```
PATCH _plugins/_security/api/rolesmapping
[
  {
    "op": "add", "path": "/security_manager", "value": { "users": ["master-user", "jdoe", "jroe"], "backend_roles": ["admins"] }
  },
  {
    "op": "add", "path": "/all_access", "value": { "users": ["master-user", "jdoe", "jroe"], "backend_roles": ["admins"] }
  },
  {
    "op": "add", "path": "/readall", "value": { "backend_roles": ["analysts"] }
  },
  {
    "op": "add", "path": "/kibana_user", "value": { "backend_roles": ["analysts"] }
  }
]
```

Enable both SP- and IdP-initiated authentication

If you want to configure both SP- and IdP-initiated authentication, you must do so through your identity provider. For example, in Okta, you can perform the following steps:

1. Within your SAML application, go to **General, SAML settings**.
2. For the **Single sign on URL**, provide your *IdP*-initiated SSO URL. For example, `https://search-domain-hash/_dashboards/_opendistro/_security/saml/idpinitiated`.
3. Enable **Allow this app to request other SSO URLs**.
4. Under **Requestable SSO URLs**, add one or more *SP*-initiated SSO URLs. For example, `https://search-domain-hash/_dashboards/_opendistro/_security/saml/acs`.

Enable SAML authentication (AWS CLI)

The following AWS CLI command enables SAML authentication for OpenSearch Dashboards on an existing domain:

```
aws opensearch update-domain-config \
--domain-name my-domain \
--advanced-security-options '{"SAMLOptions":{"Enabled":true,"MasterUserName":"my-idp-user","MasterBackendRole":"my-idp-group-or-role","Idp":{"EntityId":"entity-id","MetadataContent":"metadata-content-with-quotes-escaped"}},"RolesKey":"optional-roles-key","SessionTimeoutMinutes":180,"SubjectKey":"optional-subject-key"}'
```

You must escape all quotes and newline characters in the metadata XML. For example, use `<KeyDescriptor use=\\"signing\\>\n` instead of `<KeyDescriptor use="signing">` and a line break. For detailed information about using the AWS CLI, see the [AWS CLI Command Reference](#).

Enable SAML authentication (configuration API)

The following request to the configuration API enables SAML authentication for OpenSearch Dashboards on an existing domain:

```
POST https://es.us-east-1.amazonaws.com/2021-01-01/opensearch/domain/my-domain/config
{
    "AdvancedSecurityOptions": {
        "SAMLOptions": {
            "Enabled": true,
            "MasterUserName": "my-idp-user",
            "MasterBackendRole": "my-idp-group-or-role",
            "Idp": {
                "EntityId": "entity-id",
                "MetadataContent": "metadata-content-with-quotes-escaped"
            },
            "RolesKey": "optional-roles-key",
            "SessionTimeoutMinutes": 180,
            "SubjectKey": "optional-subject-key"
        }
    }
}
```

You must escape all quotes and newline characters in the metadata XML. For example, use <KeyDescriptor use=\\\"signing\\\">\\n instead of <KeyDescriptor use="signing"> and a line break. For detailed information about using the configuration API, see the [OpenSearch Service API reference](#).

SAML troubleshooting

Error	Details
Your request: ' <i>/some/path</i> ' is not allowed.	Verify that you provided the correct SSO URL (p. 170) (step 3) to your identity provider.
Please provide valid identity provider metadata document to enable SAML.	Your IdP metadata file does not conform to the SAML 2.0 standard. Check for errors using a validation tool.
SAML configuration options aren't visible in the console.	Update to the latest service software (p. 29) .
SAML configuration error: Something went wrong while retrieving the SAML configuration, please check your settings.	This generic error can occur for many reasons. <ul style="list-style-type: none"> Check that you provided your identity provider with the correct SP entity ID and SSO URL. Regenerate the IdP metadata file, and verify the IdP entity ID. Add any updated metadata in the AWS console. Verify that your domain access policy allows access to OpenSearch Dashboards and _plugins/_security/*. In general, we recommend an open access policy for domains that use fine-grained access control. Consult your identity provider's documentation for steps on configuring SAML.
Missing role: No roles available for this user, please contact your system administrator.	You successfully authenticated, but the username and any backend roles from the SAML assertion are not

Error	Details
	<p>mapped to any roles and thus have no permissions. These mappings are case-sensitive.</p> <p>Verify the contents of your SAML assertion using a tool like SAML-tracer and your role mapping using the following call:</p> <pre>GET _plugins/_security/api/rolesmapping</pre>
Your browser continuously redirects or receives HTTP 500 errors when trying to access OpenSearch Dashboards.	These errors can occur if your SAML assertion contains a large number of roles totaling approximately 1,500 characters. For example, if you pass 80 roles, the average length of which is 20 characters, you might exceed the size limit for cookies in your web browser.
You can't log out of ADFS.	ADFS requires all logout request to be signed, which OpenSearch Service doesn't support. Remove <code><SingleLogoutService /></code> from the IdP metadata file to force OpenSearch Service to use its own internal logout mechanism.

Disabling SAML authentication

To disable SAML authentication for OpenSearch Dashboards (console)

1. Choose the domain, **Actions**, and **Edit security configuration**.
2. Uncheck **Enable SAML authentication**.
3. Choose **Save changes**.
4. After the domain finishes processing, verify the fine-grained access control role mapping with the following request:

```
GET _plugins/_security/api/rolesmapping
```

Disabling SAML authentication for Dashboards does *not* remove the mappings for the SAML master username and/or the SAML master backend role. If you want to remove these mappings, log in to Dashboards using the internal user database (if enabled), or use the API to remove them:

```
PUT _plugins/_security/api/rolesmapping/all_access
{
  "users": [
    "master-user"
  ]
}
```

Configuring Amazon Cognito authentication for OpenSearch Dashboards

You can authenticate and protect your Amazon OpenSearch Service default installation of OpenSearch Dashboards using [Amazon Cognito](#). Amazon Cognito authentication is optional and available only

for domains using OpenSearch or Elasticsearch 5.1 or later. If you don't configure Amazon Cognito authentication, you can still protect Dashboards using an [IP-based access policy \(p. 134\)](#) and a [proxy server \(p. 280\)](#), HTTP basic authentication, or [SAML \(p. 169\)](#).

Much of the authentication process occurs in Amazon Cognito, but this section offers guidelines and requirements for configuring Amazon Cognito resources to work with OpenSearch Service domains. [Standard pricing](#) applies to all Amazon Cognito resources.

Tip

The first time you configure a domain to use Amazon Cognito authentication for Dashboards, 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 \(p. 176\)](#)
- [Configuring a domain to use Amazon Cognito \(p. 178\)](#)
- [Allowing the authenticated role \(p. 180\)](#)
- [Configuring identity providers \(p. 181\)](#)
- [\(Optional\) Configuring granular access \(p. 182\)](#)
- [\(Optional\) Customizing the sign-in page \(p. 184\)](#)
- [\(Optional\) Configuring advanced security \(p. 184\)](#)
- [Testing \(p. 184\)](#)
- [Limits \(p. 184\)](#)
- [Common configuration issues \(p. 185\)](#)
- [Disabling Amazon Cognito authentication for OpenSearch Dashboards \(p. 187\)](#)
- [Deleting domains that use Amazon Cognito authentication for OpenSearch Dashboards \(p. 187\)](#)

Prerequisites

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

- Amazon Cognito [user pool](#)
- Amazon Cognito [identity pool](#)
- IAM role that has the `AmazonOpenSearchServiceCognitoAccess` policy attached (`CognitoAccessForAmazonOpenSearch`)

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 Dashboards to multiple OpenSearch Service domains. To learn more, see [the section called "Limits" \(p. 184\)](#).

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 to create a user pool, see [Create a User Pool](#) in the *Amazon Cognito Developer Guide*.

When you create a user pool to use with OpenSearch Service, consider the following:

- Your Amazon Cognito user pool must have a [domain name](#). OpenSearch Service uses this domain name to redirect users to a login page for accessing Dashboards. 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 OpenSearch Service, 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 OpenSearch Service, 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 OpenSearch Service domain \(p. 178\)](#), Amazon Cognito disables this setting.
- You don't need to add [external identity providers](#) to the identity pool. When you configure OpenSearch Service 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 OpenSearch Service, make note of the ID.

About the `CognitoAccessForAmazonOpenSearch` role

OpenSearch Service needs permissions to configure the Amazon Cognito user and identity pools and use them for authentication. You can use `AmazonOpenSearchServiceCognitoAccess`, which is an AWS-managed policy, for this purpose. `AmazonESCognitoAccess` is a legacy policy that was replaced by `AmazonOpenSearchServiceCognitoAccess` when Amazon Elasticsearch Service was renamed to Amazon OpenSearch Service. Both policies provide the minimum Amazon Cognito permissions necessary to enable [Cognito authentication \(p. 175\)](#). For the policy JSON, see the [IAM console](#).

If you use the console to create or configure your OpenSearch Service domain, it creates an IAM role for you and attaches the `AmazonOpenSearchServiceCognitoAccess` policy (or the `AmazonESCognitoAccess` policy if it's an Elasticsearch domain) to the role. The default name for this role is `CognitoAccessForAmazonOpenSearch`.

The role permissions policies `AmazonOpenSearchServiceCognitoAccess` and `AmazonESCognitoAccess` both allow OpenSearch Service to complete the following actions on all identity and user pools:

- Action: `cognito-identity:DescribeUserPool`

- Action: cognito-idp>CreateUserPoolClient
 - Action: cognito-idp>DeleteUserPoolClient
 - Action: cognito-idp>UpdateUserPoolClient
 - Action: cognito-idp>DescribeUserPoolClient
 - Action: cognito-idp>AdminInitiateAuth
 - Action: cognito-idp>AdminUserGlobalSignOut
 - Action: cognito-idp>ListUserPoolClients
 - Action: cognito-identity>DescribeIdentityPool
 - Action: cognito-identity>SetIdentityPoolRoles
 - Action: cognito-identity>GetIdentityPoolRoles

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 OpenSearch Service domain. The role must have the following trust relationship:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "opensearchservice.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 a domain to use Amazon Cognito

After you complete the prerequisites, you can configure an OpenSearch Service domain to use Amazon Cognito for Dashboards.

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 OpenSearch Service.

Configuring Amazon Cognito authentication (console)

Because it creates the [CognitoAccessForAmazonOpenSearch](#) (p. 177) role for you, the console offers the simplest configuration experience. In addition to the standard OpenSearch Service permissions, you need the following set of permissions to use the console to create a domain that uses Amazon Cognito authentication for OpenSearch Dashboards.

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": [  
            "ec2:DescribeVpcs",
```

```
        "cognito-identity>ListIdentityPools",
        "cognito-idp>ListUserPools",
        "iam>CreateRole",
        "iam:AttachRolePolicy"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "iam:GetRole",
        "iam:PassRole"
    ],
    "Resource": "arn:aws:iam::123456789012:role/service-
role/CognitoAccessForAmazonOpenSearch"
}
]
```

For instructions to add permissions to an identity (user, user group, or role), see [Adding IAM identity permissions \(console\)](#).

If `CognitoAccessForAmazonOpenSearch` (p. 177) 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/service-
role/CognitoAccessForAmazonOpenSearch"
        }
    ]
}
```

To configure Amazon Cognito authentication for Dashboards (console)

1. Go to <https://aws.amazon.com>, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Amazon OpenSearch Service**.
3. Under **Domains**, select the domain you want to configure.
4. Choose **Actions, Edit security configuration**.
5. Select **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” \(p. 176\)](#).
8. For **Cognito identity pool**, select an identity pool or create one. For guidance, see [the section called “About the identity pool” \(p. 177\)](#).

Note

The **Create user pool** and **Create 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" \(p. 176\)](#).

9. For **IAM role name**, use the default value of `CognitoAccessForAmazonOpenSearch` (recommended) or enter a new name. To learn more about the purpose of this role, see [the section called "About the CognitoAccessForAmazonOpenSearch role" \(p. 177\)](#).
10. Choose **Save changes**.

After your domain finishes processing, see [the section called "Allowing the authenticated role" \(p. 180\)](#) and [the section called "Configuring identity providers" \(p. 181\)](#) for additional configuration steps.

Configuring Amazon Cognito authentication (AWS CLI)

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

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

Example

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

```
aws opensearch create-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/*"}]}' --engine-version "OpenSearch_1.0" --cluster-config InstanceType=m4.xlarge.search,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/CognitoAccessForAmazonOpenSearch"
```

After your domain finishes processing, see [the section called "Allowing the authenticated role" \(p. 180\)](#) and [the section called "Configuring identity providers" \(p. 181\)](#) 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 OpenSearch Service API Reference](#), including the `CognitoOptions` parameter for the `CreateDomain` and `UpdateDomainConfig` 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" \(p. 180\)](#) and [the section called "Configuring identity providers" \(p. 181\)](#) 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" \(p. 177\)](#) does not have the necessary privileges to access OpenSearch Dashboards. You must provide the role with additional permissions.

Important

If you configured [fine-grained access control \(p. 146\)](#) and use an "open" or IP-based access policy, you can skip this step.

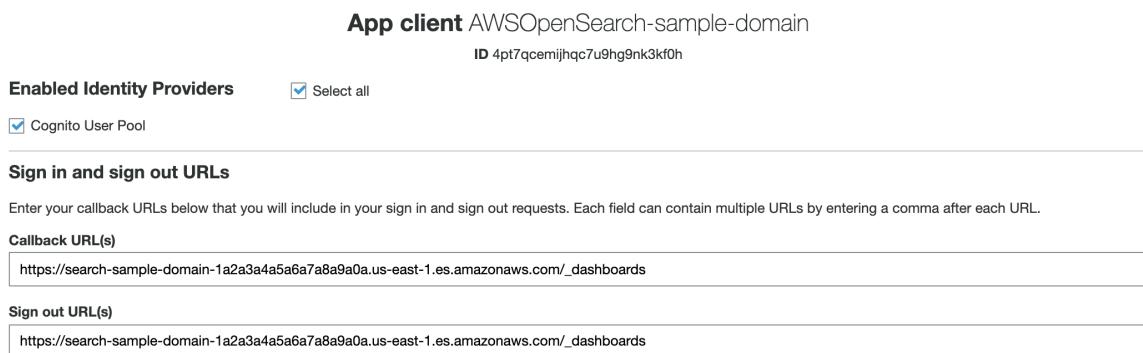
You can include these permissions in an [identity-based \(p. 132\)](#) policy, but unless you want authenticated users to have access to all OpenSearch Service domains, a [resource-based \(p. 130\)](#) policy attached to a single domain is the better approach:

```
{  
    "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 OpenSearch Service domain, see the section called ["Configuring access policies" \(p. 21\)](#).

Configuring identity providers

When you configure a domain to use Amazon Cognito authentication for Dashboards, OpenSearch Service 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.

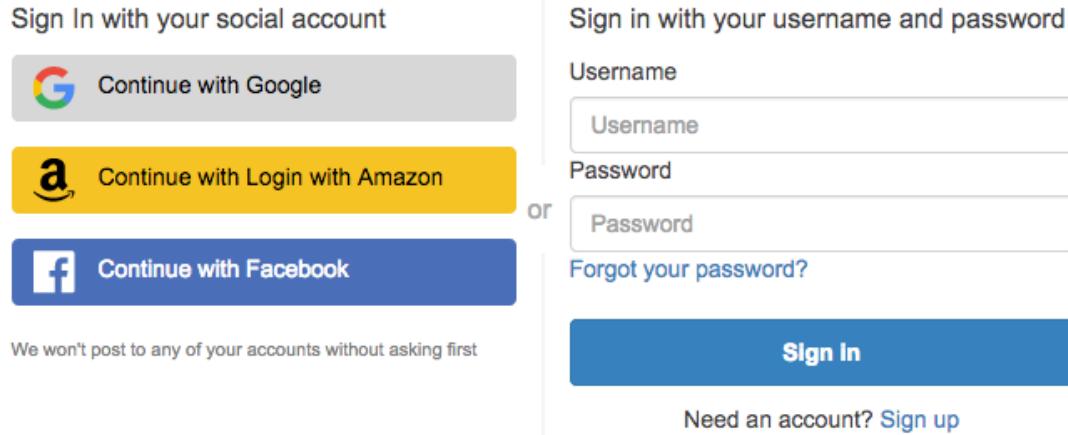


Warning

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.



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 OpenSearch Service 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_ *identitypool*Auth_Role), which means that every user can access the same AWS resources. If you want to use [fine-grained access control \(p. 146\)](#) 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” \(p. 161\)](#).

▼ 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.](#)

The screenshot shows the AWS Cognito Identity Pool configuration interface. At the top, there are tabs for Cognito, Amazon, Facebook, Google+, Twitter / Digits, OpenID, SAML, and Custom. The Custom tab is selected. Below the tabs, there is a note: "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." Two input fields are shown: "User Pool ID" with value "us-east-1_FtOMZ3OEa" and "App client id" with value "tb2cdfp327go1e1qro2gtv91p". Each field has an "Unlock" button next to it. Below each field is an example: "ex: us-east-1_Ab129faBb" and "ex: 7lhkkfbfb4q5kpp90urffao". A dropdown menu titled "Authenticated role selection" is open, showing three options: "Use default role", "Choose role with rules", and "Choose role from token". The "Use default role" option is highlighted with a blue border.

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. Select **Choose role from token**, then choose either **Use default Authenticated role** or **DENY** to specify how the identity pool handles users who aren't 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're satisfied with your configuration, verify that the user experience meets your expectations.

To access OpenSearch Dashboards

1. Navigate to `https://opensearch-domain/_dashboards` in a web browser. To log in to a specific tenant directly, append `?security_tenant=tenant-name` to the URL.
2. Sign in using your preferred credentials.
3. After OpenSearch Dashboards loads, configure at least one index pattern. Dashboards uses these patterns to identify 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. 185) for troubleshooting information.

Limits

Amazon Cognito has soft limits on many of its resources. If you want to enable Dashboards authentication for a large number of OpenSearch Service domains, review [Limits in Amazon Cognito](#) and [request limit increases](#) as necessary.

Each OpenSearch Service domain adds an [app client](#) to the user pool, which adds an [authentication provider](#) to the identity pool. If you enable OpenSearch Dashboards 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 OpenSearch Service domains that you try to configure to use Amazon Cognito authentication for Dashboards can get stuck in a configuration state of **Processing**.

Common configuration issues

The following tables list common configuration issues and solutions.

Configuring OpenSearch Service

Issue	Solution
OpenSearch Service 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. 178) .
User is not authorized to perform: <code>iam:PassRole</code> on resource <code>CognitoAccessForAmazonOpenSearch</code> (console)	You don't have <code>iam:PassRole</code> permissions for the CognitoAccessForAmazonOpenSearch (p. 177) role. Attach the following policy to your account: <pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["iam:PassRole"], "Resource": "arn:aws:iam::123456789012:role/service-role/CognitoAccessForAmazonOpenSearch" }] }</pre> <p>Alternately, you can attach the <code>IAMFullAccess</code> policy.</p>
User is not authorized to perform: <code>cognito-identity>ListIdentityPools</code> on resource	You don't have read permissions for Amazon Cognito. Attach the <code>AmazonCognitoReadOnly</code> policy to your account.
An error occurred (<code>ValidationException</code>) when calling the <code>CreateDomain</code> operation: OpenSearch Service must be allowed to use the passed role	OpenSearch Service isn't specified in the trust relationship of the <code>CognitoAccessForAmazonOpenSearch</code> role. Check that your role uses the trust relationship that is specified in the section called "About the CognitoAccessForAmazonOpenSearch role" (p. 177) . Alternately, use the console to configure Amazon Cognito authentication. The console creates a role for you.
An error occurred (<code>ValidationException</code>) when calling the <code>CreateDomain</code> operation: User is not authorized to perform: <code>cognito-idp:<i>action</i></code> on resource: <code>user pool</code>	The role specified in <code>--cognito-options</code> does not have permissions to access Amazon Cognito. Check that the role has the AWS managed <code>AmazonOpenSearchServiceCognitoAccess</code> 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 CreateDomain operation: User pool does not exist	<p>OpenSearch Service 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:</p> <pre>aws cognito-idp list-user-pools --max-results 60 --region <i>region</i></pre>
An error occurred (ValidationException) when calling the CreateDomain operation: IdentityPool not found	<p>OpenSearch Service 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:</p> <pre>aws cognito-identity list-identity-pools --max-results 60 --region <i>region</i></pre>
An error occurred (ValidationException) when calling the CreateDomain operation: Domain needs to be specified for user pool	<p>The user pool does not have a domain name. You can configure one using the Amazon Cognito console or the following AWS CLI command:</p> <pre>aws cognito-idp create-user-pool-domain --domain <i>name</i> --user-pool-id <i>id</i></pre>

Accessing OpenSearch Dashboards

Issue	Solution
The login page doesn't show my preferred identity providers.	Check that you enabled the identity provider for the OpenSearch Service app client as specified in the section called "Configuring identity providers" (p. 181) .
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. 184) .
My login credentials don't work.	<p>Check that you have configured the identity provider as specified in the section called "Configuring identity providers" (p. 181).</p> <p>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.</p>
OpenSearch Dashboards 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 Dashboards. Check that you added an access policy as specified in the section called "Allowing the authenticated role" (p. 180) .
Invalid identity pool configuration. Check assigned IAM roles for this pool.	<p>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:</p> <pre>{ "Version": "2012-10-17",</pre>

Issue	Solution
	<pre> "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" } } }] } </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 Dashboards in a new browser session.

Disabling Amazon Cognito authentication for OpenSearch Dashboards

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

To disable Amazon Cognito authentication for Dashboards (console)

1. Go to <https://aws.amazon.com>, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Amazon OpenSearch Service**.
3. In the navigation pane, under **Domains**, choose the domain you want to configure.
4. Choose **Actions, Edit security configuration**.
5. Deselect **Enable Amazon Cognito authentication**.
6. Choose **Save changes**.

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 OpenSearch Dashboards

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

Using service-linked roles for Amazon OpenSearch Service

Provide Amazon OpenSearch Service access to resources in your AWS account using [service-linked roles](#). A service-linked role is a unique type of AWS Identity and Access Management (IAM) role that's linked directly to OpenSearch Service. Service-linked roles are predefined by OpenSearch Service and include all the permissions the service requires to call other AWS services on your behalf. Amazon OpenSearch Service uses a service-linked role called **AWSServiceRoleForAmazonOpenSearchService**.

A service-linked role makes setting up OpenSearch Service easier because you don't have to manually add the necessary permissions. OpenSearch Service defines the permissions of its service-linked roles, and unless defined otherwise, only OpenSearch Service 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 OpenSearch Service resources because you can't inadvertently remove permission to access the resources.

For a list of all 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.

Legacy Elasticsearch service-linked role

Amazon OpenSearch Service uses a service-linked role called **AWSServiceRoleForAmazonOpenSearchService**. Your accounts might also contain a legacy service-linked role called **AWSServiceRoleForAmazonElasticsearchService**, which works with the deprecated Elasticsearch API endpoints.

If the legacy Elasticsearch role doesn't exist in your account, OpenSearch Service automatically creates a new OpenSearch service-linked role the first time you create an OpenSearch domain. Otherwise your account continues to use the Elasticsearch role. In order for this automatic creation to succeed, you must have permissions for the `iam:CreateServiceLinkedRole` action.

Permissions

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

- `opensearchservice.amazonaws.com`

The role permissions policy named **AmazonOpenSearchServiceRolePolicy** allows OpenSearch Service 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 *
- Action: `ec2:DescribeVpcs` on *
- Action: `elasticloadbalancing:AddListenerCertificates` on *

- Action: elasticloadbalancing:RemoveListenerCertificates on *
- Action: ec2:CreateTags on all network interfaces and VPC endpoints
- Action: ec2:DescribeTags on *
- Action: acm:DescribeCertificate on *
- Action: cloudwatch:PutMetricData on *
- Action: ec2>CreateVpcEndpoint on all VPCs, security groups, subnets, and route tables, as well as all VPC endpoints when the request contains the tag OpenSearchManaged=true
- Action: ec2:ModifyVpcEndpoint on all VPCs, security groups, subnets, and route tables, as well as all VPC endpoints when the request contains the tag OpenSearchManaged=true
- Action: ec2>DeleteVpcEndpoints on all endpoints when the request contains the tag OpenSearchManaged=true
- Action: ec2:DescribeVpcEndpoints 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

You don't need to manually create a service-linked role. When you create a VPC access domain using the AWS Management Console, OpenSearch Service creates the service-linked role for you. In order for this automatic creation to succeed, you must have permissions for the iam:CreateServiceLinkedRole action.

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

OpenSearch Service doesn't let you edit the AWSServiceRoleForAmazonOpenSearchService 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

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 AWSServiceRoleForAmazonOpenSearchService 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're unsure whether OpenSearch Service is using the `AWSServiceRoleForAmazonOpenSearchService` role, you can try to delete the role. If the service is using the role, then the deletion fails and you can view the resources using the role. If the role is being used, then you must wait for the session to end before you can delete the role, and/or delete the resources using the role. You cannot revoke the session for a service-linked role.

Manually deleting a service-linked role

Delete service-linked roles from the IAM console, API, or AWS CLI. For instructions, see [Deleting a service-linked role](#) in the *IAM User Guide*.

Sample code for Amazon OpenSearch Service

This chapter contains common sample code for working with Amazon OpenSearch Service: HTTP request signing in a variety of programming languages, compressing HTTP request bodies, and using the AWS SDKs to create domains.

Topics

- [Elasticsearch client compatibility \(p. 191\)](#)
- [Signing HTTP requests to Amazon OpenSearch Service \(p. 191\)](#)
- [Compressing HTTP requests in Amazon OpenSearch Service \(p. 203\)](#)
- [Using the AWS SDKs to interact with Amazon OpenSearch Service \(p. 205\)](#)

Elasticsearch client compatibility

The latest versions of the Elasticsearch clients might include license or version checks that artificially break compatibility. The following table includes recommendations around which versions of those clients to use for best compatibility with OpenSearch Service.

Important

These client versions are out of date and are not updated with the latest dependencies, including Log4j. We highly recommend using the OpenSearch versions of the clients when possible.

Client	Recommended version
Java low-level REST client	7.13.4
Java high-level REST client	7.13.4
Python Elasticsearch client	7.13.4
Ruby Elasticsearch client	7.13.3
Node.js Elasticsearch client	7.13.0

Signing HTTP requests to Amazon OpenSearch Service

This section includes examples of how to send signed HTTP requests to Amazon OpenSearch Service using Elasticsearch and OpenSearch clients and other common libraries. These code examples are for interacting with the OpenSearch 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 \(p. 146\)](#)), you must sign requests to the OpenSearch APIs with your IAM credentials.

For examples of how to interact with the configuration API, including operations like creating, updating, and deleting OpenSearch Service domains, see [the section called "Using the AWS SDKs" \(p. 205\)](#).

Important

The latest versions of the Elasticsearch clients might include license or version checks that artificially break compatibility. For the correct client version to use, see [the section called "Elasticsearch client compatibility" \(p. 191\)](#).

Topics

- [Java \(p. 192\)](#)
- [Python \(p. 193\)](#)
- [Ruby \(p. 196\)](#)
- [Node \(p. 198\)](#)
- [Go \(p. 202\)](#)

Java

The easiest way to send a signed request with Java is to use `AwsSdk2Transport`, introduced in `opensearch-java` version 2.1.0. The following [example](#) creates an index, writes a document, and deletes the index. You must provide values for `region` and `host`.

```
package com.amazonaws.samples;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import org.opensearch.client.opensearch.OpenSearchClient;
import org.opensearch.client.opensearch.core.IndexRequest;
import org.opensearch.client.opensearch.indices.CreateIndexRequest;
import org.opensearch.client.opensearch.indices.DeleteIndexRequest;
import org.opensearch.client.transport.aws.AwsSdk2Transport;
import org.opensearch.client.transport.aws.AwsSdk2TransportOptions;

import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;

public class IndexDocument {

    private static final String host = "search-....us-west-2.es.amazonaws.com";
    private static Region region = Region.US_WEST_2;

    public static void main(String[] args) throws IOException, InterruptedException {
        SdkHttpClient httpClient = ApacheHttpClient.builder().build();
        try {

            OpenSearchClient client = new OpenSearchClient(
                new AwsSdk2Transport(
                    httpClient,
                    host,
                    region,
                    AwsSdk2TransportOptions.builder().build()));

            // create the index
            String index = "sample-index";

            CreateIndexRequest createIndexRequest = new
            CreateIndexRequest.Builder().index(index).build();
            client.indices().create(createIndexRequest);

            // index data
            Map<String, Object> document = new HashMap<>();
        }
    }
}
```

```
        document.put("firstName", "Michael");
        document.put("lastName", "Douglas");
        IndexRequest documentIndexRequest = new IndexRequest.Builder()
            .index(index)
            .id("2")
            .document(document)
            .build();
        client.index(documentIndexRequest);

        // delete the index
        DeleteIndexRequest deleteRequest = new
DeleteIndexRequest.Builder().index(index).build();
        client.indices().delete(deleteRequest);

    } finally {
        httpClient.close();
    }
}
```

Other alternatives include using an AWS Request Signing Interceptor and/or the high-level REST client. See [this sample](#)

Tip

This sample uses the default credential chain. Run `aws configure` using the AWS CLI to set your credentials.

Python

This example uses the [opensearch-py](#) client for Python, which you can install using [pip](#). You must provide values for `region` and `host`.

```
from opensearchpy import OpenSearch, RequestsHttpConnection, AWSV4SignerAuth
import boto3

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

credentials = boto3.Session().get_credentials()
auth = AWSV4SignerAuth(credentials, region)
index_name = 'movies'

client = OpenSearch(
    hosts = [{'host': host, 'port': 443}],
    http_auth = auth,
    use_ssl = True,
    verify_certs = True,
    connection_class = RequestsHttpConnection
)

q = 'miller'
query = {
    'size': 5,
    'query': {
        'multi_match': {
            'query': q,
            'fields': ['title^2', 'director']
        }
    }
}

response = client.search(
    body = query,
```

```

        index = index_name
    )

print('\nSearch results:')
print(response)

```

Instead of the client, you might prefer [requests](#). The [requests-aws4auth](#) and [SDK for Python \(Boto3\)](#) packages simplify the authentication process, but are not strictly required. From the terminal, run the following commands:

```

pip install boto3
pip install opensearch-py
pip install requests
pip install requests-aws4auth

```

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

```

from opensearchpy import OpenSearch, 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)

search = OpenSearch(
    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"
}

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

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

```

If you don't want to use `opensearch-py`, you can just make standard HTTP requests. This example 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 OpenSearch 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)

```

Rather than static credentials, you can construct an AWS4Auth instance with automatically refreshing credentials, which is suitable for long-running applications using [AssumeRole](#). The refreshable credentials instance is used to generate valid static credentials for each request, eliminating the need to recreate the AWS4Auth instance when temporary credentials expire:

```

from requests_aws4auth import AWS4Auth
from botocore.session import Session

credentials = Session().get_credentials()

auth = AWS4Auth(region='us-west-1', service='es',
                refreshable_credentials=credentials)

```

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 opensearchpy import OpenSearch, 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)

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

search.bulk(bulk_file)

print(search.search(q='some test query'))
```

Ruby

This first example uses the Elasticsearch Ruby client and [Faraday middleware](#) to perform the request signing. Note that the latest versions of the client might include license or version checks that artificially break compatibility. For the correct client version to use, see [the section called “Elasticsearch client compatibility” \(p. 191\)](#). This example uses the recommended version 7.13.3.

From the terminal, run the following commands:

```
gem install elasticsearch -v 7.13.3
gem install faraday_middleware-aws-sigv4
```

This example code creates a new client, configures Faraday middleware to sign requests, and indexes a single document. You must provide values for `full_url_and_port` and `region`.

```
require 'elasticsearch'
require 'faraday_middleware/aws_sigv4'

full_url_and_port = '' # e.g. https://my-domain.region.es.amazonaws.com:443
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: full_url_and_port) 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-opensearchservice'

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 [opensearch-js](#) client for JavaScript to create an index and add a single document. To sign the request, it first locates credentials using the [credential-provider-node](#) module from version 3 of the SDK for JavaScript in Node.js. It then calls [aws4](#) to sign the request using [Signature Version 4](#). You must provide a value for host.

```
const { Client, Connection } = require("@opensearch-project/opensearch");
const { defaultProvider } = require("@aws-sdk/credential-provider-node");
const aws4 = require("aws4");

var host = '' // e.g. https://my-domain.region.es.amazonaws.com

const createAwsConnector = (credentials, region) => {
    class AmazonConnection extends Connection {
        buildRequestObject(params) {
            const request = super.buildRequestObject(params);
            request.service = 'es';
            request.region = region;
            request.headers = request.headers || {};
            request.headers['host'] = request.hostname;

            return aws4.sign(request, credentials);
        }
    }
    return {
        Connection: AmazonConnection
    };
};

const getClient = async () => {
    const credentials = await defaultProvider();
    return new Client({
        ...createAwsConnector(credentials, 'us-east-1'),
        node: host,
    });
}

async function search() {

    // Initialize the client.
    var client = await getClient();

    // Create an index.
    var index_name = "test-index";

    var response = await client.indices.create({
        index: index_name,
    });

    console.log("Creating index:");
    console.log(response.body);

    // Add a document to the index.
    var document = {
        "title": "Moneyball",
        "director": "Bennett Miller",
    }
}
```

```
        "year": "2011"
    };

    var response = await client.index({
        index: index_name,
        body: document
    });

    console.log(response.body);
}

search().catch(console.log);
```

This similar example uses [aws-opensearch-connector](#) rather than aws4. You must provide a value for host.

```
const { Client } = require("@opensearch-project/opensearch");
const { defaultProvider } = require("@aws-sdk/credential-provider-node");
const createAwsOpensearchConnector = require("aws-opensearch-connector");

var host = '' // e.g. https://my-domain.region.es.amazonaws.com

const getClient = async () => {
    const awsCredentials = await defaultProvider();
    const connector = createAwsOpensearchConnector({
        credentials: awsCredentials,
        region: process.env.AWS_REGION ?? 'us-east-1',
        getCredentials: function(cb) {
            return cb();
        }
    });
    return new Client({
        ...connector,
        node: host,
    });
}

async function search() {

    // Initialize the client.
    var client = await getClient();

    // Create an index.
    var index_name = "test-index";
    var response = await client.indices.create({
        index: index_name,
    });

    console.log("Creating index:");
    console.log(response.body);

    // Add a document to the index.
    var document = {
        "title": "Moneyball",
        "director": "Bennett Miller",
        "year": "2011"
    };

    var response = await client.index({
        index: index_name,
        body: document
    });

    console.log(response.body);
```

```
}
```

```
search().catch(console.log);
```

If you don't want to use `opensearch-js`, you can just make standard HTTP requests. This section includes examples for versions 2 and 3 of the SDK for JavaScript in Node.js. While version 2 is published as a single package, version 3 has a modular architecture with a separate package for each service.

Version 3

This example uses [version 3](#) of the SDK for JavaScript in Node.js. From the terminal, run the following commands:

```
npm i @aws-sdk/protocol-http
npm i @aws-sdk/credential-provider-node
npm i @aws-sdk/signature-v4
npm i @aws-sdk/node-http-handler
npm i @aws-crypto/sha256-browser
```

This example code indexes a single document. You must provide values for `region` and `domain`.

```
const { HttpRequest } = require("@aws-sdk/protocol-http");
const { defaultProvider } = require("@aws-sdk/credential-provider-node");
const { SignatureV4 } = require("@aws-sdk/signature-v4");
const { NodeHttpHandler } = require("@aws-sdk/node-http-handler");
const { Sha256 } = require("@aws-crypto/sha256-browser");

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).then(() => process.exit())

async function indexDocument(document) {

    // Create the HTTP request
    var request = new HttpRequest({
        body: JSON.stringify(document),
        headers: {
            'Content-Type': 'application/json',
            'host': domain
        },
        hostname: domain,
        method: 'PUT',
        path: index + '/' + type + '/' + id
    });

    // Sign the request
    var signer = new SignatureV4({
        credentials: defaultProvider(),
        region: region,
        service: 'es',
        sha256: Sha256
    });

    var signedRequest = await signer.sign(request);
```

```
// Send the request
var client = new NodeHttpHandler();
var { response } = await client.handle(signedRequest)
console.log(response.statusCode + ' ' + response.body.statusMessage);
var responseBody = '';
await new Promise(() => {
    response.body.on('data', (chunk) => {
        responseBody += chunk;
    });
    response.body.on('end', () => {
        console.log('Response body: ' + responseBody);
    });
}).catch((error) => {
    console.log('Error: ' + error);
});
});
```

Version 2

This example uses [version 2](#) of the SDK for JavaScript in Node.js. From the terminal, run the following command:

```
npm install aws-sdk
```

This example 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';
    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();
    return new Promise((resolve, reject) => {
        client.handleRequest(
            request,
            null,
```

```

        (response) => {
            const {statusCode, statusMessage, headers} = response;
            let body = '';
            response.on('data', (chunk) => {
                body += chunk;
            });
            response.on('end', () => {
                const data = {statusCode, statusMessage, headers};
                if (body) {
                    data.body = body;
                }
                resolve(data);
                console.log("Response body:" + body);
            });
        },
        (error) => {
            reject(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 OpenSearch 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 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.Println(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.Println(err)
}
fmt.Println(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"
```

Compressing HTTP requests in Amazon OpenSearch Service

You can compress HTTP requests and responses in Amazon OpenSearch Service domains using gzip compression. Gzip compression can help you reduce the size of your documents and lower bandwidth utilization and latency, thereby leading to improved transfer speeds.

Gzip compression is supported for all domains running OpenSearch or Elasticsearch 6.0 or later. Some OpenSearch clients have built-in support for gzip compression, and many programming languages have libraries that simplify the process.

Enabling gzip compression

Not to be confused with similar OpenSearch settings, `http_compression.enabled` is specific to OpenSearch Service and enables or disables gzip compression on a domain. Domains running OpenSearch or Elasticsearch 7.x have the gzip compression enabled by default, whereas domains running Elasticsearch 6.x have it disabled by default.

To enable gzip compression, send the following request:

```
PUT _cluster/settings
{
    "persistent" : {
        "http_compression.enabled": true
    }
}
```

Requests to `_cluster/settings` must be uncompressed, so you might need to use a separate client or standard HTTP request to update cluster settings.

Required headers

When including a gzip-compressed request body, keep the standard `Content-Type: application/json` header, and add the `Content-Encoding: gzip` header. To accept a gzip-compressed response, add the `Accept-Encoding: gzip` header, as well. If an OpenSearch client supports gzip compression, it likely includes these headers automatically.

Sample code (Python 3)

The following sample uses [opensearch-py](#) to perform the compression and send the request. This code signs the request using your IAM credentials.

```
from opensearchpy import OpenSearch, RequestsHttpConnection
from requests_aws4auth import AWS4Auth
import boto3

host = '' # e.g. 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)

# Create the client.
search = OpenSearch(
    hosts = [{'host': host, 'port': 443}],
    http_auth = awsauth,
    use_ssl = True,
    verify_certs = True,
    http_compress = True, # enables gzip compression for request bodies
    connection_class = RequestsHttpConnection
)

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

# Send the request.
print(search.index(index='movies', id='1', body=document, refresh=True))

# print(search.index(index='movies', doc_type='_doc', id='1', body=document, refresh=True))
```

Alternately, you can specify the proper headers, compress the request body yourself, and use a standard HTTP library like [Requests](#). This code signs the request using HTTP basic credentials, which your domain might support if you use [fine-grained access control \(p. 146\)](#).

```
import requests
import gzip
import json

base_url = '' # The domain with https:// and a trailing slash. For example, https://my-
test-domain.us-east-1.es.amazonaws.com/
auth = ('master-user', 'master-user-password') # For testing only. Don't store credentials
in code.
```

```
headers = {'Accept-Encoding': 'gzip', 'Content-Type': 'application/json',
           'Content-Encoding': 'gzip'}
```

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

# Compress the document.
compressed_document = gzip.compress(json.dumps(document).encode())

# Send the request.
path = 'movies/_doc?refresh=true'
url = base_url + path
response = requests.post(url, auth=auth, headers=headers, data=compressed_document)
print(response.status_code)
print(response.text)
```

Using the AWS SDKs to interact with Amazon OpenSearch Service

This section includes examples of how to use the AWS SDKs to interact with the Amazon OpenSearch Service configuration API. These code samples show how to create, update, and delete OpenSearch Service domains.

Important

For examples of how to interact with the OpenSearch APIs, such as `_index`, `_bulk`, `_search`, and `_snapshot`, see [the section called “Signing HTTP requests” \(p. 191\)](#).

Java

This section includes examples for versions 1 and 2 of the AWS SDK for Java.

Version 2

This example uses the [OpenSearchClientBuilder](#) constructor from version 2 of the AWS SDK for Java to create an OpenSearch 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.example.samples;

import java.util.concurrent.TimeUnit;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.CognitoOptions;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.CreateDomainResponse;
import software.amazon.awssdk.services.opensearch.model.DescribeDomainRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;
import software.amazon.awssdk.services.opensearch.model.DescribeDomainResponse;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainResponse;
```

```

import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;

/**
 * Sample class demonstrating how to use the Amazon Web Services SDK for Java to
 * create, update,
 * and delete Amazon OpenSearch Service domains.
 */

public class OpenSearchSample {

    public static void main(String[] args) {

        String domainName = "my-test-domain";

        // Build the client using the default credentials chain.
        // You can use the CLI and run `aws configure` to set access key, secret
        // key, and default region.

        OpenSearchClient client = OpenSearchClient.builder()
            // Unnecessary, but lets you use a region different than your default.
            .region(Region.US_EAST_1)
            // Unnecessary, but if desired, you can use a different provider chain.
            .credentialsProvider(DefaultCredentialsProvider.create())
            .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 OpenSearch Service domain with the specified options.
     * Some options require other Amazon Web Services resources, such as an Amazon
     Cognito user pool
     * and identity pool, whereas others require just an instance type or instance
     * count.
     *
     * @param client
     *          The client to use for the requests to Amazon OpenSearch Service
     * @param domainName
     *          The name of the domain you want to create
     */
}

public static void createDomain(OpenSearchClient client, String domainName) {

    // Create the request and set the desired configuration options

    try {

        ClusterConfig clusterConfig = ClusterConfig.builder()
            .dedicatedMasterEnabled(true)
            .dedicatedMasterCount(3)
            // Small, inexpensive instance types for testing. Not recommended
for production.
            .dedicatedMasterType("t2.small.search")
            .instanceType("t2.small.search")
            .instanceCount(5)
            .build();

        // Many instance types require EBS storage.
        EBSOptions ebsOptions = EBSOptions.builder()
            .ebsEnabled(true)
    }
}

```

```

        .volumeSize(10)
        .volumeType("gp2")
        .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
            .enabled(true)
            .build();

        CreateDomainRequest createRequest = CreateDomainRequest.builder()
            .domainName(domainName)
            .engineVersion("OpenSearch_1.0")
            .clusterConfig(clusterConfig)
            .ebsOptions(ebsOptions)
            .nodeToNodeEncryptionOptions(encryptionOptions)
            // You can uncomment this line and add your account ID, a user
name, and the
            // domain name to add an access policy.
            // .accessPolicies("{\"Version\":\"2012-10-17\",\"Statement\":
[{\\"Effect\\":\\"Allow\",\\\"Principal\\\":{\\\"AWS\\\":["arn:aws:iam::123456789012:user/
username\"]},\\\"Action\\\":["es:*"],\\\"Resource\\\":\\\"arn:aws:es:region:123456789012:domain/
domain-name/*\\"]}]")
            .build();

        // Make the request.
        System.out.println("Sending domain creation request...");
        CreateDomainResponse createResponse = client.createDomain(createRequest);
        System.out.println("Domain status:
"+createResponse.domainStatus().toString());
        System.out.println("Domain ID: "+createResponse.domainStatus().domainId());

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

/**
 * Updates the configuration of an Amazon OpenSearch Service domain with the
 * specified options. Some options require other Amazon Web Services resources,
such as an
 * Amazon Cognito user pool and identity pool, whereas others require just an
 * instance type or instance count.
 *
 * @param client
 *         The client to use for the requests to Amazon OpenSearch Service
 * @param domainName
 *         The name of the domain to update
 */
public static void updateDomain(OpenSearchClient client, String domainName) {

    // Updates the domain to use three data instances instead of five.
    // You can uncomment the Cognito line and fill in the strings to enable Cognito
    // authentication for OpenSearch Dashboards.

    try {

        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(5)
            .build();

        CognitoOptions cognitoOptions = CognitoOptions.builder()
            .enabled(true)
            .userPoolId("user-pool-id")
    }
}

```

```

        .identityPoolId("identity-pool-id")
        .roleArn("role-arn")
        .build();

    UpdateDomainConfigRequest updateRequest =
UpdateDomainConfigRequest.builder()
        .domainName(domainName)
        .clusterConfig(clusterConfig)
        //.cognitoOptions(cognitoOptions)
        .build();

    System.out.println("Sending domain update request...");
    UpdateDomainConfigResponse updateResponse =
client.updateDomainConfig(updateRequest);
    System.out.println("Domain config:
"+updateResponse.domainConfig().toString());

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

/**
 * Deletes an Amazon OpenSearch Service domain. Deleting a domain can take
 * several minutes.
 *
 * @param client
 *         The client to use for the requests to Amazon OpenSearch Service
 * @param domainName
 *         The name of the domain that you want to delete
 */
public static void deleteDomain(OpenSearchClient client, String domainName) {

    try {

        DeleteDomainRequest deleteRequest = DeleteDomainRequest.builder()
            .domainName(domainName)
            .build();

        System.out.println("Sending domain deletion request...");
        DeleteDomainResponse deleteResponse = client.deleteDomain(deleteRequest);
        System.out.println("Domain status: "+deleteResponse.toString());

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

/**
 * 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 client to use for the requests to Amazon OpenSearch Service
 * @param domainName
 *         The name of the domain that you want to check

```

```

        */

    public static void waitForDomainProcessing(OpenSearchClient client, String
domainName) {
    // Create a new request to check the domain status.
    DescribeDomainRequest describeRequest = DescribeDomainRequest.builder()
        .domainName(domainName)
        .build();

    // Every 15 seconds, check whether the domain is processing.
    DescribeDomainResponse describeResponse =
client.describeDomain(describeRequest);
    while (describeResponse.domainStatus().processing()) {
        try {
            System.out.println("Domain still processing...");
            TimeUnit.SECONDS.sleep(15);
            describeResponse = client.describeDomain(describeRequest);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    // Once we exit that loop, the domain is available
    System.out.println("Amazon OpenSearch Service has finished processing changes
for your domain.");
    System.out.println("Domain description: "+describeResponse.toString());
}
}

```

Version 1

This example uses the [AWSElasticsearchClientBuilder](#) constructor from version 1 of the AWS SDK for Java to create a legacy Elasticsearch 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.ResourceNotFoundException;
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 Amazon Web Services SDK for Java to
create,
* and delete Amazon OpenSearch Service domains.
*/

```

```

public class OpenSearchSample {

    public static void main(String[] args) {

        final String domainName = "my-test-domain";

        // Build the client using the default credentials chain.
        // You can use the 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 OpenSearch Service domain with the specified options.
     * Some options require other Amazon Web Services resources, such as an Amazon
     Cognito user pool
     * and identity pool, whereas others require just an instance type or instance
     * count.
     *
     * @param client
     *          The client to use for the requests to Amazon OpenSearch 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("7.10")
            .withElasticsearchClusterConfig(new ElasticsearchClusterConfig()
                .withDedicatedMasterEnabled(true)
                .withDedicatedMasterCount(3)
                // Small, inexpensive instance types for testing. Not
recommended for production
                .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/
username\"]},\"Action\":[\"es:*\"],\"Resource\":\"arn:aws:es:region:123456789012:domain/
domain-name/*\"]}]")
    }
}

```

```

// Make the request.
System.out.println("Sending domain creation request...");
CreateElasticsearchDomainResult createResponse =
client.createElasticsearchDomain(createRequest);
System.out.println("Domain creation response from Amazon OpenSearch Service:");
System.out.println(createResponse.getDomainStatus().toString());
}

/**
 * Updates the configuration of an Amazon OpenSearch Service domain with the
 * specified options. Some options require other Amazon Web Services resources,
such as an
 * Amazon Cognito user pool and identity pool, whereas others require just an
 * instance type or instance count.
 *
 * @param client
 *          The client to use for the requests to Amazon OpenSearch 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 OpenSearch Dashboards.
    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 OpenSearch
Service:");
    System.out.println(updateResponse.toString());
} catch (ResourceNotFoundException e) {
    System.out.println("Domain not found. Please check the domain name.");
}
}

/**
 * Deletes an Amazon OpenSearch Service domain. Deleting a domain can take
 * several minutes.
 *
 * @param client
 *          The client to use for the requests to Amazon OpenSearch 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 OpenSearch
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 client to use for the requests to Amazon OpenSearch 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 OpenSearch Service has finished processing changes
for your domain.");
    System.out.println("Domain description response from Amazon OpenSearch
Service:");
    System.out.println(describeResponse.toString());
}
}

```

Python

This example uses the [OpenSearchService](#) low-level Python client from the AWS SDK for Python (Boto) to create a domain, update its configuration, and delete it.

```

import boto3
import botocore

```

```

from botocore.config import Config
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default region.

my_config = Config(
    # Optionally lets you specify a region other than your default.
    region_name='us-west-2'
)

client = boto3.client('opensearch', config=my_config)

domainName = 'my-test-domain' # The name of the domain

def createDomain(client, domainName):
    """Creates an Amazon OpenSearch Service domain with the specified options."""
    response = client.create_domain(
        DomainName=domainName,
        EngineVersion='OpenSearch_1.0',
        ClusterConfig={
            'InstanceType': 't2.small.search',
            'InstanceCount': 5,
            'DedicatedMasterEnabled': True,
            'DedicatedMasterType': 't2.small.search',
            'DedicatedMasterCount': 3
        },
        # Many instance types require EBS storage.
        EBSOptions={
            'EBSEnabled': True,
            'VolumeType': 'gp2',
            'VolumeSize': 10
        },
        AccessPolicies="{\\"Version\\":\\"2012-10-17\\",\\"Statement\\":[{\\\"Effect\\\":\\"Allow\\",
        \\"Principal\\":{\\\"AWS\\":["arn:aws:iam::123456789012:user/user-name"]},\\"Action\\":["es:*"],
        \\"Resource\\":\\"arn:aws:es:us-west-2:123456789012:domain/my-test-domain/*\\"]}],"
        NodeToNodeEncryptionOptions={
            'Enabled': True
        }
    )
    print("Creating domain...")
    print(response)

def updateDomain(client, domainName):
    """Updates the domain to use three data nodes instead of five."""
    try:
        response = client.update_domain_config(
            DomainName=domainName,
            ClusterConfig={
                'InstanceCount': 3
            }
        )
        print('Sending domain update request...')
        print(response)

    except botocore.exceptions.ClientError as error:
        if error.response['Error']['Code'] == 'ResourceNotFoundException':
            print('Domain not found. Please check the domain name.')
        else:
            raise error

def deleteDomain(client, domainName):

```

```
"""Deletes an OpenSearch Service domain. Deleting a domain can take several minutes."""
try:
    response = client.delete_domain(
        DomainName=domainName
    )
    print('Sending domain deletion request...')
    print(response)

except botocore.exceptions.ClientError as error:
    if error.response['Error']['Code'] == 'ResourceNotFoundException':
        print('Domain not found. Please check the domain name.')
    else:
        raise error

def waitForDomainProcessing(client, domainName):
    """Waits for the domain to finish processing changes."""
    try:
        response = client.describe_domain(
            DomainName=domainName
        )
        # Every 15 seconds, check whether the domain is processing.
        while response["DomainStatus"]["Processing"] == True:
            print('Domain still processing...')
            time.sleep(15)
            response = client.describe_domain(
                DomainName=domainName
            )

        # Once we exit the loop, the domain is available.
        print('Amazon OpenSearch Service has finished processing changes for your domain.')
        print('Domain description:')
        print(response)

    except botocore.exceptions.ClientError as error:
        if error.response['Error']['Code'] == 'ResourceNotFoundException':
            print('Domain not found. Please check the domain name.')
        else:
            raise error

def main():
    """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)
```

Node

This example uses the version 3 of the SDK for JavaScript in Node.js [OpenSearch client](#) to create a domain, update its configuration, and delete it.

```
var {
  OpenSearchClient,
  CreateDomainCommand,
  DescribeDomainCommand,
  UpdateDomainConfigCommand,
  DeleteDomainCommand
} = require("@aws-sdk/client-opensearch");
var sleep = require('sleep');

var client = new OpenSearchClient();
```

```

var domainName = 'my-test-domain'

// 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)

async function createDomain(client, domainName) {
    // Creates an Amazon OpenSearch Service domain with the specified options.
    var command = new CreateDomainCommand({
        DomainName: domainName,
        EngineVersion: 'OpenSearch_1.0',
        ClusterConfig: {
            'InstanceType': 't2.small.search',
            'InstanceCount': 5,
            'DedicatedMasterEnabled': 'True',
            'DedicatedMasterType': 't2.small.search',
            'DedicatedMasterCount': 3
        },
        EBSOptions:{
            'EBSEnabled': 'True',
            'VolumeType': 'gp2',
            'VolumeSize': 10
        },
        AccessPolicies: "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":["AWS":["arn:aws:iam::123456789012:user/user-name"]],\"Action\":[\"es:*\"],
\"Resource\":\"arn:aws:es:us-east-1:123456789012:domain/my-test-domain/*\"]]}",
        NodeToNodeEncryptionOptions:{
            'Enabled': 'True'
        }
    });
    const response = await client.send(command);
    console.log("Creating domain...");
    console.log(response);
}

async function updateDomain(client, domainName) {
    // Updates the domain to use three data nodes instead of five.
    var command = new UpdateDomainConfigCommand({
        DomainName: domainName,
        ClusterConfig: {
            'InstanceCount': 3
        }
    });
    const response = await client.send(command);
    console.log('Sending domain update request...');
    console.log(response);
}

async function deleteDomain(client, domainName) {
    // Deletes an OpenSearch Service domain. Deleting a domain can take several minutes.
    var command = new DeleteDomainCommand({
        DomainName: domainName
    });
    const response = await client.send(command);
    console.log('Sending domain deletion request...');
    console.log(response);
}

async function waitForDomainProcessing(client, domainName) {
    // Waits for the domain to finish processing changes.
    try {
        var command = new DescribeDomainCommand({

```

```
        DomainName: domainName
    });
    var response = await client.send(command);

    while (response.DomainStatus.Processing == true) {
        console.log('Domain still processing...')
        await sleep(15000) // Wait for 15 seconds, then check the status again
        function sleep(ms) {
            return new Promise((resolve) => {
                setTimeout(resolve, ms);
            });
        }
        var response = await client.send(command);
    }
    // Once we exit the loop, the domain is available.
    console.log('Amazon OpenSearch Service has finished processing changes for your
domain.');
    console.log('Domain description:');
    console.log(response);

} catch (error) {
    if (error.name === 'ResourceNotFoundException') {
        console.log('Domain not found. Please check the domain name.')
    }
}
}
```

Indexing data in Amazon OpenSearch Service

Because Amazon OpenSearch Service 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, OpenSearch Service has clients for many programming languages. Advanced users can skip directly to the section called “[Signing HTTP requests](#)” (p. 191) or the section called “[Loading streaming data into OpenSearch Service](#)” (p. 219).

For an introduction to indexing, see the [OpenSearch documentation](#).

Naming restrictions for indexes

OpenSearch Service indexes 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. OpenSearch Service 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://opensearch-domain/dr-jane-doe/flu-patients-2018/202-555-0100/ HTTP/1.1"
```

Even if you don't have [permissions](#) (p. 130) 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.

If OpenSearch Service detects a real or perceived IP address in an index name (for example, my-index-12.34.56.78.91), it masks the IP address. A call to _cat/indices yields the following response:

```
green open my-index-x.x.x.x soY19tBERoKo71WcEScidw 5 1 0 0 2kb 1kb
```

To prevent unnecessary confusion, avoid including IP addresses in index names.

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 opensearch-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 OpenSearch Service 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 OpenSearch Service REST APIs, but is especially useful with APIs that you call frequently, such as the `_index` and `_bulk` APIs:

```
PUT opensearch-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 opensearch-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 OpenSearch Service

You can load [streaming data](#) into your Amazon OpenSearch Service domain from many different sources. Some sources, like Amazon Kinesis Data Firehose and Amazon CloudWatch Logs, have built-in support for OpenSearch Service. 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 [Getting started with Lambda](#) in the *AWS Lambda Developer Guide* and [AWS service endpoints](#) in the *AWS General Reference*.

Topics

- [Loading streaming data from Amazon S3 \(p. 219\)](#)
- [Loading streaming data from Amazon Kinesis Data Streams \(p. 223\)](#)
- [Loading streaming data from Amazon DynamoDB \(p. 226\)](#)
- [Loading streaming data from Amazon Kinesis Data Firehose \(p. 229\)](#)
- [Loading streaming data from Amazon CloudWatch \(p. 229\)](#)
- [Loading streaming data from AWS IoT \(p. 229\)](#)

Loading streaming data from Amazon S3

You can use Lambda to send data to your OpenSearch Service 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.

Prerequisites

Before proceeding, you must have the following resources.

Prerequisite	Description
Amazon S3 bucket	For more information, see Create your first S3 bucket in the <i>Amazon Simple Storage Service User Guide</i> . The bucket must reside in the same Region as your OpenSearch Service domain.

Prerequisite	Description
OpenSearch Service domain	The destination for data after your Lambda function processes it. For more information, see the section called “Creating OpenSearch Service domains” (p. 16) .

Create 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 [Lambda deployment packages](#) in the [AWS Lambda Developer Guide](#).

1. Create a directory. In this sample, we use the name `s3-to-opensearch`.
2. Create a file within 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 OpenSearch Service domain, e.g. https://search-mydomain.us-
west-1.es.amazonaws.com
index = 'lambda-s3-index'
type = '_doc'
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\w\d\d\d:\d\d\d:\d\d\d\s-\d\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:
            line = line.decode("utf-8")
            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 pip](#) if you haven't already, then install the dependencies to a new package directory:

```
cd s3-to-opensearch
pip install --target ./package requests
pip install --target ./package requests_aws4auth
```

All Lambda execution environments have [Boto3](#) installed, so you don't need to include it in your deployment package.

4. Package the application code and dependencies:

```
cd package
zip -r ..../lambda.zip .

cd ..
zip -g lambda.zip sample.py
```

Create 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 3.8), and IAM role. The IAM role defines the permissions for your function. For detailed instructions, see [Create a Lambda function with the console](#) in the *AWS Lambda Developer Guide*.

This example assumes you're using the console. Choose Python 3.9 and a role that has S3 read permissions and OpenSearch Service write permissions, as shown in the following screenshot:

Author from scratch

Start with a simple Hello World example.

Use a blueprint

Build a Lambda application from sample code and configuration presets for common use cases.

Container image

Select a container image to deploy for your function.

Basic information

Function name Enter a name that describes the purpose of your function. Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#) Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Permissions [Info](#) By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from policy templates

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Role name Enter a name for your new role. Use only letters, numbers, hyphens, or underscores with no spaces.

Policy templates - *optional* [Info](#) Choose one or more policy templates.

After you create the function, you must add a trigger. For this example, we want the code to run whenever a log file arrives in an S3 bucket:

1. Choose **Add trigger** and select **S3**.
2. Choose your bucket.
3. For **Event type**, choose **PUT**.
4. For **Prefix**, type `logs/`.
5. For **Suffix**, type `.log`.
6. Acknowledge the recursive invocation warning and choose **Add**.

Finally, you can upload your deployment package:

1. Choose **Upload from** and **.zip file**, then follow the prompts to upload your deployment package.
2. After the upload finishes, edit the **Runtime settings** and change the **Handler** to `sample.handler`. This setting tells Lambda the file (`sample.py`) and method (`handler`) that it should run after a trigger.

At this point, you have a complete set of resources: a bucket for log files, a function that runs whenever a log file is added to the bucket, code that performs the parsing and indexing, and an OpenSearch Service 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 [Upload an object to your bucket](#) in the *Amazon Simple Storage Service User Guide*.

Then use the OpenSearch Service console or OpenSearch Dashboards to verify that the lambda-s3-index index contains two documents. You can also make a standard search request:

```
GET https://domain-name/lambda-s3-index/_search?pretty  
{  
  "hits" : {  
    "total" : 2,  
    "max_score" : 1.0,  
    "hits" : [  
      {  
        "_index" : "lambda-s3-index",  
        "_type" : "_doc",  
        "_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" : "_doc",  
        "_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 from Amazon Kinesis Data Streams

You can load streaming data from Kinesis Data Streams to OpenSearch Service. 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.

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 .
OpenSearch Service Domain	The destination for data after your Lambda function processes it. For more information, see the section called "Creating OpenSearch Service domains" (p. 16)
IAM Role	<p>This role must have basic OpenSearch Service, Kinesis, and Lambda permissions, such as the following:</p> <pre>{ "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": "*" }] }</pre> <p>The role must have the following trust relationship:</p> <pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Principal": { "Service": "lambda.amazonaws.com" }, "Action": "sts:AssumeRole" }] }</pre> <p>To learn more, see Creating IAM roles in the <i>IAM User Guide</i>.</p>

Create the Lambda function

Follow the instructions in [the section called "Create the Lambda deployment package" \(p. 220\)](#), but create a directory named `kinesis-to-opensearch` 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 OpenSearch Service domain, e.g. https://search-mydomain.us-
west-1.es.amazonaws.com
index = 'lambda-kine-index'
type = '_doc'
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`.

Install `pip` if you haven't already, then use the following commands to install your dependencies:

```
cd kinesis-to-opensearch

pip install --target ./package requests
pip install --target ./package requests_aws4auth
```

Then follow the instructions in [the section called “Create the Lambda function” \(p. 221\)](#), but specify the IAM role from [the section called “Prerequisites” \(p. 224\)](#) and the following settings for the trigger:

- **Kinesis stream:** your Kinesis stream
- **Batch size:** 100
- **Starting position:** Trim horizon

To learn more, see [What is 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 runs after the stream receives new data and indexes that data, and an OpenSearch Service domain for searching and visualization.

Test 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 test --data "My test data." --partition-key partitionKey1 --region us-west-1
```

Then use the OpenSearch Service console or OpenSearch Dashboards to verify that lambda-kine-index contains a document. You can also use the following request:

```
GET https://domain-name/lambda-kine-index/_search
{
  "hits" : [
    {
      "_index": "lambda-kine-index",
      "_type": "_doc",
      "_id":
      "shardId-000000000000:49583511615762699495012960821421456686529436680496087042",
      "_score": 1,
      "_source": {
        "timestamp": 1523648740.051,
        "message": "My test data.",
        "id":
      "shardId-000000000000:49583511615762699495012960821421456686529436680496087042"
      }
    }
  ]
}
```

Loading streaming data from Amazon DynamoDB

You can use AWS Lambda to send data to your OpenSearch Service 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 on DynamoDB Tables in the <i>Amazon DynamoDB Developer Guide</i> . The table must reside in the same Region as your OpenSearch Service domain and have a stream set to New image . To learn more, see Enabling a Stream .
OpenSearch Service Domain	The destination for data after your Lambda function processes it. For more information, see the section called "Creating OpenSearch Service domains" (p. 16) .
IAM Role	This role must have basic OpenSearch Service, DynamoDB, and Lambda execution permissions, such as the following:

Prerequisite	Description
	<pre>{ "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": "*" }] }</pre>
	<p>The role must have the following trust relationship:</p> <pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Principal": { "Service": "lambda.amazonaws.com" }, "Action": "sts:AssumeRole" }] }</pre>
	<p>To learn more, see Creating IAM roles in the <i>IAM User Guide</i>.</p>

Create the Lambda function

Follow the instructions in [the section called “Create the Lambda deployment package” \(p. 220\)](#), but create a directory named ddb-to-opensearch 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 OpenSearch Service domain, e.g. https://search-mydomain.us-
west-1.es.amazonaws.com
index = 'lambda-index'
type = '_doc'
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 OpenSearch 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.

[Install pip](#) if you haven't already, then use the following commands to install your dependencies:

```
cd ddb-to-opensearch
pip install --target ./package requests
pip install --target ./package requests_aws4auth
```

Then follow the instructions in [the section called “Create the Lambda function” \(p. 221\)](#), but specify the IAM role from [the section called “Prerequisites” \(p. 226\)](#) and the following settings for the trigger:

- **Table:** your DynamoDB table
- **Batch size:** 100
- **Starting position:** Trim horizon

To learn more, see [Process New Items with DynamoDB Streams and Lambda](#) 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 runs after your source data changes and indexes those changes, and an OpenSearch Service domain for searching and visualization.

Test 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 test --item '{"director": {"S": "Kevin Costner"}, "id": {"S": "00001"}, "title": {"S": "The Postman"}}' --region us-west-1
```

Then use the OpenSearch Service console or OpenSearch Dashboards to verify that lambda-index contains a document. You can also use the following request:

```
GET https://domain-name/lambda-index/_doc/00001
{
    "_index": "lambda-index",
    "_type": "_doc",
    "_id": "00001",
    "_version": 1,
    "found": true,
    "_source": {
```

```
    "director": {  
        "S": "Kevin Costner"  
    },  
    "id": {  
        "S": "00001"  
    },  
    "title": {  
        "S": "The Postman"  
    }  
}  
}
```

Loading streaming data from Amazon Kinesis Data Firehose

Kinesis Data Firehose supports OpenSearch Service as a delivery destination. For instructions about how to load streaming data into OpenSearch Service, see [Creating a Kinesis Data Firehose Delivery Stream](#) and [Choose OpenSearch Service for Your Destination](#) in the *Amazon Kinesis Data Firehose Developer Guide*.

Before you load data into OpenSearch Service, you might need to perform transforms on the data. To learn more about using Lambda functions to perform this task, see [Amazon Kinesis Data Firehose 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 OpenSearch Service, 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 from Amazon CloudWatch

You can load streaming data from CloudWatch Logs to your OpenSearch Service 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 OpenSearch Service](#) in the *Amazon CloudWatch Developer Guide*.

Loading streaming data from AWS IoT

You can send data from AWS IoT using [rules](#). To learn more, see the [OpenSearch](#) action in the *AWS IoT Developer Guide*.

Loading data into Amazon OpenSearch Service 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 OpenSearch Service domain. The service supports all standard Logstash input plugins, including the Amazon S3 input plugin. OpenSearch Service supports the [logstash-output-opensearch](#) output plugin, which supports both basic authentication and IAM credentials. The plugin works with version 8.1 and lower of Logstash OSS.

Configuration

Logstash configuration varies based on the type of authentication your domain uses.

No matter which authentication method you use, you must set `ecs_compatibility` to `disabled` in the output section of the configuration file. Logstash 8.0 introduced a breaking change where all plugins are run in [ECS compatibility mode by default](#). You must override the default value to maintain legacy behavior.

Fine-grained access control configuration

If your OpenSearch Service domain uses [fine-grained access control \(p. 146\)](#) with HTTP basic authentication, configuration is similar to any other OpenSearch cluster. This example configuration file takes its input from the open source version of Filebeat (Filebeat OSS):

```
input {
  beats {
    port => 5044
  }
}

output {
  opensearch {
    hosts      => "https://domain-endpoint:443"
    user       => "my-username"
    password   => "my-password"
    index      => "logstash-logs-%{+YYYY.MM.dd}"
    ecs_compatibility => disabled
    ssl_certificate_verification => false
  }
}
```

Configuration varies by Beats application and use case, but your Filebeat OSS configuration might look like this:

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /path/to/logs/dir/*.log
filebeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false
setup.ilm.enabled: false
setup.ilm.check_exists: false
setup.template.settings:
  index.number_of_shards: 1
output.logstash:
  hosts: ["logstash-host:5044"]
```

IAM configuration

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 OpenSearch Service using IAM credentials.

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 {
    opensearch {
        hosts => ["domain-endpoint:443"]
        auth_type => {
            type => 'aws_iam'
            aws_access_key_id => 'your-access-key'
            aws_secret_access_key => 'your-secret-key'
            region => 'us-east-1'
        }
        index => "logstash-logs-%{+YYYY.MM.dd}"
        ecs_compatibility => disabled
    }
}
```

If you don't want to provide your IAM credentials within the configuration file, you can export them (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"
```

If your OpenSearch Service domain is in a VPC, the Logstash OSS machine must be able to connect to the 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" \(p. 39\)](#).

Searching data in Amazon OpenSearch Service

There are several common methods for searching documents in Amazon OpenSearch Service, including URI searches and request body searches. OpenSearch Service offers additional functionality that improves the search experience, such as custom packages, SQL support, and asynchronous search. For a comprehensive OpenSearch search API reference, see the [OpenSearch documentation](#).

Note

The following sample requests work with OpenSearch APIs. Some requests might not work with older Elasticsearch versions.

Topics

- [URI searches \(p. 232\)](#)
- [Request body searches \(p. 233\)](#)
- [Dashboards Query Language \(p. 237\)](#)
- [Custom packages for Amazon OpenSearch Service \(p. 238\)](#)
- [Querying your Amazon OpenSearch Service data with SQL \(p. 245\)](#)
- [k-Nearest Neighbor \(k-NN\) search in Amazon OpenSearch Service \(p. 248\)](#)
- [Cross-cluster search for Amazon OpenSearch Service \(p. 251\)](#)
- [Learning to Rank for Amazon OpenSearch Service \(p. 257\)](#)
- [Asynchronous search for Amazon OpenSearch Service \(p. 276\)](#)

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": {  
      "value": 85,  
      "relation": "eq"  
    },  
    "max_score": 6.6137657,  
    "hits": [  
      {  
        "_index": "my-index",  
        "_score": 6.6137657,  
        "_type": "my-type",  
        "_id": "1",  
        "name": "House A",  
        "location": "123 Main St",  
        "city": "Anytown",  
        "state": "CA",  
        "zip": "90210",  
        "price": 1200000  
      },  
      {  
        "_index": "my-index",  
        "_score": 6.6137657,  
        "_type": "my-type",  
        "_id": "2",  
        "name": "House B",  
        "location": "456 Elm St",  
        "city": "Anytown",  
        "state": "CA",  
        "zip": "90210",  
        "price": 1500000  
      },  
      {  
        "_index": "my-index",  
        "_score": 6.6137657,  
        "_type": "my-type",  
        "_id": "3",  
        "name": "House C",  
        "location": "789 Oak St",  
        "city": "Anytown",  
        "state": "CA",  
        "zip": "90210",  
        "price": 1800000  
      },  
      {  
        "_index": "my-index",  
        "_score": 6.6137657,  
        "_type": "my-type",  
        "_id": "4",  
        "name": "House D",  
        "location": "123 Pine St",  
        "city": "Anytown",  
        "state": "CA",  
        "zip": "90210",  
        "price": 2000000  
      },  
      {  
        "_index": "my-index",  
        "_score": 6.6137657,  
        "_type": "my-type",  
        "_id": "5",  
        "name": "House E",  
        "location": "456 Cedar St",  
        "city": "Anytown",  
        "state": "CA",  
        "zip": "90210",  
        "price": 2200000  
      },  
      {  
        "_index": "my-index",  
        "_score": 6.6137657,  
        "_type": "my-type",  
        "_id": "6",  
        "name": "House F",  
        "location": "789 Birch St",  
        "city": "Anytown",  
        "state": "CA",  
        "zip": "90210",  
        "price": 2500000  
      },  
      {  
        "_index": "my-index",  
        "_score": 6.6137657,  
        "_type": "my-type",  
        "_id": "7",  
        "name": "House G",  
        "location": "123 Maple St",  
        "city": "Anytown",  
        "state": "CA",  
        "zip": "90210",  
        "price": 2800000  
      },  
      {  
        "_index": "my-index",  
        "_score": 6.6137657,  
        "_type": "my-type",  
        "_id": "8",  
        "name": "House H",  
        "location": "456 Birch St",  
        "city": "Anytown",  
        "state": "CA",  
        "zip": "90210",  
        "price": 3000000  
      },  
      {  
        "_index": "my-index",  
        "_score": 6.6137657,  
        "_type": "my-type",  
        "_id": "9",  
        "name": "House I",  
        "location": "789 Birch St",  
        "city": "Anytown",  
        "state": "CA",  
        "zip": "90210",  
        "price": 3200000  
      },  
      {  
        "_index": "my-index",  
        "_score": 6.6137657,  
        "_type": "my-type",  
        "_id": "10",  
        "name": "House J",  
        "location": "123 Birch St",  
        "city": "Anytown",  
        "state": "CA",  
        "zip": "90210",  
        "price": 3500000  
      }  
    ]  
  }  
}
```

```
{  
    "_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/  
MV5BMTY20TQxNTc10F5BMl5BanBnXkFtZTYwNjA3NjI5._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 OpenSearch 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 OpenSearch domain-specific language (DSL) for queries. The query DSL lets you specify the full range of OpenSearch search options.

Note

You can't include Unicode special characters in a text field value, or the value will be parsed as multiple values separated by the special character. This incorrect parsing can lead to unintentional filtering of documents and potentially compromise control over their access. For more information, see [A note on Unicode special characters in text fields](#) in the OpenSearch documentation.

The following match query is similar to the final [URI search \(p. 232\)](#) 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 OpenSearch 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/MV5BMTIzODEzODE2OF5BM15BanBnXkFtZTcwNjQ30DcyMQ@@._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, OpenSearch wraps the matching string in `` 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
    }
}
```

```
}
```

Dashboards Query Language

You can use the [Dashboards Query Language \(DQL\)](#) to search for data and visualizations in OpenSearch Dashboards. DQL uses four primary query types: *terms*, *Boolean*, *date and range*, and *nested field*.

Terms query

A terms query requires you to specify the term that you're searching for.

To perform a terms query, enter the following:

```
host:www.example.com
```

Boolean query

You can use the Boolean operators AND, or, and not to combine multiple queries.

To perform a Boolean query, paste the following:

```
host.keyword:www.example.com and response.keyword:200
```

Date and range query

You can use a date and range query to find a date before or after your query.

- > indicates a search for a date after your specified date.
- < indicates a search for a date before your specified date.

```
@timestamp > "2020-12-14T09:35:33"
```

Nested field query

If you have a document with nested fields, you have to specify which parts of the document that you want to retrieve. The following is a sample document that contains nested fields:

```
{"NBA players": [
    {"player-name": "Lebron James",
     "player-position": "Power forward",
     "points-per-game": "30.3"
    },
    {"player-name": "Kevin Durant",
     "player-position": "Power forward",
     "points-per-game": "27.1"
    },
    {"player-name": "Anthony Davis",
     "player-position": "Power forward",
     "points-per-game": "23.2"
    },
    {"player-name": "Giannis Antetokounmpo",
     "player-position": "Power forward",
     "points-per-game": "29.9"
    }
]
```

To retrieve a specific field using DQL, paste the following:

```
NBA players: {player-name: Lebron James}
```

To retrieve multiple objects from the nested document, paste the following:

```
NBA players: {player-name: Lebron James} and NBA players: {player-name: Giannis Antetokounmpo}
```

To search within a range, paste the following:

```
NBA players: {player-name: Lebron James} and NBA players: {player-name: Giannis Antetokounmpo and < 30}
```

If your document has an object nested within another object, you can still retrieve data by specifying all of the levels. To do this, paste the following:

```
Top-Power-forwards.NBA players: {player-name:Lebron James}
```

Custom packages for Amazon OpenSearch Service

Amazon OpenSearch Service lets you upload custom dictionary files, such as 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 OpenSearch 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

- [Package permissions requirements \(p. 238\)](#)
- [Uploading packages to Amazon S3 \(p. 239\)](#)
- [Importing and associating packages \(p. 239\)](#)
- [Using custom packages with OpenSearch \(p. 239\)](#)
- [Updating custom packages \(console\) \(p. 241\)](#)
- [Updating custom packages \(AWS SDK\) \(p. 242\)](#)
- [Manual index updates \(p. 243\)](#)
- [Dissociating and removing packages \(p. 245\)](#)

Package permissions requirements

Users without administrator access require certain AWS Identity and Access Management (IAM) actions in order to manage packages:

- `es:CreatePackage` - create a package in an OpenSearch Service Region
- `es:DeletePackage` - delete a package from an OpenSearch Service Region
- `es:AssociatePackage` - associate a package to a domain
- `es:DissociatePackage` - dissociate a package from a domain

You also need permissions on the Amazon S3 bucket path or object where the custom package resides.

Grant all permission within IAM, not in the domain access policy. For more information, see [the section called “Identity and Access Management” \(p. 130\)](#).

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 objects](#) in the *Amazon Simple Storage Service User Guide*.

If your package contains sensitive information, specify [server-side encryption with S3-managed keys](#) when you upload it. OpenSearch Service can't access files on S3 that you protect using an AWS KMS 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
```

Certain dictionaries, such as Hunspell dictionaries, use multiple files and require their own directories on the file system. At this time, OpenSearch Service only supports single-file dictionaries.

Importing and associating packages

The console is the simplest way to import a package into OpenSearch Service and associate the package with a domain. When you import a package from Amazon S3, OpenSearch Service stores its own copy of the package and automatically encrypts that copy using AES-256 with OpenSearch Service-managed keys.

To import and associate a package with a domain (console)

1. In the Amazon OpenSearch Service console, choose **Packages**.
2. Choose **Import package**.
3. Give the package a descriptive name.
4. Provide the S3 path to the file, and then choose **Submit**.
5. Return to the **Packages** screen.
6. When the package status is **Available**, select it. Then choose **Associate to a domain**.
7. Select a domain, and then choose **Associate**.
8. In the navigation pane, choose your domain and go to the **Packages** tab.
9. When the package status is **Available**, note its ID. Use analyzers/*id* as the file path in [requests to OpenSearch \(p. 239\)](#).

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 OpenSearch Service API Reference](#).

Using custom packages with OpenSearch

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. Several objects support `synonyms_path` and `stopwords_path`, but `user_dictionary` is exclusive to the `kuromoji` plugin.

For the IK (Chinese) Analysis plugin, you can upload a custom dictionary file as a custom package and associate it to a domain, and the plugin automatically picks it up without requiring a `user_dictionary` parameter. If your file is a synonyms file, use the `synonyms_path` parameter.

The following example adds a synonyms file to a new index:

```
PUT my-index
{
  "settings": {
    "index": {
      "analysis": {
        "analyzer": {
          "my_analyzer": {
            "type": "custom",
            "tokenizer": "standard",
            "filter": ["my_filter"]
          }
        },
        "filter": {
          "my_filter": {
            "type": "synonym",
            "synonyms_path": "analyzers/F111111111",
            "updateable": true
          }
        }
      }
    },
    "mappings": {
      "properties": {
        "description": {
          "type": "text",
          "analyzer": "standard",
          "search_analyzer": "my_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 OpenSearch to use the new analyzer as its search analyzer. You can see that it still uses the standard analyzer as its index analyzer.

Finally, note the line `"updateable": true` in the token filter. This field only applies to search analyzers, not index analyzers, and is critical if you later want to [update the search analyzer \(p. 241\)](#) automatically.

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, OpenSearch 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"  
      }  
    }]  
  }  
}
```

Tip

Dictionary files use Java heap space proportional to their size. For example, a 2 GiB dictionary file might consume 2 GiB of heap space on a node. If you use large files, ensure that your nodes have enough heap space to accommodate them. [Monitor \(p. 69\)](#) the `JVMMemoryPressure` metric, and scale your cluster as necessary.

Updating custom packages (console)

Uploading a new version of a package to Amazon S3 does *not* automatically update the package on Amazon OpenSearch Service. OpenSearch Service stores its own copy of the file, so if you upload a new version to S3, you must manually update it.

Each of your associated domains stores *its* own copy of the file, as well. To keep search behavior predictable, domains continue to use their current package version until you explicitly update them. To update a custom package, modify the file in Amazon S3 Control, update the package in OpenSearch Service, and then apply the update.

1. In the OpenSearch Service console, choose **Packages**.
2. Choose a package and **Update**.
3. Provide the S3 path to the file, and then choose **Update package**.

4. Return to the **Packages** screen.
5. When the package status changes to **Available**, select it. Then choose one or more associated domains, **Apply update**, and confirm. Wait for the association status to change to **Active**.
6. The next steps vary depending on how you configured your indices:
 - If your domains runs OpenSearch or Elasticsearch 7.8 or later and only uses search analyzers with the [updateable \(p. 239\)](#) field set to true, you don't need to take any further action. OpenSearch Service automatically updates your indices using the [_plugins/_refresh_search_analyzers API](#).
 - If your domain runs Elasticsearch 7.7 or earlier, uses index analyzers, or doesn't use the updateable field, see [the section called "Manual index updates" \(p. 243\)](#).

Although the console is the simplest method, you can also use the AWS CLI, SDKs, or configuration API to update OpenSearch Service packages. For more information, see the [AWS CLI Command Reference](#) and [Amazon OpenSearch Service API Reference](#).

Updating custom packages (AWS SDK)

Instead of manually updating a package in the console, you can use the SDKs to automate the update process. The following sample Python script uploads a new package file to Amazon S3, updates the package in OpenSearch Service, and applies the new package to the specified domain. After confirming the update was successful, it makes a sample call to OpenSearch demonstrating the new synonyms have been applied.

You must provide values for `host`, `region`, `file_name`, `bucket_name`, `s3_key`, `package_id`, `domain_name`, and `query`.

```
from requests_aws4auth import AWS4Auth
import boto3
import requests
import time
import json
import sys

host = '' # The OpenSearch domain endpoint with https:// and a trailing slash. For example, https://my-test-domain.us-east-1.es.amazonaws.com/
region = '' # For example, us-east-1
file_name = '' # The path to the file to upload
bucket_name = '' # The name of the S3 bucket to upload to
s3_key = '' # The name of the S3 key (file name) to upload to
package_id = '' # The unique identifier of the OpenSearch package to update
domain_name = '' # The domain to associate the package with
query = '' # A test query to confirm the package has been successfully updated

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

def upload_to_s3(file_name, bucket_name, s3_key):
    """Uploads file to S3"""
    s3 = boto3.client('s3')
    try:
        s3.upload_file(file_name, bucket_name, s3_key)
        print('Upload successful')
        return True
    except FileNotFoundError:
        sys.exit('File not found. Make sure you specified the correct file path.')

upload_to_s3(file_name, bucket_name, s3_key)
```

```
def update_package(package_id, bucket_name, s3_key):
    """Updates the package in OpenSearch Service"""
    print(package_id, bucket_name, s3_key)
    response = client.update_package(
        PackageID=package_id,
        PackageSource={
            'S3BucketName': bucket_name,
            'S3Key': s3_key
        }
    )
    print(response)

def associate_package(package_id, domain_name):
    """Associates the package to the domain"""
    response = client.associate_package(
        PackageID=package_id, DomainName=domain_name)
    print(response)
    print('Associating...')

def wait_for_update(domain_name, package_id):
    """Waits for the package to be updated"""
    response = client.list_packages_for_domain(DomainName=domain_name)
    package_details = response['DomainPackageDetailsList']
    for package in package_details:
        if package['PackageID'] == package_id:
            status = package['DomainPackageStatus']
            if status == 'ACTIVE':
                print('Association successful.')
                return
            elif status == 'ASSOCIATION_FAILED':
                sys.exit('Association failed. Please try again.')
            else:
                time.sleep(10) # Wait 10 seconds before rechecking the status
                wait_for_update(domain_name, package_id)

def sample_search(query):
    """Makes a sample search call to OpenSearch"""
    path = '_search'
    params = {'q': query}
    url = host + path
    response = requests.get(url, params=params, auth=awsauth)
    print('Searching for ' + '""' + query + '"')
    print(response.text)
```

Note

If you receive a "package not found" error when you run the script using the AWS CLI, it likely means Boto3 is using whichever Region is specified in `~/.aws/config`, which isn't the Region your S3 bucket is in. Either run `aws configure` and specify the correct Region, or explicitly add the Region to the client:

```
client = boto3.client('opensearch', region_name='us-east-1')
```

Manual index updates

To use an updated package, you must manually update your indexes if you meet any of the following conditions:

- Your domain runs Elasticsearch 7.7 or earlier.

- You use custom packages as index analyzers.
- You use custom packages as search analyzers, but don't include the [updateable \(p. 239\)](#) field.

To update analyzers with the new package files, you have two options:

- Close and open any indexes that you want to update:

```
POST my-index/_close
POST my-index/_open
```

- Reindex the indexes. First, create an index that uses the updated synonyms file (or an entirely new 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 index analyzers, 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:

```
DELETE my-index
```

Dissociating and removing packages

Dissociating a package from a domain means that you can no longer use that file when you create new indexes. Any indexes 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 OpenSearch Service. Removing a package from OpenSearch Service does *not* remove it from its original location on Amazon S3.

To dissociate a package from a domain and remove it from OpenSearch Service (console)

1. Go to <https://aws.amazon.com>, and then choose **Sign In to the Console**.
2. Under **Analytics**, choose **Amazon OpenSearch Service**.
3. In the navigation pane, choose your domain, and then choose the **Packages** tab.
4. Select 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 OpenSearch Service API Reference](#).

Querying your Amazon OpenSearch Service data with SQL

You can use SQL to query your Amazon OpenSearch Service, rather than using the JSON-based [OpenSearch query DSL](#). Querying with SQL is useful if you're already familiar with the language or want to integrate your domain with an application that uses it.

Use the following table to find the version of the SQL plugin that's supported by each OpenSearch and Elasticsearch version.

OpenSearch

OpenSearch version	SQL plugin version	Notable features
2.3.0	2.3.0.0	Add maketime and makedate datetime functions
1.3.0	1.3.0.0	Support default query limit size, and IN clause to select from within a value list
1.2.0	1.2.0.0	Add new protocol for visualization response format
1.1.0	1.1.0.0	Support match function as filter in SQL and PPL
1.0.0	1.0.0.0	Support querying a data stream

Open Distro for Elasticsearch

Elasticsearch version	SQL plugin version	Notable features
7.10	1.13.0	NULL FIRST and LAST for window functions, CAST() function, SHOW and DESCRIBE commands
7.9	1.11.0	Add additional date/time functions, ORDER BY keyword
7.8	1.9.0	
7.7	1.8.0	
7.3	1.3.0	Multiple string and number operators
7.1	1.1.0	

SQL support is available on domains running OpenSearch or Elasticsearch 6.5 or higher. Full documentation of the SQL plugin is available in the [OpenSearch documentation](#).

Sample call

To query your data with SQL, send HTTP requests to `_sql` using the following format:

```
POST domain-endpoint/_plugins/_sql
{
  "query": "SELECT * FROM my-index LIMIT 50"
}
```

Note

If your domain is running Elasticsearch rather than OpenSearch, the format is `_opendistro/_sql`.

Notes and differences

Calls to `_plugins/_sql` include index names in the request body, so they have the same [access policy considerations \(p. 139\)](#) as the bulk, mget, and msearch operations. As always, follow the principle of [least privilege](#) when you grant permissions to API operations.

For security considerations related to using SQL with fine-grained access control, see [the section called "Fine-grained access control" \(p. 146\)](#).

The OpenSearch SQL plugin includes many [tunable settings](#). In OpenSearch Service, use the _cluster/settings path, not the plugin settings path (_plugins/_query/settings):

```
PUT _cluster/settings
{
  "transient" : {
    "plugins.sql.enabled" : true
  }
}
```

For legacy Elasticsearch domains, replace plugins with opendistro:

```
PUT _cluster/settings
{
  "transient" : {
    "opendistro.sql.enabled" : true
  }
}
```

SQL Workbench

The SQL Workbench is an OpenSearch Dashboards user interface that lets you run on-demand SQL queries, translate SQL into its REST equivalent, and view and save results as text, JSON, JDBC, or CSV. For more information, see [Query Workbench](#).

SQL CLI

The SQL CLI is a standalone Python application that you can launch with the `opensearchsql` command. For steps to install, configure, and use, see [SQL CLI](#).

JDBC driver

The Java Database Connectivity (JDBC) driver lets you integrate OpenSearch Service domains with your favorite business intelligence (BI) applications. To download the driver, click [here](#). For more information, see the [GitHub repository](#).

The following tables summarize version compatibility for the driver.

OpenSearch

OpenSearch version	JDBC driver version
1.3	1.1.0.1
1.2	1.1.0.1
1.1	1.1.0.1
1.0	1.1.0.1

Open Distro for Elasticsearch

Elasticsearch version	JDBC driver version
7.10	1.13.0
7.9	1.11.0
7.8	1.9.0
7.7	1.8.0
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

ODBC driver

The Open Database Connectivity (ODBC) driver is a read-only ODBC driver for Windows and macOS that lets you connect business intelligence and data visualization applications like [Tableau](#), [Microsoft Excel](#), and [Power BI](#) to the SQL plugin.

The drivers are available for download on the OpenSearch [artifacts page](#). For information about installing the driver, see the [SQL repository on GitHub](#).

k-Nearest Neighbor (k-NN) search in Amazon OpenSearch Service

Short for its associated *k-nearest neighbors* algorithm, k-NN for Amazon OpenSearch Service lets you search for points in a vector space and find the "nearest neighbors" for those points by Euclidean distance or cosine similarity. Use cases include recommendations (for example, an "other songs you might like" feature in a music application), image recognition, and fraud detection.

Use the following tables to find the version of the k-NN plugin running on your Amazon OpenSearch Service domain. Each k-NN plugin version corresponds to an [OpenSearch](#) or [Elasticsearch](#) version.

OpenSearch

OpenSearch version	k-NN plugin version	Notable features
2.3	2.3.0.0	
1.3	1.3.0.0	
1.2	1.2.0.0	Added support for the Faiss library
1.1	1.1.0.0	

OpenSearch version	k-NN plugin version	Notable features
1.0	1.0.0.0	Renamed REST APIs while supporting backwards compatibility, renamed namespace from opendistro to opensearch

Elasticsearch

Elasticsearch version	k-NN plugin version	Notable features
7.1	1.3.0.0	Euclidean distance
7.4	1.4.0.0	
7.7	1.8.0.0	Cosine similarity
7.8	1.9.0.0	
7.9	1.11.0.0	Warmup API, custom scoring
7.10	1.13.0.0	Hamming distance, L1 Norm distance, Painless scripting

Full documentation for the k-NN plugin is available in the [OpenSearch documentation](#). For background information about the k-nearest neighbors algorithm, see [Wikipedia](#).

Getting started with k-NN

To use k-NN, 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. k-NN 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
      }
    }
  }
}
```

k-NN differences, tuning, and limitations

OpenSearch lets you modify all [k-NN settings](#) using the `_cluster/settings` API. On OpenSearch Service, you can change all settings except `knn.memory.circuit_breaker.enabled` and `knn.circuit_breaker.triggered`. k-NN statistics are included as [Amazon CloudWatch metrics \(p. 67\)](#).

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. OpenSearch Service uses half of an instance's RAM for the Java heap (up to a heap size of 32 GiB). By default, k-NN uses up to 50% of the remaining half, so an instance type with 32 GiB of RAM can accommodate 8 GiB of graphs ($32 * 0.5 * 0.5$). Performance can suffer if graph memory usage exceeds this value.

You can't migrate a k-NN index to [UltraWarm \(p. 286\)](#) or [cold storage \(p. 295\)](#) if the index uses [approximate k-NN](#) ("`index.knn": true`"). If `index.knn` is set to `false` ([exact k-NN](#)), you can still move the index to other storage tiers.

Cross-cluster search for Amazon OpenSearch Service

Cross-cluster search in Amazon OpenSearch Service lets you perform queries and aggregations across multiple connected domains. It often makes more sense to use multiple smaller domains instead of a single large domain, especially when you're running different types of workloads.

Workload-specific domains enable you to perform the following tasks:

- Optimize each domain by choosing instance types for specific workloads.
- Establish fault-isolation boundaries across workloads. This means that if one of your workloads fails, the fault is contained within that specific domain and doesn't impact your other workloads.
- Scale more easily across domains.

Cross-cluster search supports OpenSearch Dashboards, so you can create visualizations and dashboards across all your domains.

Topics

- [Limitations \(p. 251\)](#)
- [Cross-cluster search prerequisites \(p. 252\)](#)
- [Cross-cluster search pricing \(p. 252\)](#)
- [Setting up a connection \(p. 252\)](#)
- [Removing a connection \(p. 253\)](#)
- [Setting up security and sample walkthrough \(p. 253\)](#)
- [OpenSearch Dashboards \(p. 257\)](#)

Limitations

Cross-cluster search has several important limitations:

- You can't connect an Elasticsearch domain to an OpenSearch domain.
- You can't connect to self-managed OpenSearch/Elasticsearch clusters.

- To connect domains across Regions, both domains must be on Elasticsearch 7.10 or later or OpenSearch.
- A domain can have a maximum of 20 outgoing connections. Similarly, a domain can have a maximum of 20 incoming connections. In other words, one domain can connect to a maximum of 20 other domains.
- Domains must either share the same major version, or be on the final minor version and the next major version (for example, 6.8 and 7.x are compatible).
- You can't use custom dictionaries or SQL with cross-cluster search.
- You can't use AWS CloudFormation to connect domains.
- You can't use cross-cluster search on M3 or burstable (T2 and T3) instances.

Cross-cluster search prerequisites

Before you set up cross-cluster search, make sure that your domains meet the following requirements:

- Two OpenSearch domains, or Elasticsearch domains on version 6.7 or later
- Fine-grained access control enabled
- Node-to-node encryption enabled

Cross-cluster search pricing

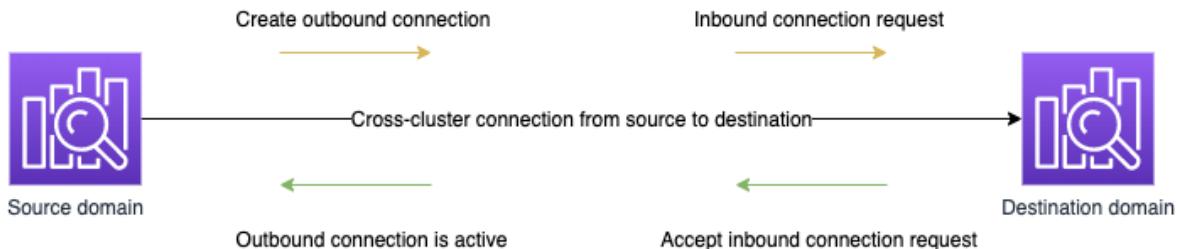
There is no additional charge for searching across domains.

Setting up a connection

The “source” domain refers to the domain that a cross-cluster search request originates from. In other words, the source domain is the one that you send the initial search request to.

The “destination” domain is the domain that the source domain queries.

A cross-cluster connection is unidirectional from the source to the destination domain. This means that the destination domain can’t query the source domain. However, you can set up another connection in the opposite direction.



The source domain creates an “outbound” connection to the destination domain. The destination domain receives an “inbound” connection request from the source domain.

To set up a connection

1. On your domain dashboard, choose a domain and go to the **Connections** tab.
2. In the **Outbound connections** section, choose **Request**.
3. For **Connection alias**, enter a name for your connection.
4. Choose between connecting to a domain in your AWS account and Region or in another account or Region.

- To connect to a cluster in your AWS account and Region, select the domain from the dropdown menu and choose **Request**.
 - To connect to a cluster in another AWS account or Region, select the ARN of the remote domain and choose **Request**. To connect domains across Regions, both domains must be running Elasticsearch version 7.10 or later or OpenSearch.
5. Cross-cluster search first validates the connection request to make sure the prerequisites are met. If the domains are found to be incompatible, the connection request enters the `Validation Failed` state.
6. After the connection request is validated successfully, it is sent to the destination domain, where it needs to be approved. Until this approval happens, the connection remains in a `Pending acceptance` state. When the connection request is accepted at the destination domain, the state changes to `Active` and the destination domain becomes available for queries.
- The domain page shows you the overall domain health and instance health details of your destination domain. Only domain owners have the flexibility to create, view, remove, and monitor connections to or from their domains.

After the connection is established, any traffic that flows between the nodes of the connected domains is encrypted. If you connect a VPC domain to a non-VPC domain and the non-VPC domain is a public endpoint that can receive traffic from the internet, the cross-cluster traffic between the domains is still encrypted and secure.

Removing a connection

Removing a connection stops any cross-cluster operation on its indices.

1. On your domain dashboard, go to the **Connections** tab.
2. Select the domain connections that you want to remove and choose **Delete**, then confirm deletion.

You can perform these steps on either the source or destination domain to remove the connection. After you remove the connection, it's still visible with a `Deleted` status for a period of 15 days.

You can't delete a domain with active cross-cluster connections. To delete a domain, first remove all incoming and outgoing connections from that domain. This ensures you take into account the cross-cluster domain users before deleting the domain.

Setting up security and sample walkthrough

1. You send a cross-cluster search request to the source domain.
2. The source domain evaluates that request against its domain access policy. Because cross-cluster search requires fine-grained access control, we recommend an open access policy on the source domain.

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

```

        ],
        "Resource": "arn:aws:es:region:account:domain/src-domain/*"
    }
]
}
]
```

Note

If you include remote indexes in the path, you must URL-encode the URI in the domain ARN. For example, use `arn:aws:es:us-east-1:123456789012:domain/my-domain/local_index,dst%3Aremote_index` rather than `arn:aws:es:us-east-1:123456789012:domain/my-domain/local_index,dst:remote_index`.

If you choose to use a restrictive access policy in addition to fine-grained access control, your policy must allow access to `es:ESHttpGet` at a minimum.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:user/test-user"
                ]
            },
            "Action": "es:ESHttpGet",
            "Resource": "arn:aws:es:region:account:domain/src-domain/*"
        }
    ]
}
```

3. [Fine-grained access control \(p. 146\)](#) on the source domain evaluates the request:

- Is the request signed with valid IAM or HTTP basic credentials?
- If so, does the user have permission to perform the search and access the data?

If the request only searches data on the destination domain (for example, `dest-alias:dest-index/_search`), you only need permissions on the destination domain.

If the request searches data on both domains (for example, `source-index,dest-alias:dest-index/_search`), you need permissions on both domains.

In fine-grained access control, users must have the `indices:admin/shards/search_shards` permission in addition to standard `read` or `search` permissions for the relevant indices.

4. The source domain passes the request to the destination domain. The destination domain evaluates this request against its domain access policy. You must include the `es:ESCrossClusterGet` permission on the destination domain:

```

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

```
}
```

Make sure that the `es:ESCrossClusterGet` permission is applied for `/dst-domain` and not `/dst-domain/*`.

However, this minimum policy only allows cross-cluster searches. To perform other operations, such as indexing documents and performing standard searches, you need additional permissions. We recommend the following policy on the destination domain:

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

Note

All cross-cluster search requests between domains are encrypted in transit by default as part of node-to-node encryption.

5. The destination domain performs the search and returns the results to the source domain.
6. The source domain combines its own results (if any) with the results from the destination domain and returns them to you.
7. We recommend [Postman](#) for testing requests:

- On the destination domain, index a document:

```
POST https://dst-domain.us-east-1.es.amazonaws.com/books/_doc/1
{
  "Dracula": "Bram Stoker"
}
```

- To query this index from the source domain, include the connection alias of the destination domain within the query.

```
GET https://src-domain.us-east-1.es.amazonaws.com/<connection_alias>:books/_search
{
  ...
  "hits": [
    {

```

```
        "_index": "source-destination:books",
        "_type": "_doc",
        "_id": "1",
        "_score": 1,
        "_source": {
            "Dracula": "Bram Stoker"
        }
    }
]
```

You can find the connection alias on the **Connections** tab on your domain dashboard.

- If you set up a connection between domain-a -> domain-b with connection alias `cluster_b` and domain-a -> domain-c with connection alias `cluster_c`, search domain-a, domain-b, and domain-c as follows:

```
GET https://src-domain.us-east-1.es.amazonaws.com/
local_index,cluster_b:b_index,cluster_c:c_index/_search
{
  "query": {
    "match": {
      "user": "domino"
    }
  }
}
```

Response

```
{
  "took": 150,
  "timed_out": false,
  "_shards": {
    "total": 3,
    "successful": 3,
    "failed": 0,
    "skipped": 0
  },
  "_clusters": {
    "total": 3,
    "successful": 3,
    "skipped": 0
  },
  "hits": {
    "total": 3,
    "max_score": 1,
    "hits": [
      {
        "_index": "local_index",
        "_type": "_doc",
        "_id": "0",
        "_score": 1,
        "_source": {
          "user": "domino",
          "message": "Lets unite the new mutants",
          "likes": 0
        }
      },
      {
        "_index": "cluster_b:b_index",
        "_type": "_doc",
        "_id": "0",
        "_score": 2,
        "
```

```
        "_source": {
            "user": "domino",
            "message": "I'm different",
            "likes": 0
        }
    },
{
    "_index": "cluster_c:c_index",
    "_type": "_doc",
    "_id": "0",
    "_score": 3,
    "_source": {
        "user": "domino",
        "message": "So am I",
        "likes": 0
    }
}
]
```

All destination clusters that you search need to be available for your search request to run successfully. Otherwise, the whole request fails—even if one of the domains is not available, no search results are returned.

OpenSearch Dashboards

You can visualize data from multiple connected domains in the same way as from a single domain, except that you must access the remote indexes using `connection-alias:index`. So, your index pattern must match `connection-alias:index`.

Learning to Rank for Amazon OpenSearch Service

OpenSearch uses a probabilistic ranking framework called BM-25 to calculate relevance scores. If a distinctive keyword appears more frequently in a document, BM-25 assigns a higher relevance score to that document. This framework, however, doesn't take into account user behavior like click-through data, which can further improve relevance.

Learning to Rank is an open-source plugin that lets you use machine learning and behavioral data to tune the relevance of documents. It uses models from the XGBoost and Ranklib libraries to rescore the search results. The [Elasticsearch LTR plugin](#) was initially developed by [OpenSource Connections](#), with significant contributions by Wikimedia Foundation, Snagajob Engineering, Bonsai, and Yelp Engineering. The OpenSearch version of the plugin is derived from the Elasticsearch LTR plugin. Full documentation, including detailed steps and API descriptions, is available in the [Learning to Rank](#) documentation.

Learning to Rank requires OpenSearch or Elasticsearch 7.7 or later.

Note

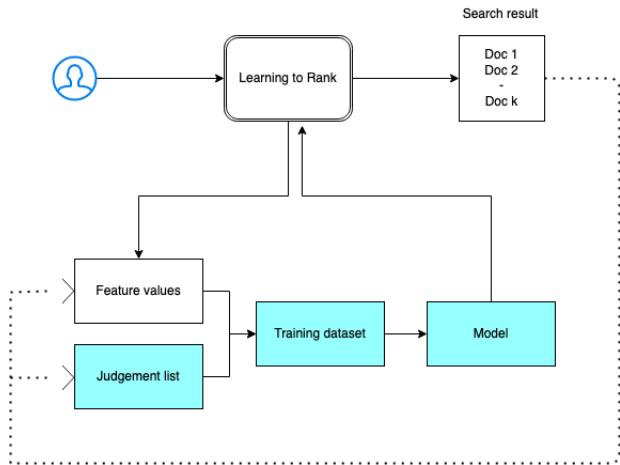
To use the Learning to Rank plugin, you must have full admin permissions. To learn more, see the section called “[Modifying the master user](#)” (p. 158).

Topics

- [Getting started with Learning to Rank \(p. 258\)](#)
- [Learning to Rank API \(p. 272\)](#)

Getting started with Learning to Rank

You need to provide a judgment list, prepare a training dataset, and train the model outside of Amazon OpenSearch Service. The parts in blue occur outside of OpenSearch Service:



Step 1: Initialize the plugin

To initialize the Learning to Rank plugin, send the following request to your OpenSearch Service domain:

```
PUT _ltr
```

```
{  
    "acknowledged" : true,  
    "shards_acknowledged" : true,  
    "index" : ".ltrstore"  
}
```

This command creates a hidden `.ltrstore` index that stores metadata information such as feature sets and models.

Step 2: Create a judgment list

Note

You must perform this step outside of OpenSearch Service.

A judgment list is a collection of examples that a machine learning model learns from. Your judgment list should include keywords that are important to you and a set of graded documents for each keyword.

In this example, we have a judgment list for a movie dataset. A grade of 4 indicates a perfect match. A grade of 0 indicates the worst match.

Grade	Keyword	Doc ID	Movie name
4	rambo	7555	Rambo
3	rambo	1370	Rambo III
3	rambo	1369	Rambo: First Blood Part II
3	rambo	1368	First Blood

Prepare your judgment list in the following format:

```
4 qid:1 # 7555 Rambo
3 qid:1 # 1370 Rambo III
3 qid:1 # 1369 Rambo: First Blood Part II
3 qid:1 # 1368 First Blood

where qid:1 represents "rambo"
```

For a more complete example of a judgment list, see [movie judgments](#).

You can create this judgment list manually with the help of human annotators or infer it programmatically from analytics data.

Step 3: Build a feature set

A feature is a field that corresponds to the relevance of a document—for example, title, overview, popularity score (number of views), and so on.

Build a feature set with a Mustache template for each feature. For more information about features, see [Working with Features](#).

In this example, we build a movie_features feature set with the title and overview fields:

```
POST _ltr/_featureset/movie_features
{
  "featureset" : {
    "name" : "movie_features",
    "features" : [
      {
        "name" : "1",
        "params" : [
          "keywords"
        ],
        "template_language" : "mustache",
        "template" : {
          "match" : {
            "title" : "{{keywords}}"
          }
        }
      },
      {
        "name" : "2",
        "params" : [
          "keywords"
        ],
        "template_language" : "mustache",
        "template" : {
          "match" : {
            "overview" : "{{keywords}}"
          }
        }
      }
    ]
  }
}
```

If you query the original .ltrstore index, you get back your feature set:

```
GET _ltr/_featureset
```

Step 4: Log the feature values

The feature values are the relevance scores calculated by BM-25 for each feature.

Combine the feature set and judgment list to log the feature values. For more information about logging features, see [Logging Feature Scores](#).

In this example, the `bool` query retrieves the graded documents with the filter, and then selects the feature set with the `sltr` query. The `ltr_log` query combines the documents and the features to log the corresponding feature values:

```
POST tmdb/_search
{
  "_source": {
    "includes": [
      "title",
      "overview"
    ]
  },
  "query": {
    "bool": {
      "filter": [
        {
          "terms": {
            "_id": [
              "7555",
              "1370",
              "1369",
              "1368"
            ]
          }
        }
      ]
    }
  },
  {
    "sltr": {
      "_name": "logged_featureset",
      "featureset": "movie_features",
      "params": {
        "keywords": "rambo"
      }
    }
  }
},
"ext": {
  "ltr_log": {
    "log_specs": {
      "name": "log_entry1",
      "named_query": "logged_featureset"
    }
  }
}
}
```

A sample response might look like the following:

```
{
  "took" : 7,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
```

```

        "failed" : 0
    },
    "hits" : {
        "total" : {
            "value" : 4,
            "relation" : "eq"
        },
        "max_score" : 0.0,
        "hits" : [
            {
                "_index" : "tmdb",
                "_type" : "movie",
                "_id" : "1368",
                "_score" : 0.0,
                "_source" : {
                    "overview" : "When former Green Beret John Rambo is harassed by local law enforcement and arrested for vagrancy, the Vietnam vet snaps, runs for the hills and rats-a-tat-tats his way into the action-movie hall of fame. Hounded by a relentless sheriff, Rambo employs heavy-handed guerilla tactics to shake the cops off his tail.",
                    "title" : "First Blood"
                },
                "fields" : {
                    "_ltrlog" : [
                        {
                            "log_entry1" : [
                                {
                                    "name" : "1"
                                },
                                {
                                    "name" : "2",
                                    "value" : 10.558305
                                }
                            ]
                        }
                    ]
                }
            },
            "matched_queries" : [
                "logged_featureset"
            ]
        },
        {
            "_index" : "tmdb",
            "_type" : "movie",
            "_id" : "7555",
            "_score" : 0.0,
            "_source" : {
                "overview" : "When governments fail to act on behalf of captive missionaries, ex-Green Beret John James Rambo sets aside his peaceful existence along the Salween River in a war-torn region of Thailand to take action. Although he's still haunted by violent memories of his time as a U.S. soldier during the Vietnam War, Rambo can hardly turn his back on the aid workers who so desperately need his help.",
                "title" : "Rambo"
            },
            "fields" : {
                "_ltrlog" : [
                    {
                        "log_entry1" : [
                            {
                                "name" : "1",
                                "value" : 11.2569065
                            },
                            {
                                "name" : "2",
                                "value" : 9.936821
                            }
                        ]
                    }
                ]
            }
        }
    ]
}

```

```
        }
    ],
},
"matched_queries" : [
    "logged_featureset"
],
{
    "_index" : "tmdb",
    "_type" : "movie",
    "_id" : "1369",
    "_score" : 0.0,
    "_source" : {
        "overview" : "Col. Troutman recruits ex-Green Beret John Rambo for a highly secret and dangerous mission. Teamed with Co Bao, Rambo goes deep into Vietnam to rescue POWs. Deserted by his own team, he's left in a hostile jungle to fight for his life, avenge the death of a woman and bring corrupt officials to justice.",
        "title" : "Rambo: First Blood Part II"
    },
    "fields" : {
        "_ltrlog" : [
            {
                "log_entry1" : [
                    {
                        "name" : "1",
                        "value" : 6.334839
                    },
                    {
                        "name" : "2",
                        "value" : 10.558305
                    }
                ]
            }
        ]
    },
    "matched_queries" : [
        "logged_featureset"
    ],
{
    "_index" : "tmdb",
    "_type" : "movie",
    "_id" : "1370",
    "_score" : 0.0,
    "_source" : {
        "overview" : "Combat has taken its toll on Rambo, but he's finally begun to find inner peace in a monastery. When Rambo's friend and mentor Col. Trautman asks for his help on a top secret mission to Afghanistan, Rambo declines but must reconsider when Trautman is captured.",
        "title" : "Rambo III"
    },
    "fields" : {
        "_ltrlog" : [
            {
                "log_entry1" : [
                    {
                        "name" : "1",
                        "value" : 9.425955
                    },
                    {
                        "name" : "2",
                        "value" : 11.262714
                    }
                ]
            }
        ]
    }
}
```

```
        },
        "matched_queries" : [
            "logged_featureset"
        ]
    }
}
```

In the previous example, the first feature doesn't have a feature value because the keyword "rambo" doesn't appear in the title field of the document with an ID equal to 1368. This is a missing feature value in the training data.

Step 5: Create a training dataset

Note

You must perform this step outside of OpenSearch Service.

The next step is to combine the judgment list and feature values to create a training dataset. If your original judgment list looks like this:

```
4 qid:1 # 7555 Rambo
3 qid:1 # 1370 Rambo III
3 qid:1 # 1369 Rambo: First Blood Part II
3 qid:1 # 1368 First Blood
```

Convert it into the final training dataset, which looks like this:

```
4 qid:1 1:12.318474 2:10.573917 # 7555 rambo
3 qid:1 1:10.357875 2:11.950391 # 1370 rambo
3 qid:1 1:7.010513 2:11.220095 # 1369 rambo
3 qid:1 1:0.0 2:11.220095 # 1368 rambo
```

You can perform this step manually or write a program to automate it.

Step 6: Choose an algorithm and build the model

Note

You must perform this step outside of OpenSearch Service.

With the training dataset in place, the next step is to use XGBoost or Ranklib libraries to build a model. XGBoost and Ranklib libraries let you build popular models such as LambdaMART, Random Forests, and so on.

For steps to use XGBoost and Ranklib to build the model, see the [XGBoost](#) and [RankLib](#) documentation, respectively. To use Amazon SageMaker to build the XGBoost model, see [XGBoost Algorithm](#).

Step 7: Deploy the model

After you have built the model, deploy it into the Learning to Rank plugin. For more information about deploying a model, see [Uploading A Trained Model](#).

In this example, we build a `my_ranklib_model` model using the Ranklib library:

```
POST _ltr/_featureset/movie_features/_createmodel?pretty
{
    "model": {
        "name": "my_ranklib_model",
        "model": {
```

```
    "type": "model/ranklib",
    "definition": """## LambdaMART
## No. of trees = 10
## No. of leaves = 10
## No. of threshold candidates = 256
## Learning rate = 0.1
## Stop early = 100

<ensemble>
  <tree id="1" weight="0.1">
    <split>
      <feature>1</feature>
      <threshold>10.357875</threshold>
      <split pos="left">
        <feature>1</feature>
        <threshold>0.0</threshold>
        <split pos="left">
          <output>-2.0</output>
        </split>
        <split pos="right">
          <feature>1</feature>
          <threshold>7.010513</threshold>
          <split pos="left">
            <output>-2.0</output>
          </split>
          <split pos="right">
            <output>-2.0</output>
          </split>
        </split>
      </split>
      <split pos="right">
        <output>2.0</output>
      </split>
    </split>
  </tree>
  <tree id="2" weight="0.1">
    <split>
      <feature>1</feature>
      <threshold>10.357875</threshold>
      <split pos="left">
        <feature>1</feature>
        <threshold>0.0</threshold>
        <split pos="left">
          <output>-1.67031991481781</output>
        </split>
        <split pos="right">
          <feature>1</feature>
          <threshold>7.010513</threshold>
          <split pos="left">
            <output>-1.67031991481781</output>
          </split>
          <split pos="right">
            <output>-1.6703200340270996</output>
          </split>
        </split>
      </split>
      <split pos="right">
        <output>1.6703201532363892</output>
      </split>
    </split>
  </tree>
  <tree id="3" weight="0.1">
    <split>
      <feature>2</feature>
      <threshold>10.573917</threshold>
      <split pos="left">
```

```
<output>1.479954481124878</output>
</split>
<split pos="right">
    <feature>1</feature>
    <threshold>7.010513</threshold>
    <split pos="left">
        <feature>1</feature>
        <threshold>0.0</threshold>
        <split pos="left">
            <output>-1.4799546003341675</output>
        </split>
        <split pos="right">
            <output>-1.479954481124878</output>
        </split>
    </split>
    <split pos="right">
        <output>-1.479954481124878</output>
    </split>
</split>
</tree>
<tree id="4" weight="0.1">
    <split>
        <feature>1</feature>
        <threshold>10.357875</threshold>
        <split pos="left">
            <feature>1</feature>
            <threshold>0.0</threshold>
            <split pos="left">
                <output>-1.3569872379302979</output>
            </split>
            <split pos="right">
                <feature>1</feature>
                <threshold>7.010513</threshold>
                <split pos="left">
                    <output>-1.3569872379302979</output>
                </split>
                <split pos="right">
                    <output>-1.3569872379302979</output>
                </split>
            </split>
        </split>
        <split pos="right">
            <output>1.3569873571395874</output>
        </split>
    </split>
</tree>
<tree id="5" weight="0.1">
    <split>
        <feature>1</feature>
        <threshold>10.357875</threshold>
        <split pos="left">
            <feature>1</feature>
            <threshold>0.0</threshold>
            <split pos="left">
                <output>-1.2721362113952637</output>
            </split>
            <split pos="right">
                <feature>1</feature>
                <threshold>7.010513</threshold>
                <split pos="left">
                    <output>-1.2721363306045532</output>
                </split>
                <split pos="right">
                    <output>-1.2721363306045532</output>
                </split>
            </split>
        </split>
    </split>
```

```
</split>
</split>
<split pos="right">
    <output>1.2721362113952637</output>
</split>
</split>
</tree>
<tree id="6" weight="0.1">
    <split>
        <feature>1</feature>
        <threshold>10.357875</threshold>
        <split pos="left">
            <feature>1</feature>
            <threshold>7.010513</threshold>
            <split pos="left">
                <feature>1</feature>
                <threshold>0.0</threshold>
                <split pos="left">
                    <output>-1.2110036611557007</output>
                </split>
                <split pos="right">
                    <output>-1.2110036611557007</output>
                </split>
            </split>
            <split pos="right">
                <output>-1.2110037803649902</output>
            </split>
        </split>
        <split pos="right">
            <output>1.2110037803649902</output>
        </split>
    </split>
</tree>
<tree id="7" weight="0.1">
    <split>
        <feature>1</feature>
        <threshold>10.357875</threshold>
        <split pos="left">
            <feature>1</feature>
            <threshold>7.010513</threshold>
            <split pos="left">
                <feature>1</feature>
                <threshold>0.0</threshold>
                <split pos="left">
                    <output>-1.165616512298584</output>
                </split>
                <split pos="right">
                    <output>-1.165616512298584</output>
                </split>
            </split>
            <split pos="right">
                <output>-1.165616512298584</output>
            </split>
        </split>
        <split pos="right">
            <output>1.165616512298584</output>
        </split>
    </split>
</tree>
<tree id="8" weight="0.1">
    <split>
        <feature>1</feature>
        <threshold>10.357875</threshold>
        <split pos="left">
            <feature>1</feature>
            <threshold>7.010513</threshold>
```

```
<split pos="left">
    <feature>1</feature>
    <threshold>0.0</threshold>
    <split pos="left">
        <output>-1.131177544593811</output>
    </split>
    <split pos="right">
        <output>-1.131177544593811</output>
    </split>
</split>
<split pos="right">
    <output>-1.131177544593811</output>
</split>
</split>
<split pos="right">
    <output>1.131177544593811</output>
</split>
</split>
</tree>
<tree id="9" weight="0.1">
    <split>
        <feature>2</feature>
        <threshold>10.573917</threshold>
        <split pos="left">
            <output>1.1046180725097656</output>
        </split>
        <split pos="right">
            <feature>1</feature>
            <threshold>7.010513</threshold>
            <split pos="left">
                <feature>1</feature>
                <threshold>0.0</threshold>
                <split pos="left">
                    <output>-1.1046180725097656</output>
                </split>
                <split pos="right">
                    <output>-1.1046180725097656</output>
                </split>
            </split>
            <split pos="right">
                <output>-1.1046180725097656</output>
            </split>
        </split>
    </split>
</tree>
<tree id="10" weight="0.1">
    <split>
        <feature>1</feature>
        <threshold>10.357875</threshold>
        <split pos="left">
            <feature>1</feature>
            <threshold>7.010513</threshold>
            <split pos="left">
                <feature>1</feature>
                <threshold>0.0</threshold>
                <split pos="left">
                    <output>-1.0838804244995117</output>
                </split>
                <split pos="right">
                    <output>-1.0838804244995117</output>
                </split>
            </split>
            <split pos="right">
                <output>-1.0838804244995117</output>
            </split>
        </split>
    </split>
</tree>
```

```
<split pos="right">
    <output>1.0838804244995117</output>
</split>
</tree>
</ensemble>
"""
}
}
```

To see the model, send the following request:

```
GET _ltr/_model/my_ranklib_model
```

Step 8: Search with learning to rank

After you deploy the model, you're ready to search.

Perform the `sltr` query with the features that you're using and the name of the model that you want to execute:

```
POST tmdb/_search
{
    "_source": {
        "includes": ["title", "overview"]
    },
    "query": {
        "multi_match": {
            "query": "rambo",
            "fields": ["title", "overview"]
        }
    },
    "rescore": {
        "query": {
            "rescore_query": {
                "sltr": {
                    "params": {
                        "keywords": "rambo"
                    },
                    "model": "my_ranklib_model"
                }
            }
        }
    }
}
```

With Learning to Rank, you see “Rambo” as the first result because we have assigned it the highest grade in the judgment list:

```
{
    "took" : 12,
    "timed_out" : false,
    "_shards" : {
        "total" : 1,
        "successful" : 1,
        "skipped" : 0,
        "failed" : 0
    },
    "hits" : {
        "total" : {
```

```
        "value" : 7,
        "relation" : "eq"
    },
    "max_score" : 13.096414,
    "hits" : [
        {
            "_index" : "tmdb",
            "_type" : "movie",
            "_id" : "7555",
            "_score" : 13.096414,
            "_source" : {
                "overview" : "When governments fail to act on behalf of captive missionaries, ex-Green Beret John James Rambo sets aside his peaceful existence along the Salween River in a war-torn region of Thailand to take action. Although he's still haunted by violent memories of his time as a U.S. soldier during the Vietnam War, Rambo can hardly turn his back on the aid workers who so desperately need his help.",
                "title" : "Rambo"
            }
        },
        {
            "_index" : "tmdb",
            "_type" : "movie",
            "_id" : "1370",
            "_score" : 11.17245,
            "_source" : {
                "overview" : "Combat has taken its toll on Rambo, but he's finally begun to find inner peace in a monastery. When Rambo's friend and mentor Col. Trautman asks for his help on a top secret mission to Afghanistan, Rambo declines but must reconsider when Trautman is captured.",
                "title" : "Rambo III"
            }
        },
        {
            "_index" : "tmdb",
            "_type" : "movie",
            "_id" : "1368",
            "_score" : 10.442155,
            "_source" : {
                "overview" : "When former Green Beret John Rambo is harassed by local law enforcement and arrested for vagrancy, the Vietnam vet snaps, runs for the hills and rats-a-tat-tats his way into the action-movie hall of fame. Hounded by a relentless sheriff, Rambo employs heavy-handed guerilla tactics to shake the cops off his tail.",
                "title" : "First Blood"
            }
        },
        {
            "_index" : "tmdb",
            "_type" : "movie",
            "_id" : "1369",
            "_score" : 10.442155,
            "_source" : {
                "overview" : "Col. Troutman recruits ex-Green Beret John Rambo for a highly secret and dangerous mission. Teamed with Co Bao, Rambo goes deep into Vietnam to rescue POWs. Deserted by his own team, he's left in a hostile jungle to fight for his life, avenge the death of a woman and bring corrupt officials to justice.",
                "title" : "Rambo: First Blood Part II"
            }
        },
        {
            "_index" : "tmdb",
            "_type" : "movie",
            "_id" : "31362",
            "_score" : 7.424202,
            "_source" : {
                "overview" : "It is 1985, and a small, tranquil Florida town is being rocked by a wave of vicious serial murders and bank robberies. Particularly sickening to the
```

```

authorities is the gratuitous use of violence by two "Rambo" like killers who dress
themselves in military garb. Based on actual events taken from FBI files, the movie
depicts the Bureau's efforts to track down these renegades.",
    "title" : "In the Line of Duty: The F.B.I. Murders"
}
},
{
    "_index" : "tmdb",
    "_type" : "movie",
    "_id" : "13258",
    "_score" : 6.43182,
    "_source" : {
        "overview" : """Will Proudfoot (Bill Milner) is looking for an escape from his
family's stifling home life when he encounters Lee Carter (Will Poulter), the school
bully. Armed with a video camera and a copy of "Rambo: First Blood", Lee plans to make
cinematic history by filming his own action-packed video epic. Together, these two
newfound friends-turned-budding-filmmakers quickly discover that their imaginative – and
sometimes mishap-filled – cinematic adventure has begun to take on a life of its own!""",
        "title" : "Son of Rambow"
    }
},
{
    "_index" : "tmdb",
    "_type" : "movie",
    "_id" : "61410",
    "_score" : 3.9719706,
    "_source" : {
        "overview" : "It's South Africa 1990. Two major events are about to happen: The
release of Nelson Mandela and, more importantly, it's Spud Milton's first year at an elite
boys only private boarding school. John Milton is a boy from an ordinary background who
wins a scholarship to a private school in Kwazulu-Natal, South Africa. Surrounded by boys
with nicknames like Gecko, Rambo, Rain Man and Mad Dog, Spud has his hands full trying to
adapt to his new home. Along the way Spud takes his first tentative steps along the path
to manhood. (The path it seems could be a rather long road). Spud is an only child. He
is cursed with parents from well beyond the lunatic fringe and a senile granny. His dad
is a fervent anti-communist who is paranoid that the family domestic worker is running
a shebeen from her room at the back of the family home. His mom is a free spirit and
a teenager's worst nightmare, whether it's shopping for Spud's underwear in the local
supermarket",
        "title" : "Spud"
    }
}
]
}
]
```

If you search without using the Learning to Rank plugin, OpenSearch returns different results:

```

POST tmdb/_search
{
    "_source": {
        "includes": ["title", "overview"]
    },
    "query": {
        "multi_match": {
            "query": "Rambo",
            "fields": ["title", "overview"]
        }
    }
}
```

```
{
    "took" : 5,
    "timed_out" : false,
```

```

    "_shards" : {
        "total" : 1,
        "successful" : 1,
        "skipped" : 0,
        "failed" : 0
    },
    "hits" : {
        "total" : {
            "value" : 5,
            "relation" : "eq"
        },
        "max_score" : 11.262714,
        "hits" : [
            {
                "_index" : "tmdb",
                "_type" : "movie",
                "_id" : "1370",
                "_score" : 11.262714,
                "_source" : {
                    "overview" : "Combat has taken its toll on Rambo, but he's finally begun to find inner peace in a monastery. When Rambo's friend and mentor Col. Trautman asks for his help on a top secret mission to Afghanistan, Rambo declines but must reconsider when Trautman is captured.",
                    "title" : "Rambo III"
                }
            },
            {
                "_index" : "tmdb",
                "_type" : "movie",
                "_id" : "7555",
                "_score" : 11.2569065,
                "_source" : {
                    "overview" : "When governments fail to act on behalf of captive missionaries, ex-Green Beret John James Rambo sets aside his peaceful existence along the Salween River in a war-torn region of Thailand to take action. Although he's still haunted by violent memories of his time as a U.S. soldier during the Vietnam War, Rambo can hardly turn his back on the aid workers who so desperately need his help.",
                    "title" : "Rambo"
                }
            },
            {
                "_index" : "tmdb",
                "_type" : "movie",
                "_id" : "1368",
                "_score" : 10.558305,
                "_source" : {
                    "overview" : "When former Green Beret John Rambo is harassed by local law enforcement and arrested for vagrancy, the Vietnam vet snaps, runs for the hills and rattat-tat-tats his way into the action-movie hall of fame. Hounded by a relentless sheriff, Rambo employs heavy-handed guerilla tactics to shake the cops off his tail.",
                    "title" : "First Blood"
                }
            },
            {
                "_index" : "tmdb",
                "_type" : "movie",
                "_id" : "1369",
                "_score" : 10.558305,
                "_source" : {
                    "overview" : "Col. Troutman recruits ex-Green Beret John Rambo for a highly secret and dangerous mission. Teamed with Co Bao, Rambo goes deep into Vietnam to rescue POWs. Deserted by his own team, he's left in a hostile jungle to fight for his life, avenge the death of a woman and bring corrupt officials to justice.",
                    "title" : "Rambo: First Blood Part II"
                }
            }
        ]
    }
}

```

```
{  
    "_index" : "tmdb",  
    "_type" : "movie",  
    "_id" : "13258",  
    "_score" : 6.4600153,  
    "_source" : {  
        "overview" : """Will Proudfoot (Bill Milner) is looking for an escape from his family's stifling home life when he encounters Lee Carter (Will Poulter), the school bully. Armed with a video camera and a copy of "Rambo: First Blood", Lee plans to make cinematic history by filming his own action-packed video epic. Together, these two newfound friends-turned-budding-filmmakers quickly discover that their imaginative – and sometimes mishap-filled – cinematic adventure has begun to take on a life of its own!""",  
        "title" : "Son of Rambow"  
    }  
}  
]  
}  
}
```

Based on how well you think the model is performing, adjust the judgment list and features. Then, repeat steps 2–8 to improve the ranking results over time.

Learning to Rank API

Use the Learning to Rank operations to programmatically work with feature sets and models.

Create store

Creates a hidden `.ltrstore` index that stores metadata information such as feature sets and models.

```
PUT _ltr
```

Delete store

Deletes the hidden `.ltrstore` index and resets the plugin.

```
DELETE _ltr
```

Create feature set

Creates a feature set.

```
POST _ltr/_featureset/<name_of_features>
```

Delete feature set

Deletes a feature set.

```
DELETE _ltr/_featureset/<name_of_feature_set>
```

Get feature set

Retrieves a feature set.

```
GET _ltr/_featureset/<name_of_feature_set>
```

Create model

Creates a model.

```
POST _ltr/_featureset/<name_of_feature_set>/_createmodel
```

Delete model

Deletes a model.

```
DELETE _ltr/_model/<name_of_model>
```

Get model

Retrieves a model.

```
GET _ltr/_model/<name_of_model>
```

Get stats

Provides information about how the plugin is behaving.

```
GET _ltr/_model/<name_of_model>
```

You can also filter by node and/or cluster:

```
GET _ltr/nodeID,nodeID,/stats/stat,stat
{
  "_nodes" : {
    "total" : 1,
    "successful" : 1,
    "failed" : 0
  },
  "cluster_name" : "873043598401:ltr-77",
  "stores" : {
    ".ltrstore" : {
      "model_count" : 1,
      "featureset_count" : 1,
      "feature_count" : 2,
      "status" : "green"
    }
  },
  "status" : "green",
  "nodes" : {
    "DjelK-_ZSfyzst05dhGGQA" : {
      "cache" : {
        "feature" : {
          "eviction_count" : 0,
          "miss_count" : 0,
          "entry_count" : 0,
          "memory_usage_in_bytes" : 0,
          "hit_count" : 0
        },
        "featureset" : {
          "eviction_count" : 2,
          "miss_count" : 2,
        }
      }
    }
  }
}
```

```

        "entry_count" : 0,
        "memory_usage_in_bytes" : 0,
        "hit_count" : 0
    },
    "model" : {
        "eviction_count" : 2,
        "miss_count" : 3,
        "entry_count" : 1,
        "memory_usage_in_bytes" : 3204,
        "hit_count" : 1
    }
},
"request_total_count" : 6,
"request_error_count" : 0
}
}
}

```

The statistics are provided at two levels, node and cluster, as specified in the following tables:

Node-level stats

Field name	Description
request_total_count	Total count of ranking requests.
request_error_count	Total count of unsuccessful requests.
cache	Statistics across all caches (features, featuresets, models). A cache hit occurs when a user queries the plugin and the model is already loaded into memory.
cache.eviction_count	Number of cache evictions.
cache.hit_count	Number of cache hits.
cache.miss_count	Number of cache misses. A cache miss occurs when a user queries the plugin and the model has not yet been loaded into memory.
cache.entry_count	Number of entries in the cache.
cache.memory_usage_in_bytes	Total memory used in bytes.
cache.cache_capacity_reached	Indicates if the cache limit is reached.

Cluster-level stats

Field name	Description
stores	Indicates where the feature sets and model metadata are stored. (The default is ".ltrstore". Otherwise, it's prefixed with ".ltrstore_ ", with a user supplied name).
stores.status	Status of the index.
stores.feature_sets	Number of feature sets.
stores.features_count	Number of features.

Field name	Description
stores.model_count	Number of models.
status	The plugin status based on the status of the feature store indices (red, yellow, or green) and circuit breaker state (open or closed).
cache.cache_capacity_reached	Indicates if the cache limit is reached.

Get cache stats

Returns statistics about the cache and memory usage.

```
GET _ltr/_cachestats

{
  "_nodes": {
    "total": 2,
    "successful": 2,
    "failed": 0
  },
  "cluster_name": "opensearch-cluster",
  "all": {
    "total": {
      "ram": 612,
      "count": 1
    },
    "features": {
      "ram": 0,
      "count": 0
    },
    "featuresets": {
      "ram": 612,
      "count": 1
    },
    "models": {
      "ram": 0,
      "count": 0
    }
  },
  "stores": {
    ".ltrstore": {
      "total": {
        "ram": 612,
        "count": 1
      },
      "features": {
        "ram": 0,
        "count": 0
      },
      "featuresets": {
        "ram": 612,
        "count": 1
      },
      "models": {
        "ram": 0,
        "count": 0
      }
    }
  },
  "nodes": {
    "total": 2,
    "successful": 2,
    "failed": 0
  }
}
```

```
"ejF6uutERF20wOFNOXB61A": {
    "name": "opensearch1",
    "hostname": "172.18.0.4",
    "stats": {
        "total": {
            "ram": 612,
            "count": 1
        },
        "features": {
            "ram": 0,
            "count": 0
        },
        "featuresets": {
            "ram": 612,
            "count": 1
        },
        "models": {
            "ram": 0,
            "count": 0
        }
    }
},
"Z2RZNWRLSveVcz2c6lHf5A": {
    "name": "opensearch2",
    "hostname": "172.18.0.2",
    "stats": {
        ...
    }
}
}
```

Clear cache

Clears the plugin cache. Use this to refresh the model.

```
POST _ltr/_clearcache
```

Asynchronous search for Amazon OpenSearch Service

With asynchronous search for Amazon OpenSearch Service you can submit a search query that gets executed in the background, monitor the progress of the request, and retrieve results at a later stage. You can retrieve partial results as they become available before the search has completed. After the search finishes, save the results for later retrieval and analysis.

Asynchronous search requires OpenSearch 1.0 or later, or Elasticsearch 7.10 or later. Full documentation for asynchronous search, including detailed steps and API descriptions, is available in the [OpenSearch documentation](#).

Sample search call

To perform an asynchronous search, send HTTP requests to `_plugins/_asynchronous_search` using the following format:

```
POST opensearch-domain/_plugins/_asynchronous_search
```

Note

If you're using Elasticsearch 7.10 instead of an OpenSearch version, replace `_plugins` with `_opendistro` in all asynchronous search requests.

You can specify the following asynchronous search options:

Options	Description	Default value	Required
<code>wait_for_completion_timeout</code>	Specifies the amount of time that you plan to wait for the results. You can see whatever results you get within this time just like in a normal search. You can poll the remaining results based on an ID. The maximum value is 300 seconds.	1 second	No
<code>keep_on_completion</code>	Specifies whether you want to save the results in the cluster after the search is complete. You can examine the stored results at a later time.	false	No
<code>keep_alive</code>	Specifies the amount of time that the result is saved in the cluster. For example, <code>2d</code> means that the results are stored in the cluster for 48 hours. The saved search results are deleted after this period or if the search is canceled. Note that this includes the query runtime. If the query overruns this time, the process cancels this query automatically.	12 hours	No

Sample request

```
POST _plugins/_asynchronous_search/
pretty&size=10&wait_for_completion_timeout=1ms&keep_on_completion=true&request_cache=false
{
  "aggs": {
    "city": {
      "terms": {
        "field": "city",
        "size": 10
      }
    }
  }
}
```

Note

All request parameters that apply to a standard `_search` query are supported. If you're using Elasticsearch 7.10 instead of an OpenSearch version, replace `_plugins` with `_opendistro`.

Asynchronous search permissions

Asynchronous search supports [fine-grained access control \(p. 146\)](#). For details on mixing and matching permissions to fit your use case, see [Asynchronous search security](#).

For domains with fine-grained access control enabled, you need the following minimum permissions for a role:

```
# Allows users to use all asynchronous search functionality
asynchronous_search_full_access:
  reserved: true
```

```
cluster_permissions:
  - 'cluster:admin/opensearch/asynchronous-search/*'
index_permissions:
  - index_patterns:
    - '*'
  allowed_actions:
    - 'indices:data/read/search*'

# Allows users to read stored asynchronous search results
asynchronous_search_read_access:
  reserved: true
  cluster_permissions:
    - 'cluster:admin/opensearch/asynchronous-search/get'
```

For domains with fine-grained access control disabled, use your IAM access and secret key to sign all requests. You can access the results with the asynchronous search ID.

Asynchronous search settings

OpenSearch lets you change all available [asynchronous search settings](#) using the `_cluster/settings` API. In OpenSearch Service, you can only change the following settings:

- `opensearch.asynchronous_search.node_concurrent_running_searches`
- `opensearch.asynchronous_search.persist_search_failures`

Cross-cluster search

You can perform an asynchronous search across clusters with the following minor limitations:

- You can run an asynchronous search only on the source domain.
- You can't minimize network round trips as part of a cross-cluster search query.

If you set up a connection between domain-a → domain-b with connection alias `cluster_b` and domain-a → domain-c with connection alias `cluster_c`, asynchronously search domain-a, domain-b, and domain-c as follows:

```
POST https://src-domain.us-east-1.es.amazonaws.com/
local_index,cluster_b:b_index,cluster_c:c_index/_plugins/_asynchronous_search/?
pretty&size=10&wait_for_completion_timeout=500ms&keep_on_completion=true&request_cache=false
{
  "size": 0,
  "_source": {
    "excludes": []
  },
  "aggs": {
    "2": {
      "terms": {
        "field": "clientip",
        "size": 50,
        "order": {
          "_count": "desc"
        }
      }
    }
  },
  "stored_fields": [
    "*"
  ],
}
```

```
    "script_fields": {},
    "docvalue_fields": [
        "@timestamp"
    ],
    "query": {
        "bool": {
            "must": [
                {
                    "query_string": {
                        "query": "status:404",
                        "analyze_wildcard": true,
                        "default_field": "*"
                    }
                },
                {
                    "range": {
                        "@timestamp": {
                            "gte": 1483747200000,
                            "lte": 1488326400000,
                            "format": "epoch_millis"
                        }
                    }
                }
            ],
            "filter": [],
            "should": [],
            "must_not": []
        }
    }
}
```

Response

```
{
    "id" : "Fm9pYzJyVG91U19xb0hIQUJnMHJfRFEAAAAAAkngQ10WVBczNZQjVEa2dMYTBXaTdEagAAAAAAAAAB",
    "state" : "RUNNING",
    "start_time_in_millis" : 1609329314796,
    "expiration_time_in_millis" : 1609761314796
}
```

For more information, see [the section called “Cross-cluster search” \(p. 251\)](#).

UltraWarm

Asynchronous searches with UltraWarm indices continue to work. For more information, see [the section called “UltraWarm storage” \(p. 286\)](#).

Note

You can monitor asynchronous search statistics in CloudWatch. For a full list of metrics, see [the section called “Asynchronous search metrics” \(p. 86\)](#).

Using OpenSearch Dashboards with Amazon OpenSearch Service

OpenSearch Dashboards is an open-source visualization tool designed to work with OpenSearch. Amazon OpenSearch Service provides an installation of OpenSearch Dashboards with every OpenSearch Service domain. You can find a link to Dashboards on your domain dashboard on the OpenSearch Service console. The URL is [*domain-endpoint*/_dashboards/](#). Queries using this default OpenSearch Dashboards installation have a 300-second timeout.

The following sections address some common Dashboards use cases:

- the section called “Controlling access to OpenSearch Dashboards” (p. 280)
- the section called “Configuring OpenSearch Dashboards to use a WMS map server” (p. 282)
- the section called “Connecting a local Dashboards server to OpenSearch Service” (p. 283)

Controlling access to OpenSearch Dashboards

Dashboards does not natively support IAM users and roles, but OpenSearch Service offers several solutions for controlling access to Dashboards:

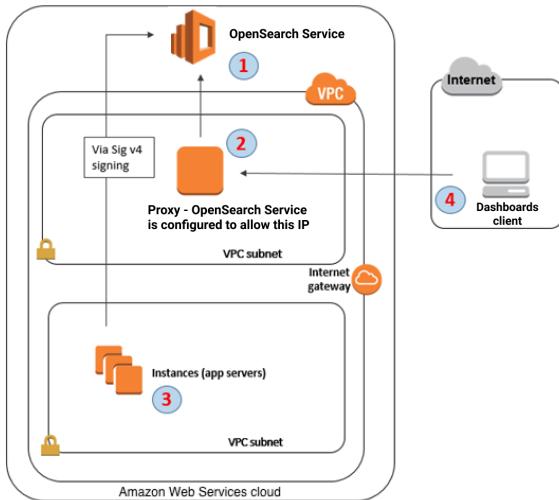
- Enable [SAML authentication for Dashboards](#) (p. 169).
- Use [fine-grained access control](#) (p. 149) with HTTP basic authentication.
- Configure [Cognito authentication for Dashboards](#) (p. 175).
- For public access domains, configure an [IP-based access policy](#) (p. 134) that either uses or does not use a [proxy server](#) (p. 280).
- For VPC access domains, use an open access policy that either uses or does not use a proxy server, and [security groups](#) to control access. To learn more, see [the section called “About access policies on VPC domains”](#) (p. 39).

Using a proxy to access OpenSearch Service from Dashboards

Note

This process is only applicable if your domain uses public access and you don't want to use [Cognito authentication](#) (p. 175). See [the section called “Controlling access to OpenSearch Dashboards”](#) (p. 280).

Because Dashboards 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 allow in order for each user to have access to Dashboards. One workaround is to place a proxy server between OpenSearch Dashboards and OpenSearch Service. 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 OpenSearch Service 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 OpenSearch Service.
4. OpenSearch Dashboards clients connect to your OpenSearch Service 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 [Cognito authentication \(p. 175\)](#), you might need to add settings for Dashboards and Amazon Cognito to avoid `redirect_mismatch` errors. See the following `nginx.conf` example:

```
server {  
    listen 443;  
    server_name $host;  
    rewrite ^$ https://$host/_plugin/_dashboards 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/_dashboards {  
        # Forward requests to Dashboards  
        proxy_pass https://$dashboards_host/_plugin/_dashboards;  
  
        # Handle redirects to Cognito  
        proxy_redirect https://$cognito_host https://$host;  
  
        # Update cookie domain and path  
        proxy_cookie_domain $dashboards_host $host;  
        proxy_cookie_path / _plugin/_dashboards/;  
  
        # 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 Dashboards  
        proxy_redirect https://$dashboards_host https://$host;  
  
        # Update cookie domain  
        proxy_cookie_domain $cognito_host $host;  
    }  
}
```

Configuring OpenSearch Dashboards to use a WMS map server

The default installation of OpenSearch Dashboards for OpenSearch Service includes a map service, except for domains in the India and China Regions. The map service supports up to 10 zoom levels.

Regardless of your Region, you can configure Dashboards to use a different Web Map Service (WMS) server for coordinate map visualizations. Region map visualizations only support the default map service.

To configure Dashboards to use a WMS map server:

1. Open Dashboards.
2. Choose **Stack Management**.
3. Choose **Advanced Settings**.
4. Locate **visualization:tileMap:WMSdefaults**.
5. Change **enabled** to **true** and **url** to the URL of a valid WMS map server:

```
{  
    "enabled": true,  
    "url": "wms-server-url",  
    "options": {  
        "format": "image/png",  
        "transparent": true  
    }  
}
```

6. Choose **Save changes**.

To apply the new default value to visualizations, you might need to reload Dashboards. If you have saved visualizations, choose **Options** after opening the visualization. Verify that **WMS map server** is enabled and **WMS url** contains your preferred map server, and then choose **Apply changes**.

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 Dashboards server to OpenSearch Service

If you already invested significant time into configuring your own OpenSearch Dashboards instance, you can use it instead of (or in addition to) the default Dashboards instance that OpenSearch Service provides. The following procedure works for domains that use [fine-grained access control \(p. 146\)](#) with an open access policy.

To connect a local OpenSearch Dashboards server to OpenSearch Service

1. On your OpenSearch Service domain, create a user with the appropriate permissions:
 - a. In Dashboards, go to **Security, Internal users**, and choose **Create internal user**.
 - b. Provide a username and password and choose **Create**.
 - c. Go to **Roles** and select a role.
 - d. Select **Mapped users** and choose **Manage mapping**.
 - e. In **Users**, add your username and choose **Map**.
2. Download and install the appropriate version of the OpenSearch [security plugin](#) on your self-managed Dashboards OSS installation.
3. On your local Dashboards server, open the config/opensearch_dashboards.yml file and add your OpenSearch Service endpoint with the username and password you created earlier:

```
opensearch.hosts: ['https://domain-endpoint']
```

```
opensearch.username: 'username'  
opensearch.password: 'password'
```

You can use the following sample `opensearch_dashboards.yml` file:

```
server.host: '0.0.0.0'  
  
opensearch.hosts: ['https://domain-endpoint']  
  
opensearch_dashboards.index: ".username"  
  
opensearch.ssl.verificationMode: none # if not using HTTPS  
  
opensearch_security.auth.type: basicauth  
opensearch_security.auth.anonymous_auth_enabled: false  
opensearch_security.cookie.secure: false # set to true when using HTTPS  
opensearch_security.cookie.ttl: 3600000  
opensearch_security.session.ttl: 3600000  
opensearch_security.session.keepalive: false  
opensearch_security.multitenancy.enabled: false  
opensearch_security.readonly_mode.roles: [opensearch_dashboards_read_only']  
opensearch_security.auth.unauthenticated_routes: []  
opensearch_security.basicauth.login.title: 'Please log in using your user name and  
password'  
  
opensearch.username: 'username'  
opensearch.password: 'password'  
opensearch.requestHeadersWhitelist:  
[  
  authorization,  
  securitytenant,  
  security_tenant,  
]
```

To see your OpenSearch Service indices, start your local Dashboards server, go to **Dev Tools** and run the following command:

```
GET _cat/indices
```

Managing indexes in OpenSearch Dashboards

The OpenSearch Dashboards installation on your OpenSearch Service domain provides a useful UI for managing indexes in different storage tiers on your domain. Choose **Index Management** from the Dashboards main menu to view all indexes in hot, [UltraWarm \(p. 286\)](#), and [cold \(p. 295\)](#) storage, as well as indexes managed by Index State Management (ISM) policies. Use index management to move indexes between warm and cold storage, and to monitor migrations between the three tiers.

Index Management

Rollup jobs

State management policies

Indices

- Hot Indices
- Warm Indices
- Cold Indices
- Policy managed indices

Cold indices

Cold storage lets you further reduce s



Search index name or stat



Index ↓



my-index-3



my-index-2



my-index-1

Note that you won't see the hot, warm, and cold index options unless you have UltraWarm and/or cold storage enabled.

Additional features

The default OpenSearch Dashboards installation on each OpenSearch Service domain has some additional features:

- User interfaces for the various [OpenSearch plugins \(p. 371\)](#)
- [Tenants \(p. 155\)](#)
- [Reports](#)

Use the **Reporting** menu to generate on-demand CSV reports from the Discover page and PDF or PNG reports of dashboards or visualizations. CSV reports have a 10,000 row limit.

- [Gantt charts](#)
- [Notebooks](#)

Managing indexes in Amazon OpenSearch Service

After you add data to Amazon OpenSearch 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, cold storage, and Index State Management. For information on the OpenSearch index APIs, see the [OpenSearch documentation](#).

Topics

- [UltraWarm storage for Amazon OpenSearch Service \(p. 286\)](#)
- [Cold storage for Amazon OpenSearch Service \(p. 295\)](#)
- [Index State Management in Amazon OpenSearch Service \(p. 305\)](#)
- [Summarizing indexes in Amazon OpenSearch Service with index rollups \(p. 313\)](#)
- [Transforming indexes in Amazon OpenSearch Service \(p. 314\)](#)
- [Cross-cluster replication for Amazon OpenSearch Service \(p. 315\)](#)
- [Migrating Amazon OpenSearch Service indexes using remote reindex \(p. 321\)](#)
- [Managing time-series data in Amazon OpenSearch Service with data streams \(p. 325\)](#)

UltraWarm storage for Amazon OpenSearch Service

UltraWarm provides a cost-effective way to store large amounts of read-only data on Amazon OpenSearch 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 indexes that you are not actively writing to, query less frequently, and don't need the same performance from, UltraWarm offers significantly lower costs per GiB of data. Because warm indexes are read-only unless you return them to hot storage, UltraWarm is best-suited to immutable data, such as logs.

In OpenSearch, warm indexes behave just like any other index. You can query them using the same APIs or use them to create visualizations in OpenSearch Dashboards.

Topics

- [Prerequisites \(p. 287\)](#)
- [UltraWarm storage requirements and performance considerations \(p. 288\)](#)
- [UltraWarm pricing \(p. 288\)](#)
- [Enabling UltraWarm \(p. 288\)](#)
- [Migrating indexes to UltraWarm storage \(p. 290\)](#)
- [Automating migrations \(p. 292\)](#)
- [Migration tuning \(p. 292\)](#)

- [Cancelled migrations \(p. 293\)](#)
- [Listing hot and warm indexes \(p. 293\)](#)
- [Returning warm indexes to hot storage \(p. 293\)](#)
- [Restoring warm indexes from automated snapshots \(p. 293\)](#)
- [Manual snapshots of warm indexes \(p. 294\)](#)
- [Migrating warm indexes to cold storage \(p. 295\)](#)
- [Disabling UltraWarm \(p. 295\)](#)

Prerequisites

UltraWarm has a few important prerequisites:

- UltraWarm requires OpenSearch or Elasticsearch 6.8 or higher.
- To use warm storage, domains must have [dedicated master nodes \(p. 358\)](#).
- If your domain uses a T2 or T3 instance type for your data nodes, you cannot use warm storage.
- If your index uses [approximate k-NN \("index.knn": true\)](#), you can't move it to warm storage.
- If the domain uses [fine-grained access control \(p. 146\)](#), users must be mapped to the `ultrawarm_manager` role in OpenSearch Dashboards to make UltraWarm API calls.

Note

The `ultrawarm_manager` role might not be defined on some preexisting OpenSearch Service domains. If you don't see the role in Dashboards, you need to [manually create it \(p. 287\)](#).

Configure permissions

If you enable UltraWarm on a preexisting OpenSearch Service domain, the `ultrawarm_manager` role might not be defined on the domain. Non-admin users must be mapped to this role in order to manage warm indexes on domains using fine-grained access control. To manually create the `ultrawarm_manager` role, perform the following steps:

1. In OpenSearch Dashboards, go to **Security** and choose **Permissions**.
2. Choose **Create action group** and configure the following groups:

Group name	Permissions
<code>ultrawarm_cluster</code>	<ul style="list-style-type: none">• <code>cluster:admin/ultrawarm/migration/list</code>• <code>cluster:monitor/nodes/stats</code>
<code>ultrawarm_index_read</code>	<ul style="list-style-type: none">• <code>indices:admin/ultrawarm/migration/get</code>• <code>indices:admin/get</code>
<code>ultrawarm_index_write</code>	<ul style="list-style-type: none">• <code>indices:admin/ultrawarm/migration/warm</code>• <code>indices:admin/ultrawarm/migration/hot</code>• <code>indices:monitor/stats</code>• <code>indices:admin/ultrawarm/migration/cancel</code>

3. Choose **Roles** and **Create role**.
4. Name the role `ultrawarm_manager`.
5. For **Cluster permissions**, select `ultrawarm_cluster` and `cluster_monitor`.
6. For **Index**, type *.

7. For **Index permissions**, select `ultrawarm_index_read`, `ultrawarm_index_write`, and `indices_monitor`.
8. Choose **Create**.
9. After you create the role, [map it \(p. 154\)](#) to any user or backend role that will manage UltraWarm indexes.

UltraWarm storage requirements and performance considerations

As covered in [the section called “Calculating storage requirements” \(p. 353\)](#), data in hot storage incurs significant overhead: replicas, Linux reserved space, and OpenSearch Service reserved space. For example, a 20 GiB primary shard with one replica shard requires roughly 58 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 20 GiB shard requires 20 GiB of warm storage. If you provision an `ultrawarm1.large.search` instance, you can use all 20 TiB of its maximum storage for primary shards. See [the section called “UltraWarm storage quotas” \(p. 398\)](#) for a summary of instance types and the maximum amount of storage that each can address.

With UltraWarm, we still recommend a maximum shard size of 50 GiB. The [number of CPU cores and amount of RAM allocated to each UltraWarm instance type \(p. 288\)](#) gives you an idea of the number of shards they can simultaneously search. Note that while only primary shards count toward UltraWarm storage in S3, OpenSearch Dashboards and `_cat/indices` still report UltraWarm index size as the *total* of all primary and replica shards.

For example, each `ultrawarm1.medium.search` instance has two CPU cores and can address up to 1.5 TiB of storage on S3. Two of these instances have a combined 3 TiB of storage, which works out to approximately 62 shards if each shard is 50 GiB. If a request to the cluster only searches four of these shards, performance might be excellent. If the request is broad and searches all 62 of them, the four CPU cores might struggle to perform the operation. Monitor the `WarmCPUUtilization` and `WarmJVMMemoryPressure` [UltraWarm metrics \(p. 81\)](#) to understand how the instances handle your workloads.

If your searches are broad or frequent, consider leaving the indexes in hot storage. Just like any other OpenSearch workload, the most important step to determining if UltraWarm meets your needs is to perform representative client testing using a realistic dataset.

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.search` 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 OpenSearch Service” \(p. 2\)](#).

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 \(p. 287\)](#). 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](#) to enable UltraWarm, specifically the `WarmEnabled`, `WarmCount`, and `WarmType` options in `ClusterConfig`.

Note

Domains support a maximum number of warm nodes. For details, see [the section called "Quotas" \(p. 397\)](#).

Sample CLI command

The following AWS CLI command creates a domain with three data nodes, three dedicated master nodes, six warm nodes, and fine-grained access control enabled:

```
aws opensearch create-domain \
--domain-name my-domain \
--engine-version Opensearch_1.0 \
--cluster-config \
  InstanceCount=3,InstanceType=r6g.large.search,DedicatedMasterEnabled=true,DedicatedMasterType=r6g.large \
  \
  --ebs-options EBSEnabled=true,VolumeType=gp2,VolumeSize=11 \
  --node-to-node-encryption-options Enabled=true \
  --encryption-at-rest-options Enabled=true \
  --domain-endpoint-options EnforceHTTPS=true,TLSecurityPolicy=Policy-Min-TLS-1-2-2019-07 \
  \
  --advanced-security-options \
    Enabled=true,InternalUserDatabaseEnabled=true,MasterUserOptions='{"MasterUserName":master-user, "MasterUserPassword":master-password}' \
    --access-policies '[{"Version": "2012-10-17", "Statement": [{"Effect": "Allow", "Principal": "AWS":["123456789012"], "Action": ["es:*"], "Resource": "arn:aws:es:us-west-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 fine-grained access control enabled and a restrictive access policy:

```
POST https://es.us-east-2.amazonaws.com/2021-01-01/opensearch/domain
{
  "ClusterConfig": {
    "InstanceCount": 3,
    "InstanceType": "r6g.large.search",
    "DedicatedMasterEnabled": true,
    "DedicatedMasterType": "r6g.large.search",
    "DedicatedMasterCount": 3,
    "ZoneAwarenessEnabled": true,
    "ZoneAwarenessConfig": {
      "AvailabilityZoneCount": 3
    },
    "WarmEnabled": true,
    "WarmCount": 6,
    "WarmType": "ultrawarm1.medium.search"
  },
  "EBSOptions": {
    "EBSEnabled": true,
```

```

        "VolumeType": "gp2",
        "VolumeSize": 11
    },
    "EncryptionAtRestOptions": {
        "Enabled": true
    },
    "NodeToNodeEncryptionOptions": {
        "Enabled": true
    },
    "DomainEndpointOptions": {
        "EnforceHTTPS": true,
        "TLSecurityPolicy": "Policy-Min-TLS-1-2-2019-07"
    },
    "AdvancedSecurityOptions": {
        "Enabled": true,
        "InternalUserDatabaseEnabled": true,
        "MasterUserOptions": {
            "MasterUserName": "master-user",
            "MasterUserPassword": "master-password"
        }
    },
    "EngineVersion": "Opensearch_1.0",
    "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 the [Amazon OpenSearch Service API Reference](#).

Migrating indexes to UltraWarm storage

If you finished 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
        }
    }
}

```

Index health must be green to perform a migration. If you migrate several indexes 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
```

OpenSearch Service migrates one index at a time to UltraWarm. You can have up to 200 migrations in the queue. Any request that exceeds the limit will be rejected. To check the current number of migrations in the queue, monitor the [HotToWarmMigrationQueueSize metric \(p. 81\)](#). Indexes remain available throughout the migration process—no downtime.

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 indexes:

```
GET my-index/_settings
{
  "my-index": {
    "settings": {
      "index": {
        "refresh_interval": "-1",
        "auto_expand_replicas": "false",
        "provided_name": "my-index",
        "creation_date": "1599241458998",
        "unassigned": {
          "node_left": {
            "delayed_timeout": "5m"
          }
        },
        "number_of_replicas": "1",
        "uuid": "GswyCdR0RSq0SJYmzsIpiw",
        "version": {
          "created": "7070099"
        },
        "routing": {
          "allocation": {
            "require": {
              "box_type": "warm"
            }
          }
        },
        "number_of_shards": "5",
        "merge": {
          "policy": {
            "max_merge_at_once_explicit": "50"
          }
        }
      }
    }
  }
}
```

```
        }
    }
}
}
```

- `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.
- `merge.policy.max_merge_at_once_explicit` specifies the number of segments to simultaneously merge during the migration.

Indexes in warm storage are read-only unless you [return them to hot storage \(p. 293\)](#), which makes UltraWarm best-suited to immutable data, such as logs. You can query the indexes and delete them, but you can't add, update, or delete individual documents. If you try, you might 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. 305) to automate the migration process after an index reaches a certain age or meets other conditions. See the [sample policy](#) (p. 306) that demonstrates this workflow.

Migration tuning

Index migrations to UltraWarm storage require a force merge. Each OpenSearch index is composed of some number of shards, and each shard is composed of some number of Lucene segments. The force merge operation purges documents that were marked for deletion and conserves disk space. By default, UltraWarm merges indexes into one segment.

You can change this value up to 1,000 segments using the `index.ultrawarm.migration.force_merge.max_num_segments` setting. Higher values speed up the migration process, but increase query latency for the warm index after the migration finishes. To change the setting, make the following request:

```
PUT my-index/_settings
{
  "index": {
    "ultrawarm": {
      "migration": {
        "force_merge": {
          "max_num_segments": 1
        }
      }
    }
}
```

```
    }  
}  
}
```

To check how long this stage of the migration process takes, monitor the `HotToWarmMigrationForceMergeLatency` metric ([p. 81](#)).

Cancelling migrations

UltraWarm handles migrations sequentially, in a queue. If a migration is in the queue, but has not yet started, you can remove it from the queue using the following request:

```
POST _ultrawarm/migration/_cancel/my-index
```

If your domain uses fine-grained access control, you must have the `indices:admin/ultrawarm/migration/cancel` permission to make this request.

Listing hot and warm indexes

UltraWarm adds two additional options, similar to `_all`, to help manage hot and warm indexes. For a list of all warm or hot indexes, make the following requests:

```
GET _warm  
GET _hot
```

You can use these options in other requests that specify indexes, such as:

```
_cat/indices/_warm  
_cluster/state/_all/_hot
```

Returning warm indexes 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. 81](#)).

After the migration finishes, check the index settings to make sure they meet your needs. Indexes return to hot storage with one replica.

Restoring warm indexes from automated snapshots

In addition to the standard repository for automated snapshots, UltraWarm adds a second repository for warm indexes, `cs-ultrawarm`. Each snapshot in this repository contains only one index. If you delete a warm index, its snapshot remains in the `cs-ultrawarm` repository for 14 days, just like any other automated snapshot.

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 you want to restore:

```
GET _snapshot/cs-ultrawarm/_all
{
  "snapshots": [
    {
      "snapshot": "snapshot-name",
      "version": "1.0",
      "indices": [
        "my-index"
      ]
    }
  ]
}
```

2. If the index already exists, delete it:

```
DELETE my-index
```

If you don't want to delete the index, return it to hot storage (p. 293) and reindex it.

3. Restore the snapshot:

```
POST _snapshot/cs-ultrawarm/snapshot-name/_restore
```

UltraWarm ignores any index settings you specify in this restore request, but you can specify options like `rename_pattern` and `rename_replacement`. For a summary of OpenSearch snapshot restore options, see the [OpenSearch documentation](#).

Manual snapshots of warm indexes

You *can* take manual snapshots of warm indexes, but we don't recommend it. The automated `cs-ultrawarm` repository already contains a snapshot for each warm index, taken during the migration, at no additional charge.

By default, OpenSearch Service does not include warm indexes in manual snapshots. For example, the following call only includes hot indexes:

```
PUT _snapshot/my-repository/my-snapshot
```

If you choose to take manual snapshots of warm indexes, several important considerations apply.

- You can't mix hot and warm indexes. For example, the following request fails:

```
PUT _snapshot/my-repository/my-snapshot
{
  "indices": "warm-index-1,hot-index-1",
  "include_global_state": false
}
```

If they include a mix of hot and warm indexes, wildcard (*) statements fail, as well.

- You can only include one warm index per snapshot. For example, the following request fails:

```
PUT _snapshot/my-repository/my-snapshot
{
```

```
    "indices": "warm-index-1,warm-index-2,other-warm-indices-\*",  
    "include_global_state": false  
}
```

This request succeeds:

```
PUT _snapshot/my-repository/my-snapshot  
{  
    "indices": "warm-index-1",  
    "include_global_state": false  
}
```

- Manual snapshots always restore to hot storage, even if they originally included a warm index.

Migrating warm indexes to cold storage

If you have data in UltraWarm that you query infrequently, consider migrating it to cold storage. Cold storage is meant for data you only access occasionally or is no longer in active use. You can't read from or write to cold indexes, but you can migrate them back to warm storage at no cost whenever you need to query them. For instructions, see [the section called "Migrating indexes to cold storage" \(p. 299\)](#).

Disabling UltraWarm

The console is the simplest way to disable UltraWarm. Choose the domain, **Actions**, and **Edit cluster configuration**. Deselect **Enable UltraWarm data nodes** and choose **Save changes**. You can also use the `WarmEnabled` option in the AWS CLI and configuration API.

Before you disable UltraWarm, you must either [delete](#) all warm indexes or [migrate them back to hot storage \(p. 293\)](#). After warm storage is empty, wait five minutes before attempting to disable UltraWarm.

Cold storage for Amazon OpenSearch Service

Cold storage lets you store any amount of infrequently accessed or historical data on your Amazon OpenSearch Service domain and analyze it on demand, at a lower cost than other storage tiers. Cold storage is appropriate if you need to do periodic research or forensic analysis on your older data. Practical examples of data suitable for cold storage include infrequently accessed logs, data that must be preserved to meet compliance requirements, or logs that have historical value.

Similar to [UltraWarm \(p. 286\)](#) storage, cold storage is backed by Amazon S3. When you need to query cold data, you can selectively attach it to existing UltraWarm nodes. You can manage the migration and lifecycle of your cold data manually or with Index State Management policies.

Topics

- [Prerequisites \(p. 296\)](#)
- [Cold storage requirements and performance considerations \(p. 297\)](#)
- [Cold storage pricing \(p. 297\)](#)
- [Enabling cold storage \(p. 297\)](#)
- [Managing cold indexes in OpenSearch Dashboards \(p. 298\)](#)
- [Migrating indexes to cold storage \(p. 299\)](#)
- [Automating migrations to cold storage \(p. 300\)](#)
- [Canceling migrations to cold storage \(p. 300\)](#)

- [Listing cold indices \(p. 300\)](#)
- [Migrating cold indexes to warm storage \(p. 303\)](#)
- [Restoring cold indexes from snapshots \(p. 304\)](#)
- [Canceling migrations from cold to warm storage \(p. 304\)](#)
- [Updating cold index metadata \(p. 304\)](#)
- [Deleting cold indices \(p. 305\)](#)
- [Disabling cold storage \(p. 305\)](#)

Prerequisites

Cold storage has the following prerequisites:

- Cold storage requires OpenSearch or Elasticsearch version 7.9 or later.
- To enable cold storage on an OpenSearch Service domain, you must also enable UltraWarm on the same domain.
- To use cold storage, domains must have [dedicated master nodes \(p. 358\)](#).
- If your domain uses a T2 or T3 instance type for your data nodes, you can't use cold storage.
- If your index uses [approximate k-NN \("index.knn": true\)](#), you can't move it to cold storage.
- If the domain uses [fine-grained access control \(p. 146\)](#), non-admin users must be [mapped \(p. 154\)](#) to the `cold_manager` role in OpenSearch Dashboards in order to manage cold indices.

Note

The `cold_manager` role might not exist on some preexisting OpenSearch Service domains. If you don't see the role in Dashboards, you need to [manually create it \(p. 296\)](#).

Configure permissions

If you enable cold storage on a preexisting OpenSearch Service domain, the `cold_manager` role might not be defined on the domain. If the domain uses [fine-grained access control \(p. 146\)](#), non-admin users must be mapped to this role in order to manage cold indices. To manually create the `cold_manager` role, perform the following steps:

1. In OpenSearch Dashboards, go to **Security** and choose **Permissions**.
2. Choose **Create action group** and configure the following groups:

Group name	Permissions
<code>cold_cluster</code>	<ul style="list-style-type: none">• <code>cluster:monitor/nodes/stats</code>• <code>cluster:admin/ultrawarm*</code>• <code>cluster:admin/cold/*</code>
<code>cold_index</code>	<ul style="list-style-type: none">• <code>indices:monitor/stats</code>• <code>indices:data/read/minmax</code>• <code>indices:admin/ultrawarm/migration/get</code>• <code>indices:admin/ultrawarm/migration/cancel</code>

3. Choose **Roles** and **Create role**.
4. Name the role `cold_manager`.
5. For **Cluster permissions**, choose the `cold_cluster` group you created.

6. For **Index**, enter *.
7. For **Index permissions**, choose the cold_index group you created.
8. Choose **Create**.
9. After you create the role, [map it \(p. 154\)](#) to any user or backend role that manages cold indices.

Cold storage requirements and performance considerations

Because cold storage uses Amazon S3, it incurs none of the overhead of hot storage, such as replicas, Linux reserved space, and OpenSearch Service reserved space. Cold storage doesn't have specific instance types because it doesn't have any compute capacity attached to it. You can store any amount of data in cold storage. Monitor the `ColdStorageSpaceUtilization` metric in Amazon CloudWatch to see how much cold storage space you're using.

Cold storage pricing

Similar to UltraWarm storage, with cold storage you only pay for data storage. There's no compute cost for cold data and you won't get billed if there's no data in cold storage.

You don't incur any transfer charges when moving data between cold and warm storage. While indexes are being migrated between warm and cold storage, you continue to pay for only one copy of the index. After the migration completes, the index is billed according to the storage tier it was migrated to. For more information about cold storage pricing, see [Amazon OpenSearch Service pricing](#).

Enabling cold storage

The console is the simplest way to create a domain that uses cold storage. While creating the domain, choose **Enable cold storage**. The same process works on existing domains as long as you meet the [prerequisites \(p. 296\)](#). Even after the domain state changes from **Processing** to **Active**, cold storage might not be available for several hours.

You can also use the [AWS CLI](#) or [configuration API](#) to enable cold storage.

Sample CLI command

The following AWS CLI command creates a domain with three data nodes, three dedicated master nodes, cold storage enabled, and fine-grained access control enabled:

```
aws opensearch create-domain \
--domain-name my-domain \
--engine-version Opensearch_1.0 \
--cluster-
config ColdStorageOptions={Enabled=true},WarmEnabled=true,WarmCount=4,WarmType=ultrawarm1.medium.search
 \
--ebs-options EBSEnabled=true,VolumeType=gp2,VolumeSize=11 \
--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-password}' \
--region us-east-2
```

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, cold storage enabled, and fine-grained access control enabled:

```
POST https://es.us-east-2.amazonaws.com/2021-01-01/opensearch/domain
{
    "ClusterConfig": {
        "InstanceCount": 3,
        "InstanceType": "r6g.large.search",
        "DedicatedMasterEnabled": true,
        "DedicatedMasterType": "r6g.large.search",
        "DedicatedMasterCount": 3,
        "ZoneAwarenessEnabled": true,
        "ZoneAwarenessConfig": {
            "AvailabilityZoneCount": 3
        },
        "WarmEnabled": true,
        "WarmCount": 4,
        "WarmType": "ultrawarm1.medium.search",
        "ColdStorageOptions": {
            "Enabled": true
        },
        "EBSOptions": {
            "EBSEnabled": true,
            "VolumeType": "gp2",
            "VolumeSize": 11
        },
        "EncryptionAtRestOptions": {
            "Enabled": true
        },
        "NodeToNodeEncryptionOptions": {
            "Enabled": true
        },
        "DomainEndpointOptions": {
            "EnforceHTTPS": true,
            "TLSecurityPolicy": "Policy-Min-TLS-1-2-2019-07"
        },
        "AdvancedSecurityOptions": {
            "Enabled": true,
            "InternalUserDatabaseEnabled": true,
            "MasterUserOptions": {
                "MasterUserName": "master-user",
                "MasterUserPassword": "master-password"
            }
        },
        "EngineVersion": "Opensearch_1.0",
        "DomainName": "my-domain"
    }
}
```

For detailed information, see the [Amazon OpenSearch Service API Reference](#).

Managing cold indexes in OpenSearch Dashboards

You can manage hot, warm and cold indexes with the existing Dashboards interface in your OpenSearch Service domain. Dashboards enables you to migrate indexes between warm and cold storage, and monitor index migration status, without using the CLI or configuration API. For more information, see [Managing indexes in OpenSearch Dashboards \(p. 284\)](#).

Migrating indexes to cold storage

When you migrate indexes to cold storage, you provide a time range for the data to make discovery easier. You can select a timestamp field based on the data in your index, manually provide a start and end timestamp, or choose to not specify one.

Parameter	Supported value	Description
timestamp_field	The date/time field from the index mapping.	The minimum and maximum values of the provided field are computed and stored as the <code>start_time</code> and <code>end_time</code> metadata for the cold index.
start_time and end_time	One of the following formats: <ul style="list-style-type: none"> strict_date_optional_time. For example: yyyy-MM-dd'T'HH:mm:ss.SSSZ or yyyy-MM-dd Epoch time in milliseconds 	The provided values are stored as the <code>start_time</code> and <code>end_time</code> metadata for the cold index.

If you don't want to specify a timestamp, add `?ignore=timestamp` to the request instead.

The following request migrates a warm index to cold storage and provides start and end times for the data in that index:

```
POST _ultrawarm/migration/my-index/_cold
{
  "start_time": "2020-03-09",
  "end_time": "2020-03-09T23:00:00Z"
}
```

Then check the status of the migration:

```
GET _ultrawarm/migration/my-index/_status
{
  "migration_status": {
    "index": "my-index",
    "state": "RUNNING_METADATA_RELOCATION",
    "migration_type": "WARM_TO_COLD"
  }
}
```

OpenSearch Service migrates one index at a time to cold storage. You can have up to 100 migrations in the queue. Any request that exceeds the limit will be rejected. To check the current number of migrations in the queue, monitor the [WarmToColdMigrationQueueSize metric \(p. 84\)](#). The migration process has the following states:

ACCEPTED_COLD_MIGRATION - Migration request is accepted and queued.
RUNNING_METADATA_MIGRATION - The migration request was selected for execution and metadata is migrating to cold storage.
FAILED_METADATA_MIGRATION - The attempt to add index metadata has failed and all retries are exhausted.
PENDING_INDEX_DETACH - Index metadata migration to cold storage is completed. Preparing to detach the warm index state from the local cluster.

RUNNING_INDEX_DETACH - Local warm index state from the cluster is being removed. Upon success, the migration request will be completed.
FAILED_INDEX_DETACH - The index detach process failed and all retries are exhausted.

Automating migrations to cold storage

You can use [Index State Management \(p. 305\)](#) to automate the migration process after an index reaches a certain age or meets other conditions. See the [sample policy \(p. 306\)](#), which demonstrates how to automatically migrate indexes from hot to UltraWarm to cold storage.

Note

An explicit `timestamp_field` is required in order to move indexes to cold storage using an Index State Management policy.

Canceling migrations to cold storage

If a migration to cold storage is queued or in a failed state, you can cancel the migration using the following request:

```
POST _ultrawarm/migration/_cancel/my-index
{
  "acknowledged" : true
}
```

If your domain uses fine-grained access control, you need the `indices:admin/ultrawarm/migration/cancel` permission to make this request.

Listing cold indices

Before querying, you can list the indexes in cold storage to decide which ones to migrate to UltraWarm for further analysis. The following request lists all cold indices, sorted by index name:

```
GET _cold/indices/_search
```

Sample response

```
{
  "pagination_id" : "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY",
  "total_results" : 3,
  "indices" : [
    {
      "index" : "my-index-1",
      "index_cold_uuid" : "hjEoh26mRRCFxRIMdgvLmg",
      "size" : 10339,
      "creation_date" : "2021-06-28T20:23:31.206Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    },
    {
      "index" : "my-index-2",
      "index_cold_uuid" : "0vIS2n-oROm0WDFmwFIgdw",
      "size" : 6068,
      "creation_date" : "2021-07-15T19:41:18.046Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    },
    {
      "index" : "my-index-3",
      "index_cold_uuid" : "12345678-9abc-def0-4321-3456789abcdef0",
      "size" : 12345,
      "creation_date" : "2021-08-20T14:55:12.123Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    }
  ]
}
```

```
        "index" : "my-index-3",
        "index_cold_uuid" : "EaeXOBodTLiDYcivKsXVLQ",
        "size" : 32403,
        "creation_date" : "2021-07-08T00:12:01.523Z",
        "start_time" : "2020-03-09T00:00Z",
        "end_time" : "2020-03-09T23:00Z"
    }
]
}
```

Filtering

You can filter cold indexes based on a prefix-based index pattern and time range offsets.

The following request lists indexes that match the prefix pattern of event-*:

```
GET _cold/_indices/_search
{
  "filters": {
    "index_pattern": "event-*"
  }
}
```

Sample response

```
{
  "pagination_id" : "je7MtGbClwBF/2Zp9Utk/h3yCo8nzbEXAMPLEKEY",
  "total_results" : 1,
  "indices" : [
    {
      "index" : "events-index",
      "index_cold_uuid" : "4eFiab7rRfSvp3sIrlsIKA",
      "size" : 32263273,
      "creation_date" : "2021-08-18T18:25:31.845Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    }
  ]
}
```

The following request returns indexes with `start_time` and `end_time` metadata fields between 2019-03-01 and 2020-03-01:

```
GET _cold/_indices/_search
{
  "filters": {
    "time_range": {
      "start_time": "2019-03-01",
      "end_time": "2020-03-01"
    }
  }
}
```

Sample response

```
{
  "pagination_id" : "je7MtGbClwBF/2Zp9Utk/h3yCo8nzbEXAMPLEKEY",
  "total_results" : 1,
  "indices" : [
    {

```

```
        "index" : "my-index",
        "index_cold_uuid" : "4eFiab7rRfSvp3slrIsIKA",
        "size" : 32263273,
        "creation_date" : "2021-08-18T18:25:31.845Z",
        "start_time" : "2019-05-09T00:00Z",
        "end_time" : "2019-09-09T23:00Z"
    }
]
}
```

Sorting

You can sort cold indexes by metadata fields such as index name or size. The following request lists all indexes sorted by size in descending order:

```
GET _cold/indices/_search
{
  "sort_key": "size:desc"
}
```

Sample response

```
{
  "pagination_id" : "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY",
  "total_results" : 5,
  "indices" : [
    {
      "index" : "my-index-6",
      "index_cold_uuid" : "4eFiab7rRfSvp3slrIsIKA",
      "size" : 32263273,
      "creation_date" : "2021-08-18T18:25:31.845Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    },
    {
      "index" : "my-index-9",
      "index_cold_uuid" : "mbD3ZRVDR16ONqgE0sJyUA",
      "size" : 57922,
      "creation_date" : "2021-07-07T23:41:35.640Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    },
    {
      "index" : "my-index-5",
      "index_cold_uuid" : "EaeXOBodTLiDYcivKsXVLQ",
      "size" : 32403,
      "creation_date" : "2021-07-08T00:12:01.523Z",
      "start_time" : "2020-03-09T00:00Z",
      "end_time" : "2020-03-09T23:00Z"
    }
  ]
}
```

Other valid sort keys are `start_time:asc/desc`, `end_time:asc/desc`, and `index_name:asc/desc`.

Pagination

You can paginate a list of cold indices. Configure the number of indexes to be returned per page with the `page_size` parameter (default is 10). Every `_search` request on your cold indexes returns a `pagination_id` which you can use for subsequent calls.

The following request paginates the results of a `_search` request of your cold indexes and displays the next 100 results:

```
GET _cold/indices/_search?page_size=100
{
  "pagination_id": "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY"
}
```

Migrating cold indexes to warm storage

After you narrow down your list of cold indexes with the filtering criteria in the previous section, migrate them back to UltraWarm where you can query the data and use it to create visualizations.

The following request migrates two cold indexes back to warm storage:

```
POST _cold/migration/_warm
{
  "indices": "my-index1,my-index2"
}

{
  "acknowledged" : true
}
```

To check the status of the migration and retrieve the migration ID, send the following request:

```
GET _cold/migration/_status
```

Sample response

```
{
  "cold_to_warm_migration_status" : [
    {
      "migration_id" : "tyLjXCA-S76zPQbPVHkOKA",
      "indices" : [
        "my-index1,my-index2"
      ],
      "state" : "RUNNING_INDEX_CREATION"
    }
  ]
}
```

To get index-specific migration information, include the index name:

```
GET _cold/migration/my-index/_status
```

Rather than specifying an index, you can list the indexes by their current migration status. Valid values are `_failed`, `_accepted`, and `_all`.

The following command gets the status of all indexes in a single migration request:

```
GET _cold/migration/_status?migration_id=my-migration-id
```

Retrieve the migration ID using the status request. For detailed migration information, add `&verbose=true`.

You can migrate indexes from cold to warm storage in batches of 10 or less, with a maximum of 100 indexes being migrated simultaneously. Any request that exceeds the limit will be rejected. To check the current number of migrations currently taking place, monitor the [ColdToWarmMigrationQueueSize metric \(p. 84\)](#). The migration process has the following states:

```
ACCEPTED_MIGRATION_REQUEST - Migration request is accepted and queued.  
RUNNING_INDEX_CREATION - Migration request is picked up for processing and will create warm  
indexes in the cluster.  
PENDING_COLD_METADATA_CLEANUP - Warm index is created and the migration service will  
attempt to clean up cold metadata.  
RUNNING_COLD_METADATA_CLEANUP - Cleaning up cold metadata from the indexes migrated to warm  
storage.  
FAILED_COLD_METADATA_CLEANUP - Failed to clean up metadata in the cold tier.  
FAILED_INDEX_CREATION - Failed to create an index in the warm tier.
```

Restoring cold indexes from snapshots

Contact [AWS Support](#) if you need to restore cold indexes from an automated snapshot, including in situations where an entire domain was accidentally deleted. OpenSearch Service retains cold indexes for 14 days after they've been deleted.

Cancelling migrations from cold to warm storage

If an index migration from cold to warm storage is queued or in a failed state, you can cancel it with the following request:

```
POST _cold/migration/my-index/_cancel  
{  
    "acknowledged" : true  
}
```

To cancel migration for a batch of indexes (maximum of 10 at a time), specify the migration ID:

```
POST _cold/migration/_cancel?migration_id=my-migration-id  
{  
    "acknowledged" : true  
}
```

Retrieve the migration ID using the status request.

Updating cold index metadata

You can update the `start_time` and `end_time` fields for a cold index:

```
PATCH _cold/my-index  
{  
    "start_time": "2020-01-01",  
    "end_time": "2020-02-01"  
}
```

You can't update the `timestamp_field` of an index in cold storage.

Note

OpenSearch Dashboards doesn't support the PATCH method. Use [curl](#), [Postman](#), or some other method to update cold metadata.

Deleting cold indices

If you're not using an ISM policy you can delete cold indexes manually. The following request deletes a cold index:

```
DELETE _cold/my-index
{
  "acknowledged" : true
}
```

Disabling cold storage

The OpenSearch Service console is the simplest way to disable cold storage. Select the domain and choose **Actions, Edit cluster configuration**, then deselect **Enable cold storage**.

To use the AWS CLI or configuration API, under `ColdStorageOptions`, set `"Enabled"="false"`.

Before you disable cold storage, you must either delete all cold indexes or migrate them back to warm storage, otherwise the disable action fails.

Index State Management in Amazon OpenSearch Service

Index State Management (ISM) in Amazon OpenSearch Service lets you define custom management policies that automate routine tasks, and apply them to indexes 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 indexes 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.

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 indexes at the same time. ISM doesn't run jobs if the cluster state is red.

ISM requires OpenSearch or Elasticsearch 6.8 or later. Full documentation is available in the [OpenSearch documentation](#).

Important

The `policy_id` setting for index templates is deprecated. You can no longer use index templates to apply ISM policies to newly created indexes. You can continue to automatically manage newly created indexes with the [ISM template field](#). This update introduces a breaking change that affects existing CloudFormation templates using this setting.

Create an ISM policy

To get started with Index State Management

1. Open the Amazon OpenSearch Service console at <https://console.aws.amazon.com/esv3/>.
2. Select the domain that you want to create an ISM policy for.

3. From the domain's dashboard, navigate to the OpenSearch Dashboards URL and sign in with your master user name and password. The URL follows this format:

```
domain-endpoint/_dashboards/
```

4. Open the left navigation panel within OpenSearch Dashboards and choose **Index Management**, then **Create policy**.
5. Use the [visual editor](#) or [JSON editor](#) to create policies. We recommend using the visual editor as it offers a more structured way of defining policies. For help creating policies, see the [sample policies \(p. 306\)](#) below.
6. After you create a policy, attach it to one or more indexes:

```
POST _plugins/_ism/add/my-index
{
  "policy_id": "my-policy-id"
}
```

Note

If your domain is running a legacy Elasticsearch version, use `_opendistro` instead of `_plugins`.

Alternatively, select the index in OpenSearch Dashboards and choose **Apply policy**.

Sample policies

The following sample policies demonstrate how to automate common ISM use cases.

Hot to warm to cold storage

This sample policy moves an index from hot storage to [UltraWarm \(p. 286\)](#), and eventually to [cold storage \(p. 295\)](#), then deletes the index.

The index is initially in the hot state. After ten days, ISM moves it to the warm state. 80 days later, after the index is 90 days old, ISM moves the index to the cold state. After a year, the service sends a notification to an Amazon Chime room that the index is being deleted and then permanently deletes it.

Note that cold indexes require the `cold_delete` operation rather than the normal `delete` operation. Also note that an explicit `timestamp_field` is required in your data in order to manage cold indexes with ISM.

```
{
  "policy": {
    "description": "Demonstrate a hot-warm-cold-delete workflow.",
    "default_state": "hot",
    "schema_version": 1,
    "states": [
      {
        "name": "hot",
        "actions": [],
        "transitions": [
          {
            "state_name": "warm",
            "conditions": {
              "min_index_age": "10d"
            }
          }
        ]
      },
      {
        "name": "warm",
        "actions": [
          {
            "action": "move",
            "target": "ultrawarm"
          }
        ],
        "transitions": [
          {
            "state_name": "cold",
            "conditions": {
              "min_index_age": "90d"
            }
          }
        ]
      },
      {
        "name": "cold",
        "actions": [
          {
            "action": "cold_delete"
          }
        ],
        "transitions": [
          {
            "state_name": "deleted"
          }
        ]
      },
      {
        "name": "deleted",
        "actions": []
      }
    ],
    "actions": [
      {
        "action": "move",
        "target": "ultrawarm"
      }
    ],
    "conditions": [
      {
        "min_index_age": "1y"
      }
    ]
  }
}
```

```

        "name": "warm",
        "actions": [
            "warm_migration": {},
            "retry": {
                "count": 5,
                "delay": "1h"
            }
        ],
        "transitions": [
            "state_name": "cold",
            "conditions": {
                "min_index_age": "90d"
            }
        ]
    },
    {
        "name": "cold",
        "actions": [
            "cold_migration": {
                "timestamp_field": "<your timestamp field>"
            }
        ],
        "transitions": [
            "state_name": "delete",
            "conditions": {
                "min_index_age": "365d"
            }
        ]
    },
    {
        "name": "delete",
        "actions": [
            "notification": {
                "destination": {
                    "chime": {
                        "url": "<URL>"
                    }
                },
                "message_template": {
                    "source": "The index {{ctx.index}} is being deleted."
                }
            },
            {
                "cold_delete": {}
            }
        ]
    }
}

```

Reduce replica count

This 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": []
                }
            ]
        }
    ]
}

```

Take an index snapshot

This sample policy uses the [snapshot](#) operation to take a snapshot of an index as soon as it contains at least one document. `repository` is the name of the manual snapshot repository you registered in Amazon S3. `snapshot` is the name of the snapshot. For snapshot prerequisites and steps to register a repository, see [the section called “Creating index snapshots” \(p. 42\)](#).

```

{
    "policy": {
        "description": "Takes an index snapshot.",
        "schema_version": 1,
        "default_state": "empty",
        "states": [
            {
                "name": "empty",
                "actions": [],
                "transitions": [
                    {
                        "state_name": "occupied",
                        "conditions": {
                            "min_doc_count": 1
                        }
                    }
                ]
            },
            {
                "name": "occupied",
                "actions": [
                    {
                        "snapshot": {
                            "repository": "<my-repository>",
                            "snapshot": "<my-snapshot>"
                        }
                    }
                ]
            }
        ]
    }
}

```

```
        }],
        "transitions": []
    }
}
}
```

ISM templates

You can set up an `ism_template` field in a policy so when you create an index that matches the template pattern, the policy is automatically attached to that index. In this example, any index you create with a name that begins with "log" is automatically matched to the ISM policy `my-policy-id`:

```
PUT _plugins/_ism/policies/my-policy-id
{
  "policy": {
    "description": "Example policy.",
    "default_state": "...",
    "states": [...],
    "ism_template": {
      "index_patterns": ["log*"],
      "priority": 100
    }
  }
}
```

For a more detailed example, see [Sample policy with ISM template for auto rollover](#).

Differences

Compared to OpenSearch and Elasticsearch, ISM for Amazon OpenSearch Service has several differences.

ISM operations

- OpenSearch Service supports three unique ISM operations, `warm_migration`, `cold_migration`, and `cold_delete`:
 - If your domain has [UltraWarm \(p. 286\)](#) enabled, the `warm_migration` action transitions the index to warm storage.
 - If your domain has [cold storage \(p. 295\)](#) enabled, the `cold_migration` action transitions the index to cold storage, and the `cold_delete` action deletes the index from cold storage.

Even if one of these actions doesn't complete within the [set timeout period](#), the migration or deletion of indexes still continues. Setting an [error_notification](#) for one of the above actions will notify you that the action failed if it didn't complete within the timeout period, but the notification is only for your own reference. The actual operation has no inherent timeout and continues to run until it eventually succeeds or fails.

- If your domain runs OpenSearch or Elasticsearch 7.4 or later, OpenSearch Service supports the ISM `open` and `close` operations.
- If your domain runs OpenSearch or Elasticsearch 7.7 or later, OpenSearch Service supports the ISM `snapshot` operation.

Cold storage ISM operations

For cold indexes, you must specify a `?type=_cold` parameter when you use the following ISM APIs:

- [Add policy](#)
- [Remove policy](#)
- [Update policy](#)
- [Retry failed index](#)
- [Explain index](#)

These APIs for cold indexes have the following additional differences:

- Wildcard operators are not supported except when you use it at the end. For example, `_plugins/_ism/<add, remove, change_policy, retry, explain>/logstash-*` is supported but `_plugins/_ism/<add, remove, change_policy, retry, explain>/iad-*>-prod` isn't supported.
- Multiple index names and patterns are not supported. For example, `_plugins/_ism/<add, remove, change_policy, retry, explain>/app-logs` is supported but `_plugins/_ism/<add, remove, change_policy, retry, explain>/app-logs,sample-data` isn't supported.

ISM settings

OpenSearch and Elasticsearch let you change all available ISM settings using the `_cluster/settings` API. On Amazon OpenSearch Service, you can only change the following settings:

- **Cluster-level settings:**
 - `enabled`
 - `history.enabled`
- **Index-level settings:**
 - `rollover_alias`

Tutorial: Automating Index State Management processes

This tutorial demonstrates how to implement an ISM policy that automates routine index management tasks and apply them to indexes and index patterns.

[Index State Management \(ISM\) \(p. 305\)](#) in Amazon OpenSearch Service lets you automate recurring index management activities, so you can avoid using additional tools to manage index lifecycles. You can create a policy that automates these operations based on index age, size, and other conditions, all from within your Amazon OpenSearch Service domain.

OpenSearch Service supports three storage tiers: the default "hot" state for active writing and low-latency analytics, UltraWarm for read-only data up to three petabytes, and cold storage for unlimited long-term archival.

This tutorial presents a sample use case of handling time-series data in daily indexes. In this tutorial, you set up a policy that takes an automated snapshot of each attached index after 24 hours. It then migrates the index from the default hot state to UltraWarm storage after two days, cold storage after 30 days, and finally deletes the index after 60 days.

Prerequisites

- Your OpenSearch Service domain must be running Elasticsearch version 6.8 or later.

- Your domain must have [UltraWarm \(p. 286\)](#) and [cold storage \(p. 295\)](#) enabled.
- You must [register a manual snapshot repository \(p. 45\)](#) for your domain.
- Your user role needs sufficient permissions to access the OpenSearch Service console. If necessary, validate and [configure access to your domain \(p. 130\)](#).

Step 1: Configure the ISM policy

First, configure an ISM policy in OpenSearch Dashboards.

1. From your domain dashboard in the OpenSearch Service console, navigate to the OpenSearch Dashboards URL and sign in with your master user name and password. The URL follows this format: [*domain-endpoint*/_dashboards/](#).
2. In OpenSearch Dashboards, choose **Add sample data** and add one or more of the sample indexes to your domain.
3. Open the left navigation panel and choose **Index Management**, then choose **Create policy**.
4. Name the policy `ism-policy-example`.
5. Replace the default policy with the following policy:

```
{  
  "policy": {  
    "description": "Move indexes between storage tiers",  
    "default_state": "hot",  
    "states": [  
      {  
        "name": "hot",  
        "actions": [],  
        "transitions": [  
          {  
            "state_name": "snapshot",  
            "conditions": {  
              "min_index_age": "24h"  
            }  
          }  
        ]  
      },  
      {  
        "name": "snapshot",  
        "actions": [  
          {  
            "retry": {  
              "count": 5,  
              "backoff": "exponential",  
              "delay": "30m"  
            },  
            "snapshot": {  
              "repository": "snapshot-repo",  
              "snapshot": "ism-snapshot"  
            }  
          }  
        ],  
        "transitions": [  
          {  
            "state_name": "warm",  
            "conditions": {  
              "min_index_age": "2d"  
            }  
          }  
        ]  
      },  
      {  
        "name": "warm",  
        "actions": [  
          {  
            "retry": {  
              "count": 5,  
              "backoff": "exponential",  
              "delay": "30m"  
            },  
            "snapshot": {  
              "repository": "warm-repo",  
              "snapshot": "ism-warm"  
            }  
          }  
        ],  
        "transitions": [  
          {  
            "state_name": "cold",  
            "conditions": {  
              "min_index_age": "14d"  
            }  
          }  
        ]  
      },  
      {  
        "name": "cold",  
        "actions": [  
          {  
            "retry": {  
              "count": 5,  
              "backoff": "exponential",  
              "delay": "30m"  
            },  
            "snapshot": {  
              "repository": "cold-repo",  
              "snapshot": "ism-cold"  
            }  
          }  
        ],  
        "transitions": [  
          {  
            "state_name": "ultra_warm",  
            "conditions": {  
              "min_index_age": "30d"  
            }  
          }  
        ]  
      },  
      {  
        "name": "ultra_warm",  
        "actions": [  
          {  
            "retry": {  
              "count": 5,  
              "backoff": "exponential",  
              "delay": "30m"  
            },  
            "snapshot": {  
              "repository": "ultra\_warm-repo",  
              "snapshot": "ism-ultra\_warm"  
            }  
          }  
        ],  
        "transitions": [  
          {  
            "state_name": "hot",  
            "conditions": {  
              "min_index_age": "24h"  
            }  
          }  
        ]  
      }  
    ]  
  }  
}
```

```
        "name": "warm",
        "actions": [
            {
                "retry": {
                    "count": 5,
                    "backoff": "exponential",
                    "delay": "1h"
                },
                "warm_migration": {}
            }
        ],
        "transitions": [
            {
                "state_name": "cold",
                "conditions": {
                    "min_index_age": "30d"
                }
            }
        ]
    },
    {
        "name": "cold",
        "actions": [
            {
                "retry": {
                    "count": 5,
                    "backoff": "exponential",
                    "delay": "1h"
                },
                "cold_migration": {
                    "start_time": null,
                    "end_time": null,
                    "timestamp_field": "@timestamp",
                    "ignore": "none"
                }
            }
        ],
        "transitions": [
            {
                "state_name": "delete",
                "conditions": {
                    "min_index_age": "60d"
                }
            }
        ]
    },
    {
        "name": "delete",
        "actions": [
            {
                "cold_delete": {}
            }
        ],
        "transitions": []
    }
],
"ism_template": [
    {
        "index_patterns": [
            "index-*"
        ],
        "priority": 100
    }
]
}
```

Note

The `ism_template` field automatically attaches the policy to any newly created index that matches one of the specified `index_patterns`. In this case, all indexes that start with `index-`. You can modify this field to match an index format in your environment. For more information, see [ISM templates \(p. 309\)](#).

6. In the snapshot section of the policy, replace `snapshot-repo` with the name of the [snapshot repository \(p. 45\)](#) that you registered for your domain. You can also optionally replace `ism-snapshot`, which will be the name of snapshot when it's created.
7. Choose **Create**. The policy is now visible on the **State management policies** page.

Step 2: Attach the policy to one or more indexes

Now that you created your policy, attach it to one or more indexes in your cluster.

1. Go to the **Hot indices** tab and search for `opensearch_dashboards_sample`, which lists all of the sample indexes that you added in step 1.
2. Select all of the indexes and choose **Apply policy**, then choose the `ism-policy-example` policy that you just created.
3. Choose **Apply**.

You can monitor the indexes as they move through the various states on the **Policy managed indices** page.

Summarizing indexes in Amazon OpenSearch Service with index rollups

Index rollups in Amazon OpenSearch Service let you reduce storage costs by periodically rolling up old data into summarized indices.

You pick the fields that interest you and use an index rollup to create a new index with only those fields aggregated into coarser time buckets. You can store months or years of historical data at a fraction of the cost with the same query performance.

Index rollups requires OpenSearch or Elasticsearch 7.9 or later. Full documentation for the feature is available in the [OpenSearch documentation](#).

Creating an index rollup job

To get started, choose **Index Management** in OpenSearch Dashboards. Select **Rollup Jobs** and choose **Create rollup job**.

Step 1: Set up indices

Set up the source and target indices. The source index is the one that you want to roll up. The target index is where the index rollup results are saved.

After you create an index rollup job, you can't change your index selections.

Step 2: Define aggregations and metrics

Select the attributes with the aggregations (terms and histograms) and metrics (avg, sum, max, min, and value count) that you want to roll up. Make sure you don't add a lot of highly granular attributes, because you won't save much space.

Step 3: Specify schedules

Specify a schedule to roll up your indexes as it's being ingested. The index rollup job is enabled by default.

Step 4: Review and create

Review your configuration and select **Create**.

Step 5: Search the target index

You can use the standard `_search` API to search the target index. You can't access the internal structure of the data in the target index because the plugin automatically rewrites the query in the background to suit the target index. This is to make sure you can use the same query for the source and target index.

To query the target index, set size to 0:

```
GET target_index/_search
{
  "size": 0,
  "query": {
    "match_all": {}
  },
  "aggs": {
    "avg_cpu": {
      "avg": {
        "field": "cpu_usage"
      }
    }
  }
}
```

Transforming indexes in Amazon OpenSearch Service

Whereas [index rollup jobs \(p. 313\)](#) let you reduce data granularity by rolling up old data into condensed indices, transform jobs let you create a different, summarized view of your data centered around certain fields, so you can visualize or analyze the data in different ways.

Index transforms have an OpenSearch Dashboards user interface and REST API. The feature requires OpenSearch 1.0 or later. Full documentation is available in the [OpenSearch documentation](#).

Creating an index transform job

If you don't have any data in your cluster, use the sample flight data within OpenSearch Dashboards to try out transform jobs. After adding the data, launch OpenSearch Dashboards. Then choose **Index Management**, **Transform Jobs**, and **Create Transform Job**.

Step 1: Choose indices

In the **Indices** section, select the source and target index. You can either select an existing target index or create a new one by entering a name for it.

If you want to transform just a subset of your source index, choose **Add Data Filter**, and use the OpenSearch [query DSL](#) to specify a subset of your source index.

Step 2: Choose fields

After choosing your indices, choose the fields you want to use in your transform job, as well as whether to use groupings or aggregations.

- You can use groupings to place your data into separate buckets in your transformed index. For example, if you want to group all of the airport destinations within the sample flight data, group the DestAirportID field into a target field of DestAirportID_terms field, and you can find the grouped airport IDs in your transformed index after the transform job finishes.
- On the other hand, aggregations let you perform simple calculations. For example, you might include an aggregation in your transform job to define a new field of sum_of_total_ticket_price that calculates the sum of all airplane tickets. Then you can analyze the new data in your transformed index.

Step 3: Specify a schedule

Transform jobs are enabled by default and run on schedules. For **transform execution interval**, specify an interval in minutes, hours, or days.

Step 4: Review and monitor

Review your configuration and select **Create**. Then monitor the **Transform job status** column.

Step 5: Search the target index

After the job finishes, you can use the standard _search API to search the target index.

For example, after running a transform job that transforms the flight data based on the DestAirportID field, you can run the following request to return all fields that have a value of SFO:

```
GET target_index/_search
{
  "query": {
    "match": {
      "DestAirportID_terms" : "SFO"
    }
  }
}
```

Cross-cluster replication for Amazon OpenSearch Service

With cross-cluster replication in Amazon OpenSearch Service, you can replicate indexes, mappings, and metadata from one OpenSearch Service domain to another. It follows an active-passive replication

model where the follower index (where the data is replicated) pulls data from the leader index. Using cross-cluster replication helps to ensure disaster recovery if there is an outage, and allows you to replicate data across geographically distant data centers to reduce latency. You pay [standard AWS data transfer charges](#) for the data transferred between domains.

Cross-cluster replication is available on domains running Elasticsearch 7.10 or OpenSearch 1.1 or later. Full documentation for cross-cluster replication is available in the [OpenSearch documentation](#).

Limitations

Cross-cluster replication has the following limitations:

- You can't replicate data between Amazon OpenSearch Service domains and self-managed OpenSearch or Elasticsearch clusters.
- A domain can be connected, through a combination of inbound and outbound connections, to a maximum of 20 other domains.
- Domains must either share the same major version, or be on the final minor version and the next major version.
- You can't use AWS CloudFormation to connect domains.
- You can't use cross-cluster replication on M3 or burstable (T2 and T3) instances.
- You can't replicate data between UltraWarm or cold indexes. Both indexes must be in hot storage.

Prerequisites

Before you set up cross-cluster replication, make sure that your domains meet the following requirements:

- Elasticsearch 7.10 or OpenSearch 1.1 or later
- [Fine-grained access control \(p. 146\)](#) enabled
- [Node-to-node encryption \(p. 129\)](#) enabled

Permissions requirements

In order to start replication, you must include the `es:ESCrossClusterGet` permission on the remote (leader) domain. We recommend the following IAM policy on the remote domain (which also lets you perform other operations, such as indexing documents and performing standard searches):

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

```
        "AWS": "*"
    },
    "Action": "es:ESCrossClusterGet",
    "Resource": "arn:aws:es:region:account:domain/leader-domain"
}
]
```

Make sure that the `es:ESCrossClusterGet` permission is applied for `/leader-domain` and not `/leader-domain/*`.

In order for non-admin users to perform replication activities, they also need to be mapped to the appropriate permissions. Most permissions correspond to specific [REST API operations](#). For example, the `indices:admin/plugins/replication/index/_resume` permission lets you resume replication of an index. For a full list of permissions, see [Replication permissions](#) in the OpenSearch documentation.

Note

The commands to start replication and create a replication rule are special cases. Because they invoke background processes on the leader and follower domains, you must pass a `leader_cluster_role` and `follower_cluster_role` in the request. OpenSearch Service uses these roles in all backend replication tasks. For information about mapping and using these roles, see [Map the leader and follower cluster roles](#) in the OpenSearch documentation.

Set up a cross-cluster connection

To replicate indexes from one domain to another, you need to set up a cross-cluster connection between the domains. The easiest way to connect domains is through the **Connections** tab of the domain dashboard. You can also use the [configuration API](#) or the [AWS CLI](#). Because cross-cluster replication follows a "pull" model, you initiate connections from the follower domain.

Note

If you previously connected two domains to perform [cross-cluster searches \(p. 251\)](#), you can't use that same connection for replication. The connection is marked as `SEARCH_ONLY` in the console. In order to perform replication between two previously connected domains, you must delete the connection and recreate it. When you've done this, the connection is available for both cross-cluster search and cross-cluster replication.

To set up a connection

1. In the Amazon OpenSearch Service console, select the follower domain, go to the **Connections** tab, and choose **Request**.
2. For **Connection alias**, enter a name for your connection.
3. Choose between connecting to a domain in your AWS account and Region or in another account or Region.
 - To connect to a domain in your AWS account and Region, select the domain and choose **Request**.
 - To connect to a domain in another AWS account or Region, specify the ARN of the remote domain and choose **Request**.

OpenSearch Service validates the connection request. If the domains are incompatible, the connection fails. If validation succeeds, it's sent to the destination domain for approval. When the destination domain approves the request, you can begin replication.

Start replication

After you establish a cross-cluster connection, you can begin to replicate data. First, create an index on the leader domain to replicate:

```
PUT leader-01
```

To replicate that index, send this command to the follower domain:

```
PUT _plugins/_replication/follower-01/_start
{
  "leader_alias": "connection-alias",
  "leader_index": "leader-01",
  "use_roles": {
    "leader_cluster_role": "all_access",
    "follower_cluster_role": "all_access"
  }
}
```

You can find the connection alias on the **Connections** tab on your domain dashboard.

This example assumes that an admin is issuing the request and uses `all_access` for the `leader_cluster_role` and `follower_cluster_role` for simplicity. In production environments, however, we recommend that you create replication users on both the leader and follower indexes, and map them accordingly. The user names must be identical. For information about these roles and how to map them, see [Map the leader and follower cluster roles](#) in the OpenSearch documentation.

Confirm replication

To confirm that replication is happening, get the replication status:

```
GET _plugins/_replication/follower-01/_status
{
  "status" : "SYNCING",
  "reason" : "User initiated",
  "leader_alias" : "connection-alias",
  "leader_index" : "leader-01",
  "follower_index" : "follower-01",
  "syncing_details" : {
    "leader_checkpoint" : -5,
    "follower_checkpoint" : -5,
    "seq_no" : 0
  }
}
```

The leader and follower checkpoint values begin as negative integers and reflect the number of shards you have (-1 for one shard, -5 for five shards, and so on). The values increment to positive integers with each change that you make. If the values are the same, it means that the indexes are fully synced. You can use these checkpoint values to measure replication latency across your domains.

To further validate replication, add a document to the leader index:

```
PUT leader-01/_doc/1
{
  "Doctor Sleep": "Stephen King"
}
```

And confirm that it shows up on the follower index:

```
GET follower-01/_search
```

```
{  
  ...  
  "max_score" : 1.0,  
  "hits" : [  
    {  
      "_index" : "follower-01",  
      "_type" : "_doc",  
      "_id" : "1",  
      "_score" : 1.0,  
      "_source" : {  
        "Doctor Sleep" : "Stephen King"  
      }  
    }  
  ]  
}
```

Pause and resume replication

You can temporarily pause replication if you need to remediate issues or reduce load on the leader domain. Send this request to the follower domain. Make sure to include an empty request body:

```
POST _plugins/_replication/follower-01/_pause  
{}
```

Then get the status to ensure that replication is paused:

```
GET _plugins/_replication/follower-01/_status  
{  
  "status" : "PAUSED",  
  "reason" : "User initiated",  
  "leader_alias" : "connection-alias",  
  "leader_index" : "leader-01",  
  "follower_index" : "follower-01"  
}
```

When you're done making changes, resume replication. Send this request to the follower domain. Make sure to include an empty request body:

```
POST _plugins/_replication/follower-01/_resume  
{}
```

You can't resume replication after it's been paused for more than 12 hours. You must stop replication, delete the follower index, and restart replication of the leader.

Stop replication

When you stop replication completely, the follower index unfollows the leader and becomes a standard index. You can't restart replication after you stop it.

Stop replication from the follower domain. Make sure to include an empty request body:

```
POST _plugins/_replication/follower-01/_stop  
{}
```

Auto-follow

You can define a set of replication rules against a single leader domain that automatically replicate indexes that match a specified pattern. When an index on the leader domain matches one of the patterns (for example, `books*`), a matching follower index is created on the follower domain. OpenSearch Service replicates any existing indexes that match the pattern, as well as new indexes that you create. It does not replicate indexes that already exist on the follower domain.

To replicate all indexes (with the exception of system-created indexes, and those that already exist on the follower domain), use a wildcard (*) pattern.

Create a replication rule

Create a replication rule on the follower domain, and specify the name of the cross-cluster connection:

```
POST _plugins/_replication/_autofollow
{
  "leader_alias" : "connection-alias",
  "name": "rule-name",
  "pattern": "books*",
  "use_roles":{
    "leader_cluster_role": "all_access",
    "follower_cluster_role": "all_access"
  }
}
```

You can find the connection alias on the [Connections](#) tab on your domain dashboard.

This example assumes that an admin is issuing the request, and it uses `all_access` as the leader and follower domain roles for simplicity. In production environments, however, we recommend that you create replication users on both the leader and follower indexes and map them accordingly. The user names must be identical. For information about these roles and how to map them, see [Map the leader and follower cluster roles](#) in the OpenSearch documentation.

To retrieve a list of existing replication rules on a domain, use the [auto-follow stats API operation](#).

To test the rule, create an index that matches the pattern on the leader domain:

```
PUT books-are-fun
```

And check that its replica appears on the follower domain:

```
GET _cat/indices
health status index          uuid                               pri  rep docs.count docs.deleted
store.size pri.store.size
green   open   books-are-fun  ldfH078xYYdxRMULuiTvSQ      1    1        0            0
208b           208b
```

Delete a replication rule

When you delete a replication rule, OpenSearch Service stops replicating *new* indices that match the pattern, but continues existing replication activity until you [stop replication \(p. 319\)](#) of those indexes.

Delete replication rules from the follower domain:

```
DELETE _plugins/_replication/_autofollow
{
```

```
    "leader_alias" : "connection-alias",
    "name": "rule-name"
}
```

Migrating Amazon OpenSearch Service indexes using remote reindex

Remote reindex lets you copy indexes from one Amazon OpenSearch Service cluster to another. You can migrate indexes from any OpenSearch Service domains or self-managed OpenSearch and Elasticsearch clusters.

Remote reindexing requires OpenSearch 1.0 or later, or Elasticsearch 6.7 or later, on the target domain. The source domain must be lower or the same major version as the target domain. Elasticsearch versions are considered to be *lower* than OpenSearch versions, meaning you can reindex data from Elasticsearch domains to OpenSearch domains. Within the same major version, the source domain can be any minor version. For example, remote reindexing from Elasticsearch 7.10.x to 7.9 is supported, but OpenSearch 1.0 to Elasticsearch 7.10.x isn't supported.

Full documentation for the `reindex` operation, including detailed steps and supported options, is available in the [OpenSearch documentation](#).

Topics

- [Prerequisites \(p. 321\)](#)
- [Reindex data between OpenSearch Service domains \(p. 321\)](#)
- [Reindex data between OpenSearch Service domains in a VPC \(p. 323\)](#)
- [Reindex data between non-OpenSearch Service domains \(p. 323\)](#)
- [Reindex large datasets \(p. 324\)](#)
- [Remote reindex settings \(p. 325\)](#)

Prerequisites

Remote reindex has the following requirements:

- The source domain must be accessible from the target domain. For a source domain that resides within a VPC, the target domain 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 "VPC support" \(p. 37\)](#).
- The request must be authorized by the source domain like any other REST request. If the source domain has fine-grained access control enabled, you must have permission to perform reindex on the target domain and read the index on the source domain. For more security considerations, see [the section called "Fine-grained access control" \(p. 146\)](#).
- We recommend you create an index with the desired setting on your target domain before you start the reindex process.

Reindex data between OpenSearch Service domains

The most basic scenario is that the source index is in the same AWS Region as your target domain with a publicly accessible endpoint and you have signed IAM credentials.

From the target domain, specify the source index to reindex from and the target index to reindex to:

```
POST _reindex
{
  "source": {
    "remote": {
      "host": "https://source-domain-endpoint:443"
    },
    "index": "source_index"
  },
  "dest": {
    "index": "target_index"
  }
}
```

You must add 443 at the end of the source domain endpoint for a validation check.

To verify that the index is copied over to the target domain, send this request to the target domain:

```
GET target_index/_search
```

If the source index is in a Region different from your target domain, pass in its Region name, such as in this sample request:

```
POST _reindex
{
  "source": {
    "remote": {
      "host": "https://source-domain-endpoint:443",
      "region": "eu-west-1"
    },
    "index": "source_index"
  },
  "dest": {
    "index": "target_index"
  }
}
```

In case of isolated Region like AWS GovCloud (US) or China Regions, the endpoint might not be accessible because your IAM user is not recognized in those Regions.

If the source domain is secured with basic authorization, specify the username and password:

```
POST _reindex
{
  "source": {
    "remote": {
      "host": "https://source-domain-endpoint:443",
      "username": "username",
      "password": "password"
    },
    "index": "source_index"
  },
  "dest": {
    "index": "target_index"
  }
}
```

If the source domain is hosted inside a VPC and does not have VPC-level connectivity, configure a proxy with a publicly accessible endpoint. The proxy domain must have a certificate signed by a public certificate authority (CA). Self-signed or private CA-signed certificates are not supported.

Reindex data between OpenSearch Service domains in a VPC

Every OpenSearch Service domain is made up of its own internal VPC infrastructure. When you create a new OpenSearch Service domain in an existing virtual private cloud (VPC), an Elastic Network Interface (ENI) is created for each data node in the OpenSearch Service VPC. Because the source reindex operation is performed from the target OpenSearch Service domain, and therefore within its own private VPC, you don't access the source OpenSearch Service domain's VPC. Instead, you need a publicly accessible reverse proxy.

A proxy is required in order to use remote reindex between two VPC domains, even if the domains are located within the same VPC. Create a proxy with a publicly accessible endpoint in front of the source cluster and pass the proxy endpoint in the reindex body. The proxy domain must have a certificate signed by a public certificate authority (CA). Self-signed or private CA-signed certificates are not supported.

To reindex data between OpenSearch Service domains in a VPC

1. Create an IAM user that has been granted access to both the local and remote OpenSearch Service domain. The following is an example access policy:

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

2. Set up an EC2 instance with a NGINX reverse proxy for the remote OpenSearch Service VPC endpoint. The EC2 instance must be within the same VPC as the OpenSearch Service domain. Because you're signing your requests, make sure that the NGINX configuration contains the following parameters:

```
proxy_set_header Host $host;  
proxy_set_header X-Real-IP $remote_addr;
```

3. Send the _reindex request and sign it with IAM credentials using [Signature Version 4](#). Send the request from a machine in the same VPC as the OpenSearch Service domain (either a running EC2 instance or a local machine connected through a VPN). Set the external parameter to true. For the source domain, specify the externally accessible URL for the NGINX reverse proxy.

Reindex data between non-OpenSearch Service domains

If the source index is hosted outside of OpenSearch Service, like in a self-managed EC2 instance, set the external parameter to true:

```
POST _reindex
```

```
{
  "source": {
    "remote": {
      "host": "https://source-domain-endpoint:443",
      "username": "username",
      "password": "password",
      "external": true
    },
    "index": "source_index"
  },
  "dest": {
    "index": "target_index"
  }
}
```

In this case, only basic authorization with a username and password is supported. The source domain must have a certificate signed by a public CA. Self-signed or private CA-signed certificates are not supported.

Reindex large datasets

Remote reindex sends a scroll request to the source domain with the following default values:

- Search context of 5 minutes
- Socket timeout of 30 seconds
- Batch size of 1,000

We recommend tuning these parameters to accommodate your data. For large documents, consider a smaller batch size and/or longer timeout. For more information, see [Scroll search](#).

```
POST _reindex?pretty=true&scroll=10h&wait_for_completion=false
{
  "source": {
    "remote": {
      "host": "https://source-domain-endpoint:443",
      "socket_timeout": "60m"
    },
    "size": 100,
    "index": "source_index"
  },
  "dest": {
    "index": "target_index"
  }
}
```

We also recommend adding the following settings to the target index for better performance:

```
PUT target_index
{
  "settings": {
    "refresh_interval": -1,
    "number_of_replicas": 0
  }
}
```

After the reindex process is complete, you can set your desired replica count and remove the refresh interval setting.

To reindex only a subset of documents that you select through a query, send this request to the target domain:

```
POST _reindex
{
  "source": {
    "remote": {
      "host": "https://source-domain-endpoint:443"
    },
    "index": "remote_index",
    "query": {
      "match": {
        "field_name": "text"
      }
    }
  },
  "dest": {
    "index": "target_index"
  }
}
```

Remote reindex doesn't support slicing, so you can't perform multiple scroll operations for the same request in parallel.

Remote reindex settings

In addition to the standard reindexing options, OpenSearch Service supports the following options:

Options	Valid values	Description	Required
external	Boolean	If the source domain is not an OpenSearch Service domain, or if you're reindexing between two VPC domains, specify as true.	No
region	String	If the source domain is in a different Region, specify the Region name.	No

Managing time-series data in Amazon OpenSearch Service with data streams

A typical workflow to manage time-series data involves multiple steps, such as creating a rollover index alias, defining a write index, and defining common mappings and settings for the backing indices.

Data streams in Amazon OpenSearch Service help simplify this initial setup process. Data streams work out of the box for time-based data such as application logs that are typically append-only in nature.

Data streams requires OpenSearch 1.0 or later. Full documentation for the feature is available in the [OpenSearch documentation](#).

Getting started with data streams

A data stream is internally composed of multiple backing indices. Search requests are routed to all the backing indices, while indexing requests are routed to the latest write index.

Step 1: Create an index template

To create a data stream, you first need to create an index template that configures a set of indexes as a data stream. The `data_stream` object indicates that it's a data stream and not a regular index template. The index pattern matches with the name of the data stream:

```
PUT _index_template/logs-template
{
  "index_patterns": [
    "my-data-stream",
    "logs-*"
  ],
  "data_stream": {},
  "priority": 100
}
```

In this case, each ingested document must have an `@timestamp` field. You can also define your own custom timestamp field as a property in the `data_stream` object:

```
PUT _index_template/logs-template
{
  "index_patterns": "my-data-stream",
  "data_stream": {
    "timestamp_field": {
      "name": "request_time"
    }
  }
}
```

Step 2: Create a data stream

After you create an index template, you can directly start ingesting data without creating a data stream.

Because we have a matching index template with a `data_stream` object, OpenSearch automatically creates the data stream:

```
POST logs-staging/_doc
{
  "message": "login attempt failed",
  "@timestamp": "2013-03-01T00:00:00"
}
```

Step 3: Ingest data into the data stream

To ingest data into a data stream, you can use the regular indexing APIs. Make sure every document that you index has a timestamp field. If you try to ingest a document that doesn't have a timestamp field, you get an error.

```
POST logs-redis/_doc
{
  "message": "login attempt",
```

```
    "@timestamp": "2013-03-01T00:00:00"  
}
```

Step 4: Searching a data stream

You can search a data stream just like you search a regular index or an index alias. The search operation applies to all of the backing indexes (all data present in the stream).

```
GET logs-redis/_search  
{  
  "query": {  
    "match": {  
      "message": "login"  
    }  
  }  
}
```

Step 5: Rollover a data stream

You can set up an [Index State Management \(ISM\) \(p. 305\)](#) policy to automate the rollover process for the data stream. The ISM policy is applied to the backing indexes at the time of their creation. When you associate a policy to a data stream, it only affects the future backing indexes of that data stream. You also don't need to provide the `rollover_alias` setting, because the ISM policy infers this information from the backing index.

Note

If you rollover a backing index to [cold storage \(p. 295\)](#), OpenSearch removes this index from the data stream. Even if you move the index back to [UltraWarm \(p. 286\)](#), the index remains independent and not part of the original data stream.

Step 6: Manage data streams in OpenSearch Dashboards

To manage data streams from OpenSearch Dashboards, open [OpenSearch Dashboards](#), choose **Index Management**, select **Indices** or **Policy managed indices**.

Step 7: Delete a data stream

The delete operation first deletes the backing indexes of a data stream and then deletes the data stream itself.

To delete a data stream and all of its hidden backing indices:

```
DELETE _data_stream/name_of_data_stream
```

Monitoring data in Amazon OpenSearch Service

Proactively monitor your data in Amazon OpenSearch Service with alerting and anomaly detection. Set up alerts to receive notifications when your data exceeds certain thresholds. Anomaly detection uses machine learning to automatically detect any outliers in your streaming data. You can pair anomaly detection with alerting to ensure you're notified as soon as an anomaly is detected.

Topics

- [Configuring alerts in Amazon OpenSearch Service \(p. 328\)](#)
- [Anomaly detection in Amazon OpenSearch Service \(p. 331\)](#)

Configuring alerts in Amazon OpenSearch Service

Configure alerts in Amazon OpenSearch Service to get notified when data from one or more 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 last 20 minutes.

Alerting requires OpenSearch or Elasticsearch 6.2 or later. For full documentation, including API descriptions, see the [OpenSearch documentation](#). This topic highlights the differences in alerting in OpenSearch Service compared to the open-source version.

To get started with alerting

1. Choose Alerting from the OpenSearch Dashboards main menu.
2. Set up a destination for the alert. Choose between Slack, Amazon Chime, a custom webhook, or Amazon SNS. As you might imagine, notifications require connectivity to the destination. For example, your OpenSearch Service domain must be able to connect to the internet to notify a Slack channel or send a custom webhook to a third-party server. The custom webhook must have a public IP address in order for an OpenSearch Service domain to send alerts to it.
3. Create a monitor in one of three ways: visually, using a query, or using an anomaly detector.
4. Define a condition to trigger the monitor.
5. (Optional) Add one or more actions to the monitor.

Tip

After an action successfully sends a message, securing access to that message (for example, access to a Slack channel) is your responsibility. If your domain contains sensitive data, consider using triggers without actions and periodically checking Dashboards for alerts.

For detailed steps, see [Monitors in the OpenSearch documentation](#).

Differences

Compared to the open-source version of OpenSearch, alerting in Amazon OpenSearch Service has some notable differences.

Amazon SNS support

OpenSearch Service supports Amazon Simple Notification Service ([Amazon SNS](#)) for notifications. This integration means that in addition to standard destinations (Slack, custom webhooks, and Amazon Chime), you can also send emails, text messages, and even run 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. Choose **Alerting** from the OpenSearch Dashboards main menu.
2. Go to the **Destinations** tab and then choose **Add destination**.
3. Provide a unique name for the destination.
4. For **Type**, choose **Amazon SNS**.
5. Provide the SNS topic ARN.
6. Provide the ARN for an IAM role within your account that has the following trust relationship and permissions (at minimum):

Trust relationship

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

We recommend that you use the `aws:SourceAccount` and `aws:SourceArn` condition keys to protect yourself against the [confused deputy problem](#). The source account is the owner of the domain and the source ARN is the ARN of the domain. Your domain must be on service software R20211203 or later in order to add these condition keys.

For example, you could add the following condition block to the trust policy:

```
"Condition": {  
    "StringEquals": {  
        "aws:SourceAccount": "account-id"  
    },  
    "ArnLike": {  
        "aws:SourceArn": "arn:aws:es:region:account-id:domain/domain-name"  
    }  
}
```

Permissions

```
{  
    "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*.

7. Choose **Create**.

Alerting settings

OpenSearch Service lets you modify the following [alerting settings](#):

- `plugins.scheduled_jobs.enabled`
- `plugins.alerting.alert_history_enabled`
- `plugins.alerting.alert_history_max_age`
- `plugins.alerting.alert_history_max_docs`
- `plugins.alerting.alert_history_retention_period`
- `plugins.alerting.alert_history_rollover_period`
- `plugins.alerting.filter_by_backend_roles`

All other settings use the default values which you can't change.

To disable alerting, send the following request:

```
PUT _cluster/settings
{
  "persistent" : {
    "plugins.scheduled_jobs.enabled" : false
  }
}
```

The following request configures alerting to automatically delete history indices after seven days, rather than the default 30 days:

```
PUT _cluster/settings
{
  "persistent": {
    "plugins.alerting.alert_history_retention_period": "7d"
  }
}
```

If you previously created monitors and want to stop the creation of daily alerting indices, delete all alert history indices:

```
DELETE .plugins-alerting-alert-history-*
```

To reduce shard count for history indices, create an index template. The following request sets history indexes for alerting to one shard and one replica:

```
PUT _index_template/template-name
{
  "index_patterns": [".opendistro-alerting-alert-history-*"],
  "template": {
    "settings": {
      "number_of_shards": 1,
      "number_of_replicas": 1
    }
  }
}
```

}

Depending on your tolerance for data loss, you might even consider using zero replicas. For more information about creating and managing index templates, see [Index templates](#) in the OpenSearch documentation.

Alerting permissions

Alerting supports [fine-grained access control \(p. 146\)](#). For details on mixing and matching permissions to fit your use case, see [Alerting security](#) in the OpenSearch documentation.

Anomaly detection in Amazon OpenSearch Service

Anomaly detection in Amazon OpenSearch Service automatically detects anomalies in your OpenSearch data in near-real time by using the Random Cut Forest (RCF) algorithm. RCF is an unsupervised machine learning algorithm that models a sketch of your incoming data stream. The algorithm computes an anomaly grade and confidence score value for each incoming data point. Anomaly detection uses these values to differentiate an anomaly from normal variations in your data.

You can pair the anomaly detection plugin with the [the section called "Alerting" \(p. 328\)](#) plugin to notify you as soon as an anomaly is detected.

Anomaly detection is available on domains running any OpenSearch version or Elasticsearch 7.4 or later. All instance types support anomaly detection except for t2.micro and t2.small. Full documentation for anomaly detection, including detailed steps and API descriptions, is available in the [OpenSearch documentation](#).

Prerequisites

Anomaly detection has the following prerequisites:

- Anomaly detection requires OpenSearch or Elasticsearch 7.4 or later.
- Anomaly detection only supports [fine-grained access control \(p. 146\)](#) on Elasticsearch versions 7.9 and later and all versions of OpenSearch. Prior to Elasticsearch 7.9, only admin users can create, view, and manage detectors.
- If your domain uses fine-grained access control, non-admin users must be [mapped \(p. 154\)](#) to the `anomaly_read_access` role in OpenSearch Dashboards in order to view detectors, or `anomaly_full_access` in order to create and manage detectors.

Getting started with anomaly detection

To get started, choose **Anomaly Detection** in OpenSearch Dashboards.

Step 1: Create a detector

A detector is an individual anomaly detection task. You can create multiple detectors, and all the detectors can run simultaneously, with each analyzing data from different sources.

Step 2: Add features to your detector

A feature is the field in your index that you check for anomalies. A detector can discover anomalies across one or more features. You must choose one of the following aggregations for each feature: `average()`, `sum()`, `count()`, `min()`, or `max()`.

Note

The `count()` aggregation method is only available in OpenSearch and Elasticsearch 7.7 or later. For Elasticsearch 7.4, use a custom expression like the following:

```
{  
  "aggregation_name": {  
    "value_count": {  
      "field": "field_name"  
    }  
  }  
}
```

The aggregation method determines what constitutes an anomaly. For example, if you choose `min()`, the detector focuses on finding anomalies based on the minimum values of your feature. If you choose `average()`, the detector finds anomalies based on the average values of your feature. You can add a maximum of five features per detector.

You can configure the following optional settings (available in Elasticsearch 7.7 and later):

- **Category field** - Categorize or slice your data with a dimension like IP address, product ID, country code, and so on.
- **Window size** - Set the number of aggregation intervals from your data stream to consider in a detection window.

After you set up your features, preview sample anomalies and adjust the feature settings if necessary.

Step 3: Observe the results

cpu_ad • Running since 11/13/2010 10:04 AM

Anomaly results

Detector configuration

Live anomalies Live

View anomaly results during the last 60 intervals (60 minutes).



Anomaly history

Choose a filled rectangle in the heat map for a more detailed view.

host

API Version 2015-01-01
Top 10 335



By s

- **Live anomalies** - displays the live anomaly results for the last 60 intervals. For example, if the interval is set to 10, it shows the results for the last 600 minutes. This chart refreshes every 30 seconds.
- **Anomaly history** - plots the anomaly grade with the corresponding measure of confidence.
- **Feature breakdown** - plots the features based on the aggregation method. You can vary the date-time range of the detector.
- **Anomaly occurrence** - shows the Start time, End time, Data confidence, and Anomaly grade for each anomaly detected.

If you set the category field, you see an additional **Heat map** chart that correlates results for anomalous entities. Choose a filled rectangle to see a more detailed view of the anomaly.

Step 4: Set up alerts

To create a monitor to send you notifications when any anomalies are detected, choose **Set up alerts**. The plugin redirects you to the [Add monitor](#) page where you can configure an alert.

Tutorial: Detect high CPU usage with anomaly detection

This tutorial demonstrates how to create an anomaly detector in Amazon OpenSearch Service to detect high CPU usage. You'll use OpenSearch Dashboards to configure a detector to monitor CPU usage, and generate an alert when your CPU usage rises above a specified threshold.

Note

These steps apply to the latest version of OpenSearch and might differ slightly for past versions.

Prerequisites

- You must have an OpenSearch Service domain running Elasticsearch 7.4 or later, or any OpenSearch version.
- You must be ingesting application log files into your cluster that contain CPU usage data.

Step 1: Create a detector

First, create a detector that identifies anomalies in your CPU usage data.

1. Open the left panel menu in OpenSearch Dashboards and choose **Anomaly Detection**, then choose **Create detector**.
2. Name the detector **high-cpu-usage**.
3. For your data source, choose your index that contains CPU usage log files where you want to identify anomalies.
4. Choose the **Timestamp field** from your data. Optionally, you can add a data filter. This data filter analyzes only a subset of the data source and reduces the noise from data that's not relevant.
5. Set the **Detector interval** to 2 minutes. This interval defines the time (by minute interval) for the detector to collect the data.
6. In **Window delay**, add a **1-minute** delay. This delay adds extra processing time to ensure that all data within the window is present.
7. Choose **Next**. On the anomaly detection dashboard, under the detector name, choose **Configure model**.
8. For **Feature name**, enter **max_cpu_usage**. For **Feature state**, select **Enable feature**.
9. For **Find anomalies based on**, choose **Field value**.

10. For **Aggregation method**, choose **max()**.
11. For **Field**, select the field in your data to check for anomalies. For example, it might be called `cpu_usage_percentage`.
12. Keep all other settings as their defaults and choose **Next**.
13. Ignore the detector jobs setup and choose **Next**.
14. In the pop-up window, choose when to start the detector (automatically or manually), and then choose **Confirm**.

Now that the detector is configured, after it initializes, you will be able to see real-time results of the CPU usage in the **Real-time results** section of your detector panel. The **Live anomalies** section displays any anomalies that occur as data is being ingested in real time.

Step 2: Configure an alert

Now that you've created a detector, create a monitor that invokes an alert to send a message to Slack when it detects CPU usage that meets the conditions specified in the detector settings. You'll receive Slack notifications when data from one or more indexes meets the conditions that invoke the alert.

1. Open the left panel menu in OpenSearch Dashboards and choose **Alerting**, then choose **Create monitor**.
2. Provide a name for the monitor.
3. For **Monitor type**, choose **Per-query monitor**. A per-query monitor runs a specified query and defines the triggers.
4. For **Monitor defining method**, choose **Anomaly detector**, then select the detector that you created in the previous section from the **Detector** dropdown menu.
5. For **Schedule**, choose how often the monitor collects data and how often you receive alerts. For the purposes of this tutorial, set the schedule to run every **7 minutes**.
6. In the **Triggers** section, choose **Add trigger**. For **Trigger name**, enter **High CPU usage**. For this tutorial, for **Severity level**, choose **1**, which is the highest level of severity.
7. For **Anomaly grade threshold**, choose **IS ABOVE**. On the menu under that, choose the grade threshold to apply. For this tutorial, set the **Anomaly grade** to **0.7**.
8. For **Anomaly confidence threshold**, choose **IS ABOVE**. On the menu under that, enter the same number as your Anomaly grade. For this tutorial, set the **Anomaly confidence threshold** to **0.7**.
9. In the **Actions** section, choose **Destination**. In the **Name** field, choose the name of the destination. On the **Type** menu, choose **Slack**. In the **Webhook URL** field, enter a webhook URL to receive alerts to. For more information, see [Sending messages using incoming webhooks](#).

10 Choose **Create**.

Related resources

- [Configuring alerts in Amazon OpenSearch Service \(p. 328\)](#)
- [Anomaly detection in Amazon OpenSearch Service \(p. 331\)](#)
- [Anomaly detection API](#)

Observability in Amazon OpenSearch Service

The default installation of OpenSearch Dashboards for Amazon OpenSearch Service includes the Observability plugin, which you can use to visualize data-driven events using Piped Processing Language (PPL) in order to explore, discover, and query data stored in OpenSearch. The plugin requires OpenSearch 1.2 or later.

The Observability plugin provides a unified experience for collecting and monitoring metrics, logs, and traces from common data sources. Data collection and monitoring in one place enables full-stack, end-to-end observability of your entire infrastructure. Full documentation for the Observability plugin is in the [OpenSearch documentation](#).

Everyone's process for exploring data is different. If you're new to exploring data and creating visualizations, we recommend trying a workflow like the following:

Explore your data with event analytics

To start, let's say that you're collecting flight data in your OpenSearch Service domain and you want to find out which airline had the most flights arriving in Pittsburgh International Airport last month. You write the following PPL query:

```
source=opensearch_dashboards_sample_data_flights |  
  stats count() by Dest, Carrier |  
  where Dest = "Pittsburgh International Airport"
```

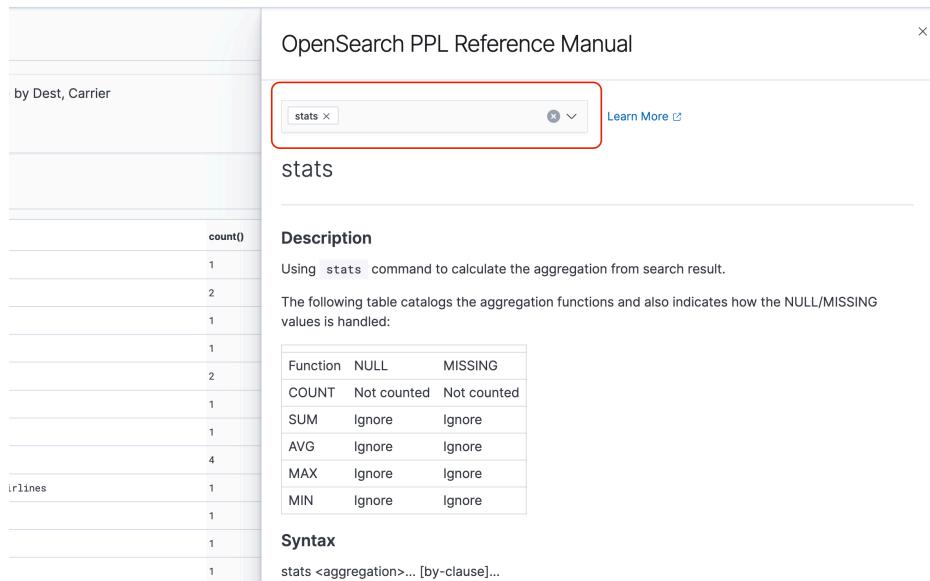
This query pulls data from the index named `opensearch_dashboards_sample_data_flights`. It then uses the `stats` command to get a total count of flights and groups it according to destination airport and carrier. Finally, it uses the `where` clause to filter the results to flights arriving in Pittsburgh International Airport.

Here's what the data looks like when displayed over the last month:

The screenshot shows the Amazon OpenSearch Service Observability Explorer interface. At the top, there is a navigation bar with 'Observability / Event analytics / Explorer'. Below the navigation, a search bar contains the PPL query: `source=opensearch_dashboards_sample_data_flights | stats count() by Dest, Carrier | where Dest = "Pittsburgh International Airport"`. To the right of the search bar are buttons for 'Month to date' (with a calendar icon), 'Show dates' (with a dropdown arrow), 'Refresh' (with a circular arrow icon), and 'Save' (with a save icon). The main area is divided into two tabs: 'Events' (selected) and 'Visualizations'. On the left, there is a sidebar with sections for 'Query fields' (Carrier, count(), Dest), 'Selected Fields', and 'Available Fields'. The main content area displays a table with the following data:

Carrier	count()	Dest
BeatsWest	5	Pittsburgh International Airport
Logstash Airways	6	Pittsburgh International Airport
OpenSearch Dashboards Airlines	6	Pittsburgh International Airport
OpenSearch-Air	11	Pittsburgh International Airport

You can choose the **PPL** button in the query editor to get usage information and examples for each PPL command:



The screenshot shows a search bar with "stats" typed in, which is highlighted with a red box. Below the search bar, the word "stats" is displayed in bold. A "Description" section follows, containing a brief explanation of the stats command and a table cataloging aggregation functions and their handling of NULL/MISSING values.

Function	NULL	MISSING
COUNT	Not counted	Not counted
SUM	Ignore	Ignore
AVG	Ignore	Ignore
MAX	Ignore	Ignore
MIN	Ignore	Ignore

A "Syntax" section provides the command structure: `stats <aggregation>... [by-clause]...`

Let's look at a more complex example, which queries for information about flight delays:

```
source=opensearch_dashboards_sample_data_flights |
  where FlightDelayMin > 0 |
  stats sum(FlightDelayMin) as minimum_delay, count() as total_delayed by Carrier, Dest |
  eval avg_delay=minimum_delay / total_delayed |
  sort - avg_delay
```

Each command in the query impacts the final output:

- `source=opensearch_dashboards_sample_data_flights` - pulls data from the same index as the previous example
- `where FlightDelayMin > 0` - filters the data to flights that were delayed
- `stats sum(FlightDelayMin) as minimum_delay, count() as total_delayed by Carrier` - for each carrier, gets the total minimum delay time and the total count of delayed flights
- `eval avg_delay=minimum_delay / total_delayed` - calculates the average delay time for each carrier by dividing the minimum delay time by the total number of delayed flights
- `sort - avg_delay` - sorts the results by average delay in descending order

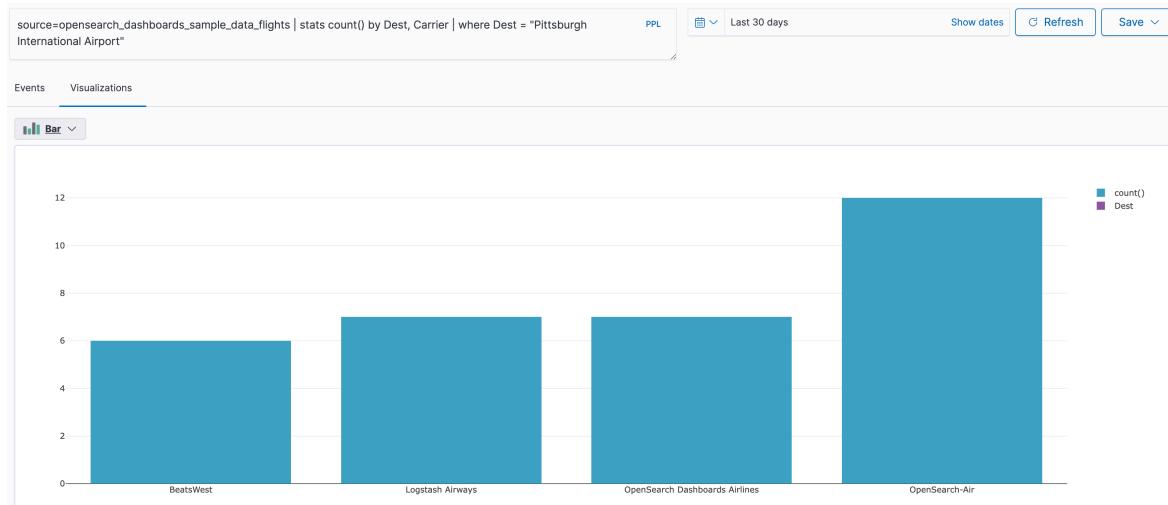
With this query, you can determine that OpenSearch Dashboards Airlines has, on average, fewer delays.

	avg_delay	Carrier	minimum_delay	total_delayed
>	212	Logstash Airways	4470	21
>	184	OpenSearch-Air	4245	23
>	155	BeatsWest	2025	13
>	153	OpenSearch Dashboards Airlines	4305	28

You can find more sample PPL queries under **Queries and Visualizations** on the **Event analytics** page.

Create visualizations

Once you correctly query the data that you're interested in, you can save those queries as visualizations:



Then add those visualizations to [operational panels](#) to compare different pieces of data. Leverage [notebooks](#) to combine different visualizations and code blocks that you can share with team members.

Dive deeper with Trace Analytics

[Trace Analytics \(p. 338\)](#) provides a way to visualize the flow of events in your OpenSearch data to identify and fix performance problems in distributed applications.

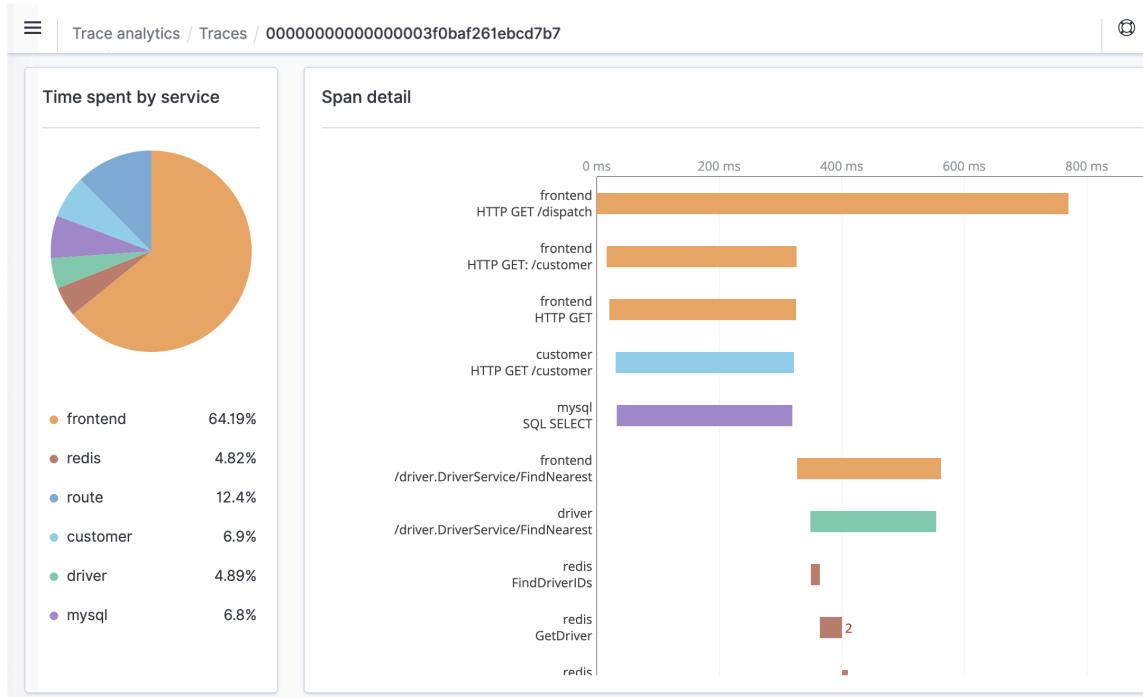


Trace Analytics for Amazon OpenSearch Service

You can use Trace Analytics, which is part of the OpenSearch Observability plugin, to analyze trace data from distributed applications. Trace Analytics requires OpenSearch or Elasticsearch 7.9 or later.

In a distributed application, a single operation, such as a user clicking a button, can trigger an extended series of events. For example, the application front end might call a backend service, which calls another service, which queries a database, which processes the query and returns a result. Then the first backend service sends a confirmation to the front end, which updates the UI.

You can use Trace Analytics to help you visualize this flow of events and identify performance problems.



Prerequisites

Trace Analytics requires you to add [instrumentation](#) to your application and generate trace data using an OpenTelemetry-supported library such as [Jaeger](#) or [Zipkin](#). This step occurs entirely outside of OpenSearch Service. The [AWS Distro for OpenTelemetry documentation](#) contains example applications for many programming languages that can help you get started, including Java, Python, Go, and JavaScript.

After you add instrumentation to your application, the [OpenTelemetry Collector](#) receives data from the application and formats it into OpenTelemetry data. See the list of receivers on [GitHub](#). AWS Distro for OpenTelemetry includes a [receiver for AWS X-Ray](#).

Finally, [Data Prepper](#), an independent OpenSearch component, formats that OpenTelemetry data for use with OpenSearch. Data Prepper runs on a machine outside of the OpenSearch Service cluster, similar to Logstash.

For a Docker Compose file that demonstrates the end-to-end flow of data, see the [OpenSearch documentation](#).

OpenTelemetry Collector sample configuration

To use the OpenTelemetry Collector with Data Prepper, try the following sample configuration:

```

receivers:
  jaeger:
    protocols:
      grpc:
  otlp:
    protocols:
      grpc:
  zipkin:

exporters:

```

```
otlp/data-prepper:  
  endpoint: data-prepper-host:21890  
  insecure: true  
  
service:  
  pipelines:  
    traces:  
      receivers: [jaeger, otlp, zipkin]  
      exporters: [otlp/data-prepper]
```

Data Prepper sample configuration

To send trace data to an OpenSearch Service domain, try the following sample configuration files.

data-prepper-config.yaml

```
ssl: true  
keyStoreFilePath: "/usr/share/data-prepper/keystore.jks" # required if ssl is true  
keyStorePassword: "password" # optional, defaults to empty string  
privateKeyPassword: "other_password" # optional, defaults to empty string  
serverPort: 4900 # port for administrative endpoints, default is 4900
```

pipelines.yaml

```
entry-pipeline:  
  # Workers is the number of application threads.  
  # Try setting this value to the number of CPU cores on the machine.  
  # We recommend the same number of workers for all pipelines.  
  workers: 4  
  delay: "100" # milliseconds  
  source:  
    otel_trace_source:  
      ssl: true  
      sslKeyCertChainFile: "config/demo-data-prepper.crt"  
      sslKeyFile: "config/demo-data-prepper.key"  
  buffer:  
    bounded_blocking:  
      # Buffer size is the number of export requests to hold in memory.  
      # We recommend the same value for all pipelines.  
      # Batch size is the maximum number of requests each worker thread processes within  
      the delay.  
      # Keep buffer size >= number of workers * batch size.  
      buffer_size: 1024  
      batch_size: 256  
  sink:  
    - pipeline:  
        name: "raw-pipeline"  
    - pipeline:  
        name: "service-map-pipeline"  
raw-pipeline:  
  workers: 4  
  # We recommend the default delay for the raw pipeline.  
  delay: "3000"  
  source:  
    pipeline:  
      name: "entry-pipeline"  
  prepper:  
    - otel_trace_raw_prepper:  
  buffer:  
    bounded_blocking:  
      buffer_size: 1024  
      batch_size: 256  
  sink:
```

```

- opensearch:
    hosts: ["https://domain-endpoint"]
    # # Basic authentication
    # username: "ta-user"
    # password: "ta-password"
    # IAM signing
    aws_sigv4: true
    aws_region: "us-east-1"
    trace_analytics_raw: true
service-map-pipeline:
    workers: 4
    delay: "100"
    source:
        pipeline:
            name: "entry-pipeline"
prepper:
    - service_map_stateful:
buffer:
    bounded_blocking:
        buffer_size: 1024
        batch_size: 256
sink:
    - opensearch:
        hosts: ["https://domain-endpoint"]
        # # Basic authentication
        # username: "ta-user"
        # password: "ta-password"
        # IAM signing
        aws_sigv4: true
        aws_region: "us-east-1"
        trace_analytics_service_map: true

```

- For IAM signing, run `aws configure` using the AWS CLI to set your credentials.
- If you use [fine-grained access control \(p. 146\)](#) with the internal user database, use the basic authentication lines instead.

If your domain uses fine-grained access control, you must map the Data Prepper user or role to the [all_access role \(p. 158\)](#).

If your domain doesn't use fine-grained access control, the Data Prepper user or role must have write permissions to several indices and templates, along with permissions to access an Index State Management (ISM) policy and retrieve cluster settings. The following policy shows the required permissions:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/data-prepper-sink-user"
            },
            "Action": "es:ESHttp*",
            "Resource": [
                "arn:aws:es:us-east-1:123456789012:domain/domain-name/otel-v1*",
                "arn:aws:es:us-east-1:123456789012:domain/domain-name/_template/otel-v1*",
                "arn:aws:es:us-east-1:123456789012:domain/domain-name/_plugins/_ism/policies/raw-span-policy",
                "arn:aws:es:us-east-1:123456789012:domain/domain-name/_alias/otel-v1*"
            ]
        },
        {
            "Effect": "Allow",

```

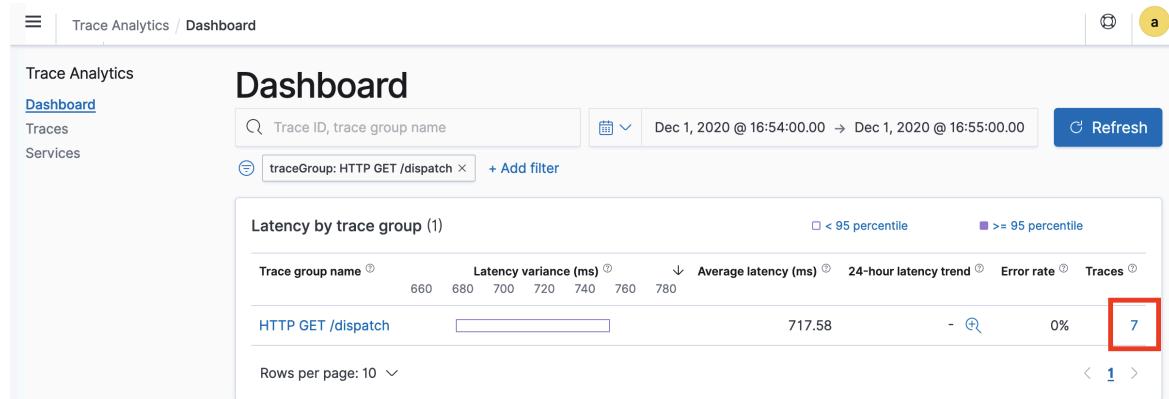
```
"Principal": {  
    "AWS": "arn:aws:iam::123456789012:user/data-prepper-sink-user"  
},  
"Action": "es:ESHttpGet",  
"Resource": "arn:aws:us-east-1:123456789012:domain/domain-name/_cluster/settings"  
}  
]  
}
```

Data Prepper uses port 21890 to receive data, and it must be able to connect to both the OpenTelemetry Collector and the OpenSearch cluster. For performance tuning, adjust the worker count and buffer settings in your configuration file, along with the Java virtual machine (JVM) heap size for the machine.

Full documentation for Data Prepper is available in the [OpenSearch documentation](#). For convenience, we also provide an [AWS CloudFormation template](#) that installs Data Prepper on an Amazon EC2 instance.

Exploring trace data

The **Dashboard** view groups traces together by HTTP method and path so that you can see the average latency, error rate, and trends associated with a particular operation. For a more focused view, try filtering by trace group name.



To drill down on the traces that make up a trace group, choose the number of traces in the right-hand column. Then choose an individual trace for a detailed summary.

The **Services** view lists all services in the application, plus an interactive map that shows how the various services connect to each other. In contrast to the dashboard (which helps identify problems by operation), the service map helps you identify problems by service. Try sorting by error rate or latency to get a sense of potential problem areas of your application.

The screenshot shows the 'Services' section of the Trace Analytics interface. On the left, there's a sidebar with 'Trace Analytics' and 'Dashboard' options, and 'Services' is selected. At the top right, there are refresh and search icons. Below the header, a date range selector shows 'Dec 1, 2020 @ 16:54:00.000 → Dec 1, 2020 @ 16:55:00.000' and a 'Refresh' button. The main area is titled 'Services (6)' and contains a table with the following data:

Name	Average latency (ms)	Error rate ↓	Throughput	No. of connected services	Connected services	Traces
redis	14.98	18.72%	203	1	driver	7
frontend	290.73	2.08%	48	3	driver, customer, route	14
route	48.88	0%	150	1	frontend	7
customer	308.72	0%	15	2	mysql, frontend	7
driver	204.94	0%	15	2	redis, frontend	7
mysql	308	0%	15	1	customer	7

Below the table, there are buttons for 'Rows per page: 10' and navigation arrows (< 1 >).

Querying Amazon OpenSearch Service data using Piped Processing Language

Piped Processing Language (PPL) is a query language that lets you use pipe (|) syntax to query data stored in Amazon OpenSearch Service.

The PPL syntax consists of commands delimited by a pipe character (|) where data flows from left to right through each pipeline. For example, the PPL syntax to find the number of hosts with HTTP 403 or 503 errors, aggregate them per host, and sort them in the order of impact is as follows:

```
source = dashboards_sample_data_logs | where response='403' or response='503' | stats count(request) as request_count by host, response | sort -request_count
```

PPL requires either OpenSearch or Elasticsearch 7.9 or later. Detailed steps and command descriptions are available in the [OpenSearch PPL reference manual](#).

To get started, choose **Query Workbench** in OpenSearch Dashboards and select **PPL**. Use the bulk operation to index some sample data:

```
PUT accounts/_bulk?refresh
{"index": {"_id": "1"}}
{"account_number": 1, "balance": 39225, "firstname": "Amber", "lastname": "Duke", "age": 32, "gender": "M", "address": "Holmes Lane", "employer": "Pyrami", "email": "amberduke@pyrami.com", "city": "Brogan", "state": "IL"}
{"index": {"_id": "6"}}
{"account_number": 6, "balance": 5686, "firstname": "Hattie", "lastname": "Bond", "age": 36, "gender": "M", "address": "Bristol Street", "employer": "Netagy", "email": "hattiebond@netagy.com", "city": "Dante", "state": "TN"}
 {"index": {"_id": "13"}}
 {"account_number": 13, "balance": 32838, "firstname": "Nanette", "lastname": "Bates", "age": 28, "gender": "F", "address": "Mady Street", "employer": "Quility", "city": "Nogal", "state": "VA"}
```

```
{"index":{"_id":"18"}}
{"account_number":18,"balance":4180,"firstname":"Dale","lastname":"Adams","age":33,"gender":"M","address":"Hutchinson Court","email":"daleadams@boink.com","city":"Orick","state":"MD"}
```

The following example returns `firstname` and `lastname` fields for documents in an accounts index with age greater than 18:

```
search source=accounts | where age > 18 | fields firstname, lastname
```

Sample Response

id	firstname	lastname
0	Amber	Duke
1	Hattie	Bond
2	Nanette	Bates
3	Dale	Adams

You can use a complete set of read-only commands like `search`, `where`, `fields`, `rename`, `dedup`, `stats`, `sort`, `eval`, `head`, `top`, and `rare`. For descriptions and examples of each command, see the [OpenSearch PPL reference manual](#).

The PPL plugin supports all SQL functions, including mathematical, trigonometric, date-time, string, aggregate, and advanced operators and expressions. To learn more, see the [OpenSearch PPL reference manual](#).

Operational best practices for Amazon OpenSearch Service

This chapter provides best practices for operating Amazon OpenSearch Service domains and includes general guidelines that apply to many use cases. Each workload is unique, with unique characteristics, so no generic recommendation is exactly right for every use case. The most important best practice is to deploy, test, and tune your domains in a continuous cycle to find the optimal configuration, stability, and cost for your workload.

Topics

- [Monitoring and alerting \(p. 345\)](#)
- [Shard strategy \(p. 346\)](#)
- [Stability \(p. 347\)](#)
- [Performance \(p. 350\)](#)
- [Security \(p. 351\)](#)
- [Cost optimization \(p. 352\)](#)
- [Sizing Amazon OpenSearch Service domains \(p. 353\)](#)
- [Petabyte scale in Amazon OpenSearch Service \(p. 357\)](#)
- [Dedicated master nodes in Amazon OpenSearch Service \(p. 358\)](#)
- [Recommended CloudWatch alarms for Amazon OpenSearch Service \(p. 361\)](#)

Monitoring and alerting

The following best practices apply to monitoring your OpenSearch Service domains.

Configure CloudWatch alarms

OpenSearch Service emits performance metrics to Amazon CloudWatch. Regularly review your [cluster and instance metrics \(p. 67\)](#) and configure [recommended CloudWatch alarms \(p. 361\)](#) based on your workload performance.

Enable log publishing

OpenSearch Service exposes OpenSearch error logs, search slow logs, indexing slow logs, and audit logs in Amazon CloudWatch Logs. Search slow logs, indexing slow logs, and error logs are useful for troubleshooting performance and stability issues. Audit logs, which are only available if you enable [fine-grained access control \(p. 146\)](#) to track user activity. For more information, see [Logs](#) in the OpenSearch documentation.

Search slow logs and indexing slow logs are an important tool for understanding and troubleshooting the performance of your search and indexing operations. [Enable search and index slow log delivery \(p. 92\)](#) for all production domains. You must also [configure logging thresholds \(p. 96\)](#)—otherwise, CloudWatch won't capture the logs.

Shard strategy

Shards distribute your workload across the data nodes in your OpenSearch Service domain. Properly configured indexes can help boost overall domain performance.

When you send data to OpenSearch Service, you send that data to an index. An index is analogous to a database table, with *documents* as the rows, and *fields* as the columns. When you create the index, you tell OpenSearch how many primary shards you want to create. The primary shards are independent partitions of the full dataset. OpenSearch Service automatically distributes your data across the primary shards in an index. You can also configure *replicas* of the index. Each replica shard comprises a full set of copies of the primary shards for that index.

OpenSearch Service maps the shards for each index across the data nodes in your cluster. It ensures that the primary and replica shards for the index reside on different data nodes. The first replica ensures that you have two copies of the data in the index. You should always use at least one replica. Additional replicas provide additional redundancy and read capacity.

OpenSearch sends index and search requests to all of the data nodes that contain shards that belong to the index in the request. It sends indexing requests first to data nodes that contain primary shards, and then to data nodes that contain replica shards.

For example, for an index with five primary shards and one replica, each indexing request uses 10 shards. In contrast, search queries are sent to n shards, where n is the number of primary shards. For an index with five primary shards and one replica, each search query uses five shards (primary or replica) from that index.

Determine shard and data node counts

Use the following best practices to determine shard and data node counts for your domain.

Shard size – The size of data on disk is a direct result of the size of your source data, and it changes as you index more data. The source-to-index ratio can vary wildly, from 1:10 to 10:1 or more, but usually it's around 1:1.10. You can use that ratio to predict the index size on disk. You can also index some data and retrieve the actual index sizes to determine the ratio for your workload. After you have a predicted index size, set a shard count so that each shard will be between 10–30 GiB (for search workloads), or between 30–50 GiB (for logs workloads). 50 GiB should be the maximum—be sure to plan for growth.

Shard count – The distribution of shards to data nodes has a large impact on a domain's performance. When you have indexes with multiple shards, try to make the shard count an even multiple of the data node count. This helps to ensure that shards are evenly distributed across data nodes, and prevents hot nodes. For example, if you have 12 primary shards, your data node count should be 2, 3, 4, 6, or 12. However, shard count is secondary to shard size—if you have 5 GiB of data, you should still use a single shard.

Shards per data node – The total number of shards that a node can hold is proportional to the node's Java virtual machine (JVM) heap memory. Aim for 25 shards or fewer per GiB of heap memory. For example, a node with 32 GiB of heap memory should hold no more than 800 shards. Although shard distribution can vary based on your workload patterns, there's a limit of 1,000 shards per node. The [cat/allocation](#) API provides a quick view of the number of shards and total shard storage across data nodes.

Shard to CPU ratio – When a shard is involved in an indexing or search request, it uses a vCPU to process the request. As a best practice, use an initial scale point of 1.5 vCPU per shard. If your instance type has 8 vCPUs, set your data node count so that each node has no more than six shards. Note that this is an approximation. Be sure to test your workload and scale your cluster accordingly.

For storage volume, shard size, and instance type recommendations, see the following resources:

- the section called "Sizing domains" (p. 353)
- the section called "Petabyte scale" (p. 357)

Avoid storage skew

Storage skew occurs when one or more nodes within a cluster holds a higher proportion of storage for one or more indexes than the others. Indications of storage skew include uneven CPU utilization, intermittent and uneven latency, and uneven queueing across data nodes. To determine whether you have skew issues, see the following troubleshooting sections:

- [the section called “Node shard and storage skew” \(p. 446\)](#)
- [the section called “Index shard and storage skew” \(p. 446\)](#)

Stability

The following best practices apply to maintaining a stable and healthy OpenSearch Service domain.

Keep current with OpenSearch

Service software updates

OpenSearch Service regularly releases [software updates \(p. 29\)](#) that add features or otherwise improve your domains. Updates don't change the OpenSearch or Elasticsearch engine version. We recommend that you schedule a recurring time to run the [DescribeDomain](#) API operation, and initiate a service software update if the `UpdateStatus` is `ELIGIBLE`. If you don't update your domain within a certain time frame (typically two weeks), OpenSearch Service automatically performs the update.

OpenSearch version upgrades

OpenSearch Service regularly adds support for community-maintained versions of OpenSearch. Always upgrade to the latest OpenSearch versions when they're available.

OpenSearch Service simultaneously upgrades both OpenSearch and OpenSearch Dashboards (or Elasticsearch and Kibana if your domain is running a legacy engine). 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. OpenSearch Dashboards might be unavailable during some or all of the upgrade.

There are two ways to upgrade a domain:

- [In-place upgrade \(p. 53\)](#) – This option is easier because you keep the same cluster.
- [Snapshot/restore upgrade \(p. 56\)](#) – This option is good for testing new versions on a new cluster or migrating between clusters.

Regardless of which upgrade process you use, we recommend that you maintain a domain that is solely for development and testing, and upgrade it to the new version *before* you upgrade your production domain. Choose **Development and testing** for the deployment type when you're creating the test domain. Make sure to upgrade all clients to compatible versions immediately following the domain upgrade.

Back up your data

You can take manual snapshots for cluster recovery, or to move data from one cluster to another. You have to initiate or schedule manual snapshots. Snapshots are stored in your own Amazon S3 bucket. For instructions on how to take and restore a snapshot, see [the section called “Creating index snapshots” \(p. 42\)](#).

Enable dedicated master nodes

Dedicated master nodes (p. 358) improve cluster stability. A dedicated master node performs cluster management tasks, but doesn't hold index data or respond to client requests. This offloading of cluster management tasks increases the stability of your domain and makes it possible for some configuration changes (p. 22) to happen without downtime.

Enable and use three dedicated master nodes for optimal domain stability across three Availability Zones. For instance type recommendations, see the section called "Choosing instance types for dedicated master nodes" (p. 360).

Deploy across multiple 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 (p. 34) in the same AWS Region. Availability Zones are isolated locations within each Region. With a two-AZ configuration, losing one Availability Zone means that you lose half of all domain capacity. Moving to three Availability Zones further reduces the impact of losing a single Availability Zone.

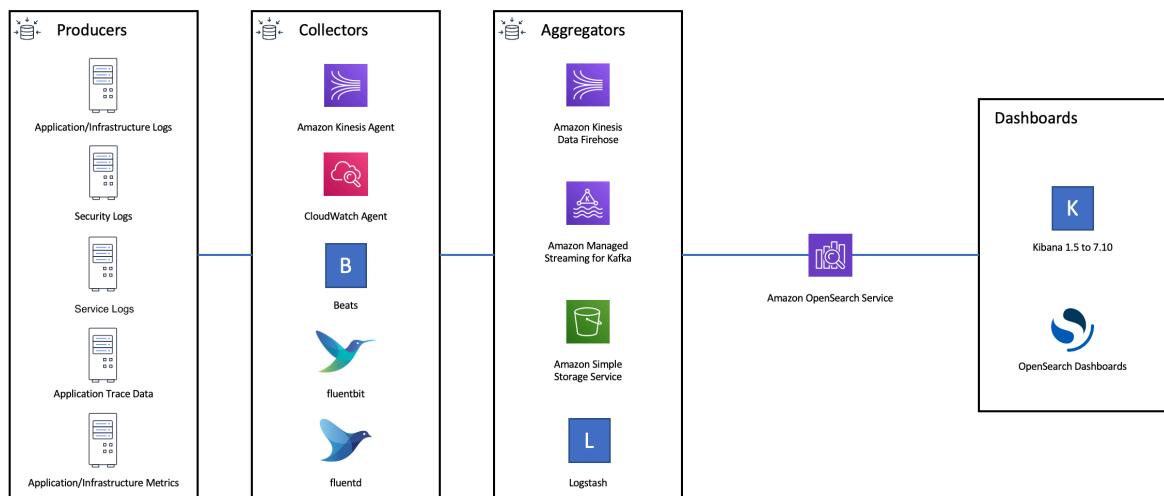
Deploy mission-critical domains across three Availability Zones and two replica shards per index. This configuration lets OpenSearch Service distribute replica shards to different AZs than their corresponding primary shards. There are no cross-AZ data transfer charges for cluster communications between Availability Zones.

Control ingest flow and buffering

We recommend that you limit the overall request count using the `_bulk` API operation. It's more efficient to send one `_bulk` request that contains 5,000 documents than it is to send 5,000 requests that contain a single document.

For optimal operational stability, it's sometimes necessary to limit or even pause the upstream flow of indexing requests. Limiting the rate of index requests is an important mechanism for dealing with unexpected or occasional spikes in requests that might otherwise overwhelm the cluster. Consider building a flow control mechanism into your upstream architecture.

The following diagram shows multiple component options for a log ingest architecture. Configure the aggregation layer to allow sufficient space to buffer incoming data for sudden traffic spikes and brief domain maintenance.



Create mappings for search workloads

For search workloads, create [mappings](#) that define how OpenSearch stores and indexes documents and their fields. Set `dynamic` to `strict` in order to prevent new fields from being added accidentally.

```
PUT my-index
{
  "mappings": {
    "dynamic": "strict",
    "properties": {
      "title": { "type" : "text" },
      "author": { "type" : "integer" },
      "year": { "type" : "text" }
    }
  }
}
```

Use index templates

You can use an [index template](#) as a way to tell OpenSearch how to configure an index when it's created. Configure index templates before creating indexes. Then, when you create an index, it inherits the settings and mappings from the template. You can apply more than one template to a single index, so you can specify settings in one template and mappings in another. This strategy allows one template for common settings across multiple indexes, and separate templates for more specific settings and mappings.

The following settings are helpful to configure in templates:

- Number of primary and replica shards
- Refresh interval (how often to refresh and make recent changes to the index available to search)
- Dynamic mapping control
- Explicit field mappings

The following example template contains each of these settings:

```
{
  "index_patterns": [
    "index-*"
  ],
  "order": 0,
  "settings": {
    "index": {
      "number_of_shards": 3,
      "number_of_replicas": 1,
      "refresh_interval": "60s"
    }
  },
  "mappings": {
    "dynamic": false,
    "properties": {
      "field_name1": {
        "type": "keyword"
      }
    }
  }
}
```

Even if they rarely change, having settings and mappings defined centrally in OpenSearch is simpler to manage than updating multiple upstream clients.

Manage indexes with Index State Management

If you're managing logs or time-series data, we recommend using [Index State Management \(p. 305\)](#) (ISM). ISM lets you automate regular index lifecycle management tasks. With ISM, you can create policies that invoke index alias rollovers, take index snapshots, move indexes between storage tiers, and delete old indexes. You can even use the ISM [rollover](#) operation as an alternative data lifecycle management strategy to avoid shard skew.

First, set up an ISM policy. For example, see [the section called "Sample policies" \(p. 306\)](#). Then, attach the policy to one or more indexes. If you include an [ISM template \(p. 309\)](#) field in the policy, OpenSearch Service automatically applies the policy to any index that matches the specified pattern.

Remove unused indexes

Regularly review the indexes in your cluster and identify any that aren't in use. Take a snapshot of those indexes so that they're stored in S3, and then delete them. When you remove unused indexes, you reduce the shard count, and make it possible to have more balanced storage distribution and resource utilization across nodes. Even when they're idle, indexes consume some resources during internal index maintenance activities.

Rather than manually deleting unused indexes, you can use ISM to automatically take a snapshot and delete indexes after a certain period of time.

Use multiple domains for high availability

To achieve high availability beyond [99.9% uptime](#) across multiple Regions, consider using two domains. For small or slowly changing datasets, you can set up [cross-cluster replication \(p. 315\)](#) to maintain an active-passive model. In this model, only the leader domain is written to, but either domain can be read from. For larger data sets and quickly changing data, configure dual delivery in your ingest pipeline so that all data is written independently to both domains in an active-active model.

Architect your upstream and downstream applications with failover in mind. Make sure to test the failover process along with other disaster recovery processes.

Performance

The following best practices apply to tuning your domains for optimal performance.

Optimize bulk request size and compression

Bulk sizing depends on your data, analysis, and cluster configuration, but a good starting point is 3–5 MiB per bulk request.

Send requests and receive responses from your OpenSearch domains by using [gzip compression \(p. 203\)](#) to reduce the payload size of requests and responses. You can use gzip compression with the [OpenSearch Python client \(p. 204\)](#), or by including the following [headers \(p. 204\)](#) from the client side:

- 'Accept-Encoding': 'gzip'
- 'Content-Encoding': 'gzip'

To optimize your bulk request sizes, start with a bulk request size of 3 MiB. Then, slowly increase the request size until indexing performance stops improving.

Note

To enable gzip compression on domains running Elasticsearch version 6.x, you must set `http_compression.enabled` at the cluster level. This setting is true by default in Elasticsearch versions 7.x and all versions of OpenSearch.

Reduce the size of bulk request responses

To reduce the size of OpenSearch responses, exclude unnecessary fields with the `filter_path` parameter. Make sure that you don't filter out any fields that are required to identify or retry failed requests. For more information and examples, see [the section called "Reducing response size" \(p. 217\)](#).

Tune refresh intervals

OpenSearch indexes have eventual read consistency. A refresh operation makes all the updates that are performed on an index available for search. The default refresh interval is one second, which means that OpenSearch performs a refresh every second while an index is being written to.

The less frequently that you refresh an index (higher refresh interval), the better the overall indexing performance is. The trade-off of increasing the refresh interval is that there's a longer delay between an index update and when the new data is available for search. Set your refresh interval as high as you can tolerate to improve overall performance.

We recommend setting the `refresh_interval` parameter for all of your indexes to 30 seconds or more.

Enable Auto-Tune

[Auto-Tune \(p. 60\)](#) uses performance and usage metrics from your OpenSearch cluster to suggest changes to queue sizes, cache sizes, and Java virtual machine (JVM) settings on your nodes. These optional changes improve cluster speed and stability. You can revert to the default OpenSearch Service settings at any time. Auto-Tune is enabled by default on new domains unless you explicitly disable it.

We recommend that you enable Auto-Tune on all domains, and either set a recurring maintenance window or periodically review its recommendations.

Security

The following best practices apply to securing your domains.

Enable fine-grained access control

[Fine-grained access control \(p. 146\)](#) lets you control who can access certain data within an OpenSearch Service domain. Compared to generalized access control, fine-grained access control gives each cluster, index, document, and field its own specified policy for access. Access criteria can be based on a number of factors, including the role of the person who is requesting access and the action that they intend to perform on the data. For example, you might give one user access to write to an index, and another user access only to read the data on the index without making any changes.

Fine-grained access control allows data with different access requirements to exist in the same storage space without running into security or compliance issues.

We recommend enabling fine-grained access control on your domains.

Deploy domains within a VPC

Placing your OpenSearch Service domain within a virtual private cloud (VPC) helps enable secure communication between OpenSearch Service 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 compared to domains that use public endpoints.

We recommend that you [create your domains within a VPC \(p. 37\)](#).

Apply a restrictive access policy

Even if your domain is deployed within a VPC, it's a best practice to implement security in layers. Make sure to [check the configuration \(p. 21\)](#) of your current access policies.

Apply a restrictive [resource-based access policy \(p. 130\)](#) to your domains and follow the [principle of least privilege](#) when granting access to the configuration API and the OpenSearch API operations. As a general rule, avoid using the anonymous user principal "Principal": {"AWS": "*"} in your access policies.

There are some situations, however, where it's acceptable to use an open access policy, such as when you enable fine-grained access control. An open access policy can enable you to access the domain in cases where request signing is difficult or impossible, such as from certain clients and tools.

Enable encryption at rest

OpenSearch Service domains offer encryption of data at rest to help prevent unauthorized access to your data. Encryption at rest 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 your domain stores sensitive data, [enable encryption of data at rest \(p. 127\)](#).

Enable node-to-node encryption

Node-to-node encryption provides an additional layer of security on top of the default security features within OpenSearch Service. It implements Transport Layer Security (TLS) for all communications between the nodes that are provisioned within OpenSearch. Node-to-node encryption, any data sent to your OpenSearch Service domain over HTTPS remains encrypted in transit while it's being distributed and replicated between nodes.

If your domain stores sensitive data, [enable node-to-node encryption \(p. 129\)](#).

Cost optimization

The following best practices apply to optimizing and saving on your OpenSearch Service costs.

Use the latest generation instance types

OpenSearch Service is always adopting new Amazon EC2 [instances types \(p. 365\)](#) that deliver better performance at a lower cost. We recommend always using the latest generation instances.

Avoid using T2 or t3.small instances for production domains because they can become unstable under sustained heavy load. t3.medium instances are an option for small production workloads (both as data nodes and as dedicated master nodes).

Use UltraWarm and cold storage for time-series log data

If you're using OpenSearch for log analytics, move your data to UltraWarm or cold storage to reduce costs. Use Index State Management (ISM) to migrate data between storage tiers and manage data retention.

[UltraWarm \(p. 286\)](#) provides a cost-effective way to store large amounts of read-only data in OpenSearch Service. UltraWarm uses Amazon S3 for storage, which means that the data is immutable and only one copy is needed. You only pay for storage that's equivalent to the size of the primary shards in your indexes. Latencies for UltraWarm queries grow with the amount of S3 data that's needed to service the query. After the data has been cached on the nodes, queries to UltraWarm indexes perform similar to queries to hot indexes.

[Cold storage \(p. 295\)](#) is also backed by S3. When you need to query cold data, you can selectively attach it to existing UltraWarm nodes. Cold data incurs the same managed storage cost as UltraWarm, but objects in cold storage don't consume UltraWarm node resources. Therefore, cold storage provides a significant amount of storage capacity without impacting UltraWarm node size or count.

UltraWarm becomes cost-effective when you have roughly 2.5 TiB of data in hot storage. Monitor your fill rate and plan to move indexes to UltraWarm before you reach that volume of data.

Review recommendations for Reserved Instances

Consider purchasing [Reserved Instances \(p. 404\)](#) (RIs) after you have a good baseline on your performance and compute consumption. Discounts start at around 30% for no-upfront, 1-year reservations and can increase up to 50% for all-upfront, 3-year commitments.

After you observe stable operation for at least 14 days, review [Reserved Instance recommendations](#) in Cost Explorer. The **Amazon OpenSearch Service** heading displays specific RI purchase recommendations and projected savings.

Sizing Amazon OpenSearch Service domains

There's no perfect method of sizing Amazon OpenSearch Service domains. However, by starting with an understanding of your storage needs, the service, and OpenSearch 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 \(p. 353\)](#)
- [Choosing the number of shards \(p. 355\)](#)
- [Choosing instance types and testing \(p. 355\)](#)

Calculating storage requirements

Most OpenSearch workloads fall into one of two broad categories:

- **Long-lived index:** You write code that processes data into one or more OpenSearch indexes and then updates those indexes periodically as the source data changes. Some common examples are website, document, and ecommerce search.
- **Rolling indexes:** Data continuously flows into a set of temporary indexes, with an indexing period and retention window (such as a set of daily indexes 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 indexes, 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:

- **Number of replicas:** Each replica is a full copy of an index and needs the same amount of disk space. By default, each OpenSearch 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. Use PUT `/my-index/_settings` to update the `number_of_replicas` setting for your index.
- **OpenSearch 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.
- **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.
- **OpenSearch Service overhead:** OpenSearch Service 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 `m6g.xlarge.search` 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.search` 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. This estimate 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 - OpenSearch Service 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 \(p. 355\)](#).

Other storage considerations exist:

- If your minimum storage requirement exceeds 1 PB, see [the section called "Petabyte scale" \(p. 357\)](#).
- If you have rolling indexes and want to use a hot-warm architecture, see [the section called "UltraWarm storage" \(p. 286\)](#).

Choosing the number of shards

After you understand your storage requirements, you can investigate your indexing strategy. By default in OpenSearch Service, each index is divided into five primary shards and one replica (total of 10 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 overall 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 general guideline is to try to keep shard size between 10–30 GiB for workloads where search latency is a key performance objective, and 30–50 GiB for write-heavy workloads such as log analytics.

Large shards can make it difficult for OpenSearch 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 OpenSearch Service 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:

$$\text{(Source Data + Room to Grow) * (1 + Indexing Overhead) / Desired Shard Size} = \text{Approximate Number of Primary Shards}$$

This equation helps compensate for data growth over time. If you expect those same 66 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, $66 * 1.1 / 10 \text{ shards} = 7.26 \text{ GiB per shard}$, 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 `m6g.large.search` 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, and all versions of OpenSearch, have a limit of 1,000 shards per node. To adjust the maximum shards per node, configure the `cluster.max_shards_per_node` setting. For an example, see [Cluster settings](#).

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.search` 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 \(p. 397\)](#) for each instance type map to the amount of CPU and memory that you might need for light workloads. For example, an `m6g.large.search` 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 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 OpenSearch Service pricing](#).

Still, even those resources might be insufficient. Some OpenSearch users report that they need many times those resources to fulfill their requirements. To find the right hardware for your workload, you have to make an educated initial estimate, test with representative workloads, adjust, and test again.

Step 1: Make an initial estimate

To start, we recommend a minimum of three nodes to avoid potential OpenSearch issues, such as a *split brain* state (when a lapse in communication leads to a cluster having two master nodes). If you have three [dedicated master nodes \(p. 358\)](#), we still recommend a minimum of two data nodes for replication.

Step 2: Calculate storage requirements per node

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 `m6g.large.search` instances, where each uses 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.search` instances. If you don't need the fast, local storage, you could also test 18 `r6g.4xlarge.search` instances, each using a 1-TiB EBS storage volume.

If your cluster includes hundreds of terabytes of data, see [the section called "Petabyte scale" \(p. 357\)](#).

Step 3: Perform representative testing

After configuring the cluster, you can [add your indexes \(p. 217\)](#) using the number of shards you calculated earlier, perform some representative client testing using a realistic dataset, and [monitor CloudWatch metrics \(p. 67\)](#) to see how the cluster handles the workload.

Step 4: Succeed or iterate

If performance satisfies your needs, tests succeed, and CloudWatch metrics are normal, the cluster is ready to use. Remember to [set CloudWatch alarms \(p. 361\)](#) 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, OpenSearch automatically rebalances the distribution of shards throughout the cluster.

Because it's 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 \(p. 358\)](#), which improve performance and cluster reliability.

Petabyte scale in Amazon OpenSearch Service

Amazon OpenSearch Service domains offer attached storage of up to 3 PB. You can configure a domain with 200 `i3.16xlarge.search` 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 \(p. 345\)](#). This section discusses considerations for creating domains, costs, storage, and shard size.

While this section frequently references the `i3.16xlarge.search` 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 OpenSearch Service pricing](#) page to ensure that the associated costs match your expectations. Examine [the section called "UltraWarm storage" \(p. 286\)](#) 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 performance benefits when compared to Amazon Elastic Block Store, EBS volumes are not an option when you select these instance types in OpenSearch Service. If you prefer EBS storage, use another instance type, such as `r6.12xlarge.search`.

Shard size and count

A common OpenSearch 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.search` 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 $450 \text{ TB} * 2 * 1.1 / 0.95 = 1.04 \text{ PB}$. For an explanation of this calculation, see [the section called "Calculating storage requirements" \(p. 353\)](#). Although $1.04 \text{ PB} / 15 \text{ TB} = 70$ instances, you might select 90 or more `i3.16xlarge.search` instances to give yourself a storage safety net, deal with node failures, 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. OpenSearch users often rotate indexes 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 some read requests, 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: $450,000 \text{ GB} * 1.1 / 100 \text{ GB per shard} = 4,950$ 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 indexes and only $1/7^{\text{th}}$ or $1/14^{\text{th}}$ 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.

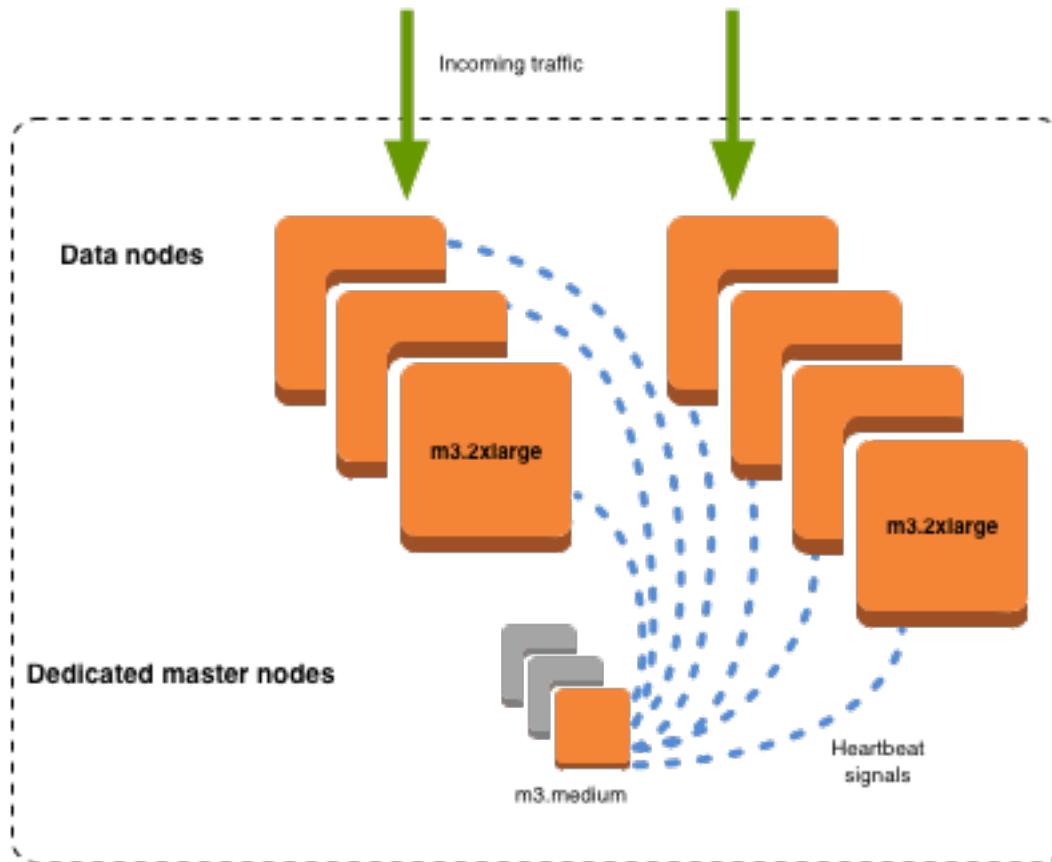
Dedicated master nodes in Amazon OpenSearch Service

Amazon OpenSearch 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. Just like all other node types, you pay an hourly rate for each dedicated master node.

Dedicated master nodes perform the following cluster management tasks:

- Track all nodes in the cluster.
- Track the number of indexes 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 OpenSearch Service domain with 10 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.



Choosing the number of dedicated master nodes

We recommend that you add **three** dedicated master nodes to each production OpenSearch Service domain. Never choose an even number of dedicated master nodes. Consider the following when choosing the number of dedicated master nodes:

- One dedicated master node is explicitly prohibited by OpenSearch Service because you have no backup in the event of a failure. You receive a validation exception if you try to create a domain with only one dedicated master node.
- If you have two dedicated master nodes, your cluster doesn't 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). 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.

- 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.
- Four dedicated master nodes are not better than three and can cause issues if you use [multiple Availability Zones \(p. 34\)](#).
 - 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.

- In a three Availability Zone configuration, two AZs have one dedicated master node, and one AZ has two. If that AZ experiences a disruption, the remaining two AZs don't have the necessary quorum (3) to elect a new master.
- 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 that you pay for four idle nodes. Many users find this level of failover protection excessive.

If a cluster has an even number of master-eligible nodes, OpenSearch and 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 OpenSearch default.

Choosing instance types for dedicated master nodes

Although dedicated master nodes don't process search and query requests, their size is highly correlated with the number of instances, indexes, 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 \(p. 361\)](#) to see if you need to use a larger instance type.

Instance count	Master node RAM size	Maximum supported shard count	Recommended minimum dedicated master instance type
1–10	8 GiB	10K	m5.large.search or m6g.large.search
11–30	16 GiB	30K	c5.2xlarge.search or c6g.2xlarge.search
31–75	32 GiB	75K	c5.4xlarge.search or c6g.4xlarge.search
76 – 125	64 GiB	75K	r5.2xlarge.search or r6g.2xlarge.search
126 – 200	128 GiB	75K	r5.4xlarge.search or r6g.4xlarge.search

- For information about how certain configuration changes can affect dedicated master nodes, see [the section called "Configuration changes" \(p. 22\)](#).
- For clarification on instance count limits, see [the section called "Domain and instance quotas" \(p. 397\)](#).
- For more information about specific instance types, including vCPU, memory, and pricing, see [Amazon OpenSearch Service prices](#).

Recommended CloudWatch alarms for Amazon OpenSearch Service

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 for Amazon OpenSearch Service and how to respond to them.

You can automatically deploy these alarms using AWS CloudFormation. For a sample stack, see the related [GitHub repository](#).

Note

If you deploy the CloudFormation stack, the KMSKeyError and KMSKeyInaccessible alarms will exist in an Insufficient Data state because these metrics only appear if a domain encounters a problem with its encryption key.

For more information about configuring 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. 437) .
ClusterStatus.yellow maximum is ≥ 1 for 1 minute, 5 consecutive times	At least one replica shard is not allocated to a node. See the section called "Yellow cluster status" (p. 439) .
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. 440) . 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	The cluster is blocking write requests. See the section called "ClusterBlockException" (p. 440) .
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. 441) .
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. 437) . For a summary of all automated snapshots and some information about failures, try one of the following requests: GET <code>domain_endpoint/_snapshot/cs-automated/_all</code> GET <code>domain_endpoint/_snapshot/cs-automated-enc/_all</code>
CPUUtilization or WarmCPUUtilization maximum is $\geq 80\%$	100% CPU utilization might occur sometimes, but sustained high usage is problematic. Consider using larger instance types or adding instances.

Alarm	Issue
for 15 minutes, 3 consecutive times	
JVMMemoryPressure maximum is \geq 95% for 1 minute, 3 consecutive times	The cluster could encounter out of memory errors if usage increases. Consider scaling vertically. OpenSearch Service 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.
OldGenJVMMemoryPressure maximum is \geq 80% for 1 minute, 3 consecutive times	
MasterCPUUtilization maximum is \geq 50% for 15 minutes, 3 consecutive times	Consider using larger instance types for your dedicated master nodes (p. 358). Because of their role in cluster stability and blue/green deployments (p. 22), dedicated master nodes should have lower CPU usage than data nodes.
MasterJVMMemoryPressure maximum is \geq 95% for 1 minute, 3 consecutive times	
MasterOldGenJVMMemoryPressure maximum is \geq 80% for 1 minute, 3 consecutive times	
KMSKeyError is \geq 1 for 1 minute, 1 consecutive time	The AWS 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. 127).
KMSKeyInaccessible is \geq 1 for 1 minute, 1 consecutive time	The AWS KMS encryption key that is used to encrypt data at rest in your domain has been deleted or has revoked its grants to OpenSearch Service. You can't recover domains that are in this state. However, 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. 127).
shards.active is \geq 30000 for 1 minute, 1 consecutive time	The total number of active primary and replica shards is greater than 30,000. You might be rotating your indexes too frequently. Consider using ISM to remove indexes once they reach a specific age.
5xx alarms \geq 10% of OpenSearchRequests	One or more data nodes might be overloaded, or requests are failing to complete within the idle timeout period. Consider switching to larger instance types or adding more nodes to the cluster. Confirm that you're following best practices (p. 353) for shard and cluster architecture.
MasterReachableFromNodes is < 1 for 1 day, 1 consecutive time	This alarm indicates that the master node stopped or is unreachable. These failures are usually the result of a network connectivity issue or an AWS dependency problem.
ThreadpoolWriteQueue average is \geq 100 for 1 minute, 1 consecutive time	The cluster is experiencing high indexing concurrency. Review and control indexing requests, or increase cluster resources.

Alarm	Issue
ThreadpoolSearchQueue average is \geq 500 for 1 minute, 1 consecutive time	The cluster is experiencing high search concurrency. Consider scaling your cluster. You can also increase the search queue size, but increasing it excessively can cause out of memory errors.
ThreadpoolSearchQueue maximum is \geq 5000 for 1 minute, 1 consecutive time	
ThreadpoolSearchRejected maximum is \geq 1 for 1 minute, 1 consecutive time	These alarms notify you of domain issues that might impact performance and stability.
ThreadpoolWriteRejected maximum is \geq 1 for 1 minute, 1 consecutive time	

Note

If you just want to view metrics, see [the section called “Monitoring cluster metrics” \(p. 67\)](#).

Other alarms you might consider

Consider configuring the following alarms depending on which OpenSearch Service features you regularly use.

Alarm	Issue
WarmFreeStorageSpace minimum is \leq 10240 for 1 minute, 1 consecutive time	An UltraWarm node in your cluster is down to 10 GiB of free storage space. See the section called “Lack of available storage space” (p. 440) . This value is in MiB, so rather than 10240, we recommend setting it to 10% of the storage space for each UltraWarm node.
HotToWarmMigrationQueueSize is \geq 20 for 1 minute, 3 consecutive times	A large number of indexes are concurrently moving from hot to UltraWarm storage. Consider scaling your cluster.
HotToWarmMigrationSuccessCount is \geq 1 day, 1 consecutive time	Configure this alarm so that you're notified if the HotToWarmMigrationSuccessCount \times latency is greater than 24 hours if you're trying to roll daily indexes.
WarmJVMMemoryPressure maximum is \geq 95% for 1 minute, 3 consecutive times	The cluster could encounter out of memory errors if usage increases. Consider scaling vertically. OpenSearch Service 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.
WarmOldGenJVMMemoryPressure maximum is \geq 80% for 1 minute, 3 consecutive times	

Alarm	Issue
WarmToColdMigrationFailureCount is >= 20 for 1 minute, 3 consecutive times	A high number of indexes are concurrently moving from UltraWarm to cold storage. Consider scaling your cluster.
HotToWarmMigrationFailureCount is >= 1 for 1 minute, 1 consecutive time	Migration 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.
WarmToColdMigrationFailureCount is >= 1 for 1 minute, 1 consecutive time	Migration might fail when attempts to migrate index metadata to cold storage fail. Failures can also happen when the warm index cluster state is being removed.
WarmToColdMigrationLatency is >= 1 day, 1 consecutive time	Configure this alarm so that you're notified if the WarmToColdMigrationSuccessCount x latency is greater than 24 hours if you're trying to roll daily indexes.
AlertingDegraded is >= 1 for 1 minute, 1 consecutive time	Either the alerting index is red, or one or more nodes is not on schedule.
ADPluginUnhealthy is >= 1 for 1 minute, 1 consecutive time	The anomaly detection plugin isn't functioning properly, either because of high failure rates or because one of the indexes being used is red.
AsynchronousSearchFailureCount is >= 1 for 1 minute, 1 consecutive time	At least one asynchronous search failed in the last minute, which likely means the coordinator node failed. The lifecycle of an asynchronous search request is managed solely on the coordinator node, so if the coordinator goes down, the request fails.
AsynchronousSearchStoreHealth is >= 1 for 1 minute, 1 consecutive time	The health of the asynchronous search response store in the persisted index is red. You might be storing large asynchronous responses, which can destabilize a cluster. Try to limit your asynchronous search responses to 10 MB or less.
SQLUnhealthy is >= 1 for 1 minute, 3 consecutive times	The SQL plugin is returning 5xx response codes or passing invalid query DSL to OpenSearch. Troubleshoot the requests that your clients are making to the plugin.
LTRStatus.red is >= 1 for 1 minute, 1 consecutive time	At least one of the indexes needed to run the Learning to Rank plugin has missing primary shards and isn't functional.

General reference for Amazon OpenSearch Service

Amazon OpenSearch Service supports a variety of instances, operations, plugins, and other resources.

Topics

- [Supported instance types in Amazon OpenSearch Service \(p. 365\)](#)
- [Features by engine version \(p. 369\)](#)
- [Plugins by engine version \(p. 371\)](#)
- [Supported operations \(p. 373\)](#)
- [Amazon OpenSearch Service quotas \(p. 397\)](#)
- [Reserved Instances in Amazon OpenSearch Service \(p. 404\)](#)
- [Other supported resources in Amazon OpenSearch Service \(p. 409\)](#)

Supported instance types in Amazon OpenSearch Service

Amazon OpenSearch Service supports the following instance types. Not all Regions support all instance types. For availability details, see [Amazon OpenSearch Service pricing](#).

For information about which instance type is appropriate for your use case, see [the section called "Sizing domains" \(p. 353\)](#), [the section called "EBS volume size quotas" \(p. 399\)](#), and [the section called "Network quotas" \(p. 402\)](#).

Current generation instance types

For the best performance, we recommend that you use the following instance types when you create new OpenSearch Service domains.

Instance type	Instances	Restrictions
C5	c5.large.search c5.xlarge.search c5.2xlarge.search c5.4xlarge.search c5.9xlarge.search c5.18xlarge.search	The C5 instance types require Elasticsearch 5.1 or later or any version of OpenSearch.
C6g	c6g.large.search c6g.xlarge.search c6g.2xlarge.search	The C6g instance types require Elasticsearch 7.9 or later or any version of OpenSearch. • C6g instances are only compatible with other Graviton instance types (M6g, R6g, R6gd). You can't combine Graviton and non-Graviton instances in the same cluster.

Instance type	Instances	Restrictions
	c6g.4xlarge.search c6g.8xlarge.search c6g.12xlarge.search	
I3	i3.large.search i3.xlarge.search i3.2xlarge.search i3.4xlarge.search i3.8xlarge.search i3.16xlarge.search	Note: I3 instance types require Elasticsearch 5.1 or later or any version of OpenSearch, and do not support EBS storage volumes.
M5	m5.large.search m5.xlarge.search m5.2xlarge.search m5.4xlarge.search m5.12xlarge.search	Note: M5 instance types require Elasticsearch 5.1 or later or any version of OpenSearch.
M6g	m6g.large.search m6g.xlarge.search m6g.2xlarge.search m6g.4xlarge.search m6g.8xlarge.search m6g.12xlarge.search	Note: The M6g instance types require Elasticsearch 7.9 or later or any version of OpenSearch. Note: M6g instances are only compatible with other Graviton instance types (C6g, R6g, R6gd). You can't combine Graviton and non-Graviton instances in the same cluster.
R5	r5.large.search r5.xlarge.search r5.2xlarge.search r5.4xlarge.search r5.12xlarge.search	Note: R5 instance types require Elasticsearch 5.1 or later or any version of OpenSearch.

Instance type	Instances	Restrictions
R6g	r6g.large.search r6g.xlarge.search r6g.2xlarge.search r6g.4xlarge.search r6g.8xlarge.search r6g.12xlarge.search	The R6g instance types require Elasticsearch 7.9 or later or any version of OpenSearch. • R6g instances are only compatible with other Graviton instance types (C6g, M6g, R6gd). You can't combine Graviton and non-Graviton instances in the same cluster.
R6gd	r6gd.large.search r6gd.xlarge.search r6gd.2xlarge.search r6gd.4xlarge.search r6gd.8xlarge.search r6gd.12xlarge.search r6gd.16xlarge.search	The R6gd instance types require Elasticsearch 7.9 or later or any version of OpenSearch and do not support EBS storage volumes. • R6gd instances are only compatible with other Graviton instance types (C6g, M6g, R6g). You can't combine Graviton and non-Graviton instances in the same cluster.
T3	t3.small.search t3.medium.search	The T3 instance types require Elasticsearch 5.6 or later or any version of OpenSearch. • You can use the t3.small and t3.medium instance types only if the instance count for your domain is 10 or fewer. • The T3 instance types do not support UltraWarm storage, cold storage, or Auto-Tune.

Previous generation instance types

OpenSearch Service offers previous generation instance types for users who have optimized their applications around them and have yet to upgrade. We encourage you to use current generation instance types to get the best performance, but we continue to support the following previous generation instance types.

Instance type	Instances	Restrictions
C4	c4.large.search c4.xlarge.search c4.2xlarge.search c4.4xlarge.search c4.8xlarge.search	
I2	i2.xlarge.search i2.2xlarge.search	

Instance type	Instances	Restrictions
M3	m3.medium.search m3.large.search m3.xlarge.search m3.2xlarge.search	The M3 instance types do not support encryption of data at rest, fine-grained access control, or cross-cluster search. • The M3 instance types have additional restrictions by OpenSearch version. To learn more, see the section called "Invalid M3 instance type" (p. 444).
M4	m4.large.search m4.xlarge.search m4.2xlarge.search m4.4xlarge.search m4.10xlarge.search	
R3	r3.large.search r3.xlarge.search r3.2xlarge.search r3.4xlarge.search r3.8xlarge.search	The R3 instance types do not support encryption of data at rest or fine-grained access control.
R4	r4.large.search r4.xlarge.search r4.2xlarge.search r4.4xlarge.search r4.8xlarge.search r4.16xlarge.search	
T2	t2.micro.search t2.small.search t2.medium.search	You can use the T2 instance types only if the instance count for your domain is 10 or fewer. • The t2.micro.search instance type supports only Elasticsearch 1.5 and 2.3. • The T2 instance types do not support encryption of data at rest, fine-grained access control, UltraWarm storage, cold storage, cross-cluster search, or Auto-Tune.

Tip

We often recommend different instance types for [dedicated master nodes \(p. 358\)](#) and data nodes.

Features by engine version

Many OpenSearch Service features have a minimum OpenSearch version requirement or legacy Elasticsearch OSS version requirement. If you meet the minimum version for a feature, but the feature isn't available on your domain, update your domain's [service software \(p. 29\)](#).

Feature	Minimum required OpenSearch version	Minimum required Elasticsearch version
VPC support	1.0	1.0
Require HTTPS for all traffic to the domain		
Multi-AZ support		
Dedicated master nodes		
Custom packages		
Custom endpoints		
Slow log publishing		
Error log publishing	1.0	5.1
Encryption of data at rest		
Cognito authentication for OpenSearch Dashboards		
In-place upgrades		
Curator support	Not included	5.1
Hourly automated snapshots	1.0	5.3
Node-to-node encryption	1.0	6.0

Feature	Minimum required OpenSearch version	Minimum required Elasticsearch version
Java high-level REST client support		
HTTP request and response compression		
Alerting	1.0	6.2
SQL	1.0	6.5
Cross-cluster search	1.0	6.7
Fine-grained access control		
SAML authentication for OpenSearch Dashboards		
Auto-Tune		
Remote reindex		
UltraWarm	1.0	6.8
Index State Management		
k-NN by Euclidean distance	1.0	7.1
Anomaly Detection	1.0	7.4
k-NN by cosine similarity	1.0	7.7
Learning to Rank		
Piped processing language	1.0	7.9
OpenSearch Dashboards reports		

Feature	Minimum required OpenSearch version	Minimum required Elasticsearch version
OpenSearch Dashboards		
Trace Analytics		
ARM-based Graviton instances		
Cold storage	1.0	7.10
Hamming distance, L1 Norm distance, and Painless scripting for k-NN		
Asynchronous search	1.0	Not included
Index transforms		
Cross-cluster replication	1.1	7.10
ML Commons	1.3	Not included

For information about plugins, which enable some of these features and additional functionality, see [the section called “Plugins by engine version” \(p. 371\)](#). For information about the OpenSearch API for each version, see [the section called “Supported operations” \(p. 373\)](#).

Plugins by engine version

Amazon OpenSearch Service domains come prepackaged with plugins from the OpenSearch community. The service automatically deploys and manages plugins for you, but it deploys different plugins depending on the version of OpenSearch or legacy Elasticsearch OSS you choose for your domain.

The following table lists plugins by OpenSearch version, as well as compatible versions of legacy Elasticsearch OSS. It only includes plugins that you might interact with—it’s not comprehensive. OpenSearch Service uses additional plugins to enable core service functionality, such as the S3 Repository plugin for snapshots and the [OpenSearch Performance Analyzer](#) plugin for optimization and monitoring. For a complete list of all plugins running on your domain, make the following request:

```
GET _cat/plugins?v
```

Plugin	Minimum required OpenSearch version	Minimum required Elasticsearch version
ICU Analysis	1.0	Included on all domains

Plugin	Minimum required OpenSearch version	Minimum required Elasticsearch version
Japanese (kuromoji) Analysis		
Phonetic Analysis	1.0	2.3
Seunjeon Korean Analysis	1.0	5.1
Smart Chinese Analysis		
Stempel Polish Analysis		
Ingest Attachment Processor		
Ingest User Agent Processor		
Mapper Murmur3		
Mapper Size	1.0	5.3
Ukrainian Analysis		
OpenSearch alerting (p. 328)	1.0	6.2
OpenSearch SQL (p. 245)	1.0	6.5
OpenSearch security (p. 146)	1.0	6.7
OpenSearch Index State Management (p. 305)	1.0	6.8
OpenSearch k-NN (p. 248)	1.0	7.1
OpenSearch anomaly detection (p. 331)	1.0	7.4
IK (Chinese) Analysis	1.0	7.7

Plugin	Minimum required OpenSearch version	Minimum required Elasticsearch version
Vietnamese Analysis		
Thai analysis		
Learning to Rank (p. 257)		
OpenSearch asynchronous search (p. 276)	1.0	7.10
OpenSearch cross-cluster replication (p. 315)	1.1	7.10
OpenSearch observability (p. 336)	1.2	Not supported
ML Commons	1.3	Not supported

Supported operations

OpenSearch Service supports many versions of OpenSearch and legacy Elasticsearch OSS. The following sections show the operations that OpenSearch Service supports for each version.

Topics

- [Notable API differences \(p. 374\)](#)
- [OpenSearch version 2.3 \(p. 375\)](#)
- [OpenSearch version 1.3 \(p. 376\)](#)
- [OpenSearch version 1.2 \(p. 377\)](#)
- [OpenSearch version 1.1 \(p. 378\)](#)
- [OpenSearch version 1.0 \(p. 379\)](#)
- [Elasticsearch version 7.10 \(p. 380\)](#)
- [Elasticsearch version 7.9 \(p. 381\)](#)
- [Elasticsearch version 7.8 \(p. 382\)](#)
- [Elasticsearch version 7.7 \(p. 383\)](#)
- [Elasticsearch version 7.4 \(p. 384\)](#)
- [Elasticsearch version 7.1 \(p. 385\)](#)
- [Elasticsearch version 6.8 \(p. 386\)](#)
- [Elasticsearch version 6.7 \(p. 387\)](#)
- [Elasticsearch version 6.5 \(p. 388\)](#)
- [Elasticsearch version 6.4 \(p. 389\)](#)
- [Elasticsearch version 6.3 \(p. 390\)](#)
- [Elasticsearch version 6.2 \(p. 391\)](#)
- [Elasticsearch version 6.0 \(p. 392\)](#)
- [Elasticsearch version 5.6 \(p. 392\)](#)
- [Elasticsearch version 5.5 \(p. 393\)](#)
- [Elasticsearch version 5.3 \(p. 394\)](#)

- [Elasticsearch version 5.1 \(p. 395\)](#)
- [Elasticsearch version 2.3 \(p. 396\)](#)
- [Elasticsearch version 1.5 \(p. 396\)](#)

Notable API differences

Settings and statistics

OpenSearch Service 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 OpenSearch Service domains supported only the HTTP PUT method, not the GET method. OpenSearch and later versions of Elasticsearch support the GET method, as shown in the following example:

```
GET https://domain-name.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_primarirecoveries": "4"
        }
      }
    },
    "indices": {
```

```

        "recovery": {
            "max_bytper_sec": "40mb"
        }
    }
}

```

If you compare responses from an open source OpenSearch cluster and OpenSearch Service for certain settings and statistics APIs, you might notice missing fields. OpenSearch Service 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 or OpenSearch versions, make the following request before starting the shrink operation:

```
PUT https://domain-name.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-name.region.es.amazonaws.com/source-index/_settings
{
  "settings": {
    "index.routing.allocation.require._name": null,
    "index.blocks.read_only": false
  }
}

PUT https://domain-name.region.es.amazonaws.com/shrunken-index/_settings
{
  "settings": {
    "index.routing.allocation.require._name": null,
    "index.blocks.read_only": false
  }
}
```

OpenSearch version 2.3

For OpenSearch 2.3, OpenSearch Service supports the following operations. For information about most of the operations, see the [OpenSearch REST API reference](#), or the API reference for the specific plugin.

- | | | |
|---|---|--|
| <ul style="list-style-type: none"> All operations in the index path (such as <code>/index-name/_forcemerge</code>, <code>/index-name/_update/id</code>, and <code>/index-name/_close</code>) <code>/_alias</code> | <ul style="list-style-type: none"> <code>/_delete_by_query</code>¹ <code>/_explain</code> <code>/_field_caps</code> <code>/_field_stats</code> <code>/_flush</code> | <ul style="list-style-type: none"> <code>/_refresh</code> <code>/_reindex</code>¹ <code>/_render</code> <code>/_resolve/index</code> <code>/_rollover</code> |
|---|---|--|

<ul style="list-style-type: none"> • <code>/_aliases</code> • <code>/_all</code> • <code>/_analyze</code> • <code>/_bulk</code> • <code>/_cat (except /_cat/nodeattrs)</code> • <code>/_cluster/allocation/explain</code> • <code>/_cluster/health</code> • <code>/_cluster/pending_tasks</code> • <code>_cluster/settings for several properties⁴:</code> <ul style="list-style-type: none"> • <code>action.auto_create_index</code> • <code>action.search.shard_count.limit</code>⁵ • <code>indices.breaker.fielddata.limit</code> • <code>indices.breaker.request.limit</code> • <code>indices.breaker.total.limit</code> • <code>cluster.max_shards_per_node</code> • <code>/_cluster/state</code> • <code>/_cluster/stats</code> • <code>/_count</code> • <code>/_dashboards</code> 	<ul style="list-style-type: none"> • <code>/_ingest/pipeline</code> • <code>/_ltr</code> • <code>/_mapping</code> • <code>/_mget</code> • <code>/_msearch</code> • <code>/_mtermvectors</code> • <code>/_nodes</code> • <code>/_plugins/_asynchronous_search</code> • <code>/_plugins/_alerting</code> • <code>/_plugins/_anomaly_detection</code> • <code>/_plugins/_ism</code> • <code>/_plugins/_ml</code> • <code>/_plugins/_ppl</code> • <code>/_plugins/_security</code> • <code>/_plugins/_sql</code> • <code>/_percolate</code> • <code>/_rank_eval</code> 	<ul style="list-style-type: none"> • <code>/_scripts³</code> • <code>/_search²</code> • <code>/_search profile</code> • <code>/_shard_stores</code> • <code>/_shrink⁵</code> • <code>/_snapshot</code> • <code>/_split</code> • <code>/_stats</code> • <code>/_status</code> • <code>/_tasks</code> • <code>/_template</code> • <code>/_update_by_query¹</code> • <code>/_validate</code>
--	---	--

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic OpenSearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

OpenSearch version 1.3

For OpenSearch 1.3, OpenSearch Service supports the following operations. For information about most of the operations, see the [OpenSearch REST API reference](#), or the API reference for the specific plugin.

<ul style="list-style-type: none"> • All operations in the index path (such as <code>/index-name/_forcemerge</code>, <code>/index-name/_update</code>/<i>id</i>, and <code>/index-name/_close</code>) • <code>/_alias</code> • <code>/_aliases</code> 	<ul style="list-style-type: none"> • <code>/_delete_by_query¹</code> • <code>/_explain</code> • <code>/_field_caps</code> • <code>/_field_stats</code> • <code>/_flush</code> • <code>/_ingest/pipeline</code> 	<ul style="list-style-type: none"> • <code>/_refresh</code> • <code>/_reindex¹</code> • <code>/_render</code> • <code>/_resolve/index</code> • <code>/_rollover</code> • <code>/_scripts³</code>
--	---	--

<ul style="list-style-type: none"> • <code>/_all</code> • <code>/_analyze</code> • <code>/_bulk</code> • <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) • <code>/_cluster/allocation/explain</code> • <code>/_cluster/health</code> • <code>/_cluster/pending_tasks</code> • <code>/_cluster/settings</code> for several properties⁴: <ul style="list-style-type: none"> • <code>action.auto_create_index</code> • <code>action.search.shard_count.lim</code>¹<code>tplugins/_ism</code> • <code>indices.breaker.fielddata.lim</code>¹<code>tplugins/_ml</code> • <code>indices.breaker.request.lim</code>¹<code>t/plugins/_ppl</code> • <code>indices.breaker.total.lim</code>¹<code>t/plugins/_ppl</code> • <code>cluster.max_shards_per_node</code>¹<code>t/plugins/_sql</code> • <code>/_cluster/state</code> • <code>/_cluster/stats</code> • <code>/_count</code> • <code>/_dashboards</code> 	<ul style="list-style-type: none"> • <code>/_ltr</code> • <code>/_mapping</code> • <code>/_mget</code> • <code>/_msearch</code> • <code>/_mtermvectors</code> • <code>/_nodes</code> • <code>/_plugins/_asynchronous_search</code> • <code>/_plugins/_alerting</code> • <code>/_plugins/_anomaly_detection</code> • <code>/_percolate</code> • <code>/_rank_eval</code> 	<ul style="list-style-type: none"> • <code>/_search</code>² • <code>/_search profile</code> • <code>/_shard_stores</code> • <code>/_shrink</code>⁵ • <code>/_snapshot</code> • <code>/_split</code> • <code>/_stats</code> • <code>/_status</code> • <code>/_tasks</code> • <code>/_template</code> • <code>/_update_by_query</code>¹ • <code>/_validate</code>
---	--	--

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic OpenSearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

OpenSearch version 1.2

For OpenSearch 1.2, OpenSearch Service supports the following operations. For information about most of the operations, see the [OpenSearch REST API reference](#), or the API reference for the specific plugin.

<ul style="list-style-type: none"> • All operations in the index path (such as <code>/index-name/_forcemerge</code>, <code>/index-name/_update</code>/<code>id</code>, and <code>/index-name/_close</code>) • <code>/_alias</code> • <code>/_aliases</code> • <code>/_all</code> 	<ul style="list-style-type: none"> • <code>/_delete_by_query</code>¹ • <code>/_explain</code> • <code>/_field_caps</code> • <code>/_field_stats</code> • <code>/_flush</code> • <code>/_ingest/pipeline</code> • <code>/_ltr</code> 	<ul style="list-style-type: none"> • <code>/_refresh</code> • <code>/_reindex</code>¹ • <code>/_render</code> • <code>/_resolve/index</code> • <code>/_rollover</code> • <code>/_scripts</code>³ • <code>/_search</code>²
--	---	---

• /_analyze	• /_mapping	• /_search profile
• /_bulk	• /_mget	• /_shard_stores
• /_cat (except /_cat/nodeattrs)	• /_msearch	• /_shrink ⁵
• /_cluster/allocation/explain	• /_mtermvectors	• /_snapshot
• /_cluster/health	• /_nodes	• /_split
• /_cluster/pending_tasks	• /_plugins/_asynchronous_search	• /_stats
• /_cluster/settings for several properties ⁴ :	• /_plugins/_alerting	• /_status
• action.auto_create_index	• /_plugins/_anomaly_detection	• /_tasks
• action.search.shard_count.limit	• /_plugins/_ism	• /_template
• indices.breaker.fielddata.limit	• /_plugins/_ppl	• /_update_by_query ¹
• indices.breaker.request.limit	• /_plugins/_security	• /_validate
• indices.breaker.total.limit	• /_plugins/_sql	
• cluster.max_shards_per_node	• /_percolate	
• /_cluster/state	• /_rank_eval	
• /_cluster/stats		
• /_count		
• /_dashboards		

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic OpenSearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

OpenSearch version 1.1

For OpenSearch 1.1, OpenSearch Service supports the following operations. For information about most of the operations, see the [OpenSearch REST API reference](#), or the API reference for the specific plugin.

• All operations in the index path (such as /_index-name/_forcemerge , /_index-name/_update / /id , and /_index-name/_close)	• /_delete_by_query ¹	• /_refresh
• /_alias	• /_explain	• /_reindex ¹
• /_aliases	• /_field_caps	• /_render
• /_all	• /_field_stats	• /_resolve/index
• /_analyze	• /_flush	• /_rollover
	• /_ingest/pipeline	• /_scripts ³
	• /_ltr	• /_search ²
	• /_mapping	• /_search_profile

<ul style="list-style-type: none"> • <code>/_bulk</code> • <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) • <code>/_cluster/allocation/explain</code> • <code>/_cluster/health</code> • <code>/_cluster/pending_tasks</code> • <code>/_cluster/settings</code> for several properties⁴: <ul style="list-style-type: none"> • <code>action.auto_create_index</code> • <code>action.search.shard_count.limit</code> • <code>indices.breaker.fielddata.limit</code> • <code>indices.breaker.request.limit</code> • <code>indices.breaker.total.limit</code> • <code>cluster.max_shards_per_node</code> • <code>/_cluster/state</code> • <code>/_cluster/stats</code> • <code>/_count</code> • <code>/_dashboards</code> 	<ul style="list-style-type: none"> • <code>/_mget</code> • <code>/_msearch</code> • <code>/_mtermvectors</code> • <code>/_nodes</code> • <code>/_plugins/_asynchronous_search</code> • <code>/_plugins/_alerting</code> • <code>/_plugins/_anomaly_detection</code> • <code>/_plugins/_ism</code> • <code>/_plugins/_ppl</code> • <code>/_plugins/_security</code> • <code>/_plugins/_sql</code> • <code>/_plugins/_transforms</code> • <code>/_percolate</code> • <code>/_rank_eval</code> 	<ul style="list-style-type: none"> • <code>/_shard_stores</code> • <code>/_shrink</code>⁵ • <code>/_snapshot</code> • <code>/_split</code> • <code>/_stats</code> • <code>/_status</code> • <code>/_tasks</code> • <code>/_template</code> • <code>/_update_by_query</code>¹ • <code>/_validate</code>
--	---	--

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic OpenSearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

OpenSearch version 1.0

For OpenSearch 1.0, OpenSearch Service supports the following operations. For information about most of the operations, see the [OpenSearch REST API reference](#), or the API reference for the specific plugin.

<ul style="list-style-type: none"> • All operations in the index path (such as <code>/index-name/_forcemerge</code>, <code>/index-name/_update</code>/<code>id</code>, and <code>/index-name/_close</code>) • <code>/_alias</code> • <code>/_aliases</code> • <code>/_all</code> • <code>/_analyze</code> • <code>/_bulk</code> 	<ul style="list-style-type: none"> • <code>/_delete_by_query</code>¹ • <code>/_explain</code> • <code>/_field_caps</code> • <code>/_field_stats</code> • <code>/_flush</code> • <code>/_ingest/pipeline</code> • <code>/_ltr</code> • <code>/_mapping</code> • <code>/_mget</code> 	<ul style="list-style-type: none"> • <code>/_refresh</code>¹ • <code>/_reindex</code>¹ • <code>/_render</code> • <code>/_resolve/index</code> • <code>/_rollover</code> • <code>/_scripts</code>³ • <code>/_search</code>² • <code>/_search_profile</code> • <code>/_shard_stores</code>
---	--	---

<ul style="list-style-type: none"> • <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) • <code>/_cluster/allocation/explain</code> • <code>/_cluster/health</code> • <code>/_cluster/pending_tasks</code> • <code>/_cluster/settings</code> for several properties⁴: <ul style="list-style-type: none"> • <code>action.auto_create_index</code> • <code>action.search.shard_count.limit</code> • <code>indices.breaker.fielddata.limit</code> • <code>indices.breaker.request.limit</code> • <code>indices.breaker.total.limit</code> • <code>cluster.max_shards_per_node</code> • <code>/_cluster/state</code> • <code>/_cluster/stats</code> • <code>/_count</code> • <code>/_dashboards</code> 	<ul style="list-style-type: none"> • <code>/_msearch</code> • <code>/_mtermvectors</code> • <code>/_nodes</code> • <code>/_plugins/_asynchronous_search</code> • <code>/_plugins/_alerting</code> • <code>/_plugins/_anomaly_detection</code> • <code>/_plugins/_ism</code> • <code>/_plugins/_ppl</code> • <code>/_plugins/_security</code> • <code>/_plugins/_sql</code> • <code>/_plugins/_transforms</code> • <code>/_percolate</code> • <code>/_rank_eval</code> 	<ul style="list-style-type: none"> • <code>/_shrink</code>⁵ • <code>/_snapshot</code> • <code>/_split</code> • <code>/_stats</code> • <code>/_status</code> • <code>/_tasks</code> • <code>/_template</code> • <code>/_update_by_query</code>¹ • <code>/_validate</code>
--	--	---

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic OpenSearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 7.10

For Elasticsearch 7.10, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> • All operations in the index path (such as <code>/index-name/_forcemerge</code>, <code>/index-name/_update</code>/<code>id</code>, and <code>/index-name/_close</code>) • <code>/_alias</code> • <code>/_aliases</code> • <code>/_all</code> • <code>/_analyze</code> • <code>/_bulk</code> • <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) 	<ul style="list-style-type: none"> • <code>/_delete_by_query</code>¹ • <code>/_explain</code> • <code>/_field_caps</code> • <code>/_field_stats</code> • <code>/_flush</code> • <code>/_index_template</code>⁶ • <code>/_ingest/pipeline</code> • <code>/_index_template</code> • <code>/_ltr</code> • <code>/_mapping</code> • <code>/_mget</code> 	<ul style="list-style-type: none"> • <code>/_refresh</code> • <code>/_reindex</code>¹ • <code>/_render</code> • <code>/_resolve/index</code> • <code>/_rollover</code> • <code>/_scripts</code>³ • <code>/_search</code>² • <code>/_search_profile</code> • <code>/_shard_stores</code> • <code>/_shrink</code>⁵ • <code>/_snapshot</code>
---	--	---

• <code>/_cluster/allocation/explain</code>	• <code>/_msearch</code>	• <code>/_split</code>
• <code>/_cluster/health</code>	• <code>/_mtermvectors</code>	• <code>/_stats</code>
• <code>/_cluster/pending_tasks</code>	• <code>/_nodes</code>	• <code>/_status</code>
• <code>/_cluster/settings</code> for several properties ⁴ :	• <code>/_opendistro/_alerting</code>	• <code>/_tasks</code>
• <code>action.auto_create_index</code>	• <code>/_opendistro/_asynchronous_search</code>	• <code>/_template⁶</code>
• <code>action.search.shard_count.limit</code>	• <code>/_opendistro/_ism/_topendistro/anomaly_detection</code>	• <code>/_update_by_query¹</code>
• <code>indices.breaker.fielddata.limit</code>	• <code>/_opendistro/_ism</code>	• <code>/_validate</code>
• <code>indices.breaker.request.limit</code>	• <code>/_opendistro/_ism/_total</code>	
• <code>indices.breaker.total.limit</code>	• <code>/_opendistro/_ppl</code>	
• <code>cluster.max_shards_per_node</code>	• <code>/_opendistro/_security</code>	
• <code>/_cluster/state</code>	• <code>/_opendistro/_sql</code>	
• <code>/_cluster/stats</code>	• <code>/_percolate</code>	
• <code>/_count</code>	• <code>/_plugin/kibana</code>	
	• <code>/_plugins/_replication</code>	
	• <code>/_rank_eval</code>	

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).
6. Legacy index templates (`_template`) were replaced by composable templates (`_index_template`) starting with Elasticsearch 7.8. Composable templates take precedence over legacy templates. If no composable template matches a given index, a legacy template can still match and be applied. The `_template` operation still works on OpenSearch and later versions of Elasticsearch OSS, but GET calls to the two template types return different results.

Elasticsearch version 7.9

For Elasticsearch 7.9, OpenSearch Service supports the following operations.

• All operations in the index path (such as <code>/index-name/_forcemerge</code> , <code>/index-name/_update/</code> <i>id</i> , and <code>/index-name/_close</code>)	• <code>/_delete_by_query¹</code>	• <code>/_refresh</code>
• <code>/_alias</code>	• <code>/_explain</code>	• <code>/_reindex¹</code>

• /_aliases	• /_index_template ⁶	• /_scripts ³
• /_all	• /_ingest/pipeline	• /_search ²
• /_analyze	• /_ltr	• /_search_profile
• /_bulk	• /_mapping	• /_shard_stores
• /_cat (except /_cat/nodeattrs)	• /_mget	• /_shrink ⁵
• /_cluster/allocation/explain	• /_msearch	• /_snapshot
• /_cluster/health	• /_mtermvectors	• /_split
• /_cluster/pending_tasks	• /_nodes	• /_stats
• /_cluster/settings for several properties ⁴ :	• /_opendistro/_alerting	• /_status
• action.auto_create_index	• /_opendistro/_anomaly_detection	• /_tasks
• action.search.shard_count.limit	• /_opendistro/_ism	• /_template ⁶
• indices.breaker.fielddata.limit	• /_opendistro/_ppl	• /_update_by_query ¹
• indices.breaker.request.limit	• /_opendistro/_security	• /_validate
• indices.breaker.total.limit	• /_opendistro/_sql	
• cluster.max_shards_per_node	• /_percolate	
• /_cluster/state	• /_plugin/kibana	
• /_cluster/stats	• /_rank_eval	
• /_count		

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic OpenSearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).
6. Legacy index templates (_template) were replaced by composable templates (_index_template) starting with Elasticsearch 7.8. Composable templates take precedence over legacy templates. If no composable template matches a given index, a legacy template can still match and be applied. The _template operation still works on OpenSearch and later versions of Elasticsearch OSS, but GET calls to the two template types return different results.

Elasticsearch version 7.8

For Elasticsearch 7.8, OpenSearch Service supports the following operations.

• All operations in the index path (such as / <i>index-name</i> /_forcemerge, / <i>index-name</i> /	• /_cluster/state	• /_refresh
	• /_cluster/stats	• /_reindex ¹
	• /_count	• /_render

<ul style="list-style-type: none"> • update/<i>id</i>, and /<i>index-name/_close</i> • /_alias • /_aliases • /_all • /_analyze • /_bulk • /_cat (except /_cat/nodeattrs) • /_cluster/allocation/explain • /_cluster/health • /_cluster/pending_tasks • /_cluster/settings for several properties⁴: <ul style="list-style-type: none"> • 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 	<ul style="list-style-type: none"> • /_delete_by_query¹ • /_explain • /_field_caps • /_field_stats • /_flush • /_index_template⁶ • /_ingest/pipeline • /_ltr • /_mapping • /_mget • /_msearch • /_mtermvectors • /_nodes • /_opendistro/alerting • /_opendistro/anomaly_detection • /_opendistro/_ism • /_opendistro/_security • /_opendistro/_sql • /_percolate • /_plugin/kibana • /_rank_eval 	<ul style="list-style-type: none"> • /_rollover • /_scripts³ • /_search² • /_search_profile • /_shard_stores • /_shrink⁵ • /_snapshot • /_split • /_stats • /_status • /_tasks • /_template⁶ • /_update_by_query¹ • /_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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).
6. Legacy index templates (_template) were replaced by composable templates (_index_template) starting with Elasticsearch 7.8. Composable templates take precedence over legacy templates. If no composable template matches a given index, a legacy template can still match and be applied. The _template operation still works on OpenSearch and later versions of Elasticsearch OSS, but GET calls to the two template types return different results.

Elasticsearch version 7.7

For Elasticsearch 7.7, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> All operations in the index path (such as <code>/index-name/_forcemerge</code>, <code>/index-name/_update</code>/<i>id</i>, and <code>/index-name/_close</code>) <code>/_alias</code> <code>/_aliases</code> <code>/_all</code> <code>/_analyze</code> <code>/_bulk</code> <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) <code>/_cluster/allocation/explain</code> <code>/_cluster/health</code> <code>/_cluster/pending_tasks</code> <code>/_cluster/settings</code> for several properties⁴: <ul style="list-style-type: none"> <code>action.auto_create_index</code> <code>action.search.shard_count.limit</code> <code>indices.breaker.fielddata.limit</code> <code>indices.breaker.request.limit</code> <code>indices.breaker.total.limit</code> <code>cluster.max_shards_per_node</code> 	<ul style="list-style-type: none"> <code>/_cluster/state</code> <code>/_cluster/stats</code> <code>/_count</code> <code>/_delete_by_query</code>¹ <code>/_explain</code> <code>/_field_caps</code> <code>/_field_stats</code> <code>/_flush</code> <code>/_ingest/pipeline</code> <code>/_ltr</code> <code>/_mapping</code> <code>/_mget</code> <code>/_msearch</code> <code>/_mtermvectors</code> <code>/_nodes</code> <code>/_opendistro/_alerting</code> <code>/_opendistro/_anomaly_detection</code> <code>/_opendistro/_ism</code> <code>/_opendistro/_security</code> <code>/_opendistro/_sql</code> <code>/_percolate</code> <code>/_plugin/kibana</code> <code>/_rank_eval</code> 	<ul style="list-style-type: none"> <code>/_refresh</code> <code>/_reindex</code>¹ <code>/_render</code> <code>/_rollover</code> <code>/_scripts</code>³ <code>/_search</code>² <code>/_search_profile</code> <code>/_shard_stores</code> <code>/_shrink</code>⁵ <code>/_snapshot</code> <code>/_split</code> <code>/_stats</code> <code>/_status</code> <code>/_tasks</code> <code>/_template</code> <code>/_update_by_query</code>¹ <code>/_validate</code>
---	---	--

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 7.4

For Elasticsearch 7.4, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> All operations in the index path (such as <code>/index-name/_forcemerge</code>, <code>/index-name/_update</code>, <code>/index-name/_id</code>, and <code>/index-name/_close</code>) <code>/_alias</code> <code>/_aliases</code> <code>/_all</code> <code>/_analyze</code> <code>/_bulk</code> <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) <code>/_cluster/allocation/explain</code> <code>/_cluster/health</code> <code>/_cluster/pending_tasks</code> <code>/_cluster/settings</code> for several properties⁴: <ul style="list-style-type: none"> <code>action.auto_create_index</code> <code>action.search.shard_count.limit</code> <code>indices.breaker.fielddata.limit</code> <code>indices.breaker.request.limit</code> <code>indices.breaker.total.limit</code> <code>cluster.max_shards_per_node</code> 	<ul style="list-style-type: none"> <code>/_cluster/state</code> <code>/_cluster/stats</code> <code>/_count</code> <code>/_delete_by_query</code>¹ <code>/_explain</code> <code>/_field_caps</code> <code>/_field_stats</code> <code>/_flush</code> <code>/_ingest/pipeline</code> <code>/_mapping</code> <code>/_mget</code> <code>/_msearch</code> <code>/_mtermvectors</code> <code>/_nodes</code> <code>/_opendistro/_alerting</code> <code>/_opendistro/_anomaly_detection</code> <code>/_opendistro/_ism</code> <code>/_opendistro/_security</code> <code>/_opendistro/_sql</code> <code>/_percolate</code> <code>/_plugin/kibana</code> <code>/_rank_eval</code> 	<ul style="list-style-type: none"> <code>/_refresh</code> <code>/_reindex</code>¹ <code>/_render</code> <code>/_rollover</code> <code>/_scripts</code>³ <code>/_search</code>² <code>/_search_profile</code> <code>/_shard_stores</code> <code>/_shrink</code>⁵ <code>/_snapshot</code> <code>/_split</code> <code>/_stats</code> <code>/_status</code> <code>/_tasks</code> <code>/_template</code> <code>/_update_by_query</code>¹ <code>/_validate</code>
---	---	--

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 7.1

For Elasticsearch 7.1, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> All operations in the index path (such as <code>/index-name/_forcemerge</code>, <code>/index-name/_update</code>, <code>/index-name/_id</code>, and <code>/index-name/_close</code>) 	<ul style="list-style-type: none"> <code>/_cluster/state</code> <code>/_cluster/stats</code> 	<ul style="list-style-type: none"> <code>/_refresh</code> <code>/_reindex</code>¹
--	--	--

<pre>_forcemerge and /index-name/ update/<i>id</i>) 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 properties⁴: • 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</pre>	<ul style="list-style-type: none"> • /_count • /_delete_by_query¹ • /_explain • /_field_caps • /_field_stats • /_flush • /_ingest/pipeline • /_mapping • /_mget • /_msearch • /_mtermvectors • /_nodes • /_opendistro/ _alerting • /_opendistro/_ism • /_opendistro/ _security • /_opendistro/_sql • /_percolate • /_plugin/kibana • /_rank_eval 	<ul style="list-style-type: none"> • /_render • /_rollover • /_scripts³ • /_search² • /_search_profile • /_shard_stores • /_shrink⁵ • /_snapshot • /_split • /_stats • /_status • /_tasks • /_template • /_update_by_query¹ • /_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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 6.8

For Elasticsearch 6.8, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> • All operations in the index path (such as /<i>index-name</i>/_forcemerge and /<i>index-name</i>/update/<i>id</i>) except /<i>index-name</i>_close • /_alias • /_aliases 	<ul style="list-style-type: none"> • /_cluster/state • /_cluster/stats • /_count • /_delete_by_query¹ • /_explain • /_field_caps 	<ul style="list-style-type: none"> • /_refresh • /_reindex¹ • /_render • /_rollover • /_scripts³ • /_search²
---	---	---

<ul style="list-style-type: none"> • <code>/_all</code> • <code>/_analyze</code> • <code>/_bulk</code> • <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) • <code>/_cluster/allocation/explain</code> • <code>/_cluster/health</code> • <code>/_cluster/pending_tasks</code> • <code>/_cluster/settings</code> for several properties⁴: <ul style="list-style-type: none"> • <code>action.auto_create_index</code> • <code>action.search.shard_count.limit</code> • <code>indices.breaker.fielddata.limit</code> • <code>indices.breaker.request.limit</code> • <code>indices.breaker.total.limit</code> • <code>cluster.max_shards_per_node</code> • <code>cluster.blocks.read_only</code> 	<ul style="list-style-type: none"> • <code>/_field_stats</code> • <code>/_flush</code> • <code>/_ingest/pipeline</code> • <code>/_mapping</code> • <code>/_mget</code> • <code>/_msearch</code> • <code>/_mtermvectors</code> • <code>/_nodes</code> • <code>/_opendistro/_alerting</code> • <code>/_opendistro/_ism</code> • <code>/_percolate</code> • <code>/_plugin/kibana</code> • <code>/_rank_eval</code> 	<ul style="list-style-type: none"> • <code>/_search profile</code> • <code>/_shard_stores</code> • <code>/_shrink</code>⁵ • <code>/_snapshot</code> • <code>/_split</code> • <code>/_stats</code> • <code>/_status</code> • <code>/_tasks</code> • <code>/_template</code> • <code>/_update_by_query</code>¹ • <code>/_validate</code>
---	---	---

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 6.7

For Elasticsearch 6.7, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> • All operations in the index path (such as <code>/index-name/_forcemerge</code> and <code>/index-name/_update/id</code>) except <code>/index-name/_close</code> • <code>/_alias</code> • <code>/_aliases</code> • <code>/_all</code> • <code>/_analyze</code> • <code>/_bulk</code> • <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) 	<ul style="list-style-type: none"> • <code>/_cluster/state</code> • <code>/_cluster/stats</code> • <code>/_count</code> • <code>/_delete_by_query</code>¹ • <code>/_explain</code> • <code>/_field_caps</code> • <code>/_field_stats</code> • <code>/_flush</code> • <code>/_ingest/pipeline</code> • <code>/_mapping</code> 	<ul style="list-style-type: none"> • <code>/_refresh</code> • <code>/_reindex</code>¹ • <code>/_render</code> • <code>/_rollover</code> • <code>/_scripts</code>³ • <code>/_search</code>² • <code>/_search profile</code> • <code>/_shard_stores</code> • <code>/_shrink</code>⁵ • <code>/_snapshot</code>
---	---	---

• <code>/_cluster/allocation/explain</code>	• <code>/_mget</code>	• <code>/_split</code>
• <code>/_cluster/health</code>	• <code>/_msearch</code>	• <code>/_stats</code>
• <code>/_cluster/pending_tasks</code>	• <code>/_mtermvectors</code>	• <code>/_status</code>
• <code>/_cluster/settings</code> for several properties ⁴ :	• <code>/_nodes</code>	• <code>/_tasks</code>
• <code>action.auto_create_index</code>	• <code>/_opendistro/_alerting</code>	• <code>/_template</code>
• <code>action.search.shard_count.limit</code>	• <code>/_opendistro/_security</code>	• <code>/_update_by_query</code> ¹
• <code>indices.breaker.fielddata.limit</code>	• <code>/_plugin/_sql</code>	• <code>/_validate</code>
• <code>indices.breaker.request.limit</code>	• <code>/_percolate</code>	
• <code>indices.breaker.total.limit</code>	• <code>/_rank_eval</code>	
• <code>cluster.max_shards_per_node</code>		

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 6.5

For Elasticsearch 6.5, OpenSearch Service supports the following operations.

• All operations in the index path (such as <code>/index-name/_forcemerge</code> and <code>/index-name/_update/id</code>) except <code>/index-name/_close</code>	• <code>/_cluster/state</code>	• <code>/_refresh</code>
• <code>/_alias</code>	• <code>/_cluster/stats</code>	• <code>/_reindex</code> ¹
• <code>/_aliases</code>	• <code>/_count</code>	• <code>/_render</code>
• <code>/_all</code>	• <code>/_delete_by_query</code> ¹	• <code>/_rollover</code>
• <code>/_analyze</code>	• <code>/_explain</code>	• <code>/_scripts</code> ³
• <code>/_bulk</code>	• <code>/_field_caps</code>	• <code>/_search</code> ²
• <code>/_cat</code> (except <code>/_cat/nodeattrs</code>)	• <code>/_field_stats</code>	• <code>/_search profile</code>
• <code>/_cluster/allocation/explain</code>	• <code>/_flush</code>	• <code>/_shard_stores</code>
• <code>/_cluster/health</code>	• <code>/_ingest/pipeline</code>	• <code>/_shrink</code> ⁵
• <code>/_cluster/pending_tasks</code>	• <code>/_mapping</code>	• <code>/_snapshot</code>
• <code>/_cluster/settings</code> for several properties ⁴ :	• <code>/_mget</code>	• <code>/_split</code>
	• <code>/_msearch</code>	• <code>/_stats</code>
	• <code>/_mtermvectors</code>	• <code>/_status</code>
	• <code>/_nodes</code>	• <code>/_tasks</code>
		• <code>/_template</code>

<ul style="list-style-type: none"> • <code>action.auto_create_index</code> • <code>action.search.shard_count.limit</code> • <code>indices.breaker.fielddata.limit</code> • <code>indices.breaker.request.limit</code> • <code>indices.breaker.total.limit</code> 	<ul style="list-style-type: none"> • <code>/_opendistro/alerting</code> • <code>/_opendistro/_sql</code> • <code>/_percolate</code> • <code>/_plugin/kibana</code> • <code>/_rank_eval</code> 	<ul style="list-style-type: none"> • <code>/_update_by_query</code>¹ • <code>/_validate</code>
---	--	---

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 6.4

For Elasticsearch 6.4, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> • All operations in the index path (such as <code>/index-name/_forcemerge</code> and <code>/index-name/_update/id</code>) except <code>/index-name/_close</code> • <code>/_alias</code> • <code>/_aliases</code> • <code>/_all</code> • <code>/_analyze</code> • <code>/_bulk</code> • <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) • <code>/_cluster/allocation/explain</code> • <code>/_cluster/health</code> • <code>/_cluster/pending_tasks</code> • <code>/_cluster/settings</code> for several properties⁴: • <code>action.auto_create_index</code> • <code>action.search.shard_count.limit</code> • <code>indices.breaker.fielddata.limit</code> • <code>indices.breaker.request.limit</code> • <code>indices.breaker.total.limit</code> 	<ul style="list-style-type: none"> • <code>/_cluster/state</code> • <code>/_cluster/stats</code> • <code>/_count</code> • <code>/_delete_by_query</code>¹ • <code>/_explain</code> • <code>/_field_caps</code> • <code>/_field_stats</code> • <code>/_flush</code> • <code>/_ingest/pipeline</code> • <code>/_mapping</code> • <code>/_mget</code> • <code>/_msearch</code> • <code>/_mtermvectors</code> • <code>/_nodes</code> • <code>/_opendistro/_alerting</code> • <code>/_percolate</code> • <code>/_rank_eval</code> 	<ul style="list-style-type: none"> • <code>/_refresh</code> • <code>/_reindex</code>¹ • <code>/_render</code> • <code>/_rollover</code> • <code>/_scripts</code>³ • <code>/_search</code>² • <code>/_search_profile</code> • <code>/_shard_stores</code> • <code>/_shrink</code>⁵ • <code>/_snapshot</code> • <code>/_split</code> • <code>/_stats</code> • <code>/_status</code> • <code>/_tasks</code> • <code>/_template</code> • <code>/_update_by_query</code>¹ • <code>/_validate</code>
---	--	--

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 6.3

For Elasticsearch 6.3, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> • All operations in the index path (such as <code>/index-name/_forcemerge</code> and <code>/index-name/update/id</code> except <code>/index-name/_close</code>) • <code>/_alias</code> • <code>/_aliases</code> • <code>/_all</code> • <code>/_analyze</code> • <code>/_bulk</code> • <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) • <code>/_cluster/allocation/explain</code> • <code>/_cluster/health</code> • <code>/_cluster/pending_tasks</code> • <code>/_cluster/settings</code> for several properties⁴: <ul style="list-style-type: none"> • <code>action.auto_create_index</code> • <code>action.search.shard_count.lim\tpugin/kibana</code> • <code>indices.breaker.fielddata.lim\frank_eval</code> • <code>indices.breaker.request.limit</code> • <code>indices.breaker.total.limit</code> 	<ul style="list-style-type: none"> • <code>/_cluster/state</code> • <code>/_cluster/stats</code> • <code>/_count</code> • <code>/_delete_by_query¹</code> • <code>/_explain</code> • <code>/_field_caps</code> • <code>/_field_stats</code> • <code>/_flush</code> • <code>/_ingest/pipeline</code> • <code>/_mapping</code> • <code>/_mget</code> • <code>/_msearch</code> • <code>/_mtermvectors</code> • <code>/_nodes</code> • <code>/_opendistro/_alerting</code> • <code>/_percolate</code> 	<ul style="list-style-type: none"> • <code>/_refresh</code> • <code>/_reindex¹</code> • <code>/_render</code> • <code>/_rollover³</code> • <code>/_scripts³</code> • <code>/_search²</code> • <code>/_search_profile</code> • <code>/_shard_stores</code> • <code>/_shrink⁵</code> • <code>/_snapshot</code> • <code>/_split</code> • <code>/_stats</code> • <code>/_status</code> • <code>/_tasks</code> • <code>/_template</code> • <code>/_update_by_query¹</code> • <code>/_validate</code>
---	--	--

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 OpenSearch Service.

3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 6.2

For Elasticsearch 6.2, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> • All operations in the index path (such as <code>/index-name/_forcemerge</code> and <code>/index-name/_update/id</code>) except <code>/index-name/_close</code> • <code>/_alias</code> • <code>/_aliases</code> • <code>/_all</code> • <code>/_analyze</code> • <code>/_bulk</code> • <code>/_cat (except /_cat/nodeattrs)</code> • <code>/_cluster/allocation/explain</code> • <code>/_cluster/health</code> • <code>/_cluster/pending_tasks</code> • <code>/_cluster/settings</code> for several properties⁴: <ul style="list-style-type: none"> • <code>action.auto_create_index</code> • <code>action.search.shard_count.lim</code>¹<code>tplugin/kibana</code> • <code>indices.breaker.fielddata.lim</code>¹<code>rank_eval</code> • <code>indices.breaker.request.limit</code> • <code>indices.breaker.total.limit</code> 	<ul style="list-style-type: none"> • <code>/_cluster/state</code> • <code>/_cluster/stats</code> • <code>/_count</code> • <code>/_delete_by_query¹</code> • <code>/_explain</code> • <code>/_field_caps</code> • <code>/_field_stats</code> • <code>/_flush</code> • <code>/_ingest/pipeline</code> • <code>/_mapping</code> • <code>/_mget</code> • <code>/_msearch</code> • <code>/_mtermvectors</code> • <code>/_nodes</code> • <code>/_opendistro/_alerting</code> • <code>/_percolate</code> 	<ul style="list-style-type: none"> • <code>/_refresh</code> • <code>/_reindex¹</code> • <code>/_render</code> • <code>/_rollover</code> • <code>/_scripts³</code> • <code>/_search²</code> • <code>/_search profile</code> • <code>/_shard_stores</code> • <code>/_shrink⁵</code> • <code>/_snapshot</code> • <code>/_split</code> • <code>/_stats</code> • <code>/_status</code> • <code>/_tasks</code> • <code>/_template</code> • <code>/_update_by_query¹</code> • <code>/_validate</code>
---	--	--

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 6.0

For Elasticsearch 6.0, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> All operations in the index path (such as <code>/index-name/_forcemerge</code> and <code>/index-name/_update?id</code>) except <code>/index-name/_close</code> <code>/_alias</code> <code>/_aliases</code> <code>/_all</code> <code>/_analyze</code> <code>/_bulk</code> <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) <code>/_cluster/allocation/explain</code> <code>/_cluster/health</code> <code>/_cluster/pending_tasks</code> <code>/_cluster/settings</code> for several properties⁴: <ul style="list-style-type: none"> <code>action.auto_create_index</code> <code>action.search.shard_count.limit</code> <code>indices.breaker.fielddata.limit</code> <code>indices.breaker.request.limit</code> <code>indices.breaker.total.limit</code> 	<ul style="list-style-type: none"> <code>/_cluster/state</code> <code>/_cluster/stats</code> <code>/_count</code> <code>/_delete_by_query¹</code> <code>/_explain</code> <code>/_field_caps</code> <code>/_field_stats</code> <code>/_flush</code> <code>/_ingest/pipeline</code> <code>/_mapping</code> <code>/_mget</code> <code>/_msearch</code> <code>/_mtermvectors</code> <code>/_nodes</code> <code>/_percolate</code> <code>/_plugin/kibana</code> <code>/_refresh</code> <code>/_reindex¹</code> 	<ul style="list-style-type: none"> <code>/_render</code> <code>/_rollover</code> <code>/_scripts³</code> <code>/_search²</code> <code>/_search_profile</code> <code>/_shard_stores</code> <code>/_shrink⁵</code> <code>/_snapshot</code> <code>/_stats</code> <code>/_status</code> <code>/_tasks</code> <code>/_template</code> <code>/_update_by_query¹</code> <code>/_validate</code>
---	---	--

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 5.6

For Elasticsearch 5.6, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> All operations in the index path (such as <code>/index-name/_</code>) 	<ul style="list-style-type: none"> <code>/_cluster/state</code> <code>/_cluster/stats</code> 	<ul style="list-style-type: none"> <code>/_render</code> <code>/_rollover</code>
---	--	--

<ul style="list-style-type: none"> • <code>_forcemerge</code> and <code>/index-name/_update/{id}</code> except <code>/index-name/_close</code> • <code>/_alias</code> • <code>/_aliases</code> • <code>/_all</code> • <code>/_analyze</code> • <code>/_bulk</code> • <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) • <code>/_cluster/_allocation/explain</code> • <code>/_cluster/_health</code> • <code>/_cluster/_pending_tasks</code> • <code>/_cluster/_settings</code> for several properties⁴: <ul style="list-style-type: none"> • <code>action.auto_create_index</code> • <code>action.search.shard_count.limit</code> • <code>indices.breaker.fielddata.limit</code> • <code>indices.breaker.request.limit</code> • <code>indices.breaker.total.limit</code> 	<ul style="list-style-type: none"> • <code>/_count</code> • <code>/_delete_by_query</code>¹ • <code>/_explain</code> • <code>/_field_caps</code> • <code>/_field_stats</code> • <code>/_flush</code> • <code>/_ingest/pipeline</code> • <code>/_mapping</code> • <code>/_mget</code> • <code>/_msearch</code> • <code>/_mtermvectors</code> • <code>/_nodes</code> • <code>/_percolate</code> • <code>/_plugin/kibana</code> • <code>/_refresh</code> 	<ul style="list-style-type: none"> • <code>/_scripts</code>³ • <code>/_search</code>² • <code>/_search/_profile</code> • <code>/_shard_stores</code> • <code>/_shrink</code>⁵ • <code>/_snapshot</code> • <code>/_stats</code> • <code>/_status</code> • <code>/_tasks</code> • <code>/_template</code> • <code>/_update_by_query</code>¹ • <code>/_validate</code>
--	---	---

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 5.5

For Elasticsearch 5.5, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> • All operations in the index path (such as <code>/index-name/_forcemerge</code> and <code>/index-name/_update/{id}</code> except <code>/index-name/_close</code>) • <code>/_alias</code> • <code>/_aliases</code> • <code>/_all</code> • <code>/_analyze</code> 	<ul style="list-style-type: none"> • <code>/_cluster/_state</code> • <code>/_cluster/_stats</code> • <code>/_count</code> • <code>/_delete_by_query</code>¹ • <code>/_explain</code> • <code>/_field_caps</code> • <code>/_field_stats</code> • <code>/_flush</code> 	<ul style="list-style-type: none"> • <code>/_render</code> • <code>/_rollover</code> • <code>/_scripts</code>³ • <code>/_search</code>² • <code>/_search/_profile</code> • <code>/_shard_stores</code> • <code>/_shrink</code>⁵ • <code>/_snapshot</code>
--	---	--

<ul style="list-style-type: none"> • <code>/_bulk</code> • <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) • <code>/_cluster/allocation/explain</code> • <code>/_cluster/health</code> • <code>/_cluster/pending_tasks</code> • <code>/_cluster/settings</code> for several properties⁴: <ul style="list-style-type: none"> • <code>action.auto_create_index</code> • <code>action.search.shard_count.limit</code>¹ • <code>indices.breaker.fielddata.limit</code> • <code>indices.breaker.request.limit</code> • <code>indices.breaker.total.limit</code> 	<ul style="list-style-type: none"> • <code>/_ingest/pipeline</code> • <code>/_mapping</code> • <code>/_mget</code> • <code>/_msearch</code> • <code>/_mtermvectors</code> • <code>/_nodes</code> • <code>/_percolate</code> • <code>/_plugin/kibana</code> • <code>/_refresh</code> • <code>/_reindex</code>¹ 	<ul style="list-style-type: none"> • <code>/_stats</code> • <code>/_status</code> • <code>/_tasks</code> • <code>/_template</code> • <code>/_update_by_query</code>¹ • <code>/_validate</code>
---	--	---

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 OpenSearch Service.
3. For considerations about using scripts, see [the section called "Other supported resources" \(p. 409\)](#).
4. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
5. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 5.3

For Elasticsearch 5.3, OpenSearch Service supports the following operations.

<ul style="list-style-type: none"> • All operations in the index path (such as <code>/index-name/_forcemerge</code> and <code>/index-name/_update/id</code>) except <code>/index-name/_close</code> • <code>/_alias</code> • <code>/_aliases</code> • <code>/_all</code> • <code>/_analyze</code> • <code>/_bulk</code> • <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) • <code>/_cluster/allocation/explain</code> • <code>/_cluster/health</code> • <code>/_cluster/pending_tasks</code> 	<ul style="list-style-type: none"> • <code>/_cluster/state</code> • <code>/_cluster/stats</code> • <code>/_count</code> • <code>/_delete_by_query</code>¹ • <code>/_explain</code> • <code>/_field_caps</code> • <code>/_field_stats</code> • <code>/_flush</code> • <code>/_ingest/pipeline</code> • <code>/_mapping</code> • <code>/_mget</code> • <code>/_msearch</code> • <code>/_mtermvectors</code> • <code>/_nodes</code> 	<ul style="list-style-type: none"> • <code>/_render</code> • <code>/_rollover</code> • <code>/_search</code>² • <code>/_search_profile</code> • <code>/_shard_stores</code> • <code>/_shrink</code>⁴ • <code>/_snapshot</code> • <code>/_stats</code> • <code>/_status</code> • <code>/_tasks</code> • <code>/_template</code> • <code>/_update_by_query</code>¹ • <code>/_validate</code>
---	---	--

- | | |
|---|--|
| <ul style="list-style-type: none"> • <code>/_cluster/settings</code> for several properties³: • <code>action.auto_create_index</code> • <code>action.search.shard_count.limit/_reindex¹</code> • <code>indices.breaker.fielddata.limit</code> • <code>indices.breaker.request.limit</code> • <code>indices.breaker.total.limit</code> | <ul style="list-style-type: none"> • <code>/_percolate</code> • <code>/_plugin/kibana</code> • <code>/_refresh</code> • <code>/_reindex¹</code> |
|---|--|

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 OpenSearch Service.
3. Refers to the PUT method. For information about the GET method, see [the section called "Notable API differences" \(p. 374\)](#). This list only refers to the generic Elasticsearch operations that OpenSearch Service supports and does not include plugin-specific supported operations for anomaly detection, ISM, and so on.
4. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 5.1

For Elasticsearch 5.1, OpenSearch Service supports the following operations.

- | | | |
|---|---|--|
| <ul style="list-style-type: none"> • All operations in the index path (such as <code>/index-name/_forcemerge</code> and <code>/index-name/_update/id</code>) except <code>/index-name/_close</code> • <code>/_alias</code> • <code>/_aliases</code> • <code>/_all</code> • <code>/_analyze</code> • <code>/_bulk</code> • <code>/_cat</code> (except <code>/_cat/nodeattrs</code>) • <code>/_cluster/allocation/explain</code> • <code>/_cluster/health</code> • <code>/_cluster/pending_tasks</code> • <code>/_cluster/settings</code> for several properties (PUT only): <ul style="list-style-type: none"> • <code>action.auto_create_index</code> • <code>action.search.shard_count.limit/_reindex¹</code> • <code>indices.breaker.fielddata.limit</code> • <code>indices.breaker.request.limit</code> • <code>indices.breaker.total.limit</code> | <ul style="list-style-type: none"> • <code>/_cluster/state</code> • <code>/_cluster/stats</code> • <code>/_count</code> • <code>/_delete_by_query¹</code> • <code>/_explain</code> • <code>/_field_caps</code> • <code>/_field_stats</code> • <code>/_flush</code> • <code>/_ingest/pipeline</code> • <code>/_mapping</code> • <code>/_mget</code> • <code>/_msearch</code> • <code>/_mtermvectors</code> • <code>/_nodes</code> • <code>/_percolate</code> • <code>/_plugin/kibana</code> • <code>/_refresh</code> | <ul style="list-style-type: none"> • <code>/_render</code> • <code>/_rollover</code> • <code>/_search²</code> • <code>/_search_profile</code> • <code>/_shard_stores</code> • <code>/_shrink³</code> • <code>/_snapshot</code> • <code>/_stats</code> • <code>/_status</code> • <code>/_tasks</code> • <code>/_template</code> • <code>/_update_by_query¹</code> • <code>/_validate</code> |
|---|---|--|

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 OpenSearch Service.
3. See [the section called "Shrink" \(p. 375\)](#).

Elasticsearch version 2.3

For Elasticsearch 2.3, OpenSearch Service supports the following operations.

- | | |
|--|---|
| <ul style="list-style-type: none">• All operations in the index path (such as <code>/index-name/_forcemerge</code> and <code>/index-name/_recovery</code>) except <code>/index-name/_close</code>• <code>/_alias</code>• <code>/_aliases</code>• <code>/_all</code>• <code>/_analyze</code>• <code>/_bulk</code>• <code>/_cache/clear</code> (index only)• <code>/_cat</code> (except <code>/_cat/nodeattrs</code>)• <code>/_cluster/health</code>• <code>/_cluster/settings</code> for several properties (PUT only):<ul style="list-style-type: none">• <code>indices.breaker.fielddata.limit</code>• <code>indices.breaker.request.limit</code>• <code>indices.breaker.total.limit</code>• <code>threadpool.get.queue_size</code>• <code>threadpool.bulk.queue_size</code>• <code>threadpool.index.queue_size</code>• <code>threadpool.percolate.queue_size</code>• <code>threadpool.search.queue_size</code>• <code>threadpool.suggest.queue_size</code> | <ul style="list-style-type: none">• <code>/_cluster/stats</code>• <code>/_count</code>• <code>/_flush</code>• <code>/_mapping</code>• <code>/_mget</code>• <code>/_msearch</code>• <code>/_nodes</code>• <code>/_percolate</code>• <code>/_plugin/kibana</code>• <code>/_refresh</code>• <code>/_render</code>• <code>/_search</code>• <code>/_snapshot</code>• <code>/_stats</code>• <code>/_status</code>• <code>/_template</code> |
|--|---|

Elasticsearch version 1.5

For Elasticsearch 1.5, OpenSearch Service supports the following operations.

- | | |
|---|---|
| <ul style="list-style-type: none">• All operations in the index path, such as <code>/index-name/_optimize</code> and <code>/index-name/_warmer</code>, except <code>/index-name/_close</code>• <code>/_alias</code> | <ul style="list-style-type: none">• <code>/_cluster/stats</code>• <code>/_count</code>• <code>/_flush</code>• <code>/_mapping</code> |
|---|---|

- | | |
|--|--|
| <ul style="list-style-type: none"> • /_aliases • /_all • /_analyze • /_bulk • /_cat • /_cluster/health • /_cluster/settings for several properties (PUT only): <ul style="list-style-type: none"> • 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 | <ul style="list-style-type: none"> • /_mget • /_msearch • /_nodes • /_percolate • /_plugin/kibana • /_plugin/kibana3 • /_plugin/migration • /_refresh • /_search • /_snapshot • /_stats • /_status • /_template |
|--|--|

Amazon OpenSearch Service quotas

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific.

To view the quotas for OpenSearch Service, open the [Service Quotas console](#). In the navigation pane, choose **AWS services** and select **Amazon OpenSearch Service**. To request a quota increase, see [Requesting a quota increase](#) in the *Service Quotas User Guide*.

Domain and instance quotas

Your AWS account has the following quotas related to OpenSearch Service domains:

Name	Default	Adjustable	Notes
Dedicated master instances per domain	5	No	You can use the T2 and T3 instance types for dedicated master nodes only if the number of data nodes is 10 or fewer.
Domains per Region	100	Yes	
Instances per domain	80	Yes	You can request an increase up to 200 instances. For example,

Name	Default	Adjustable	Notes
			your domain might have 80 data nodes and 120 warm nodes.
Instances per domain (T2 instance type)	10	No	We don't recommend T2 or t3.small instance types for production domains.
Warm instances per domain	150	No	

Your AWS account has the following additional limits:

Name	Default	Adjustable	Notes
Total storage per domain	3 PiB	No	This maximum is the sum of all data nodes and warm nodes. For example, your domain might have 45 r6gd.16xlarge.search instances and 140 ultrawarm1.large.search instances for a total of 2.88 PiB of storage.
Custom packages per Region	25	No	
Custom packages per domain	20	No	

For a list of the instance types that OpenSearch Service supports, see [Supported instance types \(p. 365\)](#).

UltraWarm storage quotas

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 storage" \(p. 286\)](#).

Instance type	Maximum storage
ultrawarm1.medium.search	1.5 TiB
ultrawarm1.large.search	20 TiB

EBS volume size quotas

The following table shows the minimum and maximum sizes for EBS volumes for each instance type that OpenSearch Service supports. For information about which instance types include instance storage and additional hardware details, see [Amazon OpenSearch 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.small and t2.medium, and all Graviton instances (M6g, C6g, R6g, and R6gd), which don't support magnetic storage. For the maximum sizes listed in the following table, choose one of the SSD options.
- 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 (gp2)	Maximum EBS size (gp3)
t2.micro.search	10 GiB	35 GiB	N/A
t2.small.search	10 GiB	35 GiB	N/A
t2.medium.search	10 GiB	35 GiB	N/A
t3.small.search	10 GiB	100 GiB	100 GiB
t3.medium.search	10 GiB	200 GiB	200 GiB
m3.medium.search	10 GiB	100 GiB	N/A
m3.large.search	10 GiB	512 GiB	N/A
m3.xlarge.search	10 GiB	512 GiB	N/A
m3.2xlarge.search	10 GiB	512 GiB	N/A
m4.large.search	10 GiB	512 GiB	N/A
m4.xlarge.search	10 GiB	1 TiB	N/A
m4.2xlarge.search	10 GiB	1.5 TiB	N/A
m4.4xlarge.search	10 GiB	1.5 TiB	N/A
m4.10xlarge.search	10 GiB	1.5 TiB	N/A
m5.large.search	10 GiB	512 GiB	1 TiB
m5.xlarge.search	10 GiB	1 TiB	2 TiB
m5.2xlarge.search	10 GiB	1.5 TiB	3 TiB
m5.4xlarge.search	10 GiB	3 TiB	6 TiB
m5.12xlarge.search	10 GiB	9 TiB	18 TiB
m6g.large.search	10 GiB	512 GiB	1 TiB
m6g.xlarge.search	10 GiB	1 TiB	2 TiB

Instance type	Minimum EBS size	Maximum EBS size (gp2)	Maximum EBS size (gp3)
m6g.2xlarge.search	10 GiB	1.5 TiB	3 TiB
m6g.4xlarge.search	10 GiB	3 TiB	6 TiB
m6g.8xlarge.search	10 GiB	6 TiB	12 TiB
m6g.12xlarge.search	10 GiB	9 TiB	18 TiB
c4.large.search	10 GiB	100 GiB	N/A
c4.xlarge.search	10 GiB	512 GiB	N/A
c4.2xlarge.search	10 GiB	1 TiB	N/A
c4.4xlarge.search	10 GiB	1.5 TiB	N/A
c4.8xlarge.search	10 GiB	1.5 TiB	N/A
c5.large.search	10 GiB	256 GiB	256 GiB
c5.xlarge.search	10 GiB	512 GiB	512 GiB
c5.2xlarge.search	10 GiB	1 TiB	1 TiB
c5.4xlarge.search	10 GiB	1.5 TiB	1.5 TiB
c5.9xlarge.search	10 GiB	3.5 TiB	3.5 TiB
c5.18xlarge.search	10 GiB	7 TiB	7 TiB
c6g.large.search	10 GiB	256 GiB	256 GiB
c6g.xlarge.search	10 GiB	512 GiB	512 GiB
c6g.2xlarge.search	10 GiB	1 TiB	1 TiB
c6g.4xlarge.search	10 GiB	1.5 TiB	1.5 TiB
c6g.8xlarge.search	10 GiB	3 TiB	3 TiB
c6g.12xlarge.search	10 GiB	4.5 TiB	4.5 TiB
r3.large.search	10 GiB	512 GiB	N/A
r3.xlarge.search	10 GiB	512 GiB	N/A
r3.2xlarge.search	10 GiB	512 GiB	N/A
r3.4xlarge.search	10 GiB	512 GiB	N/A
r3.8xlarge.search	10 GiB	512 GiB	N/A
r4.large.search	10 GiB	1 TiB	N/A
r4.xlarge.search	10 GiB	1.5 TiB	N/A
r4.2xlarge.search	10 GiB	1.5 TiB	N/A
r4.4xlarge.search	10 GiB	1.5 TiB	N/A

Instance type	Minimum EBS size	Maximum EBS size (gp2)	Maximum EBS size (gp3)
r4.8xlarge.search	10 GiB	1.5 TiB	N/A
r4.16xlarge.search	10 GiB	1.5 TiB	N/A
r5.large.search	10 GiB	1 TiB	2 TiB
r5.xlarge.search	10 GiB	1.5 TiB	3 TiB
r5.2xlarge.search	10 GiB	3 TiB	6 TiB
r5.4xlarge.search	10 GiB	6 TiB	12 TiB
r5.12xlarge.search	10 GiB	12 TiB	24 TiB
r6g.large.search	10 GiB	1 TiB	2 TiB
r6g.xlarge.search	10 GiB	1.5 TiB	3 TiB
r6g.2xlarge.search	10 GiB	3 TiB	6 TiB
r6g.4xlarge.search	10 GiB	6 TiB	12 TiB
r6g.8xlarge.search	10 GiB	8 TiB	16 TiB
r6g.12xlarge.search	10 GiB	12 TiB	24 TiB
r6gd.large.search	N/A	N/A	N/A
r6gd.xlarge.search	N/A	N/A	N/A
r6gd.2xlarge.search	N/A	N/A	N/A
r6gd.4xlarge.search	N/A	N/A	N/A
r6gd.8xlarge.search	N/A	N/A	N/A
r6gd.12xlarge.search	N/A	N/A	N/A
r6gd.16xlarge.search	N/A	N/A	N/A
i2.xlarge.search	10 GiB	512 GiB	N/A
i2.2xlarge.search	10 GiB	512 GiB	N/A
i3.large.search	N/A	N/A	N/A
i3.xlarge.search	N/A	N/A	N/A
i3.2xlarge.search	N/A	N/A	N/A
i3.4xlarge.search	N/A	N/A	N/A
i3.8xlarge.search	N/A	N/A	N/A
i3.16xlarge.search	N/A	N/A	N/A

Network quotas

The following table shows the maximum size of HTTP request payloads.

Instance type	Maximum size of HTTP request payloads
t2.micro.search	10 MiB
t2.small.search	10 MiB
t2.medium.search	10 MiB
t3.small.search	10 MiB
t3.medium.search	10 MiB
m3.medium.search	10 MiB
m3.large.search	10 MiB
m3.xlarge.search	100 MiB
m3.2xlarge.search	100 MiB
m4.large.search	10 MiB
m4.xlarge.search	100 MiB
m4.2xlarge.search	100 MiB
m4.4xlarge.search	100 MiB
m4.10xlarge.search	100 MiB
m5.large.search	10 MiB
m5.xlarge.search	100 MiB
m5.2xlarge.search	100 MiB
m5.4xlarge.search	100 MiB
m5.12xlarge.search	100 MiB
m6g.large.search	10 MiB
m6g.xlarge.search	100 MiB
m6g.2xlarge.search	100 MiB
m6g.4xlarge.search	100 MiB
m6g.8xlarge.search	100 MiB
m6g.12xlarge.search	100 MiB
c4.large.search	10 MiB
c4.xlarge.search	100 MiB
c4.2xlarge.search	100 MiB

Instance type	Maximum size of HTTP request payloads
c4.4xlarge.search	100 MiB
c4.8xlarge.search	100 MiB
c5.large.search	10 MiB
c5.xlarge.search	100 MiB
c5.2xlarge.search	100 MiB
c5.4xlarge.search	100 MiB
c5.9xlarge.search	100 MiB
c5.18xlarge.search	100 MiB
c6g.large.search	10 MiB
c6g.xlarge.search	100 MiB
c6g.2xlarge.search	100 MiB
c6g.4xlarge.search	100 MiB
c6g.8xlarge.search	100 MiB
c6g.12xlarge.search	100 MiB
r3.large.search	10 MiB
r3.xlarge.search	100 MiB
r3.2xlarge.search	100 MiB
r3.4xlarge.search	100 MiB
r3.8xlarge.search	100 MiB
r4.large.search	100 MiB
r4.xlarge.search	100 MiB
r4.2xlarge.search	100 MiB
r4.4xlarge.search	100 MiB
r4.8xlarge.search	100 MiB
r4.16xlarge.search	100 MiB
r5.large.search	100 MiB
r5.xlarge.search	100 MiB
r5.2xlarge.search	100 MiB
r5.4xlarge.search	100 MiB
r5.12xlarge.search	100 MiB
r6g.large.search	100 MiB

Instance type	Maximum size of HTTP request payloads
r6g.xlarge.search	100 MiB
r6g.2xlarge.search	100 MiB
r6g.4xlarge.search	100 MiB
r6g.8xlarge.search	100 MiB
r6g.12xlarge.search	100 MiB
r6gd.large.search	100 MiB
r6gd.xlarge.search	100 MiB
r6gd.2xlarge.search	100 MiB
r6gd.4xlarge.search	100 MiB
r6gd.8xlarge.search	100 MiB
r6gd.12xlarge.search	100 MiB
r6gd.16xlarge.search	100 MiB
i2.xlarge.search	100 MiB
i2.2xlarge.search	100 MiB
i3.large.search	100 MiB
i3.xlarge.search	100 MiB
i3.2xlarge.search	100 MiB
i3.4xlarge.search	100 MiB
i3.8xlarge.search	100 MiB
i3.16xlarge.search	100 MiB

Java process quota

OpenSearch Service 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 cluster settings” \(p. 21\)](#) and [the section called “JVM OutOfMemoryError” \(p. 441\)](#).

Domain policy quota

OpenSearch Service limits [access policies on domains \(p. 130\)](#) to 100 KiB.

Reserved Instances in Amazon OpenSearch Service

Reserved Instances (RIs) in Amazon OpenSearch Service 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.

OpenSearch Service 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.

RIs are not flexible; they only apply to the exact instance type that you reserve. For example, a reservation for eight c5.2xlarge.search instances does not apply to sixteen c5.xlarge.search instances or four c5.4xlarge.search instances. For full details, see [Amazon OpenSearch Service pricing](#) and [FAQ](#).

Topics

- [Purchasing Reserved Instances \(console\) \(p. 405\)](#)
- [Purchasing Reserved Instances \(AWS CLI\) \(p. 406\)](#)
- [Purchasing Reserved Instances \(AWS SDKs\) \(p. 407\)](#)
- [Examining costs \(p. 408\)](#)

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 **Amazon OpenSearch Service**.
3. Choose **Reserved Instance Leases** from the navigation pane.

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 Instance Leases** link, [create a domain \(p. 16\)](#) in the AWS Region.

4. Choose **Order Reserved Instance**.
5. Provide a unique and descriptive name.
6. Choose an instance type and the number of instances. For guidance, see [the section called "Sizing domains" \(p. 353\)](#).
7. Choose a term length and payment option. Review the payment details carefully.
8. Choose **Next**.
9. Review the purchase summary carefully. Purchases of Reserved Instances are non-refundable.
10. Choose **Order**.

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 opensearch describe-reserved-instance-offerings --region us-east-1
{
    "ReservedInstanceOfferings": [
        {
            "FixedPrice": x,
            "ReservedInstanceOfferingId": "1a2a3a4a5-1a2a-3a4a-5a6a-1a2a3a4a5a6a",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": y,
                    "RecurringChargeFrequency": "Hourly"
                }
            ],
            "UsagePrice": 0.0,
            "PaymentOption": "PARTIAL_UPFRONT",
            "Duration": 31536000,
            "InstanceType": "m4.2xlarge.search",
            "CurrencyCode": "USD"
        }
    ]
}
```

For an explanation of each return value, see the following table.

Field	Description
FixedPrice	The upfront cost of the reservation.
ReservedInstanceOfferingId	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 OpenSearch Service, this value is always 0.
PaymentOption	No Upfront, Partial Upfront, or All Upfront.
Duration	Length of the term in seconds: <ul style="list-style-type: none"> • 31536000 seconds is one year. • 94608000 seconds is three years.
InstanceType	The instance type for the reservation. For information about the hardware resources that are allocated to each instance type, see Amazon OpenSearch Service pricing .
CurrencyCode	The currency for FixedPrice and RecurringChargeAmount.

This next example purchases a reservation:

```
aws opensearch purchase-reserved-instance-offering --reserved-instance-offering-id 1a2a3a4a5-1a2a-3a4a-5a6a-1a2a3a4a5a6a --reservation-name my-reservation --instance-count 3 --region us-east-1
{
    "ReservationName": "my-reservation",
    "ReservedInstanceId": "9a8a7a6a-5a4a-3a2a-1a0a-9a8a7a6a5a4a"
}
```

Finally, you can list your reservations for a given Region using the following example:

```
aws opensearch describe-reserved-instances --region us-east-1
{
    "ReservedInstances": [
        {
            "FixedPrice": x,
            "ReservedInstanceOfferingId": "1a2a3a4a5-1a2a-3a4a-5a6a-1a2a3a4a5a6a",
            "ReservationName": "my-reservation",
            "PaymentOption": "PARTIAL_UPFRONT",
            "UsagePrice": 0.0,
            "ReservedInstanceId": "9a8a7a6a-5a4a-3a2a-1a0a-9a8a7a6a5a4a",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": y,
                    "RecurringChargeFrequency": "Hourly"
                }
            ],
            "State": "payment-pending",
            "StartTime": 1522872571.229,
            "InstanceCount": 3,
            "Duration": 31536000,
            "InstanceType": "m4.2xlarge.search",
            "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 OpenSearch Service API Reference](#), including the following:

- `DescribeReservedInstanceOfferings`
- `PurchaseReservedInstanceOffering`
- `DescribeReservedInstances`

This sample script uses the `OpenSearchService` low-level Python client from the AWS SDK for Python (Boto3) to purchase Reserved Instances. You must provide a value for `instance_type`.

```
import boto3
from botocore.config import Config
```

```
# Build the client using the default credential configuration.  
# You can use the CLI and run 'aws configure' to set access key, secret  
# key, and default region.  
  
my_config = Config(  
    # Optionally lets you specify a region other than your default.  
    region_name='us-east-1'  
)  
  
client = boto3.client('opensearch', config=my_config)  
instance_type = '' # e.g. m4.2xlarge.search  
  
def describe_RI_offerings(client):  
    """Gets the Reserved Instance offerings for this account"""  
  
    response = client.describe_reserved_instance_offerings()  
    offerings = (response['ReservedInstanceOfferings'])  
    return offerings  
  
def check_instance(offering):  
    """Returns True if instance type is the one you specified above"""  
  
    if offering['InstanceType'] == instance_type:  
        return True  
  
    return False  
  
def get_instance_id():  
    """Iterates through the available offerings to find the ID of the one you specified"""  
  
    instance_type_iterator = filter(  
        check_instance, describe_RI_offerings(client))  
    offering = list(instance_type_iterator)  
    id = offering[0]['ReservedInstanceOfferingId']  
    return id  
  
def purchase_RI_offering(client):  
    """Purchase Reserved Instances"""  
  
    response = client.purchase_reserved_instance_offering(  
        ReservedInstanceId = get_instance_id(),  
        ReservationName = 'my-reservation',  
        InstanceCount = 1  
    )  
    print('Purchased reserved instance offering of type ' + instance_type)  
    print(response)  
  
def main():  
    """Purchase Reserved Instances"""  
    purchase_RI_offering(client)
```

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 User Guide](#).

Other supported resources in Amazon OpenSearch Service

This topic describes additional resources that Amazon OpenSearch Service supports.

bootstrap.memory_lock

OpenSearch Service enables `bootstrap.memory_lock` in `opensearch.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.search`
- `t2.small.search`
- `t2.medium.search`
- `t3.small.search`
- `t3.medium.search`

Scripting module

OpenSearch Service supports scripting for Elasticsearch 5.x and later domains. It 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, and all OpenSearch domains, OpenSearch Service supports stored scripts using the `_scripts` endpoint. Elasticsearch 5.3 and 5.1 domains support inline scripts only.

TLS transport

OpenSearch Service supports HTTP on port 80 and HTTPS over port 443, but does not support TLS transport.

Amazon OpenSearch Service tutorials

This chapter includes several start-to-finish tutorials for working with Amazon OpenSearch Service, including how to migrate to the service, build a simple search application, and create a visualization in OpenSearch Dashboards.

Topics

- [Tutorial: Creating and searching for documents in Amazon OpenSearch Service \(p. 410\)](#)
- [Tutorial: Migrating to Amazon OpenSearch Service \(p. 415\)](#)
- [Tutorial: Creating a search application with Amazon OpenSearch Service \(p. 419\)](#)
- [Tutorial: Visualizing customer support calls with OpenSearch Service and OpenSearch Dashboards \(p. 426\)](#)

Tutorial: Creating and searching for documents in Amazon OpenSearch Service

In this tutorial, you learn how to create and search for a document in Amazon OpenSearch Service. You add data to an index in the form of a JSON document. OpenSearch Service creates an index around the first document that you add.

This tutorial explains how to make HTTP requests to create documents, automatically generate an ID for a document, and perform basic and advanced searches on your documents.

Note

This tutorial uses a domain with open access. For the highest level of security, we recommend that you put your domain inside a virtual private cloud (VPC).

Prerequisites

This tutorial has the following prerequisites:

- You must have an AWS account.
- You must have an active OpenSearch Service domain.

Adding a document to an index

To add a document to an index, you can use any HTTP tool, such as [Postman](#), cURL, or the OpenSearch Dashboards console. These examples assume that you're using the developer console in OpenSearch Dashboards. If you're using a different tool, adjust accordingly by providing the full URL and credentials, if necessary.

To add a document to an index

1. Navigate to the OpenSearch Dashboards URL for your domain. You can find the URL on the domain's dashboard in the OpenSearch Service console. The URL follows this format:

`domain-endpoint/_dashboards/`

2. Sign in using your primary user name and password.
3. Open the left navigation panel and choose **Dev Tools**.
4. The HTTP verb for creating a new resource is PUT, which is what you use to create a new document and index. Enter the following command in the console:

```
PUT fruit/_doc/1
{
  "name": "strawberry",
  "color": "red"
}
```

The PUT request creates an index named *fruit* and adds a single document to the index with an ID of 1. It produces the following response:

```
{
  "_index" : "fruit",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "_seq_no" : 0,
  "_primary_term" : 1
}
```

Creating automatically generated IDs

OpenSearch Service can automatically generate an ID for your documents. The command to generate IDs uses a POST request instead of a PUT request, and it requires no document ID (in comparison to the previous request).

Enter the following request in the developer console:

```
POST veggies/_doc
{
  "name": "beet",
  "color": "red",
  "classification": "root"
}
```

This request creates an index named *veggies* and adds the document to the index. It produces the following response:

```
{
  "_index" : "veggies",
  "_type" : "_doc",
  "_id" : "3WgyS4IB5DLqbRIvLxtF",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  }
}
```

```
        },
        "_seq_no" : 0,
        "_primary_term" : 1
    }
```

Note that additional `_id` field in the response, which indicates that an ID was automatically created.

Note

You don't provide anything after `_doc` in the URL, where the ID normally goes. Because you're creating a document with a generated ID, you don't provide one yet. That's reserved for updates.

Updating a document with a POST command

To update a document, you use an HTTP POST command with the ID number.

First, create a document with an ID of 42:

```
POST fruits/_doc/42
{
    "name": "banana",
    "color": "yellow"
}
```

Then use that ID to update the document:

```
POST fruits/_doc/42
{
    "name": "banana",
    "color": "yellow",
    "classification": "berries"
}
```

This command updates the document with the new field `classification`. It produces the following response:

```
{
    "_index" : "fruits",
    "_type" : "_doc",
    "_id" : "42",
    "_version" : 2,
    "result" : "updated",
    "_shards" : {
        "total" : 2,
        "successful" : 2,
        "failed" : 0
    },
    "_seq_no" : 1,
    "_primary_term" : 1
}
```

Note

If you try to update a document that does not exist, OpenSearch Service creates the document.

Performing bulk actions

You can use the POST `_bulk` API operation to perform multiple actions on one or more indexes in one request. Bulk action commands take the following format:

```
POST /_bulk
<action_meta>\n
<action_data>\n
<action_meta>\n
<action_data>\n
```

Each action requires two lines of JSON. First, you provide the action description or metadata. On the next line, you provide the data. Each part is separated by a newline (\n). An action description for an insert might look like this:

```
{ "create" : { "_index" : "veggies", "_type" : "_doc", "_id" : "7" } }
```

And the next line containing the data might look like this:

```
{ "name": "kale", "color": "green", "classification": "leafy-green" }
```

Taken together, the metadata and the data represent a single action in a bulk operation. You can perform many operations in one request, like this:

```
POST /_bulk
{ "create" : { "_index" : "veggies", "_type" : "_doc", "_id" : "35" } }
{ "name": "kale", "color": "green", "classification": "leafy-green" }
{ "create" : { "_index" : "veggies", "_type" : "_doc", "_id" : "36" } }
{ "name": "spinach", "color": "green", "classification": "leafy-green" }
{ "create" : { "_index" : "veggies", "_type" : "_doc", "_id" : "37" } }
{ "name": "arugula", "color": "green", "classification": "leafy-green" }
{ "create" : { "_index" : "veggies", "_type" : "_doc", "_id" : "38" } }
{ "name": "endive", "color": "green", "classification": "leafy-green" }
{ "create" : { "_index" : "veggies", "_type" : "_doc", "_id" : "39" } }
{ "name": "lettuce", "color": "green", "classification": "leafy-green" }
{ "delete" : { "_index" : "vegetables", "_type" : "_doc", "_id" : "1" } }
```

Notice that the last action is a delete. There's no data following the delete action.

Searching for documents

Now that data exists in your cluster, you can search for it. For example, you might want to search for all root vegetables, or get a count of all leafy greens, or find the number of errors logged per hour.

Basic searches

A basic search looks something like this:

```
GET veggies/_search?q=name:lettuce
```

The request produces a JSON response that contains the lettuce document.

Advanced searches

You can perform more advanced searches by providing the query options as JSON in the request body:

```
GET veggies/_search
{
  "query": {
    "term": {
      "name": "lettuce"
    }
  }
}
```

```
    },
}
```

This example also produces a JSON response with the lettuce document.

Sorting

You can perform more of this type of query using sorting. First, you need to recreate the index, because the automatic field mapping chose types that can't be sorted by default. Send the following requests to delete and recreate the index:

```
DELETE /veggies

PUT /veggies
{
  "mappings": {
    "properties": {
      "name": {
        "type": "keyword"
      },
      "color": {
        "type": "keyword"
      },
      "classification": {
        "type": "keyword"
      }
    }
  }
}
```

Then repopulate the index with data:

```
POST/_bulk
{ "create" : { "_index" : "veggies", "_type" : "_doc", "_id" : "7" } }
{ "name":"kale", "color":"green", "classification":"leafy-green" }
{ "create" : { "_index" : "veggies", "_type" : "_doc", "_id" : "8" } }
{ "name":"spinach", "color":"green", "classification":"leafy-green" }
{ "create" : { "_index" : "veggies", "_type" : "_doc", "_id" : "9" } }
{ "name":"arugula", "color":"green", "classification":"leafy-green" }
{ "create" : { "_index" : "veggies", "_type" : "_doc", "_id" : "10" } }
{ "name":"endive", "color":"green", "classification":"leafy-green" }
{ "create" : { "_index" : "veggies", "_type" : "_doc", "_id" : "11" } }
{ "name":"lettuce", "color":"green", "classification":"leafy-green" }
```

Now you can search with a sort. This request adds an ascending sort by the classification:

```
GET veggies/_search
{
  "query" : {
    "term": { "color": "green" }
  },
  "sort" : [
    "classification"
  ]
}
```

Related resources

For more information, see the following resources:

- [Getting started \(p. 11\)](#)
- [Indexing data \(p. 217\)](#)
- [Searching data \(p. 232\)](#)

Tutorial: Migrating to Amazon OpenSearch Service

Index snapshots are a popular way to migrate from a self-managed OpenSearch or legacy Elasticsearch cluster to Amazon OpenSearch 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 OpenSearch Service domain.
3. Give OpenSearch Service permissions to access the bucket, and give your user account permissions to work with snapshots.
4. Restore the snapshot on the OpenSearch Service 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 `opensearch.yml` (or `elasticsearch.yml` if using an Elasticsearch cluster), 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 `opensearch.yml` (or `Elasticsearch.yml`) on all nodes, and then restart each node.

```
path.repo: ["/my/shared/directory/snapshots"]
```

2. Register a [snapshot repository](#), which is required before you take a snapshot. A repository is just a storage location: a shared file system, Amazon S3, Hadoop Distributed File System (HDFS), etc. In this case, we'll use a shared file system ("fs"):

```
PUT _snapshot/my-snapshot-repo-name
{
  "type": "fs",
  "settings": {
    "location": "/my/shared/directory/snapshots"
  }
}
```

3. Take the snapshot:

```
PUT _snapshot/my-snapshot-repo-name/my-snapshot-name
{
  "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://bucket-name --region us-west-2
aws s3 sync . s3://bucket-name --sse AES256
```

Depending on the size of the snapshot and the speed of your internet connection, this operation can take a while.

Create a 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 opensearch create-domain \
--domain-name migration-domain \
--engine-version OpenSearch_1.0 \
--cluster-config InstanceType=c5.large.search,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,TLSecurityPolicy=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 \(p. 146\)](#) with HTTP basic authentication and all encryption settings. Use the OpenSearch Service 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 AWS Region that you used for the S3 bucket and an OpenSearch/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 OpenSearch Service domain. We recommend choosing the most recent version of OpenSearch or Elasticsearch that your snapshot supports. For a table of compatible versions, see [the section called "Using a snapshot to migrate data" \(p. 56\)](#).

Provide permissions to the S3 bucket

In the AWS Identity and Access Management (IAM) console, [create a role](#) with the following permissions and [trust relationship](#). When creating the role, choose **S3** as the **AWS Service**. Name the role **OpenSearchSnapshotRole** so it's easy to find.

Permissions

```
{
```

```
"Version": "2012-10-17",
"Statement": [{"Action": ["s3>ListBucket"], "Effect": "Allow", "Resource": ["arn:aws:s3:::bucket-name"]}, {"Action": ["s3GetObject", "s3PutObject", "s3DeleteObject"], "Effect": "Allow", "Resource": ["arn:aws:s3:::bucket-name/*"]}], ]}
```

Trust relationship

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

Then give your personal IAM user or role—whatever you used to configure the AWS CLI earlier—permissions to assume OpenSearchSnapshotRole. Create the following policy and [attach it](#) to your identity:

Permissions

```
{ "Version": "2012-10-17",
"Statement": [{"Effect": "Allow", "Action": "iamPassRole", "Resource": "arn:aws:iam::123456789012:role/OpenSearchSnapshotRole"}]}
```

Map the snapshot role in OpenSearch Dashboards (if using fine-grained access control)

If you enabled [fine-grained access control \(p. 154\)](#), even if you use HTTP basic authentication for all other purposes, you need to map the manage_snapshots role to your IAM user or role so you can work with snapshots.

To give your identity permissions to work with snapshots

1. Log in to Dashboards using the master user credentials you specified when you created the OpenSearch Service domain. You can find the Dashboards URL in the OpenSearch Service console. It takes the form of `https://domain-endpoint/_dashboards/`.
2. From the main menu choose **Security, Roles**, and select the **manage_snapshots** role.
3. Choose **Mapped users, Manage mapping**.
4. Add the domain ARN of your personal IAM user or role in the appropriate field. The ARN takes one of the following formats:

```
arn:aws:iam::123456789123:user/user-name
```

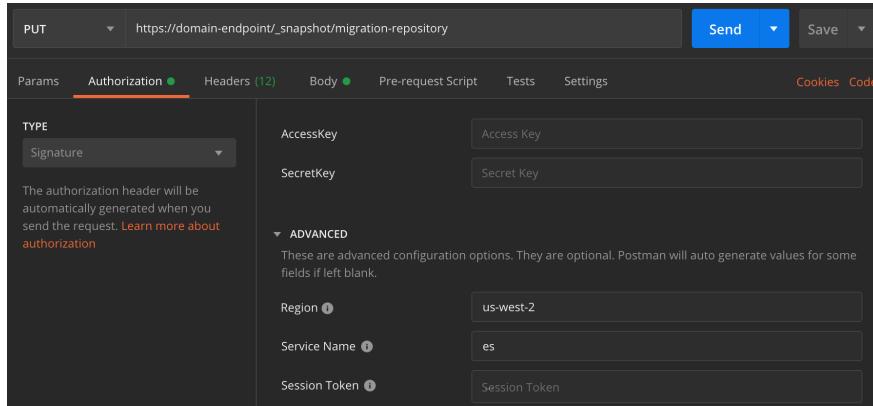
```
arn:aws:iam::123456789123:role/role-name
```

5. Select **Map** and confirm the user or role shows up under **Mapped users**.

Restore the snapshot

At this point, you have two ways to access your OpenSearch Service 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 OpenSearch Service domain.

Most programming languages have libraries to assist with [signing requests \(p. 191\)](#), but the simpler approach is to use a tool like [Postman](#) and put your IAM credentials into the **Authorization** section.



To restore the snapshot

1. Regardless of how you choose to sign your requests, the first step is to register the repository:

```
PUT _snapshot/my-snapshot-repo-name
{
  "type": "s3",
  "settings": {
    "bucket": "bucket-name",
    "region": "us-west-2",
    "role_arn": "arn:aws:iam::123456789012:role/OpenSearchSnapshotRole"
  }
}
```

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/my-snapshot-repo-name/_all
```

curl

```
curl -XGET -u 'master-user:master-user-password' https://domain-endpoint/_snapshot/my-snapshot-repo-name/_all
```

3. Restore the snapshot.

Shorthand

```
POST _snapshot/my-snapshot-repo-name/my-snapshot-name/_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/my-snapshot-repo-name/my-snapshot-name/_restore \
-H 'Content-Type: application/json' \
-d '{"indices": "migration-index1,migration-index2,other-indices-*", "include_global_state":false}'
```

4. Finally, verify that your indexes 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 OpenSearch Service endpoint, [resize the domain \(p. 353\)](#) to suit your workload, check the shard count for your indexes, switch to an [IAM master user \(p. 149\)](#), or start building visualizations in OpenSearch Dashboards.

Tutorial: Creating a search application with Amazon OpenSearch Service

A common way to create a search application with Amazon OpenSearch Service is to use web forms to send user queries to a server. Then you can authorize the server to call the OpenSearch APIs directly and have the server send requests to OpenSearch Service. However, if you want to write client-side code that doesn't rely on a server, you should compensate for the security and performance risks. Allowing unsigned, public access to the OpenSearch 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 OpenSearch APIs and AWS Lambda to sign requests from API Gateway to OpenSearch Service.

Note

Standard API Gateway and Lambda pricing applies, but within the limited usage of this tutorial, costs should be negligible.

Prerequisites

A prerequisite for this tutorial is an OpenSearch Service domain. If you don't already have one, follow the steps in [Create an OpenSearch Service domain \(p. 11\)](#) to create one.

Step 1: Index sample data

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" } }
{"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/MV5BMTQyMDE0MTY0OV5BMl5BanBnXkFtZTcwMjI20TI0Q@._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 Br\u00fchl", "Chris Hemsworth", "Olivia Wilde"], "year": 2013, "id": "tt1979320", "type": "add"}
{ "index": { "_index": "movies", "_type": "movie", "_id": "tt1951264" } }
{"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"}
...

```

For instructions, see [the section called "Option 2: Upload multiple documents" \(p. 12\)](#).

Step 2: Create the API in API Gateway

Using API Gateway lets you create a more limited API and simplifies the process of interacting with the OpenSearch _search API. API Gateway lets you enable security features like Amazon Cognito authentication and request throttling. Perform the following steps to create and deploy an API:

Create and configure the API

To create your API using the API Gateway console

1. Within API Gateway, choose **Create API**.
2. Locate **REST API** (not private) and choose **Build**.
3. Configure the following fields:
 - API name: **opensearch-api**
 - Description: **Public API for searching an Amazon OpenSearch Service domain**
 - Endpoint Type: **Regional**
4. Choose **Create API**.

5. Choose **Actions** and **Create Method**.
6. Select **GET** in the dropdown and click the checkmark to confirm.
7. Configure the following settings, then choose **Save**:

Setting	Value
Integration type	Lambda function
Use Lambda proxy integration	Yes
Lambda region	<i>us-west-1</i>
Lambda function	opensearch-lambda (you'll configure this later in Lambda)
Use default timeout	Yes

Note

If you're performing these steps in order, you'll see an error: "Function not found: arn:aws:lambda:us-west-1:123456789012:function:opensearch-lambda". You can ignore this error, as you'll configure the Lambda function in step 3.

Configure the method request

Choose **Method Request** and configure the following settings:

Setting	Value
Authorization	NONE
Request Validator	Validate query string parameters and headers
API Key Required	false

URL Query String Parameters

Setting	Value
Name	q
Required	Yes

Deploy the API and configure a stage

The API Gateway console lets you deploy an API by creating a deployment and associating it with a new or existing stage.

1. Choose **Actions** and **Deploy API**.
2. For **Deployment stage** choose **New Stage** and name the stage opensearch-api-test.
3. Choose **Deploy**.

4. Configure the following settings in the stage editor, then choose **Save Changes**:

Setting	Value
Enable throttling	Yes
Rate	1000
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 runs the opensearch-lambda function. For more information, see [Creating an API in Amazon API Gateway](#) and [Deploying a REST API in Amazon API Gateway](#).

Step 3: Create and deploy the Lambda function

After you create your API in API Gateway, create the Lambda function that it passes requests to.

Create the Lambda function

In this solution, API Gateway passes requests to the following Python 3.8 Lambda function, which queries OpenSearch Service and returns results. Name the function opensearch-lambda.

Because this sample function uses external libraries, you need to 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 [Deploy Python Lambda functions with .zip file archives](#) in the *AWS Lambda Developer Guide* and the section called "Create the Lambda deployment package" (p. 220) in this guide.

```
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 = '' # The OpenSearch domain endpoint with https:// and without a trailing slash
index = 'movies'
url = host + '/' + index + '/_search'

# Lambda execution starts here
def lambda_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": ["title^4", "plot^2", "actors", "directors"]
            }
        }
    }
```

```

        }

# Elasticsearch 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

```

Modify the handler

The *handler* is the method in your function code that processes events. You need to change the handler name according to the name of the file in your deployment package where the Lambda function is located. For example, if your file is named `function.py`, rename the handler to `function.lambda_handler`. For more information, see [Lambda function handler in Python](#).

Configure a trigger

Choose **Add trigger** and create the HTTP endpoint that invokes your function. The trigger must have the following configuration:

Trigger	API	Deployment Stage	Security
API Gateway	opensearch-api	opensearch-api-test	Open

Step 4: (Optional) Modify the domain access policy

Your OpenSearch Service domain must allow the Lambda function to make GET requests to the movies index. If your domain has an open access policy with fine-grained access control enabled, you can leave it as-is:

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

Alternatively, you can choose to make your domain access policy more granular. For example, the following minimum policy provides opensearch-lambda-role (created through Lambda) read access to the movies index. To get the exact name of the role that Lambda automatically creates, go to the AWS Identity and Access Management (IAM) console, choose **Roles**, and search for "lambda".

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::123456789012:role/service-role/opensearch-lambda-  
role-1abcdefg"  
            },  
            "Action": "es:ESHttpGet",  
            "Resource": "arn:aws:es:us-west-1:123456789012:domain/domain-name/movies/_search"  
        }  
    ]  
}
```

Note

If you have fine-grained access control enabled for the domain, you also need to [map the role to a user \(p. 154\)](#) in OpenSearch Dashboards, otherwise you'll see permissions errors.

For more information about access policies, see [the section called "Configuring access policies" \(p. 21\)](#).

Map the Lambda role (if using fine-grained access control)

Fine-grained access control introduces an additional step before you can test the application. Even if you use HTTP basic authentication for all other purposes, you need to map the Lambda role to a user, otherwise you'll see permissions errors.

1. Navigate to the OpenSearch Dashboards endpoint for the domain.
2. From the main menu, choose **Security, Roles**, and select user or role to map the Lambda role to.
3. Choose **Mapped users, Manage mapping**.
4. Under **Backend roles**, add the Amazon Resource Name (ARN) of the Lambda role.

```
arn:aws:iam::123456789123:role/opensearch-lambda-role-1abcdefg
```

5. Select **Map** and confirm the user or role shows up under **Mapped users**.

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/opensearch-api-test`. You can quickly find the endpoint in API Gateway by choosing **Stages** and selecting the name of the API.
3. Open `index.html` and try running searches for `thor`, `house`, and a few other terms.

Movie Search

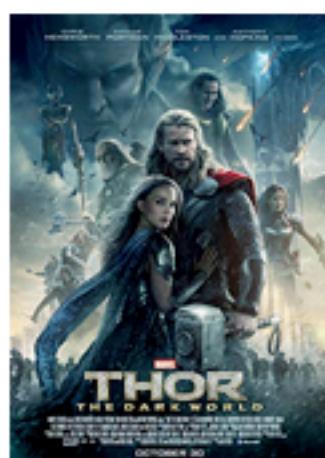
thor

Found 7 results.



Thor

2011 — The powerful but arrogant god Thor is cast down to Earth by his own father Odin amongst humans in Midgard (Earth), where he must prove himself to be their finest defenders.



Thor: The Dark World

2013 — Faced with an enemy that even God himself cannot withstand, Thor must embark on his most dangerous and personal mission yet, one that will reunite him with Jane Foster and bring back everything to save us all.



Vikingdom

API Version 2015-01-01

425

2013 — A forgotten king, Eirick, is tasked with defeating Thor, the God of Thunder.

Troubleshoot CORS errors

Even though the Lambda function includes content in the response to support CORS, you still might see the following error:

```
Access to XMLHttpRequest at '<api-gateway-endpoint>' from origin 'null' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present in the requested resource.
```

If this happens, try the following:

1. [Enable CORS](#) on the GET resource. Under **Advanced**, set **Access-Control-Allow-Credentials** to '`true`'.
2. Redeploy your API in API Gateway ([Actions](#), [Deploy API](#)).
3. Delete and re-add your Lambda function trigger.

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 OpenSearch Service 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.

Tutorial: Visualizing customer support calls with OpenSearch Service and OpenSearch Dashboards

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 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 OpenSearch Service, 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 OpenSearch and OpenSearch Dashboards 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 OpenSearch Service.

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](#).
- OpenSearch Service domain with one t2.medium instance and 10 GiB of EBS storage for several hours
 - To learn more, see [Amazon OpenSearch 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 \(p. 427\)](#)
- [Step 2: Copy sample code \(p. 428\)](#)
- [\(Optional\) Step 3: Index sample data \(p. 430\)](#)
- [Step 4: Analyze and visualize your data \(p. 432\)](#)
- [Step 5: Clean up resources and next steps \(p. 435\)](#)

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 User Guide .
OpenSearch Service domain	The destination for data. For more information, see Creating OpenSearch Service domains (p. 16) .

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 opensearch create-domain --domain-name my-transcribe-test --engine-version OpenSearch_1.0 --cluster-config InstanceType=t2.medium.search,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 Python 3 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 urllib.request

# 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 = '_doc'
region = 'us-west-2'

# Upload audio file to S3.
s3_client = boto3.client('s3')

audio_file = open(audio_file_name, 'rb')

print('Uploading ' + audio_file_name + '...')
response = s3_client.put_object(
    Body=audio_file,
    Bucket=bucket_name,
    Key=audio_file_name
)

# # Build the URL to the audio file on S3.
# # Only for the us-east-1 region.
# mp3_uri = 'https://' + bucket_name + '.s3.amazonaws.com/' + audio_file_name

# Get the necessary details and build the URL to the audio file on S3.
# For all other regions.
response = s3_client.get_bucket_location(
    Bucket=bucket_name
)
bucket_region = response['LocationConstraint']
mp3_uri = 'https://' + bucket_name + '.s3-' + bucket_region + '.amazonaws.com/' +
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 = urllib.request.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 OpenSearch 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, region,
    'opensearchservice', session_token=credentials.token)

# 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...
Detecting key phrases...
Detecting sentiment...
Indexing document...
<Response [201]>
{"_type": "call", "_seq_no": 0, "_shards": {"successful": 1, "failed": 0, "total": 2}, "_index": "support-calls4", "_version": 1, "_primary_term": 1, "result": "created", "_id": "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 OpenSearch Service.

Tip

[LibriVox](#) has public domain audiobooks that you can use for testing.

(Optional) Step 3: Index sample data

If you don't have a bunch of call recordings handy—and who does?—you can [index](#) ([p. 217](#)) 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 opensearchpy import OpenSearch, 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)

search = OpenSearch(
    hosts = [{'host': host, 'port': 443}],
    http_auth = awsauth,
    use_ssl = True,
    verify_certs = True,
    connection_class = RequestsHttpConnection
)
response = search.bulk(bulk_file)
print(json.dumps(response, indent=2, sort_keys=True))
```

2. Update the initial two variables for host and region.
3. Install the required package using the following command:

```
pip install opensearch-py
```

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:

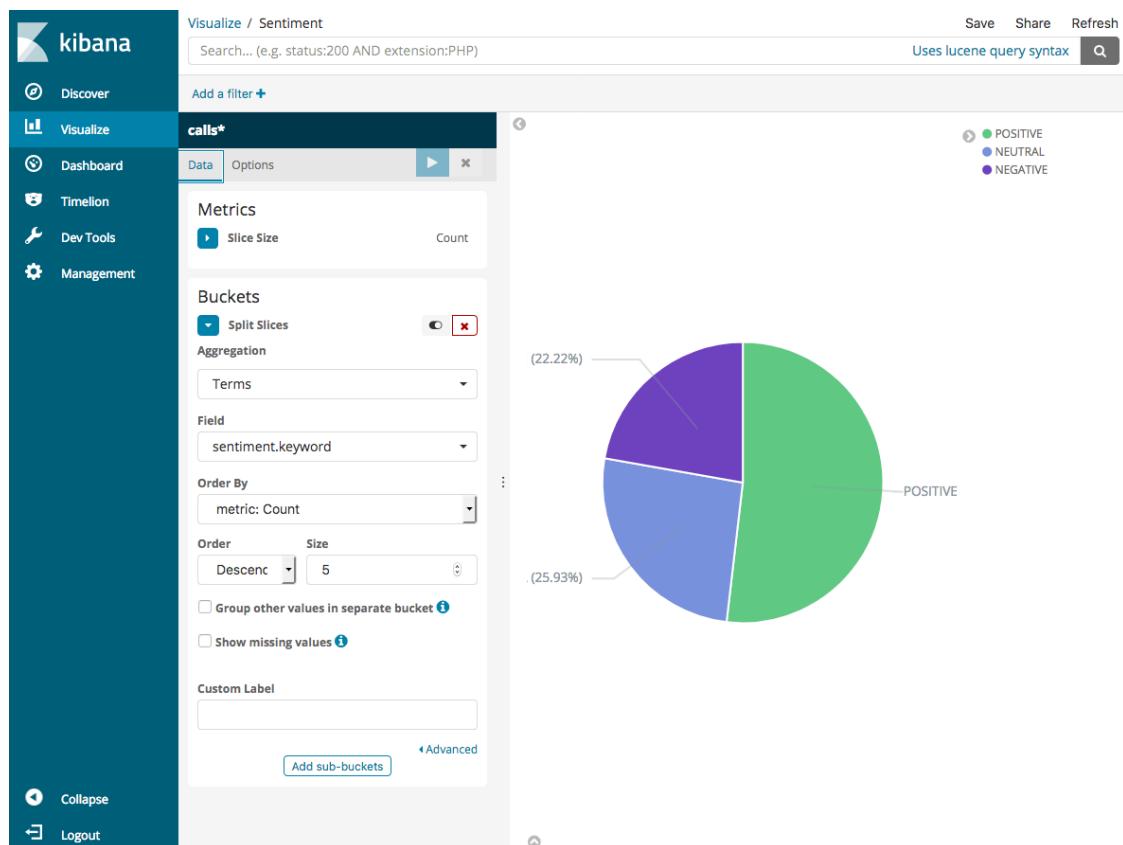
```
$ python bulk-helper.py
{
  "errors": false,
  "items": [
    {
      "index": {
        "_id": "1",
        "_index": "support-calls",
        "_primary_term": 1,
        "_seq_no": 42,
        "_shards": {
          "failed": 0,
          "successful": 1,
          "total": 2
        },
        "_type": "_doc",
        "_version": 9,
        "result": "updated",
        "status": 200
      }
    },
    ...
  ],
  "took": 27
}
```

Step 4: Analyze and visualize your data

Now that you have some data in OpenSearch Service, you can visualize it using OpenSearch Dashboards.

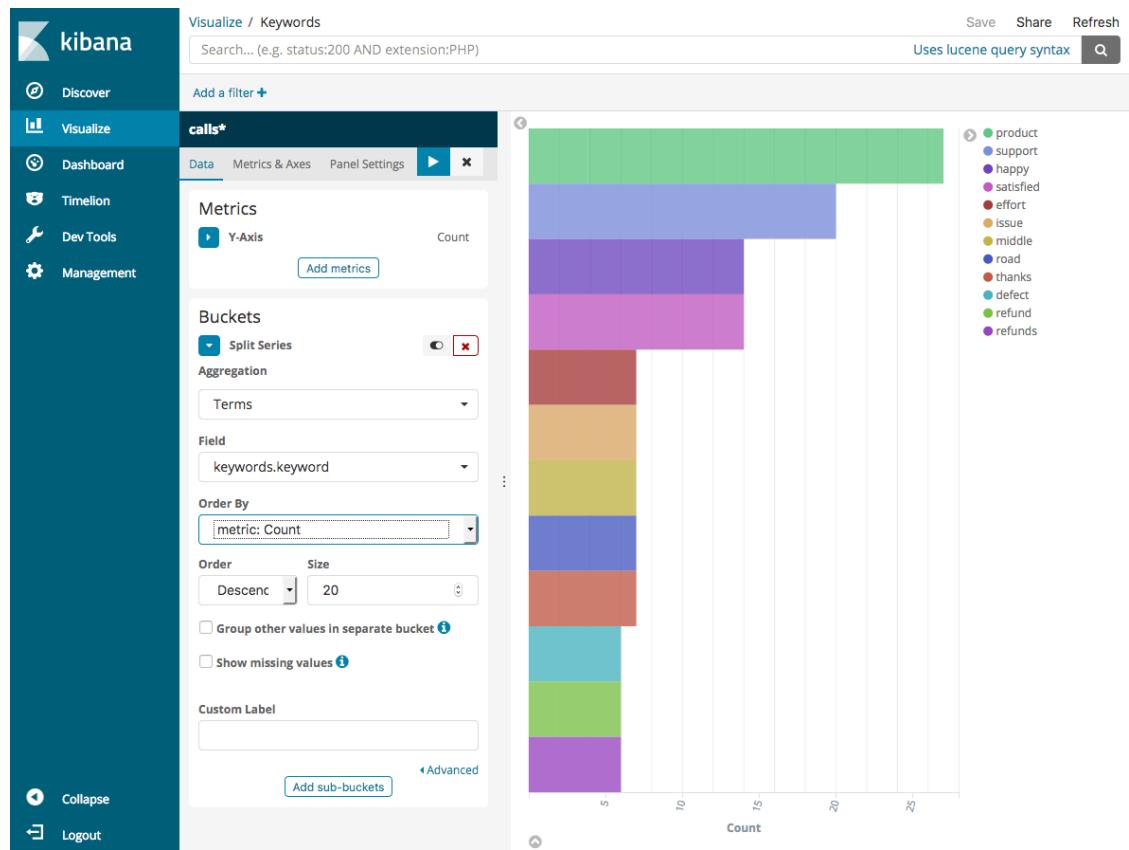
1. Navigate to [https://search-*domain.region*.es.amazonaws.com/_dashboards](https://search-<i>domain.region</i>.es.amazonaws.com/_dashboards).
2. Before you can use OpenSearch Dashboards, you need an index pattern. Dashboards uses index patterns to narrow your analysis to one or more indices. To match the support-calls index that call-center.py created, go to **Stack Management, Index Patterns**, and 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**.

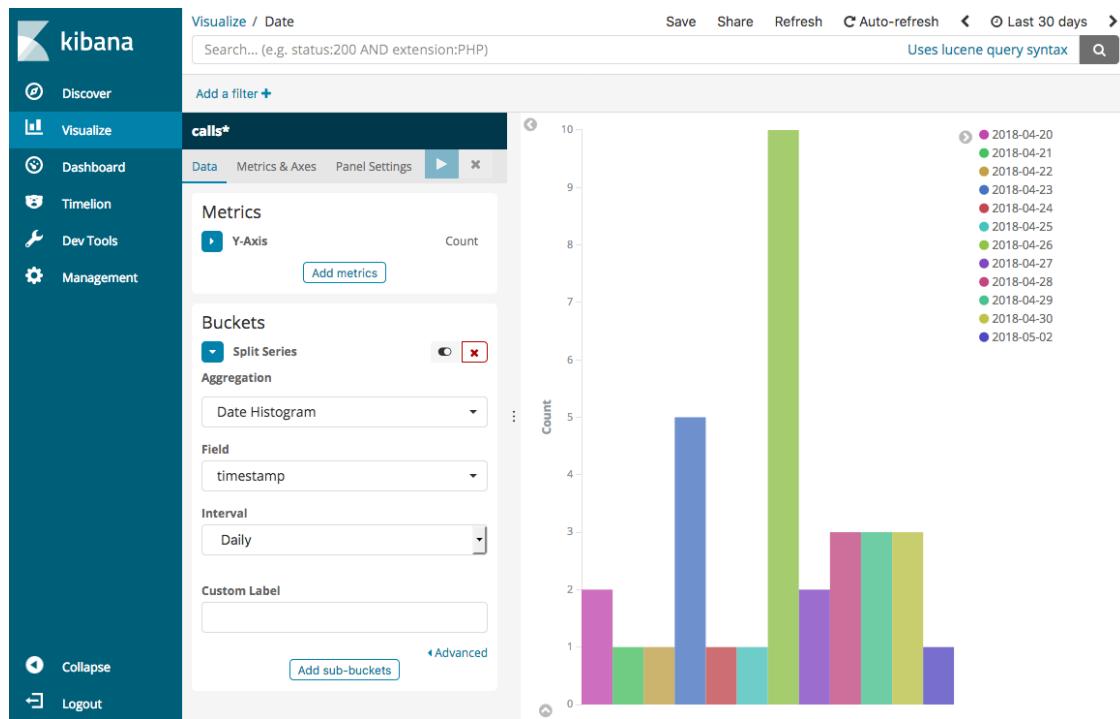


7. Return to the **Visualize** page, and add another visualization. This time, choose the horizontal bar chart.
8. Choose **Split Series**.

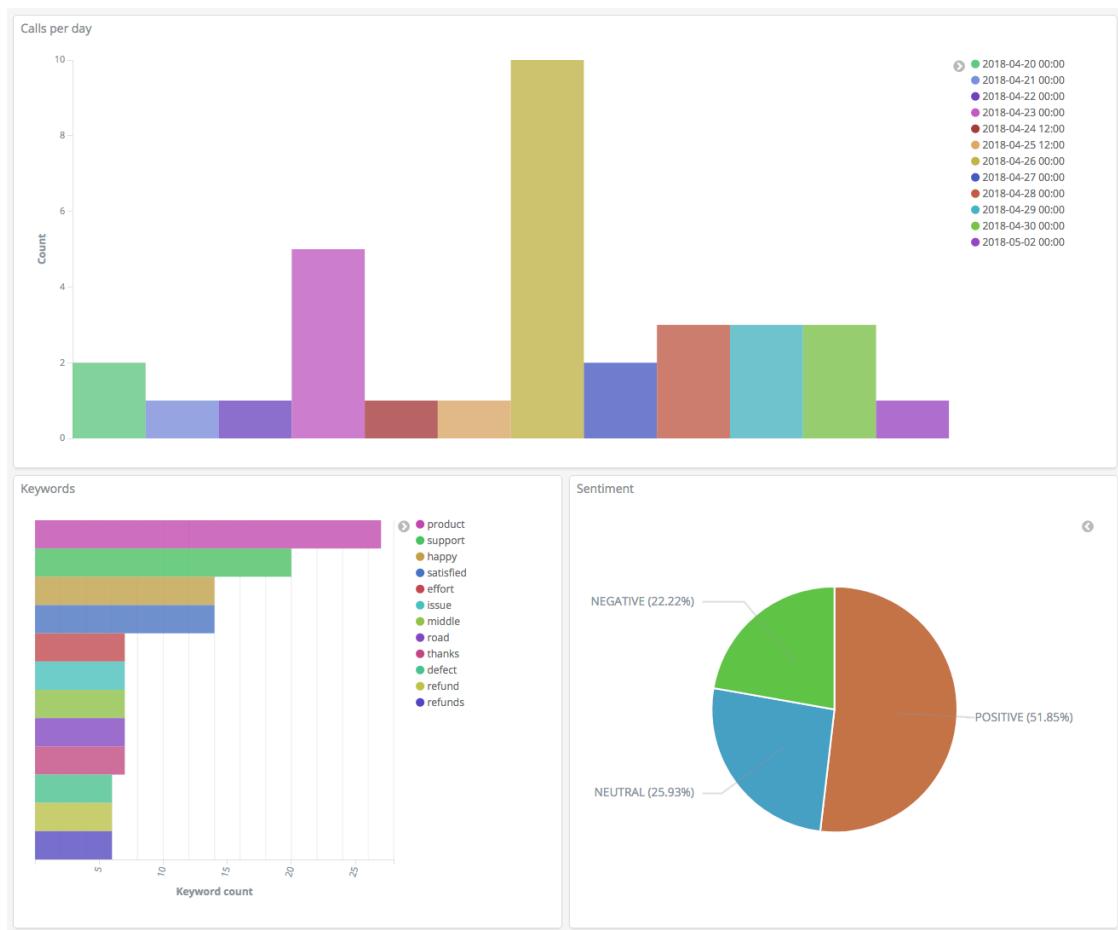
For **Aggregation**, choose **Terms**. For **Field**, choose **keywords.keyword** and change **Size** to 20. Then choose **Apply Changes** and **Save**.



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 Dashboards visualization. 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 OpenSearch Service domain. To learn more, see [Delete a Bucket](#) in the *Amazon Simple Storage Service User Guide* and [Delete an OpenSearch Service domain \(p. 15\)](#) 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.

Troubleshooting Amazon OpenSearch Service

This topic describes how to identify and solve common Amazon OpenSearch Service issues. Consult the information in this section before contacting [AWS Support](#).

Can't access OpenSearch Dashboards

The OpenSearch Dashboards 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 [Amazon Cognito authentication](#) (p. 175), you might receive the following error when you attempt to access Dashboards:

```
"User: anonymous is not authorized to perform: es:ESHttpGet"
```

If your OpenSearch Service domain uses VPC access, you might not receive this error, but 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 OpenSearch Dashboards" (p. 280), the section called "About access policies on VPC domains" (p. 39), and the section called "Identity and Access Management" (p. 130).

Can't access VPC domain

See the section called "About access policies on VPC domains" (p. 39) and the section called "Testing VPC domains" (p. 40).

Cluster in read-only state

Compared to earlier Elasticsearch versions, OpenSearch and Elasticsearch 7.x use 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, OpenSearch Service 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 indexes 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” \(p. 437\)](#) for troubleshooting steps. If cluster health is green, check that all expected indexes 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, OpenSearch Service 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, OpenSearch Service sends two events to your [AWS Health Dashboard](#). The first informs you of the loss of quorum. The second occurs after OpenSearch Service successfully restores quorum. For more information about using the AWS 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. OpenSearch Service keeps trying to take automated snapshots of all indexes regardless of their status, but the snapshots fail while the red cluster status persists.

The most common causes of a red cluster status are [failed cluster nodes \(p. 441\)](#) and the OpenSearch process crashing due to a continuous heavy processing load.

Note

OpenSearch Service stores automated snapshots for 14 days regardless of the cluster status. Therefore, if the red cluster status persists for more than two weeks, the last healthy automated snapshot will be deleted and you could permanently lose your cluster's data. If your OpenSearch Service 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 \(p. 361\)](#) to notify you when a red cluster status occurs.

Ultimately, red shards cause red clusters, and red indexes cause red shards. To identify the indexes causing the red cluster status, OpenSearch 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	sdIx_s_WDT56affFGu5KPbFQ	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		
	24.3kb	24.3kb	_8C6MIX0SxCqVYicH3jsEA	1	0	7	0

Deleting red indexes is the fastest way to fix a red cluster status. Depending on the reason for the red cluster status, you might then scale your OpenSearch Service domain to use larger instance types, more instances, or more EBS-based storage and try to recreate the problematic indexes.

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 indexes to free up disk space. The important step is to resolve the red cluster status *before* reconfiguring your OpenSearch Service 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.

Automatic remediation of red clusters

If your cluster's status is continuously red for more than an hour, OpenSearch Service attempts to automatically fix it by rerouting unallocated shards or restoring from past snapshots.

If it fails to fix one or more red indexes and the cluster status remains red for a total of 14 days, OpenSearch Service takes further action only if the cluster meets *at least one* of the following criteria:

- Has only one availability zone
- Has no dedicated master nodes
- Contains burstable instance types (T2 or T3)

At this time, if your cluster meets one of these criteria, OpenSearch Service sends you daily [notifications \(p. 32\)](#) over the next 7 days explaining that if you don't fix these indexes, all unassigned shards will be deleted. If your cluster status is still red after 21 days, OpenSearch Service deletes the unassigned shards (storage and compute) on all red indexes. You receive notifications in the **Notifications** panel of the OpenSearch Service console for each of these events. For more information, see the section called "Cluster health events" (p. 112).

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	Set memory circuit breakers for the JVM. For more information, see the section called "JVM OutOfMemoryError" (p. 441).

Relevant metric	Description	Recovery
	<p>as the Java garbage collector fails to reclaim sufficient memory. This pattern likely is due to complex queries or large data fields.</p> <p>x86 instance types use the Concurrent Mark Sweep (CMS) garbage collector, which runs alongside application threads to keep pauses short. If CMS is unable to reclaim enough memory during its normal collections, it triggers a full garbage collection, which can lead to long application pauses and impact cluster stability.</p> <p>ARM-based Graviton instance types use the Garbage-First (G1) garbage collector, which is similar to CMS, but uses additional short pauses and heap defragmentation to further reduce the need for full garbage collections.</p> <p>In either case, if memory usage continues to grow beyond what the garbage collector can reclaim during full garbage collections, OpenSearch crashes with an out of memory error. On all instance types, a good rule of thumb is to keep usage below 80%.</p> <p>The <code>_nodes/stats/jvm</code> API offers a useful summary of JVM statistics, memory pool usage, and garbage collection information:</p> <pre>GET <i>domain-endpoint</i>/_nodes/stats/jvm?pretty</pre>	If the problem persists, delete unnecessary indexes, reduce the number or complexity of requests to the domain, add instances, or use larger instance types.
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 the primary shards for all indexes are allocated to nodes in a cluster, but the replica shards for at least one index aren't. Single-node clusters always initialize with a yellow

cluster status because there's no other node to which OpenSearch Service can assign a replica. To achieve green cluster status, increase your node count. For more information, see [the section called "Sizing domains" \(p. 353\)](#).

Multi-node clusters might briefly have a yellow cluster status after creating a new index or after a node failure. This status self-resolves as OpenSearch replicates data across the cluster. [Lack of disk space \(p. 440\)](#) can also cause yellow cluster status; the cluster can only distribute replica shards if nodes have the disk space to accommodate them.

ClusterBlockException

You might receive a ClusterBlockException error for the following reasons.

Lack of available storage space

If one or more nodes in your cluster has less than 20% of available storage space, or less than 20 GB of storage space, basic write operations like adding documents and creating indexes can start to fail. [the section called "Calculating storage requirements" \(p. 353\)](#) provides a summary of how OpenSearch Service uses disk space.

To avoid issues, monitor the FreeStorageSpace metric in the OpenSearch Service console and [create CloudWatch alarms \(p. 361\)](#) 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 OpenSearch Service domain to use larger instance types, more instances, or more EBS-based storage.

High JVM memory pressure

When the **JVMMemoryPressure** metric exceeds 92% for 30 minutes, OpenSearch Service 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 indexes 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.

High JVM memory pressure can be caused by spikes in the number of requests to the cluster, unbalanced shard allocations across nodes, too many shards in a cluster, field data or index mapping explosions, or instance types that can't handle incoming loads. It can also be caused by using aggregations, wildcards, or wide time ranges in queries.

To reduce traffic to the cluster and resolve high JVM memory pressure issues, try one or more of the following:

- Scale the domain so that the maximum heap size per node is 32 GB.
- Reduce the number of shards by deleting old or unused indexes.
- Clear the data cache with the POST `index-name/_cache/clear?fielddata=true` API operation. Note that clearing the cache can disrupt in-progress queries.

In general, to avoid high JVM memory pressure in the future, follow these best practices:

- Avoid aggregating on text fields, or change the [mapping type](#) for your indexes to keyword.

- Optimize search and indexing requests by [choosing the correct number of shards \(p. 355\)](#).
- Set up Index State Management (ISM) policies to regularly [remove unused indexes \(p. 350\)](#).

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 95%.	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 40%. If you upload data with large fields, you might need to raise 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 60%. If your service requests involve calculating aggregations, you might need to raise this limit.	indices.breaker.request.limit

Failed cluster nodes

Amazon EC2 instances might experience unexpected terminations and restarts. Typically, OpenSearch Service restarts the nodes for you. However, it's possible for one or more nodes in an OpenSearch cluster to remain in a failed condition.

To check for this condition, open your domain dashboard on the OpenSearch Service console. Go to the **Cluster health** tab and find the **Total 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 \(p. 361\)](#) to notify you when this issue occurs.

Note

The **Total 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" \(p. 22\)](#).

To protect your clusters from unexpected node terminations and restarts, create at least one replica for each index in your OpenSearch Service domain.

Exceeded maximum shard limit

OpenSearch as well as 7.x versions of Elasticsearch have a default setting of no more than 1,000 shards per node. OpenSearch/Elasticsearch throw 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 \(p. 375\)](#) to reduce the number of shards on the node.

Domain stuck in processing state

Your OpenSearch Service domain enters the "Processing" state when it's in the middle of a [configuration change \(p. 22\)](#). When you initiate a configuration change, the domain status changes to "Processing" while OpenSearch Service creates a new environment. In the new environment, OpenSearch Service launches a new set of applicable nodes (such as data, master, or UltraWarm). After the migration completes, the older nodes are terminated.

The cluster can get stuck in the "Processing" state if either of these situations occurs:

- A new set of data nodes fails to launch.
- Shard migration to the new set of data nodes is unsuccessful.
- Validation check has failed with errors.

For detailed resolution steps in each of these situations, see [Why is my Amazon OpenSearch Service domain stuck in the "Processing" state?](#).

Low EBS burst balance

OpenSearch Service sends you a console notification when the EBS burst balance on one of your General Purpose (SSD) volumes is below 70%, and a follow-up notification if the balance falls below 20%. To fix this issue, you can either scale up your cluster, or reduce the read and write IOPS so that the burst balance can be credited. For more information, see [General Purpose SSD volumes \(gp2\)](#). You can monitor EBS burst balance with the `BurstBalance` CloudWatch metric.

Can't enable audit logs

You might encounter the following error when you try to enable audit log publishing using the OpenSearch Service console:

The Resource Access Policy specified for the CloudWatch Logs log group does not grant sufficient permissions for Amazon OpenSearch Service to create a log stream. Please check the Resource Access Policy.

If you encounter this error, verify that the `resource` element of your policy includes the correct log group ARN. If it does, take the following steps:

1. Wait several minutes.
2. Refresh the page in your web browser.
3. Choose **Select existing group**.
4. For **Existing log group**, choose the log group that you created before receiving the error message.
5. In the access policy section, choose **Select existing policy**.
6. For **Existing policy**, choose the policy that you created before receiving the error message.
7. Choose **Enable**.

If the error persists after repeating the process several times, contact [AWS Support](#).

Can't close index

OpenSearch Service supports the `_close` API only for OpenSearch and Elasticsearch versions 7.4 and later. If you're using an older version and 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.

Client license checks

The default distributions of Logstash and Beats include a proprietary license check and fail to connect to the open source version of OpenSearch. Make sure you use the Apache 2.0 (OSS) distributions of these clients with OpenSearch Service.

Request throttling

If you receive persistent 403 Request throttled due to too many requests or 429 Too Many Requests errors, consider scaling vertically. Amazon OpenSearch Service throttles requests if the payload would cause memory usage to exceed the maximum size of the Java heap.

Can't SSH into node

You can't use SSH to access any of the nodes in your OpenSearch cluster, and you can't directly modify `opensearch.yml`. Instead, use the console, AWS CLI, or SDKs to configure your domain. You can specify a few cluster-level settings using the OpenSearch REST APIs, as well. To learn more, see the [Amazon OpenSearch Service API Reference](#) and the section called **"Supported operations"** (p. 373).

If you need more insight into the performance of the cluster, you can [publish error logs and slow logs to CloudWatch \(p. 92\)](#).

"Not Valid for the Object's Storage Class" snapshot error

OpenSearch Service 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 \(p. 45\)](#) as a snapshot repository.

Invalid host header

OpenSearch Service 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 or proxy includes the OpenSearch Service domain endpoint (and not, for example, its IP address) in the Host header.

Invalid M3 instance type

OpenSearch Service doesn't support adding or modifying M3 instances to existing domains running OpenSearch or Elasticsearch versions 6.7 and later. You can continue to use M3 instances with Elasticsearch 6.5 and earlier.

We recommend choosing a newer instance type. For domains running OpenSearch or Elasticsearch 6.7 or later, the following restriction apply:

- If your existing domain does not use M3 instances, you can no longer change to them.
- If you change an existing domain from an M3 instance type to another instance type, you can't switch back.

Hot queries stop working after enabling UltraWarm

When you enable UltraWarm on a domain, if there are no preexisting overrides to the `search.max_buckets` setting, OpenSearch Service automatically sets the value to 10000 to prevent memory-heavy queries from saturating warm nodes. If your hot queries are using more than 10,000 buckets, they might stop working when you enable UltraWarm.

Because you can't modify this setting due to the managed nature of Amazon OpenSearch Service, you need to open a support case to increase the limit. Limit increases don't require a premium support subscription.

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 \(p. 42\)](#).

Need summary of domains for all AWS Regions

The following script uses the Amazon EC2 [describe-regions](#) AWS CLI command to create a list of all Regions in which OpenSearch Service 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 opensearch 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 OpenSearch Service is not available return "Could not connect to the endpoint URL."

Browser error when using OpenSearch Dashboards

Your browser wraps service error messages in HTTP response objects when you use Dashboards to view data in your OpenSearch Service 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 Chrome top menu bar, 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.

Node shard and storage skew

Node shard skew is when one or more nodes within a cluster has significantly more shards than the other nodes. **Node storage skew** is when one or more nodes within a cluster has significantly more storage (`disk.indices`) than the other nodes. While both of these conditions can occur temporarily, like when a domain has replaced a node and is still allocating shards to it, you should address them if they persist.

To identify both types of skew, run the [_cat/allocation](#) API operation and compare the shards and `disk.indices` entries in the response:

shards	host	ip	disk.indices	disk.used	disk.avail	disk.total	disk.percent
264	x.x.x.x	x.x.x.x	465.3mb	229.9mb	1.4tb	1.5tb	0
115	x.x.x.x	x.x.x.x	7.9mb	83.7mb	49.1gb	49.2gb	0
264	x.x.x.x	x.x.x.x	465.3mb	235.3mb	1.4tb	1.5tb	0
116	x.x.x.x	x.x.x.x	7.9mb	82.8mb	49.1gb	49.2gb	0
115	x.x.x.x	x.x.x.x	8.4mb	85mb	49.1gb	49.2gb	0
	x.x.x.x	x.x.x.x	node5				

While some storage skew is normal, anything over 10% from the average is significant. When shard distribution is skewed, CPU, network, and disk bandwidth usage can also become skewed. Because more data generally means more indexing and search operations, the heaviest nodes also tend to be the most resource-strained nodes, while the lighter nodes represent underutilized capacity.

Remediation: Use shard counts that are multiples of the data node count to ensure that each index is distributed evenly across data nodes.

Index shard and storage skew

Index shard skew is when one or more nodes hold more of an index's shards than the other nodes. **Index storage skew** is when one or more nodes hold a disproportionately large amount of an index's total storage.

Index skew is harder to identify than node skew because it requires some manipulation of the [_cat/shards](#) API output. Investigate index skew if there's some indication of skew in the cluster or node metrics. The following are common indications of index skew:

- HTTP 429 errors occurring on a subset of data nodes
- Uneven index or search operation queueing across data nodes
- Uneven JVM heap and/or CPU utilization across data nodes

Remediation: Use shard counts that are multiples of the data node count to ensure that each index is distributed evenly across data nodes. If you still see index storage or shard skew, you might need to force a shard reallocation, which occurs with every [blue/green deployment \(p. 22\)](#) of your OpenSearch Service domain.

Unauthorized operation after selecting VPC access

When you create a new domain using the OpenSearch Service console, you have the option to select VPC or public access. If you select **VPC access**, OpenSearch Service 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, OpenSearch Service needs to assume the `AWSServiceRoleForAmazonOpenSearchService` 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](#).

Denied requests to the OpenSearch API

With the introduction of tag-based access control for the OpenSearch API, you might start seeing access denied errors where you didn't before. This might be because one or more of your access policies contains Deny using the `ResourceTag` condition, and those conditions are now being honored.

For example, the following policy used to only deny access to the `CreateDomain` action from the configuration API, if the domain had the tag `environment=production`. Even though the action list also includes `ESHttpPut`, the deny statement didn't apply to that action or any other `ESHttp*` actions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "es>CreateDomain",
        "es:ESHttpPut"
      ],
      "Effect": "Deny",
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:ResourceTag/environment": [
            "production"
          ]
        }
      }
    }
  ]
}
```

With the added support of tags for OpenSearch HTTP methods, an IAM identity-based policy like the above will result in the attached user being denied access to the ESHttpPut action. Previously, in the absence of tags validation, the attached user would have still been able to send PUT requests.

If you start seeing access denied errors after updating your domains to service software R20220323 or later, check your identity-based access policies to see if this is the case and update them if necessary to allow access.

Can't connect from Alpine Linux

Alpine Linux limits DNS response size to 512 bytes. If you try to connect to your OpenSearch Service 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 OpenSearch
...
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, OpenSearch Service 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.

Document history for Amazon OpenSearch Service

This topic describes important changes to Amazon OpenSearch Service. 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" \(p. 29\)](#).

Service features are rolled out incrementally to the AWS Regions where a service is available. We update this documentation for the first release only. We don't provide information about Region availability or announce subsequent Region rollouts. For information about Region availability of service features, and to subscribe to notifications about updates, see [What's New with AWS?](#)

Relevant dates to this history:

- **Current product version**—2021-01-01
- **Latest product release**—November 15, 2022
- **Latest documentation update**—November 15, 2022

For notifications about updates, you can subscribe to the RSS feed.

Note

Patch releases: Service software versions that end in "-P" and a number, such as R20211203-P4, are patch releases. Patches are likely to include performance improvements, minor bug fixes, and security fixes or posture improvements. Since patches do not include new features or breaking changes, they generally do not have direct user or documentation impact, which is why the specifics of each patch are not included in this document history.

Change	Description	Date
OpenSearch 2.3 support	Amazon OpenSearch Service now supports OpenSearch version 2.3. This version includes all features that were part of versions 2.0, 2.1, and 2.2. For more information, see the 2.0 , 2.1 , 2.2 , and 2.3 release notes. Version 2.3 contains a breaking change . For more information, see Supported upgrade paths .	November 15, 2022
Kibana 7.1.1 support	Amazon OpenSearch Service domains running Elasticsearch 7.1 now support the latest patch release for Kibana 7.1.1, which adds bug fixes and improves security. When you update your 7.1 domains to service software R20221114, OpenSearch Service will automatically upgrade them to this patch release.	November 15, 2022

Kibana 6.8.13 support	Amazon OpenSearch Service domains running Elasticsearch 6.8 now support the latest patch release for Kibana 6.8.13, which adds bug fixes and improves security. When you update your 6.8 domains to service software R20221114, OpenSearch Service will automatically upgrade them to this patch release.	November 15, 2022
Kibana 6.3.2 support	Amazon OpenSearch Service domains running Elasticsearch 6.3 now support the latest patch release for Kibana 6.3.2, which adds bug fixes and improves security. When you update your 6.3 domains to service software R20221114, OpenSearch Service will automatically upgrade them to this patch release.	November 15, 2022
AWS PrivateLink	With Amazon OpenSearch Service-managed VPC endpoints, you can connect directly to OpenSearch Service VPC domains by using an interface VPC endpoint instead of connecting over the internet. An OpenSearch Service-managed VPC endpoint is accessible only within the VPC where the endpoint is provisioned, or from any VPCs peered with the VPC where the endpoint is provisioned, as permitted by the route tables and security groups. Your VPC domain must be running service software R20220928 or later to connect to an interface VPC endpoint.	November 7, 2022
Bug fixes and performance improvements (p. 450)	Service software R20220928 includes bug fixes and performance enhancements, including improved SAML logging. The update also changes the default tenant to Global rather than Private.	October 3, 2022

Improved API reference	Amazon OpenSearch Service offers an improved, all-encompassing configuration API reference. The new references contains all available actions and data types, sample request and response syntax, and links to the corresponding SDK references for all supported languages.	September 13, 2022
Blue/green validation	Amazon OpenSearch Service now performs a validation check prior to blue/green deployments, and surfaces validation errors if your domain is not eligible for an update.	August 16, 2022
OpenSearch 1.3 support	Amazon OpenSearch Service now supports OpenSearch version 1.3. For more information, see the 1.3 release notes .	July 27, 2022
ML Commons plugin support	Amazon OpenSearch Service adds support for the ML Commons plugin, which provides a set of common machine learning algorithms through transport and REST API calls . You can also interact with the ML Commons plugin through PPL commands.	July 27, 2022
gp3 volume support	Amazon OpenSearch Service adds support for the gp3 EBS General Purpose SSD volume type. You can specify additional provisioned IOPS and throughput when you create or modify the domain.	July 26, 2022
Enhanced best practices documentation	The Amazon OpenSearch Service documentation provides improved operational best practices and general recommendations for creating and operating OpenSearch Service domains.	July 6, 2022
Integration with Service Quotas	You can now view quotas for Amazon OpenSearch Service, and request quota increases, from the Service Quotas console.	June 29, 2022

Tag-based access control for the OpenSearch API	You can now use tags to control access to the OpenSearch APIs. Previously, you could only use tags to control access to the configuration API.	June 16, 2022
Cross-cluster search across Regions (p. 450)	Cross-cluster search is now supported across AWS Regions as long as both domains are running Elasticsearch version 7.10 or later, or any version of OpenSearch.	June 14, 2022
Single Kibana 5.6 support	Amazon OpenSearch Service adds support for single Kibana 5.6.16. With single Kibana 5.6.16, you can use Kibana 5.6 as your front end while connecting to Elasticsearch versions 5.1, 5.3, 5.5, and 5.6. You must be on service software R20220323 or later to use single Kibana 5.6.	April 4, 2022
R20220323-P1 (p. 450)	Amazon OpenSearch Service recently released service software update R20220323, but the update was subsequently rolled back because of an issue. We recommend that you update your domains to patch release R20220323-P1 or later, which fixes the issue.	April 4, 2022
OpenSearch 1.2 support	Amazon OpenSearch Service now supports OpenSearch version 1.2. For more information, see the 1.2 release notes .	April 4, 2022
Observability	The default installation of OpenSearch Dashboards for Amazon OpenSearch Service includes the Observability plugin, which you can use to visualize data-driven events using Piped Processing Language (PPL) to explore and query your data. The plugin requires OpenSearch 1.2 or later and service software R20220323 or later.	April 4, 2022

Kibana 7.7.1 support	Amazon OpenSearch Service domains running Elasticsearch 7.7 now support the latest patch release for Kibana 7.7, which adds bug fixes and improves security. When you update your 7.7 domains to service software R20220323 or later, OpenSearch Service will automatically upgrade them to this patch release.	April 4, 2022
JVM memory pressure metric changes	Amazon OpenSearch Service changed the logic for the <code>JVMMemoryPressure</code> CloudWatch metrics to more accurately reflect memory utilization. Previously, the metrics only considered the old generation memory pool of JVM heap. With this change, the metric also considers the young generation memory pool. After you update your domain to service software R20220323, you might see an increase in the <code>JVMMemoryPressure</code> , <code>MasterJVMMemoryPressure</code> , and/or <code>WarmJVMMemoryPressure</code> metrics.	April 4, 2022
Custom dictionaries with the IK (Chinese) Analysis plugin	Amazon OpenSearch Service now supports using custom dictionaries with the IK (Chinese) Analysis plugin.	April 4, 2022
Cross-cluster replication on existing domains (p. 450)	Amazon OpenSearch Service removed the limitation that you can only implement cross-cluster search and cross-cluster replication on domains created on or after June 3rd, 2020. You can now enable these features on all domains regardless of when they were created. Both domains must be on service software R20220323 or later.	April 4, 2022
Blue/green deployment visibility	Amazon OpenSearch Service now offers more visibility into the progress of blue/green deployments. You can monitor these details in the console or using the configuration API.	January 27, 2022

Fine-grained access control on existing domains	You can now enable fine-grained access control on existing domains. You can enable a temporary migration period for open/IP-based access policies to ensure that users can continue to access your domain while you create and map roles. Enabling fine-grained access control on existing domains requires service software R20211203 or later.	January 6, 2022
Renamed OpenSearch Dashboards roles	With service software R20211203, the kibana_user role was renamed to opensearch_dashboards_user, and kibana_read_only was renamed to opensearch_dashboards_read_only. This change applies to all <i>newly-created</i> OpenSearch 1.x domains. For existing OpenSearch domains that you upgrade to service software R20211203, the roles remain the same.	January 4, 2022
OpenSearch 1.1 support	Amazon OpenSearch Service now supports OpenSearch version 1.1. For more information, see the 1.1 release notes .	January 4, 2022
ISM visual editor	The default installation of OpenSearch Dashboards for Amazon OpenSearch Service now supports the visual editor for ISM policies. This feature requires OpenSearch 1.1 or later.	January 4, 2022
Cross-service confused deputy prevention update	Amazon OpenSearch Service supports using the aws:SourceArn and aws:SourceAccount global condition context keys in IAM resource policies to prevent the confused deputy problem. You must be on service software R20211203 or later to use these condition keys.	January 4, 2022

Log4j patch (p. 450)	Service software R20211203-P2 updates the version of Log4j used in OpenSearch Service as recommended by the advisories in CVE-2021-44228 and CVE-2021-45046 . The patch applies to domains running all versions of OpenSearch and Elasticsearch. OpenSearch Service will continue to update various Log4j versions internally, and they will not necessarily be restricted to the latest version of Log4j. The Log4j version on your domain depends on the software version that the domain is running. However, irrespective of the Log4j version, as long as you're running R20211203-P2 or later, your domains contain the Log4j update required to address CVE-2021-44228 and CVE-2021-45046.	December 15, 2021
Cross-cluster replication	Cross-cluster replication lets you replicate indices, mappings, and metadata from one OpenSearch Service domain to another. Cross-cluster replication requires a domain running Elasticsearch 7.10 or OpenSearch 1.1 or later.	October 5, 2021
New AWS-managed policies	The launch of Amazon OpenSearch Service includes new AWS-managed policies and the deprecation of old policies.	September 8, 2021
Kibana 6.4.3 support	Amazon OpenSearch Service domains running legacy Elasticsearch version 6.4 now support the latest patch release for Kibana 6.4, which adds bug fixes and improves security. OpenSearch Service will automatically upgrade domains to this patch release.	September 8, 2021
Data streams	Amazon OpenSearch Service adds support for data streams, which simplify the process of managing time-series data. Your domain must be running OpenSearch 1.0 or later to use data streams.	September 8, 2021

Amazon OpenSearch Service	AWS introduces Amazon OpenSearch Service. Amazon OpenSearch Service supports OpenSearch and legacy Elasticsearch OSS. When you create a cluster, you have the option of which search engine to use. OpenSearch Service offers broad compatibility with Elasticsearch OSS 7.10, the final open source version of the software.	September 8, 2021
Cold storage	Cold storage is a new storage tier for infrequently accessed or historical data. Cold indices only occupy S3 storage and have no compute attached to them. Cold storage requires a domain running Elasticsearch 7.9 or later and service software R20210426 or later.	May 13, 2021
ARM-based Graviton instances	Amazon Elasticsearch Service now supports ARM-based Graviton instance types (M6G, C6G, R6G, and R6GD). Graviton instance types are available on new and existing domains running Elasticsearch 7.9 or later and service software R20210331 or later.	May 4, 2021
ISM templates	Amazon Elasticsearch Service adds support for ISM templates, which let you automatically attach an ISM policy to an index if the index matches a pattern defined in the policy. ISM templates require service software R20210426 or later. This update also deprecates the <code>policy_id</code> setting, meaning you can no longer use index templates to apply ISM policies to newly created indices. The update introduces a breaking change for existing CloudFormation templates using this setting.	April 27, 2021
Elasticsearch 7.10 support	Amazon Elasticsearch Service now supports Elasticsearch version 7.10. For more information, see 7.10 release notes .	April 21, 2021

Asynchronous search	Amazon Elasticsearch Service now supports asynchronous search, which lets you run search requests in the background. Asynchronous search requires a domain running Elasticsearch 7.10 or later and service software R20210331 or later.	April 21, 2021
Tag-based access control for the configuration API	You can now use AWS tags to control access to the Amazon ES configuration API.	March 2, 2021
Auto-Tune	Amazon Elasticsearch Service adds Auto-Tune, which uses performance and usage metrics from your cluster to suggest changes to the JVM settings on your nodes. Auto-Tune requires a domain running Elasticsearch 6.7 or later and service software R20201117 or later.	February 24, 2021
Trace Analytics	The default installation of Kibana for Amazon Elasticsearch Service now includes the trace analytics plugin, which lets you monitor trace data from your distributed applications. The plugin requires a domain running Elasticsearch 7.9 or later and service software R20210201 or later.	February 17, 2021
Shard metrics	Amazon Elasticsearch Service adds the following CloudWatch metrics for tracking shard status: Shards.active, Shards.unassigned, Shards.delayedUnassignedShards.activePrimary, Shards.initializing, Shards.relocating. The metrics are available on domains running service software R20210201 or later.	February 17, 2021
Kibana reports	The default installation of Kibana for Amazon Elasticsearch Service now supports on-demand reports for the Discover, Visualize, and Dashboard pages. This feature requires Elasticsearch 7.9 or later and service software R20210201 or later.	February 17, 2021

Kibana 5.6.16 support (p. 450)	Amazon Elasticsearch Service domains running Elasticsearch 5.6 now support the latest patch release for Kibana 5.6, which adds bug fixes and improves security. Amazon ES will automatically upgrade domains to this patch release.	February 17, 2021
Encryption for existing domains	Amazon Elasticsearch Service now supports enabling encryption of data at rest and node-to-node encryption on existing domains running Elasticsearch 6.7 or later. After you enable these settings, you can't disable them.	January 27, 2021
Remote reindex	Amazon Elasticsearch Service now supports remote reindex, which lets you migrate indices from remote domains. This feature requires service software R20201117 or later.	November 24, 2020
Piped Processing Language	Amazon Elasticsearch Service now supports Piped Processing Language (PPL), a query language that lets you use pipe () syntax to query data stored in Elasticsearch. This feature requires service software R20201117 or later. To learn more, see .	November 24, 2020
Kibana notebooks	Amazon Elasticsearch Service adds support for Kibana notebooks, which lets you combine live visualizations and narrative text in a single interface. This feature requires service software R20201117 or later.	November 24, 2020
Gantt charts	The default installation of Kibana for Amazon Elasticsearch Service now supports a new visualization type, Gantt charts. This feature requires service software R20201117 or later.	November 24, 2020
Elasticsearch 7.9 support	Amazon Elasticsearch Service now supports Elasticsearch version 7.9. For more information, see 7.9 release notes .	November 24, 2020

Anomaly detection updates	Anomaly detection for Amazon Elasticsearch Service adds support for high cardinality, which lets you categorize anomalies with a dimension like IP address, product ID, country code, and so on. This feature requires service software R20201117 or later.	November 24, 2020
Dynamic dictionary updates	Amazon Elasticsearch Service now lets you update your search analyzers without reindexing. You can update the dictionary files on some or all of your domains, and Amazon ES tracks package versions over time so that you have a history of what changed and when. This feature requires service software R20201019 or later.	November 17, 2020
Custom endpoints	Amazon Elasticsearch Service now supports custom endpoints, which let you give your Amazon ES domain a new URL. If you ever swap domains, you can maintain the same URL. This feature requires service software R20201019 or later.	November 5, 2020
New language plugins	Amazon Elasticsearch Service now supports IK (Chinese) Analysis, Vietnamese Analysis, and Thai Analysis plugins on domains running Elasticsearch 7.7 or later with service software R20201019 or later.	October 28, 2020
Elasticsearch 7.8 support	Amazon Elasticsearch Service now supports Elasticsearch version 7.8. For more information, see 7.8 release notes .	October 28, 2020
SAML authentication for Kibana	Amazon Elasticsearch Service now supports SAML authentication for Kibana, which lets you use third-party identity providers to log in to Kibana, manage fine-grained access control, search your data, and build visualizations. This feature requires service software R20201019 or later.	October 27, 2020

T3 instances	Amazon Elasticsearch Service now supports the <code>t3.small</code> and <code>t3.medium</code> instance types.	September 23, 2020
Audit logs	Amazon Elasticsearch Service now supports audit logs for your data, which lets you track failed login attempts, user access to indices, documents, and fields, and much more. This feature requires service software R20200910 or later.	September 16, 2020
UltraWarm updates	UltraWarm for Amazon Elasticsearch Service adds new metrics, new settings, a larger migration queue, and a cancellation API. These updates require service software R20200910 or later. For more information, see .	September 14, 2020
Learning to Rank	Amazon Elasticsearch Service now supports the open source Learning to Rank plugin, which lets you use machine learning technologies to improve search relevance. This feature requires service software R20200721 or later.	July 27, 2020
k-NN cosine similarity	k-Nearest Neighbor (k-NN) now lets you search for "nearest neighbors" by cosine similarity in addition to Euclidean distance. This feature requires service software R20200721 or later.	July 23, 2020
gzip compression	Amazon Elasticsearch Service now supports gzip compression for most HTTP requests and responses, which can reduce latency and conserve bandwidth. This feature requires service software R20200721 or later.	July 23, 2020
Elasticsearch 7.7 support	Amazon Elasticsearch Service now supports Elasticsearch version 7.7. For more information, see 7.7 release notes .	July 23, 2020
Kibana map service	The default installation of Kibana for Amazon Elasticsearch Service now includes a WMS map server, except for domains in the India and China Regions.	June 18, 2020

SQL improvements	SQL support for Amazon Elasticsearch Service now supports many new operations, a dedicated Kibana user interface for data exploration, and an interactive CLI. For more information, see .	June 3, 2020
Cross-cluster search	Amazon Elasticsearch Service lets you perform cross-cluster queries and aggregations across multiple connected domains.	June 3, 2020
Anomaly detection	Amazon Elasticsearch Service lets you automatically detect anomalies in near-real time.	June 3, 2020
UltraWarm	UltraWarm storage for Amazon Elasticsearch Service has left public preview and is now generally available. The feature now supports a wider range of versions and AWS Regions. For more information, see .	May 5, 2020
Custom dictionaries	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.	April 21, 2020
Elasticsearch 7.4 Support	Amazon Elasticsearch Service now supports Elasticsearch version 7.4. For more information, see Supported versions .	March 12, 2020
k-NN	Amazon Elasticsearch Service adds support for k-Nearest Neighbor (k-NN) search. k-NN requires service software R20200302 or later.	March 3, 2020
Index State Management	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.	March 3, 2020

Elasticsearch 5.6.16 support (p. 450)	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	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)	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.	December 3, 2019
Encryption features for China Regions (p. 450)	Encryption of data at rest and node-to-node encryption are now available in the cn-north-1 China (Beijing) Region and cn-northwest-1 China (Ningxia) Region.	November 20, 2019
Require HTTPS (p. 450)	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	Amazon Elasticsearch Service now supports Elasticsearch version 7.1 and 6.8. For more information, see Supported versions .	August 13, 2019

Hourly snapshots	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.	July 8, 2019
Elasticsearch 6.7 support	Amazon Elasticsearch Service now supports Elasticsearch version 6.7. For more information, see Supported versions .	May 29, 2019
SQL support	Amazon Elasticsearch Service now lets you query your data using SQL. SQL support requires service software R20190418 or later.	May 15, 2019
5-series instance types	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 Limits .	April 24, 2019
Elasticsearch 6.5 support	Amazon Elasticsearch Service now supports Elasticsearch version 6.5.	April 8, 2019
Alerting	Alerting for Amazon Elasticsearch Service notifies you when data from one or more Amazon ES indices meets certain conditions. Alerting requires service software R20190221 or later.	March 25, 2019
Three Availability Zone support	Amazon Elasticsearch Service now supports three Availability Zones in many Regions. This release also includes a streamlined console experience. This multi-AZ requires service software R20181023 or later.	February 7, 2019
Elasticsearch 6.4 support	Amazon Elasticsearch Service now supports Elasticsearch version 6.4.	January 23, 2019
200-node clusters	Amazon ES now lets you create clusters with up to 200 data nodes for a total of 3 PB of storage.	January 22, 2019

Service software updates	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 .	November 20, 2018
New CloudWatch metrics	Amazon ES now offers node-level metrics and new Cluster health and Instance health tabs in the Amazon ES console.	November 20, 2018
China (Beijing) support (p. 450)	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	Amazon Elasticsearch Service now supports node-to-node encryption, which keeps your data encrypted as Amazon ES distributes it throughout your cluster.	September 18, 2018
In-place version upgrades	Amazon Elasticsearch Service now supports in-place version upgrades.	August 14, 2018
Elasticsearch 6.3 and 5.6 support	Amazon Elasticsearch Service now supports Elasticsearch version 6.3 and 5.6.	August 14, 2018
Error logs	Amazon ES now lets you publish Elasticsearch error logs to Amazon CloudWatch.	July 31, 2018
China (Ningxia) Reserved Instances (p. 450)	Amazon ES now offers Reserved Instances in the China (Ningxia) Region.	May 29, 2018
Reserved Instances	Amazon ES now offers support for 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 "Amazon Cognito authentication for OpenSearch Dashboards" (p. 175) .	April 2, 2018
Elasticsearch 6.2 Support	Amazon Elasticsearch Service now supports Elasticsearch version 6.2.	March 14, 2018

Change	Description	Date
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. 357) .	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. 127) .	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 Amazon OpenSearch Service domains" (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 of security and simplifies communications between Amazon ES and other services within a VPC. To learn more, see the section called "VPC support" (p. 37) .	October 17, 2017
Slow Logs Publishing	Amazon ES now supports the publishing of slow logs to CloudWatch Logs. To learn more, see the section called "Monitoring logs" (p. 92) .	October 16, 2017
Elasticsearch 5.5 Support	Amazon ES now supports Elasticsearch version 5.5. You can now restore automated snapshots without contacting AWS Support and store scripts using the _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

Change	Description	Date
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*.