aws

User Guide

# Amazon Managed Service for Prometheus



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

# Amazon Managed Service for Prometheus: User Guide

Copyright © 2026 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.

Services or capabilities described in AWS documentation might vary by Region. To see the differences applicable to the AWS European Sovereign Cloud Region, see the AWS European Sovereign Cloud User Guide.

# Table of Contents

**What is Amazon Managed Service for Prometheus? ..... 1**
Pricing ..... 1
Premium support ..... 2
**Get started ..... 3**
Set up AWS ..... 3
Sign up for an AWS account ..... 3
Create a workspace ..... 4
Ingest metrics ..... 5
Step 1: Add new Helm chart repositories ..... 6
Step 2: Create a Prometheus namespace ..... 6
Step 3: Set up IAM roles for service accounts ..... 6
Step 4: Set up the new server and start ingesting metrics ..... 7
Query metrics ..... 8
**Manage workspaces ..... 10**
Create a workspace ..... 10
Configure your workspace ..... 13
Edit a workspace alias ..... 15
Find your workspace details ..... 15
Delete a workspace ..... 17
**Ingest metrics ..... 19**
AWS managed collectors ..... 20
Integrate Amazon EKS ..... 21
Integrate Amazon MSK ..... 40
Integrate Amazon OpenSearch Service ..... 68
Prometheus-compatible metrics ..... 81
Monitor collectors ..... 82
Customer managed collectors ..... 87
Secure the ingestion of your metrics ..... 88
ADOT collectors ..... 89
Prometheus collectors ..... 106
High-availability data ..... 115
**Query your metrics ..... 123**
PromQL cheat sheet ..... 124
Basic selectors ..... 124

Range vector selectors ... 124
Aggregation operators ... 125
Common functions ... 125
Binary operators ... 126
Practical query examples ... 126
Secure your metric queries ... 127
Using AWS PrivateLink with Amazon Managed Service for Prometheus ... 88
Authentication and authorization ... 88
Use Amazon Managed Grafana ... 128
Connecting to Amazon Managed Grafana in a private VPC ... 128
Use Grafana open source ... 129
Prerequisites ... 129
Step 1: Set up AWS SigV4 ... 129
Step 2: Add the Prometheus data source in Grafana ... 131
Step 3: (optional) Troubleshooting if Save & Test doesn't work ... 133
Use Grafana in Amazon EKS ... 134
Set up AWS SigV4 ... 134
Set up IAM roles for service accounts ... 135
Upgrade the Grafana server using Helm ... 136
Add the Prometheus data source in Grafana ... 137
Use direct queries ... 137
Query with awscurl ... 138
Query statistics ... 140
**Anomaly detection ... 144**
How anomaly detection works ... 144
Getting started with anomaly detection ... 145
PreviewAnomalyDetector ... 145
Query parameter formatting ... 146
API request and response ... 146
**Recording and alerting rules ... 150**
Necessary IAM permissions ... 151
Create a rules file ... 152
Upload a rules file ... 153
Edit a rules file ... 155
Troubleshoot rule evaluations ... 157
Validate alert firing status ... 157

Resolve missing alert notifications ... 157
Check rule health status ... 158
Use offset in queries to handle ingestion delays ... 160
Common issues and solutions ... 161
Best practices for rule evaluations ... 161
Troubleshooting Ruler ... 162
**Alert manager ... 163**
Necessary IAM permissions ... 164
Create a configuration file ... 164
Set up an alert receiver ... 167
Amazon SNS ... 167
PagerDuty ... 177
Upload a configuration file ... 183
Integrate alerts with Grafana ... 186
Prerequisites ... 186
Setting up Amazon Managed Grafana ... 187
Troubleshoot alert manager ... 188
Active alerts warning ... 189
Alert aggregation group size warning ... 189
Alerts size too big warning ... 190
Empty content warning ... 190
Invalid `key/value` warning ... 191
Message limit warning ... 191
No resource based policy error ... 192
Non ASCII warning ... 192
Not authorized to call KMS ... 193
Template error ... 193
**Monitoring workspaces ... 195**
CloudWatch metrics ... 195
Setting a CloudWatch alarm ... 210
CloudWatch Logs ... 211
Configuring CloudWatch Logs ... 211
Query insights and control ... 214
Configuring query logging ... 214
Configuring query throttling thresholds ... 216
Log content ... 216

Limitations ... 217
**Understand and optimize costs ... 218**
What contributes to my costs? ... 218
What is the best way to lower my costs? How do I lower ingestion costs? ... 218
What is the best way to lower my query costs? ... 218
If I decrease the retention period of my metrics, will that help reduce my total bill? ... 219
How can I keep my alert query costs low? ... 219
Can I check my bill at any time? ... 220
What metrics can I use to monitor my costs? ... 220
How do I view my costs in AWS Cost Explorer? ... 221
How do I calculate the number of samples ingested in a month? ... 222
What data granularity is available for historical cost analysis? ... 224
What are best practices for monitoring Amazon Managed Service for Prometheus costs? ... 225
Why is my bill higher at the beginning of the month than at the end of the month? ... 225
I deleted all my Amazon Managed Service for Prometheus workspaces, but I still seem to be getting charged. What might be happening? ... 226
**Integrations ... 227**
Amazon EKS cost monitoring ... 227
AWS Observability Accelerator ... 228
Prerequisites ... 228
Using the managed metrics (agentless) example ... 229
Alternative: Self-managed OpenTelemetry Collector ... 231
Viewing dashboards ... 232
AWS Controllers for Kubernetes ... 232
Prerequisites ... 232
Deploying a workspace ... 233
Configure cluster for remote write ... 237
Amazon CloudWatch metrics with Firehose ... 239
Infrastructure ... 240
Creating a Amazon CloudWatch stream ... 242
Cleanup ... 243
**Security ... 244**
Data protection ... 245
Data collected by Amazon Managed Service for Prometheus ... 246
Encryption at rest ... 246
Identity and Access Management ... 259

Audience ... 260
Authenticating with identities ... 260
Managing access using policies ... 262
How Amazon Managed Service for Prometheus works with IAM ... 263
Identity-based policy examples ... 269
Troubleshooting ... 272
IAM permissions and policies ... 274
Amazon Managed Service for Prometheus permissions ... 274
Sample IAM policies ... 274
Compliance Validation ... 275
Resilience ... 275
Infrastructure Security ... 275
Using service-linked roles ... 276
Metric scraping role ... 276
CloudTrail logs ... 278
Amazon Managed Service for Prometheus management events in CloudTrail ... 280
Amazon Managed Service for Prometheus event examples ... 280
Set up IAM roles for service accounts ... 285
Set up service roles for the ingestion of metrics from Amazon EKS clusters ... 285
Set up IAM roles for service accounts for the querying of metrics ... 288
Interface VPC endpoints ... 291
Create an interface VPC endpoint for Amazon Managed Service for Prometheus ... 292
**Troubleshooting ... 296**
429 or limit exceeded errors ... 296
I see duplicate samples ... 298
I see errors about sample timestamps ... 298
I see an error message related to a limit ... 298
Your local Prometheus server output exceeds the limit. ... 299
Some of my data isn't appearing ... 300
HTTP 422 error: query timeout ... 302
**Tagging ... 303**
Tagging workspaces ... 304
Add a tag to a workspace ... 304
View tags for a workspace ... 306
Edit tags for a workspace ... 307
Remove a tag from a workspace ... 308

Tagging rule groups namespaces ... 310
Add a tag to a rule groups namespace ... 310
View tags for a rule groups namespace ... 312
Edit tags for a rule groups namespace ... 313
Remove a tag from a rule groups namespace ... 314
**Service quotas ... 316**
Service quotas ... 316
Active series default quotas ... 324
Scaling above the default quota ... 325
Ingestion throttling ... 325
Additional limits on ingested data ... 326
**API Reference ... 328**
Amazon Managed Service for Prometheus APIs ... 328
Using Amazon Managed Service for Prometheus with an AWS SDK ... 328
Prometheus-compatible APIs ... 329
CreateAlertManagerAlerts ... 330
DeleteAlertManagerSilence ... 331
GetAlertManagerStatus ... 332
GetAlertManagerSilence ... 333
GetLabels ... 334
GetMetricMetadata ... 337
GetSeries ... 338
ListAlerts ... 340
ListAlertManagerAlerts ... 341
ListAlertManagerAlertGroups ... 343
ListAlertManagerReceivers ... 345
ListAlertManagerSilences ... 346
ListRules ... 347
PutAlertManagerSilences ... 349
QueryMetrics ... 351
RemoteWrite ... 353
**Document History ... 355**

# What is Amazon Managed Service for Prometheus?

Amazon Managed Service for Prometheus is a serverless, Prometheus-compatible monitoring service for container metrics that makes it easier to securely monitor container environments at scale. With Amazon Managed Service for Prometheus, you can use the same open-source Prometheus data model and query language that you use today to monitor the performance of your containerized workloads, and also enjoy improved scalability, availability, and security without having to manage the underlying infrastructure.

Amazon Managed Service for Prometheus automatically scales the ingestion, storage, and querying of operational metrics as workloads scale up and down. It integrates with AWS security services to enable fast and secure access to data.

Amazon Managed Service for Prometheus is designed to be highly available using multiple Availability Zone (Multi-AZ) deployments. Data ingested into a workspace is replicated across three Availability Zones in the same Region.

Amazon Managed Service for Prometheus works with container clusters that run on Amazon Elastic Kubernetes Service and self-managed Kubernetes environments.

With Amazon Managed Service for Prometheus, you use the same open-source Prometheus data model and PromQL query language that you use with Prometheus. Engineering teams can use PromQL to filter, aggregate, and alarm on metrics and quickly gain performance visibility without any code changes. Amazon Managed Service for Prometheus provides flexible query capabilities without the operational cost and complexity.

Metrics ingested into a workspace are stored for 150 days by default, and are then automatically deleted. You can adjust the retention period by configuring your workspace up to a maximum of 1095 days (three years). For more information, see Configure your workspace.

## Pricing

You incur charges for ingestion and storage of metrics. Storage charges are based on the compressed size of metric samples and metadata. For more information, see Amazon Managed Service for Prometheus Pricing.

You can use AWS Cost Explorer and AWS Cost and Usage Reports to monitor your charges. For more information, see Exploring your data using Cost Explorer and What are AWS Cost and Usage Reports.

# Premium support

If you subscribe to any level of the AWS premium support plans, your premium support applies to Amazon Managed Service for Prometheus.

# Get started with Amazon Managed Service for Prometheus

Amazon Managed Service for Prometheus is a serverless, Prometheus-compatible service for monitoring container metrics that makes it easy to securely monitor container environments at scale. This section takes you through three key areas of using Amazon Managed Service for Prometheus:

- Create a workspace – Create a Amazon Managed Service for Prometheus workspace to store and monitor your metrics.
- Ingest metrics – Your workspace is empty until you get metrics into your workspace. You can send metrics to Amazon Managed Service for Prometheus, or have Amazon Managed Service for Prometheus scrape metrics automatically.
- Query metrics – Once you have metrics as data in your workspace, you are ready to query the data to explore or monitor those metrics.

If you are new to AWS, this section also includes details about setting up an AWS account.

**Topics**


- Set up AWS
- Create an Amazon Managed Service for Prometheus workspace
- Ingest Prometheus metrics to the workspace
- Query your Prometheus metrics


# Set up AWS

## Sign up for an AWS account

To get started with AWS, you need an AWS account. There are two ways to sign up for AWS. If you want complete control over account configuration and where your content is stored, or if you have regulated workloads such as HIPAA or FedRAMP, use Sign up for AWS (advanced). If you are a new user and want to use a login you already own to sign in to AWS and create an AWS environment with preconfigured defaults in select AWS Regions, use Sign up for AWS.

# Create an Amazon Managed Service for Prometheus workspace

A *workspace* is a logical space dedicated to the storage and querying of Prometheus metrics. A workspace supports fine-grained access control for authorizing its management such as update, list, describe, and delete, and the ingestion and querying of metrics. You can have one or more workspaces in each Region in your account.

To set up a workspace, follow these steps.

> **Note**
>
> For more detailed information about creating a workspace and the options available, see Create a Amazon Managed Service for Prometheus workspace.

**To create a Amazon Managed Service for Prometheus workspace**

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. For **Workspace alias**, enter an alias for the new workspace.

   Workspace aliases are friendly names that help you identify your workspaces. They do not have to be unique. Two workspaces could have the same alias, but all workspaces will have unique workspace IDs, which are generated by Amazon Managed Service for Prometheus.
3. (Optional) To add tags to the namespace, choose **Add new tag**.

   Then, for **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.

   To add another tag, choose **Add new tag** again.
4. Choose **Create workspace**.

   The workspace details page appears. This displays information including the status, ARN, workspace ID, and endpoint URLs for this workspace for both remote write and queries.

   Initially, the status is probably **CREATING**. Wait until the status is **ACTIVE before you move on to setting up your metric ingestion.**

Make notes of the URLs displayed for **Endpoint - remote write URL** and **Endpoint - query URL**. You'll need them when you configure your Prometheus server to remote write metrics to this workspace and when you query those metrics.

## Ingest Prometheus metrics to the workspace

One way to ingest metrics is to use a standalone Prometheus *agent* (a Prometheus instance running in agent mode) to scrape metrics from your cluster and forward them to Amazon Managed Service for Prometheus for storage and monitoring. This section explains how to set up the ingestion of metrics into your Amazon Managed Service for Prometheus workspace from Amazon EKS by setting up a new instance of Prometheus agent using Helm.

To generate metrics in Amazon EKS, such as Kubernetes or node-level metrics, you can use the Amazon EKS community add-ons. For more information, see Available community add-ons in the *Amazon EKS User Guide*.

For information about other ways to ingest data into Amazon Managed Service for Prometheus, including how to secure metrics and create high-availability metrics, see Ingest metrics to your Amazon Managed Service for Prometheus workspace.

> **Note**
>
> Metrics ingested into a workspace are stored for 150 days by default, and are then automatically deleted. You can adjust the retention period by configuring your workspace up to a maximum of 1095 days (three years). For more information, see Configure your workspace.

The instructions in this section get you up and running with Amazon Managed Service for Prometheus quickly. It assumes that you have already created a workspace. In this section, you set up a new Prometheus server in an Amazon EKS cluster, and the new server uses a default configuration to act as an agent to send metrics to Amazon Managed Service for Prometheus. This method has the following prerequisites:

- You must have an Amazon EKS cluster from which the new Prometheus server will collect metrics.
- Your Amazon EKS cluster must have an Amazon EBS CSI driver installed (required by Helm).

- You must use Helm CLI 3.0 or later.
- You must use a Linux or MacOS computer to perform the steps in the following sections.

## Step 1: Add new Helm chart repositories

To add new Helm chart repositories, enter the following commands. For more information about these commands, see Helm Repo.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo add kube-state-metrics https://kubernetes.github.io/kube-state-metrics
helm repo update
```

## Step 2: Create a Prometheus namespace

Enter the following command to create a Prometheus namespace for the Prometheus server and other monitoring components. Replace *prometheus-agent-namespace* with the name that you want for this namespace.

```
kubectl create namespace prometheus-agent-namespace
```

## Step 3: Set up IAM roles for service accounts

For this method of ingestion, you need to use IAM roles for service accounts in the Amazon EKS cluster where the Prometheus agent is running.

With IAM roles for service accounts, you can associate an IAM role with a Kubernetes service account. This service account can then provide AWS permissions to the containers in any pod that uses that service account. For more information, see IAM roles for service accounts.

If you have not already set up these roles, follow the instructions at Set up service roles for the ingestion of metrics from Amazon EKS clusters to set up the roles. The instructions in that section require the use of `eksctl`. For more information, see Getting started with Amazon Elastic Kubernetes Service – `eksctl`.

> **Note**
>
> When you are not on EKS or AWS and using just access key and secret key to access Amazon Managed Service for Prometheus, you cannot use the `EKS-IAM-ROLE` based SigV4.

# Step 4: Set up the new server and start ingesting metrics

To install the new Prometheus agent and send metrics to your Amazon Managed Service for Prometheus workspace, follow these steps.

**To install a new Prometheus agent and send metrics to your Amazon Managed Service for Prometheus workspace**

1. Use a text editor to create a file named `my_prometheus_values_yaml` with the following content.

   - Replace *IAM_PROXY_PROMETHEUS_ROLE_ARN* with the ARN of the **amp-iamproxy-ingest-role** that you created in Set up service roles for the ingestion of metrics from Amazon EKS clusters.
   - Replace *WORKSPACE_ID* with the ID of your Amazon Managed Service for Prometheus workspace.
   - Replace *REGION* with the Region of your Amazon Managed Service for Prometheus workspace.

```
## The following is a set of default values for prometheus server helm chart which
 enable remoteWrite to AMP
## For the rest of prometheus helm chart values see: https://github.com/prometheus-
community/helm-charts/blob/main/charts/prometheus/values.yaml
##
serviceAccounts:
  server:
    name: amp-iamproxy-ingest-service-account
    annotations:
      eks.amazonaws.com/role-arn: ${IAM_PROXY_PROMETHEUS_ROLE_ARN}
server:
  remoteWrite:
    - url: https://aps-workspaces.${REGION}.amazonaws.com/workspaces/
${WORKSPACE_ID}/api/v1/remote_write
      sigv4:
        region: ${REGION}
      queue_config:
        max_samples_per_send: 1000
        max_shards: 200
        capacity: 2500
```

2. Enter the following command to create the Prometheus server.

   - Replace *prometheus-chart-name* with your Prometheus release name.
   - Replace *prometheus-agent-namespace* with the name of your Prometheus namespace.

```
helm install prometheus-chart-name prometheus-community/prometheus -n prometheus-
agent-namespace \
-f my_prometheus_values_yaml
```

## Query your Prometheus metrics

Now that metrics are being ingested to the workspace, you can query them. A common way to query your metrics is to use a service such as Grafana to query the metrics. In this section, you will learn how to use Amazon Managed Grafana to query metrics from Amazon Managed Service for Prometheus.

> **Note**
>
> To learn about other ways to query your Amazon Managed Service for Prometheus metrics, or use the Amazon Managed Service for Prometheus APIs, see Query your Prometheus metrics.

This section assumes you already have a workspace created, and are ingesting metrics into it.

You perform your queries using the standard Prometheus query language, PromQL. For more information about PromQL and its syntax, see Querying Prometheus in the Prometheus documentation.

Amazon Managed Grafana is a fully managed service for open-source Grafana that simplifies connecting to open-source, third-party ISV, and AWS services for visualizing and analyzing your data sources at scale.

Amazon Managed Service for Prometheus supports using Amazon Managed Grafana to query metrics in a workspace. In the Amazon Managed Grafana console, you can add an Amazon Managed Service for Prometheus workspace as a data source by discovering your existing Amazon Managed Service for Prometheus accounts. Amazon Managed Grafana manages the configuration

of the authentication credentials that are required to access Amazon Managed Service for Prometheus. For detailed instructions on creating a connection to Amazon Managed Service for Prometheus from Amazon Managed Grafana, see the instructions in the Amazon Managed Grafana User Guide.

You may also view your Amazon Managed Service for Prometheus alerts in Amazon Managed Grafana. For instructions to set up integration with alerts, see Integrate alerts with Amazon Managed Grafana or open source Grafana.

> **Note**
>
> If you have configured your Amazon Managed Grafana workspace to use a Private VPC, you must connect your Amazon Managed Service for Prometheus workspace to the same VPC. For more information, see Connecting to Amazon Managed Grafana in a private VPC.

# Manage Amazon Managed Service for Prometheus workspaces

A *workspace* is a logical space dedicated to the storage and querying of Prometheus metrics. A workspace supports fine-grained access control for authorizing its management such as update, list, describe, and delete, and the ingestion and querying of metrics. You can have one or more workspaces in each Region in your account.

Use the procedures in this section to create and manage your Amazon Managed Service for Prometheus workspaces.

**Topics**


- Create a Amazon Managed Service for Prometheus workspace
- Configure your workspace
- Edit a workspace alias
- Find your Amazon Managed Service for Prometheus workspace details, including ARN
- Delete an Amazon Managed Service for Prometheus workspace


## Create a Amazon Managed Service for Prometheus workspace

Follow these steps to create a Amazon Managed Service for Prometheus workspace. You can choose to use the AWS CLI or the Amazon Managed Service for Prometheus console.

> **Note**
>
> If you are running an Amazon EKS cluster, you can also create a new workspace using AWS Controllers for Kubernetes.

**To create a workspace using the AWS CLI**

1. Enter the following command to create the workspace. This example creates a workspace named `my-first-workspace`, but you can use a different alias (or none) if you want. Workspace aliases are friendly names that help you identify your workspaces. They do not have to be unique. Two workspaces can have the same alias, but all workspaces have unique workspace IDs, which are generated by Amazon Managed Service for Prometheus.

(Optional) To use your own KMS key to encrypt data stored in your workspace, you can include the `kmsKeyArn` parameter with the AWS KMS key to use. While Amazon Managed Service for Prometheus does not charge you for using customer managed keys, there may be costs associated with keys from AWS Key Management Service. For more information about Amazon Managed Service for Prometheus encryption of data in the workspace, or how to create, manage, and use your own customer managed key, see Encryption at rest.

Parameters in brackets (`[]`) are optional, do not include the brackets in your command.

```
aws amp create-workspace [--alias my-first-workspace] [--kmsKeyArn arn:aws:aps:us-west-2:111122223333:workspace/ws-sample-1234-abcd-56ef-7890abcd12ef]  [--tags Status=Secret,Team=My-Team]
```

This command returns the following data:

- `workspaceId` is the unique ID for this workspace. Make a note of this ID.
- `arn` is the ARN for this workspace.
- `status` is the current status of the workspace. Immediately after you create the workspace, this will probably be `CREATING`.
- `kmsKeyArn` is the customer managed key used to encrypt the workspace data, if given.

> **Note**
>
> Workspaces created with customer managed keys cannot use AWS managed collectors for ingestion.
> Choose whether to use customer managed keys or AWS owned keys carefully. Workspaces created with customer managed keys can't be converted to use AWS owned keys later (and vice versa).

- `tags` lists the workspace's tags, if any.

2. If your `create-workspace` command returns a status of `CREATING`, you can then enter the following command to determine when the workspace is ready. Replace *`my-workspace-id`* with the value that the `create-workspace` command returned for `workspaceId`.

```
aws amp describe-workspace --workspace-id my-workspace-id
```

When the `describe-workspace` command returns `ACTIVE` for `status`, the workspace is ready to use.

**To create a workspace using the Amazon Managed Service for Prometheus console**

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. Choose **Create**.
3. For **Workspace alias**, enter an alias for the new workspace.

   Workspace aliases are friendly names that help you identify your workspaces. They do not have to be unique. Two workspaces can have the same alias, but all workspaces have unique workspace IDs, which are generated by Amazon Managed Service for Prometheus.
4. (Optional) To use your own KMS key to encrypt data stored in your workspace, you can select **Customize encryption settings**, and choose the AWS KMS key to use (or create a new one). You can choose a key in your account from the drop down list, or enter the ARN for any key that you have access to. While Amazon Managed Service for Prometheus does not charge you for using customer managed keys, there may be costs associated with keys from AWS Key Management Service.

   For more information about Amazon Managed Service for Prometheus encryption of data in the workspace, or how to create, manage, and use your own, customer managed key, see Encryption at rest.

   > **Note**
   >
   > Workspaces created with customer managed keys cannot use AWS managed collectors for ingestion.
   > Choose whether to use customer managed keys or AWS owned keys carefully. Workspaces created with customer managed keys can't be converted to use AWS owned keys later (and vice versa).
5. (Optional) To add one or more tags to the workspace, choose **Add new tag**. Then, in **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.

   To add another tag, choose **Add new tag** again.

6. Choose **Create workspace**.

   The workspace details page appears. This displays information including the status, ARN, workspace ID, and endpoint URLs for this workspace for both remote write and queries.

   The status returns **CREATING** until the workspace is ready. Wait until the status is **ACTIVE** before you move on to setting up your metric ingestion.

   Make note of the URLs that are displayed for **Endpoint - remote write URL** and **Endpoint - query URL**. You'll need them when you configure your Prometheus server to remote write metrics to this workspace and when you query those metrics.

For information about how to ingest metrics into the workspace, see Ingest Prometheus metrics to the workspace.

# Configure your workspace

You can configure your workspace for the following:

- Define *label sets* and define limits on the active time series that match your defined label sets. A label set is a set of one or more *labels*, which are name/value pairs that help give context to time series metrics.

  By defining label sets and setting active time series limits, you can limit spikes in one tenant or source to affect only that tenant or source. For example, if you set a 1,000,000 active time series limit on the label set `team=A env=prod`, then if the number of ingested time series that match that label set exceed the limit, then only the time series that match the label set are throttled. This way, other tenants or metric sources are unaffected.

  For more information about labels in Prometheus, see Data Model.
- Set a retention period to define the number of days for the data to be retained in the workspace.
- Set an *out-of-order time window* to allow the workspace to accept samples that arrive out of chronological order. All workspaces have a default out-of-order time window of 60 seconds. Setting this value to 0 disables the feature and out-of-order samples are discarded.
- Set a *rule query offset* to introduce a global delay before rule evaluation queries run. This compensates for ingestion delays or out-of-order samples, ensuring that rules evaluate against a more complete dataset. A query offset configured at the rule group level takes priority over

this workspace-level setting. For more information, see Use offset in queries to handle ingestion delays.

**To configure your workspace**

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the upper left corner of the page, choose the menu icon and then choose **All workspaces**.
3. Choose the **Workspace ID** of the workspace.
4. Choose the **Workspace configurations** tab.
5. To set the retention period for the workspace, choose **Edit** in the **Retention period** section. Then specify the new retention period in days. The maximum is 1095 days (three years).
6. To set the out-of-order time window, choose **Edit** in the **Out of order time window** section. Then specify the time window in seconds. All workspaces have a default out-of-order time window of 60 seconds. The minimum value is 0 seconds; when set to 0, out-of-order samples are discarded. The maximum value is 600 seconds. To request a higher maximum, see Amazon Managed Service for Prometheus service quotas.
7. To set the rule query offset, choose **Edit** in the **Rule query offset** section. Then specify the offset in seconds. The default value for new workspaces is 60 seconds. The minimum value is 0 seconds and the maximum value is 86,400 seconds (24 hours).

   > **Note**
   >
   > Existing workspaces have a default rule query offset of 0 seconds, which you can update through the console.
   > If you have configured an out-of-order time window, consider setting the rule query offset to a value greater than or equal to the out-of-order time window. This ensures that out-of-order samples are included in rule evaluations. A rule query offset delays rule evaluations and any resulting alerts by the configured duration.

8. To add or modify label sets and their active series limits, choose **Edit** in the **Label sets** section. Then do the following:

   a. (Optional) Enter a value in **Default bucket limit** to set a limit on the maximum number of active time series that can be ingested in the workspace, counting only time series that don't match any defined label set.

b. To define a label set, enter an active time series limit for the new label set under **Active series limit**.

   Then, enter a label and value for one label that will be used in the label set, and choose **Add label**.

c. (Optional) To define another label set, choose **Add another label set** and repeat the previous steps.

9. When you are finished, choose **Save changes**.

## Edit a workspace alias

You can edit a workspace to change its alias. To change the workspace alias using the AWS CLI, enter the following command.

```
aws amp update-workspace-alias --workspace-id my-workspace-id --alias "new-alias"
```

**To edit a workspace using the Amazon Managed Service for Prometheus console**

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the upper left corner of the page, choose the menu icon and then choose **All workspaces**.
3. Choose the workspace ID of the workspace that you want to edit, and then choose **Edit**.
4. Enter a new alias for the workspace and then choose **Save**.

## Find your Amazon Managed Service for Prometheus workspace details, including ARN

You can find the details of your Amazon Managed Service for Prometheus workspace by using either the AWS console or the AWS CLI.

Console

**To find your workspace details using the Amazon Managed Service for Prometheus console**

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.

2. In the upper left corner of the page, choose the menu icon and then choose **All workspaces**.
3. Choose the **Workspace ID** of the workspace. This will display details about your workspace, including:

   - **Current status** – The status of your workspace, for example **Active**, is displayed under **Status**.
   - **ARN** – The workspace ARN is displayed under **ARN**.
   - **ID** – The workspace ID is displayed under **Workspace ID**.
   - **URLs** – The console displays multiple URLs for the workspace, including the URLs for writing to or querying data from the workspace.

     **Note**

     By default, the URLs given are the IPv4 URLs. You can also use dualstack (IPv4 and IPv6 supported) URLs. These are the same, but are in the domain `api.aws` rather than the default `amazonaws.com`. For example, if you were to see the following (an IPv4 URL):

     ```
     https://aps-workspaces.us-east-1.amazonaws.com/workspaces/ws-abcd1234-ef56-7890-ab12-example/api/v1/remote_write
     ```

     You could create a dualstack (including support for IPv6), URL as follows:

     ```
     https://aps-workspaces.us-east-1.api.aws/workspaces/ws-abcd1234-ef56-7890-ab12-example/api/v1/remote_write
     ```

   Below this section are tabs with information about rules, alert manager, logs, configuration, and tags.

AWS CLI

**To find your workspace details using the AWS CLI**

The following command returns the details of the workspace. You must replace *my-workspace-id* with the workspace ID of the workspace for which you want the details.

```
aws amp describe-workspace --workspace-id my-workspace-id
```

This returns details about your workspace, including:

- **Current status** – The status of your workspace, for example `ACTIVE`, is returned in the `statusCode` property.
- **ARN** – The workspace ARN is returned in the `arn` property.
- **URLs** – The AWS CLI returns the base URL for the workspace in the `prometheusEndpoint` property.

> **Note**
>
> By default, the URL returned is the IPv4 URL. You can also use a dualstack (IPv4 and IPv6 supported) URL in the domain `api.aws` rather than the default `amazonaws.com`. For example, if you were to see the following (an IPv4 URL):
>
> ```
> https://aps-workspaces.us-east-1.amazonaws.com/workspaces/ws-abcd1234-ef56-7890-ab12-example/
> ```
>
> You could create a dualstack (including support for IPv6), URL as follows:
>
> ```
> https://aps-workspaces.us-east-1.api.aws/workspaces/ws-abcd1234-ef56-7890-ab12-example/
> ```
>
> You can also create the remote write and query URLs for the workspace, by adding `/api/v1/remote_write` or `/api/v1/query`, respectively.

# Delete an Amazon Managed Service for Prometheus workspace

Deleting a workspace deletes the data that has been ingested into it.

> **Note**
>
> Deleting an Amazon Managed Service for Prometheus workspace does not automatically delete any AWS managed collectors that are scraping metrics and sending them to the workspace. For more information, see Find and delete scrapers.

### To delete a workspace using the AWS CLI

Use the following command:

```
aws amp delete-workspace --workspace-id my-workspace-id
```

### To delete a workspace using the Amazon Managed Service for Prometheus console

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the upper left corner of the page, choose the menu icon and then choose **All workspaces**.
3. Choose the workspace ID of the workspace that you want to delete, and then choose **Delete**.
4. Enter **`delete`** in the confirmation box, and choose **Delete**.

# Ingest metrics to your Amazon Managed Service for Prometheus workspace

Metrics must be ingested into your Amazon Managed Service for Prometheus workspace before you can query or alert on those metrics. This section explains how to set up the ingestion of metrics into your workspace.

> **Note**
>
> Metrics ingested into a workspace are stored for 150 days by default, and are then automatically deleted. You can adjust the retention period by configuring your workspace up to a maximum of 1095 days (three years). For more information, see Configure your workspace.

There are two methods of ingesting metrics into your Amazon Managed Service for Prometheus workspace.

- **Using an AWS managed collector** – Amazon Managed Service for Prometheus provides a fully-managed, agentless scraper to automatically *scrape* metrics from your Amazon Elastic Kubernetes Service (Amazon EKS) clusters. Scraping automatically pulls the metrics from Prometheus-compatible endpoints.
- **Using a customer managed collector** – You have many options for managing your own collector. Two of the most common collectors to use are installing your own instance of Prometheus, running in agent mode, or using AWS Distro for OpenTelemetry. These are both described in detail in the following sections.

  Collectors send metrics to Amazon Managed Service for Prometheus using Prometheus remote write functionality. You can directly send metrics to Amazon Managed Service for Prometheus by using Prometheus remote write in your own application. For more details about directly using remote write, and remote write configurations, see remote_write in the Prometheus documentation.

**Topics**

- Ingest metrics with AWS managed collectors

- Customer managed collectors

# Ingest metrics with AWS managed collectors

A common use case for Amazon Managed Service for Prometheus is to monitor Kubernetes clusters managed by Amazon Elastic Kubernetes Service (Amazon EKS). Kubernetes clusters, and many applications that run within Amazon EKS, automatically export their metrics for Prometheus-compatible scrapers to access.

> **Note**
>
> Amazon EKS exposes API server metrics, `kube-controller-manager` metrics, and `kube-scheduler` metrics in a cluster. Many other technologies and applications running in Kubernetes environments provide Prometheus-compatible metrics. For a list of well-documented exporters, see Exporters and integrations in the Prometheus documentation.

Amazon Managed Service for Prometheus provides a fully managed, agent less scraper, or *collector*, that automatically discovers and pulls Prometheus-compatible metrics. You don't have to manage, install, patch, or maintain agents or scrapers. An Amazon Managed Service for Prometheus collector provides reliable, stable, highly available, automatically scaled collection of metrics for your Amazon EKS cluster. Amazon Managed Service for Prometheus managed collectors work with Amazon EKS clusters, including EC2 and Fargate.

An Amazon Managed Service for Prometheus collector creates an Elastic Network Interface (ENI) per subnet specified when creating the scraper. The collector scrapes the metrics through these ENIs, and uses `remote_write` to push the data to your Amazon Managed Service for Prometheus workspace using a VPC endpoint. The scraped data never travels on the public internet.

The following topics provide more information about how to use an Amazon Managed Service for Prometheus collector in your Amazon EKS cluster, and about the collected metrics.

**Topics**

- Set up managed collectors for Amazon EKS
- Set up managed Prometheus collectors for Amazon MSK
- Set up managed Prometheus collectors for Amazon OpenSearch Service
- What are Prometheus-compatible metrics?

- Monitor collectors with vended logs

## Set up managed collectors for Amazon EKS

To use an Amazon Managed Service for Prometheus collector, you create a scraper that discovers and pulls metrics in your Amazon EKS cluster. You can also create a scraper that integrates with Amazon Managed Streaming for Apache Kafka or with Amazon OpenSearch Service. For more information, see Integrate Amazon MSK and Integrate Amazon OpenSearch Service.

- You can create a scraper as part of your Amazon EKS cluster creation. For more information about creating an Amazon EKS cluster, including creating a scraper, see Creating an Amazon EKS cluster in the *Amazon EKS User Guide*.
- You can create your own scraper, programmatically with the AWS API or by using the AWS CLI.

An Amazon Managed Service for Prometheus collector scrapes metrics that are Prometheus-compatible. For more information about Prometheus compatible metrics, see What are Prometheus-compatible metrics?. Amazon EKS clusters expose metrics for the API server. Amazon EKS clusters that are Kubernetes version `1.28` or above also expose metrics for the `kube-scheduler` and `kube-controller-manager`. For more information, see Fetch control plane raw metrics in Prometheus format in the *Amazon EKS User Guide*.

> **Note**
>
> Scraping metrics from a cluster may incur charges for network usage. One way to optimize these costs is to configure your `/metrics` endpoint to compress the provided metrics (for example, with gzip), reducing the data that must be moved across the network. How to do this depends on the application or library providing the metrics. Some libraries gzip by default.

The following topics describe how to create, manage, and configure scrapers.

**Topics**

- Create a scraper
- Configuring your Amazon EKS cluster
- Find and delete scrapers

- Scraper configuration
- Troubleshooting scraper configuration
- Scraper limitations

## Create a scraper

An Amazon Managed Service for Prometheus collector consists of a scraper that discovers and collects metrics from an Amazon EKS cluster. Amazon Managed Service for Prometheus manages the scraper for you, giving you the scalability, security, and reliability that you need, without having to manage any instances, agents, or scrapers yourself.

There are three ways to create a scraper:

- A scraper is automatically created for you when you create an Amazon EKS cluster through the Amazon EKS console and choose to turn on Prometheus metrics.
- You can create a scraper from the Amazon EKS console for an existing cluster. Open the cluster in the Amazon EKS console, then, on the **Observability** tab, choose **Add scraper**.

  For more details on the available settings, see Turn on Prometheus metrics in the *Amazon EKS User Guide*.
- You can create a scraper using either the AWS API or the AWS CLI.

  These options are described in the following procedure.

There are a few prerequisites for creating your own scraper:

- You must have an Amazon EKS cluster created.
- Your Amazon EKS cluster must have cluster endpoint access control set to include private access. It can include private and public, but must include private.
- The Amazon VPC in which the Amazon EKS cluster resides must have DNS enabled.

> **Note**
>
> The cluster will be associated with the scraper by its Amazon resource name (ARN). If you delete a cluster, and then create a new one with the same name, the ARN will be reused

for the new cluster. Because of this, the scraper will attempt to collect metrics for the new cluster. You delete scrapers separately from deleting the cluster.

AWS API

**To create a scraper using the AWS API**

Use the `CreateScraper` API operation to create a scraper with the AWS API. The following example creates a scraper in the `us-west-2` Region. You need to replace the AWS account, workspace, security, and Amazon EKS cluster information with your own IDs, and provide the configuration to use for your scraper.

> **Note**
>
> The security group and subnets should be set to the security group and subnets for the cluster to which you are connecting.
> You must include at least two subnets, in at least two availability zones.

The `scrapeConfiguration` is a Prometheus configuration YAML file that is base64 encoded. You can download a general purpose configuration with the `GetDefaultScraperConfiguration` API operation. For more information about the format of the `scrapeConfiguration`, see Scraper configuration.

```
POST /scrapers HTTP/1.1
Content-Length: 415
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: aws-cli/1.18.147 Python/2.7.18 Linux/5.4.58-37.125.amzn2int.x86_64
 botocore/1.18.6

{
    "alias": "myScraper",
    "destination":  {
        "ampConfiguration": {
            "workspaceArn": "arn:aws:aps:us-west-2:account-id:workspace/
ws-workspace-id"
        }
    },
```

```
    "source": {
        "eksConfiguration": {
            "clusterArn": "arn:aws:eks:us-west-2:account-id:cluster/cluster-name",
            "securityGroupIds": ["sg-security-group-id"],
            "subnetIds": ["subnet-subnet-id-1", "subnet-subnet-id-2"]
        }
    },
    "scrapeConfiguration": {
        "configurationBlob": <base64-encoded-blob>
    }
}
```

AWS CLI

**To create a scraper using the AWS CLI**

Use the `create-scraper` command to create a scraper with the AWS CLI. The following example creates a scraper in the `us-west-2` Region. You need to replace the AWS account, workspace, security, and Amazon EKS cluster information with your own IDs, and provide the configuration to use for your scraper.

> **Note**
>
> The security group and subnets should be set to the security group and subnets for the cluster to which you are connecting.
> You must include at least two subnets, in at least two availability zones.

The `scrape-configuration` is a Prometheus configuration YAML file that is base64 encoded. You can download a general purpose configuration with the `get-default-scraper-configuration` command. For more information about the format of the `scrape-configuration`, see Scraper configuration.

```
aws amp create-scraper \
  --source eksConfiguration="{clusterArn='arn:aws:eks:us-west-2:account-
id:cluster/cluster-name', securityGroupIds=['sg-security-group-
id'],subnetIds=['subnet-subnet-id-1', 'subnet-subnet-id-2']}" \
  --scrape-configuration configurationBlob=<base64-encoded-blob> \
  --destination ampConfiguration="{workspaceArn='arn:aws:aps:us-west-2:account-
id:workspace/ws-workspace-id'}"
```

The following is a full list of the scraper operations that you can use with the AWS API:

- Create a scraper with the CreateScraper API operation.
- List your existing scrapers with the ListScrapers API operation.
- Update the alias, configuration, or destination of a scraper with the UpdateScraper API operation.
- Delete a scraper with the DeleteScraper API operation.
- Get more details about a scraper with the DescribeScraper API operation.
- Get a general purpose configuration for scrapers with the GetDefaultScraperConfiguration API operation.

> **Note**
>
> The Amazon EKS cluster that you are scraping must be configured to allow Amazon Managed Service for Prometheus to access the metrics. The next topic describes how to configure your cluster.

### Cross-account setup

To create a cross-account scraper when your Amazon EKS cluster and Amazon Managed Service for Prometheus workspace are in different accounts, use the following procedure. For example, you have a source account `account_id_source` containing the Amazon EKS cluster and a target account `account_id_target` containing the Amazon Managed Service for Prometheus workspace.

#### To create a scraper in a cross-account setup

1. In the source account, create a role `arn:aws:iam::`*`account_id`*`_source:role/Source` and add the following trust policy.

```
{
    "Effect": "Allow",
    "Principal": {
    "Service": [
        "scraper.aps.amazonaws.com"
     ]
    },
```

```
    "Action": "sts:AssumeRole",
    "Condition": {
        "ArnEquals": {
            "aws:SourceArn": "scraper_ARN"
        },
        "StringEquals": {
            "AWS:SourceAccount": "account_id"
        }
    }
}
```

2. On every combination of source (Amazon EKS cluster) and target (Amazon Managed Service for Prometheus workspace), you need to create a role `arn:aws:iam::`*`account_id`*`_target:role/Target` and add the following trust policy with permissions for AmazonPrometheusRemoteWriteAccess.

```
{
  "Effect": "Allow",
  "Principal": {
     "AWS": "arn:aws:iam::account_id_source:role/Source"
  },
  "Action": "sts:AssumeRole",
  "Condition": {
     "StringEquals": {
        "sts:ExternalId": "scraper_ARN"
      }
  }
}
```

3. Create a scraper with the `--role-configuration` option.

```
aws amp create-scraper \
  --source eksConfiguration="{clusterArn='arn:aws:eks:us-west-2:account-
id_source:cluster/xarw,subnetIds=[subnet-subnet-id]}" \
  --scrape-configuration configurationBlob=<base64-encoded-blob> \
  --destination ampConfiguration="{workspaceArn='arn:aws:aps:us-west-2:account-
id_target:workspace/ws-workspace-id'}"\
```

```
  --role-configuration '{"sourceRoleArn":"arn:aws:iam::account-id_source:role/
Source", "targetRoleArn":"arn:aws:iam::account-id_target:role/Target"}'
```

4. Validate the scraper creation.

```
aws amp list-scrapers
{
    "scrapers": [
        {
            "scraperId": "scraper-id",
            "arn": "arn:aws:aps:us-west-2:account_id_source:scraper/scraper-id",
            "roleArn": "arn:aws:iam::account_id_source:role/aws-service-role/
scraper.aps.amazonaws.com/
AWSServiceRoleForAmazonPrometheusScraperInternal_cc319052-41a3-4",
            "status": {
                "statusCode": "ACTIVE"
            },
            "createdAt": "2024-10-29T16:37:58.789000+00:00",
            "lastModifiedAt": "2024-10-29T16:55:17.085000+00:00",
            "tags": {},
            "source": {
                "eksConfiguration": {
                    "clusterArn": "arn:aws:eks:us-west-2:account_id_source:cluster/
xarw",
                    "securityGroupIds": [
                        "sg-security-group-id",
                        "sg-security-group-id"
                    ],
                    "subnetIds": [
                        "subnet-subnet_id"
                    ]
                }
            },
            "destination": {
                "ampConfiguration": {
                    "workspaceArn": "arn:aws:aps:us-
west-2:account_id_target:workspace/ws-workspace-id"
                }
            }
        }
    ]
}
```

### Changing between RoleConfiguration and service-linked role

When you want to switch back to a service-linked role instead of the `RoleConfiguration` to write to an Amazon Managed Service for Prometheus workspace, you must update the `UpdateScraper` and provide a workspace in the same account as the scraper without the `RoleConfiguration`. The `RoleConfiguration` will be removed from the scraper and the service-linked role will be used.

When you are changing workspaces in the same account as the scraper and you want to continue using the `RoleConfiguration`, you must again provide the `RoleConfiguration` on `UpdateScraper`.

### Creating scraper for workspaces enabled with customer managed keys

To create a scraper for ingesting metrics into a Amazon Managed Service for Prometheus workspace with customer managed keys, use the `--role-configuration` with both the source and target set to the same account.

```
aws amp create-scraper \
  --source eksConfiguration="{clusterArn='arn:aws:eks:us-west-2:account-id:cluster/
xarw,subnetIds=[subnet-subnet_id]}" \
  --scrape-configuration configurationBlob=<base64-encoded-blob> \
  --destination ampConfiguration="{workspaceArn='arn:aws:aps:us-west-2:account-
id:workspace/ws-workspace-id'}"\
  --role-configuration '{"sourceRoleArn":"arn:aws:iam::account_id:role/Source",
 "targetRoleArn":"arn:aws:iam::account_id:role/Target"}'
```

### Common errors when creating scrapers

The following are the most common issues when attempting to create a new scraper.

- Required AWS resources don't exist. The *security group*, *subnets*, and *Amazon EKS cluster* specified must exist.
- Insufficient IP address space. You must have at least one IP address available in each subnet that you pass into the `CreateScraper` API.

## Configuring your Amazon EKS cluster

Your Amazon EKS cluster must be configured to allow the scraper to access metrics. There are two options for this configuration:

- Use Amazon EKS *access entries* to automatically provide Amazon Managed Service for Prometheus collectors access to your cluster.
- Manually configure your Amazon EKS cluster for managed metric scraping.

The following topics describe each of these in more detail.

### Configure Amazon EKS for scraper access with access entries

Using access entries for Amazon EKS is the easiest way to give Amazon Managed Service for Prometheus access to scrape metrics from your cluster.

The Amazon EKS cluster that you are scraping must be configured to allow API authentication. The cluster authentication mode must be set to either `API` or `API_AND_CONFIG_MAP`. This is viewable in the Amazon EKS console on the **Access configuration** tab of the cluster details. For more information, see Allowing IAM roles or users access to Kubernetes object on your Amazon EKS cluster in the *Amazon EKS User Guide*.

You can create the scraper when creating the cluster, or after creating the cluster:

- **When creating a cluster** – You can configure this access when you create an Amazon EKS cluster through the Amazon EKS console (follow the instructions to create a scraper as part of the cluster), and an access entry policy will automatically be created, giving Amazon Managed Service for Prometheus access to the cluster metrics.
- **Adding after a cluster is created** – if your Amazon EKS cluster already exists, then set the authentication mode to either `API` or `API_AND_CONFIG_MAP`, and any scrapers you create through the Amazon Managed Service for Prometheus API or CLI or through the Amazon EKS console will automatically have the correct access entry policy created for you, and the scrapers will have access to your cluster.

### Access entry policy created

When you create a scraper and let Amazon Managed Service for Prometheus generate an access entry policy for you, it generates the following policy. For more information about access entries, see Allowing IAM roles or users access to Kubernetes in the *Amazon EKS User Guide*.

```
{
    "rules": [
        {
            "effect": "allow",
            "apiGroups": [
                ""
            ],
            "resources": [
                "nodes",
                "nodes/proxy",
                "nodes/metrics",
                "services",
                "endpoints",
                "pods",
                "ingresses",
                "configmaps"
            ],
            "verbs": [
                "get",
                "list",
                "watch"
            ]
        },
        {
            "effect": "allow",
            "apiGroups": [
                "extensions",
                "networking.k8s.io"
            ],
            "resources": [
                "ingresses/status",
                "ingresses"
            ],
            "verbs": [
                "get",
                "list",
                "watch"
            ]
        },
        {
            "effect": "allow",
            "apiGroups": [
                "metrics.eks.amazonaws.com"
```

```
            ],
            "resources": [
                "kcm/metrics",
                "ksh/metrics"
            ],
            "verbs": [
                "get"
            ]
        },
        {
            "effect": "allow",
            "nonResourceURLs": [
                "/metrics"
            ],
            "verbs": [
                "get"
            ]
        }
    ]
}
```

### Manually configuring Amazon EKS for scraper access

If you prefer to use the `aws-auth ConfigMap` to control access to your kubernetes cluster, you can still give Amazon Managed Service for Prometheus scrapers access to your metrics. The following steps will give Amazon Managed Service for Prometheus access to scrape metrics from your Amazon EKS cluster.

> **Note**
>
> For more information about `ConfigMap` and access entries, see Allowing IAM roles or users access to Kubernetes in the *Amazon EKS User Guide*.

This procedure uses `kubectl` and the AWS CLI. For information about installing `kubectl`, see Installing kubectl in the *Amazon EKS User Guide*.

### To manually configure your Amazon EKS cluster for managed metric scraping

1. Create a file, called `clusterrole-binding.yml`, with the following text:

```
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: ClusterRole
metadata:
  name: aps-collector-role
rules:
  - apiGroups: [""]
    resources: ["nodes", "nodes/proxy", "nodes/metrics", "services", "endpoints",
 "pods", "ingresses", "configmaps"]
    verbs: ["describe", "get", "list", "watch"]
  - apiGroups: ["extensions", "networking.k8s.io"]
    resources: ["ingresses/status", "ingresses"]
    verbs: ["describe", "get", "list", "watch"]
  - nonResourceURLs: ["/metrics"]
    verbs: ["get"]
  - apiGroups: ["metrics.eks.amazonaws.com"]
    resources: ["kcm/metrics", "ksh/metrics"]
    verbs: ["get"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: aps-collector-user-role-binding
subjects:
- kind: User
  name: aps-collector-user
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: aps-collector-role
  apiGroup: rbac.authorization.k8s.io
```

2. Run the following command in your cluster:

```
kubectl apply -f clusterrole-binding.yml
```

This will create the cluster role binding and rule. This example uses `aps-collector-role` as the role name, and `aps-collector-user` as the user name.

3. The following command gives you information about the scraper with the ID *`scraper-id`*. This is the scraper that you created using the command in the previous section.

```
aws amp describe-scraper --scraper-id scraper-id
```

4. From the results of the `describe-scraper`, find the `roleArn`.This will have the following format:

   ```
   arn:aws:iam::account-id:role/aws-service-role/scraper.aps.amazonaws.com/
   AWSServiceRoleForAmazonPrometheusScraper_unique-id
   ```

   Amazon EKS requires a different format for this ARN. You must adjust the format of the returned ARN to be used in the next step. Edit it to match this format:

   ```
   arn:aws:iam::account-id:role/AWSServiceRoleForAmazonPrometheusScraper_unique-id
   ```

   For example, this ARN:

   ```
   arn:aws:iam::111122223333:role/aws-service-role/scraper.aps.amazonaws.com/
   AWSServiceRoleForAmazonPrometheusScraper_1234abcd-56ef-7
   ```

   Must be rewritten as:

   ```
   arn:aws:iam::111122223333:role/
   AWSServiceRoleForAmazonPrometheusScraper_1234abcd-56ef-7
   ```

5. Run the following command in your cluster, using the modified `roleArn` from the previous step, as well as your cluster name and region.:

   ```
   eksctl create iamidentitymapping --cluster cluster-name --region region-id --
   arn roleArn --username aps-collector-user
   ```

   This allows the scraper to access the cluster using the role and user you created in the `clusterrole-binding.yml` file.

## Find and delete scrapers

You can use the AWS API or the AWS CLI to list the scrapers in your account or to delete them.

> **Note**
>
> Make sure that you are using the latest version of the AWS CLI or SDK. The latest version provides you with the latest features and functionality, as well as security updates.

Alternatively, use AWS CloudShell, which provides an always up-to-date command line experience, automatically.

To list all the scrapers in your account, use the ListScrapers API operation.

Alternatively, with the AWS CLI, call:

```
aws amp list-scrapers --region aws-region
```

`ListScrapers` returns all of the scrapers in your account, for example:

```
{
    "scrapers": [
        {
            "scraperId": "s-1234abcd-56ef-7890-abcd-1234ef567890",
            "arn": "arn:aws:aps:us-west-2:123456789012:scraper/s-1234abcd-56ef-7890-
abcd-1234ef567890",
            "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
AWSServiceRoleForAmazonPrometheusScraper_1234abcd-2931",
            "status": {
                "statusCode": "DELETING"
            },
            "createdAt": "2023-10-12T15:22:19.014000-07:00",
            "lastModifiedAt": "2023-10-12T15:55:43.487000-07:00",
            "tags": {},
            "source": {
                "eksConfiguration": {
                    "clusterArn": "arn:aws:eks:us-west-2:123456789012:cluster/my-
cluster",
                    "securityGroupIds": [
                        "sg-1234abcd5678ef90"
                    ],
                    "subnetIds": [
                        "subnet-abcd1234ef567890",
                        "subnet-1234abcd5678ab90"
                    ]
                }
            },
            "destination": {
                "ampConfiguration": {
                    "workspaceArn": "arn:aws:aps:us-west-2:123456789012:workspace/
ws-1234abcd-5678-ef90-ab12-cdef3456a78"
```

```
                }
            }
        }
    ]
}
```

To delete a scraper, find the `scraperId` for the scraper that you want to delete, using the `ListScrapers` operation, and then use the DeleteScraper operation to delete it.

Alternatively, with the AWS CLI, call:

```
aws amp delete-scraper --scraper-id scraperId
```

## Scraper configuration

You can control how your scraper discovers and collects metrics with a Prometheus-compatible scraper configuration. For example, you can change the interval that metrics are sent to the workspace. You can also use relabeling to dynamically rewrite the labels of a metric. The scraper configuration is a YAML file that is part of the definition of the scraper.

When a new scraper is created, you specify a configuration by providing a base64 encoded YAML file in the API call. You can download a general purpose configuration file with the `GetDefaultScraperConfiguration` operation in the Amazon Managed Service for Prometheus API.

To modify the configuration of a scraper, you can use the `UpdateScraper` operation. If you need to update the source of the metrics (for example, to a different Amazon EKS cluster), you must delete the scraper and recreate it with the new source.

### Supported configuration

For information about the scraper configuration format, including a detailed breakdown of the possible values, see Configuration in the Prometheus documentation. The global configuration options, and `<scrape_config>` options describe the most commonly needed options.

Because Amazon EKS is the only supported service, the only service discovery config (`<*_sd_config>`) supported is the `<kubernetes_sd_config>`.

The complete list of config sections allowed:

- `<global>`

- <scrape_config>
- <static_config>
- <relabel_config>
- <metric_relabel_configs>
- <kubernetes_sd_config>

Limitations within these sections are listed after the sample configuration file.

**Sample configuration file**

The following is a sample YAML configuration file with a 30 second scrape interval. This sample includes support for the kube API server metrics, as well as kube-controller-manager and kube-scheduler metrics. For more information, see Fetch control plane raw metrics in Prometheus format in the *Amazon EKS User Guide*.

```
global:
   scrape_interval: 30s
   external_labels:
     clusterArn: apiserver-test-2
scrape_configs:
  - job_name: pod_exporter
    kubernetes_sd_configs:
      - role: pod
  - job_name: cadvisor
    scheme: https
    authorization:
      type: Bearer
      credentials_file: /var/run/secrets/kubernetes.io/serviceaccount/token
    kubernetes_sd_configs:
      - role: node
    relabel_configs:
      - action: labelmap
        regex: __meta_kubernetes_node_label_(.+)
      - replacement: kubernetes.default.svc:443
        target_label: __address__
      - source_labels: [__meta_kubernetes_node_name]
        regex: (.+)
        target_label: __metrics_path__
        replacement: /api/v1/nodes/$1/proxy/metrics/cadvisor
  # apiserver metrics
  - scheme: https
```

```
    authorization:
      type: Bearer
      credentials_file: /var/run/secrets/kubernetes.io/serviceaccount/token
    job_name: kubernetes-apiservers
    kubernetes_sd_configs:
    - role: endpoints
    relabel_configs:
    - action: keep
      regex: default;kubernetes;https
      source_labels:
      - __meta_kubernetes_namespace
      - __meta_kubernetes_service_name
      - __meta_kubernetes_endpoint_port_name
  # kube proxy metrics
  - job_name: kube-proxy
    honor_labels: true
    kubernetes_sd_configs:
    - role: pod
    relabel_configs:
    - action: keep
      source_labels:
      - __meta_kubernetes_namespace
      - __meta_kubernetes_pod_name
      separator: '/'
      regex: 'kube-system/kube-proxy.+'
    - source_labels:
      - __address__
      action: replace
      target_label: __address__
      regex: '(.+?)(:\d+)?'
      replacement: $1:10249
  # Scheduler metrics
  - job_name: 'ksh-metrics'
    kubernetes_sd_configs:
    - role: endpoints
    metrics_path: /apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics
    scheme: https
    bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
    relabel_configs:
    - source_labels:
      - __meta_kubernetes_namespace
      - __meta_kubernetes_service_name
      - __meta_kubernetes_endpoint_port_name
      action: keep
```

```
      regex: default;kubernetes;https
  # Controller Manager metrics
  - job_name: 'kcm-metrics'
    kubernetes_sd_configs:
    - role: endpoints
    metrics_path: /apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics
    scheme: https
    bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
    relabel_configs:
    - source_labels:
      - __meta_kubernetes_namespace
      - __meta_kubernetes_service_name
      - __meta_kubernetes_endpoint_port_name
      action: keep
      regex: default;kubernetes;https
```

The following are limitations specific to AWS managed collectors:

- **Scrape interval** – The scraper config can't specify a scrape interval of less than 30 seconds.
- **Targets** – Targets in the `static_config` must be specified as IP addresses.
- **DNS resolution** – Related to the target name, the only server name that is recognized in this config is the Kubernetes api server, `kubernetes.default.svc`. All other machines names must be specified by IP address.
- **Authorization** – Omit if no authorization is needed. If it is needed, the authorization must be `Bearer`, and must point to the file `/var/run/secrets/kubernetes.io/serviceaccount/token`. In other words, if used, the authorization section must look like the following:

  ```
    authorization:
      type: Bearer
      credentials_file: /var/run/secrets/kubernetes.io/serviceaccount/token
  ```

  **Note**

  `type: Bearer` is the default, so can be omitted.

## Troubleshooting scraper configuration

Amazon Managed Service for Prometheus collectors automatically discover and scrape metrics. But how can you troubleshoot when you don't see a metric you expect to see in your Amazon Managed Service for Prometheus workspace?

> **Important**
>
> Verify that private access for your Amazon EKS cluster is enabled. For more information, see Cluster private endpoint in the *Amazon EKS User Guide*.

The up metric is a helpful tool. For each endpoint that an Amazon Managed Service for Prometheus collector discovers, it automatically vends this metric. There are three states of this metric that can help you to troubleshoot what is happening within the collector.

- up is not present – If there is no up metric present for an endpoint, then that means that the collector was not able to find the endpoint.

  If you are sure that the endpoint exists, there are several reasons why the collector might not be able to find it.

  - You might need to adjust the scrape configuration. The discovery `relabel_config` might need to be adjusted.
  - There could be a problem with the `role` used for discovery.
  - The Amazon VPC used by the Amazon EKS cluster might not have DNS enabled, which would keep the collector from finding the endpoint.
- up is present, but is always 0 – If up is present, but 0, then the collector is able to discover the endpoint, but can't find any Prometheus-compatible metrics.

  In this case, you might try using a `curl` command against the endpoint directly. You can validate that you have the details correct, for example, the protocol (`http` or `https`), the endpoint, or port that you are using. You can also check that the endpoint is responding with a valid `200` response, and follows the Prometheus format. Finally, the body of the response can't be larger than the maximum allowed size. (For limits on AWS managed collectors, see the following section.)
- up is present and greater than 0 – If up is present, and is greater than 0, then metrics are being sent to Amazon Managed Service for Prometheus.

Validate that you are looking for the correct metrics in Amazon Managed Service for Prometheus (or your alternate dashboard, such as Amazon Managed Grafana). You can use curl again to check for expected data in your `/metrics` endpoint. Also check that you haven't exceeded other limits, such as the number of endpoints per scraper. You can check the number of metrics endpoints being scraped by checking the count of up metrics, using `count(up)`.

### Scraper limitations

There are few limitations to the fully managed scrapers provided by Amazon Managed Service for Prometheus.

- **Region** – Your EKS cluster, managed scraper, and Amazon Managed Service for Prometheus workspace must all be in the same AWS Region.
- **Collectors** – You can have a maximum of 10 Amazon Managed Service for Prometheus scrapers per region per account.

> **Note**
>
> You can request an increase to this limit by requesting a quota increase.

- **Metrics response** – The body of a response from any one `/metrics` endpoint request cannot be more than 50 megabytes (MB).
- **Endpoints per scraper** – A scraper can scrape a maximum of 30,000 `/metrics` endpoints.
- **Scrape interval** – The scraper config can't specify a scrape interval of less than 30 seconds.

## Set up managed Prometheus collectors for Amazon MSK

To use an Amazon Managed Service for Prometheus collector, you create a scraper that discovers and pulls metrics in your Amazon Managed Streaming for Apache Kafka cluster. You can also create a scraper that integrates with Amazon Elastic Kubernetes Service or with Amazon OpenSearch Service. For more information, see Integrate Amazon EKS and Integrate Amazon OpenSearch Service.

For instructions on collecting consumer-group lag, committed offsets, and membership with a managed Kafka exporter, see Collect Amazon MSK consumer-group metrics. That procedure uses

AWS Identity and Access Management (IAM) authentication and does not require Open Monitoring. The following procedure collects JMX and Node Exporter metrics from Open Monitoring.

## Create a scraper

An Amazon Managed Service for Prometheus collector consists of a scraper that discovers and collects metrics from an Amazon MSK cluster. Amazon Managed Service for Prometheus manages the scraper for you, giving you the scalability, security, and reliability that you need, without having to manage any instances, agents, or scrapers yourself.

You can create a scraper using either the AWS API or the AWS CLI as described in the following procedures.

There are a few prerequisites for creating your own scraper:

- You must have an Amazon MSK cluster created.
- Configure your Amazon MSK cluster's security group to allow inbound traffic on ports **11001 (JMX Exporter)** and **11002 (Node Exporter)** within your Amazon VPC, as the scraper requires access to these DNS records to collect Prometheus metrics.
- The Amazon VPC in which the Amazon MSK cluster resides must have DNS enabled.

> **Note**
>
> The cluster will be associated with the scraper by its Amazon resource name (ARN). If you delete a cluster, and then create a new one with the same name, the ARN will be reused for the new cluster. Because of this, the scraper will attempt to collect metrics for the new cluster. You delete scrapers separately from deleting the cluster.

To create a scraper using the AWS API

Use the `CreateScraper` API operation to create a scraper with the AWS API. The following example creates a scraper in the US East (N. Virginia) Region. Replace the *`example`* content with your Amazon MSK cluster information, and provide your scraper configuration.

> **Note**
>
> Configure the security group and subnets to match your target cluster. Include at least two subnets across two availability zones.

```
               POST /scrapers HTTP/1.1
Content-Length: 415
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: aws-cli/1.18.147 Python/2.7.18 Linux/5.4.58-37.125.amzn2int.x86_64
 botocore/1.18.6

{
    "alias": "myScraper",
    "destination":  {
        "ampConfiguration": {
            "workspaceArn": "arn:aws-eusc:aps:eusc-de-
east-1:123456789012:workspace/ws-workspace-id"
        }
    },
    "source": {
        "vpcConfiguration": {
            "securityGroupIds": ["sg-security-group-id"],
            "subnetIds": ["subnet-subnet-id-1", "subnet-subnet-id-2"]
        }
    },
    "scrapeConfiguration": {
        "configurationBlob": base64-encoded-blob
    }
}
```

In the example, the `scrapeConfiguration` parameter requires a base64-encoded Prometheus configuration YAML file that specifies the DNS records of the MSK cluster.

Each DNS record represents a broker endpoint in a specific Availability Zone, allowing clients to connect to brokers distributed across your chosen AZs for high availability.

The number of DNS records in your MSK cluster properties corresponds to the number of broker nodes and Availability Zones in your cluster configuration:

- **Default configuration** – 3 broker nodes across 3 AZs = 3 DNS records
- **Custom configuration** – 2 broker nodes across 2 AZs = 2 DNS records

To get the DNS records for your MSK cluster, open the MSK console at https://eusc-de-east-1.console.amazonaws-eusc.eu/msk/home?region=us-east-1#/home/. Go to your MSK cluster. Choose **Properties**, **Brokers**, and **Endpoints**.

You have two options for configuring Prometheus to scrape metrics from your MSK cluster:

1. **Cluster-level DNS resolution (Recommended)** – Use the cluster's base DNS name to automatically discover all brokers. If your broker endpoint is `b-1.clusterName.xxx.xxx.xxx`, use `clusterName.xxx.xxx.xxx` as the DNS record. This allows Prometheus to automatically scrape all brokers in the cluster.

   **Individual broker endpoints** – Specify each broker endpoint individually for granular control. Use the full broker identifiers (b-1, b-2) in your configuration. For example:

```
dns_sd_configs:
  - names:
    - b-1.clusterName.xxx.xxx.xxx
    - b-2.clusterName.xxx.xxx.xxx
    - b-3.clusterName.xxx.xxx.xxx
```

> **Note**
>
> Replace `clusterName.xxx.xxx.xxx` with your actual MSK cluster endpoint from the AWS Console.

For more information, see <dns_sd_config> in the *Prometheus* documentation.

The following is an example of the scraper configuration file:

```
global:
  scrape_interval: 30s
  external_labels:
    clusterArn: msk-test-1
```

```
scrape_configs:
  - job_name: msk-jmx
    scheme: http
    metrics_path: /metrics
    scrape_timeout: 10s
    dns_sd_configs:
      - names:
          - dns-record-1
          - dns-record-2
          - dns-record-3
        type: A
        port: 11001
    relabel_configs:
      - source_labels: [__meta_dns_name]
        target_label: broker_dns
      - source_labels: [__address__]
        target_label: instance
        regex: '(.*)'
        replacement: '${1}'

  - job_name: msk-node
    scheme: http
    metrics_path: /metrics
    scrape_timeout: 10s
    dns_sd_configs:
      - names:
          - dns-record-1
          - dns-record-2
          - dns-record-3
        type: A
        port: 11002
    relabel_configs:
      - source_labels: [__meta_dns_name]
        target_label: broker_dns
      - source_labels: [__address__]
        target_label: instance
        regex: '(.*)'
        replacement: '${1}'
```

Run one of the following commands to convert the YAML file to base64. You can also use any online base64 converter to convert the file.

**Example Linux/macOS**

```
echo -n scraper config updated with dns records | base64
```

**Example Windows PowerShell**

```
[Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes(scraper config
 updated with dns records))
```

To create a scraper using the AWS CLI

Use the `create-scraper` command to create a scraper using the AWS Command Line Interface. The following example creates a scraper in the US East (N. Virginia) Region. Replace the *example* content with your Amazon MSK cluster information, and provide your scraper configuration.

> **Note**
>
> Configure the security group and subnets to match your target cluster. Include at least two subnets across two availability zones.

```
aws amp create-scraper \
 --source vpcConfiguration="{securityGroupIds=['sg-security-group-
id'],subnetIds=['subnet-subnet-id-1', 'subnet-subnet-id-2']}" \
--scrape-configuration configurationBlob=base64-encoded-blob \
 --destination ampConfiguration="{workspaceArn='arn:aws-eusc:aps:us-
west-2:123456789012:workspace/ws-workspace-id'}"
```

- The following is a full list of the scraper operations that you can use with the AWS API:

  Create a scraper with the CreateScraper API operation.
- List your existing scrapers with the ListScrapers API operation.
- Update the alias, configuration, or destination of a scraper with the UpdateScraper API operation.
- Delete a scraper with the DeleteScraper API operation.
- Get more details about a scraper with the DescribeScraper API operation.

## Cross-account setup

To create a scraper in a cross-account setup when your Amazon MSK cluster from which you want to collect metrics is in a different account from the Amazon Managed Service for Prometheus collector, use the procedure below.

For example, when you have two accounts, the first source account `account_id_source` where the Amazon MSK is located, and a second target account `account_id_target` where the Amazon Managed Service for Prometheus workspace resides.

**To create a scraper in a cross-account setup**

1. In the source account, create a role `arn:aws-eusc:iam::111122223333:role/Source` and add the following trust policy.

```
{
    "Effect": "Allow",
    "Principal": {
    "Service": [
        "scraper.aps.amazonaws.com"
     ]
    },
    "Action": "sts:AssumeRole",
    "Condition": {
        "ArnEquals": {
            "aws:SourceArn": "arn:aws-eusc:aps:aws-
region:111122223333:scraper/scraper-id"
        },
        "StringEquals": {
            "AWS:SourceAccount": "111122223333"
        }
    }
}
```

2. On every combination of source (Amazon MSK cluster) and target (Amazon Managed Service for Prometheus workspace), you need to create a role `arn:aws-eusc:iam::444455556666:role/Target` and add the following trust policy with permissions for AmazonPrometheusRemoteWriteAccess.

```
{
```

```
  "Effect": "Allow",
  "Principal": {
     "AWS": "arn:aws-eusc:iam::111122223333:role/Source"
  },
  "Action": "sts:AssumeRole",
  "Condition": {
     "StringEquals": {
        "sts:ExternalId": "arn:aws-eusc:aps:aws-
region:111122223333:scraper/scraper-id"
      }
  }
}
```

3. Create a scraper with the `--role-configuration` option.

```
aws amp create-scraper \ --source vpcConfiguration="{subnetIds=[subnet-
subnet-id], "securityGroupIds": ["sg-security-group-id"]}" \ --
scrape-configuration configurationBlob=<base64-encoded-blob> \ --
destination ampConfiguration="{workspaceArn='arn:aws-eusc:aps:aws-
region:444455556666:workspace/ws-workspace-id'}"\ --role-configuration
 '{"sourceRoleArn":"arn:aws-eusc:iam::111122223333:role/Source",
 "targetRoleArn":"arn:aws-eusc:iam::444455556666:role/Target"}'
```

4. Validate the scraper creation.

```
aws amp list-scrapers
{
    "scrapers": [
        {
            "scraperId": "s-example123456789abcdef0",
            "arn": "arn:aws-eusc:aps:aws-region:111122223333:scraper/s-
example123456789abcdef0": "arn:aws-eusc:iam::111122223333:role/Source",
            "status": "ACTIVE",
            "creationTime": "2025-10-27T18:45:00.000Z",
            "lastModificationTime": "2025-10-27T18:50:00.000Z",
            "tags": {},
            "statusReason": "Scraper is running successfully",
            "source": {
                "vpcConfiguration": {
                    "subnetIds": ["subnet-subnet-id"],
                    "securityGroupIds": ["sg-security-group-id"]
```

```
                }
            },
            "destination": {
                "ampConfiguration": {
                    "workspaceArn": "arn:aws-eusc:aps:aws-
region:444455556666:workspace/ws-workspace-id'"
                }
            },
            "scrapeConfiguration": {
                "configurationBlob": "<base64-encoded-blob>"
            }
        }
    ]
}
```

## Changing between RoleConfiguration and service-linked role

When you want to switch back to a service-linked role instead of the `RoleConfiguration` to write to an Amazon Managed Service for Prometheus workspace, you must update the `UpdateScraper` and provide a workspace in the same account as the scraper without the `RoleConfiguration`. The `RoleConfiguration` will be removed from the scraper and the service-linked role will be used.

When you are changing workspaces in the same account as the scraper and you want to continue using the `RoleConfiguration`, you must again provide the `RoleConfiguration` on `UpdateScraper`.

## Find and delete scrapers

You can use the AWS API or the AWS CLI to list the scrapers in your account or to delete them.

> **Note**
>
> Make sure that you are using the latest version of the AWS CLI or SDK. The latest version provides you with the latest features and functionality, as well as security updates.

Alternatively, use AWS CloudShell, which provides an always up-to-date command line experience, automatically.

To list all the scrapers in your account, use the ListScrapers API operation.

Alternatively, with the AWS CLI, call:

```
aws amp list-scrapers
```

`ListScrapers` returns all of the scrapers in your account, for example:

```
{
    "scrapers": [
        {
            "scraperId": "s-1234abcd-56ef-7890-abcd-1234ef567890",
            "arn": "arn:aws-eusc:aps:aws-region:123456789012:scraper/
s-1234abcd-56ef-7890-abcd-1234ef567890",
            "roleArn": "arn:aws-eusc:iam::123456789012:role/aws-service-role/
AWSServiceRoleForAmazonPrometheusScraper_1234abcd-2931",
            "status": {
                "statusCode": "DELETING"
            },
            "createdAt": "2023-10-12T15:22:19.014000-07:00",
            "lastModifiedAt": "2023-10-12T15:55:43.487000-07:00",
            "tags": {},
            "source": {
                "vpcConfiguration": {
                   "securityGroupIds": [
                        "sg-1234abcd5678ef90"
                    ],
                    "subnetIds": [
                        "subnet-abcd1234ef567890",
                        "subnet-1234abcd5678ab90"
                    ]
                }
            },
            "destination": {
                "ampConfiguration": {
                    "workspaceArn": "arn:aws-eusc:aps:aws-
region:123456789012:workspace/ws-1234abcd-5678-ef90-ab12-cdef3456a78"
                }
```

```
            }
        }
    ]
}
```

To delete a scraper, find the `scraperId` for the scraper that you want to delete, using the `ListScrapers` operation, and then use the DeleteScraper operation to delete it.

Alternatively, with the AWS CLI, call:

```
aws amp delete-scraper --scraper-id scraperId
```

## Metrics collected from Amazon MSK

When you integrate with Amazon MSK, the Amazon Managed Service for Prometheus collector automatically scrapes the following metrics:

**Metrics: jmx_exporter and pod_exporter jobs**

| Metric | Description / Purpose |
|---|---|
| jmx_config_reload_failure_total | Total number of times the JMX exporter failed to reload its configuration file. |
| jmx_scrape_duration_seconds | Time taken to scrape JMX metrics in seconds for the current collection cycle. |
| jmx_scrape_error | Indicates whether an error occurred during JMX metric scraping (1 = error, 0 = success). |
| java_lang_Memory_HeapMemoryUsage_used | Amount of heap memory (in bytes) currently used by the JVM. |
| java_lang_Memory_HeapMemoryUsage_max | Maximum amount of heap memory (in bytes) that can be used for memory management. |
| java_lang_Memory_NonHeapMemoryUsage_used | Amount of non-heap memory (in bytes) currently used by the JVM. |

| Metric | Description / Purpose |
| --- | --- |
| kafka_cluster_Partition_Value | Current state or value related to Kafka cluster partitions, broken down by partition ID and topic. |
| kafka_consumer_consumer_coordinator_metrics_assigned_partitions | Number of partitions currently assigned to this consumer. |
| kafka_consumer_consumer_coordinator_metrics_commit_latency_avg | Average time taken to commit offsets in milliseconds. |
| kafka_consumer_consumer_coordinator_metrics_commit_rate | Number of offset commits per second. |
| kafka_consumer_consumer_coordinator_metrics_failed_rebalance_total | Total number of failed consumer group rebalances. |
| kafka_consumer_consumer_coordinator_metrics_last_heartbeat_seconds_ago | Number of seconds since the last heartbeat was sent to the coordinator. |
| kafka_consumer_consumer_coordinator_metrics_rebalance_latency_avg | Average time taken for consumer group rebalances in milliseconds. |
| kafka_consumer_consumer_coordinator_metrics_rebalance_total | Total number of consumer group rebalances. |
| kafka_consumer_consumer_fetch_manager_metrics_bytes_consumed_rate | Average number of bytes consumed per second by the consumer. |
| kafka_consumer_consumer_fetch_manager_metrics_fetch_latency_avg | Average time taken for a fetch request in milliseconds. |
| kafka_consumer_consumer_fetch_manager_metrics_fetch_rate | Number of fetch requests per second. |
| kafka_consumer_consumer_fetch_manager_metrics_records_consumed_rate | Average number of records consumed per second. |

| Metric | Description / Purpose |
| --- | --- |
| kafka_consumer_consumer_fetch_manager_metrics_records_lag_max | Maximum lag in terms of number of records for any partition in this consumer. |
| kafka_consumer_consumer_metrics_connection_count | Current number of active connections. |
| kafka_consumer_consumer_metrics_incoming_byte_rate | Average number of bytes received per second from all servers. |
| kafka_consumer_consumer_metrics_last_poll_seconds_ago | Number of seconds since the last consumer poll() call. |
| kafka_consumer_consumer_metrics_request_rate | Number of requests sent per second. |
| kafka_consumer_consumer_metrics_response_rate | Number of responses received per second. |
| kafka_consumer_group_ConsumerLagMetrics_Value | Current consumer lag value for a consumer group, indicating how far behind the consumer is. |
| kafka_controller_KafkaController_Value | Current state or value of the Kafka controller (1 = active controller, 0 = not active). |
| kafka_controller_ControllerEventManager_Count | Total number of controller events processed. |
| kafka_controller_ControllerEventManager_Mean | Mean (average) time taken to process controller events. |
| kafka_controller_ControllerStats_MeanRate | Mean rate of controller statistics operations per second. |
| kafka_coordinator_group_GroupMetadataManager_Value | Current state or value of the group metadata manager for consumer groups. |
| kafka_log_LogFlushStats_Count | Total number of log flush operations. |

| Metric | Description / Purpose |
| --- | --- |
| kafka_log_LogFlushStats_Mean | Mean (average) time taken for log flush operations. |
| kafka_log_LogFlushStats_MeanRate | Mean rate of log flush operations per second. |
| kafka_network_RequestMetrics_Count | Total count of network requests processed. |
| kafka_network_RequestMetrics_Mean | Mean (average) time taken to process network requests. |
| kafka_network_RequestMetrics_MeanRate | Mean rate of network requests per second. |
| kafka_network_Acceptor_MeanRate | Mean rate of accepted connections per second. |
| kafka_server_Fetch_queue_size | Current size of the fetch request queue. |
| kafka_server_Produce_queue_size | Current size of the produce request queue. |
| kafka_server_Request_queue_size | Current size of the general request queue. |
| kafka_server_BrokerTopicMetrics_Count | Total count of broker topic operations (messages in/out, bytes in/out). |
| kafka_server_BrokerTopicMetrics_MeanRate | Mean rate of broker topic operations per second. |
| kafka_server_BrokerTopicMetrics_OneMinuteRate | One-minute moving average rate of broker topic operations. |
| kafka_server_DelayedOperationPurgatory_Value | Current number of delayed operations in the purgatory (waiting to be completed). |
| kafka_server_DelayedFetchMetrics_MeanRate | Mean rate of delayed fetch operations per second. |
| kafka_server_FetcherLagMetrics_Value | Current lag value for replica fetcher threads (how far behind the leader). |

| Metric | Description / Purpose |
| --- | --- |
| kafka_server_FetcherStats_MeanRate | Mean rate of fetcher operations per second. |
| kafka_server_ReplicaManager_Value | Current state or value of the replica manager. |
| kafka_server_ReplicaManager_MeanRate | Mean rate of replica manager operations per second. |
| kafka_server_LeaderReplication_byte_rate | Rate of bytes replicated per second for partitions where this broker is the leader. |
| kafka_server_group_coordinator_metrics_group_completed_rebalance_count | Total number of completed consumer group rebalances. |
| kafka_server_group_coordinator_metrics_offset_commit_count | Total number of offset commit operations. |
| kafka_server_group_coordinator_metrics_offset_commit_rate | Rate of offset commit operations per second. |
| kafka_server_socket_server_metrics_connection_count | Current number of active connections. |
| kafka_server_socket_server_metrics_connection_creation_rate | Rate of new connection creation per second. |
| kafka_server_socket_server_metrics_connection_close_rate | Rate of connection closures per second. |
| kafka_server_socket_server_metrics_failed_authentication_total | Total number of failed authentication attempts. |
| kafka_server_socket_server_metrics_incoming_byte_rate | Rate of incoming bytes per second. |
| kafka_server_socket_server_metrics_outgoing_byte_rate | Rate of outgoing bytes per second. |

| Metric | Description / Purpose |
| --- | --- |
| kafka_server_socket_server_metrics_request_rate | Rate of requests per second. |
| kafka_server_socket_server_metrics_response_rate | Rate of responses per second. |
| kafka_server_socket_server_metrics_network_io_rate | Rate of network I/O operations per second. |
| kafka_server_socket_server_metrics_io_ratio | Fraction of time spent in I/O operations. |
| kafka_server_controller_channel_metrics_connection_count | Current number of active connections for controller channels. |
| kafka_server_controller_channel_metrics_incoming_byte_rate | Rate of incoming bytes per second for controller channels. |
| kafka_server_controller_channel_metrics_outgoing_byte_rate | Rate of outgoing bytes per second for controller channels. |
| kafka_server_controller_channel_metrics_request_rate | Rate of requests per second for controller channels. |
| kafka_server_replica_fetcher_metrics_connection_count | Current number of active connections for replica fetcher. |
| kafka_server_replica_fetcher_metrics_incoming_byte_rate | Rate of incoming bytes per second for replica fetcher. |
| kafka_server_replica_fetcher_metrics_request_rate | Rate of requests per second for replica fetcher. |
| kafka_server_replica_fetcher_metrics_failed_authentication_total | Total number of failed authentication attempts for replica fetcher. |
| kafka_server_ZooKeeperClientMetrics_Count | Total count of ZooKeeper client operations. |
| kafka_server_ZooKeeperClientMetrics_Mean | Mean latency of ZooKeeper client operations. |

| Metric | Description / Purpose |
| --- | --- |
| kafka_server_KafkaServer_Value | Current state or value of the Kafka server (typically indicates server is running). |
| node_cpu_seconds_total | Total seconds the CPUs spent in each mode (user, system, idle, etc.), broken down by CPU and mode. |
| node_disk_read_bytes_total | Total number of bytes read successfully from disks, broken down by device. |
| node_disk_reads_completed_total | Total number of reads completed successfully for disks, broken down by device. |
| node_disk_writes_completed_total | Total number of writes completed successfully for disks, broken down by device. |
| node_disk_written_bytes_total | Total number of bytes written successfully to disks, broken down by device. |
| node_filesystem_avail_bytes | Available filesystem space in bytes for non-root users, broken down by device and mount point. |
| node_filesystem_size_bytes | Total size of the filesystem in bytes, broken down by device and mount point. |
| node_filesystem_free_bytes | Free filesystem space in bytes, broken down by device and mount point. |
| node_filesystem_files | Total number of file nodes (inodes) on the filesystem, broken down by device and mount point. |
| node_filesystem_files_free | Number of free file nodes (inodes) on the filesystem, broken down by device and mount point. |

| Metric | Description / Purpose |
| --- | --- |
| node_filesystem_readonly | Indicates whether the filesystem is mounted read-only (1 = read-only, 0 = read-write). |
| node_filesystem_device_error | Indicates whether an error occurred while getting filesystem statistics (1 = error, 0 = success). |

## Collect Amazon MSK consumer-group metrics

With a managed Kafka exporter, an Amazon Managed Service for Prometheus collector collects consumer-group metrics from your Amazon MSK cluster, such as consumer lag, committed offsets, and the number of members in each consumer group. Amazon Managed Service for Prometheus runs the exporter for you, so you do not have to install or operate an exporter yourself. Your destination can be an Amazon Managed Service for Prometheus workspace or Amazon CloudWatch. These metrics complement the JMX and Node Exporter metrics that the collector scrapes from Open Monitoring.

The scraper connects to your cluster with AWS Identity and Access Management (IAM) authentication. You grant the scraper access to the cluster with an Amazon MSK cluster policy. To collect consumer-group metrics, create a scrape configuration, create a scraper, and then grant the scraper access to your cluster, as described in the following sections.

For information about the metrics, see Consumer-group metrics. To send the metrics to Amazon CloudWatch, see Collect Amazon MSK consumer-group metrics in CloudWatch.

### Prerequisites

There are a few prerequisites for collecting consumer-group metrics:

- An active Amazon MSK Provisioned cluster with Standard brokers and Apache Kafka version 3.6.0 or later, in the same account and AWS Region as the scraper.
- IAM access control turned on for the cluster. You can keep TLS client authentication turned on at the same time, so clients that use certificates can continue to connect.
- At least two subnets in different Availability Zones in the cluster's Amazon VPC, with DNS enabled.

- Security groups that allow the scraper to reach the cluster's private IAM broker endpoints on port 9098. Add an inbound rule to the cluster's security group that allows TCP traffic on port 9098 from the security group that you provide for the scraper.
- An Amazon Managed Service for Prometheus workspace in the scraper's Region, and an AWS CLI version that supports `kafkaConfiguration` in `create-scraper` and `update-scraper`.
- Permission to create or update a scraper and to validate the cluster: `kafka:DescribeClusterV2`, `kafka:GetBootstrapBrokers`, `kafka-cluster:Connect`, and `kafka-cluster:DescribeTopic`. To add the cluster policy, you also need `kafka:GetClusterPolicy` and `kafka:PutClusterPolicy`. For the permissions to create the scraper's service-linked role, see Service-linked role permissions.

The scraper uses its service-linked role to connect to the cluster, so you do not provide a role for the exporter. By default, the role has no permissions for your cluster. You grant them in Grant the scraper access to the cluster.

The managed Kafka exporter does not support the following clusters:

- Amazon MSK Serverless clusters. Serverless clusters do not accept cluster policies with `kafka-cluster` actions. Use the metrics that Amazon MSK sends to CloudWatch instead.
- Clusters that do not have IAM authentication turned on.
- Clusters in a different account or Region from the scraper.

Open Monitoring and broker ports 11001 and 11002 are needed only if you also collect JMX and Node Exporter metrics.

**Create a scrape configuration**

You provide a scrape configuration when you create the scraper. The managed collector connects to the cluster that you specify and collects its metrics automatically, so you do not specify scrape targets in the configuration. The configuration must include a `scrape_configs` section with a job whose `job_name` is exactly `kafka-exporter`. A configuration that does not contain a job named `kafka-exporter` is rejected. The collector sets the scrape interval for this job.

Use the scrape configuration to add labels and to filter the collected metrics. The following example keeps consumer lag, committed offsets, and group membership. It also adds a `clusterArn` label so that you can tell clusters apart when you query. Replace the cluster ARN with

your cluster's ARN, and save the configuration as `msk-consumer.yaml`. Use the same cluster ARN when you create the scraper.

```
global:
  scrape_interval: 30s
scrape_configs:
  - job_name: kafka-exporter
    relabel_configs:
      - target_label: clusterArn
        replacement: 'arn:aws:kafka:us-west-2:111122223333:cluster/my-cluster/cluster-
id'
    metric_relabel_configs:
      - source_labels: [__name__]
        regex: kafka_consumergroup_(current_offset|members|lag)
        action: keep
```

The exporter can also send other metrics, such as topic and partition offsets and partition counts. Amazon MSK already provides some of this information through Open Monitoring and the metrics that Amazon MSK sends to CloudWatch. To collect more exporter metrics, add them to the `regex`, for example `kafka_consumergroup_(current_offset|members|lag)|kafka_topic_partitions`. Each metric that you add increases the number of series that you ingest.

To add the exporter to an existing scraper, add the `kafka-exporter` job to the scraper's full scrape configuration, and keep its other jobs. Then update the scraper as described in Update or remove the exporter. You can relabel metrics in this job, but you cannot use relabel rules to change how the collector connects to the exporter.

**Create a scraper**

To tell the scraper which cluster to collect from, you specify the cluster in the `exporters` field of the request. For an Amazon MSK cluster, provide a `kafkaConfiguration` with the `clusterArn` of the cluster. You can also set a `topicFilter` and a `consumerGroupFilter` to select topics and consumer groups by name. You provide the networking (subnets and security group) separately in the `source` field.

Use the `create-scraper` command to create a scraper using the AWS Command Line Interface. The following example selects topics that start with `orders` and consumer groups that start with `payments`. Replace the *`example`* content, including both filters, with your own cluster, networking, workspace, and filter values. To select all topics or all consumer groups, omit the filter.

> **Note**
>
> Configure the security group and subnets to match the Amazon VPC of your Amazon MSK cluster. Include at least two subnets across two Availability Zones.

```
aws amp create-scraper \
  --region us-west-2 \
  --alias msk-consumer-metrics \
  --source '{"vpcConfiguration":{"subnetIds":["subnet-subnet-id-1","subnet-subnet-
id-2"],"securityGroupIds":["sg-security-group-id"]}}' \
  --exporters '[{"kafkaConfiguration":{"clusterArn":"arn:aws:kafka:us-
west-2:111122223333:cluster/my-cluster/cluster-id","topicFilter":"^orders.*
$","consumerGroupFilter":"^payments.*$"}}]' \
  --cli-binary-format base64 \
  --scrape-configuration "configurationBlob=$(base64 < msk-consumer.yaml | tr -d '\n')"
 \
  --destination '{"ampConfiguration":{"workspaceArn":"arn:aws:aps:us-
west-2:111122223333:workspace/workspace-id"}}'
```

Both filters are optional. If you omit a filter, the scraper selects all topics or all consumer groups. Each filter must be a regular expression that starts with ^ and ends with $, with a maximum length of 256 characters. A scraper can have one exporter.

The `topicFilter` selects the topics that the exporter reads topic metrics for. Consumer-group metrics can also include other topics that a selected consumer group consumes. To control which metrics reach your destination, use `metric_relabel_configs` in your scrape configuration.

Record the scraper ID. Check its status with the following command, and continue when the status is `ACTIVE`:

```
aws amp describe-scraper \
  --region us-west-2 \
  --scraper-id scraper-id \
  --query 'scraper.status.statusCode' --output text
```

The scraper does not collect Kafka metrics until you grant it access to the cluster.

### Grant the scraper access to the cluster

The scraper needs permission to connect to your cluster and to read its topic and consumer-group metadata. Add an Amazon MSK cluster policy that allows the scraper's service-linked role the following actions:

- `kafka-cluster:Connect` on the cluster
- `kafka-cluster:DescribeTopic` on its topics
- `kafka-cluster:DescribeGroup` on its consumer groups

Without this policy, the scraper is `ACTIVE` but does not collect Kafka metrics. After you add the policy, the scraper starts collecting automatically, and the first metrics can take a few minutes to appear. You do not need to update the scraper.

Grant the scraper's service-linked role, even if the scraper uses `roleConfiguration` to send metrics to a workspace in another account. Do not grant the source role or the target role.

### Grant one scraper

Get the scraper's role ARN with the following AWS CLI command:

```
aws amp describe-scraper \
  --region us-west-2 \
  --scraper-id scraper-id \
  --query scraper.roleArn --output text
```

Each scraper has its own role, `AWSServiceRoleForAmazonPrometheusScraper_`*`suffix`*. If you re-create a scraper, grant the new role.

Save the following policy as `policy.json`. Replace the role ARN, Region, account ID, cluster name, and cluster ID.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "AWS": "scraper-role-arn" },
      "Action": "kafka-cluster:Connect",
      "Resource": "arn:aws:kafka:us-west-2:111122223333:cluster/my-cluster/cluster-id"
```

```
    },
    {
      "Effect": "Allow",
      "Principal": { "AWS": "scraper-role-arn" },
      "Action": "kafka-cluster:DescribeTopic",
      "Resource": "arn:aws:kafka:us-west-2:111122223333:topic/my-cluster/cluster-id/*"
    },
    {
      "Effect": "Allow",
      "Principal": { "AWS": "scraper-role-arn" },
      "Action": "kafka-cluster:DescribeGroup",
      "Resource": "arn:aws:kafka:us-west-2:111122223333:group/my-cluster/cluster-id/*"
    }
  ]
}
```

To grant another scraper, add the three statements again with that scraper's role ARN. To grant fewer topics or consumer groups, edit the policy as described in Limit the topics and consumer groups before you apply it.

**Apply the policy**

> **Important**
>
> Do not use an account root principal, such as `arn:aws:iam::111122223333:root`. Amazon MSK accepts it, but it grants the scraper nothing. Do not use a wildcard in a role ARN as the principal. Amazon MSK rejects it with the error `The cluster policy is not valid`. To grant every scraper in the account, see Optional: Grant every scraper in the account.

To read and change the cluster policy, you need `kafka:GetClusterPolicy` and `kafka:PutClusterPolicy`.

**To add the cluster policy**

1. Check whether the cluster already has a policy:

```
aws kafka get-cluster-policy \
  --region us-west-2 \
  --cluster-arn arn:aws:kafka:us-west-2:111122223333:cluster/my-cluster/cluster-id
```

A `NotFoundException` error means that the cluster has no policy.

2. If the cluster has no policy, add yours:

```
aws kafka put-cluster-policy \
  --region us-west-2 \
  --cluster-arn arn:aws:kafka:us-west-2:111122223333:cluster/my-cluster/cluster-id
 \
  --policy file://policy.json
```

3. If the cluster already has a policy, merge the statements. `put-cluster-policy` replaces the whole policy document. The `Policy` value from `get-cluster-policy` is the existing policy as a JSON string. Add its statements to the `Statement` list in `policy.json`, so that the file has both the existing statements and the new ones. Then pass the `CurrentVersion` value from `get-cluster-policy`:

```
aws kafka put-cluster-policy \
  --region us-west-2 \
  --cluster-arn arn:aws:kafka:us-west-2:111122223333:cluster/my-cluster/cluster-id
 \
  --current-version current-version \
  --policy file://policy.json
```

**Optional: Grant every scraper in the account**

We recommend that you grant access to individual scraper role ARNs. If you need to grant every current and future scraper in the account, you can use the following policy instead. You can add it before you create a scraper, and it keeps working when you replace one. Keep both conditions in every statement. Save the following policy as `policy.json`. Replace the Region, account ID, cluster name, and cluster ID in each statement, and apply it as described in Apply the policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "AWS": "*" },
      "Action": "kafka-cluster:Connect",
      "Resource": "arn:aws:kafka:us-west-2:111122223333:cluster/my-cluster/cluster-id",
      "Condition": {
```

```
        "StringEquals": { "aws:PrincipalAccount": "111122223333" },
        "ArnLike": { "aws:PrincipalArn": "arn:aws:iam::111122223333:role/aws-service-
role/scraper.aps.amazonaws.com/AWSServiceRoleForAmazonPrometheusScraper_*" }
      }
    },
    {
      "Effect": "Allow",
      "Principal": { "AWS": "*" },
      "Action": "kafka-cluster:DescribeTopic",
      "Resource": "arn:aws:kafka:us-west-2:111122223333:topic/my-cluster/cluster-id/*",
      "Condition": {
        "StringEquals": { "aws:PrincipalAccount": "111122223333" },
        "ArnLike": { "aws:PrincipalArn": "arn:aws:iam::111122223333:role/aws-service-
role/scraper.aps.amazonaws.com/AWSServiceRoleForAmazonPrometheusScraper_*" }
      }
    },
    {
      "Effect": "Allow",
      "Principal": { "AWS": "*" },
      "Action": "kafka-cluster:DescribeGroup",
      "Resource": "arn:aws:kafka:us-west-2:111122223333:group/my-cluster/cluster-id/*",
      "Condition": {
        "StringEquals": { "aws:PrincipalAccount": "111122223333" },
        "ArnLike": { "aws:PrincipalArn": "arn:aws:iam::111122223333:role/aws-service-
role/scraper.aps.amazonaws.com/AWSServiceRoleForAmazonPrometheusScraper_*" }
      }
    }
  ]
}
```

This policy allows every current and future scraper in the account. The `aws:PrincipalAccount` and `ArnLike` conditions are what limit it. Do not remove either condition.

**Limit the topics and consumer groups**

Both policies grant all topics and consumer groups in the cluster. To grant fewer, replace the final `*` in the topic and group ARNs, for example `topic/my-cluster/cluster-id/orders-*`. These ARN patterns are IAM wildcards, not the `topicFilter` or `consumerGroupFilter` regular expressions. If the policy grants fewer topics or groups than the filters select, the scraper reports fewer topics and offsets.

To narrow a policy that you already applied, replace your earlier statements with the narrowed ones. Do not merge them, because the earlier statements still grant all topics and groups.

### Validate metrics collection

To confirm that the scraper is collecting metrics, run each of the following PromQL queries separately against your workspace. For more information, see Query your Prometheus metrics. Replace the cluster ARN. Metrics appear only for consumer groups that match your consumer-group filter and that have members or committed offsets.

```
kafka_consumergroup_members{clusterArn="arn:aws:kafka:us-west-2:111122223333:cluster/my-cluster/cluster-id"}
```

```
kafka_consumergroup_current_offset{clusterArn="arn:aws:kafka:us-west-2:111122223333:cluster/my-cluster/cluster-id"}
```

```
kafka_consumergroup_lag{clusterArn="arn:aws:kafka:us-west-2:111122223333:cluster/my-cluster/cluster-id"}
```

The first metrics can take a few minutes to appear after you add the cluster policy. If no metrics appear after that, check the following. An empty result does not mean that consumer lag is zero.

- If the scraper is `ACTIVE`, but `up{job="kafka-exporter"}` is `0` and no `kafka_*` metrics appear, check the cluster policy. A successful create or update request does not mean that the scraper can access the cluster. Run `aws kafka get-cluster-policy`, and confirm that the policy grants `kafka-cluster:Connect`, `kafka-cluster:DescribeTopic`, and `kafka-cluster:DescribeGroup` to the scraper's current service-linked role ARN. If you re-created the scraper, its role ARN changed. If you use the account-wide policy, confirm that every statement keeps both `StringEquals` on `aws:PrincipalAccount` and `ArnLike` on `aws:PrincipalArn`. You do not need to update the scraper after you fix the policy.
- If creation or update fails with a permissions error, give the caller the scraper and cluster permissions in Prerequisites, then retry the command.
- If the scraper cannot connect to the brokers, turn on IAM authentication for the cluster. Check VPC DNS and allow the scraper's security group to reach the private IAM broker endpoints on port 9098.
- If the scraper is `ACTIVE` but the queries return no metrics, check that `consumerGroupFilter` matches an existing consumer group. Keep the metric names and the `clusterArn`, `consumergroup`, `topic`, and `partition` labels in the scrape configuration. Use the same cluster ARN in the configuration and query. If the cluster policy grants fewer topics or groups than the filters select, fewer topics and offsets appear.

- If the cluster policy is correct but `up{job="kafka-exporter"}` is still `0` on a cluster with many topics and consumer groups, try narrowing `topicFilter` and `consumerGroupFilter`.

### Update or remove the exporter

To change the filters, update the scraper with its full scrape configuration. The following command selects one topic and one consumer group. Keep the `kafka-exporter` job and the `clusterArn` label in your scrape configuration.

```
aws amp update-scraper \
  --region us-west-2 \
  --scraper-id scraper-id \
  --exporters '[{"kafkaConfiguration":{"clusterArn":"arn:aws:kafka:us-
west-2:111122223333:cluster/my-cluster/cluster-id","topicFilter":"^orders
$","consumerGroupFilter":"^payments$"}}]' \
  --cli-binary-format base64 \
  --scrape-configuration "configurationBlob=$(base64 < msk-consumer.yaml | tr -d '\n')"
```

Omitting `exporters` in an update keeps the current exporter. To remove it, pass an empty list, `--exporters '[]'`, and send the full scrape configuration with the `kafka-exporter` job removed. Keep any other jobs that you still need.

The following AWS CLI command deletes the scraper and stops all of its metric collection. It does not delete the Amazon MSK cluster or the destination workspace.

```
aws amp delete-scraper \
  --region us-west-2 \
  --scraper-id scraper-id
```

### Remove the scraper's access to the cluster

Removing the cluster policy does not stop collection right away. To stop collection, remove the exporter or delete the scraper first. Then remove the policy statements.

#### To remove the scraper's access

1. Note the scraper's role ARN. Then remove the exporter or delete the scraper, as described earlier in this section.

2. Run `aws amp describe-scraper`. Confirm that the scraper status is `ACTIVE` and that the scraper has no exporter, or that the scraper no longer exists. Other scrapers can continue to send metrics for the same cluster.
3. Run `aws kafka get-cluster-policy`. Remove only the statements that you added for the scraper. Keep statements that grant access to other clients. If you used the account-wide policy and other scrapers still need access, replace its statements with statements that name the role ARN of each scraper that keeps access.
4. Run `aws kafka put-cluster-policy` with the remaining statements and the `CurrentVersion` value, as in Apply the policy. If no statements remain, delete the policy instead with `aws kafka delete-cluster-policy`. This command needs the `kafka:DeleteClusterPolicy` permission.

## Consumer-group metrics

When you collect consumer-group metrics with the example configuration in Collect Amazon MSK consumer-group metrics, the Amazon Managed Service for Prometheus collector sends the following metrics. Each metric has the `clusterArn` label from the scrape configuration, so that you can tell clusters apart.

| Metric | Description / Purpose |
| --- | --- |
| kafka_consumergroup_lag | Difference between the newest offset in a partition and the consumer group's committed offset. This is approximately the number of messages that the group has not committed yet. Labels: `consumergroup` , `topic`, `partition` . |
| kafka_consumergroup_current_offset | Last offset that the consumer group committed for a partition. Labels: `consumergroup` , `topic`, `partition` . |
| kafka_consumergroup_members | Number of members in the consumer group. Label: `consumergroup` . |

A value of -1 for lag or committed offset means that the consumer group has not committed an offset for that partition yet. Leave these values out when you add up lag. For example, the following query shows the total lag of each consumer group for each topic:

```
sum by (clusterArn, consumergroup, topic) (kafka_consumergroup_lag >= 0)
```

This total does not include partitions where the consumer group has not committed an offset.

The exporter calculates lag from the offsets that it reads from your cluster. Its values can differ from the consumer-lag metrics that Amazon MSK provides, because each reads offsets at different times.

### Limitations

The current Amazon MSK integration with Amazon Managed Service for Prometheus has the following limitations:

- Only supported for Amazon MSK Provisioned clusters (not available for Amazon MSK Serverless)
- Not supported for Amazon MSK clusters with public access enabled in combination with KRaft metadata mode
- Currently supports a 1:1 mapping between Amazon MSK clusters and Amazon Managed Service for Prometheus collectors/workspaces

## Set up managed Prometheus collectors for Amazon OpenSearch Service

The Amazon Managed Service for Prometheus managed collector for Amazon OpenSearch Service automatically scrapes Prometheus-compatible metrics from your OpenSearch Service domain and forwards them to your destination. Amazon Managed Service for Prometheus manages the collector for you, giving you the scalability, security, and reliability that you need, without having to manage any instances, agents, or scrapers yourself. Your destination can be an Amazon Managed Service for Prometheus workspace or Amazon CloudWatch.

You can also create a scraper that integrates with Amazon Elastic Kubernetes Service or with Amazon Managed Streaming for Apache Kafka. For more information, see Integrate Amazon EKS and Integrate Amazon MSK.

## Create a scraper

This procedure assumes that you are familiar with Amazon OpenSearch Service domain administration and Amazon Virtual Private Cloud networking concepts.

There are a few prerequisites for creating a scraper for an OpenSearch Service domain:

- An Amazon OpenSearch Service domain with Amazon Virtual Private Cloud access. Managed collectors support only OpenSearch Service domains that have VPC access. Domains with public access are not supported.
- Security groups that allow the managed collector to reach your OpenSearch Service domain endpoint over HTTPS (port 443). Add an inbound rule to the domain's security group that allows HTTPS traffic from the security group that you provide for the collector.

To tell the scraper which OpenSearch Service domain to collect from, you specify the domain in the `exporters` field of the request. The `exporters` field takes a list of exporter configurations; for an OpenSearch Service domain, provide an `openSearchConfiguration` with the `domainArn` of the domain. You provide the networking (subnets and security group) separately in the `source` field, as shown in the following examples.

> **Note**
>
> The scraper uses the domain's VPC endpoint as resolved when you create the scraper. If you recreate the domain, it gets a new endpoint, so you must create a new scraper or update the existing one with UpdateScraper to collect from it.

You provide a scrape configuration when you create the scraper. The managed collector connects to the domain that you specify and collects its metrics automatically, so you do not specify scrape targets in the configuration. The configuration must include a `scrape_configs` section with a job whose `job_name` is exactly `opensearch-exporter`. A configuration that does not contain a job named `opensearch-exporter` is rejected. Use the scrape configuration to set the scrape interval and, optionally, to filter or relabel the collected metrics. The following is an example scrape configuration:

```
global:
  external_labels:
    domain_name: my-opensearch-domain
```

```
scrape_configs:
  - job_name: opensearch-exporter
    scrape_interval: 60s
```

For more information about the supported scrape configuration options, see Scraper configuration.

To create a scraper using the AWS API

Use the `CreateScraper` API operation to create a scraper with the AWS API. The following example creates a scraper in the US East (N. Virginia) Region that collects metrics from an OpenSearch Service domain and sends them to an Amazon Managed Service for Prometheus workspace. Replace the *`example`* content with your own domain, networking, and workspace information, and provide your scraper configuration.

> **Note**
>
> Configure the security group and subnets to match the Amazon VPC of your OpenSearch Service domain. Include at least two subnets across two Availability Zones.

```
POST /scrapers HTTP/1.1

{
    "alias": "myScraper",
    "source": {
        "vpcConfiguration": {
            "securityGroupIds": ["sg-security-group-id"],
            "subnetIds": ["subnet-subnet-id-1", "subnet-subnet-id-2"]
        }
    },
    "exporters": [
        {
            "openSearchConfiguration": {
                "domainArn": "arn:aws-eusc:es:eusc-de-east-1:123456789012:domain/my-
opensearch-domain"
            }
        }
    ],
    "destination": {
        "ampConfiguration": {
```

```
            "workspaceArn": "arn:aws-eusc:aps:eusc-de-
east-1:123456789012:workspace/ws-workspace-id"
        }
    },
    "scrapeConfiguration": {
        "configurationBlob": "base64-encoded-blob"
    }
}
```

To send the metrics to Amazon CloudWatch instead of an Amazon Managed Service for Prometheus workspace, replace the `destination` with a CloudWatch configuration:

```
    "destination": {
        "cloudWatchConfiguration": {
            "datasetArn": "arn:aws-eusc:cloudwatch:eusc-de-
east-1:123456789012:dataset/default"
        }
    }
```

The `scrapeConfiguration` parameter requires a base64-encoded Prometheus configuration YAML file. Run one of the following commands to convert the YAML file to base64. You can also use any online base64 converter to convert the file.

**Example Linux/macOS**

```
base64 -w0 scraper-config.yaml
```

**Example Windows PowerShell**

```
[Convert]::ToBase64String([System.IO.File]::ReadAllBytes("scraper-config.yaml"))
```

To create a scraper using the AWS CLI

Use the `create-scraper` command to create a scraper using the AWS Command Line Interface. The following example creates a scraper in the US East (N. Virginia) Region. Replace the *example* content with your own domain, networking, and workspace information, and provide your scraper configuration.

> **Note**
>
> Configure the security group and subnets to match the Amazon VPC of your OpenSearch Service domain. Include at least two subnets across two Availability Zones.

```
aws amp create-scraper \
 --source '{"vpcConfiguration":{"securityGroupIds":["sg-security-group-id"],"subnetIds":["subnet-subnet-id-1","subnet-subnet-id-2"]}}' \
 --exporters '[{"openSearchConfiguration":{"domainArn":"arn:aws-eusc:es:eusc-de-east-1:123456789012:domain/my-opensearch-domain"}}]' \
 --scrape-configuration configurationBlob=base64-encoded-blob \
 --destination '{"ampConfiguration":{"workspaceArn":"arn:aws-eusc:aps:eusc-de-east-1:123456789012:workspace/ws-workspace-id"}}'
```

- The following is a full list of the scraper operations that you can use with the AWS API:

  Create a scraper with the CreateScraper API operation.
- List your existing scrapers with the ListScrapers API operation.
- Update the alias, configuration, or destination of a scraper with the UpdateScraper API operation.
- Delete a scraper with the DeleteScraper API operation.
- Get more details about a scraper with the DescribeScraper API operation.

## Cross-account setup

To create a scraper in a cross-account setup, when the OpenSearch Service domain that you want to collect metrics from is in a different account from the Amazon Managed Service for Prometheus collector, use the following procedure.

For example, you have two accounts: a source account `account_id_source` where the OpenSearch Service domain is located, and a target account `account_id_target` where the Amazon Managed Service for Prometheus workspace resides.

**Note**

The trust policies in the following steps refer to the scraper by its ARN, which includes a *scraper-id* that does not exist until you create the scraper. To avoid this ordering problem, first create the roles with a wildcard (`scraper/*`) in place of the specific scraper ARN, create the scraper, and then update both trust policies to replace the wildcard with the actual scraper ARN that `CreateScraper` returns.

**To create a scraper in a cross-account setup**

1. In the source account, create a role `arn:aws-eusc:iam::`*111122223333*`:role/Source` and add the following trust policy.

```
{
    "Effect": "Allow",
    "Principal": {
    "Service": [
        "scraper.aps.amazonaws.com"
     ]
    },
    "Action": "sts:AssumeRole",
    "Condition": {
        "ArnEquals": {
            "aws:SourceArn": "arn:aws-eusc:aps:aws-
region:111122223333:scraper/scraper-id"
        },
        "StringEquals": {
            "AWS:SourceAccount": "111122223333"
        }
    }
}
```

2. In the target account, create a role `arn:aws-eusc:iam::`*444455556666*`:role/Target` and add the following trust policy, which allows the source role to assume it.

```
{
  "Effect": "Allow",
  "Principal": {
     "AWS": "arn:aws-eusc:iam::111122223333:role/Source"
  },
```

```
  "Action": "sts:AssumeRole",
  "Condition": {
     "StringEquals": {
        "sts:ExternalId": "arn:aws-eusc:aps:aws-
region:111122223333:scraper/scraper-id"
      }
  }
}
```

Attach a permissions policy to the target role that allows it to write to your destination. If your destination is an Amazon Managed Service for Prometheus workspace, attach AmazonPrometheusRemoteWriteAccess (which grants `aps:RemoteWrite`). If your destination is Amazon CloudWatch, attach a policy that grants `cloudwatch:PutMetricData` on the dataset.

3. Create a scraper with the `--role-configuration` option.

```
aws amp create-scraper \
 --source '{"vpcConfiguration":{"securityGroupIds":["sg-security-group-
id"],"subnetIds":["subnet-subnet-id-1","subnet-subnet-id-2"]}}' \
 --exporters '[{"openSearchConfiguration":{"domainArn":"arn:aws-eusc:es:aws-
region:111122223333:domain/my-opensearch-domain"}}]' \
 --scrape-configuration configurationBlob=<base64-encoded-blob> \
 --destination '{"ampConfiguration":{"workspaceArn":"arn:aws-eusc:aps:aws-
region:444455556666:workspace/ws-workspace-id"}}' \
 --role-configuration '{"sourceRoleArn":"arn:aws-eusc:iam::111122223333:role/
Source", "targetRoleArn":"arn:aws-eusc:iam::444455556666:role/Target"}'
```

4. Validate the scraper creation.

```
aws amp list-scrapers
```

## Find and delete scrapers

You can use the AWS API or the AWS CLI to list the scrapers in your account or to delete them.

> **Note**
>
> Make sure that you are using the latest version of the AWS CLI or SDK. The latest version provides you with the latest features and functionality, as well as security updates.

Alternatively, use AWS CloudShell, which provides an always up-to-date command line experience, automatically.

To list all the scrapers in your account, use the ListScrapers API operation. Alternatively, with the AWS CLI, call:

```
aws amp list-scrapers
```

To delete a scraper, find the `scraperId` for the scraper that you want to delete, using the `ListScrapers` operation, and then use the DeleteScraper operation to delete it. Alternatively, with the AWS CLI, call:

```
aws amp delete-scraper --scraper-id scraperId
```

## Metrics collected from Amazon OpenSearch Service

When you integrate with Amazon OpenSearch Service, the Amazon Managed Service for Prometheus collector automatically scrapes Prometheus-compatible metrics that describe the health and performance of your domain. The collector emits several hundred metrics; the following tables list representative metrics from each category. The `opensearch_indices_` metrics in the following tables are aggregated across the domain, and many have primary-only and total variants (suffixed `_primary` or `_total`). The collector also emits the same statistics per index, prefixed `opensearch_index_stats_`, and per-shard metrics prefixed `opensearch_indices_shards_`.

### Cluster metrics

| Metric | Description / Purpose |
|---|---|
| opensearch_cluster_health_status | Cluster health status (green, yellow, or red). |
| opensearch_cluster_health_number_of_nodes | Number of nodes in the cluster. |
| opensearch_cluster_health_number_of_data_nodes | Number of data nodes in the cluster. |
| opensearch_cluster_health_active_primary_shards | Number of active primary shards. |

| Metric | Description / Purpose |
| --- | --- |
| opensearch_cluster_health_active_shards | Total number of active shards, including primary and replica shards. |
| opensearch_cluster_health_relocating_shards | Number of shards that are relocating between nodes. |
| opensearch_cluster_health_initializing_shards | Number of shards that are initializing. |
| opensearch_cluster_health_unassigned_shards | Number of shards that are not assigned to a node. |
| opensearch_cluster_health_delayed_unassigned_shards | Number of unassigned shards whose assignment is delayed. |
| opensearch_cluster_health_number_of_pending_tasks | Number of cluster-level tasks that are queued and waiting to run. |
| opensearch_cluster_health_number_of_in_flight_fetch | Number of shard fetch requests in progress. |
| opensearch_cluster_health_task_max_waiting_in_queue_millis | Longest time that a task has waited in the queue, in milliseconds. |

**Node metrics**

| Metric | Description / Purpose |
| --- | --- |
| opensearch_os_cpu_percent | Percentage of CPU used by the operating system on the node. |
| opensearch_os_load1 | Operating system load average over the last 1 minute. The collector also emits `opensearch_os_load5` and `opensearch_os_load15`. |
| opensearch_os_mem_used_bytes | Amount of physical memory used, in bytes. The collector also emits opensearc |

| Metric | Description / Purpose |
| --- | --- |
| | h_os_mem_free_bytes , opensearc h_os_mem_actual_used_bytes , and opensearch_os_mem_actual_fr ee_bytes . |
| opensearch_jvm_memory_used_bytes | Amount of JVM memory used, in bytes, by memory area (heap and non-heap). Related metrics include opensearch_jvm_mem ory_committed_bytes and opensearc h_jvm_memory_max_bytes . |
| opensearch_jvm_gc_collection_seconds_count | Total number of JVM garbage collection events. Use with opensearch_jvm_gc_ collection_seconds_sum , the total time spent in garbage collection. |
| opensearch_jvm_uptime_seconds | JVM uptime, in seconds. |
| opensearch_thread_pool_active_count | Number of active threads in each thread pool. Related metrics include opensearc h_thread_pool_queue_count , opensearch_thread_pool_reje cted_count , and opensearch_thread_ pool_completed_count . |
| opensearch_filesystem_data_available_bytes | Amount of disk space available to the node, in bytes. Related metrics include opensearc h_filesystem_data_free_bytes and opensearch_filesystem_data_ size_bytes . |

| Metric | Description / Purpose |
| --- | --- |
| opensearch_filesystem_io_stats_device_read_operations_count | Number of disk read operations. The collector also emits write and total operation counts and read and write sizes (for example, `opensearch_filesystem_io_stats_device_read_size_kilobytes_sum`). |
| opensearch_process_cpu_percent | Percentage of CPU used by the OpenSearch process. |
| opensearch_process_mem_resident_size_bytes | Resident memory size of the OpenSearch process, in bytes. |
| opensearch_process_open_files_count | Number of file descriptors open by the OpenSearch process. |
| opensearch_breakers_tripped | Total number of times each circuit breaker has tripped. Related metrics include `opensearch_breakers_estimated_size_bytes` and `opensearch_breakers_limit_size_bytes` . |
| opensearch_transport_rx_size_bytes_total | Total amount of data received over the transport layer between nodes, in bytes. The collector also emits `opensearch_transport_tx_size_bytes_total` and packet counts. |
| opensearch_indexing_pressure_current_all_in_bytes | Current memory consumed by indexing requests, in bytes, with `opensearch_indexing_pressure_limit_in_bytes` as the configured limit. |

### Index metrics

| Metric | Description / Purpose |
| --- | --- |
| opensearch_indices_docs | Number of documents. Related metrics include `opensearch_indices_docs_deleted` , `opensearch_indices_docs_primary` , and `opensearch_indices_docs_total` . |
| opensearch_indices_store_size_bytes | Total size of the indices on disk, in bytes, with `opensearch_indices_store_size_bytes_primary` and `opensearch_indices_store_size_bytes_total` variants. |
| opensearch_indices_indexing_index_total | Total number of documents indexed. Use with `opensearch_indices_indexing_index_time_seconds_total` for indexing latency. |
| opensearch_indices_indexing_is_throttled | Indicates whether indexing is currently throttled, with `opensearch_indices_indexing_throttle_time_seconds_total` for the total throttled time. |
| opensearch_indices_search_query_total | Total number of search queries. Use with `opensearch_indices_search_query_time_seconds` for query latency. |
| opensearch_indices_search_fetch_total | Total number of fetch operations, with `opensearch_indices_search_fetch_time_seconds` for fetch latency. |
| opensearch_indices_get_total | Total number of get operations, with `opensearch_indices_get_time_seconds` for get latency. |

| Metric | Description / Purpose |
| --- | --- |
| opensearch_indices_merges_total | Total number of segment merges completed . Related metrics include opensearch_indices_merges_current and opensearch_indices_merges_total_time_seconds_total . |
| opensearch_indices_refresh_total | Total number of index refresh operations, with opensearch_indices_refresh_time_seconds_total for refresh time. |
| opensearch_indices_flush_total | Total number of index flush operations, with opensearch_indices_flush_time_seconds for flush time. |
| opensearch_indices_segments_count | Number of segments. The collector also emits per-segment memory metrics (for example, opensearch_indices_segments_memory_bytes and opensearch_indices_segment_terms_memory_total ). |
| opensearch_indices_translog_operations | Number of operations in the transaction log, with opensearch_indices_translog_size_in_bytes for its size. |
| opensearch_indices_fielddata_memory_size_bytes | Memory used by the field data cache, in bytes, with opensearch_indices_fielddata_evictions for evictions. |
| opensearch_indices_query_cache_memory_size_bytes | Memory used by the query cache, in bytes. Related metrics include opensearch_indices_query_cache_count and opensearch_indices_query_cache_evictions . |

| Metric | Description / Purpose |
| --- | --- |
| opensearch_indices_request_cache_memory_size_bytes | Memory used by the request cache, in bytes. Related metrics include `opensearch_indices_request_cache_count` and `opensearch_indices_request_cache_evictions` . |

### Limitations

The Amazon OpenSearch Service integration with Amazon Managed Service for Prometheus has the following limitations:

- Only supported for OpenSearch Service domains that have Amazon Virtual Private Cloud access. Domains with public access are not supported.
- A scraper collects metrics from a single OpenSearch Service domain. To collect metrics from more than one domain, create a separate scraper for each domain.

## What are Prometheus-compatible metrics?

To scrape Prometheus metrics from your applications and infrastructure for use in Amazon Managed Service for Prometheus, they must instrument and expose *Prometheus-compatible metrics* from Prometheus-compatible `/metrics` endpoints. You can implement your own metrics, but you don't have to. Kubernetes (including Amazon EKS) and many other libraries and services implement these metrics directly.

When metrics in Amazon EKS are exported to a Prometheus-compatible endpoint, you can have those metrics automatically scraped by the Amazon Managed Service for Prometheus collector.

For more information, see the following topics:

- For more information about existing libraries and services that export metrics as Prometheus metrics, see Exporters and integrations in the Prometheus documentation.
- For more information about exporting Prometheus-compatible metrics from your own code, see Writing exporters in the Prometheus documentation.

- For more information about how to set up an Amazon Managed Service for Prometheus collector to scrape metrics from your Amazon EKS clusters automatically, see Set up managed collectors for Amazon EKS.

## Monitor collectors with vended logs

Amazon Managed Service for Prometheus collectors provide vended logs to help you monitor and troubleshoot the metrics collection process. These logs are automatically sent to Amazon CloudWatch Logs and provide visibility into service discovery, metric collection, and data export operations. The collector vends logs for three main components of the metrics collection pipeline:

**Topics**


- Service discovery logs
- Collector logs
- Exporter logs
- Understanding and using collector vended logs


### Service discovery logs

Service discovery logs provide information about the target discovery process, including:

- Authentication or permission issues when accessing Kubernetes API resources.
- Configuration errors in service discovery settings.

The following examples demonstrate common authentication and permission errors you might encounter during service discovery:

**Nonexistent Amazon EKS cluster**

When the specified Amazon EKS cluster does not exist, you receive the following error:

```
{
  "component": "SERVICE_DISCOVERY",
  "timestamp": "2025-04-30T17:25:41.946Z",
  "message": {
    "log": "Failed to watch Service - Verify your scraper source exists."
  },
  "scrapeConfigId": "s-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

```
}
```

**Invalid permissions for services**

When the collector lacks proper Role-Based Access Control (RBAC) permissions to watch Services, you receive this error:

```
{
  "component": "SERVICE_DISCOVERY",
  "timestamp": "2025-04-30T17:25:41.946Z",
  "message": {
    "log": "Failed to watch Service - Verify your scraper source permissions are
 valid."
  },
  "scrapeConfigId": "s-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

**Invalid permissions for endpoints**

When the collector lacks proper Role-Based Access Control (RBAC) permissions to watch Endpoints, you receive this error:

```
{
  "component": "SERVICE_DISCOVERY",
  "timestamp": "2025-04-30T17:25:41.946Z",
  "message": {
    "log": "Failed to watch Endpoints - Verify your scraper source permissions are
 valid."
  },
  "scrapeConfigId": "s-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

## Collector logs

Collector logs provide information about the metric scraping process, including:

- Scrape failures due to endpoints not being available.
- Connection issues when attempting to scrape targets.
- Timeouts during scrape operations.
- HTTP status errors returned by scrape targets.

The following examples demonstrate common collector errors you might encounter during the metric scraping process:

**Missing metrics endpoint**

When the `/metrics` endpoint is not available on the target instance, you receive this error:

```
{
    "component": "COLLECTOR",
    "message": {
        "log": "Failed to scrape Prometheus endpoint - verify /metrics endpoint is
 available",
        "job": "pod_exporter",
        "targetLabels": "{__name__=\"up\", instance=\10.24.34.0\", job=
\"pod_exporter\"}"
    },
    "timestamp": "1752787969551",
    "scraperId": "s-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

**Connection refused**

When the collector cannot establish a connection to the target endpoint, you receive this error:

```
{
  "scrapeConfigId": "s-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "timestamp": "2025-04-30T17:25:41.946Z",
  "message": {
    "message": "Scrape failed",
    "scrape_pool": "pod_exporter",
    "target": "http://10.24.34.0:80/metrics",
    "error": "Get \"http://10.24.34.0:80/metrics\": dial tcp 10.24.34.0:80: connect:
 connection refused"
  },
  "component": "COLLECTOR"
}
```

## Exporter logs

Exporter logs provide information about the process of sending collected metrics to your Amazon Managed Service for Prometheus workspace, including:

- Number of metrics and data points processed.
- Export failures due to workspace issues.
- Permission errors when attempting to write metrics.
- Dependency failures in the export pipeline.

The following example demonstrates a common exporter error you might encounter during the metric export process:

**Workspace not found**

When the target workspace for metric export cannot be found, you receive this error:

```
{
    "component": "EXPORTER",
    "message": {
        "log": "Failed to export to the target workspace - Verify your scraper
 destination.",
        "samplesDropped": 5
    },
    "timestamp": "1752787969664",
    "scraperId": "s-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

## Understanding and using collector vended logs

**Log structure**

All collector vended logs follow a consistent structure with these fields:

**scrapeConfigId**

The unique identifier of the scrape configuration that generated the log.

**timestamp**

The time when the log entry was generated.

**message**

The log message content, which may include additional structured fields.

**component**

The component that generated the log (SERVICE_DISCOVERY, COLLECTOR, or EXPORTER)

### Using vended logs for troubleshooting

The collector vended logs help you troubleshoot common issues with metrics collection:

1. Service discovery issues
   - Check **SERVICE_DISCOVERY** logs for authentication or permission errors.
   - Verify that the collector has the necessary permissions to access Kubernetes resources.
2. Metric scraping issues
   - Check **COLLECTOR** logs for scrape failures.
   - Verify that target endpoints are accessible and returning metrics.
   - Ensure that firewall rules allow the collector to connect to target endpoints.
3. Metric export issues
   - Check **EXPORTER** logs for export failures.
   - Verify that the workspace exists and is correctly configured.
   - Ensure that the collector has the necessary permissions to write to the workspace.

### Accessing collector vended logs

Collector vended logs are automatically sent to Amazon CloudWatch Logs. To access these logs:

1. Open the CloudWatch console at https://eusc-de-east-1.console.amazonaws-eusc.eu/cloudwatch/.
2. In the navigation pane, choose **Log groups**.
3. Find and select the log group for your collector: `/aws/prometheus/workspace_id/collector/collector_id`.
4. Browse or search the log events to find relevant information.

You can also use CloudWatch Logs Insights to query and analyze your collector logs. For example, to find all service discovery errors:

```
fields @timestamp, message.message
```

```
| filter component = "SERVICE_DISCOVERY" and message.message like /Failed/
| sort @timestamp desc
```

**Best practices for monitoring collectors**

To effectively monitor your Amazon Managed Service for Prometheus collectors:

1. Set up CloudWatch alarms for critical collector issues, such as persistent scrape failures or export errors. For more information, see Alarms in the *Amazon CloudWatch User Guide*.
2. Create CloudWatch dashboards to visualize collector performance metrics alongside vended log data. For more information, see Dashboards in the *Amazon CloudWatch User Guide*.
3. Regularly review service discovery logs to ensure targets are being discovered correctly.
4. Monitor the number of dropped targets to identify potential configuration issues.
5. Track export failures to ensure metrics are being successfully sent to your workspace.

# Customer managed collectors

This section contains information about ingesting data by setting up your own collectors that send metrics to Amazon Managed Service for Prometheus using Prometheus remote write.

When you use your own collectors to send metrics to Amazon Managed Service for Prometheus, you are responsible for securing your metrics and making sure that the ingestion process meets your availability needs.

Most customer managed collectors use one of the following tools:

- **AWS Distro for OpenTelemetry (ADOT)** – ADOT is a fully supported, secure, production-ready open source distribution of OpenTelemetry that provides agents to collect metrics. You can use ADOT to collect metrics and send them to your Amazon Managed Service for Prometheus workspace. For more information about the ADOT Collector, see AWS Distro for OpenTelemetry.
- **Prometheus agent** – You can set up your own instance of the open source Prometheus server, running as an agent, to collect metrics and forward them to your Amazon Managed Service for Prometheus workspace.

The following topics describe using both of these tools and include general information about setting up your own collectors.

**Topics**


- Secure the ingestion of your metrics
- Using AWS Distro for OpenTelemetry as a collector
- Using a Prometheus instance as a collector
- Set up Amazon Managed Service for Prometheus for high availability data


# Secure the ingestion of your metrics

Amazon Managed Service for Prometheus provides ways of helping you secure the ingestion of your metrics.

## Using AWS PrivateLink with Amazon Managed Service for Prometheus

The network traffic of ingesting the metrics into Amazon Managed Service for Prometheus can be done over a public internet endpoint, or by a VPC endpoint through AWS PrivateLink. Using AWS PrivateLink ensures that the network traffic from your VPCs is secured within the AWS network without going over the public internet. To create an AWS PrivateLink VPC endpoint for Amazon Managed Service for Prometheus, see Using Amazon Managed Service for Prometheus with interface VPC endpoints.

## Authentication and authorization

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources. Amazon Managed Service for Prometheus integrates with IAM to help you keep your data secure. When you set up Amazon Managed Service for Prometheus, you need to create some IAM roles that enable it to ingest metrics from Prometheus servers, and that enable Grafana servers to query the metrics that are stored in your Amazon Managed Service for Prometheus workspaces. For more information about IAM, see What is IAM?.

Another AWS security feature that can help you set up Amazon Managed Service for Prometheus is the AWS Signature Version 4 signing process (AWS SigV4). Signature Version 4 is the process to add authentication information to AWS requests sent by HTTP. For security, most requests to AWS must be signed with an access key, which consists of an access key ID and secret access key. These two keys are commonly referred to as your security credentials. For more information about SigV4, see Signature Version 4 signing process.

# Using AWS Distro for OpenTelemetry as a collector

This section describes how to configure the AWS Distro for OpenTelemetry (ADOT) Collector to scrape from a Prometheus-instrumented application, and send the metrics to Amazon Managed Service for Prometheus. For more information about the ADOT Collector, see AWS Distro for OpenTelemetry.

The following topics describe three different ways to set up ADOT as a collector for your metrics, based on whether your metrics are coming from Amazon EKS, Amazon ECS, or an Amazon EC2 instance.

**Topics**

- Set up metrics ingestion using AWS Distro for OpenTelemetry on an Amazon Elastic Kubernetes Service cluster
- Set up metrics ingestion from Amazon ECS using AWS Distro for Open Telemetry
- Set up metrics ingestion from an Amazon EC2 instance using remote write

## Set up metrics ingestion using AWS Distro for OpenTelemetry on an Amazon Elastic Kubernetes Service cluster

You can use the AWS Distro for OpenTelemetry (ADOT) collector to scrape metrics from a Prometheus-instrumented application, and send the metrics to Amazon Managed Service for Prometheus.

> **Note**
>
> For more information about the ADOT collector, see AWS Distro for OpenTelemetry. For more information about Prometheus-instrumented applications, see What are Prometheus-compatible metrics?.

Collecting Prometheus metrics with ADOT involves three OpenTelemetry components: the Prometheus Receiver, the Prometheus Remote Write Exporter, and the Sigv4 Authentication Extension.

You can configure the Prometheus Receiver using your existing Prometheus configuration to perform service discovery and metric scraping. The Prometheus Receiver scrapes metrics in the

Prometheus exposition format. Any applications or endpoints that you want to scrape should be configured with the Prometheus client library. The Prometheus Receiver supports the full set of Prometheus scraping and re-labeling configurations described in Configuration in the Prometheus documentation. You can paste these configurations directly into your ADOT Collector configurations.

The Prometheus Remote Write Exporter uses the `remote_write` endpoint to send the scraped metrics to your management portal workspace. The HTTP requests to export data will be signed with AWS SigV4, the AWS protocol for secure authentication, with the Sigv4 Authentication Extension. For more information, see Signature Version 4 signing process.

The collector automatically discovers Prometheus metrics endpoints on Amazon EKS and uses the configuration found in <kubernetes_sd_config>.

The following demo is an example of this configuration on a cluster running Amazon Elastic Kubernetes Service or self-managed Kubernetes. To perform these steps, you must have AWS credentials from any of the potential options in the default AWS credentials chain. For more information, see Configuring the AWS SDK for Go. This demo uses a sample app that is used for integration tests of the process. The sample app exposes metrics at the `/metrics` endpoint, like the Prometheus client library.

**Prerequisites**

Before you begin the following ingestion setup steps, you must set up your IAM role for the service account and trust policy.

**To set up the IAM role for service account and trust policy**

1. Create the IAM role for the service account by following the steps in Set up service roles for the ingestion of metrics from Amazon EKS clusters.

   The ADOT Collector will use this role when it scrapes and exports metrics.
2. Next, edit the trust policy. Open the IAM console at https://eusc-de-east-1.console.amazonaws-eusc.eu/iam/.
3. In the left navigation pane, choose **Roles** and find the **amp-iamproxy-ingest-role** that you created in step 1.
4. Choose the **Trust relationships** tab and choose **Edit trust relationship**.
5. In the trust relationship policy JSON, replace `aws-amp` with `adot-col` and then choose **Update Trust Policy**. Your resulting trust policy should look like the following:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws-eusc:iam::111122223333:oidc-provider/
oidc.eks.eusc-de-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.eusc-de-east-1.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:adot-
col:amp-iamproxy-ingest-service-account",
          "oidc.eks.eusc-de-east-1.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

6. Choose the **Permissions** tab and make sure that the following permissions policy is attached to the role.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aps:RemoteWrite",
                "aps:GetSeries",
                "aps:GetLabels",
                "aps:GetMetricMetadata"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

### Enabling Prometheus metric collection

> **Note**
>
> When you create a namespace in Amazon EKS, `alertmanager` and node exporter are disabled by default.

#### To enable Prometheus collection on an Amazon EKS or Kubernetes cluster

1. Fork and clone the sample app from the repository at aws-otel-community.

   Then run the following commands.

   ```
   cd ./sample-apps/prometheus-sample-app
   docker build . -t prometheus-sample-app:latest
   ```

2. Push this image to a registry such as Amazon ECR or DockerHub.
3. Deploy the sample app in the cluster by copying this Kubernetes configuration and applying it. Change the image to the image that you just pushed by replacing `{{PUBLIC_SAMPLE_APP_IMAGE}}` in the `prometheus-sample-app.yaml` file.

   ```
   curl https://raw.githubusercontent.com/aws-observability/aws-otel-collector/
   main/examples/eks/aws-prometheus/prometheus-sample-app.yaml -o prometheus-sample-
   app.yaml
   kubectl apply -f prometheus-sample-app.yaml
   ```

4. Enter the following command to verify that the sample app has started. In the output of the command, you will see `prometheus-sample-app` in the `NAME` column.

   ```
   kubectl get all -n aoc-prometheus-pipeline-demo
   ```

5. Start a default instance of the ADOT Collector. To do so, first enter the following command to pull the Kubernetes configuration for ADOT Collector.

```
curl https://raw.githubusercontent.com/aws-observability/aws-otel-collector/main/
examples/eks/aws-prometheus/prometheus-daemonset.yaml -o prometheus-daemonset.yaml
```

Then edit the template file, substituting the **remote_write** endpoint for your Amazon Managed Service for Prometheus workspace for `YOUR_ENDPOINT` and your Region for `YOUR_REGION`. Use the **remote_write** endpoint that is displayed in the Amazon Managed Service for Prometheus console when you look at your workspace details.

You'll also need to change `YOUR_ACCOUNT_ID` in the service account section of the Kubernetes configuration to your AWS account ID.

In this example, the ADOT Collector configuration uses an annotation (`scrape=true`) to tell which target endpoints to scrape. This allows the ADOT Collector to distinguish the sample app endpoint from kube-system endpoints in your cluster. You can remove this from the re-label configurations if you want to scrape a different sample app.

6. Enter the following command to deploy the ADOT collector.

```
kubectl apply -f prometheus-daemonset.yaml
```

7. Enter the following command to verify that the ADOT collector has started. Look for `adot-col` in the `NAMESPACE` column.

```
kubectl get pods -n adot-col
```

8. Verify that the pipeline works by using the logging exporter. Our example template is already integrated with the logging exporter. Enter the following commands.

```
kubectl get pods -A
kubectl logs -n adot-col name_of_your_adot_collector_pod
```

Some of the scraped metrics from the sample app will look like the following example.

```
Resource labels:
     -> service.name: STRING(kubernetes-service-endpoints)
     -> host.name: STRING(192.168.16.238)
     -> port: STRING(8080)
     -> scheme: STRING(http)
InstrumentationLibraryMetrics #0
```

```
Metric #0
Descriptor:
     -> Name: test_gauge0
     -> Description: This is my gauge
     -> Unit:
     -> DataType: DoubleGauge
DoubleDataPoints #0
StartTime: 0
Timestamp: 1606511460471000000
Value: 0.000000
```

9. To test whether Amazon Managed Service for Prometheus received the metrics, use `awscurl`. This tool enables you to send HTTP requests through the command line with AWS Sigv4 authentication, so you must have AWS credentials set up locally with the correct permissions to query from Amazon Managed Service for Prometheus For instructions on installing `awscurl`, see awscurl.

   In the following command, replace `AMP_REGION`, and `AMP_ENDPOINT` with the information for your Amazon Managed Service for Prometheus workspace.

   ```
   awscurl --service="aps" --region="AMP_REGION" "https://AMP_ENDPOINT/api/v1/query?
   query=adot_test_gauge0"
   {"status":"success","data":{"resultType":"vector","result":[{"metric":
   {"__name__":"adot_test_gauge0"},"value":[1606512592.493,"16.87214000011479"]}]}}
   ```

   If you receive a metric as the response, that means your pipeline setup has been successful and the metric has successfully propagated from the sample app into Amazon Managed Service for Prometheus.

### Cleaning up

To clean up this demo, enter the following commands.

```
kubectl delete namespace aoc-prometheus-pipeline-demo
kubectl delete namespace adot-col
```

### Advanced configuration

The Prometheus Receiver supports the full set of Prometheus scraping and re-labeling configurations described in Configuration in the Prometheus documentation. You can paste these configurations directly into your ADOT Collector configurations.

The configuration for the Prometheus Receiver includes your service discovery, scraping configurations, and re-labeling configurations. The receiver configuration looks like the following.

```
receivers:
  prometheus:
    config:
      [Your Prometheus configuration]
```

The following is an example configuration.

```
receivers:
  prometheus:
    config:
      global:
        scrape_interval: 1m
        scrape_timeout: 10s

      scrape_configs:
      - job_name: kubernetes-service-endpoints
        sample_limit: 10000
        kubernetes_sd_configs:
        - role: endpoints
        tls_config:
          ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
          insecure_skip_verify: true
        bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
```

If you have an existing Prometheus configuration, you must replace the $ characters with $$ to avoid having the values replaced with environment variables. *This is especially important for the replacement value of the relabel_configurations. For example, if you start with the following relabel_configuration:

```
relabel_configs:
- source_labels:
 [__meta_kubernetes_ingress_scheme,__address__,__meta_kubernetes_ingress_path]
```

```
  regex: (.+);(.+);(.+)
  replacement: ${1}://${2}${3}
  target_label: __param_target
```

It would become the following:

```
relabel_configs:
- source_labels:
 [__meta_kubernetes_ingress_scheme,__address__,__meta_kubernetes_ingress_path]
  regex: (.+);(.+);(.+)
  replacement: $${1}://${2}${3}
  target_label: __param_target
```

**Prometheus remote write exporter and Sigv4 authentication extension**

The configuration for the Prometheus Remote Write Exporter and Sigv4 Authentication Extension are simpler than the Prometheus receiver. At this stage in the pipeline, metrics have already been ingested, and we're ready to export this data to Amazon Managed Service for Prometheus. The minimum requirement for a successful configuration to communicate with Amazon Managed Service for Prometheus is shown in the following example.

```
extensions:
  sigv4auth:
    service: "aps"
    region: "user-region"
exporters:
  prometheusremotewrite:
    endpoint: "https://aws-managed-prometheus-endpoint/api/v1/remote_write"
    auth:
      authenticator: "sigv4auth"
```

This configuration sends an HTTPS request that is signed by AWS SigV4 using AWS credentials from the default AWS credentials chain. For more information, see Configuring the AWS SDK for Go. You must specify the service to be `aps`.

Regardless of the method of deployment, the ADOT collector must have access to one of the listed options in the default AWS credentials chain. The Sigv4 Authentication Extension depends on the AWS SDK for Go and uses it to fetch credentials and authenticate. You must ensure that these credentials have remote write permissions for Amazon Managed Service for Prometheus.

## Set up metrics ingestion from Amazon ECS using AWS Distro for Open Telemetry

This section explains how to collect metrics from Amazon Elastic Container Service (Amazon ECS) and ingest them into Amazon Managed Service for Prometheus using AWS Distro for Open Telemetry (ADOT). It also describes how to visualize your metrics in Amazon Managed Grafana.

### Prerequisites

**Important**

Before you begin, you must have an Amazon ECS environment on an AWS Fargate cluster with default settings, an Amazon Managed Service for Prometheus workspace, and an Amazon Managed Grafana workspace. We assume that you are familiar with container workloads, Amazon Managed Service for Prometheus, and Amazon Managed Grafana.

For more information, see the following links:

- For information about how to create an Amazon ECS environment on a Fargate cluster with default settings, see Creating a cluster in the *Amazon ECS Developer Guide*.
- For information about how to create an Amazon Managed Service for Prometheus workspace, see Create a workspace in the *Amazon Managed Service for Prometheus User Guide*.
- For information about how to create an Amazon Managed Grafana workspace, see Creating a workspace in the *Amazon Managed Grafana User Guide*.

### Step 1: Define a custom ADOT collector container image

Use the following config file as a template to define your own ADOT collector container image. Replace *`my-remote-URL`* and *`my-region`* with your `endpoint` and `region` values. Save the config in a file called *adot-config.yaml*.

**Note**

This configuration uses the `sigv4auth` extension to authenticate calls to Amazon Managed Service for Prometheus. For more information about configuring `sigv4auth`, see Authenticator - Sigv4 on GitHub.

```
receivers:
  prometheus:
    config:
      global:
        scrape_interval: 15s
        scrape_timeout: 10s
      scrape_configs:
      - job_name: "prometheus"
        static_configs:
        - targets: [ 0.0.0.0:9090 ]
  awsecscontainermetrics:
    collection_interval: 10s
processors:
  filter:
    metrics:
      include:
        match_type: strict
        metric_names:
          - ecs.task.memory.utilized
          - ecs.task.memory.reserved
          - ecs.task.cpu.utilized
          - ecs.task.cpu.reserved
          - ecs.task.network.rate.rx
          - ecs.task.network.rate.tx
          - ecs.task.storage.read_bytes
          - ecs.task.storage.write_bytes
exporters:
  prometheusremotewrite:
    endpoint: my-remote-URL
    auth:
      authenticator: sigv4auth
  logging:
    loglevel: info
extensions:
  health_check:
  pprof:
    endpoint: :1888
  zpages:
    endpoint: :55679
  sigv4auth:
    region: my-region
    service: aps
service:
```

```
  extensions: [pprof, zpages, health_check, sigv4auth]
  pipelines:
    metrics:
      receivers: [prometheus]
      exporters: [logging, prometheusremotewrite]
    metrics/ecs:
      receivers: [awsecscontainermetrics]
      processors: [filter]
      exporters: [logging, prometheusremotewrite]
```

**Step 2: Push your ADOT collector container image to an Amazon ECR repository**

Use a Dockerfile to create and push your container image to an Amazon Elastic Container Registry (ECR) repository.

1. Build the Dockerfile to copy and add your container image to the OTEL Docker image.

```
FROM public.ecr.aws/aws-observability/aws-otel-collector:latest
COPY adot-config.yaml /etc/ecs/otel-config.yaml
CMD ["--config=/etc/ecs/otel-config.yaml"]
```

2. Create an Amazon ECR repository.

```
# create repo:
COLLECTOR_REPOSITORY=$(aws ecr create-repository --repository aws-otel-collector \
                              --query repository.repositoryUri --output text)
```

3. Create your container image.

```
# build ADOT collector image:
docker build -t $COLLECTOR_REPOSITORY:ecs .
```

> **Note**
>
> This assumes you are building your container in the same environment that it will run in. If not, you may need to use the `--platform` parameter when building the image.

4. Sign in to the Amazon ECR repository. Replace *`my-region`* with your `region` value.

```
# sign in to repo:
aws ecr get-login-password --region my-region | \
```

```
        docker login --username AWS --password-stdin $COLLECTOR_REPOSITORY
```

5. Push your container image.

```
# push ADOT collector image:
docker push $COLLECTOR_REPOSITORY:ecs
```

### Step 3: Create an Amazon ECS task definition to scrape Amazon Managed Service for Prometheus

Create an Amazon ECS task definition to scrape Amazon Managed Service for Prometheus. Your task definition should include a container named `adot-collector` and a container named `prometheus`. `prometheus` generates metrics, and `adot-collector` scrapes `prometheus`.

> **Note**
>
> Amazon Managed Service for Prometheus runs as a service, collecting metrics from containers. The containers in this case run Prometheus locally, in Agent mode, which send the local metrics to Amazon Managed Service for Prometheus.

#### Example: Task definition

The following is an example of how your task definition might look. You can use this example as a template to create your own task definition. Replace the `image` value of `adot-collector` with your repository URL and image tag (`$COLLECTOR_REPOSITORY:ecs`). Replace the `region` values of `adot-collector` and `prometheus` with your `region` values.

```
{
  "family": "adot-prom",
  "networkMode": "awsvpc",
  "containerDefinitions": [
    {
      "name": "adot-collector",
      "image": "account_id.dkr.ecr.region.amazonaws.com/image-tag",
      "essential": true,
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
```

```
          "awslogs-group": "/ecs/ecs-adot-collector",
          "awslogs-region": "my-region",
          "awslogs-stream-prefix": "ecs",
          "awslogs-create-group": "True"
        }
      }
    },
    {
      "name": "prometheus",
      "image": "prom/prometheus:main",
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/ecs-prom",
          "awslogs-region": "my-region",
          "awslogs-stream-prefix": "ecs",
          "awslogs-create-group": "True"
        }
      }
    }
  ],
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "cpu": "1024"
}
```

**Step 4: Give your task permissions to access Amazon Managed Service for Prometheus**

To send the scraped metrics to Amazon Managed Service for Prometheus, your Amazon ECS task must have the correct permissions to call the AWS API operations for you. You must create an IAM role for your tasks and attach the `AmazonPrometheusRemoteWriteAccess` policy to it. For more information about creating this role and attaching the policy, see Creating an IAM role and policy for your tasks.

After you attach `AmazonPrometheusRemoteWriteAccess` to your IAM role, and use that role for your tasks, Amazon ECS can send your scraped metrics to Amazon Managed Service for Prometheus.

### Step 5: Visualize your metrics in Amazon Managed Grafana

> **Important**
>
> Before you begin, you must run a Fargate task on your Amazon ECS task definition. Otherwise, Amazon Managed Service for Prometheus can't consume your metrics.

1. From the navigation pane in your Amazon Managed Grafana workspace, choose **Data sources** under the AWS icon.
2. On the **Data sources** tab, for **Service**, select **Amazon Managed Service for Prometheus** and choose your **Default Region**.
3. Choose **Add data source**.
4. Use the `ecs` and `prometheus` prefixes to query and view your metrics.

## Set up metrics ingestion from an Amazon EC2 instance using remote write

This section explains how to run a Prometheus server with remote write in an Amazon Elastic Compute Cloud (Amazon EC2) instance. It explains how to collect metrics from a demo application written in Go and send them to an Amazon Managed Service for Prometheus workspace.

### Prerequisites

> **Important**
>
> Before you start, you must have installed Prometheus v2.26 or later. We assume that you're familiar with Prometheus, Amazon EC2, and Amazon Managed Service for Prometheus. For information about how to install Prometheus, see Getting started on the Prometheus website.

If you're unfamiliar with Amazon EC2 or Amazon Managed Service for Prometheus, we recommend that you start by reading the following sections:

- What is Amazon Elastic Compute Cloud?
- What is Amazon Managed Service for Prometheus?

### Create an IAM role for Amazon EC2

To stream metrics, you must first create an IAM role with the AWS managed policy **AmazonPrometheusRemoteWriteAccess**. Then, you can launch an instance with the role and stream metrics into your Amazon Managed Service for Prometheus workspace.

1. Open the IAM console at https://eusc-de-east-1.console.amazonaws-eusc.eu/iam/.
2. From the navigation pane, choose **Roles**, and then choose **Create role**.
3. For the type of trusted entity, choose **AWS service**. For the use case, choose **EC2**. Choose **Next: Permissions**.
4. In the search bar, enter **AmazonPrometheusRemoteWriteAccess**. For **Policy name**, select **AmazonPrometheusRemoteWriteAccess**, and then choose **Attach policy**. Choose **Next:Tags**.
5. (Optional) Create IAM tags for your IAM role. Choose **Next: Review**.
6. Enter a name for your role. Choose **Create policy**.

### Launch an Amazon EC2 instance

To launch an Amazon EC2 instance, follow the instructions at Launch an instance in the *Amazon Elastic Compute Cloud User Guide for Linux Instances*.

### Run the demo application

After creating your IAM role, and launching an EC2 instance with the role, you can run a demo application to see it work.

### To run a demo application and test metrics

1. Use the following template to create a Go file named `main.go`.

```
package main

import (
    "github.com/prometheus/client_golang/prometheus/promhttp"
    "net/http"
)

func main() {
    http.Handle("/metrics", promhttp.Handler())
```

```
    http.ListenAndServe(":8000", nil)
}
```

2. Run the following commands to install the correct dependencies.

```
sudo yum update -y
sudo yum install -y golang
go get github.com/prometheus/client_golang/prometheus/promhttp
```

3. Run the demo application.

```
go run main.go
```

The demo application should run on port 8000 and show all of the exposed Prometheus metrics. The following is an example of these metrics.

```
curl -s http://localhost:8000/metrics
...
process_max_fds 4096# HELP process_open_fds Number of open file descriptors.# TYPE
 process_open_fds gauge
process_open_fds 10# HELP process_resident_memory_bytes Resident memory size in
 bytes.# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 1.0657792e+07# HELP process_start_time_seconds Start
 time of the process since unix epoch in seconds.# TYPE process_start_time_seconds
 gauge
process_start_time_seconds 1.61131955899e+09# HELP process_virtual_memory_bytes
 Virtual memory size in bytes.# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 7.77281536e+08# HELP process_virtual_memory_max_bytes
 Maximum amount of virtual memory available in bytes.# TYPE
 process_virtual_memory_max_bytes gauge
process_virtual_memory_max_bytes -1# HELP
 promhttp_metric_handler_requests_in_flight Current number of scrapes being
 served.# TYPE promhttp_metric_handler_requests_in_flight gauge
promhttp_metric_handler_requests_in_flight 1# HELP
 promhttp_metric_handler_requests_total Total number of scrapes by HTTP status
 code.# TYPE promhttp_metric_handler_requests_total counter
promhttp_metric_handler_requests_total{code="200"} 1
promhttp_metric_handler_requests_total{code="500"} 0
promhttp_metric_handler_requests_total{code="503"} 0
```

### Create an Amazon Managed Service for Prometheus workspace

To create an Amazon Managed Service for Prometheus workspace, follow the instructions at Create a workspace.

### Run a Prometheus server

1. Use the following example YAML file as a template to create a new file named `prometheus.yaml`. For `url`, replace *my-region* with your Region value and *my-workspace-id* with the workspace ID that Amazon Managed Service for Prometheus generated for you. For `region`, replace *my-region* with your Region value.

   **Example: YAML file**

   ```
   global:
     scrape_interval: 15s
     external_labels:
       monitor: 'prometheus'

   scrape_configs:
     - job_name: 'prometheus'
       static_configs:
         - targets: ['localhost:8000']

   remote_write:
     -
       url: https://aps-workspaces.my-region.amazonaws.com/workspaces/my-workspace-id/
   api/v1/remote_write
       queue_config:
           max_samples_per_send: 1000
           max_shards: 200
           capacity: 2500
       sigv4:
            region: my-region
   ```

2. Run the Prometheus server to send the demo application's metrics to your Amazon Managed Service for Prometheus workspace.

   ```
   prometheus --config.file=prometheus.yaml
   ```

The Prometheus server should now send the demo application's metrics to your Amazon Managed Service for Prometheus workspace.

## Using a Prometheus instance as a collector

You can use a Prometheus instance, running in *agent* mode (known as a *Prometheus agent*), to scrape metrics and send them to your Amazon Managed Service for Prometheus workspace.

The following topics describe different ways to set up a Prometheus instance running in agent mode as a collector for your metrics.

> **Warning**
>
> When you create a Prometheus agent, you are responsible for its configuration and maintenance. Avoid exposing Prometheus scrape endpoints to the public internet by enabling security features.

If you set up multiple Prometheus instances that monitor the same set of metrics and sent them to a single Amazon Managed Service for Prometheus workspace for high availability, you need to set up deduplication. If you don't follow the steps to set up deduplication, you will be charged for all data samples sent to Amazon Managed Service for Prometheus, including duplicate samples. For instructions about setting up deduplication, see Deduplicating high availability metrics sent to Amazon Managed Service for Prometheus.

**Topics**

- Set up ingestion from a new Prometheus server using Helm
- Set up ingestion from an existing Prometheus server in Kubernetes on EC2
- Set up ingestion from an existing Prometheus server in Kubernetes on Fargate

### Set up ingestion from a new Prometheus server using Helm

The instructions in this section get you up and running with Amazon Managed Service for Prometheus quickly. You set up a new Prometheus server in an Amazon EKS cluster, and the new server uses a default configuration to send metrics to Amazon Managed Service for Prometheus. This method has the following prerequisites:

- You must have an Amazon EKS cluster from which the new Prometheus server will collect metrics.
- Your Amazon EKS cluster must have an Amazon EBS CSI driver installed (required by Helm).
- You must use Helm CLI 3.0 or later.
- You must use a Linux or macOS computer to perform the steps in the following sections.

### Step 1: Add new Helm chart repositories

To add new Helm chart repositories, enter the following commands. For more information about these commands, see Helm Repo.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo add kube-state-metrics https://kubernetes.github.io/kube-state-metrics
helm repo update
```

### Step 2: Create a Prometheus namespace

Enter the following command to create a Prometheus namespace for the Prometheus server and other monitoring components. Replace *prometheus-namespace* with the name that you want for this namespace.

```
kubectl create namespace prometheus-namespace
```

### Step 3: Set up IAM roles for service accounts

For the method of onboarding that we are documenting, you need to use IAM roles for service accounts in the Amazon EKS cluster where the Prometheus server is running.

With IAM roles for service accounts, you can associate an IAM role with a Kubernetes service account. This service account can then provide AWS permissions to the containers in any pod that uses that service account. For more information, see IAM roles for service accounts.

If you have not already set up these roles, follow the instructions at Set up service roles for the ingestion of metrics from Amazon EKS clusters to set up the roles. The instructions in that section require the use of `eksctl`. For more information, see Getting started with Amazon Elastic Kubernetes Service – `eksctl`.

> **Note**
>
> When you are not on EKS or AWS and using just access key and secret key to access Amazon Managed Service for Prometheus, you cannot use the `EKS-IAM-ROLE` based SigV4.

### Step 4: Set up the new server and start ingesting metrics

To install the new Prometheus server that sends metrics to your Amazon Managed Service for Prometheus workspace, follow these steps.

**To install a new Prometheus server to send metrics to your Amazon Managed Service for Prometheus workspace**

1. Use a text editor to create a file named `my_prometheus_values_yaml` with the following content.

   - Replace *IAM_PROXY_PROMETHEUS_ROLE_ARN* with the ARN of the **amp-iamproxy-ingest-role** that you created in Set up service roles for the ingestion of metrics from Amazon EKS clusters.
   - Replace *WORKSPACE_ID* with the ID of your Amazon Managed Service for Prometheus workspace.
   - Replace *REGION* with the Region of your Amazon Managed Service for Prometheus workspace.

```
## The following is a set of default values for prometheus server helm chart which
 enable remoteWrite to AMP
## For the rest of prometheus helm chart values see: https://github.com/prometheus-
community/helm-charts/blob/main/charts/prometheus/values.yaml
##
serviceAccounts:
  server:
    name: amp-iamproxy-ingest-service-account
    annotations:
      eks.amazonaws.com/role-arn: ${IAM_PROXY_PROMETHEUS_ROLE_ARN}
server:
  remoteWrite:
    - url: https://aps-workspaces.${REGION}.amazonaws.com/workspaces/
${WORKSPACE_ID}/api/v1/remote_write
```

```
      sigv4:
        region: ${REGION}
      queue_config:
        max_samples_per_send: 1000
        max_shards: 200
        capacity: 2500
```

2. Enter the following command to create the Prometheus server.

   - Replace *prometheus-chart-name* with your Prometheus release name.
   - Replace *prometheus-namespace* with the name of your Prometheus namespace.

```
helm install prometheus-chart-name prometheus-community/prometheus -n prometheus-
namespace \
-f my_prometheus_values_yaml
```

> **Note**
>
> You can customize the `helm install` command in many ways. For more information, see Helm install in the *Helm documentation*.

## Set up ingestion from an existing Prometheus server in Kubernetes on EC2

Amazon Managed Service for Prometheus supports ingesting metrics from Prometheus servers in clusters running Amazon EKS and in self-managed Kubernetes clusters running on Amazon EC2. The detailed instructions in this section are for a Prometheus server in an Amazon EKS cluster. The steps for a self-managed Kubernetes cluster on Amazon EC2 are the same, except that you will need to set up the OIDC provider and IAM roles for service accounts yourself in the Kubernetes cluster.

The instructions in this section use Helm as the Kubernetes package manager.

### Topics


- Step 1: Set up IAM roles for service accounts
- Step 2: Upgrade your existing Prometheus server using Helm

### Step 1: Set up IAM roles for service accounts

For the method of onboarding that we are documenting, you need to use IAM roles for service accounts in the Amazon EKS cluster where the Prometheus server is running. These roles are also called *service roles*.

With service roles, you can associate an IAM role with a Kubernetes service account. This service account can then provide AWS permissions to the containers in any pod that uses that service account. For more information, see IAM roles for service accounts.

If you have not already set up these roles, follow the instructions at Set up service roles for the ingestion of metrics from Amazon EKS clusters to set up the roles.

### Step 2: Upgrade your existing Prometheus server using Helm

The instructions in this section include setting up remote write and sigv4 to authenticate and authorize the Prometheus server to remote write to your Amazon Managed Service for Prometheus workspace.

#### Using Prometheus version 2.26.0 or later

Follow these steps if you are using a Helm chart with Prometheus Server image of version 2.26.0 or later.

#### To set up remote write from a Prometheus server using Helm chart

1. Create a new remote write section in your Helm configuration file:

   - Replace `${IAM_PROXY_PROMETHEUS_ROLE_ARN}` with the ARN of the **amp-iamproxy-ingest-role** that you created in Step 1: Set up IAM roles for service accounts. The role ARN should have the format of `arn:aws:iam::`*`your account ID`*`:role/amp-iamproxy-ingest-role`.
   - Replace `${WORKSPACE_ID}` with your Amazon Managed Service for Prometheus workspace ID.
   - Replace `${REGION}` with the Region of the Amazon Managed Service for Prometheus workspace (such as `us-west-2`).

```
## The following is a set of default values for prometheus server helm chart which
 enable remoteWrite to AMP
```

```
    ## For the rest of prometheus helm chart values see: https://github.com/
prometheus-community/helm-charts/blob/main/charts/prometheus/values.yaml
    ##
    serviceAccounts:
      server:
        name: amp-iamproxy-ingest-service-account
        annotations:
          eks.amazonaws.com/role-arn: ${IAM_PROXY_PROMETHEUS_ROLE_ARN}
    server:
      remoteWrite:
        - url: https://aps-workspaces.${REGION}.amazonaws.com/workspaces/
${WORKSPACE_ID}/api/v1/remote_write
          sigv4:
            region: ${REGION}
          queue_config:
            max_samples_per_send: 1000
            max_shards: 200
            capacity: 2500
```

2. Update your existing Prometheus Server configuration using Helm:

   - Replace `prometheus-chart-name` with your Prometheus release name.
   - Replace `prometheus-namespace` with the Kubernetes namespace where your Prometheus Server is installed.
   - Replace `my_prometheus_values_yaml` with the path to your Helm configuration file.
   - Replace `current_helm_chart_version` with the current version of your Prometheus Server Helm chart. You can find the current chart version by using the helm list command.

```
helm upgrade prometheus-chart-name prometheus-community/prometheus \
       -n prometheus-namespace \
       -f my_prometheus_values_yaml \
       --version current_helm_chart_version
```

### Using earlier versions of Prometheus

Follow these steps if you are using a version of Prometheus earlier than 2.26.0. These steps use a sidecar approach, because earlier versions of Prometheus don't natively support AWS Signature Version 4 signing process (AWS SigV4).

These instructions assume that you are using Helm to deploy Prometheus.

**To set up remote write from a Prometheus server**

1. On your Prometheus server, create a new remote write configuration. First, create a new update file. We will call the file `amp_ingest_override_values.yaml`.

   Add the following values to the YAML file.

```
serviceAccounts:
        server:
            name: "amp-iamproxy-ingest-service-account"
            annotations:
                eks.amazonaws.com/role-arn:
 "${SERVICE_ACCOUNT_IAM_INGEST_ROLE_ARN}"
    server:
        sidecarContainers:
            - name: aws-sigv4-proxy-sidecar
              image: public.ecr.aws/aws-observability/aws-sigv4-proxy:1.0
              args:
              - --name
              - aps
              - --region
              - ${REGION}
              - --host
              - aps-workspaces.${REGION}.amazonaws.com
              - --port
              - :8005
              ports:
              - name: aws-sigv4-proxy
                containerPort: 8005
        statefulSet:
            enabled: "true"
        remoteWrite:
            - url: http://localhost:8005/workspaces/${WORKSPACE_ID}/api/v1/
remote_write
```

   Replace `${REGION}` with the Region of the Amazon Managed Service for Prometheus workspace.

   Replace `${SERVICE_ACCOUNT_IAM_INGEST_ROLE_ARN}` with the ARN of the **amp-iamproxy-ingest-role** that you created in Step 1: Set up IAM roles for service accounts. The role ARN should have the format of `arn:aws:iam::`*`your account ID`*`:role/amp-iamproxy-ingest-role`.

Replace ${WORKSPACE_ID} with your workspace ID.

2. Upgrade your Prometheus Helm chart. First, find your Helm chart name by entering the following command. In the output from this command, look for a chart with a name that includes prometheus.

```
helm ls --all-namespaces
```

Then enter the following command.

```
helm upgrade --install prometheus-helm-chart-name prometheus-community/prometheus -
n prometheus-namespace -f ./amp_ingest_override_values.yaml
```

Replace *prometheus-helm-chart-name* with the name of the Prometheus helm chart returned in the previous command. Replace *prometheus-namespace* with the name of your namespace.

#### Downloading Helm charts

If you don't already have Helm charts downloaded locally, you can use the following command to download them.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
    helm pull prometheus-community/prometheus --untar
```

## Set up ingestion from an existing Prometheus server in Kubernetes on Fargate

Amazon Managed Service for Prometheus supports ingesting metrics from Prometheus servers in self-managed Kubernetes clusters running on Fargate. To ingest metrics from Prometheus servers in Amazon EKS clusters running on Fargate, override the default configs in a config file named amp_ingest_override_values.yaml as follows:

```
prometheus-node-exporter:
  enabled: false

alertmanager:
  enabled: false
```

```
serviceAccounts:
  server:
    name: amp-iamproxy-ingest-service-account
    annotations:
      eks.amazonaws.com/role-arn: ${IAM_PROXY_PROMETHEUS_ROLE_ARN}

server:
  persistentVolume:
    enabled: false
  remoteWrite:
    - url: https://aps-workspaces.${REGION}.amazonaws.com/workspaces/${WORKSPACE_ID}/
api/v1/remote_write
      sigv4:
        region: ${REGION}
      queue_config:
        max_samples_per_send: 1000
        max_shards: 200
        capacity: 2500
```

Install Prometheus using the overrides with the following command:

```
helm install prometheus-for-amp prometheus-community/prometheus \
                  -n prometheus \
                  -f amp_ingest_override_values.yaml
```

Note that in the Helm chart configuration we disabled the node exporter and the alert manager as well as running the Prometheus server deployment.

You can verify the install with the following example test query.

```
$ awscurl --region region --service aps "https://aps-
workspaces.region_id.amazonaws.com/workspaces/workspace_id/api/v1/query?
query=prometheus_api_remote_read_queries"
Output:
    {"status":"success","data":{"resultType":"vector","result":[{"metric":
{"__name__":"prometheus_api_remote_read_queries","instance":"localhost:9090","job":"prometheus"
[1648461236.419,"0"]}]}}21
```

# Set up Amazon Managed Service for Prometheus for high availability data

When you send data to Amazon Managed Service for Prometheus, it is automatically replicated across AWS Availability Zones in the Region, and is served to you from a cluster of hosts that provide scalability, availability, and security. You might want to add additional high availability fail-safes, depending on your particular setup. There are two common ways that you might additional high availability safeties to your setup:

- If you have multiple containers or instances that have the same data, you can send that data to Amazon Managed Service for Prometheus and have the data automatically de-duplicated. This helps to ensure that your data will be sent to your Amazon Managed Service for Prometheus workspace.

  For more information about de-duplicating high-availability data, see Deduplicating high availability metrics sent to Amazon Managed Service for Prometheus.
- If you want to ensure that you have access to your data, even when the AWS Region is not available, you can send your metrics to a second workspace, in another Region.

  For more information about sending metrics data to multiple workspaces, see Use cross Region workspaces to add high availability in Amazon Managed Service for Prometheus.

**Topics**


- Deduplicating high availability metrics sent to Amazon Managed Service for Prometheus
- Send high availability data to Amazon Managed Service for Prometheus with Prometheus
- Set up high availability data to Amazon Managed Service for Prometheus using the Prometheus Operator Helm chart
- Send high-availability data to Amazon Managed Service for Prometheus with AWS Distro for OpenTelemetry
- Send high availability data to Amazon Managed Service for Prometheus with the Prometheus community Helm chart
- Answers to common questions about high availability configuration in Amazon Managed Service for Prometheus
- Use cross Region workspaces to add high availability in Amazon Managed Service for Prometheus

## Deduplicating high availability metrics sent to Amazon Managed Service for Prometheus

You can send data from multiple Prometheus *agents* (Prometheus instances running in Agent mode) to your Amazon Managed Service for Prometheus workspace. If some of these instances are recording and sending the same metrics, your data will have a higher availability (even if one of the agents stops sending data, the Amazon Managed Service for Prometheus workspace will still receive the data from another instance). However, you want your Amazon Managed Service for Prometheus workspace to automatically de-duplicate the metrics so that you don't see the metrics multiple times, and aren't charged for the data ingestion and storage multiple times.

For Amazon Managed Service for Prometheus to automatically de-duplicate data from multiple Prometheus agents, you give the set of agents that are sending the duplicate data a single *cluster name*, and each of the instances a *replica name*. The cluster name identifies the instances as having shared data, and the replica name allows Amazon Managed Service for Prometheus to identify the source of each metric. The final stored metrics include the cluster label, but not the replica, so the metrics appear to be coming from a single source.

> **Note**
>
> Certain versions of Kubernetes (1.28 and 1.29) may emit their own metric with a `cluster` label. This can cause issues with Amazon Managed Service for Prometheus deduplication. See the High availability FAQ for more information.

The following topics show how to send data and include the `cluster` and `__replica__` labels, so that Amazon Managed Service for Prometheus de-duplicates the data automatically.

> **Important**
>
> If you do not set up deduplication, you will be charged for all data samples that are sent to Amazon Managed Service for Prometheus. These data samples include duplicate samples.

## Send high availability data to Amazon Managed Service for Prometheus with Prometheus

To set up a high availability configuration with Prometheus, you must apply external labels on all instances of a high availability group, so Amazon Managed Service for Prometheus can identify them. Use the `cluster` label to identify a Prometheus instance agent as part of a high availability group. Use the `__replica__` label to identify each replica in the group separately. You need to apply both `__replica__` and `cluster` labels for de-duplication to work.

> **Note**
>
> The `__replica__` label is formatted with two underscore symbols before and after the word `replica`.

### Example: code snippets

In the following code snippets, the `cluster` label identifies the Prometheus instance agent `prom-team1`, and the `_replica_` label identifies the replicas `replica1` and `replica2`.

```
cluster: prom-team1
__replica__: replica1
```

```
cluster: prom-team1
__replica__: replica2
```

As Amazon Managed Service for Prometheus stores data samples from high availability replicas with these labels, it strips the `replica` label when the samples are accepted. This means that you will only have a 1:1 series mapping for your current series instead of a series per replica. The `cluster` label is kept.

> **Note**
>
> Certain versions of Kubernetes (1.28 and 1.29) may emit their own metric with a `cluster` label. This can cause issues with Amazon Managed Service for Prometheus deduplication. See the High availability FAQ for more information.

## Set up high availability data to Amazon Managed Service for Prometheus using the Prometheus Operator Helm chart

To set up a high availability configuration with the Prometheus Operator in Helm, you must apply external labels on all instances of a high availability group, so Amazon Managed Service for Prometheus can identify them. You also must set the attributes `replicaExternalLabelName` and `externalLabels` on the Prometheus Operator Helm chart.

**Example: YAML header**

In the following YAML header, `cluster` is added to `externalLabel` to identify a Prometheus instance agent as part of a high-availability group, and `replicaExternalLabels` identifies each replica in the group.

```
replicaExternalLabelName: __replica__
externalLabels:
cluster: prom-dev
```

> **Note**
>
> Certain versions of Kubernetes (1.28 and 1.29) may emit their own metric with a `cluster` label. This can cause issues with Amazon Managed Service for Prometheus deduplication. See the High availability FAQ for more information.

## Send high-availability data to Amazon Managed Service for Prometheus with AWS Distro for OpenTelemetry

AWS Distro for OpenTelemetry (ADOT) is a secure and production-ready distribution of the OpenTelemetry project. ADOT provides you with source APIs, libraries, and agents, so you can collect distributed traces and metrics for application monitoring. For information about ADOT, see About AWS Distro for Open Telemetry.

To set up ADOT with a high availability configuration, you must configure an ADOT collector container image and apply the external labels `cluster` and `__replica__` to the AWS Prometheus remote write exporter. This exporter sends your scraped metrics to your Amazon Managed Service for Prometheus workspace via the `remote_write` endpoint. When you set these labels on the remote write exporter, you prevent duplicate metrics from being kept while redundant replicas run. For more information about the AWS Prometheus remote write exporter,

see Getting started with Prometheus remote write exporter for Amazon Managed Service for Prometheus.

> **Note**
>
> Certain versions of Kubernetes (1.28 and 1.29) may emit their own metric with a `cluster` label. This can cause issues with Amazon Managed Service for Prometheus deduplication. See the High availability FAQ for more information.

## Send high availability data to Amazon Managed Service for Prometheus with the Prometheus community Helm chart

To set up a high availability configuration with the Prometheus community Helm chart, you must apply external labels on all instances of a high availability group, so Amazon Managed Service for Prometheus can identify them. Here is an example of how you could add the `external_labels` to a single instance of Prometheus from the Prometheus community Helm chart.

```
server:
global:
  external_labels:
      cluster: monitoring-cluster
      __replica__: replica-1
```

> **Note**
>
> If you want multiple replicas, you have to deploy the chart multiple times with different replica values, because the Prometheus community Helm chart does not let you dynamically set the replica value when increasing the number of replicas directly from the controller group. If you prefer to have the `replica` label auto-set, use the prometheus-operator Helm chart.

> **Note**
>
> Certain versions of Kubernetes (1.28 and 1.29) may emit their own metric with a `cluster` label. This can cause issues with Amazon Managed Service for Prometheus deduplication. See the High availability FAQ for more information.

## Answers to common questions about high availability configuration in Amazon Managed Service for Prometheus

### Should I include the value *__replica__* into another label to track the sample points?

In a high availability setting, Amazon Managed Service for Prometheus ensures data samples are not duplicated by electing a leader in the cluster of Prometheus instances. If the leader replica stops sending data samples for 30 seconds, Amazon Managed Service for Prometheus automatically makes another Prometheus instance a leader replica and ingests data from the new leader, including any missed data. Therefore, the answer is no, it is not recommended.  Doing so may cause issues like:

- Querying a `count` in **PromQL** may return higher than expected value during the period of electing a new leader.
- The number of `active series` gets increased during a period of electing a new leader and it reaches the `active series limits`. See AMP Quotas for more info.

### Kubernetes seems to have it's own *cluster* label, and is not deduplicating my metrics. How can I fix this?

A new metric, `apiserver_storage_size_bytes` was introduced in Kubernetes 1.28, with a `cluster` label. This can cause issues with deduplication in Amazon Managed Service for Prometheus, which depends on the `cluster` label. In Kubernetes 1.3, the label is renamed to `storage-cluster_id` (it is also renamed in later patches of 1.28 and 1.29). If your cluster is emitting this metric with the `cluster` label, Amazon Managed Service for Prometheus can't dedupe the associated time series. We recommend you upgrade your Kubernetes cluster to the latest patched version to avoid this problem. Alternately, you can relabel the `cluster` label on your `apiserver_storage_size_bytes` metric before ingesting it into Amazon Managed Service for Prometheus.

> **Note**
>
> For more details about the change to Kubernetes, see Rename Label cluster to storage_cluster_id for apiserver_storage_size_bytes metric in the *Kubernetes GitHub project*.

## Use cross Region workspaces to add high availability in Amazon Managed Service for Prometheus

To add cross-Region availability to your data, you can send metrics to multiple workspaces across AWS Regions. Prometheus supports both multiple writers and cross-Region writing.

The following example shows how to set up a Prometheus server running in Agent mode to send metrics to two workspaces in different Regions with Helm.

```
extensions:
      sigv4auth:
        service: "aps"

    receivers:
      prometheus:
        config:
          scrape_configs:
            - job_name: 'kubernetes-kubelet'
              scheme: https
              tls_config:
                ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
                insecure_skip_verify: true
              bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
              kubernetes_sd_configs:
              - role: node
              relabel_configs:
              - action: labelmap
                regex: __meta_kubernetes_node_label_(.+)
              - target_label: __address__
                replacement: kubernetes.default.svc.cluster.local:443
              - source_labels: [__meta_kubernetes_node_name]
                regex: (.+)
                target_label: __metrics_path__
                replacement: /api/v1/nodes/$${1}/proxy/metrics

    exporters:
      prometheusremotewrite/one:
        endpoint: "https://aps-workspaces.workspace_1_region.amazonaws.com/workspaces/
ws-workspace_1_id/api/v1/remote_write"
        auth:
          authenticator: sigv4auth
      prometheusremotewrite/two:
```

```
        endpoint: "https://aps-workspaces.workspace_2_region.amazonaws.com/workspaces/
ws-workspace_2_id/api/v1/remote_write"
        auth:
          authenticator: sigv4auth

    service:
      extensions: [sigv4auth]
      pipelines:
        metrics/one:
          receivers: [prometheus]
          exporters: [prometheusremotewrite/one]
        metrics/two:
          receivers: [prometheus]
          exporters: [prometheusremotewrite/two]
```

# Query your Prometheus metrics

Now that metrics are being ingested to the workspace, you can query them.

To create dashboards with visual representations of your metrics, you can use a service such as Amazon Managed Grafana. Amazon Managed Grafana (or a standalone instance of Grafana) can build a graphical interface that shows your metrics in a wide variety of display presentation styles. For more information about Amazon Managed Grafana see the Amazon Managed Grafana User Guide.

You can also create one-off queries, explore your data, or write your own applications that use your metrics by using direct queries. Direct queries use the Amazon Managed Service for Prometheus API and the standard Prometheus query language, PromQL, to get data from your Prometheus workspace. For more information about PromQL and its syntax, see Querying Prometheus in the Prometheus documentation.

**Topics**


- PromQL cheat sheet
- Basic selectors
- Range vector selectors
- Aggregation operators
- Common functions
- Binary operators
- Practical query examples
- Secure your metric queries
- Set up Amazon Managed Grafana for use with Amazon Managed Service for Prometheus
- Set up Grafana open source or Grafana Enterprise for use with Amazon Managed Service for Prometheus
- Query using Grafana running in an Amazon EKS cluster
- Query using Prometheus-compatible APIs
- Get statistics about your query usage for each query

# PromQL cheat sheet

Use this PromQL (Prometheus Query Language) cheat sheet as a quick reference when querying metrics in your Amazon Managed Service for Prometheus workspace. With PromQL, you can select and aggregate time series data in real time through its functional query language.

For more details about PromQL, see PromQL Cheat Sheet on the *PromLabs* website.

## Basic selectors

Select time series by metric name and label matchers:

```
# Select all time series with the metric name http_requests_total
http_requests_total

# Select time series with specific label values
http_requests_total{job="prometheus", method="GET"}

# Use label matchers
http_requests_total{status_code!="200"}          # Not equal
http_requests_total{status_code=~"2.."}          # Regex match
http_requests_total{status_code!~"4.."}          # Negative regex match
```

## Range vector selectors

Select a range of samples over time:

```
# Select 5 minutes of data
http_requests_total[5m]

# Time units: s (seconds), m (minutes), h (hours), d (days), w (weeks), y (years)
cpu_usage[1h]
memory_usage[30s]
```

# Aggregation operators

Aggregate data across multiple time series:

```
# Sum all values
sum(http_requests_total)

# Sum by specific labels
sum by (job) (http_requests_total)
sum without (instance) (http_requests_total)

# Other aggregation operators
avg(cpu_usage)                    # Average
min(response_time)                # Minimum
max(response_time)                # Maximum
count(up)                         # Count of series
stddev(cpu_usage)                 # Standard deviation
```

# Common functions

Apply functions to transform your data:

```
# Rate of increase per second (for counters)
rate(http_requests_total[5m])

# Increase over time range
increase(http_requests_total[1h])

# Derivative (for gauges)
deriv(cpu_temperature[5m])

# Mathematical functions
abs(cpu_usage - 50)               # Absolute value
round(cpu_usage, 0.1)             # Round to nearest 0.1
sqrt(memory_usage)                # Square root

# Time functions
time()                            # Current Unix timestamp
```

```
hour()                          # Hour of day (0-23)
day_of_week()                   # Day of week (0-6, Sunday=0)
```

## Binary operators

Perform arithmetic and logical operations:

```
# Arithmetic operators
cpu_usage + 10
memory_total - memory_available
disk_usage / disk_total * 100

# Comparison operators (return 0 or 1)
cpu_usage > 80
memory_usage < 1000
response_time >= 0.5

# Logical operators
(cpu_usage > 80) and (memory_usage > 1000)
(status_code == 200) or (status_code == 201)
```

## Practical query examples

Common monitoring queries you can use in your Amazon Managed Service for Prometheus workspace:

```
# CPU usage percentage
100 - (avg by (instance) (rate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)

# Memory usage percentage
(1 - (node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes)) * 100

# Request rate per second
sum(rate(http_requests_total[5m])) by (job)

# Error rate percentage
```

```
sum(rate(http_requests_total{status_code=~"5.."}[5m])) /
sum(rate(http_requests_total[5m])) * 100

# 95th percentile response time
histogram_quantile(0.95, sum(rate(http_request_duration_seconds_bucket[5m])) by (le))

# Top 5 instances by CPU usage
topk(5, avg by (instance) (cpu_usage))
```

# Secure your metric queries

Amazon Managed Service for Prometheus provides ways of helping you secure the querying of your metrics.

## Using AWS PrivateLink with Amazon Managed Service for Prometheus

The network traffic for querying metrics in Amazon Managed Service for Prometheus can be done over a public internet endpoint, or by a VPC endpoint through AWS PrivateLink. When you use AWS PrivateLink, network traffic from your VPCs is secured within the AWS network without going over the public internet. To create an AWS PrivateLink VPC endpoint for Amazon Managed Service for Prometheus, see Using Amazon Managed Service for Prometheus with interface VPC endpoints.

## Authentication and authorization

AWS Identity and Access Management is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources. Amazon Managed Service for Prometheus integrates with IAM to help you keep your data secure. When you set up Amazon Managed Service for Prometheus, you'll need to create some IAM roles that enable Grafana servers to query metrics stored in Amazon Managed Service for Prometheus workspaces. For more information about IAM, see What is IAM?.

Another AWS security feature that can help you set up Amazon Managed Service for Prometheus is the AWS Signature Version 4 signing process (AWS SigV4). Signature Version 4 is the process to add authentication information to AWS requests sent by HTTP. For security, most requests to AWS must be signed with an access key, which consists of an access key ID and secret access key. These two keys are commonly referred to as your security credentials. For more information about SigV4, see Signature Version 4 signing process.

# Set up Amazon Managed Grafana for use with Amazon Managed Service for Prometheus

Amazon Managed Grafana is a fully managed service for open-source Grafana that simplifies connecting to open-source, third-party ISV, and AWS services for visualizing and analyzing your data sources at scale.

Amazon Managed Service for Prometheus supports using Amazon Managed Grafana to query metrics in a workspace. In the Amazon Managed Grafana console, you can add an Amazon Managed Service for Prometheus workspace as a data source by discovering your existing Amazon Managed Service for Prometheus accounts. Amazon Managed Grafana manages the configuration of the authentication credentials that are required to access Amazon Managed Service for Prometheus. For detailed instructions on creating a connection to Amazon Managed Service for Prometheus from Amazon Managed Grafana, see the instructions in the Amazon Managed Grafana User Guide.

You may also view your Amazon Managed Service for Prometheus alerts in Amazon Managed Grafana. For instructions to set up integration with alerts, see Integrate alerts with Amazon Managed Grafana or open source Grafana.

## Connecting to Amazon Managed Grafana in a private VPC

Amazon Managed Service for Prometheus provides a service endpoint for Amazon Managed Grafana to connect to when querying metrics and alerts.

You can configure Amazon Managed Grafana to use a private VPC (for details on setting up a private VPC in Grafana, see Connecting to Amazon VPC in the *Amazon Managed Grafana User Guide*). Depending on the settings, this VPC may not have access to the Amazon Managed Service for Prometheus service endpoint.

To add Amazon Managed Service for Prometheus as a data source to an Amazon Managed Grafana workspace that is configured to use a specific private VPC, you must first connect your Amazon Managed Service for Prometheus to the same VPC by creating a VPC endpoint. For more information about creating a VPC endpoint, see Create an interface VPC endpoint for Amazon Managed Service for Prometheus.

# Set up Grafana open source or Grafana Enterprise for use with Amazon Managed Service for Prometheus

You can use an instance of Grafana to query your metrics in Amazon Managed Service for Prometheus. This topic takes you through how to query metrics from Amazon Managed Service for Prometheus using a standalone instance of Grafana.

## Prerequisites

**Grafana instance** – You must have a Grafana instance that is capable of authenticating with Amazon Managed Service for Prometheus.

Amazon Managed Service for Prometheus supports the use of Grafana version 7.3.5 and later to query metrics in a workspace. Versions 7.3.5 and later include support for AWS Signature Version 4 (SigV4) authentication.

To check your Grafana version, enter the following command, replacing *grafana_install_directory* with the path to your Grafana installation:

```
grafana_install_directory/bin/grafana-server -v
```

If you do not already have a standalone Grafana, or need a newer version, you can install a new instance. For instructions to set up a standalone Grafana, see Install Grafana in the Grafana documentation. For information about getting started with Grafana, see Getting started with Grafana in the Grafana documentation.

**AWS account** – You must have an AWS account with the correct permissions to access your Amazon Managed Service for Prometheus metrics.

To set up Grafana to work with Amazon Managed Service for Prometheus, you must be logged on to an account that has the **AmazonPrometheusQueryAccess** policy or the `aps:QueryMetrics`, `aps:GetMetricMetadata`, `aps:GetSeries`, and `aps:GetLabels`permissions. For more information, see IAM permissions and policies.

The next section describes setting up authentication from Grafana in more detail.

## Step 1: Set up AWS SigV4

Amazon Managed Service for Prometheus works with AWS Identity and Access Management (IAM) to secure all calls to Prometheus APIs with IAM credentials. By default, the Prometheus data

source in Grafana assumes that Prometheus requires no authentication. To enable Grafana to take advantage of Amazon Managed Service for Prometheus authentication and authorization capabilities, you will need to enable SigV4 authentication support in the Grafana data source. Follow the steps on this page when you are using a self-managed Grafana open-source or a Grafana enterprise server. If you are using Amazon Managed Grafana, SIGv4 authentication is fully automated. For more information about Amazon Managed Grafana, see What is Amazon Managed Grafana?

To enable SigV4 on Grafana, start Grafana with the `AWS_SDK_LOAD_CONFIG` and `GF_AUTH_SIGV4_AUTH_ENABLED` environment variables set to `true`. The `GF_AUTH_SIGV4_AUTH_ENABLED` environment variable overrides the default configuration for Grafana to enable SigV4 support. For more information, see Configuration in the Grafana documentation.

**Linux**

To enable SigV4 on a standalone Grafana server on Linux, enter the following commands.

```
export AWS_SDK_LOAD_CONFIG=true
```

```
export GF_AUTH_SIGV4_AUTH_ENABLED=true
```

```
cd grafana_install_directory
```

```
./bin/grafana-server
```

**Windows**

To enable SigV4 on a standalone Grafana on Windows using the Windows command prompt, enter the following commands.

```
set AWS_SDK_LOAD_CONFIG=true
```

```
set GF_AUTH_SIGV4_AUTH_ENABLED=true
```

```
cd grafana_install_directory
```

```
.\bin\grafana-server.exe
```

## Step 2: Add the Prometheus data source in Grafana

The following steps explain how to set up the Prometheus data source in Grafana to query your Amazon Managed Service for Prometheus metrics.

**To add the Prometheus data source in your Grafana server**

1. Open the Grafana console.
2. Under **Configurations**, choose **Data sources**.
3. Choose **Add data source**.
4. Choose **Prometheus**.
5. For the HTTP URL, specify the **Endpoint - query URL** displayed in the workspace details page in the Amazon Managed Service for Prometheus console.
6. In the HTTP URL that you just specified, remove the `/api/v1/query` string that is appended to the URL, because the Prometheus data source will automatically append it.

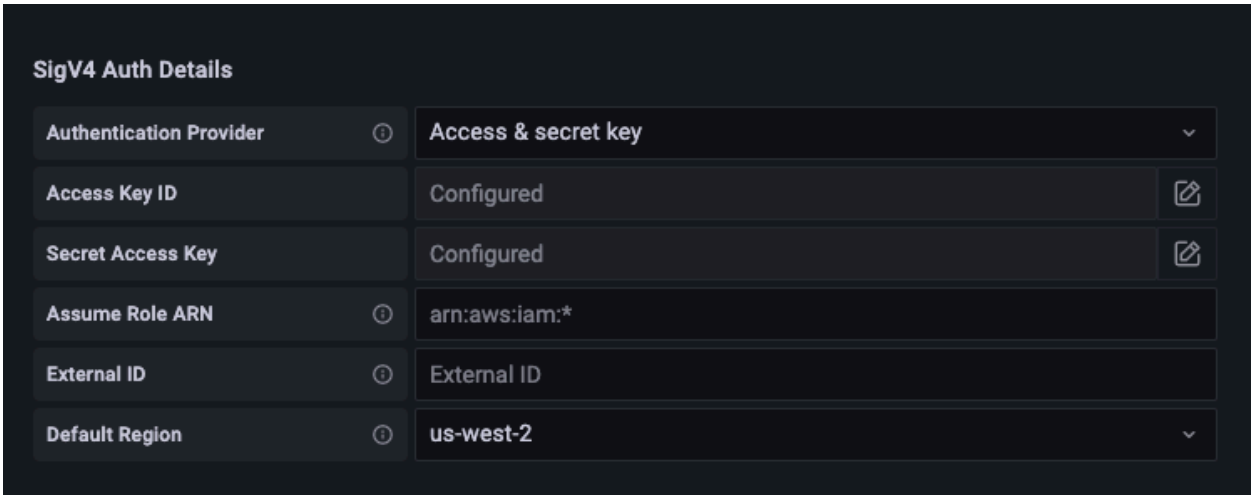
   The correct URL should look similar to **https://aps-workspaces.us-west-2.amazonaws.com/workspaces/ws-1234a5b6-78cd-901e-2fgh-3i45j6k178l9**.
7. Under **Auth**, select the toggle for **SigV4 Auth** to enable it.
8. You can either configure SigV4 authorization by specifying your long-term credentials directly in Grafana, or by using a default provider chain. Specifying your long-term credentials directly gets you started quicker, and the following steps give those instructions first. Once you are more familiar with using Grafana with Amazon Managed Service for Prometheus, we recommend that you use a default provider chain, because it provides better flexibility and security. For more information about setting up your default provider chain, see Specifying Credentials.
   - To use your long-term credentials directly, do the following:
     a. Under **SigV4 Auth Details**, for **Authentication Provider** choose **Access & secret key**.
     b. For **Access Key ID**, enter your AWS access key ID.
     c. For **Secret Access Key**, enter your AWS secret access key.
     d. Leave the **Assume Role ARN** and **External ID** fields blank.

e. For **Default Region**, choose the Region of your Amazon Managed Service for Prometheus workspace. This Region should match the Region contained in the URL that you listed in step 5.

f. Choose **Save & Test**.

You should see the following message: **Data source is working**

The following screenshot shows the Access key, Secret key SigV4 auth detail setting.




- To use a default provider chain instead (recommended for a production environment), do the following:

a. Under **SigV4 Auth Details**, for **Authentication Provider** choose **AWS SDK Default**.

b. Leave the **Assume Role ARN** and **External ID** fields blank.

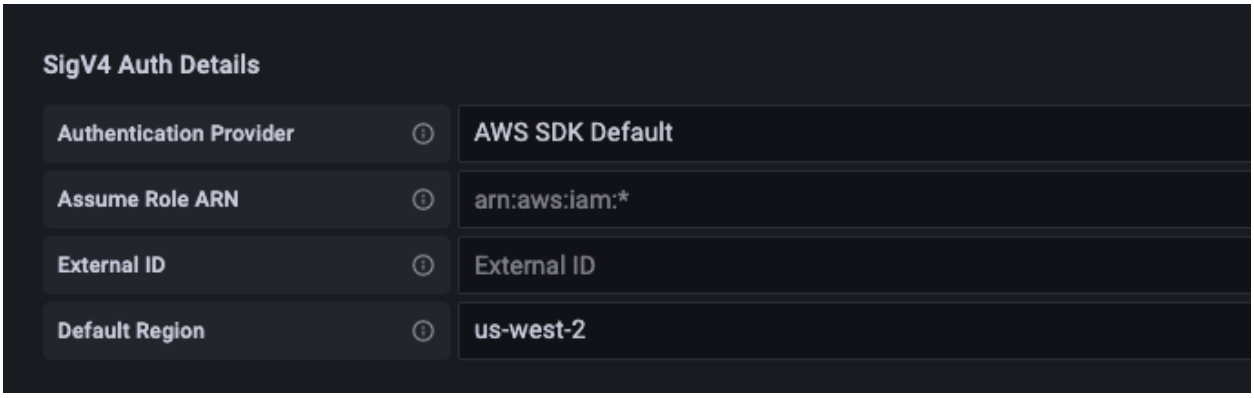
c. For **Default Region**, choose the Region of your Amazon Managed Service for Prometheus workspace. This Region should match the Region contained in the URL that you listed in step 5.

d. Choose **Save & Test**.

You should see the following message: **Data source is working**

If you do not see that message, the next section provides troubleshooting tips for connecting.

The following screenshot shows the SDK default SigV4 auth detail setting.




9. Test a PromQL query against the new data source:

   a. Choose **Explore**.

   b. Run a sample PromQL query such as:

   ```
   prometheus_tsdb_head_series
   ```

## Step 3: (optional) Troubleshooting if Save & Test doesn't work

In the previous procedure, if you see an error when you choose **Save & Test**, check the following.

### HTTP Error Not Found

Make sure that the workspace ID in the URL is correct.

### HTTP Error Forbidden

This error means that the credentials are not valid. Check the following:

- Check that the Region specified in **Default Region** is correct.
- Check your credential for typos.
- Make sure that the credential that you are using has the **AmazonPrometheusQueryAccess** policy. For more information, see IAM permissions and policies.
- Make sure that the credential that you are using has access to this Amazon Managed Service for Prometheus workspace.

### HTTP Error Bad Gateway

Look at the Grafana server log to troubleshoot this error. For more information, see Troubleshooting in the Grafana documentation.

If you see **`Error http: proxy error: NoCredentialProviders: no valid providers in chain`**, the default credential provider chain was not able to find a valid AWS credential to use. Make sure you have set up your credentials as documented in Specifying Credentials. If you want to use a shared configuration, make sure that the `AWS_SDK_LOAD_CONFIG` environment is set to `true`.

# Query using Grafana running in an Amazon EKS cluster

Amazon Managed Service for Prometheus supports the use of Grafana version 7.3.5 and later to query metrics in a Amazon Managed Service for Prometheus workspace. Versions 7.3.5 and later include support for AWS Signature Version 4 (SigV4) authentication.

To set up Grafana to work with Amazon Managed Service for Prometheus, you must be logged on to an account that has the **AmazonPrometheusQueryAccess** policy or the `aps:QueryMetrics`, `aps:GetMetricMetadata`, `aps:GetSeries`, and `aps:GetLabels` permissions. For more information, see IAM permissions and policies.

## Set up AWS SigV4

Grafana has added a new feature to support AWS Signature Version 4 (SigV4) authentication. For more information, see Signature Version 4 signing process. This feature is not enabled by default on Grafana servers. The following instructions for enabling this feature assume that you are using Helm to deploy Grafana on a Kubernetes cluster.

**To enable SigV4 on your Grafana 7.3.5 or later server**

1. Create a new update file to override your Grafana configuration, and name it `amp_query_override_values.yaml`.
2. Enter the following content into the file, and save the file. Replace *`account-id`* with the AWS account ID where the Grafana server is running.

```
serviceAccount:
    name: "amp-iamproxy-query-service-account"
    annotations:
        eks.amazonaws.com/role-arn: "arn:aws:iam::account-id:role/amp-iamproxy-
query-role"
```

```
grafana.ini:
  auth:
    sigv4_auth_enabled: true
```

In that YAML file content, `amp-iamproxy-query-role` is the name of the role that you will create in the next section, Set up IAM roles for service accounts. You can replace this role with your own role name if you already have a role created for querying your workspace.

You will use this file later, in Upgrade the Grafana server using Helm.

## Set up IAM roles for service accounts

If you are using a Grafana server in an Amazon EKS cluster, we recommend that you use IAM roles for service accounts, also known as service roles, for your access control. When you do this to associate an IAM role with a Kubernetes service account, the service account can then provide AWS permissions to the containers in any pod that uses that service account. For more information, see IAM roles for service accounts.

If you have not already set up these service roles for querying, follow the instructions at Set up IAM roles for service accounts for the querying of metrics to set up the roles.

You then need to add the Grafana service account in the conditions of the trust relationship.

**To add the Grafana service account in the conditions of the trust relationship**

1. From a terminal window, determine the namespace and the service account name for your Grafana server. For example, you could use the following command.

   ```
   kubectl get serviceaccounts -n grafana_namespace
   ```

2. In the Amazon EKS console, open the IAM role for service accounts that is associated with the EKS cluster.
3. Choose **Edit trust relationship**.
4. Update the **Condition** to include the Grafana namespace and the Grafana service account name that you found in the output of the command in step 1. The following is an example.

   JSON

   ```
   {
   ```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws-eusc:iam::111122223333:oidc-provider/
oidc.eks.eusc-de-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.eusc-de-east-1.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": [
            "system:serviceaccount:aws-amp:amp-iamproxy-query-service-
account",
            "system:serviceaccount:grafana-namespace:grafana-service-account-
name"
          ],
          "oidc.eks.eusc-de-east-1.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

5. Choose **Update trust policy**.

## Upgrade the Grafana server using Helm

This step upgrades the Grafana server to use the entries that you added to the `amp_query_override_values.yaml` file in the previous section.

Run the following commands. For more information about Helm charts for Grafana, see Grafana Community Kubernetes Helm Charts.

```
helm repo add grafana https://grafana.github.io/helm-charts
```

```
helm upgrade --install grafana grafana/grafana -n grafana_namespace -f ./
amp_query_override_values.yaml
```

## Add the Prometheus data source in Grafana

The following steps explain how to set up the Prometheus data source in Grafana to query your Amazon Managed Service for Prometheus metrics.

**To add the Prometheus data source in your Grafana server**

1. Open the Grafana console.
2. Under **Configurations**, choose **Data sources**.
3. Choose **Add data source**.
4. Choose **Prometheus**.
5. For the HTTP URL, specify the **Endpoint - query URL** displayed in the workspace details page in the Amazon Managed Service for Prometheus console.
6. In the HTTP URL that you just specified, remove the `/api/v1/query` string that is appended to the URL, because the Prometheus data source will automatically append it.
7. Under **Auth**, select the toggle for **SigV4 Auth** to enable it.

   Leave the **Assume Role ARN** and **External ID** fields blank. Then for **Default Region**, select the Region where your Amazon Managed Service for Prometheus workspace is.
8. Choose **Save & Test**.

   You should see the following message: **Data source is working**
9. Test a PromQL query against the new data source:

   a. Choose **Explore**.

   b. Run a sample PromQL query such as:

   ```
   prometheus_tsdb_head_series
   ```

# Query using Prometheus-compatible APIs

Although using a tool such as Amazon Managed Grafana is the easiest way to view and query your metrics, Amazon Managed Service for Prometheus also supports several Prometheus-compatible APIs that you can use to query your metrics. For more information about all the available Prometheus-compatible APIs, see Prometheus-compatible APIs.

The Prometheus-compatible APIs use the Prometheus query language, PromQL, to specify the data that you want to return. For details about PromQL and its syntax, see Querying Prometheus in the Prometheus documentation.

When you use these APIs to query your metrics, the requests must be signed with the AWS Signature Version 4 signing process. You can set up AWS Signature Version 4 to simplify the signing process. For more information, see aws-sigv4-proxy.

Signing through AWS SigV4 proxy can be performed using `awscurl`. The following topic Using awscurl to query Prometheus-compatible APIs walks you through using `awscurl` to set up AWS SigV4.

**Topics**

- Use awscurl to query with Prometheus-compatible APIs

## Use awscurl to query with Prometheus-compatible APIs

API requests for Amazon Managed Service for Prometheus must be signed with SigV4. You can use awscurl to simplify the querying process.

To install `awscurl`, you need to have Python 3 and pip package manager installed.

On a Linux based instance, the following command installs `awscurl`.

```
$ pip3 install awscurl
```

On a macOS machine, the following command installs `awscurl`.

```
$ brew install awscurl
```

The following example is a sample `awscurl` query. Replace the *Region*, *Workspace-id* and *QUERY* inputs with appropriate values for your use case:

```
# Define the Prometheus query endpoint URL. This can be found in the Amazon Managed
 Service for Prometheus console page
# under the respective workspace.

$ export AMP_QUERY_ENDPOINT=https://aps-workspaces.Region.amazonaws.com/
workspaces/Workspace-id/api/v1/query
```

```
# credentials are infered from the default profile
$ awscurl -X POST --region Region \
                  --service aps "${AMP_QUERY_ENDPOINT}" -d 'query=QUERY'  --header
 'Content-Type: application/x-www-form-urlencoded'
```

> **Note**
>
> Your query string must be url encoded.

For a query like `query=up`, you could get results such as:

```
{
  "status": "success",
  "data": {
    "resultType": "vector",
    "result": [
      {
        "metric": {
          "__name__": "up",
          "instance": "localhost:9090",
          "job": "prometheus",
          "monitor": "monitor"
        },
        "value": [
          1652452637.636,
          "1"
        ]
      },
    ]
  }
}
```

In order for `awscurl` to sign the provided requests, you will need to pass the valid credentials in one of the following ways:

- Provide the access key ID and the Secret key for the IAM role. You can find the access key and the secret key for the role in the https://eusc-de-east-1.console.amazonaws-eusc.eu/iam/.

  For example:

```
$ export AMP_QUERY_ENDPOINT=https://aps-workspaces.Region.amazonaws.com/
workspaces/Workspace_id/api/v1/query

$ awscurl -X POST --region <Region> \
                  --access_key <ACCESS_KEY> \
                  --secret_key <SECRET_KEY> \
                  --service aps "$AMP_QUERY_ENDPOINT?query=<QUERY>"
```

- Reference the configuration files stored in the `.aws/credentials` and `/aws/config` file. You can also choose to specify the name of the profile to be used. If unspecified, the `default` file will be used. For example:

```
$ export AMP_QUERY_ENDPOINT=https://aps-workspaces.<Region>.amazonaws.com/workspaces/
<Workspace_ID>/api/v1/query
$ awscurl -X POST --region <Region> \
                  --profile <PROFILE_NAME>
                  --service aps "$AMP_QUERY_ENDPOINT?query=<QUERY>"
```

- Use the instance profile associated with the EC2 instance.

### Executing query requests using awscurl container

When installing a different version of **Python** and the associated dependencies is not feasible, a container can be used to package the `awscurl` application and its dependencies. The following example uses a **Docker** runtime to deploy `awscurl`, but any OCI compliant runtime and image will work.

```
$ docker pull okigan/awscurl
$ export AMP_QUERY_ENDPOINT=https://aps-workspaces.Region.amazonaws.com/
workspaces/Workspace_id/api/v1/query
$ docker run --rm -it okigan/awscurl --access_key $AWS_ACCESS_KEY_ID  --secret_key
 $AWS_SECRET_ACCESS_KEY \ --region Region --service aps "$AMP_QUERY_ENDPOINT?
query=QUERY"
```

## Get statistics about your query usage for each query

Query pricing is based on the total number of query samples processed in a month from executed queries. You can get statistics about each query that you make to keep track of your samples

processed. The query response for a `query` or a `queryRange` API can include the statistics data about query samples processed by including the query parameter `stats=all` in the request. A `samples` object is created in the `stats` object and the `stats` data is returned in the response.

The `samples` object consists of the following attributes:

| Attribute | Description |
| --- | --- |
| `totalQueryableSamples` | Total number of query samples processed. This is the information to be used for billing. |
| `totalQueryableSamplesPerStep` | The number of query samples processed per each step. This is structured as an array of arrays with the timestamp in epoch and the number of samples loaded on the specific step. |

Sample requests and responses that include the `stats` information in the response are as follows:

Example for `query`:

**GET**

```
endpoint/api/v1/query?query=up&time=1652382537&stats=all
```

**Response**

```
{
    "status": "success",
    "data": {
        "resultType": "vector",
        "result": [
            {
                "metric": {
                    "__name__": "up",
                    "instance": "localhost:9090",
                    "job": "prometheus"
                },
                "value": [
                    1652382537,
                    "1"
```

```
                ]
            }
        ],
        "stats": {
            "timings": {
                "evalTotalTime": 0.00453349,
                "resultSortTime": 0,
                "queryPreparationTime": 0.000019363,
                "innerEvalTime": 0.004508405,
                "execQueueTime": 0.000008786,
                "execTotalTime": 0.004554219
            },
            "samples": {
                "totalQueryableSamples": 1,
                "totalQueryableSamplesPerStep": [
                    [
                        1652382537,
                        1
                    ]
                ]
            }
        }
    }
}
```

Example for queryRange:

**GET**

```
endpoint/api/v1/query_range?query=sum+%28rate+%28go_gc_duration_seconds_count%5B1m%5D
%29%29&start=1652382537&end=1652384705&step=1000&stats=all
```

**Response**

```
{
    "status": "success",
    "data": {
        "resultType": "matrix",
        "result": [
            {
                "metric": {},
                "values": [
                    [
```

```
                        1652383000,
                        "0"
                    ],
                    [
                        1652384000,
                        "0"
                    ]
                ]
            }
        ],
        "stats": {
            "samples": {
                "totalQueryableSamples": 8,
                "totalQueryableSamplesPerStep": [
                    [
                        1652382000,
                        0
                    ],
                    [
                        1652383000,
                        4
                    ],
                    [
                        1652384000,
                        4
                    ]
                ]
            }
        }
    }
}
```

# Anomaly detection

Amazon Managed Service for Prometheus provides anomaly detection capabilities that use machine learning algorithms to automatically identify unusual patterns in your metric data. This feature helps you proactively detect potential issues, reduce alert fatigue, and improve your monitoring effectiveness by focusing on truly anomalous behavior rather than static thresholds.

Anomaly detection in Amazon Managed Service for Prometheus uses the Random Cut Forest (RCF) algorithm, which analyzes your time series data to establish normal behavior patterns and identify deviations from those patterns. The algorithm adapts to seasonal trends, handles missing data gracefully, and provides confidence scores for detected anomalies.

## How anomaly detection works

Amazon Managed Service for Prometheus anomaly detection uses machine learning to identify unusual patterns in metrics data without manual threshold configuration. The system learns normal behavior patterns and seasonal variations, reducing false positives and enabling early issue detection. It continuously adapts to application changes, making it suitable for dynamic cloud environments.

Anomaly detection monitors application performance metrics like response times and error rates, tracks infrastructure health through CPU and memory usage, detects unusual user behavior, identifies capacity planning needs through traffic analysis, and monitors business metrics for unexpected changes. It works best with predictable patterns, seasonal variations, or gradual growth trends.

The Random Cut Forest (RCF) algorithm is used to analyze time series data. RCF creates decision trees that partition data space and identifies isolated points far from normal distribution. The algorithm learns from incoming data to build a dynamic model of normal behavior for each metric.

When enabled, it analyzes historical data to establish baseline patterns and seasonal trends, then generates predictions for expected values and identifies deviations. The algorithm produces four key outputs:

- *upper_band* - The upper boundary of expected normal values
- *lower_band* - The lower boundary of expected normal values
- *score* - A numerical anomaly score indicating how unusual the data point is

- *value* - The actual observed metric value

## Getting started with anomaly detection

To begin using anomaly detection with your Prometheus metrics, you need sufficient historical data for the algorithm to learn normal patterns. We recommend having at least 14 days of consistent metric data before enabling anomaly detection for optimal results.

You can preview how anomaly detection will work with your metrics using the `PreviewAnomalyDetector` API. Use `PreviewAnomalyDetector` to test the algorithm against your historical data and evaluate its effectiveness before implementing it in production monitoring. For more information, see PreviewAnomalyDetector API.

When implementing anomaly detection, consider these best practices:

- **Start with stable metrics** – Begin with metrics that have consistent patterns and avoid highly volatile or sparse data initially.
- **Use aggregated data** – Apply anomaly detection to aggregated metrics (such as averages or sums) rather than raw, high-cardinality data for better performance and accuracy.
- **Tune sensitivity** – Adjust the algorithm parameters based on your specific use case and tolerance for false positives versus missed anomalies.
- **Monitor algorithm performance** – Regularly review detected anomalies to ensure the algorithm continues to provide valuable insights as your system evolves.

## PreviewAnomalyDetector API

Use the `PreviewAnomalyDetector` operation to create an endpoint that demonstrates how your metric data will be analyzed by the anomaly detection algorithm during your specified time period. This endpoint helps you evaluate and validate the detector's performance before implementation.

Valid HTTP verbs

`GET`, `POST`

Supported payload types

URL-encoded parameters

`application/x-www-form-urlencoded` for `POST`

Supported parameters

`query=<string>` A Prometheus expression query string.

`start=<rfc3339 | unix_timestamp>` Start timestamp if you are using `query_range` to query for a range of time.

`end=<rfc3339 | unix_timestamp>` End timestamp if you are using `query_range` to query for a range of time.

`step=<duration | float>` Query resolution step width in `duration` format or as a `float` number of seconds. Use only if you are using `query_range` to query for a range of time, and required for such queries.

## Query parameter formatting

Wrap your original PromQL expression with the RandomCutForest (RCF) pseudo function in the query parameter. For more information, see RandomCutForestConfiguration in the *Amazon Managed Service for Prometheus API Reference*.

The RCF function uses this format:

```
RCF(<query>
[,shingle size
[,sample size
[,ignore near expected from above
[,ignore near expected from below
[,ignore near expected from above ratio
[,ignore near expected from below ratio]]]]])
```

All parameters except the query are optional and use default values when omitted. The minimal syntax is:

```
RCF(<query>)
```

You must wrap your query with an aggregation function. To use specific optional parameters while omitting others, leave empty positions in the function:

```
RCF(<query>,,,,,1.0,1.0)
```

This example sets only the ratio parameters that ignore anomaly detection spikes and drops based on the ratio between expected and observed values.

## API request and response

Successful calls return the same format as the QueryMetrics API. In addition to the original time series, the API returns these new time series when sufficient samples are available:

- `anomaly_detector_preview:lower_band` – Lower band for the expected value of the PromQL expression result
- `anomaly_detector_preview:score` – Anomaly score between 0 and 1, where 1 indicates high confidence of an anomaly at that data point
- `anomaly_detector_preview:upper_band` – Upper band for the expected value of the PromQL expression result

### Sample request

```
POST /workspaces/workspace-id/anomalydetectors/preview
Content-Type: application/x-www-form-urlencoded

query=RCF%28avg%28vector%28time%28%29%29%29%2C%208%2C
%20256%29&start=1735689600&end=1735695000&step=1m
```

### Sample response

```
200 OK
...

{
  "status": "success",
  "data": {
    "result": [
      {
        "metric": {},
        "values": [
          [
            1735689600,
            "1735689600"
```

```
          ],
          [
            1735689660,
            "1735689660"
          ],
          .........
        ]
      },
      {
        "metric": {
          "anomaly_detector_preview": "upper_band"
        },
        "values": [
          [
            1735693500,
            "1.7356943E9"
          ],
          [
            1735693560,
            "1.7356945E9"
          ]
          ],
          .........
        ]
      },
      {
        "metric": {
          "anomaly_detector_preview": "lower_band"
        },
        "values": [
          [
            1735693500,
            "1.7356928E9"
          ],
          [
            1735693560,
            "1.7356929E9"
          ],
          .........
        ]
      },
      {
        "metric": {
          "anomaly_detector_preview": "score"
```

```
        },
        "values": [
          [
            1735693500,
            "0.0"
          ],
          [
            1735695000,
            "0.0"
          ],
          .........
        ]
      }
    ],
    "resultType": "matrix"
  }
}
```

# Using rules to modify or monitor metrics as they are received

You can set up rules to act upon metrics as they are received by Amazon Managed Service for Prometheus. These rules can monitor the metrics or even create new, computed, metrics based on the metrics received.

Amazon Managed Service for Prometheus supports two types of *rules* that it evaluates at regular intervals:

- *Recording rules* allow you to precompute frequently needed or computationally expensive expressions and save their results as a new set of time series. Querying the precomputed result is often much faster than running the original expression every time it is needed.
- *Alerting rules* allow you to define alert conditions based on PromQL and a threshold. When the rule triggers the threshold, a notification is sent to alert manager, which can be configured to managed the rules, or forward them to notification downstream to receivers such as Amazon Simple Notification Service.

To use rules in Amazon Managed Service for Prometheus, you create one or more YAML rules files that define the rules. An Amazon Managed Service for Prometheus rules file has the same format as a rules file in standalone Prometheus. For more information, see Defining Recording rules and Alerting rules in the Prometheus documentation.

You can have multiple rules files in a workspace. Each separate rules file is contained within a separate *namespace*. Having multiple rules files lets you import existing Prometheus rules files to a workspace without having to change or combine them. Different rule group namespaces can also have different tags.

**Rule sequencing**

Within a rules file, rules are contained within *rules groups*. Rules within a single rules group in a rules file are always evaluated in order from top to bottom. Therefore, in recording rules, the result of one recording rule can be used in the computation of a later recording rule or in an alerting rule in the same rule group. However, because you can't specify the order in which to run separate rules files, you can't use the results from one recording rule to compute a rule in a different rule group or a different rules file.

**Topics**


- Understanding IAM permissions needed for using rules
- Create a rules file
- Upload a rules configuration file to Amazon Managed Service for Prometheus
- Edit or replace a rules configuration file
- Troubleshoot rule evaluations
- Troubleshooting Ruler


# Understanding IAM permissions needed for using rules

You must give users permissions to use rules in Amazon Managed Service for Prometheus. Create an AWS Identity and Access Management (IAM) policy with the following permissions, and assign the policy to your users, groups, or roles.

> **Note**
>
> For more information about IAM, see Identity and Access Management for Amazon Managed Service for Prometheus.

### Policy to give access to use rules

The following policy gives access to use rules for all resources in your account.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aps:CreateRuleGroupsNamespace",
                "aps:ListRuleGroupsNamespaces",
                "aps:DescribeRuleGroupsNamespace",
                "aps:PutRuleGroupsNamespace",
                "aps:DeleteRuleGroupsNamespace"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

**Policy to give access to only one namespace**

You can also create policy that gives access to only specific policies. The following sample policy gives access only to the `RuleGroupNamespace` specified. To use this policy, replace *<account>*, *<region>*, *<workspace-id>*, and *<namespace-name>* with appropriate values for your account.

# Create a rules file

To use rules in Amazon Managed Service for Prometheus, you create a rules file that defines the rules. An Amazon Managed Service for Prometheus rules file is a YAML text file that has the same format as a rules file in standalone Prometheus. For more information, see Defining Recording rules and Alerting rules in the *Prometheus* documentation.

The following is a basic example of a rules file:

```
groups:
  - name: cpu_metrics
    interval: 60s
    rules:
     - record: avg_cpu_usage
       expr: avg(rate(node_cpu_seconds_total[5m])) by (instance)
     - alert: HighAverageCPU
       expr: avg_cpu_usage > 0.8
       for: 10m
       keep_firing_for: 20m
       labels:
         severity: critical
       annotations:
         summary: "Average CPU usage across cluster is too high"
```

This example creates a rule group `cpu_metrics` which is evaluated every 60 seconds. This rule group creates a new metric using a recording rule, called `avg_cpu_usage` and then uses that in an alert. The following describes some of the properties used. For more information about alerting rules and other properties you can include, see Alerting rules in the *Prometheus* documentation.

- `record: avg_cpu_usage` – This recording rule creates a new metric called `avg_cpu_usage`.
- The default evaluation interval of rule groups is 60 seconds if the `interval` property is not specified.
- `expr: avg(rate(node_cpu_seconds_total[5m])) by (instance)` – This expression for the recording rule calculates the average rate of CPU usage over the last 5 minutes for each node, grouping by the `instance` label.
- `alert: HighAverageCPU` – This alert rule creates a new alert called `HighAverageCPU`
- `expr: avg_cpu_usage > 0.8` – This expression tells the alert to look for samples where the average CPU usage goes over 80%.
- `for: 10m` – The alert will only fire if the average CPU usage exceeds 80% for at least 10 minutes.

  In this case, the metric is calculated as an average over the last 5 minutes. So the alert will only fire if there are at least two consecutive 5-minute samples (10 minutes total) where the average CPU usage is above 80%.
- `keep_firing_for: 20m` – This alert will continue to fire until the samples are below the threshold for at least 20 minutes. This can be useful to avoid the alert going up and down repeatedly in succession.

> **Note**
>
> You can create a rules definition file locally and then upload it to Amazon Managed Service for Prometheus, or you can create, edit and upload the definition directly within the Amazon Managed Service for Prometheus console. Either way, the same formatting rules apply. To learn more about uploading and editing your file, see Upload a rules configuration file to Amazon Managed Service for Prometheus.

# Upload a rules configuration file to Amazon Managed Service for Prometheus

Once you know what rules you want in your rules configuration file, you can either create and edit it within the console, or you can upload a file with the console or AWS CLI.

> **Note**
>
> If you are running an Amazon EKS cluster, you can also upload a rule configuration file using AWS Controllers for Kubernetes.

**To use the Amazon Managed Service for Prometheus console to edit or replace your rules configuration and create the namespace**

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the upper left corner of the page, choose the menu icon, and then choose **All workspaces**.
3. Choose the workspace ID of the workspace, and then choose the **Rules management** tab.
4. Choose **Add namespace**.
5. Choose **Choose file**, and select the rules definition file.

   Alternately, you can create and edit a rules definition file directly in the Amazon Managed Service for Prometheus console by selecting **Define configuration**. This will create a sample default definition file that you edit before uploading.
6. (Optional) To add tags to the namespace, choose **Add new tag**.

   Then, for **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.

   To add another tag, choose **Add new tag**.
7. Choose **Continue**. Amazon Managed Service for Prometheus creates a new namespace with the same name as the rules file that you selected.

**To use the AWS CLI to upload an alert manager configuration to a workspace in a new namespace**

1. Base64 encode the contents of your alert manager file. On Linux, you can use the following command:

   ```
   base64 input-file output-file
   ```

   On macOS, you can use the following command:

```
openssl base64 input-file output-file
```

2. Enter one of the following commands to create the namespace and upload the file.

   On AWS CLI version 2, enter:

   ```
   aws amp create-rule-groups-namespace --data file://path_to_base_64_output_file --
   name namespace-name  --workspace-id my-workspace-id --region region
   ```

   On AWS CLI version 1, enter:

   ```
   aws amp create-rule-groups-namespace --data fileb://path_to_base_64_output_file --
   name namespace-name  --workspace-id my-workspace-id --region region
   ```

3. It takes a few seconds for your alert manager configuration to become active. To check the status, enter the following command:

   ```
   aws amp describe-rule-groups-namespace --workspace-id workspace_id --
   name namespace-name --region region
   ```

   If the `status` is `ACTIVE`, your rules file has taken effect.

## Edit or replace a rules configuration file

If you want to change the rules in a rule file that you have already uploaded to Amazon Managed Service for Prometheus, you can either upload a new rules file to replace the existing configuration, or you can edit the current configuration directly in the console. Optionally, you can download the current file, edit it in a text editor, then upload the new version.

**To use the Amazon Managed Service for Prometheus console to edit your rules configuration**

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the upper left corner of the page, choose the menu icon, and then choose **All workspaces**.
3. Choose the workspace ID of the workspace, and then choose the **Rules management** tab.
4. Select the name of the rules configuration file that you want to edit.

5. (Optional) If you want to download the current rules configuration file, choose **Download** or **Copy**.
6. Choose **Modify** to edit the configuration directly within the console. Choose **Save** when complete.

   Alternately, you can choose **Replace configuration** to upload a new configuration file. If so, select the new rules definition file, and choose **Continue** to upload it.

**To use the AWS CLI to edit a rules configuration file**

1. Base64 encode the contents of your rules file. On Linux, you can use the following command:

   ```
   base64 input-file output-file
   ```

   On macOS, you can use the following command:

   ```
   openssl base64 input-file output-file
   ```

2. Enter one of the following commands to upload the new file.

   On AWS CLI version 2, enter:

   ```
   aws amp put-rule-groups-namespace --data file://path_to_base_64_output_file --
   name namespace-name  --workspace-id my-workspace-id --region region
   ```

   On AWS CLI version 1, enter:

   ```
   aws amp put-rule-groups-namespace --data fileb://path_to_base_64_output_file --
   name namespace-name  --workspace-id my-workspace-id --region region
   ```

3. It takes a few seconds for your rules file to become active. To check the status, enter the following command:

   ```
   aws amp describe-rule-groups-namespace --workspace-id workspace_id --
   name namespace-name --region region
   ```

   If the `status` is `ACTIVE`, your rules file has taken effect. Until then, the previous version of this rules file is still active.

# Troubleshoot rule evaluations

This guide provides step-by-step troubleshooting procedures for common issues with rule evaluations in Amazon Managed Service for Prometheus (AMP). Follow these procedures to diagnose and resolve problems with your alerting and recording rules.

**Topics**


- Validate alert firing status
- Resolve missing alert notifications
- Check rule health status
- Use offset in queries to handle ingestion delays
- Common issues and solutions
- Best practices for rule evaluations


## Validate alert firing status

When troubleshooting rule evaluation issues, first verify if your alert has fired by querying the synthetic time series ALERTS. The ALERTS time series include the following labels:

- **alertname** – The name of the alert.
- **alertstate** – Either **pending** or **firing**.
  - **pending** – The alert is waiting for the duration specified in the `for` clause.
  - **firing** – The alert has met the conditions for the specified duration. Additional labels are defined in your alerting rule.

> **Note**
>
> While an alert is **firing** or **pending**, the sample value is **1**. When your alert is idle, no samples are produced.

## Resolve missing alert notifications

If alerts are firing but notifications are not arriving, verify the following Alertmanager settings:

1. **Verify your Alertmanager configuration** – Check that routes receivers, and settings are correctly configured. Review route block settings, including wait times, time intervals, and required labels, which can affect alert firing. Compare alerting rules with their corresponding routes and receivers to confirm proper matching. For routes with `time_interval`, verify that timestamps fall within the specified intervals.
2. **Check alert receiver permissions** – When using an Amazon SNS topic, verify AMP has the required permissions to publish notifications. For more information, see Giving Amazon Managed Service for Prometheus permission to send alert messages to your Amazon SNS topic.
3. **Validate receiver payload compatibility** – Confirm your alert receiver accepts Alertmanager's payload format. For Amazon SNS requirements, see Understanding Amazon SNS message validation rules.
4. **Review Alertmanager logs** – AMP provides vended logs from Alertmanager to help debug notification issues. For more information, see Monitor Amazon Managed Service for Prometheus events with CloudWatch Logs.

For more information about Alertmanager, see Managing and forwarding alerts in Amazon Managed Service for Prometheus with alert manager.

## Check rule health status

Malformed rules can cause evaluation failures. Use the following methods to identify why a rule failed to evaluate:

**Example**

**Use the ListRules API**

The ListRules API provides information about rule health. Check the `health` and `lastError` fields to diagnose issues.

**Example response:**

```
{
  "status": "success",
  "data": {
    "groups": [
      {
        "name": "my_rule_group",
        "file": "my_namespace",
```

```
        "rules": [
          {
            "state": "firing",
            "name": "broken_alerting_rule",
            "query": "...",
            "duration": 0,
            "keepFiringFor": 0,
            "labels": {},
            "annotations": {},
            "alerts": [],
            "health": "err",
            "lastError": "vector contains metrics with the same labelset after applying
 alert labels",
            "type": "alerting",
            "lastEvaluation": "1970-01-01T00:00:00.00000000Z",
            "evaluationTime": 0.08
          }
        ]
      }
    ]
  }
}
```

**Example**

**Use vended logs**

The ListRules API only displays the most recent information. For a more detailed history, enable vended logs in your workspace to access:

- Timestamps of evaluation failures
- Detailed error messages
- Historical evaluation data

**Example vended log message:**

```
{
  "workspaceId": "ws-a2c55905-e0b4-4065-a310-d83ce597a391",
  "message": {
    "log": "Evaluating rule failed, name=broken_alerting_rule, group=my_rule_group,
 namespace=my_namespace, err=vector contains metrics with the same labelset after
 applying alert labels",
```

```
    "level": "ERROR",
    "name": "broken_alerting_rule",
    "group": "my_rule_group",
    "namespace": "my_namespace"
  },
  "component": "ruler"
}
```

For more examples of logs from Ruler or Alertmanager, see Troubleshooting Ruler and Managing and forwarding alerts in Amazon Managed Service for Prometheus with alert manager.

## Use offset in queries to handle ingestion delays

By default, expressions are evaluated with no offset (instant query), using values at the evaluation time. If metrics ingestion is delayed, recording rules might not represent the same values as when you manually evaluate the expression after all metrics are ingested.

> **Tip**
>
> You can configure a workspace-level ruler query offset to apply a global delay to all rule evaluations without modifying individual rule expressions. For more information, see Configure your workspace. Alternatively, you can use the PromQL offset modifier on individual expressions. For more information, see Offset modifier in the *Prometheus documentation*.

**Example: Handling delayed metrics**

If your rule evaluates at 12:00, but the latest sample for the metric is from 11:45 due to ingestion delay, the rule will find no samples at the 12:00 timestamp. To mitigate this, add an offset, such as: **`my_metric_name offset 15m`** .

**Example: Handle metrics from multiple sources**

When metrics originate from different sources, such as two servers, they might be ingested at different times. To mitigate this, form an expression, such as: **`metric_from_server_A / metric_from_server_B`**

If the rule evaluates between the ingestion times of server A and server B, you'll get unexpected results. Using an offset can help align the evaluation times.

# Common issues and solutions

### Gaps in recording rule data

If you notice gaps in your recording rule data compared to manual evaluation (when you directly execute the recording rule's original PromQL expression through the query API or UI), this might be due to one of the following:

1. **Long evaluation times** – A rule group cannot have multiple simultaneous evaluations. If evaluation time exceeds the configured interval, subsequent evaluations may be missed. Multiple consecutive missed evaluations exceeding the configured interval can cause the recording rule to become stale. For more information, see Staleness in the *Prometheus documentation*. You can monitor evaluation duration using the CloudWatch metric `RuleGroupLastEvaluationDuration` to identify rule groups that are taking too long to evaluate.
2. **Monitoring missed evaluations** – AMP provides the `RuleGroupIterationsMissed` CloudWatch metric to track when evaluations are skipped. The ListRules API displays the evaluation time and last evaluation time for each rule/group, which can help identify patterns of missed evaluations. For more information, see ListRules.

### Recommendation: Split rules into separate groups

To reduce evaluation durations, split rules into separate rule groups. Rules within a group execute sequentially, while rule groups can execute in parallel. Keep related rules that depend on each other in the same group. Generally, smaller rule groups ensure more consistent evaluations and fewer gaps.

# Best practices for rule evaluations

1. **Optimize rule group size** – Keep rule groups small to ensure consistent evaluations. Group related rules together, but avoid large rule groups.
2. **Set appropriate evaluation intervals** – Balance between timely alerts and system load. Review the stability patterns of your monitored metrics to understand their normal fluctuation ranges.
3. **Use offset modifiers for delayed metrics** – Add offsets to compensate for ingestion delays. Adjust offset duration based on observed ingestion patterns.
4. **Monitor evaluation performance** – Track the `RuleGroupIterationsMissed` metric. Review evaluation times in the ListRules API.

5. **Validate rule expressions** – Ensure expressions match exactly between rule definitions and manual queries. Test expressions with different time ranges to understand behavior.
6. **Review rule health regularly** – Check for errors in rule evaluations. Monitor vended logs for recurring issues.

By following these troubleshooting steps and best practices, you can identify and resolve common issues with rule evaluations in Amazon Managed Service for Prometheus.

# Troubleshooting Ruler

Using Monitor Amazon Managed Service for Prometheus events with CloudWatch Logs, you can troubleshoot Alert Manager and Ruler related issues. This section contains ruler related troubleshooting topics.

### When the log contains the following ruler failure error

```
{
    "workspaceId": "ws-12345c67-89c0-4d12-345b-f14db70f7a99",
    "message": {
        "log": "Evaluating rule failed, name=failure,
 group=canary_long_running_vl_namespace, namespace=canary_long_running_vl_namespace,
 err=found duplicate series for the match group {dimension1=\\\"1\\\"} on the right
 hand-side of the operation: [{__name__=\\\"fake_metric2\\\", dimension1=\\\"1\\
\", dimension2=\\\"b\\\"}, {__name__=\\\"fake_metric2\\\", dimension1=\\\"1\\\",
 dimension2=\\\"a\\\"}];many-to-many matching not allowed: matching labels must be
 unique on one side",
        "level": "ERROR",
        "name": "failure",
        "group": "canary_long_running_vl_namespace",
        "namespace": "canary_long_running_vl_namespace"
    },
    "component": "ruler"
}
```

This means that some error occurred while executing the rule.

### Action to take

Use the error message to troubleshoot the rule execution.

# Managing and forwarding alerts in Amazon Managed Service for Prometheus with alert manager

When the alerting rules that Amazon Managed Service for Prometheus runs are firing, alert manager handles the alerts that are sent. It de-duplicates, groups, and routes the alerts to downstream receivers. Amazon Managed Service for Prometheus supports only Amazon Simple Notification Service as a receiver, and can route messages to Amazon SNS topics in the same account. You can also use alert manager to silence and inhibit alerts.

Alert manager provides similar functionality to Alertmanager in Prometheus.

You can use alert manager's configuration file for the following:

- **Grouping** – Grouping collects similar alerts into a single notification. This is especially useful during larger outages when many systems fail at once and hundreds of alerts might fire simultaneously. For example, suppose that a network failure causes many of your nodes to fail at the same time. If these types of alerts are grouped, alert manager sends you a single notification.

  Alert grouping and the timing for the grouped notifications are configured by a routing tree in the alert manager configuration file. For more information, see <route>.
- **Inhibition** – Inhibition suppresses notifications for certain alerts if certain other alerts are already firing. For example, if an alert is firing about a cluster being unreachable, you can configure alert manager to mute all other alerts concerning this cluster. This prevents notifications for hundreds or thousands of firing alerts that are unrelated to the actual issue. For more information about how to write inhibition rules, see <inhibit_rule>.
- **Silences** – Silences mute alerts for a specified time, such as during a maintenance window. Incoming alerts are checked for whether they match all the equality or regular expression matchers of an active silence. If they do, no notifications are sent for that alert.

  To create a silence, you use the `PutAlertManagerSilences` API. For more information, see PutAlertManagerSilences.

### Prometheus templating

Standalone Prometheus supports templating, using separate template files. Templates can use conditionals and format data, among other things.

In Amazon Managed Service for Prometheus, you put your templating in the same alert manager configuration file as your alert manager configuration.

**Topics**


- Understanding IAM permissions needed for working with alert manager
- Create an alert manager configuration in Amazon Managed Service for Prometheus to manage and route alerts
- Forward alerts to an alert receiver with alert manager in Amazon Managed Service for Prometheus
- Upload your alert manager configuration file to Amazon Managed Service for Prometheus
- Integrate alerts with Amazon Managed Grafana or open source Grafana
- Troubleshoot alert manager with CloudWatch Logs


# Understanding IAM permissions needed for working with alert manager

You must give users permissions to use alert manager in Amazon Managed Service for Prometheus. Create an AWS Identity and Access Management (IAM) policy with the following permissions, and assign the policy to your users, groups, or roles.

# Create an alert manager configuration in Amazon Managed Service for Prometheus to manage and route alerts

To use alert manager and templating in Amazon Managed Service for Prometheus, you create an alert manager configuration YAML file. An Amazon Managed Service for Prometheus alert manager file has two main sections:

- `template_files:` contains the templates used for messages sent by receivers. For more information, see Template Reference and Template Examples in the Prometheus documentation.
- `alertmanager_config:` contains the alert manager configuration. This uses the same structure as an alert manager config file in standalone Prometheus. For more information, see Configuration in the Alertmanager documentation.

> **Note**
>
> The `repeat_interval` configuration described in the Prometheus documentation above has an additional limitation in Amazon Managed Service for Prometheus. The maximum allowed value is five days. If you set it higher than five days, it will be treated as five days and notifications will be sent again after the five day period has passed.

> **Note**
>
> You can also edit the configuration file directly in the Amazon Managed Service for Prometheus console, but it must still follow the format specified here. For more information on uploading or editing a configuration file, see Upload your alert manager configuration file to Amazon Managed Service for Prometheus.

In Amazon Managed Service for Prometheus, your alert manager configuration file must have all your alert manager configuration content inside of an `alertmanager_config` key at the root of the YAML file.

The following is a basic example alert manager configuration file:

```
alertmanager_config: |
  route:
    receiver: 'default'
  receivers:
    - name: 'default'
      sns_configs:
      - topic_arn: arn:aws:sns:us-east-2:123456789012:My-Topic
        sigv4:
          region: us-east-2
        attributes:
          key: key1
          value: value1
```

The only receiver currently supported is Amazon Simple Notification Service (Amazon SNS). If you have other types of receivers listed in the configuration, it will be rejected.

Here is another sample alert manager configuration file that uses both the `template_files` block and the `alertmanager_config` block.

```
template_files:
  default_template: |
    {{ define "sns.default.subject" }}[{{ .Status | toUpper }}{{ if eq .Status
 "firing" }}:{{ .Alerts.Firing | len }}{{ end }}]{{ end }}
    {{ define "__alertmanager" }}AlertManager{{ end }}
    {{ define "__alertmanagerURL" }}{{ .ExternalURL }}/#/alerts?receiver={{ .Receiver |
 urlquery }}{{ end }}
alertmanager_config: |
  global:
  templates:
    - 'default_template'
  route:
    receiver: default
  receivers:
    - name: 'default'
      sns_configs:
      - topic_arn: arn:aws:sns:us-east-2:accountid:My-Topic
        sigv4:
          region: us-east-2
        attributes:
          key: severity
          value: SEV2
```

### Default Amazon SNS template block

The default Amazon SNS configuration uses the following template unless you explicitly override it.

```
{{ define "sns.default.message" }}{{ .CommonAnnotations.SortedPairs.Values | join "
 " }}
  {{ if gt (len .Alerts.Firing) 0 -}}
  Alerts Firing:
    {{ template "__text_alert_list" .Alerts.Firing }}
  {{- end }}
  {{ if gt (len .Alerts.Resolved) 0 -}}
  Alerts Resolved:
    {{ template "__text_alert_list" .Alerts.Resolved }}
  {{- end }}
{{- end }}
```

# Forward alerts to an alert receiver with alert manager in Amazon Managed Service for Prometheus

When an alert is raised by an alert rule, it is sent to alert manager. Alert manager performs functions such as de-duplicating alerts, inhibiting alerts during maintenance, or grouping them as needed. It then forwards the alert as a message to an *alert receiver*. You can set up an alert receiver that can notify operators, have automated responses, or respond to the alerts in other ways.

You can configure Amazon Simple Notification Service (Amazon SNS) and PagerDuty as alert receivers in Amazon Managed Service for Prometheus. The following topics describe how to create and configure your alert receiver.

**Topics**


- Use Amazon SNS as an alert receiver
- Use PagerDuty as an alert receiver


## Use Amazon SNS as an alert receiver

You can use an existing Amazon SNS topic as an alert receiver for Amazon Managed Service for Prometheus, or you can create a new one. We recommend that you use a topic of the **Standard** type, so that you can forward alerts from the topic to email, SMS, or HTTP.

To create a new Amazon SNS topic to use as your alert manager receiver, follow the steps in Step 1: Create a topic. Be sure to choose **Standard** for the topic type.

If you want to receive emails every time a message is sent to that Amazon SNS topic, follow the steps in Step 2: Create a subscription to the topic.

Whether you use a new or existing Amazon SNS topic, you will need the Amazon Resource Name (ARN) of your Amazon SNS topic to complete the following tasks.

**Topics**


- Giving Amazon Managed Service for Prometheus permission to send alert messages to your Amazon SNS topic
- Configure alert manager to send messages to your Amazon SNS topic
- Configure alert manager to send messages to Amazon SNS as JSON

- Configure Amazon SNS to send messages for alerts to other destinations
- Understanding Amazon SNS message validation rules

## Giving Amazon Managed Service for Prometheus permission to send alert messages to your Amazon SNS topic

You must give Amazon Managed Service for Prometheus permission to send messages to your Amazon SNS topic. The following policy statement will give that permission. It includes a `Condition` statement to help prevent a security problem known as the *confused deputy* problem. The `Condition` statement restricts access to the Amazon SNS topic to allow only operations coming from this specific account and Amazon Managed Service for Prometheus workspace. For more information about the confused deputy problem, see Cross-service confused deputy prevention.

**To give Amazon Managed Service for Prometheus permission to send messages to your Amazon SNS topic**

1. Open the Amazon SNS console at https://eusc-de-east-1.console.amazonaws-eusc.eu/sns/v3/home.
2. In the navigation pane, choose **Topics**.
3. Choose the name of the topic that you are using with Amazon Managed Service for Prometheus.
4. Choose **Edit**.
5. Choose **Access policy** and add the following policy statement to the existing policy.

```
{
    "Sid": "Allow_Publish_Alarms",
    "Effect": "Allow",
    "Principal": {
        "Service": "aps.amazonaws.com"
    },
    "Action": [
        "sns:Publish",
        "sns:GetTopicAttributes"
    ],
    "Condition": {
        "ArnEquals": {
            "aws:SourceArn": "workspace_ARN"
```

```
        },
        "StringEquals": {
            "AWS:SourceAccount": "account_id"
        }
    },
    "Resource": "arn:aws:sns:region:account_id:topic_name"
}
```

[Optional] If your Amazon SNS topic is service side encryption (SSE) enabled, you need to allow Amazon Managed Service for Prometheus to send messages to this encrypted topic by adding the `kms:GenerateDataKey*` and `kms:Decrypt` permissions to the AWS KMS key policy of the key used to encrypt the topic.

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

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "aps.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

For more information, see AWS KMS Permissions for SNS Topic.

6. Choose **Save changes**.

> **Note**
>
> By default, Amazon SNS creates the access policy with condition on `AWS:SourceOwner`. For more information, see SNS Access Policy.

**Note**

IAM follows the Most-restrictive policy first rule. In your SNS topic, if there is a policy block that is more restrictive than the documented Amazon SNS policy block, the permission for the topic policy is not granted. To evaluate your policy and find out what's been granted, see Policy evaluation logic.

### SNS topic configuration for opt-in regions

You can use `aps.amazonaws.com` to configure an Amazon SNS topic in the same AWS Region as your Amazon Managed Service for Prometheus workspace. To use an SNS topic from a non-opt-in Region (such as us-east-1) with an opt-in Region (such as af-south-1), you need to use the Regional service principal format. In the Regional service principle, replace *`us-east-1`* with the non-opt-in Region you want to use: **`aps.us-east-1.amazonaws.com`**.

The following table lists the opt-in Regions and their corresponding Regional service principals:

**Opt-in Regions and their Regional service principals**

| Region name | Region | Regional service principal |
|---|---|---|
| Africa (Cape Town) | af-south-1 | af-south-1.aps.amazonaws.com |
| Asia Pacific (Hong Kong) | ap-east-1 | ap-east-1.aps.amazonaws.com |
| Asia Pacific (Thailand) | ap-southeast-7 | ap-southeast-7.aps.amazonaws.com |
| Europe (Milan) | eu-south-1 | eu-south-1.aps.amazonaws.com |
| Europe (Zurich) | eu-central-2 | eu-central-2.aps.amazonaws.com |
| Middle East (UAE) | me-central-1 | me-central-1.aps.amazonaws.com |

| Region name | Region | Regional service principal |
| --- | --- | --- |
| Asia Pacific (Malaysia) | ap-southeast-5 | ap-southeast-5.aps .amazonaws.com |

For information on enabling an opt-in Region, see Managing AWS Regions in the *IAM User Guide* in the Amazon Web Services General Reference.

When configuring your Amazon SNS topic for these opt-in Regions, ensure you use the correct Regional service principal to enable cross-region delivery of alerts.

**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 Managed Service for Prometheus gives to Amazon SNS to the resource. If you use both global condition context keys, 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.

The value of `aws:SourceArn` must be the ARN of the Amazon Managed Service for Prometheus workspace.

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:servicename::123456789012:*`.

The policy shown in Giving Amazon Managed Service for Prometheus permission to send alert messages to your Amazon SNS topic shows how you can use the `aws:SourceArn` and

`aws:SourceAccount` global condition context keys in Amazon Managed Service for Prometheus to prevent the confused deputy problem.

## Configure alert manager to send messages to your Amazon SNS topic

After you have a (new or existing) **Standard** type Amazon SNS topic, you can add it to your alert manager configuration as an alert receiver. Alert manager can forward your alerts to a configured alert receiver. To complete this, you must know the Amazon Resource Name (ARN) of your Amazon SNS topic.

For more information about Amazon SNS receiver configuration, see <sns_configs> in the Prometheus configuration documentation.

### Unsupported properties

Amazon Managed Service for Prometheus supports Amazon SNS as the alert receiver. However, because of service constraints, not all of the properties of the Amazon SNS receiver are supported. The following properties are not allowed in an Amazon Managed Service for Prometheus alert manager configuration file:

- `api_url:` – Amazon Managed Service for Prometheus sets the `api_url` for you, so this property is not allowed.
- `Http_config` – This property allows you to set external proxies. Amazon Managed Service for Prometheus does not currently support this feature.

Additionally, SigV4 settings are required to have a Region property. Without the Region property, Amazon Managed Service for Prometheus doesn't have enough information to make the authorization request.

### To configure alert manager with your Amazon SNS topic as the receiver

1. If you are using an existing alert manager configuration file, open it in a text editor.
2. If there are current receivers other than Amazon SNS in the `receivers` block, remove them. You can configure multiple Amazon SNS topics to be receivers by putting them in separate `sns_config` blocks within the `receivers` block.
3. Add the following YAML block within the `receivers` section.

```
- name: name_of_receiver
  sns_configs:
```

```
    - sigv4:
        region: AWS Region
      topic_arn: ARN_of_SNS_topic
      subject: yoursubject
      attributes:
        key: yourkey
        value: yourvalue
```

If a `subject`is not specified, by default, a subject would be generated with the default template with the label name and values, which may result in a value that is too long for SNS. To change the template that is applied to the subject, refer to Configure alert manager to send messages to Amazon SNS as JSON in this guide.

Now you must upload your alert manager configuration file to Amazon Managed Service for Prometheus. For more information, see Upload your alert manager configuration file to Amazon Managed Service for Prometheus.

## Configure alert manager to send messages to Amazon SNS as JSON

By default, Amazon Managed Service for Prometheus alert manager outputs messages in a plain text list format. This can be more difficult for other services to parse. You can configure alert manager to send alerts in JSON format instead. JSON can make it simpler to process the messages downstream from Amazon SNS in AWS Lambda or in webhook-receiving endpoints. Instead of using the default template, you can define a custom template to output the message contents in JSON, making it easier to parse in downstream functions.

To output messages from alert manager to Amazon SNS in JSON format, update your alert manager configuration to contain the following code inside your `template_files` root section:

```
default_template: |
   {{ define "sns.default.message" }}{{ "{" }}"receiver": "{{ .Receiver }}","status":
 "{{ .Status }}","alerts": [{{ range $alertIndex, $alerts := .Alerts }}{{ if
 $alertIndex }}, {{ end }}{{ "{" }}"status": "{{ $alerts.Status }}"{{ if
 gt (len $alerts.Labels.SortedPairs) 0 -}},"labels": {{ "{" }}{{ range
 $index, $label := $alerts.Labels.SortedPairs }}{{ if $index }},
 {{ end }}"{{ $label.Name }}": "{{ $label.Value }}"{{ end }}
{{ "}" }}{{- end }}{{ if gt (len $alerts.Annotations.SortedPairs )
 0 -}},"annotations": {{ "{" }}{{ range $index, $annotations :=
 $alerts.Annotations.SortedPairs }}{{ if $index }}, {{ end }}"{{ $annotations.Name }}":
 "{{ $annotations.Value }}"{{ end }}{{ "}" }}{{- end }},"startsAt":
```

```
 "{{ $alerts.StartsAt }}","endsAt": "{{ $alerts.EndsAt }}","generatorURL":
 "{{ $alerts.GeneratorURL }}","fingerprint": "{{ $alerts.Fingerprint }}"{{ "}" }}
{{ end }}]{{ if gt (len .GroupLabels) 0 -}},"groupLabels": {{ "{" }}{{ range
 $index, $groupLabels := .GroupLabels.SortedPairs }}{{ if $index }},
 {{ end }}"{{ $groupLabels.Name }}": "{{ $groupLabels.Value }}"{{ end }}
{{ "}" }}{{- end }}{{ if gt (len .CommonLabels) 0 -}},"commonLabels": {{ "{" }}
{{ range $index, $commonLabels := .CommonLabels.SortedPairs }}{{ if $index }},
 {{ end }}"{{ $commonLabels.Name }}": "{{ $commonLabels.Value }}"{{ end }}{{ "}" }}{{-
 end }}{{ if gt (len .CommonAnnotations) 0 -}},"commonAnnotations": {{ "{" }}{{ range
 $index, $commonAnnotations := .CommonAnnotations.SortedPairs }}{{ if $index }},
 {{ end }}"{{ $commonAnnotations.Name }}": "{{ $commonAnnotations.Value }}"{{ end }}
{{ "}" }}{{- end }}{{ "}" }}{{ end }}
   {{ define "sns.default.subject" }}[{{ .Status | toUpper }}{{ if eq .Status
 "firing" }}:{{ .Alerts.Firing | len }}{{ end }}]{{ end }}
```

> **Note**
>
> This template creates JSON from alphanumeric data. If your data has special characters, encode them before using this template.

To make sure that this template is used in outgoing notifications, reference it in your `alertmanager_config` block as follows:

```
alertmanager_config: |
  global:
  templates:
    - 'default_template'
```

> **Note**
>
> This template is for the entire message body as JSON. This template overwrites the entire message body. You cannot override the message body if you wish to use this specific template. Any overrides that are manually done will take precedence over the template.

For more information about:

- The alert manager configuration file, see Create an alert manager configuration in Amazon Managed Service for Prometheus to manage and route alerts.

- Uploading your configuration file, see Upload your alert manager configuration file to Amazon Managed Service for Prometheus.

## Configure Amazon SNS to send messages for alerts to other destinations

Amazon Managed Service for Prometheus can only send alert messages to Amazon Simple Notification Service (Amazon SNS). To send those messages to other destinations, such as email, webhook, Slack, or OpsGenie, you must configure Amazon SNS to forward the messages on to those endpoints.

The following sections describing configuring Amazon SNS to forward alerts to other destinations.

**Topics**

- Email
- Webhook
- Slack
- OpsGenie

### Email

To configure an Amazon SNS topic to output messages to email, create a subscription. In the Amazon SNS console, choose the **Subscriptions** tab to open the **Subscriptions** list page. Choose **Create Subscription** and select **Email**. Amazon SNS sends a confirmation email to the listed email address. After you accept the confirmation, you are able to receive Amazon SNS notifications as emails from the topic you subscribed to. For more information, see Subscribing to an Amazon SNS topic.

### Webhook

To configure an Amazon SNS topic to output messages to a webhook endpoint, create a subscription. In the Amazon SNS console, choose the **Subscriptions** tab to open the **Subscriptions** list page. Choose **Create Subscription** and select **HTTP/HTTPS**. After you create the subscription, you must follow the confirmation steps to activate it. When it is active, your HTTP endpoint should receive the Amazon SNS notifications. For more information, see Subscribing to an Amazon SNS topic. For more information about using Slack webhooks to publish messages to various destinations, see How do I use webhooks to publish Amazon SNS messages to Amazon Chime, Slack, or Microsoft Teams?

### Slack

To configure an Amazon SNS topic to output messages to Slack, you have two options. You can either integrate with Slack's email-to-channel integration, which allows Slack to accept email messages and forward them to a Slack channel, or you can use a Lambda function to rewrite the Amazon SNS notification to Slack. For more information about forwarding emails to slack channels, see Confirming AWS SNS Topic Subscription for Slack Webhook. For more information about constructing a Lambda function to convert Amazon SNS messages to Slack, see How to integrate Amazon Managed Service for Prometheus with Slack.

### OpsGenie

For information about how to configure an Amazon SNS topic to output messages to OpsGenie, see Integrate Opsgenie with Incoming Amazon SNS.

## Understanding Amazon SNS message validation rules

Amazon Simple Notification Service (Amazon SNS) requires messages to meet certain standards. Messages that don't meed these standards will be modified when they are received. The alert messages will be validated, truncated, or modified, if necessary, by the Amazon SNS receiver based on the following rules:

- Message contains non-utf characters.
  - Message will be replaced by **Error - not a valid UTF-8 encoded string**.
  - One message attribute will be added with the key of **truncated** and the value of **true**.
  - One message attribute will be added with the key of **modified** and the value of **Message: Error - not a valid UTF-8 encoded string**.
- Message is empty.
  - Message will be replaced by **Error - Message should not be empty**.
  - One message attribute will be added with the key of **modified** and the value of **Message: Error - Message should not be empty**.
- Message has been truncated.
  - Message will have the truncated content.
  - One message attribute will be added with the key of **truncated** and the value of **true**.
  - One message attribute will be added with the key of "modified" and the value of **Message: Error - Message has been truncated from *X* KB, because it exceeds the 256 KB size limit**.
- Subject contains control or non-ASCII characters.

  - If the subject contains control characters or non-ASCII characters, SNS replaces the subject with **Error - contains control- or non-ASCII characters**.
  - For SNS email subjects, remove control characters, such as newlines: \n.
- Subject is not ASCII.
  - Subject will be replaced by **Error - contains non printable ASCII characters**.
  - One message attribute will be added with the key of **modified** and the value of **Subject: Error - contains non-printable ASCII characters**.
- Subject has been truncated.
  - Subject will have the truncated content.
  - One message attribute will be added with the key of **modified** and the value of **Subject: Error - Subject has been truncated from *X* characters, because it exceeds the 100 character size limit**.
- Message attribute has invalid key/value.
  - Invalid message attribute will be removed.
  - One message attribute will be added with the key of **modified** and the value of **MessageAttribute: Error - *X* of the message attributes have been removed because of invalid MessageAttributeKey or MessageAttributeValue**.
- Message attribute has been truncated.
  - Extra message attributes will be removed.
  - One message attribute will be added with the key of **modified** and the value of **MessageAttribute: Error - *X* of the message attributes have been removed, because it exceeds the 256KB size limit**.

## Use PagerDuty as an alert receiver

You can configure Amazon Managed Service for Prometheus to send alerts directly to PagerDuty. This integration requires you to store your PagerDuty integration key in AWS Secrets Manager and grant Amazon Managed Service for Prometheus permission to read the secret.

PagerDuty integration enables automated incident response workflows and ensures critical alerts reach the right team members at the right time. When you use PagerDuty as an alert receiver, you can take advantage of PagerDuty's escalation policies, on-call scheduling, and incident management features to ensure that alerts are acknowledged and resolved quickly. This integration is particularly valuable for production environments where rapid response to system

issues is essential for maintaining service availability and meeting SLA requirements. For more information, see PagerDuty Knowledge Base on the *PagerDuty website*.

## PagerDuty configuration options

| Option | Description | Required |
|---|---|---|
| `routing_key` | The PagerDuty routing key for an integration on a service. You must specify this as an Secrets Manager ARN | Yes |
| `service_key` | The PagerDuty service key for an integration on a service. You must specify this as an Secrets Manager ARN | Yes (for Events API v1) |
| `client` | The client identification of the notifier | No |
| `client_url` | A backlink to the sender of the notification | No |
| `description` | Description of the incident | No |
| `details` | A set of arbitrary key/value pairs that provide further detail about the incident | No |
| `severity` | Severity of the incident | No |
| `class` | The class, or type, of the event | No |
| `component` | Component of the source machine that is responsible for the event | No |

| Option | Description | Required |
| --- | --- | --- |
| `group` | Logical grouping of components | No |
| `source` | The unique location of the affected system | No |

> **Note**
>
> The `url`, `service_key_file`, `routing_key_file`, and `http_config` options are not supported.

The following topics describe how to configure PagerDuty as an alert receiver in Amazon Managed Service for Prometheus.

**Topics**

- Configure AWS Secrets Manager and permissions
- Configure alert manager to send alerts to PagerDuty

## Configure AWS Secrets Manager and permissions

Before you can send alerts to PagerDuty, you must securely store your PagerDuty integration key and configure the necessary permissions. This process involves creating a secret in AWS Secrets Manager, encrypting it with a customer-managed AWS Key Management Service (AWS KMS) key, and granting Amazon Managed Service for Prometheus the required permissions to access both the secret and its encryption key. The following procedures guide you through each step of this configuration process.

**To create a secret in Secrets Manager for PagerDuty**

To use PagerDuty as an alert receiver, you must store your PagerDuty integration key in Secrets Manager. Follow these steps:

1. Open the Secrets Manager console.
2. Choose **Store a new secret**.

3. For **Secret type**, choose **Other type of secret**.
4. For **Key/value pairs**, enter your PagerDuty integration key as the secret value. This is either the routing key or service key from your PagerDuty integration.
5. Choose **Next**.
6. Enter a name and description for your secret, then choose **Next**.
7. Configure rotation settings if desired, then choose **Next**.
8. Review your settings and choose **Store**.
9. After creating the secret, note its ARN. You'll need this when configuring the alert manager.

**To encrypt your secret with a customer-managed AWS KMS key**

You must grant Amazon Managed Service for Prometheus permission to access your secret and its encryption key:

1. **Secret resource policy**: Open your secret in the Secrets Manager console.
   a. Choose **Resource permissions**.
   b. Choose **Edit permissions**.
   c. Add the following policy statement. In the statement, replace the *highlighted values* with your specific values.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "aps.amazonaws.com"
  },
  "Action": "secretsmanager:GetSecretValue",
  "Resource": "*",
  "Condition": {
    "ArnEquals": {
      "aws:SourceArn": "arn:aws:aps:aws-
region:123456789012:workspace/WORKSPACE_ID"
    },
    "StringEquals": {
      "aws:SourceAccount": "123456789012"
    }
  }
}
```

d. Choose **Save**.

2. **KMS key policy**: Open your AWS KMS key in the AWS KMS console.

   a. Choose **Key policy**.

   b. Choose **Edit**.

   c. Add the following policy statement. In the statement, replace the *highlighted values* with your specific values.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "aps.amazonaws.com"
  },
  "Action": "kms:Decrypt",
  "Resource": "*",
  "Condition": {
    "ArnEquals": {
      "aws:SourceArn": "arn:aws:aps:aws-
region:123456789012:workspace/WORKSPACE_ID"
    },
    "StringEquals": {
      "aws:SourceAccount": "123456789012"
    }
  }
}
```

   d. Choose **Save**.

**Next steps** – Continue to the next topic, Configure alert manager to send alerts to PagerDuty.

## Configure alert manager to send alerts to PagerDuty

To configure alert manager to send alerts to PagerDuty, you need to update your alert manager definition. You can do this using the AWS Management Console, AWS CLI, or AWS SDKs.

### Example alert manager configuration

Following, is an example alert manager configuration that sends alerts to PagerDuty. In the example, replace the *highlighted values* with your specific values.

```
alertmanager_config: |
```

```
  route:
    receiver: 'pagerduty-receiver'
    group_by: ['alertname']
    group_wait: 30s
    group_interval: 5m
    repeat_interval: 1h
  receivers:
    - name: 'pagerduty-receiver'
      pagerduty_configs:
      - routing_key:
          aws_secrets_manager:
            secret_arn: 'arn:aws:secretsmanager:aws-
region:123456789012:secret:YOUR_SECRET_NAME'
            secret_key: 'YOUR_SECRET_KEY'
            refresh_interval: 5m
        description: '{{ .CommonLabels.alertname }}'
        severity: 'critical'
        details:
          firing: '{{ .Alerts.Firing | len }}'
          status: '{{ .Status }}'
          instance: '{{ .CommonLabels.instance }}'
```

### Example AWS CLI

Following, is an AWS CLI command used to update your alert manager definition. In the example, replace the *highlighted values* with your specific values.

```
aws amp put-alert-manager-definition \
  --workspace-id WORKSPACE_ID \
  --data file://alertmanager-config.yaml
```

### Troubleshooting PagerDuty integration

If alerts are not being sent to PagerDuty, check the following items:

- Verify that your secret exists and contains the correct PagerDuty integration key.
- Confirm that your secret is encrypted with a customer-managed KMS key.
- Ensure that the resource policies for both the secret and the KMS key grant the necessary permissions to Amazon Managed Service for Prometheus.

- Check that the ARN in your alert manager configuration correctly references your secret.
- Verify that your PagerDuty integration key is valid and active in your PagerDuty account.

Amazon Managed Service for Prometheus supports Amazon CloudWatch Logs, and the following CloudWatch metrics, to help with troubleshooting. For more information, see Monitor Amazon Managed Service for Prometheus events with CloudWatch Logs and Use CloudWatch metrics to monitor Amazon Managed Service for Prometheus resources.

- `SecretFetchFailure`
- `AlertManagerNotificationsThrottledByIntegration`
- `AlertManagerNotificationsFailedByIntegration`

## Upload your alert manager configuration file to Amazon Managed Service for Prometheus

Once you know what you want in your Alert manager configuration file, you can create and edit it within the console, or you can upload an existing file with the Amazon Managed Service for Prometheus console or AWS CLI.

> **Note**
>
> If you are running an Amazon EKS cluster, you can also upload an Alert manager configuration file using AWS Controllers for Kubernetes.

**To use the Amazon Managed Service for Prometheus console to edit or replace your alert manager configuration**

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the upper left corner of the page, choose the menu icon, and then choose **All workspaces**.
3. Choose the workspace ID of the workspace, and then choose the **Alert manager** tab.
4. If the workspace doesn't already have an alert manager definition, choose **Add definition**.

> **Note**
>
> If the workspace has an alert manager definition that you want to replace, choose **Modify** instead.

5. Choose **Choose file**, select the alert manager definition file, and choose **Continue**.

> **Note**
>
> Alternately, you can create a new file and edit it directly in the console, by choosing the **Create definition** option. This will create a sample default configuration that you edit before uploading.

**To use the AWS CLI to upload an alert manager configuration to a workspace for the first time**

1. Base64 encode the contents of your alert manager file. On Linux, you can use the following command:

```
base64 input-file output-file
```

On macOS, you can use the following command:

```
openssl base64 input-file output-file
```

2. To upload the file, enter one of the following commands.

On AWS CLI version 2, enter:

```
aws amp create-alert-manager-definition --data file://path_to_base_64_output_file
 --workspace-id my-workspace-id --region region
```

On AWS CLI version 1, enter:

```
aws amp create-alert-manager-definition --data fileb://path_to_base_64_output_file
 --workspace-id my-workspace-id --region region
```

3. It takes a few seconds for your alert manager configuration to become active. To check the status, enter the following command:

```
aws amp describe-alert-manager-definition --workspace-id workspace_id --
region region
```

If the `status` is `ACTIVE`, your new alert manager definition has taken effect.

**To use the AWS CLI to replace a workspace's alert manager configuration with a new one**

1. Base64 encode the contents of your alert manager file. On Linux, you can use the following command:

```
base64 input-file output-file
```

On macOS, you can use the following command:

```
openssl base64 input-file output-file
```

2. To upload the file, enter one of the following commands.

On AWS CLI version 2, enter:

```
aws amp put-alert-manager-definition --data file://path_to_base_64_output_file --
workspace-id my-workspace-id --region region
```

On AWS CLI version 1, enter:

```
aws amp put-alert-manager-definition --data file://path_to_base_64_output_file --
workspace-id my-workspace-id --region region
```

3. It takes a few seconds for your new alert manager configuration to become active. To check the status, enter the following command:

```
aws amp describe-alert-manager-definition --workspace-id workspace_id --
region region
```

If the `status` is `ACTIVE`, your new alert manager definition has taken effect. Until that time, your previous alert manager configuration is still active.

# Integrate alerts with Amazon Managed Grafana or open source Grafana

Alert rules that you have created in Alertmanager within Amazon Managed Service for Prometheus can be forwarded and viewed in Amazon Managed Grafana and Grafana, unifying your alert rules and alerts in a single environment. Within Amazon Managed Grafana, you can view your alert rules and the alerts that are generated.

## Prerequisites

Before starting to integrate Amazon Managed Service for Prometheus into Amazon Managed Grafana, you must have completed the following prerequisites:

- You must have an existing AWS account and IAM credentials to create Amazon Managed Service for Prometheus and IAM roles programmatically.

  For more information about creating an AWS account and IAM credentials, see Set up AWS.
- You must have an Amazon Managed Service for Prometheus workspace, and be ingesting data into it. To set up a new workspace, see Create an Amazon Managed Service for Prometheus workspace. You should also be familiar with the Prometheus concepts such as Alertmanager and Ruler. For more information about these topics, see the Prometheus documentation.
- You have an Alertmanager configuration and a rules file already configured in Amazon Managed Service for Prometheus. For more information about Alertmanager in Amazon Managed Service for Prometheus, see Managing and forwarding alerts in Amazon Managed Service for Prometheus with alert manager. For more information about rules, see Using rules to modify or monitor metrics as they are received.
- You must either have Amazon Managed Grafana set up, or the open source version of Grafana running.
  - If you are using Amazon Managed Grafana, you must be using Grafana alerting. For more information see Migrating legacy dashboard alerts to Grafana alerting.
  - If you are using the open source version of Grafana, you must be running version 9.1 or higher.

> **Note**
>
> You can use earlier versions of Grafana, but you must enable the unified alerting (Grafana alerting) feature, and you might have to set up a sigv4 proxy to make calls

from Grafana to Amazon Managed Service for Prometheus. For more information, see Set up Grafana open source or Grafana Enterprise for use with Amazon Managed Service for Prometheus.

- Amazon Managed Grafana must have the following permissions for your Prometheus resources. You must add them to either the service-managed or customer-managed policies described in https://docs.aws.eu/grafana/latest/userguide/AMG-manage-permissions.html.
  - `aps:ListRules`
  - `aps:ListAlertManagerSilences`
  - `aps:ListAlertManagerAlerts`
  - `aps:GetAlertManagerStatus`
  - `aps:ListAlertManagerAlertGroups`
  - `aps:PutAlertManagerSilences`
  - `aps:DeleteAlertManagerSilence`

## Setting up Amazon Managed Grafana

If you have already set up rules and alerts in your Amazon Managed Service for Prometheus instance, the configuration to use Amazon Managed Grafana as a dashboard for those alerts is done entirely within Amazon Managed Grafana.

**To configure Amazon Managed Grafana as your alerts dashboard**

1. Open the Grafana console for your workspace.
2. Under **Configurations**, choose **Data sources**.
3. Either create or open your Prometheus data source. If you have not previously set up a Prometheus data source, see Step 2: Add the Prometheus data source in Grafana for more information.
4. In the Prometheus data source, select **Manage alerts via Alertmanager UI**.
5. Go back to the **Data sources** interface.
6. Create a new Alertmanager data source.
7. In the Alertmanager data source configuration page, add the following settings:
   - Set **Implementation** to `Prometheus`.

- For the **URL** setting, use the URL for your Prometheus workspace, remove everything after the workspace ID, and append `/alertmanager` to the end. In the following example, replace the *`variables`* with you own (account specific) information:

  ```
  https://aps-workspaces.US East (N. Virginia).amazonaws.com/workspaces/ws-example-1234-5678-abcd-xyz00000001/alertmanager.
  ```

- Under **Auth**, turn on **SigV4Auth**. This tells Grafana to use the AWS authentication for the requests.
- Under **SigV4Auth Details**, for **Default Region**, provide the region of your Prometheus instance, for example `us-east-1`.
- Set the **Default** option to `true`.

8. Choose **Save and test**.
9. Your Amazon Managed Service for Prometheus alerts should now be configured to work with your Grafana instance. Verify that you can see any **Alert rules**, **Alert groups** (including active alerts), and **Silences** from your Amazon Managed Service for Prometheus instance in the Grafana **Alerting** page.

# Troubleshoot alert manager with CloudWatch Logs

Using Monitor Amazon Managed Service for Prometheus events with CloudWatch Logs, you can troubleshoot Alert Manager and Ruler related issues. This section contains Alert Manager related troubleshooting topics.

**Topics**


- Active alerts warning
- Alert aggregation group size warning
- Alerts size too big warning
- Empty content warning
- Invalid key/value warning
- Message limit warning
- No resource based policy error
- Non ASCII warning
- Not authorized to call KMS

- Template error

## Active alerts warning

### When the log contains the following warning

```
{
    "workspaceId": "ws-efdc5b42-b051-11ec-b123-4567ac120002",
    "message": {
        "log": "too many alerts, limit: 1000",
        "level": "WARN"
    },
    "component": "alertmanager"
}
```

This means that the Alert manager **Active alerts** quota is exceeded.

### Action to take

Request a quota increase. Sign in to the AWS Management Console and open the Service Quotas console at https://eusc-de-east-1.console.amazonaws-eusc.eu/servicequotas/.

## Alert aggregation group size warning

### When the log contains the following warning

```
{
    "workspaceId": "ws-efdc5b42-b051-11ec-b123-4567ac120002",
    "message": {
        "log": "Too many aggregation groups, cannot create new group for alert,
 groups=1000, limit=1000, alert=sample-alert",
        "level": "WARN"
    },
    "component": "alertmanager"
}
```

This means that the Alert manager Alert aggregation group size quota has been exceeded.

### Action to take

Reduce the Alert aggregation group size by using the `group_by` parameter. For more information, see Route-related settings in the*Prometheus documentation*.

You can also request a quota increase. Sign in to the AWS Management Console and open the Service Quotas console at https://eusc-de-east-1.console.amazonaws-eusc.eu/servicequotas/.

## Alerts size too big warning

### When the log contains the following warning

```
{
    "workspaceId": "ws-efdc5b42-b051-11ec-b123-4567ac120002",
    "message": {
        "log": "alerts too big, total size limit: 20000000 bytes",
        "level": "WARN"
    },
    "component": "alertmanager"
}
```

This means that Alert manager Alerts per workspace, in size quota has been exceeded.

### Action to take

Remove unnecessary annotations and labels to reduce alert size.

## Empty content warning

### When the log contains the following warning

```
{
   "workspaceId": "ws-abcd1234-ef56-78ab-cd90-1234abcd0000",
   "message": {
       "log": "Message has been modified because the content was empty."
       "level": "WARN"
   },
   "component": "alertmanager"
}
```

This means that the Alert manager template resolved the outbound alert to an empty message.

### Action to take

Validate your Alert manager template and ensure that you have a valid template for all receiver pathways.

# Invalid key/value warning

### When the log contains the following warning

```
{
   "workspaceId": "ws-abcd1234-ef56-78ab-cd90-1234abcd0000",
   "message": {
       "log": "MessageAttributes has been removed because of invalid key/value,
 numberOfRemovedAttributes=1"
       "level": "WARN"
   },
   "component": "alertmanager"
}
```

This means that some of the message attributes have been removed due to keys/values being invalid.

### Action to take

Re-evaluate the templates you are using to populate the message attributes, and ensure it is resolving to a valid SNS message attribute. For more information about validating a message to an Amazon SNS topic, see Validating SNS topic

# Message limit warning

### When the log contains the following warning

```
{
   "workspaceId": "ws-abcd1234-ef56-78ab-cd90-1234abcd0000",
   "message": {
       "log": "Message has been truncated because it exceeds size limit,
 originSize=266K, truncatedSize=12K"
       "level": "WARN"
   },
   "component": "alertmanager"
}
```

This means that some of the message size is too big.

### Action to take

Look at the Alert receiver message template and re-work it to fit within the size limit.

# No resource based policy error

### When the log contains the following error

```
{
   "workspaceId": "ws-abcd1234-ef56-78ab-cd90-1234abcd0000",
   "message": {
       "log": "Notify for alerts failed, AMP is not authorized to perform: SNS:Publish
 on resource: arn:aws:sns:us-west-2:12345:testSnsReceiver because no resource-based
 policy allows the SNS:Publish action"
       "level": "ERROR"
   },
   "component": "alertmanager"
}
```

This means that Amazon Managed Service for Prometheus does not have the permissions to submit the alert to the SNS topic specified.

### Action to take

Validate that the access policy on your Amazon SNS topic grants Amazon Managed Service for Prometheus the ability to send SNS messages to the topic. Create an SNS Access Policy giving the service `aps.amazonaws.com` (Amazon Managed Service for Prometheus) access to your Amazon SNS topic. For more information about SNS Access Policies, see Using the Access Policy Language and Example cases for Amazon SNS access control in the *Amazon Simple Notification Service Developer Guide*.

# Non ASCII warning

### When the log contains the following warning

```
{
   "workspaceId": "ws-abcd1234-ef56-78ab-cd90-1234abcd0000",
   "message": {
       "log": "Subject has been modified because it contains control or non-ASCII
 characters."
       "level": "WARN"
   },
   "component": "alertmanager"
}
```

This means that the subject has non-ASCII characters.

### Action to take

Remove references in subject field of your template to the labels that might contain non-ASCII characters.

## Not authorized to call KMS

### When the log contains the following AWS KMS error

```
{
   "workspaceId": "ws-abcd1234-ef56-78ab-cd90-1234abcd0000",
   "message": {
       "log": "Notify for alerts failed, AMP is not authorized to call KMS",
       "level": "ERROR"
   },
   "component": "alertmanager"
}
```

### Action to take

Validate that the key policy of the key used to encrypt the Amazon SNS topic allows the Amazon Managed Service for Prometheus service principal `aps.amazonaws.com` to perform the following actions: `kms:GenerateDataKey*`, and `kms:Decrypt`. For more information, see AWS KMS Permissions for SNS Topic.

## Template error

### When the log contains the following error

```
             {
  "workspaceId": "ws-efdc5b42-b051-11ec-b123-4567ac120002",
  "message": {
      "log": "Notify for alerts failed. There is an error in a receiver that is using
templates in the AlertManager definition. Make sure that the syntax is correct and
only template functions and variables that exist are used in the receiver 'default',
sns_configs position #2, section 'attributes'"
      "level": "ERROR"
  },
  "component": "alertmanager"
```

```
}
```

This means that there is an error in a template being used in the AlertManager definition. The error entry contains directions about what receiver, the position in the sns_configs and the property that contains errors.

**Action to take**

Validate your Alert Manager definition. Make sure that the syntax is correct and that you reference template variables and functions that exist. For more information, see the Notification Template Reference in the *Prometheus* open-source documentation.

# Logging and monitoring Amazon Managed Service for Prometheus workspaces

Amazon Managed Service for Prometheus uses Amazon CloudWatch to provide data about its operation. You can use CloudWatch metrics to learn about resource usage and requests to your Amazon Managed Service for Prometheus workspaces. You can turn on CloudWatch Logs support to get logs for events that happen in your workspaces.

The following topics describe using CloudWatch in more detail.

## Use CloudWatch metrics to monitor Amazon Managed Service for Prometheus resources

Amazon Managed Service for Prometheus vends usage metrics to CloudWatch. These metrics provide visibility about your workspace utilization. The vended metrics can be found in the `AWS/Usage` and `AWS/Prometheus` namespaces in CloudWatch. These metrics are available in CloudWatch for no charge. For more information about usage metrics, see CloudWatch usage metrics.

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| ResourceCount* | CreateAlertManagerAlertsTPS | `AWS/Usage` | The maximum number of `CreateAlertManagerAlerts` API operations per second, per workspace |
| ResourceCount* | DeleteAlertManagerSilencesTPS | `AWS/Usage` | The maximum number of `DeleteAlertManagerSilences` API operations per second, per workspace |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| ResourceCount* | GetAlertManagerSilenceTPS | AWS/Usage | The maximum number of GetAlertManagerSilence API operations per second, per workspace |
| ResourceCount* | GetAlertManagerStatusTPS | AWS/Usage | The maximum number of GetAlertManagerStatus API operations per second, per workspace |
| ResourceCount* | GetLabelsTPS | AWS/Usage | The maximum number of GetLabels API operations per second, per workspace |
| ResourceCount* | GetMetricMetadataTPS | AWS/Usage | The maximum number of GetMetricMetadata API operations per second, per workspace |
| ResourceCount* | GetSeriesTPS | AWS/Usage | The maximum number of GetSeries API operations per second, per workspace |
| ResourceCount | InhibitionRulesInAlertManagerDefinition | AWS/Usage | The maximum number of inhibition rules in alert manager definition file. |
| ResourceCount* | ListAlertManagerAlertGroupInfosTPS | AWS/Usage | The maximum number of ListAlertManagerAlertGroupInfos API operations per second, per workspace |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| ResourceCount* | ListAlertManagerAlertGroupsTPS | AWS/Usage | The maximum number of ListAlertManagerAlertGroups API operations per second, per workspace |
| ResourceCount* | ListAlertManagerAlertsTPS | AWS/Usage | The maximum number of ListAlertManagerAlerts API operations per second, per workspace |
| ResourceCount* | ListAlertManagerReceiversTPS | AWS/Usage | The maximum number of ListAlertManagerReceivers API operations per second, per workspace |
| ResourceCount* | ListAlertManagerSilencesTPS | AWS/Usage | The maximum number of ListAlertManagerSilences API operations per second, per workspace |
| ResourceCount* | ListAlertsTPS | AWS/Usage | The maximum number of ListAlerts API operations per second, per workspace |
| ResourceCount* | ListRulesTPS | AWS/Usage | The maximum number of ListRules API operations per second, per workspace |
| ResourceCount* | PutAlertManagerSilencesTPS | AWS/Usage | The maximum number of PutAlertManagerSilences API operations per second, per workspace |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| ResourceCount | HAReplicaGroupCount | AWS/Usage | Number of high availability replica groups |
| ResourceCount* | QueryMetricsTPS | AWS/Usage | Query operations per second |
| ResourceCount* | RemoteWriteTPS | AWS/Usage | Remote write operations per second |
| ResourceCount | ActiveAlerts | AWS/Usage | Number of active alerts per workspace<br>Units: Count<br>Valid Statistics: Average, Minimum, Maximum |
| ResourceCount | ActiveSeries | AWS/Usage | Number of active series per workspace<br>Units: Count<br>Valid Statistics: Average, Minimum, Maximum |
| ResourceCount | AlertAggregationGroupSize | AWS/Usage | The maximum size of an alert aggregation group in alert manager definition file. Each label value combination of `group_by` would create an aggregation group. |
| ResourceCount | AlertManagerDefinitionSizeBytes | AWS/Usage | The maximum size of an alert manager definition file, in bytes. |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| ResourceCount | AllSilences | AWS/Usage | Maximum number of silences, including expired, active, and pending silences, per workspace. |
| ResourceCount | IngestionRate | AWS/Usage | Sample ingestion rate<br><br>Units: Count per second<br><br>Valid Statistics: Average, Minimum, Maximum |
| ResourceCount | RuleEvaluationInterval | AWS/Usage | The minimum rule evaluation interval |
| ResourceCount | RuleGroupNamespaceDefinitionSizeBytes | AWS/Usage | The maximum size of a rule group namespace definition file, in bytes. |
| ResourceCount | TemplatesInAlertManagerDefinition | AWS/Usage | The maximum number of templates in the alert manager definition file. |
| ResourceCount | WorkspaceCount | AWS/Usage | The maximum number of workspaces per Region, per accountc. |
| ResourceCount | SizeOfAlerts | AWS/Usage | Total size of all alerts in the workspace, in bytes<br><br>Units: Bytes<br><br>Valid Statistics: Average, Minimum, Maximum |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| ResourceCount | SuppressedAlerts | AWS/Usage | Number of alerts in suppressed state per workspace. An alert can be suppressed by a silence or inhibition.<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum |
| ResourceCount | UnprocessedAlerts | AWS/Usage | Number of alerts in unprocessed state per workspace. An alert is in unprocessed state once it is received by AlertManager, but is waiting for the next aggregation group evaluation.<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum |
| ResourceCount | AllAlerts | AWS/Usage | Number of alerts in any state per workspace<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| ResourceCount | AllRules | AWS/Usage | Number of rules in any state per workspace<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum |
| ResourceCount | NativeHistogramActiveSeries | AWS/Usage | The number of unique native histogram active series per workspace<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum |
| ResourceCount | NativeHistogramIngestionRate | AWS/Usage | Native histogram sample ingestion rate per workspace per second<br><br>Units: Count per second<br><br>Valid Statistics: Average, Minimum, Maximum |
| ActiveSeriesPerLabelSet | - | AWS/Prometheus | The current active series usage for each user-defined label set<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| ActiveSeriesLimitPerLabelSet | - | AWS/Prometheus | The current active series limit value for each user-defined label set<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| AlertManagerAlertsReceived | - | AWS/Prometheus | Total successful alerts received by alert manager<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| AlertManagerNotificationsFailed | - | AWS/Prometheus | Number of failed alert deliveries<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| AlertManagerNotificationsThrottled | - | AWS/Prometheus | Number of throttled alerts<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| AnomalyDetectors | WorkspaceId | AWS/Prometheus | Total number of anomaly detectors for a given workspace<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum |
| AnomalyDetectorEvaluations | WorkspaceId, AnomalyDetectorId | AWS/Prometheus | Total number of anomaly detector evaluations<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| AnomalyDetectorEvaluationFailures | WorkspaceId, AnomalyDetectorId | AWS/Prometheus | Number of anomaly detector failures in the interval<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| AnomalyDetectorLastEvaluationDuration | WorkspaceId, AnomalyDetectorId | AWS/Prometheus | Duration of an anomaly detector's last evaluation<br><br>Units: Seconds<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| AnomalyDetectorMissedEvaluations | WorkspaceId, AnomalyDetectorId | AWS/Prometheus | Number of missed anomaly detector evaluations in the interval<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| Discarded Samples** | - | AWS/Prometheus | Number of discarded samples by reason<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| Discarded Series** | - | AWS/Prometheus | Number of series that contain a discarded sample by reason<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| DiscardedSamplesPerLabelSet | - | AWS/Prometheus | The count of discarded samples for each user-defined label set<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| DiscardedSeriesPerLabelSet | - | AWS/Prometheus | The count of series that contain a discarded sample for each user-defined label set<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| IngestionRatePerLabelSet | - | AWS/Prometheus | The ingestion rate for each user-defined label set<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| NativeHistogramIngestedBucketsRate | - | AWS/Prometheus | Rate of populated buckets received per second across all native histogram samples. Excludes rejected buckets.<br><br>Units: Count per second<br><br>Valid Statistics: Average, Minimum, Maximum |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| NativeHistogramReducedResolutionCount | - | AWS/Prometheus | Count of native histogram samples with automatically reduced bucket resolution. Resolution is reduced when a sample exceeds the maximum bucket count limit.<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| OutOfOrderIngestionRate | - | AWS/Prometheus | Out-of-order sample ingestion rate<br><br>Units: Count per second<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| OutOfOrderSampleAge*** | - | AWS/Prometheus | The difference between an out-of-order sample's timestamp and its ingestion time, which indicates how old the sample is when ingested.<br><br>Units: Seconds<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| QuerySamplesProcessed | - | AWS/Prometheus | Number of query samples processed<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| RuleEvaluations | - | AWS/Prometheus | Total number of rule evaluations<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| RuleEvaluationFailures | - | AWS/Prometheus | Number of rule evaluation failures in the interval<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |
| RuleGroupIterationsMissed | - | AWS/Prometheus | Number of Rule Group iterations missed in the interval.<br><br>Units: Count<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |

| CloudWatch metric name | Resource name | CloudWatch namespace | Description |
|---|---|---|---|
| RuleGroup LastEvaluationDuration | - | `AWS/Prometheus` | Duration of a rule group's last evaluation.<br><br>Units: Seconds<br><br>Valid Statistics: Average, Minimum, Maximum, Sum |

*TPS metrics are generated every minute and are a per-second average over that minute. Short burst periods will not be captured in the TPS metrics.

**Some of the reasons that cause samples to be discarded are as follows. Not all reasons below appear in the DiscardedSeries metric.

| Reason | Meaning |
|---|---|
| greater_than_max_sample_age | Discarding samples which are older than one hour. |
| new-value-for-timestamp | Duplicate samples are sent with the same timestamp as the previous sample but with different values. |
| per_labelset_series_limit | User has hit the total number of active series per label set limit. |
| per_metric_series_limit | User has hit the active series per metric limit. |
| per_user_series_limit | User has hit the total number of active series limit. |
| rate_limited | Ingestion rate limited. |
| sample-out-of-order | Samples are sent out of order and cannot be processed. |
| sample-too-old | Sample is older than the configured out-of-order time window and cannot be processed. |

| Reason | Meaning |
| --- | --- |
| out-of-order-rate-limit | Out-of-order ingestion rate limit was reached and the sample cannot be processed. For more information, see Amazon Managed Service for Prometheus service quotas. |
| label_value_too_long | Label value is longer than allowed character limit. |
| max_label_names_per_series | User has hit the label names per metric. |
| missing_metric_name | Metric name is not provided. |
| metric_name_invalid | Invalid metric name provided. |
| label_invalid | Invalid label provided. |
| duplicate_label_names | Duplicate label names provided. |
| native_histogram_sample_size_bytes_exceeded | Native histogram sample exceeds the maximum allowed sample size in bytes. |
| native_histogram_invalid_schema | Native histogram has an invalid schema value. Valid schemas range from -4 to 8. |
| native_histogram_invalid | Native histogram fails validation (for example, negative bucket counts, mismatched bucket counts, or malformed bucket spans). |
| native_histogram_buckets_exceeded | Native histogram exceeds the maximum bucket count limit and resolution cannot be automatically reduced. |
| native_histogram_rate_limited | Native histogram sample was rejected because the native histogram ingestion rate limit was reached. |
| per_user_native_histogram_series_limit | User has hit the native histogram active series limit per workspace. |

***The `OutOfOrderSampleAge` metric includes a `Percentile` dimension. You can use this metric to determine the appropriate out of order time window for your workspace. The valid values for the `Percentile` dimension are as follows.

| Percentile | Description |
|---|---|
| p50 | The 50th percentile age of out-of-order samples. |
| p99 | The 99th percentile age of out-of-order samples. |
| max | The maximum age of out-of-order samples. |

**Note**

A metric not existing or missing is the same as the value of that metric being 0.

**Note**

`RuleGroupIterationsMissed`, `RuleEvaluations`, `RuleEvaluationFailures`, and `RuleGroupLastEvaluationDuration` have the `RuleGroup` dimension of the following structure:
*`RuleGroupNamespace`*;*`RuleGroup`*

## Setting a CloudWatch alarm on Prometheus vended metrics

You can monitor usage of Prometheus resources using CloudWatch alarms.

**To set an alarm on the number of ActiveSeries in Prometheus**

1. Choose the **Graphed metrics** tab and scroll down to the **ActiveSeries** label.

   In the **Graphed metrics** view, only the metrics currently being ingested will appear.
2. Choose the **notification** icon in the **Actions** column.
3. In **Specify metric and conditions**, enter the threshold condition in the **Conditions value** field and choose **Next**.

4. In **Configure actions**, select an existing SNS topic or create a new SNS topic to send the notification to.
5. In **Add name and description**, add the name of the alarm and an optional description.
6. Choose **Create alarm**.

# Monitor Amazon Managed Service for Prometheus events with CloudWatch Logs

Amazon Managed Service for Prometheus logs Alert Manager and Ruler error and warning events in log groups in Amazon CloudWatch Logs. For more information about Alert Manager and Rulers, see Alert Manager topic in this guide. You can publish the workspace logs data to log streams in CloudWatch Logs. You can configure the logs that you wish to monitor in the Amazon Managed Service for Prometheus console or by using the AWS CLI. You can view or query these logs in the CloudWatch console. For more information about viewing CloudWatch Logs log streams in the console, see Working with log groups and log streams in CloudWatch in the CloudWatch user guide.

The CloudWatch free tier allows up to 5Gb of logs to be published in CloudWatch Logs. The logs that exceed the free tier allowance will be charged based on the CloudWatch pricing plan.

**Topics**

- Configuring CloudWatch Logs

## Configuring CloudWatch Logs

Amazon Managed Service for Prometheus logs Alert Manager and Ruler error and warning events in log groups in Amazon CloudWatch Logs.

You can set CloudWatch Logs logging configuration in Amazon Managed Service for Prometheus console or in the AWS CLI by calling the `create-logging-configuration` API request.

**Prerequisites**

Before calling `create-logging-configuration`, attach the following policy or equivalent permissions to the ID or role you will use to configure CloudWatch Logs.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups",
                "aps:CreateLoggingConfiguration",
                "aps:UpdateLoggingConfiguration",
                "aps:DescribeLoggingConfiguration",
                "aps:DeleteLoggingConfiguration"
            ],
            "Resource": "*"
        }
    ]
}
```

**To configure CloudWatch Logs**

You can configure logging in Amazon Managed Service for Prometheus using either the AWS console or the AWS CLI.

Console

To configure logging in Amazon Managed Service for Prometheus console

1. Navigate to the **Logs** tab in your workspace details panel.
2. Choose **Manage logs** on the upper right side of the **Logs** panel.
3. Choose **all** in the **Log level** dropdown list.
4. Choose the log group that you want to publish your logs to in the **Log Group** dropdown list.

You can also create a new log group in CloudWatch console.

5. Choose **Save changes**.

AWS CLI

You can set logging configuration using the AWS CLI.

To configure logging using the AWS CLI

- Using the AWS CLI, run the following command.

```
aws amp create-logging-configuration --workspace-id my_workspace_ID
                                --log-group-arn my-log-group-arn
```

## Limitations

- **Not all events logged**

  Amazon Managed Service for Prometheus only logs events that are at the `warning` or `error` level.

- **Policy size limits**

  CloudWatch Logs resource policies are limited to 5120 characters. When CloudWatch Logs detect that a policy approaches this size limit, it automatically enables log groups that start with `/aws/vendedlogs/`.

  When you create an alert rule with logging enabled, Amazon Managed Service for Prometheus must update your CloudWatch Logs resource policy with the log group you specify. To avoid reaching the CloudWatch Logs resource policy size limit, prefix your CloudWatch Logs log group names with `/aws/vendedlogs/`. When you create a log group in the Amazon Managed Service for Prometheus console, the log group names are prefixed with `/aws/vendedlogs/`. For more information, see Enabling Logging from Certain AWS Services in the CloudWatch Logs User Guide.

# Managing the query cost in Amazon Managed Service for Prometheus

Amazon Managed Service for Prometheus offers the ability to limit query cost by providing limits on how much Query Samples Processed (QSP) can be used by a single query. You can configure two types of thresholds for QSP, *warning* and *error* to help manage and control query costs effectively.

When queries hit the *warning* threshold, a warning message appears in the API query response. For queries viewed through Amazon Managed Grafana, the warning will be visible in the Amazon Managed Grafana UI, helping users identify expensive queries. Queries that hit the *error* threshold are not charged and will be rejected with an error.

In addition to query throttling, Amazon Managed Service for Prometheus offers the ability to log query performance data to CloudWatch Logs. This feature allows you to analyze queries in detail, helping you optimize your Amazon Managed Service for Prometheus queries and manage costs more effectively. Query logging captures information about queries that exceed specified Query Samples Processed (QSP) thresholds. This data is then published to CloudWatch Logs, enabling you to investigate and analyze query performance. Logged queries include both API queries and Rule queries. By default, query logging is disabled to minimize unnecessary CloudWatch Logs usage. You can enable this feature when needed for query analysis.

**Topics**


- Configuring query logging
- Configuring query throttling thresholds
- Log content
- Limitations


## Configuring query logging

You can configure query logging in Amazon Managed Service for Prometheus console or in the AWS CLI by calling the `create-query-logging-configuration` API request. This API body contains list of destinations, but for now, we only support CloudWatch Logs as a destination and destinations should contain exactly one element with CloudWatch configurations.

## Prerequisites

Make sure the `logGroup` is already created. The ID or role used to configure should have the following policy or equivalent permissions.

JSON

```
{
"Version":"2012-10-17",
    "Statement": [
        {
"Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups",
                "aps:CreateQueryLoggingConfiguration",
                "aps:UpdateQueryLoggingConfiguration",
                "aps:DescribeQueryLoggingConfiguration",
                "aps:DeleteQueryLoggingConfiguration"
            ],
            "Resource": "*"
        }
    ]
}
```

## Configure CloudWatch Logs

You can configure CloudWatch Logs by logging into Amazon Managed Service for Prometheus using either the AWS Management Console or the AWS CLI.

**To configure query logging using Amazon Managed Service for Prometheus console**

1. Navigate to the **Logs** tab in your workspace details panel.
2. Under **Query Insights**, choose **Create**.

3. Select the **Log Group** drop down and choose the log group to publish your logs.

   You can also create a new log group in the CloudWatch console.
4. Enter the **Threshold (QSP)**.
5. Choose **Save**.

**To configure query logging using the AWS CLI use the command**

```
aws amp create-query-logging-configuration \
--workspace-id my_workspace_ID \
--destinations '[{"cloudWatchLogs":{"logGroupArn":"$my-log-group-arn"},"filters":
{"qspThreshold":$qspThreshold}}]'
```

For information on how to update, delete, and describe operations, see Amazon Managed Service for Prometheus API Reference.

## Configuring query throttling thresholds

To configure QSP thresholds, you must provide the query parameters in the QueryMetrics API.

- max_samples_processed_warning_threshold – Sets the warning threshold for query samples processed
- max_samples_processed_error_threshold – Sets the error threshold for query samples processed

For Amazon Managed Grafana users, you can use grafana data source configuration to apply limits to all the queries from the datasource:

1. Browse to the Amazon Managed Service for Prometheus data source configuration in Amazon Managed Grafana.
2. Under **Custom query parameters**, add the threshold headers.
3. Choose **Save**.

## Log content

For queries that originate from rules, you will see the following information about the query in the CloudWatch Logs:

```
{
    workspaceId: "workspace_id",
    message: {
        query: "avg(rate(go_goroutines[1m])) > 1",
        name: "alert_rule",
        kind: "alerting",
        group: "test-alert",
        namespace: "test",
        samples: "59321",
    },
    component: "ruler"
}
```

For queries that originate from API calls, you will see the following information about the query in the CloudWatch Logs:

```
{
    workspaceId: "ws-5e7658c2-7ccf-4c30-9de9-2ab26fa30639",
    message: {
        query: "sum by (instance) (go_memstats_alloc_bytes{job=\"node\"})",
        queryType: "range",
        start: "1683308700000",
        end: "1683913500000",
        step: "300000",
        samples: "11496",
        userAgent: "AWSPrometheusDPJavaClient/2.0.436.0 ",
        dashboardUid: "11234",
        panelId: "12"
    },
    component: "query-frontend"
}
```

## Limitations

**Policy size limits** – CloudWatch Logs resource policies are limited to 5120 characters. When CloudWatch Logs detects that the policy is approaching the size limit, it automatically enables log groups that start with `/aws/vendedlogs/`. When you enable query logging, Amazon Managed Service for Prometheus must update your CloudWatch Logs resource policy with the log group you specify. To avoid reaching the CloudWatch Logs resource policy size limit, prefix your CloudWatch Logs log group names with `/aws/vendedlogs/`.

# Understand and optimize costs in Amazon Managed Service for Prometheus

The following frequently asked questions and their answers may be helpful in understanding and optimizing costs associated with Amazon Managed Service for Prometheus.

## What contributes to my costs?

For most customers, metric *ingestion* contributes the majority of costs. Customers with high query usage will also see some cost based on *query samples processed*, with *metrics storage* being a small driver of overall costs. For more information about the prices for each of these, see Pricing in the *Amazon Managed Service for Prometheus product page*.

## What is the best way to lower my costs? How do I lower ingestion costs?

Ingestion rates (not storage of the metrics) is the majority of costs for most customers. You can reduce ingestion rates by reducing the collection frequency (increasing the collection interval) or by reducing the number of active series ingested.

You can increase the collection (scraping) interval from your collection agent: Both the Prometheus server (running in Agent mode) and the AWS Distro for OpenTelemetry (ADOT) collector support the `scrape_interval` configuration. For example, increasing the collection interval from 30 seconds to 60 seconds will reduce your ingestion usage by half.

You can also filter the metrics sent to Amazon Managed Service for Prometheus by using the `<relabel_config>`. For more information about relabeling in the Prometheus agent configuration, see https://prometheus.io/docs/prometheus/latest/configuration/configuration/#relabel_config in the Prometheus documentation.

## What is the best way to lower my query costs?

Query costs are based on the number of samples processed. You can reduce the frequency of queries to reduce your query costs.

To get more visibility into the queries that are contributing the most to your query costs, see Managing the query cost in Amazon Managed Service for Prometheus.

## If I decrease the retention period of my metrics, will that help reduce my total bill?

You can reduce your retention period, however, this is unlikely to substantially reduce your costs.

For information about how to configure the retention period of a workspace, see Configure your workspace.

## How can I keep my alert query costs low?

Alerting creates queries against your data, which add to your query costs. Here are some strategies that you can use to optimize your alert queries, and keep your costs lower.

- **Use Amazon Managed Service for Prometheus alerting** – Alerting systems external to Amazon Managed Service for Prometheus may require additional queries to add resiliency or high availability, as the external service queries the metrics from multiple availability zones or regions. This includes alerting in Grafana for high availability. This can multiply your cost by three times or more. The alerting in Amazon Managed Service for Prometheus is optimized and will give you high availability and resiliency with the fewest number of queries.

  We recommend using the native alerting in Amazon Managed Service for Prometheus rather than external alerting systems.
- **Optimize your alert interval** – One quick way to optimize your alert queries is to increase the auto-refresh interval. If you have an alert that queries every minute, but is only needed every five minutes, increasing the auto-refresh interval could save you five times your query costs for that alert.
- **Use an optimal lookback** – A larger lookback window in your query increases the costs of the query, as it pulls more data. Ensure that the lookback window in your PromQL query is reasonably sized for the data you need to alert. For example, in the following rule, the expression includes a ten minute lookback window:

  ```
  - alert: metric:alerting_rule
    expr: avg(rate(container_cpu_usage_seconds_total[10m])) > 0
    for: 2m
  ```

  Changing the `expr` to `avg(rate(container_cpu_usage_seconds_total[5m])) > 0` can help to reduce your query costs.

In general, look at your alerting rules and make sure that you are alerting on the best metrics for your service. It's easy to create overlapping alerts on the same metrics or multiple alerts that give you the same information, especially as you add alerts over time. If you find that you often see groups of alerts happening at the same time, it's possible that you can optimize your alerts and not include all of them.

These suggestions can help you to reduce costs. Ultimately, you must balance the costs with creating the right set of alerts for understanding the state of your system.

For more information about alerting in Amazon Managed Service for Prometheus, see Managing and forwarding alerts in Amazon Managed Service for Prometheus with alert manager.

## Can I check my bill at any time?

The AWS Cost and Usage Report tracks your AWS usage and provides estimated charges associated with your account within a billing period. For more information, see What are AWS Cost and Usage Reports? in the *AWS Cost and Usage Reports User Guide*

## What metrics can I use to monitor my costs?

Metric samples you ingest are the primary cost driver for Amazon Managed Service for Prometheus. The number of samples ingested directly determines your monthly charges, making it essential to monitor and understand your ingestion patterns.

AWS Cost Explorer is the source of truth for monitoring your Amazon Managed Service for Prometheus costs. You can monitor Cost Explorer for historical and day-by-day trends for cost on Amazon Managed Service for Prometheus across multiple dimensions including ingested samples. AWS Cost Anomaly Detection can also provide you the ability to monitor unexpected changes in your spending patterns.

Using `IngestionRate` metrics provides an auxiliary method to monitor trends in ingestion that are directly correlated to cost. The advantages of using `IngestionRate` as an additional metric include:

- **Workspace-level tracking** – Monitor ingestion on a per-workspace basis rather than only at the account level.
- **Granular visibility** – Track ingestion patterns on an hourly or even minute-by-minute basis for real-time insights.

- **Proactive monitoring** – Set CloudWatch alarms to detect usage spikes before they appear in billing.

> **Note**
>
> `IngestionRate` can be used to estimate costs and trends or attribute cost per workspace, but it is not 100% accurate. Because `IngestionRate` reports an average rate sampled at 1-minute intervals, multiplying this rate by time provides an approximation rather than an exact count of samples ingested. Additionally, Amazon CloudWatch's data retention policy affects the granularity available for historical queries, with data older than 63 days limited to 1-hour intervals.

For more information about monitoring Amazon Managed Service for Prometheus metrics in CloudWatch, see Use CloudWatch metrics to monitor Amazon Managed Service for Prometheus resources.

**Monitoring native histogram costs** – Native histograms are metered based on populated buckets: each populated bucket (a bucket with non-zero observations) counts as 0.25 of a metric sample, and empty buckets are not metered. For example, a native histogram with 10 populated buckets is metered as 2.5 metric samples. To monitor your native histogram cost drivers, you can use the `NativeHistogramIngestedBucketsRate` metric to monitor the rate of populated buckets being ingested. You can also use `NativeHistogramIngestionRate` to track the overall native histogram sample ingestion rate and `NativeHistogramActiveSeries` to monitor the number of active native histogram series in your workspace.

## How do I view my costs in AWS Cost Explorer?

As your source of truth for Amazon Managed Service for Prometheus costs, AWS Cost Explorer provides your actual billed usage and charges for Amazon Managed Service for Prometheus samples ingested, including historical billing data by month and region. Use Cost Explorer for your final billed amounts and day-by-day cost trends.

To view your Amazon Managed Service for Prometheus costs:

**Access AWS Cost Explorer**

1. Sign in to the AWS Management Console.

2. Navigate to the Billing and Cost Management dashboard.
3. Select **Cost Explorer** from the left navigation menu.
4. Choose **Launch Cost Explorer** (if this is your first time using it).

#### Configure the report

1. Set your time range to the desired billing period (for example, March 2025 - February 2026).
2. Under **Filters**, select:
   - **Service**: Choose "Amazon Managed Service for Prometheus".
   - **Usage Type**: Filter for "MetricSampleCount" to isolate sample ingestion charges.

#### Group and view data

1. Under **Group by**, select **Region** to view cost and usage data per region.
2. Choose your preferred visualization (bar chart, line chart, or table).
3. Choose **Apply** to generate the report.

#### Export data (optional)

1. Choose **Download CSV** in the top right corner to export the data.
2. The CSV file will contain: billing period, region, usage type, billed amount, and usage quantity (number of samples billed).

> **Note**
>
> Cost Explorer data typically has a 24-hour delay. For the most current billing period, data may not be available until the following day.

## How do I calculate the number of samples ingested in a month?

You can calculate the approximate number of samples ingested using Amazon CloudWatch's `IngestionRate` metrics with the AWS Command Line Interface. This is useful for reviewing monthly bills and understanding usage patterns across workspaces.

To retrieve ingestion data:

```
aws cloudwatch get-metric-data \
    --region your-region \
    --start-time start-timestamp \
    --end-time end-timestamp \
    --metric-data-queries '[
        {
            "Id": "e1",
            "Expression": "SUM(METRICS())",
            "Period": 3600
        },
        {
            "Id": "ws1",
            "MetricStat": {
                "Metric": {
                    "Namespace": "AWS/Usage",
                    "MetricName": "ResourceCount",
                    "Dimensions": [
                        {"Name": "Service",    "Value": "Prometheus"},
                        {"Name": "Resource",   "Value": "IngestionRate"},
                        {"Name": "Type",       "Value": "Resource"},
                        {"Name": "Class",      "Value": "None"},
                        {"Name": "ResourceId", "Value": "YOUR_AMP_WORKSPACE_ID"}
                    ]
                },
                "Period": 3600,
                "Stat": "Average"
            }
        }
    ]'
```

The command returns hourly average `IngestionRate` values, measured in samples per second. To calculate the approximate number of samples ingested in a month, multiply each hourly data point by 3600 (seconds per hour) to get the samples ingested in that hour, then sum all hourly totals across the month:

```
Monthly samples ≈ Σ (hourly IngestionRate average × 3600)
```

For example, if a single hour returns an average `IngestionRate` of 500 samples per second, that hour contributed approximately 500 × 3600 = 1,800,000 samples. Repeat this for every hour in the month and sum the results to get your approximate monthly ingestion count.

Key parameters:

- `Period`: 3600 (1 hour in seconds)
- `StartTime`: Beginning of your month (for example, `2026-02-01T00:00:00Z`)
- `EndTime`: End of your month (for example, `2026-03-01T00:00:00Z`)
- `Stat`: Average

To find your workspace IDs:

```
aws amp list-workspaces --region your-region
```

Use the workspace ID to filter the metrics to show data only for the specified workspace, rather than aggregating across all Prometheus resources in the region.

## What data granularity is available for historical cost analysis?

Amazon CloudWatch's data retention policy affects the granularity available for historical queries:

- **Data less than 15 days old**: Query at 1-minute intervals (`Period`: 60)
- **Data 15–63 days old**: Query at 5-minute intervals (`Period`: 300)
- **Data older than 63 days**: Limited to 1-hour intervals (`Period`: 3600)

For historical analysis beyond 63 days, CloudWatch automatically downsamples data to a minimum 1-hour period. When reviewing billing for months older than 63 days, you must use hourly aggregated data. The monthly sample calculation uses these hourly-averaged data points, summing each value multiplied by 3600 across the entire month.

This reduced granularity further contributes to why `IngestionRate` provides estimates rather than exact counts for older data—always refer to Cost Explorer for your authoritative billed amounts.

For more details on CloudWatch metric retention, see Metrics retention in the *Amazon CloudWatch User Guide*.

# What are best practices for monitoring Amazon Managed Service for Prometheus costs?

To effectively manage and optimize your Amazon Managed Service for Prometheus spending, consider implementing the following monitoring practices:

- Monitor Cost Explorer regularly to track actual spending trends and identify cost anomalies across multiple dimensions including ingested samples.
- Enable AWS Cost Anomaly Detection to receive alerts for unexpected cost increases in your Amazon Managed Service for Prometheus spending.
- Set up CloudWatch alarms on `IngestionRate` for workspace-level monitoring and early detection of ingestion spikes.
- Export Cost Explorer data regularly for long-term cost analysis and reporting.

# Why is my bill higher at the beginning of the month than at the end of the month?

Amazon Managed Service for Prometheus has a tiered pricing model for ingestion, which results in costs in your initial usage being higher. As your usage reaches higher ingest tiers, with lower costs, your costs are lower. For more information about pricing, including ingest tiers, see Pricing in the *Amazon Managed Service for Prometheus product page*.

**Note**

- Tiers are for usage *within a region*, not across regions. Usage within a region must reach the next tier to use the lower rate.
- In an organization in AWS Organizations, tier usage is tallied *per payer account*, not per account (the payer account is always the organization management account). When the total ingested metrics (within a region) for *all accounts in an organization* reaches the next tier, all accounts are charged the lower rate.

# I deleted all my Amazon Managed Service for Prometheus workspaces, but I still seem to be getting charged. What might be happening?

One possibility in this case is that you still have AWS managed scrapers that are setup to send metrics to your deleted workspaces. Follow the instructions to Find and delete scrapers.

# Integrating with other AWS services

Amazon Managed Service for Prometheus integrates with other AWS services. This section describes integrating with Amazon Elastic Kubernetes Service (Amazon EKS) cost monitoring (with Kubecost), and how to ingest metrics from CloudWatch using Amazon Data Firehose. It also describes setting up and managing Amazon Managed Service for Prometheus with AWS Observability Accelerator Terraform modules, or by using AWS Controllers for Kubernetes.

**Topics**


- Integrating with Amazon EKS cost monitoring
- Set up Amazon Managed Service for Prometheus with AWS Observability Accelerator
- Manage Amazon Managed Service for Prometheus with AWS Controllers for Kubernetes
- Integrating CloudWatch metrics with Amazon Managed Service for Prometheus


# Integrating with Amazon EKS cost monitoring

Amazon Managed Service for Prometheus integrates with Amazon Elastic Kubernetes Service (Amazon EKS) cost monitoring (with Kubecost) to perform cost allocation calculations and provide insights into optimizing your Kubernetes clusters. Using Amazon Managed Service for Prometheus with Kubecost, you can reliably scale your cost monitoring to support larger clusters.

Integrating with Kubecost gives you granular visibility into your Amazon EKS cluster costs. You can aggregate costs by the majority of Kubernetes contexts, from the container level up to the cluster level, and even multi-cluster level. You can generate reports across containers or clusters to track costs for show back or chargeback purposes.

The following give instructions for integrating with Kubecost in a single- or multi-cluster scenario:

- **Single-cluster integration** – To learn how to integrate Amazon EKS cost monitoring with a single cluster, see the AWS blog post Integrating Kubecost with Amazon Managed Service for Prometheus.
- **Multi-cluster integration** – To learn how to integrate Amazon EKS cost monitoring with a multiple clusters, see the AWS blog post Multi-cluster cost monitoring for Amazon EKS using Kubecost and Amazon Managed Service for Prometheus.

> **Note**
>
> For more information about using Kubecost, see Cost monitoring in the *Amazon EKS User Guide*.

# Set up Amazon Managed Service for Prometheus with AWS Observability Accelerator

AWS provides observability tools, including monitoring, logging, alerting, and dashboards, for your Amazon Elastic Kubernetes Service (Amazon EKS) projects. This includes Amazon Managed Service for Prometheus, Amazon Managed Grafana, AWS Distro for OpenTelemetry, and other tools. To help you use these tools together, AWS provides Terraform modules that configure observability with these services, called the AWS Observability Accelerator.

AWS Observability Accelerator provides two collector profiles for Amazon Managed Service for Prometheus:

- **Managed metrics (agentless)** – Uses the Amazon Managed Service for Prometheus collector, a fully managed, agentless scraper that runs outside your cluster. No collector pods to manage. Metrics only.
- **Self-managed** – Deploys an OpenTelemetry Collector via Helm in your cluster. Supports metrics, traces (AWS X-Ray), and logs (Amazon CloudWatch).

This section walks through both options, starting with the recommended agentless approach.

The Terraform templates and detailed instructions can be found on the AWS Observability Accelerator for Terraform GitHub page.

## Prerequisites

To use AWS Observability Accelerator, you must have an existing Amazon EKS cluster, and the following prerequisites:

- AWS CLI – used to call AWS functionality from the command line.
- kubectl – used to control your EKS cluster from the command line.

- Terraform (>= 1.5.0) – used to automate creation of the resources for this solution. You must have the AWS provider set up with an IAM role that has access to create and manage Amazon Managed Service for Prometheus, Amazon Managed Grafana, and IAM within your AWS account. For more information about how to configure the AWS provider for Terraform, see AWS provider in the *Terraform documentation*.

## Using the managed metrics (agentless) example

This example uses the Amazon Managed Service for Prometheus collector to scrape Prometheus metrics from your Amazon EKS cluster without deploying any collector pods. The collector requires at least two subnets in two distinct Availability Zones. For more details, see the eks-amp-managed example on GitHub.

**To use the agentless infrastructure monitoring Terraform module**

1. From the folder you want to create your project in, clone the repo using the following command.

```
git clone https://github.com/aws-observability/terraform-aws-observability-
accelerator.git
```

2. Initialize Terraform with the following commands.

```
cd examples/eks-amp-managed

terraform init
```

3. Create a new `terraform.tfvars` file, as in the following example. Use the AWS Region, cluster ID, and VPC networking details for your Amazon EKS cluster. The collector requires at least two subnets in two distinct Availability Zones.

```
# (mandatory) AWS Region where your resources will be located
aws_region = "eu-west-1"

# (mandatory) EKS Cluster name
eks_cluster_id = "my-eks-cluster"

# (mandatory) Subnets for the managed scraper (>= 2 AZs)
scraper_subnet_ids = ["subnet-aaa", "subnet-bbb"]
```

```
# (mandatory) Security group allowing scraper access to the EKS API
scraper_security_group_ids = ["sg-xxx"]
```

4. Create an Amazon Managed Grafana workspace, if you don't already have one that you want to use. For information about how to create a new workspace, see Create your first workspace in the *Amazon Managed Grafana User Guide.*

5. Create two variables for Terraform to use your Grafana workspace by running the following commands at the command line. You will need to replace the *grafana-workspace-id* with the ID from your Grafana workspace.

```
export TF_VAR_managed_grafana_workspace_id=grafana-workspace-id
export TF_VAR_grafana_api_key=`aws grafana create-workspace-api-key --key-name
 "observability-accelerator-$(date +%s)" --key-role ADMIN --seconds-to-live 1200 --
workspace-id $TF_VAR_managed_grafana_workspace_id --query key --output text`
```

6. [Optional] To use an existing Amazon Managed Service for Prometheus workspace, add the ID to the `terraform.tfvars` file, as in the following example, replacing the *prometheus-workspace-id* with your Prometheus workspace ID. If you do not specify an existing workspace, then a new Prometheus workspace will be created for you.

```
# (optional) Leave it empty for a new workspace to be created
managed_prometheus_workspace_id = "prometheus-workspace-id"
```

7. Deploy the solution with the following command.

```
terraform apply -var-file=terraform.tfvars
```

This will create resources in your AWS account, including the following:

- A new Amazon Managed Service for Prometheus workspace (unless you opted to use an existing workspace).
- An Amazon Managed Service for Prometheus collector (agentless scraper) configured to scrape Prometheus metrics from your Amazon EKS cluster.
- Prometheus recording and alerting rules in your Amazon Managed Service for Prometheus workspace.
- kube-state-metrics and node-exporter deployed in your Amazon EKS cluster for infrastructure metrics.

- New Amazon Managed Grafana data source and dashboards in your current workspace. The dashboards will be listed under **EKS Monitoring**.

## Alternative: Self-managed OpenTelemetry Collector

If you need traces, logs, or full control over the collection pipeline, use the self-managed profile. This deploys an OpenTelemetry Collector via Helm in your Amazon EKS cluster, configured to scrape Prometheus metrics and remote-write to Amazon Managed Service for Prometheus. It also supports traces (AWS X-Ray) and logs (Amazon CloudWatch). For more details, see the eks-amp-otel example on GitHub.

**To use the self-managed Terraform module**

1. Clone the repo and initialize Terraform.

```
git clone https://github.com/aws-observability/terraform-aws-observability-
accelerator.git
cd examples/eks-amp-otel
terraform init
```

2. Create a new `terraform.tfvars` file, as in the following example.

```
# (mandatory) AWS Region where your resources will be located
aws_region = "eu-west-1"

# (mandatory) EKS Cluster name
eks_cluster_id = "my-eks-cluster"
```

3. Set up your Amazon Managed Grafana workspace and API key using the same steps as the managed metrics example (steps 4–6 above).
4. Deploy the solution with the following command.

```
terraform apply -var-file=terraform.tfvars
```

This will create the following resources in your AWS account (unlike the agentless approach, the collector runs inside your cluster):

- An Amazon Managed Service for Prometheus workspace (if not provided).

- An Amazon Managed Grafana workspace with data source and dashboards.
- An OpenTelemetry Collector deployed via Helm in your Amazon EKS cluster, configured to scrape Prometheus metrics and remote-write to Amazon Managed Service for Prometheus.
- An IAM role for service accounts (IRSA) for the OpenTelemetry Collector.
- Traces pipeline to AWS X-Ray (enabled by default).
- Logs pipeline to Amazon CloudWatch (enabled by default).

### Viewing dashboards

To view your new dashboards, open the specific dashboard in your Amazon Managed Grafana workspace. The infrastructure dashboards are provisioned automatically by Terraform. For more information about using Amazon Managed Grafana, see Working in your Grafana workspace, in the *Amazon Managed Grafana User Guide*.

## Manage Amazon Managed Service for Prometheus with AWS Controllers for Kubernetes

Amazon Managed Service for Prometheus is integrated with AWS Controllers for Kubernetes (ACK), with support for managing your workspace, Alert Manager, and Ruler resources in Amazon EKS. You can use AWS Controllers for Kubernetes custom resource definitions (CRDs) and native Kubernetes objects without having to define any resources outside of your cluster.

This section describes how to set up AWS Controllers for Kubernetes and Amazon Managed Service for Prometheus in an existing Amazon EKS cluster.

You can also read the blog posts introducing AWS Controllers for Kubernetes and introducing the ACK controller for Amazon Managed Service for Prometheus.

### Prerequisites

Before starting to integrate AWS Controllers for Kubernetes and Amazon Managed Service for Prometheus with your Amazon EKS cluster, you must have the following prerequisites.

- You must have an existing AWS account and permissions to create Amazon Managed Service for Prometheus and IAM roles programmatically.

- You must have an existing Amazon EKS cluster with OpenID Connect (OIDC) enabled.

  If you do not have OIDC enabled, you can use the following command to enable it. Remember to replace the *YOUR_CLUSTER_NAME* and *AWS_REGION* with the correct values for your account.

  ```
  eksctl utils associate-iam-oidc-provider \
      --cluster ${YOUR_CLUSTER_NAME} --region ${AWS_REGION} \
      --approve
  ```

  For more information about using OIDC with Amazon EKS, see OIDC identity provider authentication and Creating an IAM OIDC provider in the *Amazon EKS User Guide*.
- You must have the Amazon EBS CSI driver installed in your Amazon EKS cluster.
- You must have the AWS CLI installed. The AWS CLI is used to call AWS functionality from the command line.
- Helm, the package manager for Kubernetes, must be installed.
- Control plane metrics with Prometheus must be set up in your Amazon EKS cluster.
- You must have an Amazon Simple Notification Service (Amazon SNS) topic where you want to send alerts from your new workspace. Make sure that you have given Amazon Managed Service for Prometheus permission to send messages to the topic.

When your Amazon EKS cluster is configured appropriately, you should be able to see metrics formatted for Prometheus by calling `kubectl get --raw /metrics`. Now you are ready to install an AWS Controllers for Kubernetes service controller and use it to deploy Amazon Managed Service for Prometheus resources.

## Deploying a workspace with AWS Controllers for Kubernetes

To deploy a new Amazon Managed Service for Prometheus workspace, you will install an AWS Controllers for Kubernetes controller, and then use that to create the workspace.

**To deploy a new Amazon Managed Service for Prometheus workspace with AWS Controllers for Kubernetes**

1. Use the following commands to use Helm to install the Amazon Managed Service for Prometheus service controller. For more information see Install an ACK Controller in the AWS Controllers for Kubernetes documentation on GitHub. Use the correct *region* for your system, such as `us-east-1`.

```
export SERVICE=prometheusservice
export RELEASE_VERSION=`curl -sL https://api.github.com/repos/aws-controllers-k8s/
$SERVICE-controller/releases/latest | jq -r '.tag_name | ltrimstr("v")'`
export ACK_SYSTEM_NAMESPACE=ack-system
export AWS_REGION=region

aws ecr-public get-login-password --region us-east-1 | helm registry login --
username AWS --password-stdin public.ecr.aws
helm install --create-namespace -n $ACK_SYSTEM_NAMESPACE ack-$SERVICE-controller \
  oci://public.ecr.aws/aws-controllers-k8s/$SERVICE-chart --version=
$RELEASE_VERSION --set=aws.region=$AWS_REGION
```

After a few moments, you should see a response similar to the following indicating success.

```
You are now able to create Amazon Managed Service for Prometheus (AMP) resources!
The controller is running in "cluster" mode.
The controller is configured to manage AWS resources in region: "us-east-1"
```

You can optionally verify that the AWS Controllers for Kubernetes controller has been successfully installed with the following command.

```
helm list --namespace $ACK_SYSTEM_NAMESPACE -o yaml
```

This will return information about the controller `ack-prometheusservice-controller`, including the `status: deployed`.

2. Create a file called `workspace.yaml` with the following text. This will be used as configuration for the workspace you are creating.

```
apiVersion: prometheusservice.services.k8s.aws/v1alpha1
kind: Workspace
metadata:
  name: my-amp-workspace
spec:
  alias: my-amp-workspace
  tags:
    ClusterName: EKS-demo
```

3. Run the following command to create your workspace (this command depends on the system variables that you set up in step 1).

```
kubectl apply -f workspace.yaml -n $ACK_SYSTEM_NAMESPACE
```

Within a few moments, you should be able to see a new workspace, called `my-amp-workspace` in your account.

Running the following command to view the details and status of your workspace including the *workspace ID*. Alternately, you can view the new workspace in the Amazon Managed Service for Prometheus console.

```
kubectl describe workspace my-amp-workspace -n $ACK_SYSTEM_NAMESPACE
```

> **Note**
>
> You can also use an existing workspace rather than create a new one.

4. Create two new yaml files as configuration for the Rulegroups and AlertManager that you will create next using the following configuration.

   Save this configuration as `rulegroup.yaml`. Replace *WORKSPACE-ID* with the workspace ID from the previous step.

```
apiVersion: prometheusservice.services.k8s.aws/v1alpha1
kind: RuleGroupsNamespace
metadata:
  name: default-rule
spec:
  workspaceID: WORKSPACE-ID
  name: default-rule
  configuration: |
    groups:
    - name: example
      rules:
      - alert: HostHighCpuLoad
        expr: 100 - (avg(rate(node_cpu_seconds_total{mode="idle"}[2m])) * 100) > 60
        for: 5m
        labels:
          severity: warning
          event_type: scale_up
        annotations:
```

```
          summary: Host high CPU load (instance {{ $labels.instance }})
          description: "CPU load is > 60%\n  VALUE = {{ $value }}\n  LABELS =
 {{ $labels }}"
      - alert: HostLowCpuLoad
        expr: 100 - (avg(rate(node_cpu_seconds_total{mode="idle"}[2m])) * 100) < 30
        for: 5m
        labels:
          severity: warning
          event_type: scale_down
        annotations:
          summary: Host low CPU load (instance {{ $labels.instance }})
          description: "CPU load is < 30%\n  VALUE = {{ $value }}\n  LABELS =
 {{ $labels }}"
```

Save the following configuration as `alertmanager.yaml`. Replace *WORKSPACE-ID* with the workspace ID from the previous step. Replace *TOPIC-ARN* with the ARN for the Amazon SNS topic to send notifications to, and *REGION* with the AWS Region you are using. Remember that Amazon Managed Service for Prometheus must have permissions to the Amazon SNS topic.

```
apiVersion: prometheusservice.services.k8s.aws/v1alpha1
kind: AlertManagerDefinition
metadata:
  name: alert-manager
spec:
  workspaceID: WORKSPACE-ID
  configuration: |
    alertmanager_config: |
      route:
         receiver: default_receiver
      receivers:
        - name: default_receiver
          sns_configs:
          - topic_arn: TOPIC-ARN
            sigv4:
              region: REGION
            message: |
              alert_type: {{ .CommonLabels.alertname }}
              event_type: {{ .CommonLabels.event_type }}
```

> **Note**
>
> To learn more about the formats of these configuration files, see RuleGroupsNamespaceData and AlertManagerDefinitionData.

5. Run the following commands to create your rule group and alert manager configuration (this command depends on the system variables that you set up in step 1).

```
kubectl apply -f rulegroup.yaml -n $ACK_SYSTEM_NAMESPACE
kubectl apply -f alertmanager.yaml -n $ACK_SYSTEM_NAMESPACE
```

The changes will be available within a few moments.

> **Note**
>
> To update a resource, rather than create it, you simply update the yaml file, and run the `kubectl apply` command again.
> To delete a resource, run the following command. Replace *ResourceType* with the type of resource you want to delete `Workspace`, `AlertManagerDefinition`, or `RuleGroupNamespace`. Replace *ResourceName* with the name of the resource to delete.
>
> ```
> kubectl delete ResourceType ResourceName -n $ACK_SYSTEM_NAMESPACE
> ```

That completes deploying the new workspace. The next section describes configuring your cluster to send metrics to that workspace.

## Configuring your Amazon EKS cluster to write to the Amazon Managed Service for Prometheus workspace

This section describes how to use Helm to configure the Prometheus running in your Amazon EKS cluster to remote write metrics to the Amazon Managed Service for Prometheus workspace that you created in the previous section.

For this procedure, you will need the name of the IAM role you have created to use for ingesting metrics. If you have not done this already, see Set up service roles for the ingestion of metrics from

Amazon EKS clusters for more information and instructions. If you follow those instructions, the IAM role will be called `amp-iamproxy-ingest-role`.

**To configure your Amazon EKS cluster for remote write**

1. Use the following command to get the `prometheusEndpoint` for your workspace. Replace *WORKSPACE-ID* with the workspace ID from the previous section.

   ```
   aws amp describe-workspace --workspace-id WORKSPACE-ID
   ```

   The prometheusEndpoint will be in the return results, and be formatted like this:

   ```
   https://aps-workspaces.us-west-2.amazonaws.com/workspaces/ws-a1b2c3d4-a123-b456-
   c789-ac1234567890/
   ```

   Save this URL for use in the next few steps.

2. Create a new file with the following text and call it `prometheus-config.yaml`. Replace *account* with your account ID, *workspaceURL/* with the URL you just found, and *region* with the appropriate AWS Region for your system.

   ```
   serviceAccounts:
           server:
               name: "amp-iamproxy-ingest-service-account"
               annotations:
                   eks.amazonaws.com/role-arn: "arn:aws:iam::account:role/amp-
   iamproxy-ingest-role"
   server:
       remoteWrite:
           - url: workspaceURL/api/v1/remote_write
             sigv4:
               region: region
             queue_config:
               max_samples_per_send: 1000
               max_shards: 200
               capacity: 2500
   ```

3. Find the Prometheus chart and namespace names as well as the chart version with the following Helm command.

   ```
   helm ls --all-namespaces
   ```

Based on the steps so far, the Prometheus chart and namespace should both be named `prometheus`, and the chart version may be `15.2.0`

4. Run the following command, using the *PrometheusChartName*, *PrometheusNamespace*, and *PrometheusChartVersion* found in the previous step.

```
helm upgrade PrometheusChartName prometheus-community/prometheus -
n PrometheusNamespace -f prometheus-config.yaml --version PrometheusChartVersion
```

After a few minutes, you'll see a message that the upgrade was successful.

5. Optionally, validate that metrics are succesfully being sent by querying the Amazon Managed Service for Prometheus endpoint via `awscurl`. Replace *Region* with the AWS Region that you are using, and *workspaceURL/* with the URL you found in step 1.

```
awscurl --service="aps" --region="Region" "workspaceURL/api/v1/query?
query=node_cpu_seconds_total"
```

You have now created an Amazon Managed Service for Prometheus workspace and connected to it from your Amazon EKS cluster, using YAML files as configuration. These files, called custom resource definitions (CRDs), live within your Amazon EKS cluster. You can use the AWS Controllers for Kubernetes controller to manage all of your Amazon Managed Service for Prometheus resources directly from the cluster.

# Integrating CloudWatch metrics with Amazon Managed Service for Prometheus

It can help to have all your metrics in one place. Amazon Managed Service for Prometheus does not automatically ingest Amazon CloudWatch metrics. However, you can use Amazon Data Firehose and AWS Lambda to push CloudWatch metrics to Amazon Managed Service for Prometheus.

This section describes how to instrument a Amazon CloudWatch metric stream and use Amazon Data Firehose and AWS Lambda to ingest metrics into Amazon Managed Service for Prometheus.

You will set up a stack using AWS Cloud Development Kit (CDK) to create a Firehose Delivery Stream, a Lambda, and an Amazon S3 bucket to demonstrate a complete scenario.

# Infrastructure

The first thing you must do is set up the infrastructure for this recipe.

CloudWatch metric streams allow forwarding of the streaming metric data to an HTTP endpoint or Amazon S3 bucket.

Setting up the infrastructure will consist of 4 steps:

- Configuring prerequisites
- Creating an Amazon Managed Service for Prometheus workspace
- Installing dependencies
- Deploying the stack

### Prerequisites

- The AWS CLI is installed and configured in your environment.
- The AWS CDK Typescript is installed in your environment.
- Node.js and Go are installed in your environment.
- The AWS observability CloudWatch metrics exporter github repository (`CWMetricsStreamExporter`) has been cloned to your local machine.

### To create a Amazon Managed Service for Prometheus workspace

1. The demo application in this recipe will be running on top of Amazon Managed Service for Prometheus. Create your Amazon Managed Service for Prometheus Workspace via the following command:

   ```
   aws amp create-workspace --alias prometheus-demo-recipe
   ```

2. Ensure your workspace has been created with the following command:

   ```
   aws amp list-workspaces
   ```

   For more information about Amazon Managed Service for Prometheus, see Amazon Managed Service for Prometheus User Guide.

**To install dependencies**

1. **Install dependencies**

   From the root of the `aws-o11y-recipes` repository, change your directory to `CWMetricStreamExporter` using the command:

   ```
   cd sandbox/CWMetricStreamExporter
   ```

   This will now be considered the root of the repo, going forward.

2. Change directory to `/cdk` via the following command:

   ```
   cd cdk
   ```

3. Install the CDK dependencies via the following command:

   ```
   npm install
   ```

4. Change directory back to the root of the repo, and then change directory to `/lambda` using the following command:

   ```
   cd lambda
   ```

5. Once in the `/lambda` folder, install the Go dependencies using:

   ```
   go get
   ```

   All the dependencies are now installed.

**To deploy the stack**

1. In the root of the repo, open `config.yaml` and modify the Amazon Managed Service for Prometheus workspace URL by replacing the `{workspace}` with the newly created workspace id, and the region your Amazon Managed Service for Prometheus workspace is in.

   For example, modify the following with:

   ```
   AMP:
   ```

```
    remote_write_url: "https://aps-workspaces.us-east-2.amazonaws.com/workspaces/
{workspaceId}/api/v1/remote_write"
    region: us-east-2
```

Change the names of the Firehose delivery stream and Amazon S3 bucket to your liking.

2. To build the AWS CDK and the Lambda code, in the root of the repo run the following commend:

```
npm run build
```

This build step ensures that the Go Lambda binary is built, and deploys the CDK to CloudFormation.

3. To complete the deployment, review and accept the IAM changes that the stack requires.
4. (Optional) You can very if that the stack has been created by running the following command.

```
aws cloudformation list-stacks
```

A stack named `CDK Stack` will be in the list.

## Creating a Amazon CloudWatch stream

Now that you have a lambda function to handle the metrics, you can create the metrics stream from Amazon CloudWatch.

**To create an CloudWatch metrics stream**

1. Navigate to the CloudWatch console, at https://console.aws.amazon.com/cloudwatch/home#metric-streams:streamsList and select **Create metric stream**.
2. Select the metrics needed, either all metrics, or only from selected namespaces.
3. Under `Configuration`, choose **Select an existing Firehose owned by your account**.
4. You will be using the Firehose created earlier by the CDK. In the **Select your Kinesis data Firehose stream** drop down, select the stream created earlier. It will have a name like `CdkStack-KinesisFirehoseStream123456AB-sample1234`.
5. Change the output format to **JSON**.
6. Give the metric stream a name that is meaningful to you.
7. Choose **Create metric stream**.

8. (Optional) To verify the Lambda function invocation, navigate to the Lambda console and choose the function `KinesisMessageHandler`. Select the **Monitor** tab and **Logs** subtab, and under **Recent Invocations** there should be entries of the Lambda function being triggered.

   > **Note**
   >
   > It may take up to 5 minutes before invocations begin to show in the **Monitor** tab.

Your metrics are now being streamed from Amazon CloudWatch to Amazon Managed Service for Prometheus.

## Cleanup

You may want to clean up the resources that were used in this example. The following procedure explains how to do so. This will stop the metrics stream that you created.

**To clean up resources**

1. Start by deleting the CloudFormation stack with the following commands:

   ```
   cd cdk
   cdk destroy
   ```

2. Remove the Amazon Managed Service for Prometheus workspace:

   ```
   aws amp delete-workspace --workspace-id \
       `aws amp list-workspaces --alias prometheus-sample-app --query
    'workspaces[0].workspaceId' --output text`
   ```

3. Finally, remove the Amazon CloudWatch metric stream using the Amazon CloudWatch console.

# Security in Amazon Managed Service for Prometheus

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

Security is a shared responsibility between AWS and you. The 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 .
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

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

**Topics**


- Data protection in Amazon Managed Service for Prometheus
- Identity and Access Management for Amazon Managed Service for Prometheus
- IAM permissions and policies
- Compliance Validation for Amazon Managed Service for Prometheus
- Resilience in Amazon Managed Service for Prometheus
- Infrastructure Security in Amazon Managed Service for Prometheus
- Using service-linked roles for Amazon Managed Service for Prometheus
- Logging Amazon Managed Service for Prometheus API calls using AWS CloudTrail
- Set up IAM roles for service accounts
- Using Amazon Managed Service for Prometheus with interface VPC endpoints

# Data protection in Amazon Managed Service for Prometheus

The AWS shared responsibility model applies to data protection in Amazon Managed Service for Prometheus. 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. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see Data Privacy FAQ. For information about data protection in Europe, see the General Data Protection Regulation (GDPR) Center.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or 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 require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- 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 sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 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) 140-3.

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

**Topics**

- Data collected by Amazon Managed Service for Prometheus
- Encryption at rest


## Data collected by Amazon Managed Service for Prometheus

Amazon Managed Service for Prometheus collects and stores operational metrics that you configure to be sent from Prometheus servers running in your account to Amazon Managed Service for Prometheus. This data includes the following:

- Metric values
- Metric labels (or arbitrary key-value pairs) that help identify and classify data
- Timestamps for data samples

Unique tenant IDs isolate data from different customers. These IDs limit what customer data is accessible. Customers can't change tenant IDs.

Amazon Managed Service for Prometheus encrypts the data that it stores with AWS Key Management Service (AWS KMS) keys. Amazon Managed Service for Prometheus manages these keys.

> **Note**
>
> Amazon Managed Service for Prometheus supports the creation of customer managed keys for encrypting your data. For more information about the keys that Amazon Managed Service for Prometheus uses by default, and how to use your own customer managed keys, see Encryption at rest.

Data in transit is encrypted with HTTPS automatically. Amazon Managed Service for Prometheus secures connections between Availability Zones within an AWS Region using HTTPS internally.

## Encryption at rest

By default, Amazon Managed Service for Prometheus automatically provides you with encryption at rest and does this using AWS owned encryption keys.

- **AWS owned keys** – Amazon Managed Service for Prometheus uses these keys to automatically encrypt data uploaded to your workspace. You can't view, manage or use AWS owned keys, or

audit their use. However, you don't have to take any action or change any programs to protect the keys that encrypt your data. For more information, see AWS owned keys in the *AWS Key Management Service Developer Guide*.

Encryption of data at rest helps reduce the operational overhead and complexity that goes into protecting sensitive customer data, such as personally identifiable information. It allows you to build secure applications that meet strict encryption compliance and regulatory requirements.

You can alternatively choose to use a customer managed key when you create your workspace:

- **Customer managed keys** – Amazon Managed Service for Prometheus supports the use of a symmetric customer managed key that you create, own, and manage to encrypt the data in your workspace. Because you have full control of this encryption, you can perform such tasks as:
  - Establishing and maintaining key policies
  - Establishing and maintaining IAM policies and grants
  - Enabling and disabling key policies
  - Rotating key cryptographic material
  - Adding tags
  - Creating key aliases
  - Scheduling keys for deletion

  For more information, see customer managed keys in the *AWS Key Management Service Developer Guide.*

Choose whether to use customer managed keys or AWS owned keys carefully. Workspaces created with customer managed keys can't be converted to use AWS owned keys later (and vice versa).

> **Note**
>
> Amazon Managed Service for Prometheus automatically enables encryption at rest using AWS owned keys to protect your data at no charge.
> However, AWS KMS charges apply for using a customer managed key. For more information about pricing, see AWS Key Management Service pricing.

For more information on AWS KMS, see What is AWS Key Management Service?

> **Note**
>
> Workspaces created with customer managed keys cannot use AWS managed collectors for ingestion.

## How Amazon Managed Service for Prometheus uses grants in AWS KMS

Amazon Managed Service for Prometheus requires three grants to use your customer managed key.

When you create an Amazon Managed Service for Prometheus workspace encrypted with a customer managed key, Amazon Managed Service for Prometheus creates the three grants on your behalf by sending CreateGrant requests to AWS KMS. Grants in AWS KMS are used to give Amazon Managed Service for Prometheus access to the KMS key in your account, even when not called directly on your behalf (for example, when storing metrics data that has been scraped from an Amazon EKS cluster.

Amazon Managed Service for Prometheus requires the grants to use your customer managed key for the following internal operations:

- Send DescribeKey requests to AWS KMS to verify that the symmetric customer managed KMS key given when creating a workspace is valid.
- Send GenerateDataKey requests to AWS KMS to generate data keys encrypted by your customer managed key.
- Send Decrypt requests to AWS KMS to decrypt the encrypted data keys so that they can be used to encrypt your data.

Amazon Managed Service for Prometheus creates three grants to the AWS KMS key that allow Amazon Managed Service for Prometheus to use the key on your behalf. You can remove access to the key by changing the key policy, by disabling the key, or by revoking the grant. You should understand the consequences of these actions before performing them. This can cause data loss in your workspace.

If you remove access to any of the grants in any way, Amazon Managed Service for Prometheus won't be able to access any of the data encrypted by the customer managed key, nor store new data sent to the workspace, which affects operations that are dependent on that data. New data sent to the workspace will not be accessible and may be permanently lost.

**Warning**

- If you disable the key, or remove Amazon Managed Service for Prometheus access in the key policy, the workspace data is no longer accessible. New data being sent to the workspace will not be accessible and may be permanently lost.

  You can get access to the workspace data and start receiving new data again by restoring Amazon Managed Service for Prometheus access to the key.
- If you *revoke* a grant, it can't be recreated, and the data in the workspace is lost permanently.

## Step 1: Create a customer managed key

You can create a symmetric customer managed key by using the AWS Management Console, or the AWS KMS APIs. The key does not need to be in the same account as the Amazon Managed Service for Prometheus workspace, as long as you provide the correct access through policy, as described below.

**To create a symmetric customer managed key**

Follow the steps for Creating symmetric customer managed key in the *AWS Key Management Service Developer Guide*.

**Key policy**

Key policies control access to your customer managed key. Every customer managed key must have exactly one key policy, which contains statements that determine who can use the key and how they can use it. When you create your customer managed key, you can specify a key policy. For more information, see Managing access to customer managed keys in the *AWS Key Management Service Developer Guide*.

To use your customer managed key with your Amazon Managed Service for Prometheus workspaces, the following API operations must be permitted in the key policy:

- `kms:CreateGrant` – Adds a grant to a customer managed key. Grants control access to a specified KMS key, which allows access to grant operations Amazon Managed Service for Prometheus requires. For more information, see Using Grants in the *AWS Key Management Service Developer Guide*.

This allows Amazon Managed Service for Prometheus to do the following:

- Call `GenerateDataKey` to generate an encrypted data key and store it, because the data key isn't immediately used to encrypt.
- Call `Decrypt` to use the stored encrypted data key to access encrypted data.

- `kms:DescribeKey` – Provides the customer managed key details to allow Amazon Managed Service for Prometheus to validate the key.

The following are policy statement examples you can add for Amazon Managed Service for Prometheus:

```
 "Statement" : [
   {
     "Sid" : "Allow access to Amazon Managed Service for Prometheus principal within
your account",
     "Effect" : "Allow",
     "Principal" : {
       "AWS" : "*"
     },
     "Action" : [
       "kms:DescribeKey",
       "kms:CreateGrant",
       "kms:GenerateDataKey",
       "kms:Decrypt"
     ],
     "Resource" : "*",
     "Condition" : {
       "StringEquals" : {
         "kms:ViaService" : "aps.region.amazonaws.com",
         "kms:CallerAccount" : "111122223333"
       }
   },
   {
     "Sid": "Allow access for key administrators - not required for Amazon Managed
Service for Prometheus",
     "Effect": "Allow",
     "Principal": {
       "AWS": "arn:aws:iam::111122223333:root"
      },
     "Action" : [
       "kms:*"
```

```
      ],
      "Resource": "arn:aws:kms:region:111122223333:key/key_ID"
    },
    <other statements needed for other non-Amazon Managed Service for Prometheus
scenarios>
  ]
```

- For more information about specifying permissions in a policy, see the *AWS Key Management Service Developer Guide*.
- For more information about troubleshooting key access, see the *AWS Key Management Service Developer Guide*.

## Step 2: Specifying a customer managed key for Amazon Managed Service for Prometheus

When you create a workspace, you can specify the customer managed key by entering a **KMS Key ARN**, which Amazon Managed Service for Prometheus uses to encrypt the data stored by the workspace.

## Step 3: Accessing data from other services, such as Amazon Managed Grafana

*This step is optional — it is only required if you need to access your Amazon Managed Service for Prometheus data from another service.*

Your encrypted data is not accessible from other services, unless they also have access to use the AWS KMS key. For example, if you want to use Amazon Managed Grafana to create a dashboard or alert on your data, you must give Amazon Managed Grafana access to the key.

**To give Amazon Managed Grafana access to your customer managed key**

1. In your Amazon Managed Grafana workspaces list, select the name for the workspace that you want to have access to Amazon Managed Service for Prometheus. This shows you summary information about your Amazon Managed Grafana workspace.
2. Note the name of the IAM role used by your workspace. The name is in the format `AmazonGrafanaServiceRole-<unique-id>`. The console shows you the full ARN for the role. You will specify this name in the AWS KMS console in a later step.

3. In your AWS KMS Customer managed keys list, choose the customer managed key you used during creation of your Amazon Managed Service for Prometheus workspace. This opens the key configuration details page.
4. Next to **Key users**, select the **Add** button.
5. From the list of names, choose the Amazon Managed Grafana IAM role that you noted above. To make it easier to find, you can search by the name, as well.
6. Choose **Add** to add the IAM role to the list of Key users.

Your Amazon Managed Grafana workspace can now access the data in your Amazon Managed Service for Prometheus workspace. You can add other users or roles to the key users to enable other services to access your workspace.

## Amazon Managed Service for Prometheus encryption context

An encryption context is an optional set of key-value pairs that contain additional contextual information about the data.

AWS KMS uses the encryption context as additional authenticated data to support authenticated encryption. When you include an encryption context in a request to encrypt data, AWS KMS binds the encryption context to the encrypted data. To decrypt data, you include the same encryption context in the request.

### Amazon Managed Service for Prometheus encryption context

Amazon Managed Service for Prometheus uses the same encryption context in all AWS KMS cryptographic operations, where the key is `aws:amp:arn` and the value is the Amazon Resource Name (ARN) of the workspace.

### Example

```
"encryptionContext": {
    "aws:aps:arn": "arn:aws:aps:us-west-2:111122223333:workspace/ws-sample-1234-
abcd-56ef-7890abcd12ef"
}
```

### Using encryption context for monitoring

When you use a symmetric customer managed key to encrypt your workspace data, you can also use the encryption context in audit records and logs to identify how the customer managed key is

being used. The encryption context also appears in [logs generated by AWS CloudTrail or Amazon CloudWatch Logs](#).

**Using encryption context to control access to your customer managed key**

You can use the encryption context in key policies and IAM policies as `conditions` to control access to your symmetric customer managed key. You can also use encryption context constraints in a grant.

Amazon Managed Service for Prometheus uses an encryption context constraint in grants to control access to the customer managed key in your account or region. The grant constraint requires that the operations that the grant allows use the specified encryption context.

**Example**

The following are example key policy statements to give access to a customer managed key for a specific encryption context. The condition in this policy statement requires that the grants have an encryption context constraint that specifies the encryption context.

```
{
    "Sid": "Enable DescribeKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleReadOnlyRole"
     },
     "Action": "kms:DescribeKey",
     "Resource": "*"
},
{
     "Sid": "Enable CreateGrant",
     "Effect": "Allow",
     "Principal": {
         "AWS": "arn:aws:iam::111122223333:role/ExampleReadOnlyRole"
     },
     "Action": "kms:CreateGrant",
     "Resource": "*",
     "Condition": {
         "StringEquals": {
             "kms:EncryptionContext:aws:aps:arn": "arn:aws:aps:us-
west-2:111122223333:workspace/ws-sample-1234-abcd-56ef-7890abcd12ef"
          }
     }
```

```
}
```

## Monitoring your encryption keys for Amazon Managed Service for Prometheus

When you use an AWS KMS customer managed key with your Amazon Managed Service for Prometheus workspaces, you can use AWS CloudTrail or Amazon CloudWatch Logs to track requests that Amazon Managed Service for Prometheus sends to AWS KMS.

The following examples are AWS CloudTrail events for `CreateGrant`, `GenerateDataKey`, `Decrypt`, and `DescribeKey` to monitor KMS operations called by Amazon Managed Service for Prometheus to access data encrypted by your customer managed key:

CreateGrant

When you use an AWS KMS customer managed key to encrypt your workspace, Amazon Managed Service for Prometheus sends three `CreateGrant` requests on your behalf to access the KMS key you specified. The grants that Amazon Managed Service for Prometheus creates are specific to the resource associated with the AWS KMS customer managed key.

The following example event records a `CreateGrant` operation:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "TESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE-KEY-ID1",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "TESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-22T17:02:00Z"
            }
```

```
        },
        "invokedBy": "aps.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "retiringPrincipal": "aps.region.amazonaws.com",
        "operations": [
            "GenerateDataKey",
            "Decrypt",
            "DescribeKey"
        ],
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "granteePrincipal": "aps.region.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

### GenerateDataKey

When you enable an AWS KMS customer managed key for your workspace, Amazon Managed Service for Prometheus creates a unique key. It sends a `GenerateDataKey` request to AWS KMS that specifies the AWS KMScustomer managed key for the resource.

The following example event records the `GenerateDataKey` operation:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "aps.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "encryptionContext": {
            "aws:aps:arn": "arn:aws:aps:us-west-2:111122223333:workspace/ws-
sample-1234-abcd-56ef-7890abcd12ef"
        },
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
```

```
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "57f5dbee-16da-413e-979f-2c4c6663475e"
}
```

### Decrypt

When a query is generated on an encrypted workspace, Amazon Managed Service for Prometheus calls the `Decrypt` operation to use the stored encrypted data key to access the encrypted data.

The following example event records the `Decrypt` operation:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "aps.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:10:51Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "encryptionContext": {
            "aws:aps:arn": "arn:aws:aps:us-west-2:111122223333:workspace/ws-
sample-1234-abcd-56ef-7890abcd12ef"
        },
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "dc129381-1d94-49bd-b522-f56a3482d088"
}
```

DescribeKey

Amazon Managed Service for Prometheus uses the `DescribeKey` operation to verify if the AWS KMS customer managed key associated with your workspace exists in the account and region.

The following example event records the `DescribeKey` operation:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "TESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE-KEY-ID1",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "TESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-22T17:02:00Z"
            }
        },
        "invokedBy": "aps.amazonaws.com"
    },
```

```
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "keyId": "00dd0db0-0000-0000-ac00-b0c000SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```


### Learn more

The following resources provide more information about data encryption at rest.

- For more information about AWS Key Management Service basic concepts, see the *AWS Key Management Service Developer Guide*.
- For more information about Security best practices for AWS Key Management Service, see the *AWS Key Management Service Developer Guide*.

# Identity and Access Management for Amazon Managed Service for Prometheus

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon Managed Service for Prometheus resources. IAM is an AWS service that you can use with no additional charge.

**Topics**


- Audience
- Authenticating with identities
- Managing access using policies
- How Amazon Managed Service for Prometheus works with IAM
- Identity-based policy examples for Amazon Managed Service for Prometheus
- Troubleshooting Amazon Managed Service for Prometheus identity and access


## Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting Amazon Managed Service for Prometheus identity and access)
- **Service administrator** - determine user access and submit permission requests (see How Amazon Managed Service for Prometheus works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policy examples for Amazon Managed Service for Prometheus)

## Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

### AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

### Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

### IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

### IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

# Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

## Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

## Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

## Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

### Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

## How Amazon Managed Service for Prometheus works with IAM

Before you use IAM to manage access to Amazon Managed Service for Prometheus, learn what IAM features are available to use with Amazon Managed Service for Prometheus.

**IAM features you can use with Amazon Managed Service for Prometheus**

| IAM feature | Amazon Managed Service for Prometheus support |
|---|---|
| Identity-based policies | Yes |
| Resource-based policies | Yes |
| Policy actions | Yes |
| Policy resources | Yes |
| Policy condition keys | No |

| IAM feature | Amazon Managed Service for Prometheus support |
| --- | --- |
| ACLs | No |
| ABAC (tags in policies) | Yes |
| Temporary credentials | Yes |
| Forward access sessions (FAS) | No |
| Service roles | No |
| Service-linked roles | Yes |

To get a high-level view of how Amazon Managed Service for Prometheus and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

## Identity-based policies for Amazon Managed Service for Prometheus

**Supports identity-based policies:** Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

### Identity-based policy examples for Amazon Managed Service for Prometheus

To view examples of Amazon Managed Service for Prometheus identity-based policies, see Identity-based policy examples for Amazon Managed Service for Prometheus.

## Resource-based policies within Amazon Managed Service for Prometheus

**Supports resource-based policies:** Yes

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

## Policy actions for Amazon Managed Service for Prometheus

**Supports policy actions:** Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Amazon Managed Service for Prometheus actions, see Actions defined by Amazon Managed Service for Prometheus in the *Service Authorization Reference*.

Policy actions in Amazon Managed Service for Prometheus use the following prefix before the action:

```
aps
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "aps:action1",
      "aps:action2"
         ]
```

To view examples of Amazon Managed Service for Prometheus identity-based policies, see Identity-based policy examples for Amazon Managed Service for Prometheus.

## Policy resources for Amazon Managed Service for Prometheus

**Supports policy resources:** Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of Amazon Managed Service for Prometheus resource types and their ARNs, see Resources defined by Amazon Managed Service for Prometheus in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Amazon Managed Service for Prometheus.

To view examples of Amazon Managed Service for Prometheus identity-based policies, see Identity-based policy examples for Amazon Managed Service for Prometheus.

## Policy condition keys for Amazon Managed Service for Prometheus

**Supports service-specific policy condition keys:** No

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Amazon Managed Service for Prometheus condition keys, see Condition keys for Amazon Managed Service for Prometheus in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by Amazon Managed Service for Prometheus.

To view examples of Amazon Managed Service for Prometheus identity-based policies, see Identity-based policy examples for Amazon Managed Service for Prometheus.

## Access control lists (ACLs) in Amazon Managed Service for Prometheus

**Supports ACLs:** No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

## Attribute-based access control (ABAC) with Amazon Managed Service for Prometheus

**Supports ABAC (tags in policies):** Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

## Using temporary credentials with Amazon Managed Service for Prometheus

**Supports temporary credentials:** Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

## Forward access sessions for Amazon Managed Service for Prometheus

**Supports forward access sessions (FAS):** No

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

## Service roles for Amazon Managed Service for Prometheus

**Supports service roles:** No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break Amazon Managed Service for Prometheus functionality. Edit service roles only when Amazon Managed Service for Prometheus provides guidance to do so.

## Service-linked roles for Amazon Managed Service for Prometheus

**Supports service-linked roles:** Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing Amazon Managed Service for Prometheus service-linked roles, see Using service-linked roles for Amazon Managed Service for Prometheus.

# Identity-based policy examples for Amazon Managed Service for Prometheus

By default, users and roles don't have permission to create or modify Amazon Managed Service for Prometheus resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by Amazon Managed Service for Prometheus, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon Managed Service for Prometheus in the *Service Authorization Reference*.

**Topics**


- Policy best practices
- Using the Amazon Managed Service for Prometheus console
- Allow users to view their own permissions


## Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon Managed Service for Prometheus resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.

- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

## Using the Amazon Managed Service for Prometheus console

To access the Amazon Managed Service for Prometheus console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon Managed Service for Prometheus resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the Amazon Managed Service for Prometheus console, also attach the Amazon Managed Service for Prometheus `ConsoleAccess` or `ReadOnly` AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

## Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws-eusc:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

# Troubleshooting Amazon Managed Service for Prometheus identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon Managed Service for Prometheus and IAM.

**Topics**


- I am not authorized to perform an action in Amazon Managed Service for Prometheus
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my Amazon Managed Service for Prometheus resources


## I am not authorized to perform an action in Amazon Managed Service for Prometheus

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `aps:`*`GetWidget`* permissions.

```
User: arn:aws-eusc:iam::123456789012:user/mateojackson is not authorized to perform:
 aps:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `aps:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

## I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Amazon Managed Service for Prometheus.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon Managed Service for Prometheus. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws-eusc:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

## I want to allow people outside of my AWS account to access my Amazon Managed Service for Prometheus resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon Managed Service for Prometheus supports these features, see How Amazon Managed Service for Prometheus works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

# IAM permissions and policies

Access to Amazon Managed Service for Prometheus actions and data requires credentials. Those credentials must have permissions to perform the actions and to access the AWS resources, such as retrieving Amazon Managed Service for Prometheus data about your cloud resources. The following sections provide details about how you can use AWS Identity and Access Management (IAM) and Amazon Managed Service for Prometheus to help secure your resources, by controlling who can access them. For more information, see Policies and permissions in IAM.

## Amazon Managed Service for Prometheus permissions

To see the list of possible Amazon Managed Service for Prometheus actions. resource types, and condition keys, see Actions, resources, and condition keys for Amazon Managed Service for Prometheus.

## Sample IAM policies

This section provides examples of other self-managed policies that you can create.

The following IAM policy grants full access to Amazon Managed Service for Prometheus and also enables a user to discover Amazon EKS clusters and see the details about them.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aps:*",
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

# Compliance Validation for Amazon Managed Service for Prometheus

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

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

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. For more information about your compliance responsibility when using AWS services, see AWS Security Documentation.

# Resilience in Amazon Managed Service for Prometheus

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 zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

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

In addition to the AWS global infrastructure, Amazon Managed Service for Prometheus offers several features to help support your data resiliency and backup needs, including support for high-availability data.

# Infrastructure Security in Amazon Managed Service for Prometheus

As a managed service, Amazon Managed Service for Prometheus is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Amazon Managed Service for Prometheus through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

# Using service-linked roles for Amazon Managed Service for Prometheus

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

A service-linked role makes setting up Amazon Managed Service for Prometheus easier because you don't have to manually add the necessary permissions. Amazon Managed Service for Prometheus defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon Managed Service for Prometheus 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.

## Using roles for scraping metrics from EKS

When automatically scraping metrics using Amazon Managed Service for Prometheus managed collector, the AWSServiceRoleForAmazonPrometheusScraper service-linked role is used to make setting up managed collector easier, because you don't have to manually add the necessary permissions. Amazon Managed Service for Prometheus defines the permissions, and only Amazon Managed Service for Prometheus can assume the role.

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

## Service-linked role permissions for Amazon Managed Service for Prometheus

Amazon Managed Service for Prometheus uses a service-linked role named with the prefix **AWSServiceRoleForAmazonPrometheusScraper** to allow Amazon Managed Service for Prometheus to automatically scrape metrics in your Amazon EKS clusters.

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

- `scraper.aps.amazonaws.com`

The role permissions policy named AmazonPrometheusScraperServiceRolePolicy allows Amazon Managed Service for Prometheus to complete the following actions on the specified resources:

- Read and modify network configuration to connect to the network that contains your Amazon EKS cluster.
- Read metrics from Amazon EKS clusters and write metrics to your Amazon Managed Service for Prometheus workspaces.
- Write metrics to Amazon CloudWatch Metrics datasets in the same account.

The role has no Amazon MSK permissions by default. To collect consumer-group metrics with a managed Kafka exporter, grant the role access with an Amazon MSK cluster policy. For more information, see Grant the scraper access to the cluster.

You must configure permissions to allow your users, groups, or roles to create a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

## Creating a service-linked role for Amazon Managed Service for Prometheus

You don't need to manually create a service-linked role. When you create an managed collector instance using Amazon EKS or Amazon Managed Service for Prometheus in the AWS Management Console, the AWS CLI, or the AWS API, Amazon Managed Service for Prometheus creates the service-linked role for you.

> **Important**
>
> This service-linked role can appear in your account if you completed an action in another service that uses the features supported by this role. To learn more, see A new role appeared in my AWS account.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create an managed collector instance using Amazon EKS or Amazon Managed Service for Prometheus, Amazon Managed Service for Prometheus creates the service-linked role for you again.

### Editing a service-linked role for Amazon Managed Service for Prometheus

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

### Deleting a service-linked role for Amazon Managed Service for Prometheus

You don't need to manually delete the AWSServiceRoleForAmazonPrometheusScraper role. When you delete all managed collector instances associated with the role in the AWS Management Console, the AWS CLI, or the AWS API, Amazon Managed Service for Prometheus cleans up the resources and deletes the service-linked role for you.

## Logging Amazon Managed Service for Prometheus API calls using AWS CloudTrail

Amazon Managed Service for Prometheus is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures all API calls for Amazon Managed Service for Prometheus as events. The calls captured include calls from the Amazon Managed Service for Prometheus console and code calls to the Amazon Managed Service for Prometheus API operations. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Managed Service for Prometheus, the IP address from which the request was made, when it was made, and additional details.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or user credentials.
- Whether the request was made on behalf of an IAM Identity Center user.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

CloudTrail is active in your AWS account when you create the account and you automatically have access to the CloudTrail **Event history**. The CloudTrail **Event history** provides a viewable, searchable, downloadable, and immutable record of the past 90 days of recorded management events in an AWS Region. For more information, see Working with CloudTrail Event history in the *AWS CloudTrail User Guide*. There are no CloudTrail charges for viewing the **Event history**.

For an ongoing record of events in your AWS account past 90 days, create a trail or a CloudTrail Lake event data store.

**CloudTrail trails**

A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. All trails created using the AWS Management Console are multi-Region. You can create a single-Region or a multi-Region trail by using the AWS CLI. Creating a multi-Region trail is recommended because you capture activity in all AWS Regions in your account. If you create a single-Region trail, you can view only the events logged in the trail's AWS Region. For more information about trails, see Creating a trail for your AWS account and Creating a trail for an organization in the *AWS CloudTrail User Guide*.

You can deliver one copy of your ongoing management events to your Amazon S3 bucket at no charge from CloudTrail by creating a trail, however, there are Amazon S3 storage charges. For more information about CloudTrail pricing, see AWS CloudTrail Pricing. For information about Amazon S3 pricing, see Amazon S3 Pricing.

**CloudTrail Lake event data stores**

*CloudTrail Lake* lets you run SQL-based queries on your events. CloudTrail Lake converts existing events in row-based JSON format to Apache ORC format. ORC is a columnar storage format that is optimized for fast retrieval of data. Events are aggregated into *event data stores*, which are immutable collections of events based on criteria that you select by applying advanced event selectors. The selectors that you apply to an event data store control which events persist

and are available for you to query. For more information about CloudTrail Lake, see Working with AWS CloudTrail Lake in the *AWS CloudTrail User Guide*.

CloudTrail Lake event data stores and queries incur costs. When you create an event data store, you choose the pricing option you want to use for the event data store. The pricing option determines the cost for ingesting and storing events, and the default and maximum retention period for the event data store. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

## Amazon Managed Service for Prometheus management events in CloudTrail

Management events provide information about management operations that are performed on resources in your AWS account. These are also known as control plane operations. By default, CloudTrail logs management events.

Amazon Managed Service for Prometheus logs all Amazon Managed Service for Prometheus control plane operations as management events. For a list of the Amazon Managed Service for Prometheus control plane operations that Amazon Managed Service for Prometheus logs to CloudTrail, see the Amazon Managed Service for Prometheus API Reference.

## Amazon Managed Service for Prometheus event examples

An event represents a single request from any source and includes information about the requested API operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so events don't appear in any specific order.

**Example: CreateWorkspace**

The following example shows a CloudTrail log entry that demonstrates the CreateWorkspace action.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE123EXAMPLE123-1234567890616",
        "arn": "arn:aws:sts::123456789012:assumed-role/Admin/admin",
        "accountId": "123456789012",
```

```
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "webIdFederationData": {

            },
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-11-30T23:39:29Z"
            }
        }
    },
    "eventTime": "2020-11-30T23:43:21Z",
    "eventSource": "aps.amazonaws.com",
    "eventName": "CreateWorkspace",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.1",
    "userAgent": "aws-cli/1.11.167 Python/2.7.10 Darwin/16.7.0 botocore/1.7.25",
    "requestParameters": {
        "alias": "alias-example",
        "clientToken": "12345678-1234-abcd-1234-12345abcd1"
    },
    "responseElements": {
        "Access-Control-Expose-Headers": "x-amzn-errortype,x-amzn-requestid,x-amzn-
trace-id,x-amzn-errormessage,x-amz-apigw-id,date",
        "arn": "arn:aws:aps:us-west-2:123456789012:workspace/ws-abc123456-
abcd-1234-5678-1234567890",
        "status": {
            "statusCode": "CREATING"
        },
        "workspaceId": "ws-12345678-1234-abcd-1234-1234567890"
    },
    "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
    "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
```

```
    "recipientAccountId": "123456789012"
}
```

### Example: CreateAlertManagerDefinition

The following example shows a CloudTrail log entry that demonstrates the CreateAlertManagerDefinition action.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE123EXAMPLE123-1234567890616",
        "arn": "arn:aws:sts::123456789012:assumed-role/Admin/admin",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "webIdFederationData": {

            },
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-09-23T20:20:14Z"
            }
        }
    },
    "eventTime": "2021-09-23T20:22:43Z",
    "eventSource": "aps.amazonaws.com",
    "eventName": "CreateAlertManagerDefinition",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.1",
    "userAgent": "Boto3/1.17.46 Python/3.6.14 Linux/4.14.238-182.422.amzn2.x86_64 exec-
env/AWS_ECS_FARGATE Botocore/1.20.46",
    "requestParameters": {
        "data":
 "YWxlcnRtYW5hZ2VyX2NvbmZpZzogfAogIGdsb2JhbDoKICAgIHNtdHBfc21hcnRob3N0OiAnbG9jYWxob3N0OjI1JwogI
```

```
        "clientToken": "12345678-1234-abcd-1234-12345abcd1",
        "workspaceId": "ws-12345678-1234-abcd-1234-1234567890"
    },
    "responseElements": {
        "Access-Control-Expose-Headers": "x-amzn-errortype,x-amzn-requestid,x-amzn-
trace-id,x-amzn-errormessage,x-amz-apigw-id,date",
        "status": {
            "statusCode": "CREATING"
        }
    },
    "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
    "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123456789012"
}
```

**Example: CreateRuleGroupsNamespace**

The following example shows a CloudTrail log entry that demonstrates the CreateRuleGroupsNamespace action.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE123EXAMPLE123-1234567890616",
        "arn": "arn:aws:sts::123456789012:assumed-role/Admin/admin",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "webIdFederationData": {

            },
```

```
            "attributes": {
                "creationDate": "2021-09-23T20:22:19Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2021-09-23T20:25:08Z",
    "eventSource": "aps.amazonaws.com",
    "eventName": "CreateRuleGroupsNamespace",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "34.212.33.165",
    "userAgent": "Boto3/1.17.63 Python/3.6.14 Linux/4.14.238-182.422.amzn2.x86_64 exec-
env/AWS_ECS_FARGATE Botocore/1.20.63",
    "requestParameters": {
        "data":
 "Z3JvdXBzOgogIC0gbmFtZTogdGVzdFJ1bGVHcm91cHNOYW1lc3BhY2UKICAgIHJ1bGVzOgogICAgLSBhbGVydDogdGVzd
        "clientToken": "12345678-1234-abcd-1234-12345abcd1",
        "name": "exampleRuleGroupsNamespace",
        "workspaceId": "ws-12345678-1234-abcd-1234-1234567890"
    },
    "responseElements": {
        "Access-Control-Expose-Headers": "x-amzn-errortype,x-amzn-requestid,x-amzn-
trace-id,x-amzn-errormessage,x-amz-apigw-id,date",
        "name": "exampleRuleGroupsNamespace",
        "arn": "arn:aws:aps:us-west-2:492980759322:rulegroupsnamespace/ws-
ae46a85c-1609-4c22-90a3-2148642c3b6c/exampleRuleGroupsNamespace",
        "status": {
            "statusCode": "CREATING"
        },
        "tags": {}
    },
    "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
    "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123456789012"
}
```


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

# Set up IAM roles for service accounts

With IAM roles for service accounts, you can associate an IAM role with a Kubernetes service account. This service account can then provide AWS permissions to the containers in any pod that uses that service account. For more information, see IAM roles for service accounts.

IAM roles for service accounts are also known as *service roles*.

In Amazon Managed Service for Prometheus, using service roles can help you get the roles you need to authorize and authenticate between Amazon Managed Service for Prometheus, Prometheus servers, and Grafana servers.

**Prerequisites**

The procedures on this page require that you have the AWS CLI and EKSCTL command line interface installed.

## Set up service roles for the ingestion of metrics from Amazon EKS clusters

To set up the service roles to enable Amazon Managed Service for Prometheus to ingest metrics from Prometheus servers in Amazon EKS clusters, you must be logged on to an account with the following permissions:

- `iam:CreateRole`
- `iam:CreatePolicy`
- `iam:GetRole`
- `iam:AttachRolePolicy`
- `iam:GetOpenIDConnectProvider`

**To set up the service role for ingestion into Amazon Managed Service for Prometheus**

1. Create a file named `createIRSA-AMPIngest.sh` with the following content. Replace `<my_amazon_eks_clustername>` with the name of your cluster, and replace `<my_prometheus_namespace>` with your Prometheus namespace.

```
#!/bin/bash -e
CLUSTER_NAME=<my_amazon_eks_clustername>
```

```
SERVICE_ACCOUNT_NAMESPACE=<my_prometheus_namespace>
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
OIDC_PROVIDER=$(aws eks describe-cluster --name $CLUSTER_NAME --query
 "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\/\///")
SERVICE_ACCOUNT_AMP_INGEST_NAME=amp-iamproxy-ingest-service-account
SERVICE_ACCOUNT_IAM_AMP_INGEST_ROLE=amp-iamproxy-ingest-role
SERVICE_ACCOUNT_IAM_AMP_INGEST_POLICY=AMPIngestPolicy
#
# Set up a trust policy designed for a specific combination of K8s service account
 and namespace to sign in from a Kubernetes cluster which hosts the OIDC Idp.
#
cat <<EOF > TrustPolicy.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::${AWS_ACCOUNT_ID}:oidc-provider/
${OIDC_PROVIDER}"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "${OIDC_PROVIDER}:sub": "system:serviceaccount:
${SERVICE_ACCOUNT_NAMESPACE}:${SERVICE_ACCOUNT_AMP_INGEST_NAME}"
        }
      }
    }
  ]
}
EOF
#
# Set up the permission policy that grants ingest (remote write) permissions for
 all AMP workspaces
#
cat <<EOF > PermissionPolicyIngest.json
{
  "Version": "2012-10-17",
   "Statement": [
       {"Effect": "Allow",
        "Action": [
           "aps:RemoteWrite",
           "aps:GetSeries",
```

```
            "aps:GetLabels",
            "aps:GetMetricMetadata"
        ],
        "Resource": "*"
      }
   ]
}
EOF

function getRoleArn() {
  OUTPUT=$(aws iam get-role --role-name $1 --query 'Role.Arn' --output text 2>&1)

  # Check for an expected exception
  if [[ $? -eq 0 ]]; then
    echo $OUTPUT
  elif [[ -n $(grep "NoSuchEntity" <<< $OUTPUT) ]]; then
    echo ""
  else
    >&2 echo $OUTPUT
    return 1
  fi
}

#
# Create the IAM Role for ingest with the above trust policy
#
SERVICE_ACCOUNT_IAM_AMP_INGEST_ROLE_ARN=$(getRoleArn
 $SERVICE_ACCOUNT_IAM_AMP_INGEST_ROLE)
if [ "$SERVICE_ACCOUNT_IAM_AMP_INGEST_ROLE_ARN" = "" ];
then
  #
  # Create the IAM role for service account
  #
  SERVICE_ACCOUNT_IAM_AMP_INGEST_ROLE_ARN=$(aws iam create-role \
  --role-name $SERVICE_ACCOUNT_IAM_AMP_INGEST_ROLE \
  --assume-role-policy-document file://TrustPolicy.json \
  --query "Role.Arn" --output text)
  #
  # Create an IAM permission policy
  #
  SERVICE_ACCOUNT_IAM_AMP_INGEST_ARN=$(aws iam create-policy --policy-name
 $SERVICE_ACCOUNT_IAM_AMP_INGEST_POLICY \
  --policy-document file://PermissionPolicyIngest.json \
  --query 'Policy.Arn' --output text)
```

```
  #
  # Attach the required IAM policies to the IAM role created above
  #
  aws iam attach-role-policy \
  --role-name $SERVICE_ACCOUNT_IAM_AMP_INGEST_ROLE \
  --policy-arn $SERVICE_ACCOUNT_IAM_AMP_INGEST_ARN
else
    echo "$SERVICE_ACCOUNT_IAM_AMP_INGEST_ROLE_ARN IAM role for ingest already
 exists"
fi
echo $SERVICE_ACCOUNT_IAM_AMP_INGEST_ROLE_ARN
#
# EKS cluster hosts an OIDC provider with a public discovery endpoint.
# Associate this IdP with AWS IAM so that the latter can validate and accept the
 OIDC tokens issued by Kubernetes to service accounts.
# Doing this with eksctl is the easier and best approach.
#
eksctl utils associate-iam-oidc-provider --cluster $CLUSTER_NAME --approve
```

2. Enter the following command to give the script the necessary privileges.

```
chmod +x createIRSA-AMPIngest.sh
```

3. Run the script.

## Set up IAM roles for service accounts for the querying of metrics

To set up the IAM role for service account (service role) to enable the querying of metrics from Amazon Managed Service for Prometheus workspaces, you must be logged on to an account with the following permissions:

- `iam:CreateRole`
- `iam:CreatePolicy`
- `iam:GetRole`
- `iam:AttachRolePolicy`
- `iam:GetOpenIDConnectProvider`

**To set up service roles for the querying of Amazon Managed Service for Prometheus metrics;**

1. Create a file named `createIRSA-AMPQuery.sh` with the following content. Replace <my_amazon_eks_clustername> with the name of your cluster, and replace <my_prometheus_namespace> with your Prometheus namespace.

```
#!/bin/bash -e
CLUSTER_NAME=<my_amazon_eks_clustername>
SERVICE_ACCOUNT_NAMESPACE=<my_prometheus_namespace>
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
OIDC_PROVIDER=$(aws eks describe-cluster --name $CLUSTER_NAME --query
 "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\/\///")
SERVICE_ACCOUNT_AMP_QUERY_NAME=amp-iamproxy-query-service-account
SERVICE_ACCOUNT_IAM_AMP_QUERY_ROLE=amp-iamproxy-query-role
SERVICE_ACCOUNT_IAM_AMP_QUERY_POLICY=AMPQueryPolicy
#
# Setup a trust policy designed for a specific combination of K8s service account
 and namespace to sign in from a Kubernetes cluster which hosts the OIDC Idp.
#
cat <<EOF > TrustPolicy.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::${AWS_ACCOUNT_ID}:oidc-provider/
${OIDC_PROVIDER}"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "${OIDC_PROVIDER}:sub": "system:serviceaccount:
${SERVICE_ACCOUNT_NAMESPACE}:${SERVICE_ACCOUNT_AMP_QUERY_NAME}"
        }
      }
    }
  ]
}
EOF
#
# Set up the permission policy that grants query permissions for all AMP workspaces
#
```

```
cat <<EOF > PermissionPolicyQuery.json
{
  "Version": "2012-10-17",
   "Statement": [
       {"Effect": "Allow",
        "Action": [
           "aps:QueryMetrics",
           "aps:GetSeries",
           "aps:GetLabels",
           "aps:GetMetricMetadata"
        ],
        "Resource": "*"
      }
   ]
}
EOF

function getRoleArn() {
  OUTPUT=$(aws iam get-role --role-name $1 --query 'Role.Arn' --output text 2>&1)

  # Check for an expected exception
  if [[ $? -eq 0 ]]; then
    echo $OUTPUT
  elif [[ -n $(grep "NoSuchEntity" <<< $OUTPUT) ]]; then
    echo ""
  else
    >&2 echo $OUTPUT
    return 1
  fi
}

#
# Create the IAM Role for query with the above trust policy
#
SERVICE_ACCOUNT_IAM_AMP_QUERY_ROLE_ARN=$(getRoleArn
 $SERVICE_ACCOUNT_IAM_AMP_QUERY_ROLE)
if [ "$SERVICE_ACCOUNT_IAM_AMP_QUERY_ROLE_ARN" = "" ];
then
  #
  # Create the IAM role for service account
  #
  SERVICE_ACCOUNT_IAM_AMP_QUERY_ROLE_ARN=$(aws iam create-role \
  --role-name $SERVICE_ACCOUNT_IAM_AMP_QUERY_ROLE \
  --assume-role-policy-document file://TrustPolicy.json \
```

```
  --query "Role.Arn" --output text)
  #
  # Create an IAM permission policy
  #
  SERVICE_ACCOUNT_IAM_AMP_QUERY_ARN=$(aws iam create-policy --policy-name
 $SERVICE_ACCOUNT_IAM_AMP_QUERY_POLICY \
  --policy-document file://PermissionPolicyQuery.json \
  --query 'Policy.Arn' --output text)
  #
  # Attach the required IAM policies to the IAM role create above
  #
  aws iam attach-role-policy \
  --role-name $SERVICE_ACCOUNT_IAM_AMP_QUERY_ROLE \
  --policy-arn $SERVICE_ACCOUNT_IAM_AMP_QUERY_ARN
else
    echo "$SERVICE_ACCOUNT_IAM_AMP_QUERY_ROLE_ARN IAM role for query already
 exists"
fi
echo $SERVICE_ACCOUNT_IAM_AMP_QUERY_ROLE_ARN
#
# EKS cluster hosts an OIDC provider with a public discovery endpoint.
# Associate this IdP with AWS IAM so that the latter can validate and accept the
 OIDC tokens issued by Kubernetes to service accounts.
# Doing this with eksctl is the easier and best approach.
#
eksctl utils associate-iam-oidc-provider --cluster $CLUSTER_NAME --approve
```

2. Enter the following command to give the script the necessary privileges.

```
chmod +x createIRSA-AMPQuery.sh
```

3. Run the script.

# Using Amazon Managed Service for Prometheus with interface VPC endpoints

If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources, you can establish private connections between your VPC and Amazon Managed Service for Prometheus. You can use these connections to enable Amazon Managed Service for Prometheus to communicate with your resources on your VPC without going through the public internet.

Amazon VPC is an AWS service that you can use to launch AWS resources in a virtual network that you define. With a VPC, you have control over your network settings, such the IP address range, subnets, route tables, and network gateways. To connect your VPC to Amazon Managed Service for Prometheus, you define an *interface VPC endpoint* to connect your VPC to AWS services. The endpoint provides reliable, scalable connectivity to Amazon Managed Service for Prometheus without requiring an internet gateway, a network address translation (NAT) instance, or a VPN connection. For more information, see What Is Amazon VPC in the *Amazon VPC User Guide*.

Interface VPC endpoints are powered by AWS PrivateLink, an AWS technology that enables private communication between AWS services using an elastic network interface with private IP addresses. For more information, see the New – AWS PrivateLink for AWS Services blog post.

The following information is for Amazon VPC users. For information about how to get started with Amazon VPC, see Getting Started in the *Amazon VPC User Guide*.

## Create an interface VPC endpoint for Amazon Managed Service for Prometheus

Create an interface VPC endpoint to begin using Amazon Managed Service for Prometheus. Choose from the following service name endpoints:

- `com.amazonaws.region.aps-workspaces`

  Choose this service name to work with Prometheus-compatible APIs. For more information, see Prometheus-compatible APIs in the *Amazon Managed Service for Prometheus User Guide*.

- `com.amazonaws.region.aps`

  Choose this service name to perform workspace management tasks. For more information, see Amazon Managed Service for Prometheus APIs in the *Amazon Managed Service for Prometheus User Guide*.

> **Note**
>
> If you are using remote_write in a VPC without direct internet access, you must also create an interface VPC endpoint for AWS Security Token Service, to allow sigv4 to work through the endpoint. For information about creating a VPC endpoint for AWS STS, see Using AWS

STS interface VPC endpoints in the *AWS Identity and Access Management User Guide*. You must set AWS STS to use regionalized endpoints.

For more information, including step-by-step instructions to create an interface VPC endpoint, see Creating an interface endpoint in the *Amazon VPC User Guide*.

**Note**

You can use **VPC endpoint policies** to control access to your Amazon Managed Service for Prometheus interface VPC endpoint. See the next section for more information.

If you created an interface VPC endpoint for Amazon Managed Service for Prometheus and already have data flowing to the workspaces located on your VPC, the metrics will flow through the interface VPC endpoint by default. Amazon Managed Service for Prometheus uses public endpoints or private interface endpoints (whichever are in use) to perform this task.

## Controlling access to your Amazon Managed Service for Prometheus VPC endpoint

You can use VPC endpoint policies to control access to your Amazon Managed Service for Prometheus interface VPC endpoint. A VPC endpoint policy is an IAM resource policy that you attach to an endpoint when you create or modify the endpoint. If you don't attach a policy when you create an endpoint, Amazon VPC attaches a default policy for you that allows full access to the service. An endpoint policy doesn't override or replace IAM identity-based policies or service-specific policies. It's a separate policy for controlling access from the endpoint to the specified service.

For more information, see Controlling Access to Services with VPC Endpoints in the *Amazon VPC User Guide*.

The following is an example of an endpoint policy for Amazon Managed Service for Prometheus. This policy allows users with the role `PromUser` connecting to Amazon Managed Service for Prometheus through the VPC to view workspaces and rule groups, but not, for example, to create or delete workspaces.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonManagedPrometheusPermissions",
            "Effect": "Allow",
            "Action": [
                "aps:DescribeWorkspace",
                "aps:DescribeRuleGroupsNamespace",
                "aps:ListRuleGroupsNamespaces",
                "aps:ListWorkspaces"
            ],
            "Resource": "arn:aws-eusc:aps:*:*:/workspaces*",
            "Principal": {
                "AWS": [
                    "arn:aws-eusc:iam::111122223333:role/PromUser"
                ]
            }
        }
    ]
}
```

The following example shows a policy that only allows requests coming from a specified IP address in the specified VPC to succeed. Requests from other IP addresses will fail.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "aps:*",
            "Effect": "Allow",
            "Principal": "*",
            "Resource": "*",
            "Condition": {
                "IpAddress": {
                    "aws:VpcSourceIp": "192.0.2.123"
                },
        "StringEquals": {
                    "aws:SourceVpc": "vpc-0123456789abcdef0"
```

```
                }
            }
        }
    ]
}
```

# Troubleshoot Amazon Managed Service for Prometheus errors

Use the following sections to help troubleshoot issues with Amazon Managed Service for Prometheus.

**Topics**


- 429 or limit exceeded errors
- I see duplicate samples
- I see errors about sample timestamps
- I see an error message related to a limit
- Your local Prometheus server output exceeds the limit.
- Some of my data isn't appearing
- HTTP 422 error: query timeout


## 429 or limit exceeded errors

If you see a 429 error similar to the following example, your requests have exceeded Amazon Managed Service for Prometheus ingestion quotas.

```
ts=2020-10-29T15:34:41.845Z caller=dedupe.go:112 component=remote level=error
 remote_name=e13b0c
url=http://iamproxy-external.prometheus.uswest2-prod.eks:9090/workspaces/workspace_id/
api/v1/remote_write
msg="non-recoverable error" count=500 err="server returned HTTP status 429
Too Many Requests: ingestion rate limit (6666.666666666667) exceeded while adding 499
 samples and 0 metadata
```

If you see a 429 error similar to the following example, your requests have exceeded the Amazon Managed Service for Prometheus quota for the number of active metrics in a workspace.

```
ts=2020-11-05T12:40:33.375Z caller=dedupe.go:112 component=remote level=error
 remote_name=aps
url=http://iamproxy-external.prometheus.uswest2-prod.eks:9090/workspaces/workspace_id/
api/v1/remote_write
```

```
msg="non-recoverable error" count=500 err="server returned HTTP status 429 Too Many
 Requests: user=accountid_workspace_id:
per-user series limit (local limit: 0 global limit: 3000000 actual local limit: 500000)
 exceeded
```

If you see a 429 error similar to the following example, your requests have exceeded the Amazon Managed Service for Prometheus quota for the rate (transactions per second) that you can send data to your workspace using the `RemoteWrite` Prometheus compatible API.

```
ts=2024-03-26T16:50:21.780708811Z caller=dedupe.go:112 component=remote level=error
 remote_name=ab123c
url=https://aps-workspaces.us-east-1.amazonaws.com/workspaces/workspace_id/api/v1/
remote_write
msg="non-recoverable error" count=1000 exemplarCount=0 err="server returned HTTP status
 429 Too Many Requests: {\"message\":\"Rate exceeded\"}"
```

If you see a 400 error similar to the following example, your requests have exceeded Amazon Managed Service for Prometheus quota for active time series. For details about how active time series quotas are handled, see Active series default quotas.

```
ts=2024-03-26T16:50:21.780708811Z caller=push.go:53 level=warn
url=https://aps-workspaces.us-east-1.amazonaws.com/workspaces/workspace_id/api/v1/
remote_write
msg="non-recoverable error" count=500 exemplarCount=0
err="server returned HTTP status 400 Bad Request: maxFailure (quorum) on a given error
 family, rpc error: code = Code(400)
desc = addr=10.1.41.23:9095 state=ACTIVE zone=us-east-1a, rpc error: code = Code(400)
desc = user=accountid_workspace_id: per-user series limit of 10000000 exceeded,
Capacity from 2,000,000 to 10,000,000 is automatically adjusted based on the last 30
 min of usage.
If throttled above 10,000,000 or in case of incoming surges, please contact
 administrator to raise it.
(local limit: 0 global limit: 10000000 actual local limit: 92879)"
```

For more information about Amazon Managed Service for Prometheus service quotas and about how to request increases, see Amazon Managed Service for Prometheus service quotas

# I see duplicate samples

If you are using a high-availability Prometheus group, you need to use external labels on your Prometheus instances to set up deduplication. For more information, see Deduplicating high availability metrics sent to Amazon Managed Service for Prometheus.

Other issues around duplicated data are discussed in the next section.

# I see errors about sample timestamps

Amazon Managed Service for Prometheus ingests data in order, and expects each sample to have a timestamp later than the previous sample.

If your data does not arrive in order, you can see errors about `out-of-order samples`, `duplicate sample for timestamp`, or `samples with different value but same timestamp`. These issues are typically caused by incorrect setup of the client that is sending data to Amazon Managed Service for Prometheus. If you are using a Prometheus client running in agent mode, check the configuration for rules with duplicate series name, or duplicated targets. If your metrics provide the timestamp directly, check that they are not out of order.

You can configure an out-of-order time window for your workspace to accept out-of-order samples. For more information, see Configure your workspace.

For more details about how this works, or ways to check your setup, see the blog post Understanding Duplicate Samples and Out-of-order Timestamp Errors in Prometheus from *Prom Labs*.

# I see an error message related to a limit

> **Note**
>
> Amazon Managed Service for Prometheus provides CloudWatch usage metrics to monitor Prometheus resource usage. Using the CloudWatch usage metrics alarm feature, you can monitor Prometheus resources and usage to prevent limit errors.

If you see one of the following error messages, you can request an increase in one of the Amazon Managed Service for Prometheus quotas to solve the issue. For more information, see Amazon Managed Service for Prometheus service quotas.

- per-user series limit of *<value>* exceeded, please contact administrator to raise it
- per-metric series limit of *<value>* exceeded, please contact administrator to raise it
- ingestion rate limit (...) exceeded
- series has too many labels (...) series: '%s'
- the query time range exceeds the limit (query length: xxx, limit: yyy)
- the query hit the max number of chunks limit while fetching chunks from ingesters
- Limit exceeded. Maximum workspaces per account.

## Your local Prometheus server output exceeds the limit.

Amazon Managed Service for Prometheus has service quotas for the amount of data that a workspace can receive from Prometheus servers. To find the amount of data that your Prometheus server is sending to Amazon Managed Service for Prometheus, you can run the following queries on your Prometheus server. If you find that your Prometheus output is exceeding a Amazon Managed Service for Prometheus limit, you can request an increase of the corresponding service quota. For more information, see Amazon Managed Service for Prometheus service quotas.

**Queries against your local self-run Prometheus server to find the output limits.**

| Type of data | Query to use |
|---|---|
| Current active series | `prometheus_tsdb_head_series` |
| Current ingestion rate | `rate(prometheus_tsdb_head_samples_appended_total[5m])` |
| Most-to-least list of active series per metric name | `sort_desc(count by(__name__)` |

| Type of data | Query to use |
| --- | --- |
| | `({__name__!=""}))` |
| Number of labels per metric series | `group by(mylabelname) ({__name__!=""})` |

## Some of my data isn't appearing

Data that is sent to Amazon Managed Service for Prometheus can be discarded for various reasons. The following table shows reasons that data might be discarded rather than being ingested.

You can track the amount and reasons that data is discarded using Amazon CloudWatch. For more information, see Use CloudWatch metrics to monitor Amazon Managed Service for Prometheus resources.

| Reason | Meaning |
| --- | --- |
| greater_than_max_sample_age | Discarding log lines which are older than the current time |
| new-value-for-timestamp | Duplicate samples are sent with the same timestamp as the previous sample but with different values. |
| per_metric_series_limit | User has hit the active series per metric limit |
| per_user_series_limit | User has hit the total number of active series limit |
| rate_limited | Ingestion rate limited |
| sample-out-of-order | Samples are sent out of order and cannot be processed . This reason is only emitted when the out-of-order time window is set to 0. |

| Reason | Meaning |
| --- | --- |
| sample-too-old | Sample is older than the configured out-of-order time window and cannot be processed. |
| out-of-order-rate-limit | Out-of-order ingestion rate limit was reached and the sample cannot be processed. For more information, see Amazon Managed Service for Prometheus service quotas. |
| label_value_too_long | Label value is longer than allowed character limit |
| max_label_names_per_series | User has hit the label names per metric |
| missing_metric_name | Metric name is not provided |
| metric_name_invalid | Invalid metric name provided |
| label_invalid | Invalid label provided |
| duplicate_label_names | Duplicate label names provided |
| native_histogram_sample_size_bytes_exceeded | Native histogram sample exceeds the maximum allowed sample size. Reduce the number of buckets or spans in the histogram on the client side. |
| native_histogram_invalid_schema | Native histogram has an invalid schema value. Valid schemas range from -4 to 8. Check your Prometheus client configuration for the `NativeHistogramBucketFactor` setting. |
| native_histogram_invalid | Native histogram fails validation (for example, negative bucket counts, mismatched bucket counts, or malformed bucket spans). This typically indicates a bug in the instrumentation library. |

| Reason | Meaning |
| --- | --- |
| native_histogram_buckets_exceeded | Native histogram exceeds the maximum bucket count limit and resolution cannot be automatically reduced. Reduce the number of buckets on the client side using `NativeHistogramMaxBucketNumber` . |
| native_histogram_rate_limited | Native histogram sample was rejected because the native histogram ingestion rate limit was reached. Reduce the number of native histogram series or the scrape frequency, or request a limit increase. |
| per_user_native_histogram_series_limit | User has hit the native histogram active series limit per workspace. Reduce the number of native histogram series or request a limit increase. |

## HTTP 422 error: query timeout

If your query runs for more than 1 minute, Amazon Managed Service for Prometheus returns an HTTP 422 status code. The following example shows this error:

```
{"status":"error","errorType":"execution","error":"query timed out - consider
 simplifying your query"}
```

To reduce evaluation time, use one or more of the following approaches:

- Specify a shorter time window to evaluate fewer samples.
- Add label matchers to reduce the number of series that your query evaluates.
- Use recording rules to precompute expensive aggregations that you query frequently.
- Avoid regex matchers on high-cardinality labels (labels with many unique values).

For more information, see *Query your metrics*.

# Tagging in Amazon Managed Service for Prometheus

A *tag* is a custom attribute label that you or AWS assigns to an AWS resource. Each AWS tag has two parts:

- A *tag key* (for example, `CostCenter`, `Environment`, `Project`, or `Secret`). Tag keys are case sensitive.
- An optional field known as a *tag value* (for example, `111122223333`, `Production`, or a team name). Omitting the tag value is the same as using an empty string. Like tag keys, tag values are case sensitive.

Together these are known as key-value pairs. You can have as many as 50 tags assigned to each workspace.

Tags help you identify and organize your AWS resources. Many AWS services support tagging, so you can assign the same tag to resources from different services to indicate that the resources are related. For example, you can assign the same tag to an Amazon Managed Service for Prometheus workspace that you assign to an Amazon S3 bucket. For more information about tagging strategies, see Tagging AWS Resources.

In Amazon Managed Service for Prometheus, both workspaces and rule groups namespaces can be tagged. You can use the console, the AWS CLI, APIs, or SDKs to add, manage, and remove tags for these resources. In addition to identifying, organizing, and tracking your workspaces and rule groups namespaces with tags, you can use tags in IAM policies to help control who can view and interact with your Amazon Managed Service for Prometheus resources.

**Tag restrictions**

The following basic restrictions apply to tags:

- Each resource can have a maximum of 50 tags.
- For each resource, each tag key must be unique, and each tag key can have only one value.
- The maximum tag key length is 128 Unicode characters in UTF-8.
- The maximum tag value length is 256 Unicode characters in UTF-8.
- If your tagging schema is used across multiple AWS services and resources, remember that other services may have restrictions on allowed characters. Generally allowed characters are letters, numbers, spaces representable in UTF-8, and the following characters: **. : + = @ _ / -** (hyphen).

- Tag keys and values are case sensitive. As a best practice, decide on a strategy for capitalizing tags and consistently implement that strategy across all resource types. For example, decide whether to use `Costcenter`, `costcenter`, or `CostCenter` and use the same convention for all tags. Avoid using similar tags with inconsistent case treatment.
- Don't use `aws:`, `AWS:`, or any upper or lowercase combination of such as a prefix for either keys or values. These are reserved only for AWS use. You can't edit or delete tag keys or values with this prefix. Tags with this prefix do not count against your tags-per-resource limit.

**Topics**


- Tag Amazon Managed Service for Prometheus workspaces
- Tagging rule groups namespaces


# Tag Amazon Managed Service for Prometheus workspaces

Tags are custom labels that can be assigned to a resource. They include a unique key and an optional value (in a key-value pair). Tags help you identify and organize your AWS resources. In Amazon Managed Service for Prometheus, workspaces (and rule groups namespaces) can be tagged. You can use the console, the AWS CLI, APIs, or SDKs to add, manage, and remove tags for these resources. In addition to identifying, organizing, and tracking your workspaces with tags, you can use tags in IAM policies to help control who can view and interact with your Amazon Managed Service for Prometheus resources.

Use the procedures in this section to work with tags for Amazon Managed Service for Prometheus workspaces.

**Topics**


- Add a tag to a workspace
- View tags for a workspace
- Edit tags for a workspace
- Remove a tag from a workspace


## Add a tag to a workspace

Adding tags to an Amazon Managed Service for Prometheus workspace can help you identify and organize your AWS resources and manage access to them. First, you add one or more tags (key-

value pairs) to a workspace. After you have tags, you can create IAM policies to manage access to the workspace based on these tags. You can use the the console or the AWS CLI to add tags to an Amazon Managed Service for Prometheus workspace.

> **Important**
>
> Adding tags to a workspace can impact access to that workspace. Before you add a tag to a workspace, make sure to review any IAM policies that might use tags to control access to resources.

For more information about adding tags to an Amazon Managed Service for Prometheus workspace when you create it, see Create a Amazon Managed Service for Prometheus workspace.

**Topics**


- Add a tag to a workspace (console)
- Add a tag to a workspace (AWS CLI)


## Add a tag to a workspace (console)

You can use the console to add one or more tags to a Amazon Managed Service for Prometheus workspace.

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the workspace ID of the workspace that you want to manage.
5. Choose the **Tags** tab.
6. If no tags have been added to the Amazon Managed Service for Prometheus workspace, choose **Create tag**. Otherwise, choose **Manage tags**.
7. In **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.
8. (Optional) To add another tag, choose **Add tag** again.
9. When you have finished adding tags, choose **Save changes**.

### Add a tag to a workspace (AWS CLI)

Follow these steps to use the AWS CLI to add a tag to an Amazon Managed Service for Prometheus workspace. To add a tag to a workspace when you create it, see Create a Amazon Managed Service for Prometheus workspace.

In these steps, we assume that you have already installed a recent version of the AWS CLI or updated to the current version. For more information, see Installing the AWS Command Line Interface.

At the terminal or command line, run the **tag-resource** command, specifying the Amazon Resource Name (ARN) of the workspace where you want to add tags and the key and value of the tag you want to add. You can add more than one tag to an workspace. For example, to tag an Amazon Managed Service for Prometheus workspace named **My-Workspace** with two tags, a tag key named *Status* with the tag value of *Secret*, and a tag key named *Team* with the tag value of *My-Team*:

```
aws amp tag-resource --resource-arn arn:aws:aps:us-
west-2:123456789012:workspaces/IDstring
--tags Status=Secret,Team=My-Team
```

If successful, this command returns nothing.

## View tags for a workspace

Tags can help you identify and organize your AWS resources and manage access to them. For more information about tagging strategies, see Tagging AWS Resources.

### View tags for an Amazon Managed Service for Prometheus workspace (console)

You can use the console to view the tags associated with a Amazon Managed Service for Prometheus workspace.

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the workspace ID of the workspace that you want to manage.
5. Choose the **Tags** tab.

### View tags for an Amazon Managed Service for Prometheus workspace (AWS CLI)

Follow these steps to use the AWS CLI to view the AWS tags for an workspace. If no tags have been added, the returned list is empty.

At the terminal or command line, run the **list-tags-for-resource** command. For example, to view a list of tag keys and tag values for a workspace:

```
aws amp list-tags-for-resource --resource-arn arn:aws:aps:us-
west-2:123456789012:workspace/IDstring
```

If successful, this command returns information similar to the following:

```
{
    "tags": {
        "Status": "Secret",
        "Team": "My-Team"
    }
}
```

## Edit tags for a workspace

You can change the value for a tag associated with a workspace. You can also change the name of the key, which is equivalent to removing the current tag and adding a different one with the new name and the same value as the other key.

> **Important**
>
> Editing tags for an Amazon Managed Service for Prometheus workspace can impact access to that workspace. Before you edit the name (key) or value of a tag for a workspace, make sure to review any IAM policies that might use the key or value for a tag to control access to resources such as repositories.

### Edit a tag for an Amazon Managed Service for Prometheus workspace (console)

You can use the console to edit the tags associated with a Amazon Managed Service for Prometheus workspace.

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the workspace ID of the workspace that you want to manage.
5. Choose the **Tags** tab.
6. If no tags have been added to the workspace, choose **Create tag**. Otherwise, choose **Manage tags**.
7. In **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.
8. (Optional) To add another tag, choose **Add tag** again.
9. When you have finished adding tags, choose **Save changes**.

### Edit tags for an Amazon Managed Service for Prometheus workspace (AWS CLI)

Follow these steps to use the AWS CLI to update a tag for a workspace. You can change the value for an existing key, or add another key.

At the terminal or command line, run the **tag-resource** command, specifying the Amazon Resource Name (ARN) of the Amazon Managed Service for Prometheus workspace where you want to update a tag and specify the tag key and tag value:

```
aws amp tag-resource --resource-arn arn:aws:aps:us-
west-2:123456789012:workspace/IDstring --tags Team=New-Team
```

## Remove a tag from a workspace

You can remove one or more tags associated with a workspace. Removing a tag does not delete the tag from other AWS resources that are associated with that tag.

> **Important**
>
> Removing tags for a Amazon Managed Service for Prometheus workspace can impact access to that workspace. Before you remove a tag from a workspace, make sure to review any IAM policies that might use the key or value for a tag to control access to resources such as repositories.

## Remove a tag from an Amazon Managed Service for Prometheus workspace (console)

You can use the console to remove the association between a tag and a workspace.

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the workspace ID of the workspace that you want to manage.
5. Choose the **Tags** tab.
6. Choose **Manage tags**.
7. Find the tag that you want to delete, and choose **Remove**.

## Remove a tag from an Amazon Managed Service for Prometheus workspace (AWS CLI)

Follow these steps to use the AWS CLI to remove a tag from an workspace. Removing a tag does not delete it, but simply removes the association between the tag and the workspace.

> **Note**
>
> If you delete an Amazon Managed Service for Prometheus workspace, all tag associations are removed from the deleted workspace. You do not have to remove tags before you delete a workspace.

At the terminal or command line, run the **untag-resource** command, specifying the Amazon Resource Name (ARN) of the workspace where you want to remove tags and the tag key of the tag you want to remove. For example, to remove a tag on a workspace named **My-Workspace** with the tag key *Status*:

```
aws amp untag-resource --resource-arn arn:aws:aps:us-
west-2:123456789012:workspace/IDstring --tag-keys Status
```

If successful, this command returns nothing. To verify the tags associated with the workspace, run the **list-tags-for-resource** command.

# Tagging rule groups namespaces

Tags are custom labels that can be assigned to a resource. They include a unique key and an optional value (in a key-value pair). Tags help you identify and organize your AWS resources. In Amazon Managed Service for Prometheus, rule groups namespaces (and workspaces) can be tagged. You can use the console, the AWS CLI, APIs, or SDKs to add, manage, and remove tags for these resources. In addition to identifying, organizing, and tracking your rule groups namespaces with tags, you can use tags in IAM policies to help control who can view and interact with your Amazon Managed Service for Prometheus resources.

Use the procedures in this section to work with tags for Amazon Managed Service for Prometheus rule groups namespaces.

**Topics**


- Add a tag to a rule groups namespace
- View tags for a rule groups namespace
- Edit tags for a rule groups namespace
- Remove a tag from a rule groups namespace


## Add a tag to a rule groups namespace

Adding tags to an Amazon Managed Service for Prometheus rule groups namespaces can help you identify and organize your AWS resources and manage access to them. First, you add one or more tags (key-value pairs) to a rule groups namespace. After you have tags, you can create IAM policies to manage access to the namespace based on these tags. You can use the the console or the AWS CLI to add tags to an Amazon Managed Service for Prometheus rule groups namespace.

> **Important**
>
> Adding tags to a rule groups namespace can impact access to that rule groups namespace. Before you add a tag, make sure to review any IAM policies that might use tags to control access to resources.

For more information about adding tags to a rule groups namespace when you create it, see Create a rules file.

**Topics**


- Add a tag to a rule groups namespace (console)
- Add a tag to a rule groups namespace (AWS CLI)


## Add a tag to a rule groups namespace (console)

You can use the console to add one or more tags to a Amazon Managed Service for Prometheus rule groups namespace.

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the workspace ID of the workspace that you want to manage.
5. Choose the **Rules management** tab.
6. Choose the button next to the namespace name and choose **Edit**.
7. Choose **Create tags**, **Add new tag**.
8. In **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.
9. (Optional) To add another tag, choose **Add new tag** again.
10. When you have finished adding tags, choose **Save changes**.

## Add a tag to a rule groups namespace (AWS CLI)

Follow these steps to use the AWS CLI to add a tag to an Amazon Managed Service for Prometheus rule groups namespace. To add a tag to a rule groups namespace when you create it, see Upload a rules configuration file to Amazon Managed Service for Prometheus.

In these steps, we assume that you have already installed a recent version of the AWS CLI or updated to the current version. For more information, see Installing the AWS Command Line Interface.

At the terminal or command line, run the **tag-resource** command, specifying the Amazon Resource Name (ARN) of the rule groups namespace where you want to add tags and the key and value of the tag you want to add. You can add more than one tag to an rule groups namespace. For

example, to tag an Amazon Managed Service for Prometheus namespace named **My-Workspace** with two tags, a tag key named *Status* with the tag value of *Secret*, and a tag key named *Team* with the tag value of *My-Team*:

```
aws amp tag-resource \
    --resource-arn arn:aws:aps:us-
west-2:123456789012:rulegroupsnamespace/IDstring/namespace_name \
    --tags Status=Secret,Team=My-Team
```

If successful, this command returns nothing.

## View tags for a rule groups namespace

Tags can help you identify and organize your AWS resources and manage access to them. For more information about tagging strategies, see Tagging AWS Resources.

### View tags for an Amazon Managed Service for Prometheus rule groups namespace (console)

You can use the console to view the tags associated with a Amazon Managed Service for Prometheus rule groups namespace.

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the workspace ID of the workspace that you want to manage.
5. Choose the **Rules management** tab.
6. Choose the namespace name.

### View tags for an Amazon Managed Service for Prometheus workspace (AWS CLI)

Follow these steps to use the AWS CLI to view the AWS tags for a rule groups namespace. If no tags have been added, the returned list is empty.

At the terminal or command line, run the **list-tags-for-resource** command. For example, to view a list of tag keys and tag values for a rule groups namespace:

```
aws amp list-tags-for-resource --resource-arn rn:aws:aps:us-
west-2:123456789012:rulegroupsnamespace/IDstring/namespace_name
```

If successful, this command returns information similar to the following:

```
{
    "tags": {
        "Status": "Secret",
        "Team": "My-Team"
    }
}
```

## Edit tags for a rule groups namespace

You can change the value for a tag associated with a rule groups namespace. You can also change the name of the key, which is equivalent to removing the current tag and adding a different one with the new name and the same value as the other key.

> **Important**
>
> Editing tags for an rule groups namespace can impact access to it. Before you edit the name (key) or value of a tag for a resource, make sure to review any IAM policies that might use the key or value for a tag to control access to resources.

### Edit a tag for an Amazon Managed Service for Prometheus rule groups namespace (console)

You can use the console to edit the tags associated with a Amazon Managed Service for Prometheus rule groups namespace.

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.
2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the workspace ID of the workspace that you want to manage.
5. Choose the **Rules management** tab.

6. Choose the name of the namespace.
7. Choose **Manage tags**, **Add new tag**.
8. To change the value of an existing tag, enter the new value for **Value**.
9. o add an additional tag, choose **Add new tag**.
10. When you have finished adding and editing tags, choose **Save changes**.

### Edit tags for an Amazon Managed Service for Prometheus rule groups namespace (AWS CLI)

Follow these steps to use the AWS CLI to update a tag for a rule groups namespace. You can change the value for an existing key, or add another key.

At the terminal or command line, run the **tag-resource** command, specifying the Amazon Resource Name (ARN) of the resource where you want to update a tag and specify the tag key and tag value:

```
aws amp tag-resource --resource-arn rn:aws:aps:us-
west-2:123456789012:rulegroupsnamespace/IDstring/namespace_name --tags Team=New-Team
```

## Remove a tag from a rule groups namespace

You can remove one or more tags associated with a rule groups namespace. Removing a tag does not delete the tag from other AWS resources that are associated with that tag.

> **Important**
>
> Removing tags for a resource can impact access to that resource. Before you remove a tag from a resource, make sure to review any IAM policies that might use the key or value for a tag to control access to resources such as repositories.

### Remove a tag from an Amazon Managed Service for Prometheus rule groups namespace (console)

You can use the console to remove the association between a tag and a rule groups namespace.

1. Open the Amazon Managed Service for Prometheus console at https://eusc-de-east-1.console.amazonaws-eusc.eu/prometheus/.

2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the workspace ID of the workspace that you want to manage.
5. Choose the **Rules management** tab.
6. Choose the name of the namespace.
7. Choose **Manage tags**.
8. Next to the tag you want to delete, choose **Remove**.
9. When you have finished, choose **Save changes**.

## Remove a tag from an Amazon Managed Service for Prometheus rule groups namespace (AWS CLI)

Follow these steps to use the AWS CLI to remove a tag from an rule groups namespace. Removing a tag does not delete it, but simply removes the association between the tag and the rule groups namespace.

> **Note**
>
> If you delete an Amazon Managed Service for Prometheus rule groups namespace, all tag associations are removed from the deleted nnamespace. You do not have to remove tags before you delete a namespace.

At the terminal or command line, run the **untag-resource** command, specifying the Amazon Resource Name (ARN) of the rule groups namespace where you want to remove tags and the tag key of the tag you want to remove. For example, to remove a tag on a workspace named **My-Workspace** with the tag key *Status*:

```
aws amp untag-resource --resource-arn rn:aws:aps:us-
west-2:123456789012:rulegroupsnamespace/IDstring/namespace_name --tag-keys Status
```

If successful, this command returns nothing. To verify the tags associated with the resource, run the **list-tags-for-resource** command.

# Amazon Managed Service for Prometheus service quotas

The following two sections describe the quotas and limits associated with Amazon Managed Service for Prometheus.

## Service quotas

Amazon Managed Service for Prometheus has the following quotas. Amazon Managed Service for Prometheus vends CloudWatch usage metrics to monitor Prometheus resource usage. Using the Amazon CloudWatch usage metrics alarm feature, you can monitor Prometheus resources and usage to prevent limit errors.

As your projects and workspaces grow, the most common quotas that you should monitor or request an increase for are: **Active series per workspace**, and **Ingestion rate per workspace**.

For all adjustable quotas, you can request a quota increase by choosing the link in the **Adjustable** column, or by requesting a quota increase.

The *Active series per workspace* limit is dynamically applied. For more information, see Active series default quotas. The *Ingestion rate per workspace* quota determines how quickly you can ingest data into your workspace. For more information see Ingestion throttling.

> **Note**
>
> Unless otherwise noted, these quotas are per workspace. The maximum value for active series per workspace is 2 billion.

| Name | Default | Adjuse | Description |
|---|---|---|---|
| Active metrics with metadata per workspace | Each supported Region: 20,000 | No | The number of unique active metrics with metadata per workspace . Note: If the limit is reached, metric sample is |

| Name | Default | Adjustable | Description |
|---|---|---|---|
| | | | recorded, but metadata over the limit is dropped. |
| Active series per workspace | Each supported Region: 50,000,000 | Yes | The number of unique active series per workspace (up to a maximum of 2 billion). A series is active if a sample has been reported in the past 2 hours. Capacity from 2 M to 50 M is automatically adjusted based on the last 30 min of usage. |
| Alert aggregation group size in alert manager definition file | Each supported Region: 1,000 | Yes | The maximum size of an alert aggregation group in alert manager definition file. Each label value combination of group_by would create an aggregation group. |
| Alert manager definition file size | Each supported Region: 1,000,000 | No | The maximum size of an alert manager definition file, in bytes. |
| Alert payload size in Alert Manager | Each supported Region: 20,000,000 | No | The maximum alert payload size of all Alert Manager alerts per workspace, in bytes. Alert size is dependent on labels and annotations. |

| Name | Default | Adjustable | Description |
|---|---|---|---|
| Alerts in Alert Manager | Each supported Region: 1,000 | Yes | The maximum number of concurrent Alert Manager alerts per workspace. |
| HA tracker clusters | Each supported Region: 500 | No | The maximum number of clusters that HA tracker will keep track of for ingested samples per workspace. |
| Ingestion rate per workspace | Each supported Region: 1,666,666 | Yes | Metric sample ingestion rate per workspace per second. The limit is automatically adjusted to be 1/30 of the active series per workspace limit, up to 1,666,666. |
| Inhibition rules in alert manager definition file | Each supported Region: 100 | Yes | The maximum number of inhibition rules in alert manager definition file. |
| Label size | Each supported Region: 7 | No | The maximum combined size of all labels and label values accepted for a series, in kilobytes. |
| LabelSet limits per workspace | Each supported Region: 100 | Yes | The maximum number of labelset limits that can be created per workspace . |
| Labels per metric series | Each supported Region: 150 | Yes | Number of labels per metric series. |

| Name | Default | Adjustable | Description |
|---|---|---|---|
| Max Native Histogram bucket count | Each supported Region: 200 | No | The maximum total number of buckets allowed in a Native Histogram sample. The resolution of a histogram with more buckets will be reduced until the number of buckets is within the limit. |
| Metadata length | Each supported Region: 1 | No | The maximum length accepted for metric metadata, in kilobytes. Metadata refers to Metric Name, Type, Unit and Help Text. |
| Metadata per metric | Each supported Region: 10 | No | The number of metadata per metric. Note: If the limit is reached, metric sample is recorded, but metadata over the limit is dropped. |

| Name | Default | Adjustable | Description |
|---|---|---|---|
| Native Histogram active series per workspace | Each supported Region: 5,000,000 | Yes | The number of unique Native Histogram active series per workspace . A series is active if a sample has been reported in the past 2 hours. The limit is automatically adjusted to be 10-percent of the workspaces active series limit, or the default, whichever is greater. |
| Native Histogram ingestion rate per workspace | Each supported Region: 166,666 | Yes | Native Histogram sample ingestion rate per workspace per second. The limit is automatic ally adjusted to be 10-percent of the workspace s ingestion rate limit, or the default, whichever is greater. |
| Nodes in alert manager routing tree | Each supported Region: 100 | Yes | The maximum number of nodes in the alert manager routing tree. |

| Name | Default | Adjustable | Description |
| --- | --- | --- | --- |
| Number of API operations per region in transactions per second | Each supported Region: 10 | Yes | The maximum number of API operations per second per region for all Amazon Managed Service for Prometheus APIs, including workspace CRUD APIs, tagging APIs, rule groups namespace CRUD APIs, and alert manager definition CRUD APIs. |
| Number of GetSeries, GetLabels and GetMetricMetadata API operations per workspace in transactions per second | Each supported Region: 10 | No | The maximum number of GetSeries, GetLabels and GetMetricMetadata Prometheus-compatible API operations per second per workspace. |
| Number of QueryMetrics API operations per workspace in transactions per second | Each supported Region: 300 | Yes | The maximum number of QueryMetrics Prometheus-compatible API operations per second per workspace. |
| Number of RemoteWrite API operations per workspace in transactions per second | Each supported Region: 3,000 | Yes | The maximum number of RemoteWrite Prometheus-compatible API operations per second per workspace. |

| Name | Default | Adjustable | Description |
|---|---|---|---|
| Number of other Prometheus-compatible API operations per workspace in transactions per second | Each supported Region: 100 | No | The maximum number of API operations per second per workspace for all other Prometheus-compatible APIs including ListAlerts, ListRules, etc. |
| Out of order ingestion rate per workspace | Each supported Region: 83,333 | Yes | Out of order samples ingestion rate per workspace per second. Unless overwritten, the limit is automatically adjusted to be 5-percent of the ingestion rate per workspace limit. |
| Out of order time window per workspace | Each supported Region: 600 | Yes | The maximum time window for out of order samples per workspace, in seconds. The default out-of-order time window when a new workspace is created is 60 seconds, and it can be configured up to a maximum of 600 seconds. |
| Query bytes for instant queries | Each supported Region: 5 | No | The maximum bytes that can be scanned by a single instant query, in gigabytes. |

| Name | Default | Adjustable | Description |
|---|---|---|---|
| Query bytes for range queries | Each supported Region: 5 | No | The maximum bytes that can be scanned per 24-hour interval in a single range query, in gigabytes. |
| Query samples | Each supported Region: 50,000,000 | No | The maximum number of samples that can be scanned per 24-hour interval in a single range query, or a single instant query. |
| Query series fetched | Each supported Region: 12,000,000 | No | The maximum number of series that can be scanned per 24-hour interval in a single range query, or a single instant query. |
| Query time range in days | Each supported Region: 95 | No | The maximum time range of QueryMetrics, GetSeries, and GetLabels APIs. |
| Request size | Each supported Region: 1 | No | The maximum request size for ingestion or query, in megabytes. |
| Rule evaluation interval | Each supported Region: 30 | Yes | The minimum rule evaluation interval of a rule group per workspace, in seconds. |

| Name | Default | Adjustable | Description |
|---|---|---|---|
| Rule group namespace definition file size | Each supported Region: 1,000,000 | No | The maximum size of a rule group namespace definition file, in bytes. |
| Rules per workspace | Each supported Region: 2,000 | Yes | The maximum number of rules per workspace. |
| Silences per workspace | Each supported Region: 1,000 | Yes | Maximum number of silences, including expired, active, and pending silences, per workspace. |
| Templates in alert manager definition file | Each supported Region: 100 | Yes | The maximum number of templates in the alert manager definition file. |
| Workspaces per region per account | Each supported Region: 25 | Yes | The maximum number of workspaces per region. |

## Active series default quotas

Amazon Managed Service for Prometheus workspaces automatically adapt to your ingestion usage. As your usage increases, the service automatically increases your time series capacity up to the default quota.

Your Amazon Managed Service for Prometheus workspace scales automatically, based on your usage, in two ways:

1. When your 30-minute average usage is below 5 million series, the capacity doubles (for example, a workspace with 3.5M usage gets 7M capacity).
2. When usage exceeds 5 million series, the workspace adds a 10 million buffer (for example, a workspace with 25M usage gets 35M capacity).

Amazon Managed Service for Prometheus automatically allocates more capacity as your ingestion increases, up to your quota. This helps ensure your workload does not experience sustained throttling. However, throttling can occur if you double or exceed 10 million above your previous baseline computed over the last 30 minutes. To avoid throttling, Amazon Managed Service for Prometheus recommends gradually increasing ingestion when increasing beyond your previous baseline.

> **Note**
>
> The minimum capacity for active time series is 2 million, and there is no throttling when you have less than 2 million series.
> To go beyond your default quota, you can request a quota increase.

## Scaling above the default quota

When you request a quota increase above the default active series quota, Amazon Managed Service for Prometheus adjusts your workspace capacity accordingly. If you don't fully utilize the increased capacity, the service will reclaim the unused portion over time. As your usage grows, the workspace will scale up again automatically.

However, throttling can occur if you more than double or exceed 50 million active time series over your previous baseline computed from the last 2 hours. For example:

- If your quota is 100 million and your baseline is 30 million, you can scale up to 60 million within 2 hours without throttling.
- If your quota is 100 million and your baseline is 50 million, you can scale up to the full 100 million within 2 hours without throttling.

## Ingestion throttling

Amazon Managed Service for Prometheus throttles ingestion for each workspace, based on your current limits. This helps maintain the performance of the workspace. If you exceed the limit, you will see `DiscardedSamples` in CloudWatch metrics (with the `rate_limited` reason). You can use CloudWatch to monitor your ingestion, and to create an alarm to warn you when you are close to reaching the throttling limits. For more information, see Use CloudWatch metrics to monitor Amazon Managed Service for Prometheus resources.

Amazon Managed Service for Prometheus uses the token bucket algorithm to implement ingestion throttling. With this algorithm, your account has a *bucket* that holds a specific number of *tokens*. The number of tokens in the bucket represents your ingestion limit at any given second.

Each data sample ingested removes one token from the bucket. If your bucket size (*Ingestion rate per workspace*) is *1,000,000*, your workspace can ingest one million data samples in one second. If it exceeds one million samples to ingest, it will be throttled, and will not ingest any more records. Additional data samples will be discarded.

The bucket automatically refills at a set rate. If the bucket is below its maximum capacity, a set number of tokens is added back to it every second until it reaches its maximum capacity. If the bucket is full when the refill tokens arrive, they are discarded. The bucket can't hold more than its maximum number of tokens. The refill rate for sample ingestion is set by the *Ingestion rate per workspace* limit. If your *Ingestion rate per workspace* is set to 170,000, then the refill rate for the bucket is 170,000 tokens per second.

If your workspace ingests 1,000,000 data samples in a second, your bucket is immediately reduced to zero tokens. The bucket is then refilled by 170,000 tokens every second, until it reaches it's maximum capacity of 1,000,000 tokens. If there is no more ingestion, the previously empty bucket will return to it's maximum capacity in 6 seconds.

> **Note**
>
> Ingestion happens in batched requests. If you have 100 tokens available, and send a request with 101 samples, the entire request is rejected. Amazon Managed Service for Prometheus does not partially accept requests. If you are writing a collector, you can manage retries (with smaller batches or after some time has passed).

You do not need to wait for the bucket to be full before your workspace can ingest more data samples. You can use tokens as they are added to the bucket. If you immediately use the refill tokens, the bucket does not reach its maximum capacity. For example, if you deplete the bucket, you can continue to ingest 170,000 data samples per second. The bucket can refill to maximum capacity only if you ingest fewer than 170,000 data samples per second.

## Additional limits on ingested data

Amazon Managed Service for Prometheus also has the following additional requirements for data ingested into the workspace. These are not adjustable.

- Metric samples older than 1 hour are refused from being ingested.
- Every sample and metadata must have a metric name.
- Native histogram samples must have a schema value between -4 and 8. Samples with a schema outside this range are discarded.
- Native histogram samples must not exceed 2048 bytes in size. Samples that exceed this limit are discarded.

# Amazon Managed Service for Prometheus API Reference

Amazon Managed Service for Prometheus offers two types of APIs:

1. **Amazon Managed Service for Prometheus APIs** – These APIs allow you to create and manage your Amazon Managed Service for Prometheus workspaces, including operations for workspaces, scrapers, alert manager definitions, rule groups namespaces, and logging. You use the AWS SDKs, available for various programming languages, to interact with these APIs.
2. **Prometheus-compatible APIs** – Amazon Managed Service for Prometheus supports HTTP APIs that are compatible with Prometheus. These APIs enable building custom applications, automate workflows, integrate with other services or tools, and query and interact with your monitoring data using the Prometheus query language (PromQL).

This section lists the API operations and data structures supported by Amazon Managed Service for Prometheus.

For information about quotas for the series, labels, and API requests, see Amazon Managed Service for Prometheus service quotas in the *Amazon Managed Service for Prometheus User Guide*.

**Topics**


- Amazon Managed Service for Prometheus APIs
- Prometheus-compatible APIs


# Amazon Managed Service for Prometheus APIs

Amazon Managed Service for Prometheus provides API operations creating and maintaining your Amazon Managed Service for Prometheus workspaces. This includes APIs for workspaces, scrapers, alert manager definitions, rule groups namespaces, and logging.

For detailed information about the Amazon Managed Service for Prometheus APIs, see the Amazon Managed Service for Prometheus API Reference.

## Using Amazon Managed Service for Prometheus with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that makes it easier for developers

to build AWS applications in their preferred language. For a list of SDKs and tools by language, see Tools to Build on AWS in the AWS *Developer Center*.

> **SDK Versions**
>
> We recommend that you use the most recent build of the AWS SDK, and any other SDKs, that you use in your projects, and to keep the SDKs up to date. The AWS SDK provides you with the latest features and functionality, and also security updates.

# Prometheus-compatible APIs

Amazon Managed Service for Prometheus supports the following Prometheus-compatible APIs.

For more information about using Prometheus-compatible APIs, see Query using Prometheus-compatible APIs.

**Topics**

- CreateAlertManagerAlerts
- DeleteAlertManagerSilence
- GetAlertManagerStatus
- GetAlertManagerSilence
- GetLabels
- GetMetricMetadata
- GetSeries
- ListAlerts
- ListAlertManagerAlerts
- ListAlertManagerAlertGroups
- ListAlertManagerReceivers
- ListAlertManagerSilences
- ListRules
- PutAlertManagerSilences
- QueryMetrics
- RemoteWrite

# CreateAlertManagerAlerts

The `CreateAlertManagerAlerts` operation creates an alert in the workspace.

Valid HTTP verbs:

POST

Valid URIs:

`/workspaces/workspaceId/alertmanager/api/v2/alerts`

URL query parameters:

`alerts` An array of objects, where each object represents one alert. The following is an example of an alert object:

```
[
  {
    "startsAt": "2021-09-24T17:14:04.995Z",
    "endsAt": "2021-09-24T17:14:04.995Z",
    "annotations": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "labels": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "generatorURL": "string"
  }
]
```

### Sample request

```
POST /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/alertmanager/api/v2/alerts
 HTTP/1.1
Content-Length: 203,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

```
[
  {
    "labels": {
      "alertname": "test-alert"
    },
    "annotations": {
      "summary": "this is a test alert used for demo purposes"
    },
    "generatorURL": "https://www.amazon.com/"
  }
]
```

**Sample response**

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 0
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin
```

## DeleteAlertManagerSilence

The `DeleteSilence` deletes one alert silence.

Valid HTTP verbs:

`DELETE`

Valid URIs:

`/workspaces/workspaceId/alertmanager/api/v2/silence/silenceID`

URL query parameters: none

**Sample request**

```
DELETE /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/alertmanager/api/v2/silence/
d29d9df3-9125-4441-912c-70b05f86f973 HTTP/1.1
```

```
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 0
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin
```

## GetAlertManagerStatus

The `GetAlertManagerStatus` retrieves information about the status of alert manager.

Valid HTTP verbs:

`GET`

Valid URIs:

`/workspaces/workspaceId/alertmanager/api/v2/status`

URL query parameters: none

### Sample request

```
GET /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/alertmanager/api/v2/status
 HTTP/1.1
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response

```
HTTP/1.1 200 OK
```

```
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 941
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin

{
    "cluster": null,
    "config": {
        "original": "global:\n  resolve_timeout: 5m\n  http_config:\n
 follow_redirects: true\n  smtp_hello: localhost\n  smtp_require_tls: true\nroute:
\n  receiver: sns-0\n  group_by:\n  - label\n  continue: false\nreceivers:\n-
 name: sns-0\n  sns_configs:\n  - send_resolved: false\n    http_config:\n
      follow_redirects: true\n    sigv4: {}\n    topic_arn: arn:aws:sns:us-
west-2:123456789012:test\n    subject: '{{ template \"sns.default.subject\" . }}'\n
    message: '{{ template \"sns.default.message\" . }}'\n    workspace_arn:
 arn:aws:aps:us-west-2:123456789012:workspace/ws-58a6a446-5ec4-415b-9052-a449073bbd0a
\ntemplates: []\n"
    },
    "uptime": null,
    "versionInfo": null
}
```

## GetAlertManagerSilence

The `GetAlertManagerSilence` retrieves information about one alert silence.

Valid HTTP verbs:

`GET`

Valid URIs:

`/workspaces/`*`workspaceId`*`/alertmanager/api/v2/silence/`*`silenceID`*

URL query parameters: none

### Sample request

```
GET /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/alertmanager/api/v2/silence/
d29d9df3-9125-4441-912c-70b05f86f973 HTTP/1.1
```

```
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 310
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin

{
    "id": "d29d9df3-9125-4441-912c-70b05f86f973",
    "status": {
        "state": "active"
    },
    "updatedAt": "2021-10-22T19:32:11.763Z",
    "comment": "hello-world",
    "createdBy": "test-person",
    "endsAt": "2023-07-24T01:05:36.000Z",
    "matchers": [
        {
            "isEqual": true,
            "isRegex": true,
            "name": "job",
            "value": "hello"
        }
    ],
    "startsAt": "2021-10-22T19:32:11.763Z"
}
```

## GetLabels

The `GetLabels` operation retrieves the labels associated with a time series.

Valid HTTP verbs:

GET, POST

Valid URIs:

`/workspaces/workspaceId/api/v1/labels`

`/workspaces/workspaceId/api/v1/label/label-name/values` This URI supports only GET requests.

URL query parameters:

`match[]=<series_selector>` Repeated series selector argument that selects the series from which to read the label names. Optional.

`start=<rfc3339 | unix_timestamp>` Start timestamp. Optional.

`end=<rfc3339 | unix_timestamp>` End timestamp. Optional.

### Sample request for /workspaces/workspaceId/api/v1/labels

```
GET /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/api/v1/labels HTTP/1.1
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response for /workspaces/workspaceId/api/v1/labels

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 1435
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin

{
    "status": "success",
    "data": [
        "__name__",
        "access_mode",
        "address",
        "alertname",
```

```
        "alertstate",
        "apiservice",
        "app",
        "app_kubernetes_io_instance",
        "app_kubernetes_io_managed_by",
        "app_kubernetes_io_name",
        "area",
        "beta_kubernetes_io_arch",
        "beta_kubernetes_io_instance_type",
        "beta_kubernetes_io_os",
        "boot_id",
        "branch",
        "broadcast",
        "buildDate",
        ...
    ]
}
```

### Sample request for /workspaces/workspaceId/api/v1/label/label-name/values

```
GET /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/api/v1/label/access_mode/values
 HTTP/1.1
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response for /workspaces/workspaceId/api/v1/label/label-name/values

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 74
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin

{
    "status": "success",
    "data": [
        "ReadWriteOnce"
    ]
```

```
}
```

## GetMetricMetadata

The `GetMetricMetadata` operation retrieves metadata about metrics that are currently being scraped from targets. It does not provide any target information.

The data section of the query result consists of an object where each key is a metric name and each value is a list of unique metadata objects, as exposed for that metric name across all targets.

Valid HTTP verbs:

`GET`

Valid URIs:

`/workspaces/workspaceId/api/v1/metadata`

URL query parameters:

`limit=<number>` The maximum number of metrics to return.

`metric=<string>` A metric name to filter metadata for. If you keep this empty, all metric metadata is retrieved.

### Sample request

```
GET /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/api/v1/metadata HTTP/1.1
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
Transfer-Encoding: chunked
```

```
{
    "status": "success",
    "data": {
        "aggregator_openapi_v2_regeneration_count": [
            {
                "type": "counter",
                "help": "[ALPHA] Counter of OpenAPI v2 spec regeneration count broken
 down by causing APIService name and reason.",
                "unit": ""
            }
        ],
        ...
    }
}
```

## GetSeries

The `GetSeries` operation retrieves list of time series that match a certain label set.

Valid HTTP verbs:

`GET`, `POST`

Valid URIs:

`/workspaces/`*`workspaceId`*`/api/v1/series`

URL query parameters:

`match[]=<series_selector>` Repeated series selector argument that selects the series to return. At least one `match[]` argument must be provided.

`start=<rfc3339 | unix_timestamp>` Start timestamp. Optional

`end=<rfc3339 | unix_timestamp>` End timestamp. Optional

### Sample request

```
POST /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/api/v1/series --data-urlencode
 'match[]=node_cpu_seconds_total{app="prometheus"}' --data-urlencode 'start=1634936400'
 --data-urlencode 'end=1634939100' HTTP/1.1
Content-Length: 0,
```

```
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
content-encoding: gzip

{
    "status": "success",
    "data": [
        {
            "__name__": "node_cpu_seconds_total",
            "app": "prometheus",
            "app_kubernetes_io_managed_by": "Helm",
            "chart": "prometheus-11.12.1",
            "cluster": "cluster-1",
            "component": "node-exporter",
            "cpu": "0",
            "heritage": "Helm",
            "instance": "10.0.100.36:9100",
            "job": "kubernetes-service-endpoints",
            "kubernetes_name": "servicesstackprometheuscf14a6d7-node-exporter",
            "kubernetes_namespace": "default",
            "kubernetes_node": "ip-10-0-100-36.us-west-2.compute.internal",
            "mode": "idle",
            "release": "servicesstackprometheuscf14a6d7"
        },
        {
            "__name__": "node_cpu_seconds_total",
            "app": "prometheus",
            "app_kubernetes_io_managed_by": "Helm",
            "chart": "prometheus-11.12.1",
            "cluster": "cluster-1",
            "component": "node-exporter",
            "cpu": "0",
            "heritage": "Helm",
```

```
            "instance": "10.0.100.36:9100",
            "job": "kubernetes-service-endpoints",
            "kubernetes_name": "servicesstackprometheuscf14a6d7-node-exporter",
            "kubernetes_namespace": "default",
            "kubernetes_node": "ip-10-0-100-36.us-west-2.compute.internal",
            "mode": "iowait",
            "release": "servicesstackprometheuscf14a6d7"
        },
        ...
    ]
}
```

## ListAlerts

The `ListAlerts` operation retrieves currently active alerts in the workspace.

Valid HTTP verbs:

`GET`

Valid URIs:

`/workspaces/workspaceId/api/v1/alerts`

### Sample request

```
GET /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/api/v1/alerts HTTP/1.1
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 386
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin
```

```
{
  "status": "success",
  "data": {
    "alerts": [
      {
        "labels": {
          "alertname": "test-1.alert",
          "severity": "none"
        },
        "annotations": {
          "message": "message"
        },
        "state": "firing",
        "activeAt": "2020-12-01T19:37:25.429565909Z",
        "value": "1e+00"
      }
    ]
  },
  "errorType": "",
  "error": ""
}
```

## ListAlertManagerAlerts

The `ListAlertManagerAlerts` retrieves information about the alerts currently firing in alert manager in the workspace.

Valid HTTP verbs:

`GET`

Valid URIs:

`/workspaces/`*`workspaceId`*`/alertmanager/api/v2/alerts`

URL query parameters:

`active` Boolean. If true, the returned list includes active alerts. The default is true. Optional

`silenced` Boolean. If true, the returned list includes silenced alerts. The default is true. Optional

`inhibited` Boolean. If true, the returned list includes inhibited alerts. The default is true. Optional

`filter` An array of strings. A list of matchers to filter alerts by. Optional

`receiver` String. A regular expression matching receivers to filter alerts by. Optional

### Sample request

```
GET /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/alertmanager/api/v2/alerts
 HTTP/1.1
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 354
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin

[
    {
        "annotations": {
            "summary": "this is a test alert used for demo purposes"
        },
        "endsAt": "2021-10-21T22:07:31.501Z",
        "fingerprint": "375eab7b59892505",
        "receivers": [
            {
                "name": "sns-0"
            }
        ],
        "startsAt": "2021-10-21T22:02:31.501Z",
        "status": {
            "inhibitedBy": [],
            "silencedBy": [],
            "state": "active"
        },
```

```
        "updatedAt": "2021-10-21T22:02:31.501Z",
        "labels": {
            "alertname": "test-alert"
        }
    }
]
```

## ListAlertManagerAlertGroups

The `ListAlertManagerAlertGroups` operation retrieves a list of alert groups configured in alert manager in the workspace.

Valid HTTP verbs:

`GET`

Valid URIs:

`/workspaces/`*`workspaceId`*`/alertmanager/api/v2/alerts/groups`

URL query parameters:

`active` Boolean. If true, the returned list includes active alerts. The default is true. Optional

`silenced` Boolean. If true, the returned list includes silenced alerts. The default is true. Optional

`inhibited` Boolean. If true, the returned list includes inhibited alerts. The default is true. Optional

`muted` Boolean. If true, the returned list includes muted alerts. The default is true. Optional

`filter` An array of strings. A list of matchers to filter alerts by. Optional

`receiver` String. A regular expression matching receivers to filter alerts by. Optional

### Sample request

```
GET /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/alertmanager/api/v2/alerts/
groups HTTP/1.1
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
```

```
User-Agent: Grafana/8.1.0
```

### Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 443
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin

[
    {
        "alerts": [
            {
                "annotations": {
                    "summary": "this is a test alert used for demo purposes"
                },
                "endsAt": "2021-10-21T22:07:31.501Z",
                "fingerprint": "375eab7b59892505",
                "receivers": [
                    {
                        "name": "sns-0"
                    }
                ],
                "startsAt": "2021-10-21T22:02:31.501Z",
                "status": {
                    "inhibitedBy": [],
                    "silencedBy": [],
                    "state": "unprocessed"
                },
                "updatedAt": "2021-10-21T22:02:31.501Z",
                "generatorURL": "https://www.amazon.com/",
                "labels": {
                    "alertname": "test-alert"
                }
            }
        ],
        "labels": {},
        "receiver": {
            "name": "sns-0"
```

```
        }
    }
]
```

## ListAlertManagerReceivers

The `ListAlertManagerReceivers` operation retrieves information about the receivers configured in alert manager.

Valid HTTP verbs:

`GET`

Valid URIs:

`/workspaces/workspaceId/alertmanager/api/v2/receivers`

URL query parameters: none

### Sample request

```
GET /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/alertmanager/api/v2/receivers
 HTTP/1.1
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 19
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin

[
    {
        "name": "sns-0"
```

```
    }
]
```

## ListAlertManagerSilences

The `ListAlertManagerSilences` operation retrieves information about the alert silences configured in the workspace.

Valid HTTP verbs:

`GET`

Valid URIs:

`/workspaces/workspaceId/alertmanager/api/v2/silences`

URL query parameters:

`filter` An array of strings. A list of matchers to filter silences by. Optional

### Sample request

```
GET /workspaces/ws-58a6a446-5ec4-415b-9052-a449073bbd0a/alertmanager/api/v2/silences
 HTTP/1.1
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 312
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin

[
    {
```

```
        "id": "d29d9df3-9125-4441-912c-70b05f86f973",
        "status": {
            "state": "active"
        },
        "updatedAt": "2021-10-22T19:32:11.763Z",
        "comment": "hello-world",
        "createdBy": "test-person",
        "endsAt": "2023-07-24T01:05:36.000Z",
        "matchers": [
            {
                "isEqual": true,
                "isRegex": true,
                "name": "job",
                "value": "hello"
            }
        ],
        "startsAt": "2021-10-22T19:32:11.763Z"
    }
]
```

## ListRules

The `ListRules` operation retrieves information about the rules configured in the workspace.

A single response contains at most 100,000 rules. Because rule groups are never split across pages, truncation always lands on a rule group boundary. A response can therefore contain fewer than 100,000 rules if including the next rule group exceeds the cap. When truncation occurs, the response includes the `groupNextToken` property. To retrieve the next page, pass that value as the `group_next_token` query parameter, which you must send together with `group_limit`.

Valid HTTP verbs:

`GET`

Valid URIs:

`/workspaces/workspaceId/api/v1/rules`

URL query parameters:

`type=<alert|record>` Return only the alerting rules (for example, `type=alert`) or the recording rules (for example, `type=record`). When the parameter is absent or empty, no filtering is done. Optional

`rule_name[]=<string>` Filters rules using the provided names. When multiple filters are provided, the rule name must satisfy at least one of the given filters. Optional

`rule_group[]=<string>` Filters rule groups using the provided names. When multiple filters are provided, the rule group name must satisfy at least one of the given filters. Optional

`file[]=<string>` Filters rule group namespaces using the provided names. When multiple filters are provided, the rule group namespace name must satisfy at least one of the given filters. Optional

`exclude_alerts=<true|false>` Filters rules without any active alerts when set to `true`. Optional

`match[]=<series_selector>` Only return rules that have configured labels that satisfy the label selectors. If the parameter is repeated, rules that match any of the sets of label selectors are returned. Note that matching is on the labels in the definition of each rule, not on the values after template expansion (for alerting rules). Optional

`group_limit=<number>` Maximum number of rule groups returned in a single response. In subsequent requests to paginate, use the `groupNextToken` property from this response to set the `group_next_token` parameter. Optional

`group_next_token=<string>` The pagination token returned by a previous request when you set the `group_limit` property or when the response reaches the 100,000-rule cap. Use this token to iteratively paginate over the remaining rule groups. When you use this parameter, you must also send the `group_limit` parameter. Optional

### Sample request

```
GET /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/api/v1/rules HTTP/1.1
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 423
```

```
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin

{
    "status": "success",
    "data": {
        "groups": [
            {
                "name": "test-1.rules",
                "file": "test-rules",
                "rules": [
                    {
                        "name": "record:1",
                        "query": "sum(rate(node_cpu_seconds_total[10m:1m]))",
                        "labels": {},
                        "health": "ok",
                        "lastError": "",
                        "type": "recording",
                        "lastEvaluation": "2021-10-21T21:22:34.429565909Z",
                        "evaluationTime": 0.001005399
                    }
                ],
                "interval": 60,
                "lastEvaluation": "2021-10-21T21:22:34.429563992Z",
                "evaluationTime": 0.001010504
            }
        ]
    },
    "errorType": "",
    "error": ""
}
```

## PutAlertManagerSilences

The `PutAlertManagerSilences` operation creates a new alert silence or updates an existing one.

Valid HTTP verbs:

POST

Valid URIs:

`/workspaces/workspaceId/alertmanager/api/v2/silences`

URL query parameters:

`silence` An object that represents the silence. The following is the format:

```
{
  "id": "string",
  "matchers": [
    {
      "name": "string",
      "value": "string",
      "isRegex": Boolean,
      "isEqual": Boolean
    }
  ],
  "startsAt": "timestamp",
  "endsAt": "timestamp",
  "createdBy": "string",
  "comment": "string"
}
```

### Sample request

```
POST /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/alertmanager/api/v2/silences
 HTTP/1.1
Content-Length: 281,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0

{
   "matchers":[
      {
         "name":"job",
         "value":"up",
         "isRegex":false,
         "isEqual":true
      }
   ],
   "startsAt":"2020-07-23T01:05:36+00:00",
```

```
   "endsAt":"2023-07-24T01:05:36+00:00",
   "createdBy":"test-person",
   "comment":"test silence"
}
```

**Sample response**

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 53
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin

{
    "silenceID": "512860da-74f3-43c9-8833-cec026542b32"
}
```

## QueryMetrics

The `QueryMetrics` operation evaluates an instant query at a single point in time or over a range of time.

Valid HTTP verbs:

`GET`, `POST`

Valid URIs:

`/workspaces/`*`workspaceId`*`/api/v1/query` This URI evaluates an instant query at a single point in time.

`/workspaces/`*`workspaceId`*`/api/v1/query_range` This URI evaluates an instant query over a range of time.

URL query parameters:

`query=<string>` A Prometheus expression query string. Used in both `query` and `query_range`.

`time=<rfc3339 | unix_timestamp>` (Optional) Evaluation timestamp if you are using the `query` for an instant query at a single point in time.

`timeout=<duration>` (Optional) Evaluation timeout. Defaults to and is capped by the value of the `-query.timeout` flag. Used in both `query` and `query_range`.

`start=<rfc3339 | unix_timestamp>` Start timestamp if you are using `query_range` to query for a range of time.

`end=<rfc3339 | unix_timestamp>` End timestamp if you are using `query_range` to query for a range of time.

`step=<duration | float>` Query resolution step width in `duration` format or as a `float` number of seconds. Use only if you are using `query_range` to query for a range of time, and required for such queries.

`max_samples_processed_warning_threshold=<integer>` (Optional) Sets the warning threshold for Query Samples Processed (QSP). When queries hit this threshold, a warning message will be returned in the API response.

`max_samples_processed_error_threshold=<integer>>` (Optional) Sets the error threshold for Query Samples Processed (QSP). Queries that exceed this threshold will be rejected with an error and will not be charged. Used to prevent excessive query costs.

**Duration**

A `duration` in a Prometheus-compatible API is a number, followed immediately by one of the following units:

- `ms` milliseconds
- `s` seconds
- `m` minutes
- `h` hours
- `d` days, assuming a day always has 24h
- `w` weeks, assuming a week always has 7d
- `y` years, assuming a year always has 365d

**Sample request**

```
POST /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/api/v1/query?
query=sum(node_cpu_seconds_total) HTTP/1.1
```

```
Content-Length: 0,
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Grafana/8.1.0
```

### Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 132
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
content-encoding: gzip

{
    "status": "success",
    "data": {
        "resultType": "vector",
        "result": [
            {
                "metric": {},
                "value": [
                    1634937046.322,
                    "252590622.81000024"
                ]
            }
        ]
    }
}
```

## RemoteWrite

The `RemoteWrite` operation writes metrics from a Prometheus server to a remote URL in a standardized format. Typically, you will use an existing client such as a Prometheus server to call this operation.

Valid HTTP verbs:

- POST

Valid URIs:

`/workspaces/workspaceId/api/v1/remote_write`

URL query parameters:

None

`RemoteWrite` has an ingestion rate of 70,000 samples per second and ingestion burst size of 1,000,000 samples.

### Sample request

```
POST /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568/api/v1/remote_write --data-
binary "@real-dataset.sz" HTTP/1.1
Authorization: AUTHPARAMS
X-Amz-Date: 20201201T193725Z
User-Agent: Prometheus/2.20.1
Content-Type: application/x-protobuf
Content-Encoding: snappy
X-Prometheus-Remote-Write-Version: 0.1.0

body
```

**Note**

For the request body syntax, see to the protocol buffer definition at https://github.com/prometheus/prometheus/blob/1c624c58ca934f618be737b4995e22051f5724c1/prompb/remote.pb.go#L64.

### Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length:0
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Content-Type: application/json
Server: amazon
vary: Origin
```

# Document History for Amazon Managed Service for Prometheus User Guide

The following table describes important documentation updates in the Amazon Managed Service for Prometheus User Guide. For notification about updates to this documentation, you can subscribe to an RSS feed.

| Change | Description | Date |
| --- | --- | --- |
| Increased active series limit | To support larger workloads, Amazon Managed Service for Prometheus has increased the maximum active series limit per workspace to 2 billion, so you can monitor more series without hitting quota limits. | November 17, 2026 |
| Updated the AmazonPrometheusScraperServiceRolePolicy managed IAM policy | AWS managed collectors can now write collected metrics to Amazon CloudWatch Metrics datasets as a new destination type. | July 14, 2026 |
| Added native histogram support | Amazon Managed Service for Prometheus now supports ingestion, storage, and querying of Prometheus native histograms. New CloudWatch usage metrics `NativeHistogramIngestionRate` and `NativeHistogramActiveSeries` are available to monitor native histogram usage against quota limits. A new service metric, | June 10, 2026 |

| | | |
|---|---|---|
| | `NativeHistogramIngestedBucketsRate`, provides visibility into bucket consumption. For more information, see Monitor with CloudWatch metrics. | |
| Added out-of-order ingestion and rule query offset support | Amazon Managed Service for Prometheus now supports configuring an out-of-order time window to accept samples that arrive out of chronological order, and a rule query offset to introduce a global delay before rule evaluations. Two new CloudWatch metrics, `OutOfOrderIngestionRate` and `OutOfOrderSampleAge`, are also available to monitor out-of-order ingestion. For more information, see Configure your workspace. | May 14, 2026 |
| Launched support for PagerDuty | Amazon Managed Service for Prometheus adds support for PagerDuty integration that enables automated incident response workflows and ensures critical alerts reach the right team members at the right time. For more information, see Use PagerDuty as an alert receiver. | August 29, 2025 |

| | | |
|---|---|---|
| Added resource based policy support | The following API Actions are now available:<br>• DeleteResourcePolicy<br>• DescribeResourcePolicy<br>• PutResourcePolicy | August 15, 2025 |
| Update to the AmazonPrometheusConsoleFullAccess managed IAM policy. | The AmazonPrometheusConsoleFullAccess policy was updated. The `aps:CreateQueryLoggingConfiguration` , `aps:UpdateQueryLoggingConfiguration` , `aps:DeleteQueryLoggingConfiguration` , `aps:DescribeQueryLoggingConfiguration` permissions were added to the policy. | May 5, 2025 |
| Added editing of rules definition files and Alert manager configuration files in the console | Amazon Managed Service for Prometheus adds support for editing Alert manager configuration files and rules definition files from within the Amazon Managed Service for Prometheus console. | May 16, 2024 |

| | | |
|---|---|---|
| Added simpler AWS managed collector setup with access entries for Amazon EKS | Amazon Managed Service for Prometheus adds support for Amazon EKS access entries to simplify setting up AWS managed collectors. The AmazonPrometheusScraperServiceRolePolicy managed policy for AWS managed collectors is updated to allow deleting access entries that are no longer used. | May 2, 2024 |
| Move AWS API to a separate API reference guide | The Amazon Managed Service for Prometheus AWS APIs are now available in their own reference, the Amazon Managed Service for Prometheus API Reference. Prometheus-compatible APIs continue to be documented in the Amazon Managed Service for Prometheus User Guide. | February 7, 2024 |
| Added customer managed keys for workspace encryption | Amazon Managed Service for Prometheus adds support for customer managed keys for workspace encryption. For more information, see Encryption at rest. | December 21, 2023 |
| Added new permissions to AmazonPrometheusFullAccess | Added new permissions to the AmazonPrometheusFullAccess managed policy to support creating AWS managed collectors for Amazon EKS clusters. | November 26, 2023 |

| | | |
|---|---|---|
| Added new managed policy, AmazonPrometheusScraperServiceLinkedRolePolicy | Added a new managed policy, AmazonPrometheusScraperServiceLinkedRolePolicy for AWS managed collectors to collect metrics from Amazon EKS clusters. | November 26, 2023 |
| Added AWS managed collectors as ingestion method | Amazon Managed Service for Prometheus adds support for AWS managed collectors. | November 26, 2023 |
| Added support for integrating with Amazon Managed Grafana | Amazon Managed Service for Prometheus adds support for integrating with Amazon Managed Grafana alerts. | November 23, 2022 |
| Added new permissions to AmazonPrometheusConsoleFullAccess | Added new permissions to the AmazonPrometheusConsoleFullAccess managed policy to support logging alert manager and ruler events in CloudWatch Logs. | October 24, 2022 |
| Added Amazon EKS observability solution. | Amazon Managed Service for Prometheus adds a new solution using AWS Observability Accelerator. For more information, see Using AWS Observability Accelerator. | October 14, 2022 |
| Added support for integrating into Amazon EKS cost monitoring. | Amazon Managed Service for Prometheus adds support for integrating into Amazon EKS cost monitoring. For more information, see Integrating with Amazon EKS cost monitoring. | September 22, 2022 |

| | | |
|---|---|---|
| Launched support for Alert Manager and Ruler logs in Amazon CloudWatch Logs. | Amazon Managed Service for Prometheus launches support for Alert Manager and Ruler error logs in Amazon CloudWatch Logs. For more information, see Amazon CloudWatch Logs. | September 1, 2022 |
| Added custom storage retention support. | Amazon Managed Service for Prometheus adds custom storage retention support, per workspace, by modifying the quota for that workspace . For more information about quotas in Amazon Managed Service for Prometheus, see Service quotas. | August 12, 2022 |
| Added usage metrics to Amazon CloudWatch. | Amazon Managed Service for Prometheus adds support for sending usage metrics to Amazon CloudWatch. For more information, see Amazon CloudWatch metrics. | May 6, 2022 |
| Added support for the Europe (London) Region. | Amazon Managed Service for Prometheus adds support for the Europe (London) Region. | May 4, 2022 |
| Amazon Managed Service for Prometheus is generally available, and adds support for rules and alert manager. | Amazon Managed Service for Prometheus is generally available. It also supports rules and alert manager. For more information, see Recording rules and alerting rules and Alert manager and templating. | September 29, 2021 |

| | | |
|---|---|---|
| Tagging support added. | Amazon Managed Service for Prometheus supports tagging of Amazon Managed Service for Prometheus workspaces. | September 7, 2021 |
| Active series and ingestion rate quotas increased. | The active series quota increased to 1,000,000 and the ingestion rate quota increased to 70,000 samples per second. | February 22, 2021 |
| Amazon Managed Service for Prometheus preview release. | The preview of Amazon Managed Service for Prometheus is released. | December 15, 2020 |