Amazon Managed Service for Prometheus

User Guide

Amazon Managed Service for Prometheus: User Guide

Table of Contents

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, and are then automatically deleted.

Supported Regions

Amazon Managed Service for Prometheus currently supports the following Regions:

- US East (Ohio)
- US East (N. Virginia)
- US West (Oregon)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (Stockholm)

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 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.

Setting up

Complete the tasks in this section to get set up with AWS for the first time. If you already have an AWS account, skip ahead to Getting started (p. 6).

When you sign up for AWS, your AWS account automatically has access to all services in AWS, including Amazon Managed Service for Prometheus. However, you are charged only for the services that you use.

Get an AWS account and your root user credentials

To access AWS, you must sign up for an AWS account.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Creating an IAM user

If your account already includes an IAM user with full AWS administrative permissions, you can skip this section.

When you first create an Amazon Web Services (AWS) account, you begin with a single sign-in identity. That identity has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user*. When you sign in, enter the email address and password that you used to create the account.

> **Important**
> We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks. To view the tasks that require you to sign in as the root user, see Tasks that require root user credentials.

To create an administrator user for yourself and add the user to an administrators group (console)

1. Sign in to the IAM console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   > **Note**
   > We strongly recommend that you adhere to the best practice of using the **`Administrator`** IAM user that follows and securely lock away the root user credentials. Sign in as the root user only to perform a few account and service management tasks.

2. In the navigation pane, choose **Users** and then choose **Add user**.
3. For **User name**, enter **`Administrator`**.
4. Select the check box next to **AWS Management Console access**. Then select **Custom password**, and then enter your new password in the text box.
5. (Optional) By default, AWS requires the new user to create a new password when first signing in. You can clear the check box next to **User must create a new password at next sign-in** to allow the new user to reset their password after they sign in.
6. Choose **Next: Permissions**.
7. Under **Set permissions**, choose **Add user to group**.
8. Choose **Create group**.
9. In the **Create group** dialog box, for **Group name** enter **`Administrators`**.
10. Choose **Filter policies**, and then select **AWS managed - job function** to filter the table contents.
11. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.

    Note
    You must activate IAM user and role access to Billing before you can use the `AdministratorAccess` permissions to access the AWS Billing and Cost Management console. To do this, follow the instructions in step 1 of the tutorial about delegating access to the billing console.
12. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
13. Choose **Next: Tags**.
14. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see Tagging IAM entities in the *IAM User Guide*.
15. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

You can use this same process to create more groups and users and to give your users access to your AWS account resources. To learn about using policies that restrict user permissions to specific AWS resources, see Access management and Example policies.

Signing in as an IAM user

Sign in to the IAM console by choosing **IAM user** and entering your AWS account ID or account alias. On the next page, enter your IAM user name and your password.

Note
For your convenience, the AWS sign-in page uses a browser cookie to remember your IAM user name and account information. If you previously signed in as a different user, choose the sign-in link beneath the button to return to the main sign-in page. From there, you can enter your AWS account ID or account alias to be redirected to the IAM user sign-in page for your account.

Creating IAM user access keys

Access keys consist of an access key ID and secret access key, which are used to sign programmatic requests that you make to AWS. If you don't have access keys, you can create them from the AWS Management Console. As a best practice, do not use the AWS account root user access keys for any task where it's not required. Instead, create a new administrator IAM user with access keys for yourself.

The only time that you can view or download the secret access key is when you create the keys. You cannot recover them later. However, you can create new access keys at any time. You must also have

permissions to perform the required IAM actions. For more information, see Permissions required to access IAM resources in the *IAM User Guide*.

To create access keys for an IAM user

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users**.
3. Choose the name of the user whose access keys you want to create, and then choose the **Security credentials** tab.
4. In the **Access keys** section, choose **Create access key**.
5. To view the new access key pair, choose **Show**. You will not have access to the secret access key again after this dialog box closes. Your credentials will look something like this:

   - Access key ID: AKIAIOSFODNN7EXAMPLE
   - Secret access key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. To download the key pair, choose **Download .csv file**. Store the keys in a secure location. You will not have access to the secret access key again after this dialog box closes.

   Keep the keys confidential in order to protect your AWS account and never email them. Do not share them outside your organization, even if an inquiry appears to come from AWS or Amazon.com. No one who legitimately represents Amazon will ever ask you for your secret key.
7. After you download the `.csv` file, choose **Close**. When you create an access key, the key pair is active by default, and you can use the pair right away.

Related topics

- What is IAM? in the *IAM User Guide*
- AWS security credentials in *AWS General Reference*

Getting started

This section explains how to create Amazon Managed Service for Prometheus workspaces, set up the ingestion of Prometheus metrics to those workspaces, and query those metrics.

Topics

Create a 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.

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 if you want. 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.

   ```
   aws amp create-workspace [ --alias my-first-workspace]
   ```

   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`.
2. If your `create-workspace` command returns a status of `CREATING`, you can then enter the following command to find 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 console

1. Open the Amazon Managed Service for Prometheus console at https://console.aws.amazon.com/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

This section explains how to set up the ingestion of metrics into your workspace.

Ingesting metrics is done using Prometheus remote write. Remote write enables the sending of samples to a remote storage destination. For more details about the remote write configurations, see remote_write in the Prometheus documentation.

Metrics ingested into a workspace are stored for 150 days, and are then automatically deleted.

Topics

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 a AWS PrivateLink VPC endpoint for Amazon Managed Service for Prometheus, see Using Amazon Managed Service for Prometheus with interface VPC endpoints (p. 75).

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'll 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.

Ingestion methods and how to set them up

There are two options for ingesting metrics into your Amazon Managed Service for Prometheus workspace. Both methods use remote write.

- Ingest metrics from a Prometheus server.
- Use AWS Distro for OpenTelemetry.

> **Important**
> 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 Send high-availability data with Prometheus or the Prometheus Operator (p. 23).

Topics

Quick Start: 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 where the new Prometheus server will collect metrics from.
- 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 (p. 70) 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`.

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 (p. 70).
   - Replace *WORKSPACE_ID* with the ID of your Amazon Managed Service for Prometheus workspace.
   - Replace *AWS_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: ${IAM_PROXY_PROMETHEUS_ROLE_ARN}
server:
  remoteWrite:
    - url: https://aps-workspaces.${AWS_REGION}.amazonaws.com/workspaces/
${WORKSPACE_ID}/api/v1/remote_write
      sigv4:
        region: ${AWS_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
```

Set up ingestion from an existing Prometheus server in Kubernetes

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

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 (p. 70) to set up the roles.

Step 2: Upgrade your existing Prometheus server using Helm

The instructions in this section includes 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 `${IAM_PROXY_PROMETHEUS_ROLE_ARN}` with the ARN of the **amp-iamproxy-ingest-role** that you created in Step 1: Set up IAM roles for service accounts (p. 10). The role ARN should have the format of `arn:aws:iam::`*`your account ID`*`:role/amp-iamproxy-ingest-role`.
   - Replace `${WORKSPACE_ID}` with your Amazon Managed Service for Prometheus workspace ID.
   - Replace `${AWS_REGION}` with the Region of the Amazon Managed Service for Prometheus workspace (i.e. `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: ${IAM_PROXY_PROMETHEUS_ROLE_ARN}
server:
  remoteWrite:
    - url: https://aps-workspaces.${AWS_REGION}.amazonaws.com/workspaces/
${WORKSPACE_ID}/api/v1/remote_write
      sigv4:
        region: ${AWS_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 older 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 older 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
          - ${AWS_REGION}
          - --host
          - aps-workspaces.${AWS_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 `${AWS_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 (p. 10). 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 metrics ingestion using AWS Distro for Open Telemetry on an Amazon Elastic Kubernetes Service cluster

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.

Collecting Prometheus metrics with ADOT involves two OpenTelemetry components: the Prometheus Receiver and the AWS Prometheus Remote Write Exporter.

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 AWS 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. 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 use a sample app that is used for integration tests of the process. The sample app exposes metrics at the `/metrics` endpoint, just as the Prometheus client library does.

Prerequisites

Before you begin the ingestion setup steps below, you must set up the your IAM role for 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 (p. 70).

   The ADOT Collector will use this role when it scrapes and exports metrics.
2. Next, edit the trust policy. Open the IAM console at at https://console.aws.amazon.com/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:

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

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

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

Enabling Prometheus metric collection

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
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 eks-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 configurations 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
```

AWS Prometheus remote write exporter

The configuration for the AWS Prometheus Remote Write Exporter is 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.

```
exporters:
  awsprometheusremotewrite:
    endpoint: "https://aws-managed-prometheus-endpoint/v1/api/remote_write"
    aws_auth:
      service: "aps"
      region: "user-region"
```

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 AWS Prometheus Remote Write Exporter depends on the AWS SDK for Go AWS Go SDK 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*.

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*.

```
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:
  awsprometheusremotewrite:
    endpoint: my-remote-URL
  aws_auth:
    region: my-region
      service: aps
  logging:
    loglevel: info
extensions:
  health_check:
  pprof:
    endpoint: :1888
  zpages:
    endpoint: :55679
service:
  extensions: [pprof, zpages, health_check]
  pipelines:
    metrics:
      receivers: [prometheus]
      exporters: [logging, awsprometheusremotewrite]
    metrics/ecs:
```

```
      receivers: [awsecscontainermetrics]
      processors: [filter]
      exporters: [logging, awsprometheusremotewrite]
```

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
```

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
```

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`.

Example: Task definition

The following is an example of how your task defintion may 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": "aws_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"
}
```

Attach the AWS managed policy `AmazonPrometheusRemoteWriteAccess` to your IAM role

When you created your Amazon ECS environment, Amazon ECS provided you with the IAM role `ecsTaskExecutionRole`. This role grants Amazon ECS and Fargate permission to call the AWS API for you. For more information about this role, see Amazon ECS task execution IAM role.

1. Open the IAM console at https://console.aws.amazon.com/iam/
2. In the navigation pane, choose **Roles**.
3. Search the list of roles, and select **`ecsTaskExecutionRole`**.

   If the role does not exist, see Creating the task execution IAM role.
4. In the search bar, enter **`AmazonPrometheusRemoteWriteAccess`**.
5. From the dropdown, select **`AmazonPrometheusRemoteWriteAccess`** and choose **Attach policy**.

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

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. 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://console.aws.amazon.com/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

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 in the *Amazon Managed Service for Prometheus User Guide*.

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.

Send high-availability data with Prometheus or the Prometheus Operator

With Amazon Managed Service for Prometheus, you can use multiple Prometheus instances as servers in high-availability mode. This section shows you how to set up Prometheus servers as collectors with a high-availability configuration, so Amazon Managed Service for Prometheus deduplicates your metrics and doesn't charge you twice.

When you set up deduplication, Amazon Managed Service for Prometheus makes one Prometheus instance a leader replica and drops data samples only from that replica. If the leader replica stops sending data samples to Amazon Managed Service for Prometheus for 30 seconds, Amazon Managed Service for Prometheus automatically makes another Prometheus instance a leader replica and drops data from the new leader.

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.

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.

Send high-availability data to Amazon Managed Service for Prometheus with the Prometheus Operator

To set up a high-availability configuration with the Prometheus Operator, 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
```

FAQ: High Availability Data Setup

1. **Should I include the value "__replica__" into another label to track the sample points?**

   No

Understanding your Prometheus server output

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 (p. 92).

Type of data	Query to use	Default Amazon Managed Service for Prometheus quota
Current active series	`prometheus_tsdb_head_series`	1,000,000

Type of data	Query to use	Default Amazon Managed Service for Prometheus quota
Current ingestion rate	`rate(prometheus_tsdb_head_samples_appende`	70,000 samples/second
Most-to-least list of active series per metric name	`sort_desc(count by(__name__) ({__name__!=""}))`	200,000
Number of labels per metric series	`group by(mylabelname) ({__name__!=""})`	70

Query your Prometheus metrics

Now that metrics are being ingested to the workspace, you can query them. You can use a service such as Grafana to query the metrics, or you can use Amazon Managed Service for Prometheus APIs.

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.

Topics

Securing 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. 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 a AWS PrivateLink VPC endpoint for Amazon Managed Service for Prometheus, see Using Amazon Managed Service for Prometheus with interface VPC endpoints (p. 75).

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 Grafana open source or Grafana Enterprise for use 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.

For instructions for setting up a standalone Grafana using the tar.gz or zip file, see Install Grafana in the Grafana documentation. If you install a new standalone Grafana, you will be prompted for username and password. The default is **`admin/admin`**. You will be prompted to change the password after you log in for the first time. For more information, see Getting started with Grafana in the Grafana documentation.

To check your Grafana version, enter the following command.

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

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 (p. 65).

Set up AWS SigV4

Amazon Managed Service for Prometheus is integrated with AWS Identity and Access Management (IAM) to ensure that all calls to Prometheus APIs, such as query and ingest, are secured 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 Service for Grafana, SIGv4 authentication is fully automated. For more information about Amazon Managed Service for Grafana, see What is Amazon Managed Service for 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
```

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**

- 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**

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
```

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 (p. 65).
- 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 (p. 65).

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 (p. 29). 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 (p. 30).

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 (p. 72) 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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::account-id:oidc-provider/
oidc.eks.aws_region.amazonaws.com/id/openid"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.aws_region.amazonaws.com/id/openid:sub": [
            "system:serviceaccount:aws-amp:amp-iamproxy-query-service-account",
            "system:serviceaccount:grafana-namespace:grafana-service-account-name"
          ]
        }
      }
    }
  ]
}
```

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

Amazon Managed Service for Prometheus 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 (p. 110).

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 as a sidecar or as a pod. For more information, see aws-sigv4-proxy.

Managing workspaces

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

Topics

Create a workspace

Follow these steps to create a Amazon Managed Service for Prometheus workspace.

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 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.

   ```
   aws amp create-workspace [ --alias my-first-workspace] [--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`.
   - `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://console.aws.amazon.com/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 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.

5. 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 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 (p. 7).

Edit a workspace

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://console.aws.amazon.com/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 workspace ARN

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

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

1. Open the Amazon Managed Service for Prometheus console at https://console.aws.amazon.com/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.

   The workspace ARN is displayed under **ARN**.

To use the AWS CLI to find your workspace ARN, enter the following command.

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

Find the value of `arn` in the results.

Delete a workspace

When you delete a workspace, the data that has been ingested into it is not immediately deleted. It will be permanently deleted within one month.

To delete a workspace using the AWS CLI, enter 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://console.aws.amazon.com/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**.

Recording rules and alerting rules

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 forwards the 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

Necessary IAM permissions

To use rules in Amazon Managed Service for Prometheus, you must be signed in to an account that the required permissions. You can create an AWS Identity and Access Management (IAM) policy with the following permissions and attach it to your IAM user or IAM role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aps: CreateRuleGroupsNamespace",
```

```
                "aps: ListRuleGroupsNamespaces",
                "aps: DescribeRuleGroupsNamespace",
                "aps: PutRuleGroupsNamespace",
                "aps: DeleteRuleGroupsNamespace",
            ],
            "Resource": "*"
        }
    ]
}
```

Creating 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 has the same format as a rules file in standalone Prometheus. For more information, see Defining Recording rules and Alerting rules.

The following is a basic example of a rules file:

```
groups:
  - name: test
    rules:
    - record: metric:recording_rule
      expr: avg(rate(container_cpu_usage_seconds_total[5m]))
  - name: alert-test
    rules:
    - alert: metric:alerting_rule
      expr: avg(rate(container_cpu_usage_seconds_total[5m])) > 0
      for: 2m
```

Uploading a rules configuration file to Amazon Managed Service for Prometheus

Now you must upload this rules configuration file to Amazon Managed Service for Prometheus. You can use either the console or the AWS CLI to upload it.

To use the Amazon Managed Service for Prometheus console to upload your rules configuration and create the namespace

1. Open the Amazon Managed Service for Prometheus console at https://console.aws.amazon.com/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.
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.

Editing a rules configuration file

You can't edit a rules configuration file directly in the console. Instead, you upload a new rules file to replace it. 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://console.aws.amazon.com/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. (Optional) If you want to start by editing the current rules configuration file, choose **Download** or **Copy**.
5. When your new rules file is ready, choose **Replace**.
6. Choose **Choose file**, select the new rules definition file, and choose **Continue**.

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.

Alert manager and templating

When the alerting rules that Amazon Managed Service for Prometheus runs are firing, alert manager handles the alerts that are sent. It deduplicates, 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 (p. 115).

Prometheus templating

Standalone Prometheus supports templating, using separating 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

Necessary IAM permissions

To use rules in Amazon Managed Service for Prometheus, you must be signed in to an account that has the required permissions. You can create an AWS Identity and Access Management (IAM) policy with the following permissions and attach it to your IAM user or IAM role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aps: CreateAlertManagerDefinition",
                "aps: DescribeAlertManagerSilence",
                "aps: DescribeAlertManagerDefinition",
                "aps: PutAlertManagerDefinition",
                "aps: DeleteAlertManagerDefinition",
                "aps: ListAlerts",
                "aps: ListRules",
                "aps: ListAlertManagerReceivers",
                "aps: ListAlertManagerSilences",
                "aps: ListAlertManagerAlerts",
                "aps: ListAlertManagerAlertGroups",
                "aps: GetAlertManagerStatus",
                "aps: GetAlertManagerSilence",
                "aps: ListAlertManagerSilences",
                "aps: PutAlertManagerSilences",
                "aps: DeleteAlertManagerSilence",
                "aps: CreateAlertManagerAlerts",
            ],
            "Resource": "*"
        }
    ]
}
```

Creating an alert manager configuration file

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. 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.

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 "__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 }}
```

Setting up your alert receiver

The only alert receiver currently supported in Amazon Managed Service for Prometheus is Amazon Simple Notification Service (Amazon SNS). For more information, see What is Amazon SNS?.

Topics

- (Optional) Sending from Amazon SNS to other destinations (p. 45)

(Optional) Creating a new Amazon SNS topic

You can use an existing Amazon SNS topic or 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.

Giving Amazon Managed Service for Prometheus permission to send 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 includes a `Condition` statement to help prevent the *confused deputy* security 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 (p. 43).

To give Amazon Managed Service for Prometheus permission to send messages to your Amazon SNS topic

1. Open the Amazon SNS console at https://console.aws.amazon.com/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"
}
```

6. Choose **Save changes**.

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 messages to your Amazon SNS topic (p. 42) 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.

Specifying your Amazon SNS topic in the alert manager configuration file

Now, you can add your Amazon SNS receiver to your alert manager configuration. To do 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_config:
    sigv4:
        region: region
    topic_arn: ARN_of_SNS_topic
    attributes:
       key: somekey
       value: somevalue
```

Now you must upload your alert manager configuration file to Amazon Managed Service for Prometheus. For more information, see Uploading your alert manager configuration file to Amazon Managed Service for Prometheus (p. 45).

(Optional) Configuring alert manager to output JSON to Amazon SNS

You can configure alert manager to send alerts in JSON format, so that they can be processed downstream from Amazon SNS in AWS Lambda or in webhook-receiving endpoints. The default template provided with the Amazon Managed Service for Prometheus alert manager outputs the message payload in a text list format, which might not be easy to parse. 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 }}
```

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'
```

(Optional) Sending from Amazon SNS to other destinations

Currently, Amazon Managed Service for Prometheus can send alert messages directly to Amazon SNS only. You can configure Amazon SNS to send those messages on to other destinations such as email, webhook, Slack, and 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.

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 do I use webhooks to publish Amazon SNS messages to Amazon Chime, Slack, or Microsoft Teams?.

OpsGenie

For information about how to configure an Amazon SNS topic to output messages to OpsGenie, see Integrate Opsgenie with Incoming Amazon SNS.

Uploading your alert manager configuration file to Amazon Managed Service for Prometheus

Now, you must upload your alert manager configuration file to Amazon Managed Service for Prometheus. You can use either the console or the AWS CLI to upload it.

To use the Amazon Managed Service for Prometheus console to upload your alert manager configuration

1. Open the Amazon Managed Service for Prometheus console at https://console.aws.amazon.com/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**. If the workspace has an alert manager definition that you want to replace, choose **Replace definition**.
5. Choose **Choose file**, select the alert manager definition file, and choose **Continue**.

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 fileb://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.

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 shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon Managed Service for Prometheus, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon 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

Amazon Managed Service for Prometheus conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this

infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN Partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

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

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

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form 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 Amazon Managed Service for Prometheus or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

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

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.

Customer data can be stored on Amazon Elastic Block Store for short-term storage and forwarded to Amazon Simple Storage Service for long-term storage.

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 doesn't support the creation of customer managed keys. Amazon Managed Service for Prometheus is not intended to store highly sensitive data. Server-side data is encrypted on your behalf using AWS managed keys. For more information about these keys, see AWS managed keys in the *AWS Key Management Service Developer Guide*.

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

Encryption 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.

By default, Amazon Managed Service for Prometheus automatically provides you with encryption at rest and does this using AWS owned keys. These keys are a collection of AWS KMS keys that an AWS service owns and manages for use in multiple AWS accounts. You can't view, manage, or audit the use of these keys. You aren't required to take action or change programs to protect the keys that encrypt your data.

For more information, see AWS owned keys in 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

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Amazon Managed Service for Prometheus.

Service user – If you use the Amazon Managed Service for Prometheus service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon Managed Service for Prometheus features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon Managed Service for Prometheus, see Troubleshooting Amazon Managed Service for Prometheus identity and access (p. 63).

Service administrator – If you're in charge of Amazon Managed Service for Prometheus resources at your company, you probably have full access to Amazon Managed Service for Prometheus. It's your job to determine which Amazon Managed Service for Prometheus features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon Managed Service for Prometheus, see How Amazon Managed Service for Prometheus works with IAM (p. 54).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon Managed Service for Prometheus. To view example Amazon Managed Service for Prometheus identity-based policies that you can use in IAM, see Identity-based policy examples for Amazon Managed Service for Prometheus (p. 59).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see Signing in to the AWS Management Console as an IAM user or root user in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email address or your IAM user name. You can access AWS programmatically using your root user or IAM users access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 signing process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing access keys for IAM users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated users and roles in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, resources, and condition keys for Amazon Managed Service for Prometheus in the *Service Authorization Reference*.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – 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 IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions.

You can sign in as the root user or an IAM user, or you can assume an IAM role. When you then make a request, AWS evaluates the related identity-based or resource-based policies. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

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.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how 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 (p. 55)	Yes
Resource-based policies (p. 55)	No
Policy actions (p. 56)	Yes
Policy resources (p. 56)	Yes
Policy condition keys (p. 57)	No
ACLs (p. 57)	No
ABAC (tags in policies) (p. 57)	No
Temporary credentials (p. 58)	Yes
Principal permissions	No

IAM feature	Amazon Managed Service for Prometheus support
Service roles (p. 58)	No
Service-linked roles (p. 58)	No

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 Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for 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 (p. 59).

Resource-based policies within Amazon Managed Service for Prometheus

Supports resource-based policies	No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or 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. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Policy actions for 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. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of 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 (p. 59).

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. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of 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 (p. 59).

Policy condition keys for Amazon Managed Service for Prometheus

Supports 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 (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. 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 (p. 59).

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)	No

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using Temporary credentials with Amazon Managed Service for Prometheus

Supports temporary credentials	Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

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 Creating 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	No

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 IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for Amazon Managed Service for Prometheus

By default, IAM users and roles don't have permission to create or modify Amazon Managed Service for Prometheus resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform actions on the resources that they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating policies on the JSON tab in the *IAM User Guide*.

Topics

Policy best practices

Identity-based policies are very powerful. They 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 using AWS managed policies** – To start using Amazon Managed Service for Prometheus quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get started using permissions with AWS managed policies in the *IAM User Guide*.
- **Grant least privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant least privilege in the *IAM User Guide*.
- **Enable MFA for sensitive operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.
- **Use policy conditions for extra security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON policy elements: Condition 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 (IAM 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 you'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*.

AWS-managed IAM policies for Amazon Managed Service for Prometheus

Amazon Managed Service for Prometheus includes the following AWS-managed policies.

AmazonPrometheusFullAccess

The contents of **AmazonPrometheusFullAccess** are as follows:

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

AmazonPrometheusConsoleFullAccess

The contents of **AmazonPrometheusConsoleFullAccess** are as follows. This policy includes `tag:GetTagKeys` and `tag:GetTagValues` so that users with this policy can see tag suggestions when they use the Amazon Managed Service for Prometheus console to add tags to Amazon Managed Service for Prometheus resources.

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

```
            "Effect": "Allow",
            "Action": [
                "aps:CreateWorkspace",
                "aps:DescribeWorkspace",
                "aps:UpdateWorkspaceAlias",
                "aps:DeleteWorkspace",
                "aps:ListWorkspaces",
                "aps:DescribeAlertManagerDefinition",
                "aps:DescribeRuleGroupsNamespace",
                "aps:CreateAlertManagerDefinition",
                "aps:CreateRuleGroupsNamespace",
                "aps:DeleteAlertManagerDefinition",
                "aps:DeleteRuleGroupsNamespace",
                "aps:ListRuleGroupsNamespaces",
                "aps:PutAlertManagerDefinition",
                "aps:PutRuleGroupsNamespace",
                "aps:TagResource",
                "aps:UntagResource"
            ],
            "Resource": "*"
        }
    ]
}
```

AmazonPrometheusRemoteWriteAccess

The contents of **AmazonPrometheusRemoteWriteAccess** are as follows:

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

AmazonPrometheusQueryAccess

The contents of **AmazonPrometheusQueryAccess** are as follows:

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

Amazon Managed Service for Prometheus updates to AWS managed policies

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

Change	Description	Date
AmazonPrometheusConsoleFullAccess (p. 60) – Update to an existing policy	Amazon Managed Service for Prometheus added new permissions to **AmazonPrometheusConsoleFullAccess** to support new Amazon Managed Service for Prometheus features and so that users with this policy can see a list of tag suggestions when they apply tags to Amazon Managed Service for Prometheus resources. The `tag:GetTagKeys`, `tag:GetTagValues`, `aps:CreateAlertManagerDefinition`, `aps:CreateRuleGroupsNamespace`, `aps:DeleteAlertManagerDefinition`, `aps:DeleteRuleGroupsNamespace`, `aps:DescribeAlertManagerDefinition`, `aps:DescribeRuleGroupsNamespace`, `aps:ListRuleGroupsNamespaces`, `aps:PutAlertManagerDefinition`, `aps:PutRuleGroupsNamespace`, `aps:TagResource`, and `aps:UntagResource` permissions were added.	September 29, 2021
Amazon Managed Service for Prometheus started tracking changes	Amazon Managed Service for Prometheus started tracking changes for its AWS managed policies.	September 15, 2021

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: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

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

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 does not have the fictional `aps:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 aps:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *my-example-widget* resource using the `aps:GetWidget` action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies 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 granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

> **Important**
> Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing access keys in the *IAM User Guide*.

I'm an administrator and want to allow others to access Amazon Managed Service for Prometheus

To allow others to access Amazon Managed Service for Prometheus, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Amazon Managed Service for Prometheus.

To get started right away, see Creating your first IAM delegated user and group in the *IAM User Guide*.

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 (p. 54).
- 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 How IAM roles differ from resource-based policies 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

The following table displays possible Amazon Managed Service for Prometheus actions and their required permissions:

Action	Required permission
Create alerts.	`aps:CreateAlertManagerAlerts`
Create an alert manager definition in a workspace. For more information, see Alert manager and templating (p. 39).	`aps:CreateAlertManagerDefinition`
Create a rule groups namespace in a workspace. For more information, see Recording rules and alerting rules (p. 35).	`aps:CreateRuleGroupsNamespace`
Create an Amazon Managed Service for Prometheus workspace.A *workspace* is a logical space dedicated to the storage and querying of Prometheus metrics.	`aps:CreateWorkspace`

Action	Required permission
Delete an alert manager definition from a workspace.	`aps:DeleteAlertManagerDefinition`
Delete alert silences.	`aps:DeleteAlertManagerSilence`
Delete an Amazon Managed Service for Prometheus workspace.	`aps:DeleteWorkspace`
Retrieve detailed information about alert manager definitions.	`aps:DescribeAlertManagerDefinition`
Retrieve detailed information about rule groups namespaces.	`aps:DescribeRuleGroupsNamespace`
Retrieve detailed information about an Amazon Managed Service for Prometheus workspace.	`aps:DescribeWorkspace`
Retrieve detailed information about an alert silence.	`aps:GetAlertManagerSilence`
Retrieve the status of the alert manager in a workspace.	`aps:GetAlertManagerStatus`
Retrieve labels.	`aps:GetLabels`
Retrieve metadata for Amazon Managed Service for Prometheus metrics.	`aps:GetMetricMetadata`
Retrieve time series data.	`aps:GetSeries`
Retrieve a list of the alert groups that are defined in the alert manager definition.	`aps:ListAlertManagerAlertGroups`
Retrieve a list of the alerts that are defined in alert manager.	`aps:ListAlertManagerAlerts`
Retrieve a list of the receivers that are defined in the alert manager definition.	`aps:ListAlertManagerReceivers`
Retrieve a list of the alert silences that are defined.	`aps:ListAlertManagerSilences`
Retrieve a list of the active alerts.	`aps:ListAlerts`
Retrieve a list of the rules in the rule groups namespaces in your workspaces.	`aps:ListRules`
Retrieve a list of the rule groups namespaces in your workspaces.	`aps:ListRuleGroupsNamespaces`
Retrieve the tags that are associated with your Amazon Managed Service for Prometheus resources.	`aps:ListTagsForResource`
Retrieve a list of the Amazon Managed Service for Prometheus workspaces that exist in the account.	`aps:ListWorkspaces`

Action	Required permission
Update an existing alert manager definition in a workspace.	`aps:PutAlertManagerDefinition`
Create alert silences.	`aps:PutAlertManagerSilences`
Update an existing rule groups namespace.	`aps:PutRuleGroupsNamespace`
Run a query on Amazon Managed Service for Prometheus metrics.	`aps:QueryMetrics`
Perform a remote write operation to initiate the streaming of metrics from a Prometheus server to Amazon Managed Service for Prometheus.	`aps:RemoteWrite`
Assign tags to Amazon Managed Service for Prometheus resources.	`aps:TagResource`
Remove tags from Amazon Managed Service for Prometheus resources.	`aps:UntagResource`
Modify the aliases of existing workspaces.	`aps:UpdateWorkspaceAlias`

Built-in AWS-managed policies

AWS provides several built-in, AWS-managed policies for Amazon Managed Service for Prometheus.

AmazonPrometheusFullAccess

This policy provides full access to all Amazon Managed Service for Prometheus actions and resources.

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

AmazonPrometheusConsoleFullAccess

This policy provides access to all Amazon Managed Service for Prometheus console actions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "aps:CreateWorkspace",
                "aps:DescribeWorkspace",
                "aps:UpdateWorkspaceAlias",
                "aps:DeleteWorkspace",
                "aps:ListWorkspaces"
```

```
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

AmazonPrometheusQueryAccess

This policy provides access to query the metrics stored in all Amazon Managed Service for Prometheus workspaces in the account.

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

AmazonPrometheusRemoteWriteAccess

This policy provides permissiont to remote write metrics into all Amazon Managed Service for Prometheus workspaces in the account.

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

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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aps:*",
                "eks:DescribeCluster",
```

```
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

Compliance Validation for Amazon Managed Service for Prometheus

Third-party auditors assess the security and compliance of AWS services as part of multiple AWS compliance programs, such as SOC, PCI, FedRAMP, and HIPAA.

To learn whether Amazon Managed Service for Prometheus or other AWS services are in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

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

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

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.

  Note
  Not all services are compliant with HIPAA.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

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.

Infrastructure Security in Amazon Managed Service for Prometheus

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

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

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

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.

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 a private connection 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 data in the workspace.
- `com.amazonaws.region.aps`

  Choose this service name to perform workspace management tasks.

For more information about how to create an interface VPC enpoint, see Creating an interface endpoint in the *Amazon VPC User Guide*.

You don't need to change your settings to call another AWS service. 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.

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, a role, or an AWS service in Amazon Managed Service for Prometheus.

CloudTrail captures all API calls for Amazon Managed Service for Prometheus as events. The calls that are captured include calls from the Amazon Managed Service for Prometheus console and code calls to the Amazon Managed Service for Prometheus API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon Managed Service for Prometheus. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. 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, who made the request, when it was made, and additional details.

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

Amazon Managed Service for Prometheus information in CloudTrail

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

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

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

Amazon Managed Service for Prometheus supports logging the following actions:

- CreateAlertManagerDefinition (p. 96)
- CreateRuleGroupsNamespace (p. 96)
- CreateWorkspace (p. 96)
- DeleteAlertManagerDefinition (p. 98)
- DeleteWorkspace (p. 98)
- DeleteRuleGroupsNamespace (p. 98)
- DescribeAlertManagerDefinition (p. 99)
- DescribeRuleGroupsNamespace (p. 99)
- DescribeWorkspace (p. 100)
- ListRuleGroupsNamespaces (p. 101)
- ListWorkspaces (p. 102)
- PutAlertManagerDefinition (p. 103)
- PutRuleGroupsNamespace (p. 104)
- UpdateWorkspaceAlias (p. 106)

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

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

For more information, see the CloudTrail userIdentity element.

Understanding Amazon Managed Service for Prometheus log file entries

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

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":
 "YWxlcnRtYW5hZ2VyX2NvbmZpZzogfAogIGdsb2JhbDoKICAgIHNtdHBfc21hcnRob3N0OiAnbG9jYWxob3N0OjI1JwogICAgc210c
        "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":
 "Z3JvdXBzOgogIC0gbmFtZTogdGVzdFJ1bGVHcm91cHNOYW1lc3BhY2UKICAgIHJ1bGVzOgogICAgLSBhbGVydDogdGVzdEFsZXJ0U
        "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"
}
```

Troubleshooting

Use the following sections to help troubleshoot issues with Amazon Managed Service for Prometheus.

429 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
```

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 (p. 92)

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 Send high-availability data with Prometheus or the Prometheus Operator (p. 23).

I see an error message related to a limit

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 (p. 92).

- per-user series limit of 1000000 exceeded, please contact administrator to raise it
- per-metric series limit of 200000 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.

Tagging

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

- Tagging rule groups namespaces (p. 87)

Tagging workspaces

Use the procedures in this section to work with tags for Amazon Managed Service for Prometheus workspaces.

Topics

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 workspace (p. 32).

Topics

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://console.aws.amazon.com/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 workspace (p. 32).

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:workspace/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://console.aws.amazon.com/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://console.aws.amazon.com/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://console.aws.amazon.com/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

Use the procedures in this section to work with tags for Amazon Managed Service for Prometheus rule groups namespaces.

Topics

- Remove a tag from a rule groups namespace (p. 90)

Add a tag to an 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 Creating a rules file (p. 36).

Topics

- Add a tag to a rule groups namespace (console) (p. 88)
- Add a tag to a rule groups namespace (AWS CLI) (p. 88)

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://console.aws.amazon.com/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 ??? (p. 32).

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://console.aws.amazon.com/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://console.aws.amazon.com/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://console.aws.amazon.com/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

Series and labels quotas

Amazon Managed Service for Prometheus has the following quotas for series, labels, and API requests. The **Possible error message** column shows what error message you might see if your Prometheus data exceeds a limit. If you see one of these error messages, you should request an increase to the corresponding limit.

Resource	Default quota	Possible error message
Active series (metrics that have reported data in the past 5 minutes) per workspace	1,000,000. You can request a quota increase.	per-user series limit of 1000000 exceeded, please contact administrator to raise it
Active series per metric name	200,000. You can request a quota increase.	per-metric series limit of 200000 exceeded, please contact administrator to raise it
Ingestion rate	70,000 samples per second. You can request a quota increase.	ingestion rate limit (...) exceeded
Ingestion burst size	1,000,000 samples. You can request a quota increase.	ingestion rate limit (...) exceeded
Labels per metric series.	70. You can request a quota increase.	series has too many labels (...) series: '%s'
Query bytes for instant queries	750MB that can be scanned by a single instant query. This quota can't be changed.	the query hit the aggregated chunks size limit (A *chunk* stores raw samples of series for a certain time span.)

Resource	Default quota	Possible error message
Query bytes for range queries	750MB that can be scanned per 24-hour interval in a single range query. This quota can't be changed.	the query hit the aggregated chunks size limit (A *chunk* stores raw samples of series for a certain time span.)
Query time range	32 days between the start time and the end time of a PromQL query. This quota cannot be changed.	the query time range exceeds the limit (query length: xxx, limit: yyy)
Query samples	12,000,000 samples that can be scanned during a single query. This quota can't be changed.	query processing would load too many samples into memory in query execution
Retention time for ingested data	150 days. Data older than this is deleted from the workspace.	
Workspaces per Region per account	10. You can request a quota increase.	Limit exceeded. Maximum workspaces per account.
Workspace management API operations, including CreateWorkspace, DeleteWorkspace, DescribeWorkspaces, ListWorkspaces, and UpdateWorkspaceAlias	10 transactions per second (TPS), with a burst limit of 10 TPS. You can request a quota increase.	

Rules and alert manager quotas

Resource	Default quota
Active alerts	1000. You can request a quota increase.
Alert aggregation group size	1K. You can request a quota increase.
Alert manager API operations, including CreateAlertManagerDefinition, DeleteAlertManagerDefinition,	100 transactions per second (TPS).

Resource	Default quota
DescribeAlertManagerDefinition, and PutAlertManagerDefinition	
Alert manager definition file size	1 MB. This quota can't be changed.
Alerts per workspace, in size	20 MB. You can request a quota increase.
Inhibition rules	100. You can request a quota increase.
Nodes in the alert manager routing tree	100. You can request a quota increase.
Rule evaluation interval	Rules are evaluated every 60 seconds. This can't be changed.
Rule groups in a workspace	100. You can request a quota increase.
Rule groups namespaces in a workspace	10. You can request a quota increase.
Rules in a rule group	20. You can request a quota increase.
Rules definition file size	1 MB. This quota can't be changed.
Templates in the alert manager definition file	100. You can request a quota increase.

Additional quotas on ingested data

Amazon Managed Service for Prometheus also has additional quotas for data ingested into the workspace.

- Metric samples older than 1 hour are refused from being ingested.
- Every sample and metadata must have a metric name.
- Maximum length accepted for label names: 1024 bytes
- Maximum length accepted for label value: 2048 bytes
- Maximum number of metadata per metric: 10
- Maximum size of ingestion or query request: 1 MB
- Maximum length accepted for metric metadata, which includes metric name, HELP, and UNIT: 1024 bytes
- Maximum number of active metrics with metadata per workspace: 20,000
- Maximum retention time for ingested metrics: 150 days. Data older than this is deleted from the workspace.

API reference

This section lists the API operations and data structures supported by Amazon Managed Service for Prometheus.

Amazon Managed Service for Prometheus APIs

The following Amazon Managed Service for Prometheus API operations and data structures are supported.

Topics

CreateAlertManagerDefinition

The `CreateAlertManagerDefinition` operation creates the alert manager definition in a workspace. If a workspace already has an alert manager definition, don't use this operation to update it. Instead, use PutAlertManagerDefinition (p. 103).

Request parameters

- *workspaceId*—The ID of the workspace to add the alert manager definition to.
- *data*— The alert manager definition to add. This is an AlertManagerDefinitionData structure (p. 107).
- *clientToken*—(Optional) A unique, case-sensitive identifier that you can provide to ensure the idempotency of the request.

Response elements

- *status*—A structure that displays the current status of the alert manager definition. For more information, see AlertManagerDefinitionStatus structure (p. 107).

CreateRuleGroupsNamespace

The `CreateRuleGroupsNamespace` operation creates a rule groups namespace within a workspace. A rule groups namespace is associated with exactly one rules file. A workspace can have multiple rule groups namespaces.

Use this operation only to create new rule groups namespaces. To update an existing rule groups namespace, use PutRuleGroupsNamespace (p. 104).

Request parameters

- *workspaceId*—The ID of the workspace to add the rule groups namespace.
- *name*—The name for the new rule groups namespace.
- *data*— The rules file to use in the new namespace. This is a RuleGroupsNamespaceData structure (p. 107).
- *tags*— (Optional) The list of tag keys and values to associate with the rule groups namespace.
- *clientToken*—(Optional) A unique, case-sensitive identifier that you can provide to ensure the idempotency of the request.

Response elements

- *name*—The name of the new rule groups namespace.
- *status*—A structure that displays the current status of the rule groups namespace. For more information, see RuleGroupsNamespaceStatus structure (p. 108).
- *arn*—The Amazon Resource Name (ARN) of the new rule groups namespace.
- *tags*— The list of tag keys and values that are associated with the namespace.

CreateWorkspace

The `CreateWorkspace` operation creates a *workspace*. A workspace is a logical space dedicated to the storage and querying of Prometheus metrics. You can have one or more workspaces in each Region in your account.

Request parameters

- *alias*—(Optional) An alias that you assign to this workspace to help you identify it. It does not need to be unique.

  The alias can be up to 100 characters and can include any type of characters. Amazon Managed Service for Prometheus will automatically strip any blank spaces from the beginning and end of the alias that you specify.
- *tags*— (Optional) The list of tag keys and values to associate with the workspace.
- *clientToken*—(Optional) A unique, case-sensitive identifier that you can provide to ensure the idempotency of the request.

Response elements

- *workspaceId*—The unique ID for the new workspace.
- *arn*—The ARN for the new workspace.
- *tags*— (Optional) The list of tag keys and values that are associated with the workspace.
- *status*—The current status of the new workspace. Immediately after you create the workspace, the status is usually `CREATING`.

Sample request

```
POST /workspaces HTTP/1.1
Content-Length: 79,
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": "myWorkspace",
    "tags": {
       "Status": "Secret",
       "Team": "My-Team"
   },
    "clientToken": "6e5b2726-4bb0-4010-a2e3-de91980cf783"
}
```

Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 185
Access-Control-Allow-Methods: OPTION,GET,POST,DELETE
Access-Control-Expose-Headers: x-amzn-errortype,x-amzn-requestid,x-amzn-trace-id,x-amzn-
errormessage,x-amz-apigw-id,date
Strict-Transport-Security: strict-transport-security: max-age=31536000; includeSubDomains
x-amz-apigw-id: 1234EXAMPLE=
X-Amzn-Trace-Id: Root=1-12345678-39c98b574fd600282a1a0f21
Connection: keep-alive
Cache-Control: no-store,no-cache,must-revalidate
Date: Tue, 01 Dec 2020 19:37:25 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: *,Authorization,Date,X-Amz-Date,content-type,x-amz-content-
sha256,x-amz-user-agent,x-amzn-platform-id,x-amzn-trace-id,X-Amz-Security-Token,X-Amz-
Target
Content-Type: application/x-amz-json-1.1

{
    "arn": "arn:aws:aps:us-west-2:123456789012:workspace/ws-12345678-1234-46a9-933a-
ac50479a5568",
```

```
    "status": {
        "statusCode": "CREATING"
    },
    "tags": {
       "Status": "Secret",
       "Team": "My-Team"
   },
    "workspaceId": "ws-b226cc2a-a446-46a9-933a-ac50479a5568"
}
```

DeleteAlertManagerDefinition

The `DeleteAlertManagerDefinition` operation deletes the alert manager definition from a workspace.

Request parameters

- *workspaceId*— The ID of the workspace to delete the alert manager definition from.
- *clientToken*—(Optional) A unique, case-sensitive identifier that you can provide to ensure the idempotency of the request.

Response elements

- None

DeleteRuleGroupsNamespace

The `DeleteRuleGroupsNamespace` operation deletes one rule groups namespace and its associated rule groups definition.

Request parameters

- *workspaceId*—The ID of the workspace containing the rule groups namespace and definition to delete.
- *name*—The name of the rule groups namespace to delete.
- *clientToken*—(Optional) A unique, case-sensitive identifier that you can provide to ensure the idempotency of the request.

Response elements

- None

DeleteWorkspace

The `DeleteWorkspace` operation deletes an existing workspace.

When you delete a workspace, the data that has been ingested into it is not immediately deleted. It will be permanently deleted within one month.

Request parameters

- *workspaceId*—The ID of the workspace to delete.
- *clientToken*—(Optional) A unique, case-sensitive identifier that you can provide to ensure the idempotency of the request.

Response elements

- None

Sample request

```
DELETE /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568?
clientToken=99ce509d-014c-4911-8f71-9844d9df0a97 HTTP/1.1
Content-Length: 79,
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": "myWorkspace",
    "clientToken": "6e5b2726-4bb0-4010-a2e3-de91980cf783"
}
```

Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 185
Access-Control-Allow-Methods: OPTION,GET,POST,DELETE
Access-Control-Expose-Headers: x-amzn-errortype,x-amzn-requestid,x-amzn-trace-id,x-amzn-
errormessage,x-amz-apigw-id,date
Strict-Transport-Security: strict-transport-security: max-age=31536000; includeSubDomains
x-amz-apigw-id: 1234EXAMPLE=
X-Amzn-Trace-Id: Root=1-12345678-39c98b574fd600282a1a0f21
Connection: keep-alive
Cache-Control: no-store,no-cache,must-revalidate
Date: Tue, 01 Dec 2020 19:37:25 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: *,Authorization,Date,X-Amz-Date,content-type,x-amz-content-
sha256,x-amz-user-agent,x-amzn-platform-id,x-amzn-trace-id,X-Amz-Security-Token,X-Amz-
Target
Content-Type: application/x-amz-json-1.1
```

DescribeAlertManagerDefinition

The `DescribeAlertManagerDefinition` operation retrieves the full information about the alert manager definition for one workspace.

Request parameters

- *workspaceId*—The ID of the workspace to retrieve the alert manager definition from.

Response elements

- *alertManagerDefinition*— The alert manager definition contents. This is an AlertManagerDefinitionDescription structure (p. 107).

DescribeRuleGroupsNamespace

The `DescribeRuleGroupsNamespace` operation returns complete information about one rule groups namespace. To retrieve a list of rule groups namespaces, use ListRuleGroupsNamespaces (p. 101).

Request parameters

- *workspaceId*—The ID of the workspace containing the rule groups namespace.
- *name*—The name of the rule groups namespace that you want information for.

Response elements

- *ruleGroupsNamespace*— The information about the rule groups namespace. This is a RuleGroupsNamespaceDescription structure (p. 107).

DescribeWorkspace

The `DescribeWorkspace` operation displays information about an existing workspace.

Request parameters

- *workspaceId*—The ID of the workspace to describe.

Response elements

- *workspace*—A `WorkspaceDescription` structure that contains details about the workspace. For more information, see WorkspaceDescription structure (p. 108).

Sample request

```
GET /workspaces/ws-b226cc2a-a446-46a9-933a-ac50479a5568 HTTP/1.1
Content-Length: 79,
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": "myWorkspace",
    "clientToken": "6e5b2726-4bb0-4010-a2e3-de91980cf783"
}
```

Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 185
Access-Control-Allow-Methods: OPTION,GET,POST,DELETE
Access-Control-Expose-Headers: x-amzn-errortype,x-amzn-requestid,x-amzn-trace-id,x-amzn-
errormessage,x-amz-apigw-id,date
Strict-Transport-Security: strict-transport-security: max-age=31536000; includeSubDomains
x-amz-apigw-id: 1234EXAMPLE=
X-Amzn-Trace-Id: Root=1-12345678-39c98b574fd600282a1a0f21
Connection: keep-alive
Cache-Control: no-store,no-cache,must-revalidate
Date: Tue, 01 Dec 2020 19:37:25 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: *,Authorization,Date,X-Amz-Date,content-type,x-amz-content-
sha256,x-amz-user-agent,x-amzn-platform-id,x-amzn-trace-id,X-Amz-Security-Token,X-Amz-
Target
Content-Type: application/x-amz-json-1.1
```

```
{
    "workspace": {
        "alias": "myWorkspace",
        "arn": "arn:aws:aps:us-west-2:123456789012:workspace/ws-12345678-a446-46a9-933a-
ac50479a5568",
        "createdAt": 1.606851445922E9,
        "prometheusEndpoint": "https://aps-workspaces.us-west-2.amazonaws.com/workspaces/
ws-b226cc2a-a446-46a9-933a-ac50479a5568/",
        "status": {
            "statusCode": "ACTIVE"
        },
        "tags": {
         "Status": "Secret",
         "Team": "My-Team"
     },
        "workspaceId": "ws-12345678-a446-46a9-933a-ac50479a5568"
    }
}
```

ListRuleGroupsNamespaces

The `ListRuleGroupsNamespaces` operation returns a list of rule groups namespaces in a workspace.

Request parameters

- *workspaceId*—The ID of the workspace containing the rule groups namespaces.
- *name*—(Optional) Use this parameter to filter the rule groups namespaces that are returned. Only the namespaces with names that begin with the value that you specify are returned.
- *maxResults*— (Optional) The maximum number of results to return. Valid values are 1–1000. The default is 100.
- *nextToken*—(Optional) The token for the next set of items to return. (You received this token from a previous call.)

Response elements

- *ruleGroupsNamespaces*— The returned list of rule groups namespaces. Each item in the list is a RuleGroupsNamespaceSummary structure (p. 108).
- *nextToken*—A token indicating that there are more results to retrieve. You can use this token as part of your next `ListRuleGroupsNamespaces` operation to retrieve those results.

ListTagsForResource

The `ListTagsForResource` operation returns the tags that are associated with an Amazon Managed Service for Prometheus resource. Currently, the only resource that can be tagged is a workspace.

Request parameters

- *resourceArn*—The ARN of the workspace.

Response elements

- *tags*—The list of tag keys and values associated with the resource.

Sample request

```
ET /tags/arn%3Aaws%3Aaps%3Aus-east-2%3A621014813884%3Aworkspace%2Fws-effda945-fb72-4e8b-
bbb7-c0ca7e72d5b6 HTTP/1.1
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

{
    "resourceArn": ""arn:aws:aps:us-west-2:123456789012:workspace/ws-12345678-
a446-46a9-933a-ac50479a5568"",
}
```

Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 43
Connection: keep-alive
Date: Tue, 01 Dec 2020 19:37:25 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: *,Authorization,Date,X-Amz-Date,content-type,x-amz-content-
sha256,x-amz-user-agent,x-amzn-platform-id,x-amzn-trace-id,X-Amz-Security-Token,X-Amz-
Target
Content-Type: application/x-amz-json-1.1

{
    "tags": {
        "Status": "Secret",
        "Team": "My-Team"
    }
}
```

ListWorkspaces

The `ListWorkspaces` operation lists all of the Amazon Managed Service for Prometheus workspaces in your account. This includes workspaces being created or deleted.

Request parameters

- *alias*—(Optional) If this is included, it filters the results to only the workspaces with names that start with the value that you specify here.

  Amazon Managed Service for Prometheus will automatically strip any blank spaces from the beginning and end of the alias that you specify.
- *maxResults*—(Optional) The maximum number of workspaces to return in one `ListWorkspaces` operation. The range is 1–1000.

  If you omit this parameter, the default of 100 is used.
- *nextToken*—(Optional) The token for the next set of items to return. (You received this token from a previous call.)

Response elements

- *workspaces*—An array of `WorkspaceSummary` structures. For more information, see WorkspaceSummary structure (p. 108).
- *nextToken*—A token indicating that there are more results to retrieve. You can use this token as part of your next `ListWorkspaces` operation to retrieve those results.

Sample request

```
GET /workspaces?alias=myWorkspace HTTP/1.1
Content-Length: 79,
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": "myWorkspace",
    "tags": {
        "Status": "Secret",
        "Team": "My-Team"
    },
    "clientToken": "6e5b2726-4bb0-4010-a2e3-de91980cf783"
}
```

Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 185
Access-Control-Allow-Methods: OPTION,GET,POST,DELETE
Access-Control-Expose-Headers: x-amzn-errortype,x-amzn-requestid,x-amzn-trace-id,x-amzn-
errormessage,x-amz-apigw-id,date
Strict-Transport-Security: strict-transport-security: max-age=31536000; includeSubDomains
x-amz-apigw-id: 1234EXAMPLE=
X-Amzn-Trace-Id: Root=1-12345678-39c98b574fd600282a1a0f21
Connection: keep-alive
Cache-Control: no-store,no-cache,must-revalidate
Date: Tue, 01 Dec 2020 19:37:25 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: *,Authorization,Date,X-Amz-Date,content-type,x-amz-content-
sha256,x-amz-user-agent,x-amzn-platform-id,x-amzn-trace-id,X-Amz-Security-Token,X-Amz-
Target
Content-Type: application/x-amz-json-1.1

{
    "nextToken": null,
    "workspaces": [
        {
            "alias": "myWorkspace",
            "arn": "arn:aws:aps:us-west-2:123456789012:workspace/ws-b226cc2a-
a446-46a9-933a-ac50479a5568",
            "createdAt": 1.606851445922E9,
            "status": {
                "statusCode": "ACTIVE"
            },
            "workspaceId": "ws-12345678-a446-46a9-933a-ac50479a5568"
        }
    ]
}
```

PutAlertManagerDefinition

The `PutAlertManagerDefinition` operation updates an existing alert manager definition in a workspace. If a workspace does not already have an alert manager definition, don't use this operation to create it. Instead, use CreateAlertManagerDefinition (p. 96).

Request parameters

- *workspaceId*—The ID of the workspace to update the alert manager definition in.
- *data*— The new alert manager definition to use. This is an AlertManagerDefinitionData structure (p. 107).
- *clientToken*—(Optional) A unique, case-sensitive identifier that you can provide to ensure the idempotency of the request.

Response elements

- *status*—A structure that displays the current status of the alert manager definition. For more information, see AlertManagerDefinitionStatus structure (p. 107).

PutRuleGroupsNamespace

The `PutRuleGroupsNamespace` operation updates an existing rule groups namespace within a workspace. A rule groups namespace is associated with exactly one rules file. A workspace can have multiple rule groups namespaces.

Use this operation only to update existing rule groups namespaces. To create a new rule groups namespace, use CreateRuleGroupsNamespace (p. 96).

You can't use this operation to add tags to an existing rule groups namespace. Instead, use TagResource (p. 104).

Request parameters

- *workspaceId*—The ID of the workspace where you are updating the rule groups namespace.
- *name*—The name of the rule groups namespace that you are updating.
- *data*— The new rules file to use in the namespace. This is a RuleGroupsNamespaceData structure (p. 107).
- *clientToken*—(Optional) A unique, case-sensitive identifier that you can provide to ensure the idempotency of the request.

Response elements

- *name*—The name of the new rule groups namespace.
- *status*—A structure that displays the current status of the rule groups namespace. For more information, see RuleGroupsNamespaceStatus structure (p. 108).
- *arn*—The ARN of the rule groups namespace.
- *tags*— The list of tag keys and values that are associated with the namespace.

TagResource

The `TagResource` operation associates tags with an Amazon Managed Service for Prometheus resource. Currently, the only resources that can be tagged are workspaces and rule groups namespaces.

If you specify a new tag key for the resource, this tag is appended to the list of tags associated with the resource. If you specify a tag key that is already associated with the resource, the new tag value that you specify replaces the previous value for that tag.

Request parameters

- *resourceArn*—The ARN of the workspace or rule groups namespace to apply tags to.

- *tags*—The list of tag keys and values to associate with the resource.

Response elements

If the action is successful, the service sends back an HTTP 204 response with an empty HTTP body.

Sample request

```
POST /tags/arn%3Aaws%3Aaps%3Aus-east-2%31234567893884%3Aworkspace%2Fws-effda945-fb72-4e8b-
bbb7-1234567895b6 HTTP/1.1
Content-Type: application/json

X-Amz-Date: 20210829T223002Z
Authorization: AUTHPARAMS
User-Agent: aws-cli/1.18.147 Python/2.7.18 Linux/5.4.58-37.125.amzn2int.x86_64
 botocore/1.18.6

{
    "resourceArn": "arn:aws:aps:us-west-2:123456789012:workspace/ws-12345678-
a446-46a9-933a-ac50479a5568",
    "tags": {
       "Status": "Secret",
       "Team": "My-Team"
   }
}
```

Sample response

```
HTTP/1.1 204 No Content
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 0
Content-Type: application/x-amz-json-1.1
```

UntagResource

The `UntagResource` operation removes the specified tags from an Amazon Managed Service for Prometheus resource. Currently, the only resources that can be tagged are workspaces and rule groups namespaces.

Request parameters

- *resourceArn*—The ARN of the workspace or rule groups namespace.
- *tagKeys*—The keys of the tags to remove.

Response elements

If the action is successful, the service sends back an HTTP 204 response with an empty HTTP body.

Sample request

```
DELETE /tags/arn:aws:aps:us-west-2:123456789012:workspace/ws-12345678-a446-46a9-933a-
ac50479a5568?tagKeys=myNewTag HTTP/1.1
Content-Type: application/json
X-Amz-Date: 20210829T223002Z
Authorization: AUTHPARAMS
User-Agent: aws-cli/1.18.147 Python/2.7.18 Linux/5.4.58-37.125.amzn2int.x86_64
 botocore/1.18.6
```

Sample response

```
HTTP/1.1 204 No Content
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 0
Content-Type: application/x-amz-json-1.1
```

UpdateWorkspaceAlias

The `UpdateWorkspaceAlias` operation updates the alias of an existing workspace.

Request parameters

- *workspaceId*—The ID of the workspace to update.
- *alias*—The new alias for the workspace. It does not need to be unique.

  The alias can be up to 100 characters and can include any type of characters. Amazon Managed Service for Prometheus will automatically strip any blank spaces from the beginning and end of the alias that you specify.
- *clientToken*—(Optional) A unique, case-sensitive identifier that you can provide to ensure the idempotency of the request.

Response elements

- None

Sample request

```
POST /workspaces/ws-1234567890-a446-46a9-933a-ac50479a5568/alias HTTP/1.1
Content-Length: 79,
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": "myWorkspace",
    "clientToken": "6e5b2726-4bb0-4010-a2e3-de91980cf783"
}
```

Sample response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 12345678-abcd-4442-b8c5-262b45e9b535
Content-Length: 185
Access-Control-Allow-Methods: OPTION,GET,POST,DELETE
Access-Control-Expose-Headers: x-amzn-errortype,x-amzn-requestid,x-amzn-trace-id,x-amzn-
errormessage,x-amz-apigw-id,date
Strict-Transport-Security: strict-transport-security: max-age=31536000; includeSubDomains
x-amz-apigw-id: 1234EXAMPLE=
X-Amzn-Trace-Id: Root=1-12345678-39c98b574fd600282a1a0f21
Connection: keep-alive
Cache-Control: no-store,no-cache,must-revalidate
Date: Tue, 01 Dec 2020 19:37:25 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: *,Authorization,Date,X-Amz-Date,content-type,x-amz-content-
sha256,x-amz-user-agent,x-amzn-platform-id,x-amzn-trace-id,X-Amz-Security-Token,X-Amz-
Target
```

```
Content-Type: application/x-amz-json-1.1
```

AlertManagerDefinitionData structure

The `AlertManagerDefinitionData` structure contains the actual alert manager definition as a blob (binary data).

AlertManagerDefinitionDescription structure

The `AlertManagerDefinitionDescription` structure contains the full details about one alert manager definition.

Structure parameters

- *alertManagerDefinitionData*—The actual alert manager definition.
- *alertManagerDefinitionStatus*—A structure that displays the current status of the alert manager definition. For more information, see AlertManagerDefinitionStatus structure (p. 107).
- *createdAt*—The date and time that the alert manager definition was created.
- *modifiedAt*—The date and time that the alert manager definition was most recently changed.

AlertManagerDefinitionStatus structure

The `AlertManagerDefinitionStatus` structure displays status information about the alert manager.

Structure parameters

- *statusCode*—The current status of the alert manager.
- *statusReason*—The reason for the failure, if any.

RuleGroupsNamespaceData structure

The `RuleGroupsNamespaceData` structure contains the rule groups file as a blob (binary data).

RuleGroupsNamespaceDescription structure

The `RuleGroupsNamespaceDescription` structure contains the full details about one rule groups namespace.

Structure parameters

- *name*—The name of the rule groups namespace.
- *arn*—The ARN of the rule groups namespace.
- *status*—A structure that displays the current status of the rule groups namespace. For more information, see RuleGroupsNamespaceStatus structure (p. 108).
- *RuleGroupsNamespaceData*—The rule groups file used in the namespace.
- *createdAt*—The date and time that the rule groups namespace was created.
- *modifiedAt*—The date and time that the rule groups namespace was most recently changed.
- *tags*— (Optional) The list of tag keys and values that are associated with the rule groups namespace.

RuleGroupsNamespaceStatus structure

The `RuleGroupsNamespaceStatus` structure displays status information about the rule groups namespace.

Structure parameters

- *statusCode*—The current status of the namespace. The possible values are `CREATING`, `ACTIVE`, `UPDATING`, `DELETING`, `CREATION_FAILED`, and `UPDATE_FAILED`.
- *statusReason*—The reason for the failure, if any.

RuleGroupsNamespaceSummary structure

The `RuleGroupsNamespaceSummary` structure contains some information about one rule groups namespace. To retrieve more information about one rule groups namespace, use DescribeRuleGroupsNamespace (p. 99).

Structure parameters

- *name*—The name of the rule groups namespace.
- *arn*—The ARN of the rule groups namespace.
- *status*—A structure that displays the current status of the rule groups namespace. For more information, see RuleGroupsNamespaceStatus structure (p. 108).
- *createdAt*—The date and time that the rule groups namespace was created.
- *modifiedAt*—The date and time that the rule groups namespace was most recently changed.
- *tags*— (Optional) The list of tag keys and values that are associated with the rule groups namespace.

WorkspaceDescription structure

The `WorkspaceDescription` structure contains the full details about one Amazon Managed Service for Prometheus workspace in your account.

Structure parameters

- *workspaceId*—The unique ID for the workspace.
- *arn*—The ARN of the workspace.
- *status*—The current status of the workspace. The possible values are `CREATING`, `ACTIVE`, `UPDATING`, `DELETING`, and `CREATION_FAILED`.
- *prometheusEndpoint*—The Prometheus endpoint available for this workspace.
- *createdAt*—The date and time that the workspace was created.
- *tags*— (Optional) The list of tag keys and values that are associated with the workspace.

WorkspaceSummary structure

The `WorkspaceSummary` structure contains information about one Amazon Managed Service for Prometheus workspace in your account.

Structure parameters

- *workspaceId*—The unique ID for the workspace.

- *arn*—The ARN of the workspace.
- *status*—The current status of the workspace. The possible values are `CREATING`, `ACTIVE`, `UPDATING`, `DELETING`, and `CREATION_FAILED`.
- *createdAt*—The date and time that the workspace was created.
- *tags*— (Optional) The list of tag keys and values that are associated with the workspace.

Common errors

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

IncompleteSignature

The request signature does not conform to AWS standards.

HTTP Status Code: 400

InternalFailure

The request processing has failed because of an unknown error, exception, or failure.

HTTP Status Code: 500

InvalidAction

The action or operation requested is not valid. Verify that the action is typed correctly.

HTTP Status Code: 400

InvalidClientTokenId

The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

InvalidParameterCombination

Parameters that must not be used together were used together.

HTTP Status Code: 400

InvalidParameterValue

The supplied value for the input parameter is not valid or is out-of-range.

HTTP Status Code: 400

InvalidQueryParameter

The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

MalformedQueryString

The query string contains a syntax error.

HTTP Status Code: 404

MissingAction

The request is missing an action or a required parameter.

HTTP Status Code: 400

MissingAuthenticationToken

The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

MissingParameter

A required parameter for the specified action is not supplied.

HTTP Status Code: 400

NotAuthorized

You do not have permission to perform this action.

HTTP Status Code: 400

OptInRequired

The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

RequestExpired

The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for presigned URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

ServiceUnavailable

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

ValidationError

The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400

Prometheus-compatible APIs

Amazon Managed Service for Prometheus supports the following Prometheus-compatible APIs.

Topics

DeleteAlertManagerSilence

The `DeleteSilence` deletes one alert silence.

Valid HTTP verbs:

```
DELETE
```

Valid URIs:

```
/workspaces/workspaceId/alertmanager/api/v2/silence/silenceID
```

URL query parameters: none

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

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

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.

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.

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

ListAlertManagerAlerts

The `ListAlertManagerAlerts` retrieves information about the alerts configured in the workspace.

Valid HTTP verbs:

`GET`

Valid URIs:

`/workspaces/workspaceId/alertmanager/api/v2/alerts`

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

`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

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

ListRules

The `ListRules` retrieves information about the rules configured in the workspace.

Valid HTTP verbs:

`GET`

Valid URIs:

`/workspaces/workspaceId/alertmanager/api/v1/rules`

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`

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"
  }
]
```

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 an example:

```
{
  "id": "string",
  "matchers": [
    {
      "name": "string",
      "value": "string",
      "isRegex": true,
      "isEqual": true
    }
  ],
  "startsAt": "2021-09-23T17:01:53.269Z",
  "endsAt": "2021-09-24T17:01:53.269Z",
  "createdBy": "string",
  "comment": "string"
}
```

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.

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

RemoteWrite

The `RemoteWrite` operation writes metrics from a Prometheus server to a remote URL in a standardized format.

Valid HTTP verbs:

`POST`

Valid URIs:

`/workspaces/workspaceId/api/v1/remote_write`

URL query parameters:

None

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.

update-history-change	update-history-description	update-history-date
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

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.