GitHub. The Metrics Server is commonly used by other Kubernetes add ons, such as the Horizontal Pod Autoscaler (p. 349) or the Kubernetes Dashboard (p. 401). For more information, see Resource metrics pipeline in the Kubernetes documentation. This topic explains how to deploy the Kubernetes Metrics Server on your Amazon EKS cluster.

> **Important**
> Don't use Metrics Server when you need an accurate source of resource usage metrics or as a monitoring solution.

Deploy the Metrics Server

1. Deploy the Metrics Server with the following command:

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/
download/components.yaml
```

2. Verify that the `metrics-server` deployment is running the desired number of pods with the following command.

```
kubectl get deployment metrics-server -n kube-system
```

The example output is as follows.

```
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           6m
```

Control plane metrics with Prometheus

The Kubernetes API server exposes a number of metrics that are useful for monitoring and analysis. These metrics are exposed internally through a metrics endpoint that refers to the `/metrics` HTTP API. Like other endpoints, this endpoint is exposed on the Amazon EKS control plane. This topic explains some of the ways you can use this endpoint to view and analyze what your cluster is doing.

Viewing the raw metrics

To view the raw metrics output, use `kubectl` with the `--raw` flag. This command allows you to pass any HTTP path and returns the raw response.

```
kubectl get --raw /metrics
```

The example output is as follows.

```
...
# HELP rest_client_requests_total Number of HTTP requests, partitioned by status code,
 method, and host.
# TYPE rest_client_requests_total counter
rest_client_requests_total{code="200",host="127.0.0.1:21362",method="POST"} 4994
rest_client_requests_total{code="200",host="127.0.0.1:443",method="DELETE"} 1
rest_client_requests_total{code="200",host="127.0.0.1:443",method="GET"} 1.326086e+06
rest_client_requests_total{code="200",host="127.0.0.1:443",method="PUT"} 862173
rest_client_requests_total{code="404",host="127.0.0.1:443",method="GET"} 2
rest_client_requests_total{code="409",host="127.0.0.1:443",method="POST"} 3
rest_client_requests_total{code="409",host="127.0.0.1:443",method="PUT"} 8
# HELP ssh_tunnel_open_count Counter of ssh tunnel total open attempts
# TYPE ssh_tunnel_open_count counter
ssh_tunnel_open_count 0
# HELP ssh_tunnel_open_fail_count Counter of ssh tunnel failed open attempts
# TYPE ssh_tunnel_open_fail_count counter
ssh_tunnel_open_fail_count 0
```

This raw output returns verbatim what the API server exposes. These metrics are represented in a Prometheus format. This format allows the API server to expose different metrics broken down by line. Each line includes a metric name, tags, and a value.

```
metric_name{"tag"="value"[,...]} value
```

While this endpoint is useful if you are looking for a specific metric, you typically want to analyze these metrics over time. To do this, you can deploy Prometheus into your cluster. Prometheus is a monitoring and time series database that scrapes exposed endpoints and aggregates data, allowing you to filter, graph, and query the results.

Deploying Prometheus

This topic helps you deploy Prometheus into your cluster with Helm V3. If you already have Helm installed, you can check your version with the `helm version` command. Helm is a package manager for Kubernetes clusters. For more information about Helm and how to install it, see Using Helm with Amazon EKS (p. 411).

After you configure Helm for your Amazon EKS cluster, you can use it to deploy Prometheus with the following steps.

To deploy Prometheus using Helm

1. Create a Prometheus namespace.

   ```
   kubectl create namespace prometheus
   ```

2. Add the `prometheus-community` chart repository.

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

3. Deploy Prometheus.

   ```
   helm upgrade -i prometheus prometheus-community/prometheus \
       --namespace prometheus \
       --set
    alertmanager.persistentVolume.storageClass="gp2",server.persistentVolume.storageClass="gp2"
   ```

   Note
   If you get the error `Error: failed to download "stable/prometheus" (hint: running `helm repo update` may help)` when executing this command, run `helm repo update`, and then try running the Step 2 command again.
   If you get the error `Error: rendered manifests contain a resource that already exists`, run `helm uninstall *your-release-name* -n *namespace*`, then try running the Step 3 command again.

4. Verify that all of the pods in the `prometheus` namespace are in the `READY` state.

   ```
   kubectl get pods -n prometheus
   ```

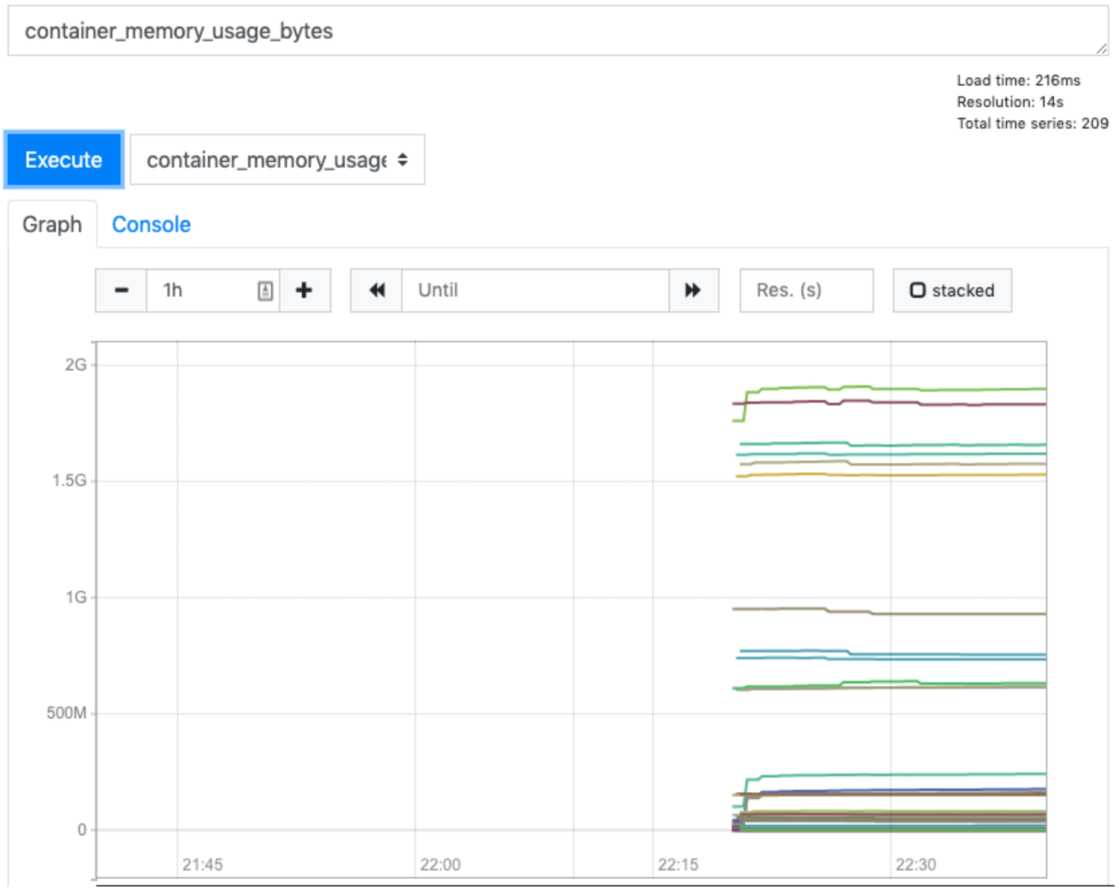
The example output is as follows.

```
NAME                                           READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-59b4c8c744-r7bgp       1/2     Running   0          48s
prometheus-kube-state-metrics-7cfd87cf99-jkz2f 1/1     Running   0          48s
prometheus-node-exporter-jcjqz                 1/1     Running   0          48s
prometheus-node-exporter-jxv2h                 1/1     Running   0          48s
prometheus-node-exporter-vbdks                 1/1     Running   0          48s
prometheus-pushgateway-76c444b68c-82tnw        1/1     Running   0          48s
prometheus-server-775957f748-mmht9             1/2     Running   0          48s
```

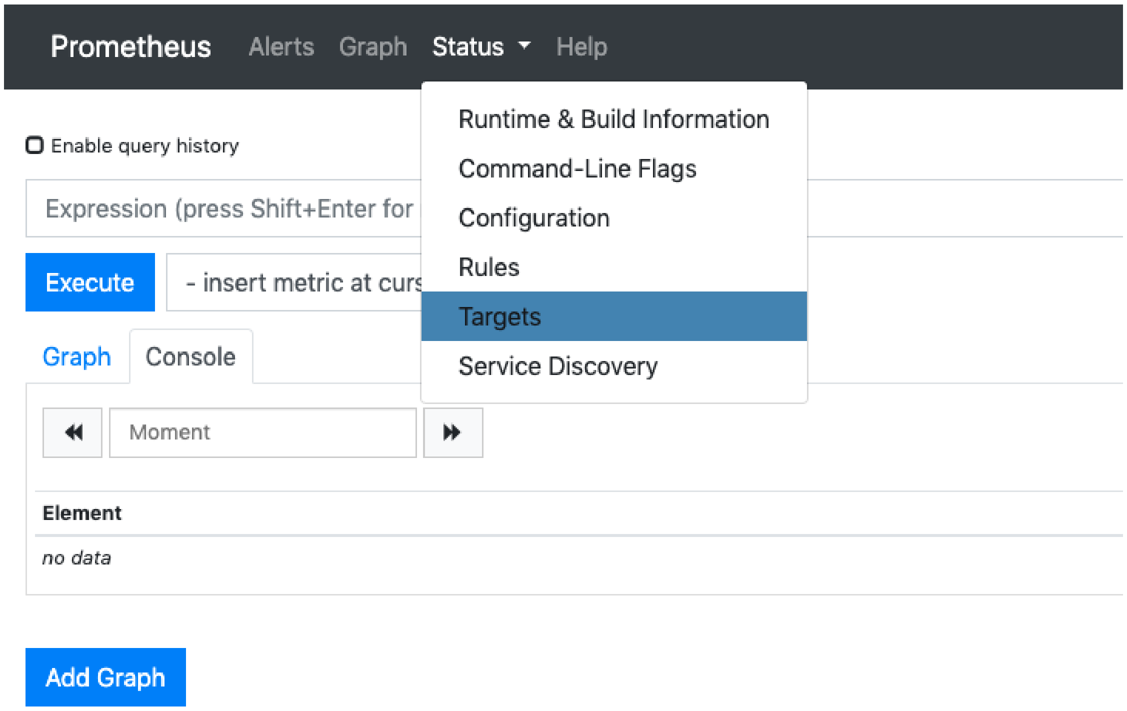
5. Use `kubectl` to port forward the Prometheus console to your local machine.

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. Point a web browser to localhost:9090 to view the Prometheus console.
7. Choose a metric from the **- insert metric at cursor** menu, then choose **Execute**. Choose the **Graph** tab to show the metric over time. The following image shows `container_memory_usage_bytes` over time.



8. From the top navigation bar, choose **Status**, then **Targets**.



All of the Kubernetes endpoints that are connected to Prometheus using service discovery are displayed.

Store your Prometheus metrics in Amazon Managed Service for Prometheus

Amazon Managed Service for Prometheus is a Prometheus-compatible monitoring and alerting service that makes it easy to monitor containerized applications and infrastructure at scale. It is a fully-managed service that automatically scales the ingestion, storage, querying, and alerting of your metrics. It also integrates with AWS security services to enable fast and secure access to your data. You can use the open-source PromQL query language to query your metrics and alert on them.

For more information, see Getting started with Amazon Managed Service for Prometheus.

Using Helm with Amazon EKS

The Helm package manager for Kubernetes helps you install and manage applications on your Kubernetes cluster. For more information, see the Helm documentation. This topic helps you install and run the Helm binaries so that you can install and manage charts using the Helm CLI on your local system.

Important

Before you can install Helm charts on your Amazon EKS cluster, you must configure `kubectl` to work for Amazon EKS. If you have not already done this, see Create a `kubeconfig` for Amazon EKS (p. 392) before proceeding. If the following command succeeds for your cluster, you're properly configured.

```
kubectl get svc
```

To install the Helm binaries on your local system

1. Run the appropriate command for your client operating system.

   - If you're using macOS with Homebrew, install the binaries with the following command.

   ```
   brew install helm
   ```

   - If you're using Windows with Chocolatey, install the binaries with the following command.

   ```
   choco install kubernetes-helm
   ```

   - If you're using Linux, install the binaries with the following commands.

   ```
   curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 >
    get_helm.sh
   chmod 700 get_helm.sh
   ./get_helm.sh
   ```

   Note
   If you get a message that `openssl` must first be installed, you can install it with the following command.

   ```
   sudo yum install openssl
   ```

2. To pick up the new binary in your `PATH`, Close your current terminal window and open a new one.
3. See the version of Helm that you installed.

   ```
   helm version --short | cut -d + -f 1
   ```

   The example output is as follows.

   ```
   v3.9.0
   ```

4. At this point, you can run any Helm commands (such as `helm install chart-name`) to install, modify, delete, or query Helm charts in your cluster. If you're new to Helm and don't have a specific chart to install, you can:

   - Experiment by installing an example chart. See Install an example chart in the Helm Quickstart guide.
   - Create an example chart and push it to Amazon ECR. For more information, see Pushing a Helm chart in the *Amazon Elastic Container Registry User Guide*.
   - Install an Amazon EKS chart from the eks-charts GitHub repo or from ArtifactHub.

Tagging your Amazon EKS resources

You can use *tags* to help you manage your Amazon EKS resources. This topic provides an overview of the tags function and shows how you can create tags.

Topics

- Tag basics (p. 413)
- Tagging your resources (p. 413)
- Tag restrictions (p. 414)

- Tagging your resources for billing (p. 414)
- Working with tags using the console (p. 415)
- Working with tags using the CLI, API, or eksctl (p. 415)

Note
Tags are a type of metadata that's separate from Kubernetes labels and annotations. For more information about these other metadata types, see the following sections in the Kubernetes documentation:

- Labels and Selectors
- Annotations

Tag basics

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

With tags, you can categorize your AWS resources. For example, you can categorize resources by purpose, owner, or environment. When you have many resources of the same type, you can use the tags that you assigned to a specific resource to quickly identify that resource. For example, you can define a set of tags for your Amazon EKS clusters to help you track each cluster's owner and stack level. We recommend that you devise a consistent set of tag keys for each resource type. You can then search and filter the resources based on the tags that you add.

After you add a tag, you can edit tag keys and values or remove tags from a resource at any time. If you delete a resource, any tags for the resource are also deleted.

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

If you use AWS Identity and Access Management (IAM), you can control which users in your AWS account have permission to manage tags.

Tagging your resources

The following Amazon EKS resources support tags:

- clusters
- managed node groups
- Fargate profiles

You can tag these resources using the following:

- If you're using the Amazon EKS console, you can apply tags to new or existing resources at any time. You can do this by using the **Tags** tab on the relevant resource page. For more information, see Working with tags using the console (p. 415).
- If you're using `eksctl`, you can apply tags to resources when they're created using the `--tags` option.
- If you're using the AWS CLI, the Amazon EKS API, or an AWS SDK, you can apply tags to new resources using the `tags` parameter on the relevant API action. You can apply tags to existing resources using the `TagResource` API action. For more information, see TagResource.

When you use some resource-creating actions, you can also specify tags for the resource at the same time that you create it. If tags can't be applied while the resource is being created, the resource fails to be created. This mechanism ensures that resources that you intend to tag are either created with the tags that you specify or not created at all. If you tag resources when you create them, you don't need to run custom tagging scripts after you create the resource.

Tags don't propagate to other resources that are associated with the resource that you create. For example, Fargate profile tags don't propagate to other resources that are associated with the Fargate profile, such as the pods that are scheduled with it.

Tag restrictions

The following restrictions apply to tags:

- A maximum of 50 tags can be associated with a resource.
- Tag keys can't be repeated for one resource. Each tag key must be unique, and can only have one value.
- Keys can be up to 128 characters long in UTF-8.
- Values can be up to 256 characters long in UTF-8.
- If multiple AWS services and resources use your tagging schema, limit the types of characters you use. Some services might have restrictions on allowed characters. Generally, allowed characters are letters, numbers, spaces, and the following characters: **+ - = . _ : /** `@`.
- Tag keys and values are case sensitive.
- Don't use `aws:`, `AWS:`, or any upper or lowercase combination of such as a prefix for either keys or values. These are reserved only for AWS use. You can't edit or delete tag keys or values with this prefix. Tags with this prefix don't count against your tags-per-resource limit.

Tagging your resources for billing

When you apply tags to Amazon EKS clusters, you can use them for cost allocation in your **Cost & Usage Reports**. The metering data in your **Cost & Usage Reports** shows usage across all of your Amazon EKS clusters. For more information, see AWS cost and usage report in the *AWS Billing User Guide*.

The AWS generated cost allocation tag, specifically `aws:eks:cluster-name`, lets you break down Amazon EC2 instance costs by individual Amazon EKS cluster in **Cost Explorer**. However, this tag doesn't capture the control plane expenses. The tag is automatically added to Amazon EC2 instances that participate in an Amazon EKS cluster. This behavior happens regardless of whether the instances are provisioned using Amazon EKS managed node groups, Karpenter, or directly with Amazon EC2. This specific tag doesn’t count towards the 50 tags limit. To use the tag, the account owner must activate it in the AWS Billing console or by using the API. When an AWS Organizations management account owner activates the tag, it's also activated for all organization member accounts.

You can also organize your billing information based on resources that have the same tag key values. For example, you can tag several resources with a specific application name, and then organize your billing information. That way, you can see the total cost of that application across several services. For more information about setting up a cost allocation report with tags, see The Monthly Cost Allocation Report in the *AWS Billing User Guide*.

Note
If you just enabled reporting, data for the current month is available for viewing after 24 hours.

Cost Explorer is a reporting tool that’s available as part of the AWS Free Tier. You can use **Cost Explorer** to view charts of your Amazon EKS resources from the last 13 months. You can also forecast how much you're likely to spend for the next three months. You can see patterns in how much you spend on AWS resources over time. For example, you can use it to identify areas that need further inquiry and see trends that you can use to understand your costs. You also can specify time ranges for the data, and view time data by day or by month.

Working with tags using the console

Using the Amazon EKS console, you can manage the tags that are associated with new or existing clusters and managed node groups.

When you select a resource-specific page in the Amazon EKS console, the page displays a list of those resources. For example, if you select **Clusters** from the left navigation pane, the console displays a list of Amazon EKS clusters. When you select a resource from one of these lists (for example, a specific cluster) that supports tags, you can view and manage its tags on the **Tags** tab.

You can also use **Tag Editor** in the AWS Management Console, which provides a unified way to manage your tags. For more information, see Working with Tag Editor in the *AWS Resource Groups and Tags User Guide*.

Adding tags on a resource on creation

You can add tags to Amazon EKS clusters, managed node groups, and Fargate profiles when you create them. For more information, see Creating an Amazon EKS cluster (p. 24).

Adding and deleting tags on a resource

You can add or delete the tags that are associated with your clusters directly from the resource's page.

To add or delete a tag on an individual resource

1. Open the Amazon EKS console at https://console.aws.amazon.com/eks/home#/clusters.
2. On the navigation bar, select the AWS Region to use.
3. In the left navigation pane, choose **Clusters**.
4. Choose a specific cluster.
5. Choose the **Tags** tab, and then choose **Manage tags**.
6. On the **Manage tags** page, add or delete your tags as necessary.
   - To add a tag, choose **Add tag**. Then specify the key and value for each tag.
   - To delete a tag, choose **Remove tag**.
7. Repeat this process for each tag that you want to add or delete.
8. Choose **Update** to finish.

Working with tags using the CLI, API, or `eksctl`

Use the following AWS CLI commands or Amazon EKS API operations to add, update, list, and delete the tags for your resources. You can only use `eksctl` to add tags while simultaneously creating the new resources with one command.

Tagging support for Amazon EKS resources

Task	AWS CLI	AWS Tools for Windows PowerShell	API action
Add or overwrite one or more tags.	`tag-resource`	`Add-EKSResourceTag`	`TagResource`