ALIBABA CLOUD

Alibaba Cloud

Elastic Container Instance
Logging & monitoring

Document Version: 20210826

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Log collection ---- 05
1.1. Use Log Service to collect container logs from an ECI ---- 05
1.2. Parse user logs in the JSON format ---- 06
2.Connect ASK clusters to ARMS ---- 08
2.1. Connect ASK clusters to ARMS Application Monitoring ---- 08
2.2. Connect ASK clusters to ARMS Prometheus Monitoring ---- 12
2.3. Use ARMS Prometheus Monitoring to monitor a GPU-accel... ---- 16
2.4. Use ARMS Prometheus Monitoring to monitor disks ---- 18

1.Log collection

1.1. Use Log Service to collect container logs from an ECI

Prerequisites:

- The virtual-kubelet node is deployed in the target Kubernetes cluster. Note that a serverless Kubernetes cluster is embedded with the virtual-kubelet node.
- Log Service is enabled for the Kubernetes cluster.

Collect container logs from an ECI

You can use environment variables to specify collection configurations and custom tags for a container. Then, you can use the volumes and volumeMounts fields to configure a volume and the directory to which the volume is mounted based on the log collection configuration. The following configuration file of a simple pod shows how to use environment variables to specify collection configurations and custom tags for a container:

```
apiVersion: v1
kind: Pod
metadata:
 name: say-hello
spec:
 containers:
 - image: registry.cn-beijing.aliyuncs.com/dzf/busybox:1.28.3
  imagePullPolicy: IfNotPresent
  name: busybox
  command: ["/bin/sh","-c","while true; do echo $(date) hello logfile. >> /var/log/sayhi.log echo $(date) hello,
stdout.>>1 ; sleep 10; done"]
  env:
  - name: aliyun_logs_log-stdout
   value: stdout
  - name: aliyun_logs_log-varlog
   value: /var/log/*.log
  - name: aliyun_logs_appname_tags
   value: appname=say-hello
  - name: aliyun_logs_version_tags
   value: version=1.28.3
  volumeMounts:
  - name: volumn-sls-sayhi
   mountPath: /var/log
 volumes:
 - name: volumn-sls-sayhi
  emptyDir: {}
```

Note: A Logstore name cannot contain underscores (_). You can use hyphens (-) instead.

Use environment variables to specify **collection configurations** and **custom tags**. All environment variables related to log collection must be prefixed with `aliyun_logs_`.

Specify the following configurations in order based on your needs:

1. **Logstore**

```
- name: aliyun_logs_{Logstore name}
  value: {Log path}
```

In the preceding example, two environment variables are used to specify collection configurations. The `aliyun_logs_log-stdout` environment variable instructs the system to create a Logstore named log-stdout, which collects the standard output of the container.

2. **Custom tags**

```
- name: aliyun_logs_{Tag name without underscores (_)}_tags
  value: {Tag name}={Tag value}
```

After a custom tag is specified, it is automatically appended to certain log fields when logs from the specified container are collected.

3. **Path for collecting log files other than the standard output**

If you specify a path for collecting log files other than the standard output, you need to add the volumnMounts field. In the preceding example, the .log files in the /var/log directory are to be collected. Therefore, the volumeMounts field is added, where mountPath is set to /var/log.

For more information about the advanced configurations of environment variables, see the Advanced configurations section in Use Log Service to collect Kubernetes cluster logs.

1.2. Parse user logs in the JSON format

The standard output and error logs collected by Elastic Container Instance (ECI) are logs flushed to disks in the native format of Kubernetes. Kubernetes prefixes each line of log with information such as the timestamp and source, which corrupts the native format of user logs. For example, if the standard output is in the JSON format, Log Service fails to parse the standard output after Kubernetes adds the information to it. The following data is the sample of standard output with the information added by Kubernetes.

```
2020-04-02T15:40:05.440500764+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:07.442412564+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:09.442774495+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:11.443799303+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:13.445099622+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:15.445934358+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:17.447064707+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:19.448112987+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:21.449393263+08:00 stdout F {"key1":"val1","key2":"val2"}
```

This topic describes how to use a Log Service processor to parse user logs in the JSON format.

User logs of ECI are collected to a Logstore under your account. Log on to the Log Service console. Find the target Logstore and modify its configurations. On the Logtail Config page, set the Mode parameter to Simple Mode and turn on the Enable Plug-in Processing switch.

Enter the following code in the Plug-in Config field. For more information, see sls-json-processor.

```
{
  "processors": [
    {
      "type": "processor_anchor",
      "detail": {
        "SourceKey": "content",
        "Anchors": [
          {
            "Start": "stdout F ",
            "Stop": "",
            "FieldName": "json_content",
            "FieldType": "string",
            "ExpondJson": false
          }
        ]
      }
    },
    {
      "type": "processor_json",
      "detail": {
        "SourceKey": "json_content",
        "KeepSource": false,
        "ExpandConnector": ""
      }
    }
  ]
}
```

Save the configurations. A few seconds later, you can view the logs parsed in the correct format.

In this way, Log Service correctly parses user logs in the JSON format.

2.Connect ASK clusters to ARMS

2.1. Connect ASK clusters to ARMS Application Monitoring

You can use Application Real-Time Monitoring Service (ARMS) Application Monitoring to monitor the topologies, API requests, abnormal transactions, slow transactions, and slow SQL queries of applications in serverless Kubernetes (ASK) clusters. This topic describes how to connect ASK clusters to ARMS Application Monitoring.

Prerequisites

- An ASK cluster is created.
- ARMS is activated. For more information, see Activate and upgrade ARMS.

> **Note**
>
> You can receive a free 15-day trial of the Application Monitoring sub-service of ARMS. After the free trial expires, you must activate the Basic Edition or Pro Edition of ARMS Application Monitoring for continual use. For more information, see the Application Real-Time Monitoring Service pricing page.

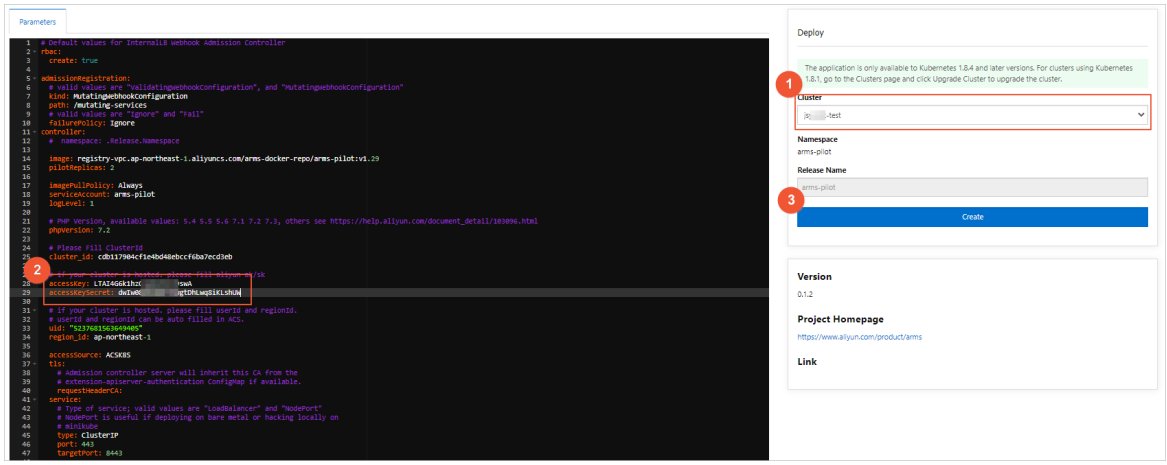
Background information

ARMS is an Application Performance Management (APM) service that contains modules such as Application Monitoring and Prometheus Monitoring. ARMS can help you perform full-stack performance monitoring and full-trace analysis in an end-to-end manner to simplify application O&M.

After you install the ARMS Application Monitoring agent in an ASK cluster, ARMS can perform comprehensive monitoring on the applications deployed in the ASK cluster and help you easily identify abnormal and slow API operations, view request parameters, and detect system bottlenecks. This way, the efficiency of online problem diagnostics can be significantly improved. For more information, see Overview.

Step 1: Install the Application Monitoring agent

1. Log on to the Container Service - Kubernetes console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**.
3. On the **Alibaba Cloud Apps** tab, click the **ack-arms-pilot** application.
4. Configure the parameters and select the cluster for which you want to install the agent.
   i. Select the cluster for which you want to install the agent from the drop-down list.
   ii. On the **Parameters** tab, specify the accessKey and accessKeySecret parameters in the YAML template.
   iii. Click **Create**.



5. Check the installation result.

   Click the cluster name to go to the cluster information page. In the left-side navigation pane, choose **Applications > Helm** to check whether the status of arms-pilot is **Deployed**.

Step 2: Enable Application Monitoring for applications

You can add annotations to the YAML template of an application to enable Application Monitoring. Add annotations to spec > template > metadata.

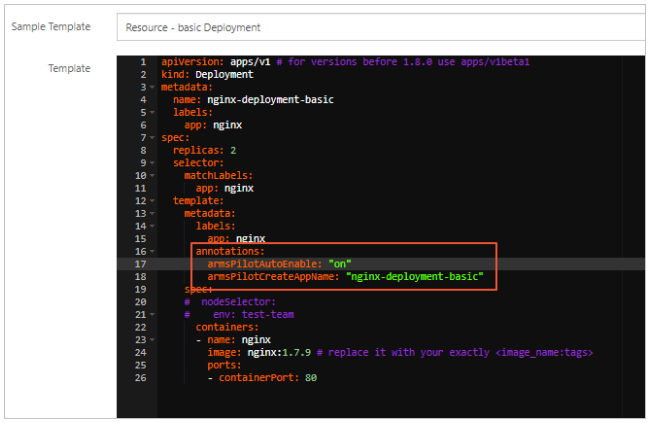
```
annotations:
 armsPilotAutoEnable: "on"
 armsPilotCreateAppName: "<your-deployment-name>"
```

- Enable Application Monitoring for a new application

  i. In the Container Service - Kubernetes console, find the cluster in which you want to create an application and click the cluster name. On the page that appears, choose **Workloads > Deployments** in the left-side navigation pane.

  ii. In the upper-right corner, click **Create from YAML**.

  iii. Select a namespace and a sample template from the drop-down lists, add annotations to spec > template > metadata, and then click **Create**.

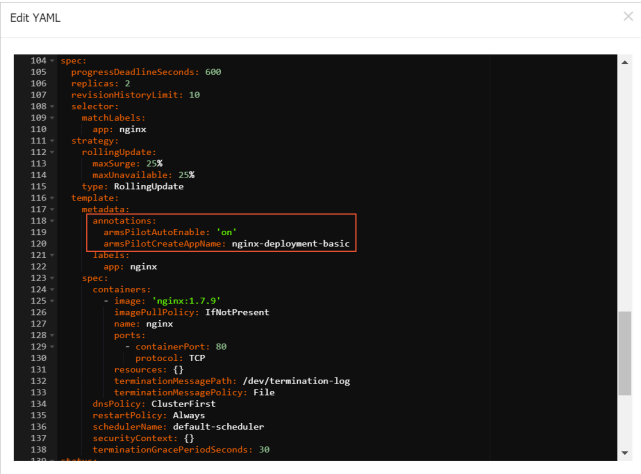


The following example describes a complete YAML template used to enable ARMS Application Monitoring for an application. The Java application Spring Cloud Eureka Server is used in the example.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: register-server
spec:
 replicas: 3
 serviceName: register-server
 selector:
  matchLabels:
   app: register-server
 template:
  metadata:
   labels:
    app: register-server
   annotations:
    armsPilotAutoEnable: "on"             # Enable ARMS Application Monitoring.
    armsPilotCreateAppName: "register-server"   # Specify the name of the application for which to e
nable ARMS Application Monitoring.
  spec:
   containers:
    - name: register-server
      image: registry.cn-hangzhou.aliyuncs.com/shuangling/eureka-server:v1
      imagePullPolicy: Always
      env:
      - name: EUREKA_DEFAULT_ZONE
        value: "http://register-server-0.register-server:8000/eureka/,http://register-server-1.register-se
rver:8000/eureka/,http://register-server-2.register-server:8000/eureka/"
      - name: JVM_OPTS
        value: " -Xms1024m -Xmx1536m "
      - name: MY_POD_NAME
        valueFrom:
          fieldRef:
            fieldPath: metadata.name
      ports:
        - name: http
          containerPort: 8000
          protocol: TCP
      readinessProbe:
        httpGet:
          path: /actuator/health
          port: 8001
          scheme: HTTP
        failureThreshold: 3
        initialDelaySeconds: 60
        periodSeconds: 10
        successThreshold: 1
        timeoutSeconds: 10
      volumeMounts:
      - mountPath: /Charts
        name: data
   volumes:
   - name: data
     emptyDir: {}
 podManagementPolicy: "Parallel"
---
```

```
apiVersion: v1
kind: Service
metadata:
 name: register-server
 labels:
  app: register-server
spec:
 clusterIP: None
 type: ClusterIP
 ports:
  - port: 8000
    targetPort: http
    protocol: TCP
    name: http
 selector:
  app: register-server
```

- Enable Application Monitoring for an existing application
  i. In the Container Service - Kubernetes console, find the cluster where the application is deployed and click the cluster name. On the page that appears, choose **Workloads > Deployments** or **Workloads > StatefulSets** in the left-side navigation pane.
  ii. Find the application for which you want to enable Application Monitoring, click **More**, and then select **View in YAML** in the Actions column.
  iii. Edit the YAML template by adding annotations to spec > template > metadata.



  iv. Click **Update**.

  After the YAML template is updated, the container groups are recreated. This process may take some time. Wait until the rolling update is completed for all container groups before you can view the monitoring data.

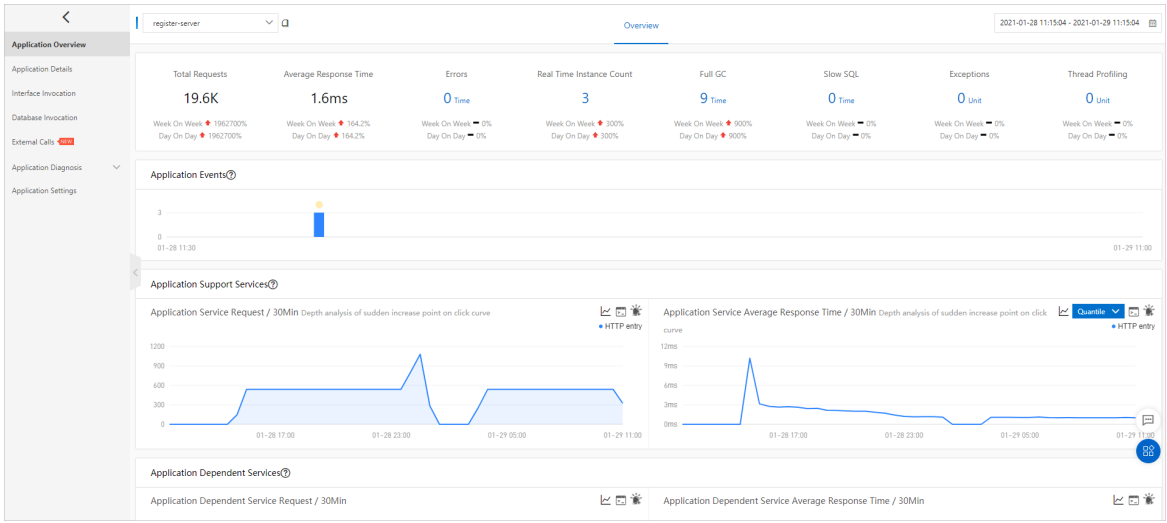
Step 3: View monitoring data

After ARMS Application Monitoring is enabled, you can view data on the **Application Monitoring** page in the ARMS console.

1. In the Container Service - Kubernetes console, find the cluster where the application is deployed and click the cluster name. On the page that appears, choose **Applications > Deployments** or **Applications > StatefulSets** in the left-side navigation pane.

2. Find the application whose monitoring data you want to view, and click **ARMS Console** in the Actions column.
3. View monitoring data of the application.

   ARMS Application Monitoring can discover application topologies, capture abnormal and slow transactions, and diagnose performance in a real-time manner. For more information about how to use ARMS Application Monitoring, see Application overview.



2.2. Connect ASK clusters to ARMS Prometheus Monitoring

After serverless Kubernetes (ASK) clusters are connected to Application Real-Time Monitoring Service (ARMS) Prometheus Monitoring, you can use the dashboard predefined in ARMS to monitor multiple performance metrics of the ASK clusters. This topic describes how to connect ASK clusters to ARMS Prometheus Monitoring.

Prerequisites

- An ASK cluster is created.

  > **Note**
  >
  > If the security group in the cluster is manually modified after it is automatically created, make sure that ports 8080, 8081, and 9335 are enabled.

- ARMS is activated. For more information, see Activate and upgrade ARMS.

  > **Note**
  >
  > You can receive a free 15-day trial of the Prometheus Monitoring sub-service of ARMS. After the free trial expires, you must activate the Pro Edition of ARMS Prometheus Monitoring for continual use. For more information, see the Application Real-Time Monitoring Service pricing page.

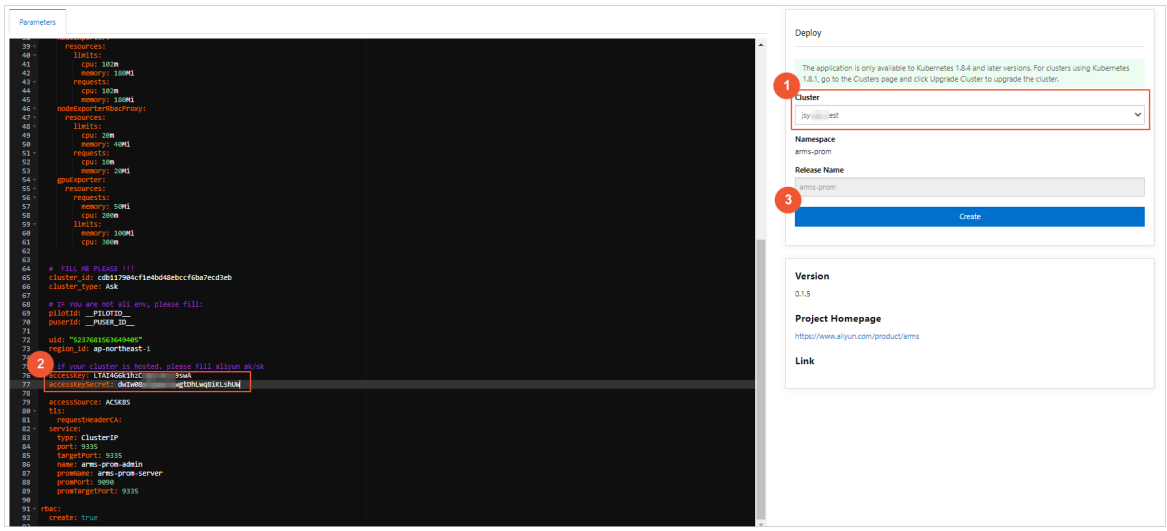
Background information

ARMS is an Application Performance Management (APM) service that contains modules such as Application Monitoring and Prometheus Monitoring. ARMS can help you perform full-stack performance monitoring and full-trace analysis in an end-to-end manner to simplify application O&M.

ARMS Prometheus Monitoring is a managed monitoring service of ARMS and compatible with the open source Prometheus ecosystem. ARMS Prometheus Monitoring monitors a wide variety of components and provides various ready-to-use predefined dashboards. You no longer need to concern yourself with managing underlying services such as data storage, data presentation, and system O&M. For more information, see What is Prometheus Service?

Install the Prometheus Monitoring agent

1. Log on to the Container Service - Kubernetes console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**.
3. On the **Alibaba Cloud Apps** tab, click the **ack-arms-prometheus** application.
4. Configure the parameters and select the cluster for which you want to install the agent.
   i. Select the cluster for which you want to install the agent from the drop-down list.
   ii. On the **Parameters** tab, specify the accessKey and accessKeySecret parameters in the YAML template.
   iii. Click **Create**.



5. Check the installation result.

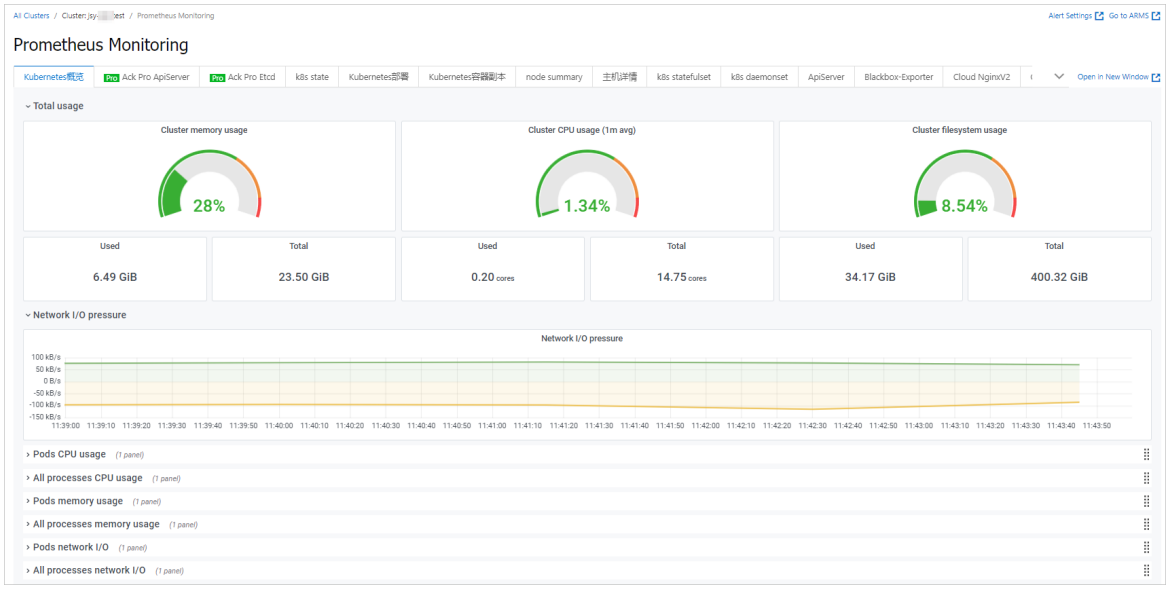
   Click the cluster name to go to the cluster information page. In the left-side navigation pane, choose **Applications > Helm** to check whether the status of arms-prom is **Deployed**.

View ARMS Prometheus Monitoring metrics

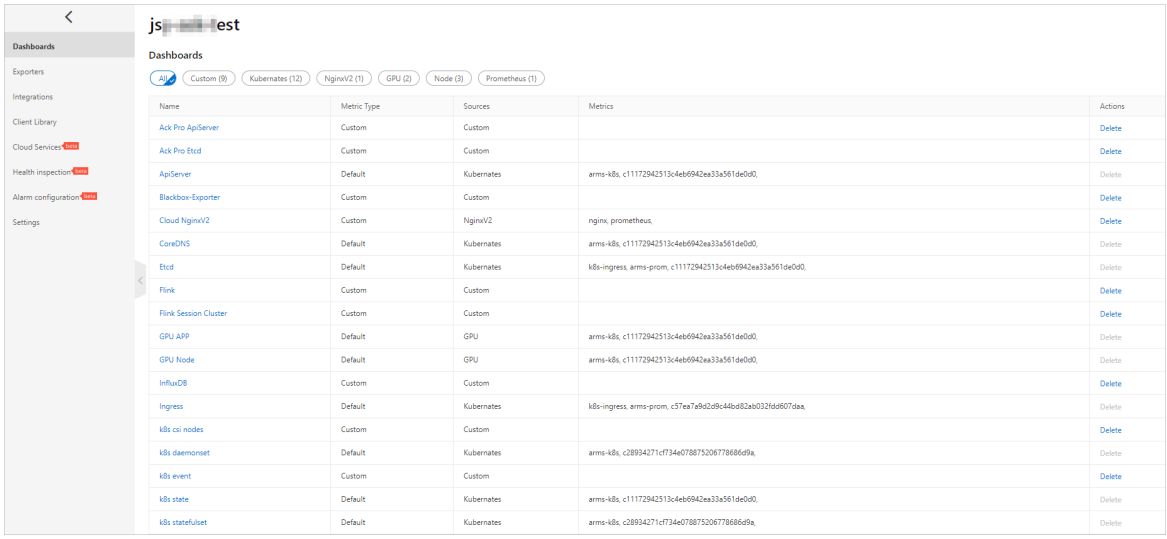
After the monitoring agent is installed, you can view the detailed monitoring information on the **Prometheus Monitoring** page in the ARMS console.

1. In the left-side navigation pane of the Container Service - Kubernetes console, click **Clusters**.
2. Find the cluster whose monitoring data you want to view and click the cluster name.
3. On the **Cluster Information** page, click **Prometheus Monitoring** in the upper-right corner.
4. View the Prometheus Monitoring metrics.

You can switch between tabs to view the metrics displayed on different boards. For more information, see View Prometheus Monitoring metrics.



In the upper-right corner, click **Go to ARMS** to redirect to the ARMS console and configure the Prometheus Monitoring dashboard for the cluster.



Configure a data collection rule for Prometheus Monitoring

ARMS Prometheus Monitoring is compatible with and provides three types of mainstream collection rules:

- The standard open source collection rule, which can be obtained by editing the prometheus.yaml file.
- The collection rule suitable for customizing monitoring within Kubernetes clusters. This rule can be obtained by adding ServiceMonitor.
- The default collection rule, which can be obtained by adding annotations.

You can configure the three types of rules by using the following methods:

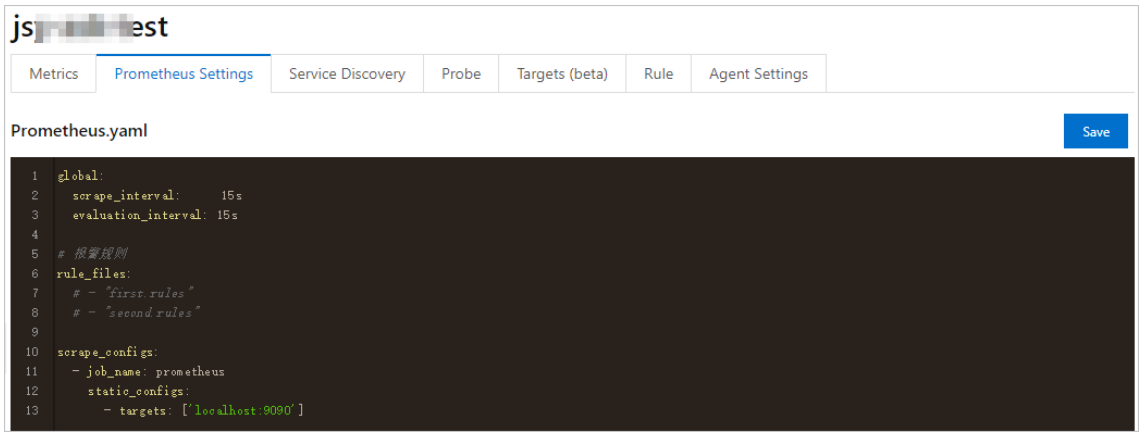
- Edit prometheus.yaml

You do not need to restart Prometheus Monitoring. You can dynamically update a collection rule by editing the prometheus.yaml file.

i. Log on to the ARMS console and go to the **Prometheus Monitoring** page. Find the cluster for which you want to configure a collection rule and click **Settings**.

ii. On the page that appears, click the **Prometheus Settings** tab.

iii. Edit the Prometheus.yaml file and click **Save**.



- Add ServiceMonitor

  After you add ServiceMonitor, you can monitor business data of applications within a Kubernetes cluster.

  i. Log on to the ARMS console and go to the **Prometheus Monitoring** page. Find the cluster for which you want to configure a collection rule and click **Settings**.

  ii. On the page that appears, click the **Service Discovery** tab. Click **Add ServiceMonitor**.

  iii. Enter the content by referring to the following example and click **OK**:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
 # Enter a unique name.
 name: tomcat-demo
 # Enter a namespace.
 namespace: default
spec:
 endpoints:
 - interval: 30s
  # Enter the value of the Name field for Port of Prometheus Exporter.
  port: tomcat-monitor
  # Enter the value of the Path field for Prometheus Exporter.
  path: /prometheus-metrics
 namespaceSelector:
  any: true
 selector:
  matchLabels:
   # Enter the label field of service.yaml to locate the destination service.yaml file.
   app: tomcat
```

- Add annotations

In the YAML file of the application, enter the following content to add annotations:

```
annotations:
      prometheus.io/scrape: "true"
      prometheus.io/port: "9090"
      prometheus.io/path: "/metrics"
```

2.3. Use ARMS Prometheus Monitoring to monitor a GPU-accelerated elastic container instance

After a serverless Kubernetes (ASK) cluster is connected to Application Real-Time Monitoring Service (ARMS) Prometheus Monitoring, you can use the dashboards predefined in ARMS to monitor performance metrics of the GPU-accelerated elastic container instances in the cluster. This topic describes how to use ARMS Prometheus Monitoring to monitor a GPU-accelerated elastic container instance.

Prerequisites

An ASK cluster is created and connected to ARMS Prometheus Monitoring. For more information, see Connect ASK clusters to ARMS Prometheus Monitoring.

Procedure

1. Log on to the Container Service console.
2. Create a GPU-accelerated elastic container instance.

   YAML example:

   ```
   apiVersion: v1
   kind: Pod
   metadata:
    name: cg-gpu-0
    annotations:
     # Specify a GPU-accelerated instance type.
     k8s.aliyun.com/eci-use-specs : "ecs.gn6i-c4g1.xlarge"
   spec:
    containers:
    - image: nginx
     name: cg
     resources:
      limits:
       cpu: 500m
       # Specify the number of GPUs allocated to a container.
       nvidia.com/gpu: '1'
     command: ["bash","-c","sleep 100000"]
    dnsPolicy: ClusterFirst
    restartPolicy: Always
   ```

3. View GPU metrics.

i. Find the cluster to which the created GPU-accelerated elastic container instance belongs and click the cluster name.

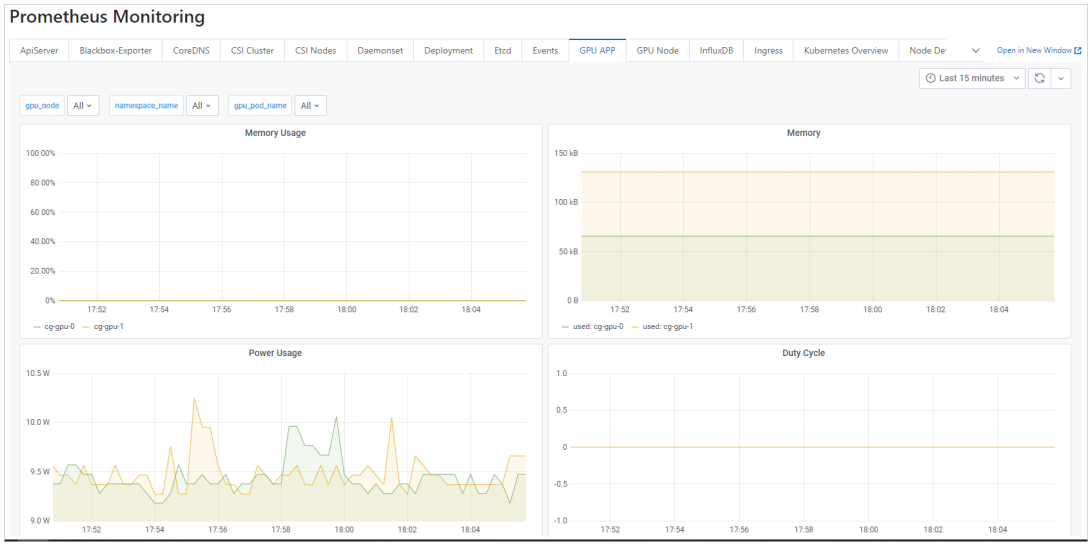
ii. On the **Cluster Information** page, click **Prometheus Monitoring** in the upper-right corner.

iii. On the **GPU APP** or **GPU Node** tab, view monitoring data.

After an ASK cluster is connected to ARMS Prometheus Monitoring, you can monitor the GPU-accelerated elastic container instances in the cluster without the need to install additional plug-ins. By default, ARMS Prometheus Monitoring provides ready-to-use predefined monitoring dashboards.

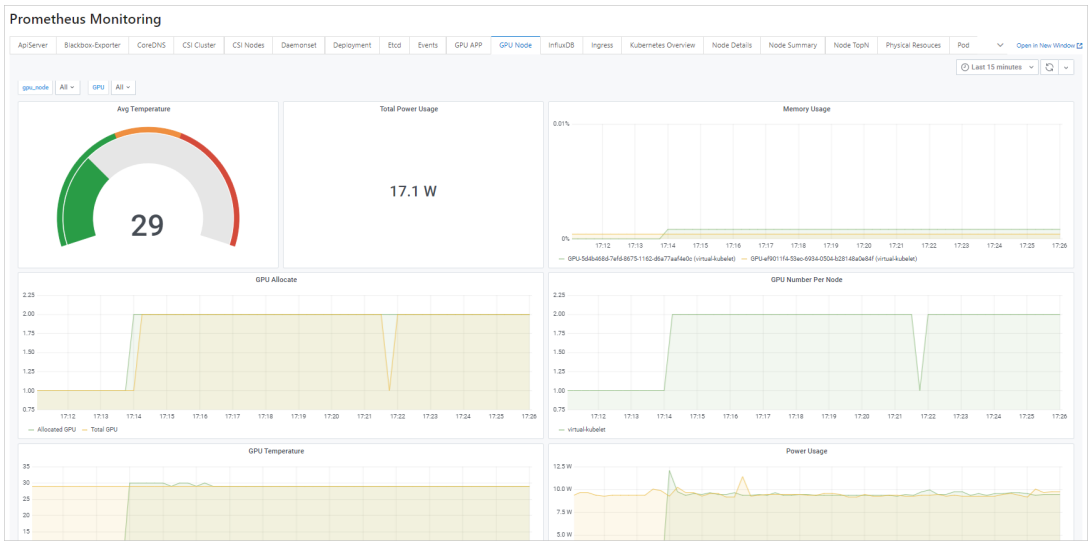
- GPU APP

  In the GPU APP dashboard, you can view monitoring data about GPUs on a single pod, as shown in the following figure.



- GPU Node

  In the GPU Node dashboard, you can view monitoring data about all GPUs on the node, as shown in the following figure.



2.4. Use ARMS Prometheus Monitoring to monitor disks

After elastic container instances are created in a Kubernetes cluster, the instances run on the virtual node. Due to the absence of real nodes, disks are related to pods, instead of to nodes. You must configure pod-level monitoring of disks before you can monitor disk performance metrics. This topic describes how to use Application Real-Time Monitoring Service (ARMS) Prometheus Monitoring to monitor the disks attached to elastic container instances.

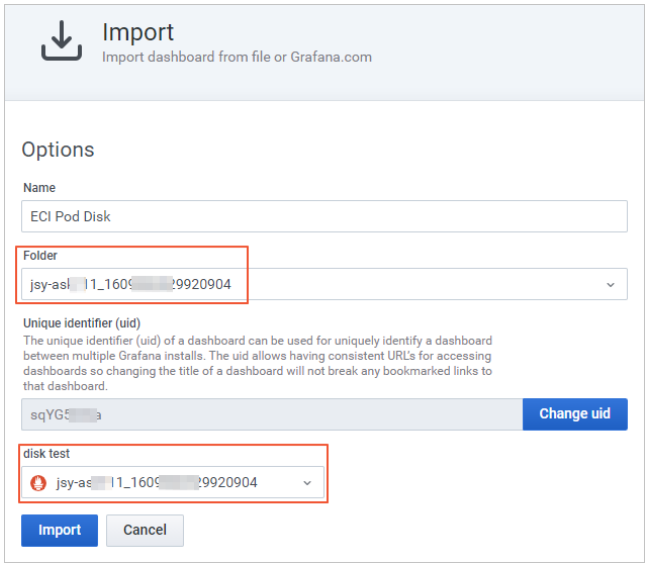
Prerequisites

A serverless Kubernetes (ASK) cluster is created and connected to ARMS Prometheus Monitoring. For more information, see Connect ASK clusters to ARMS Prometheus Monitoring.

Procedure

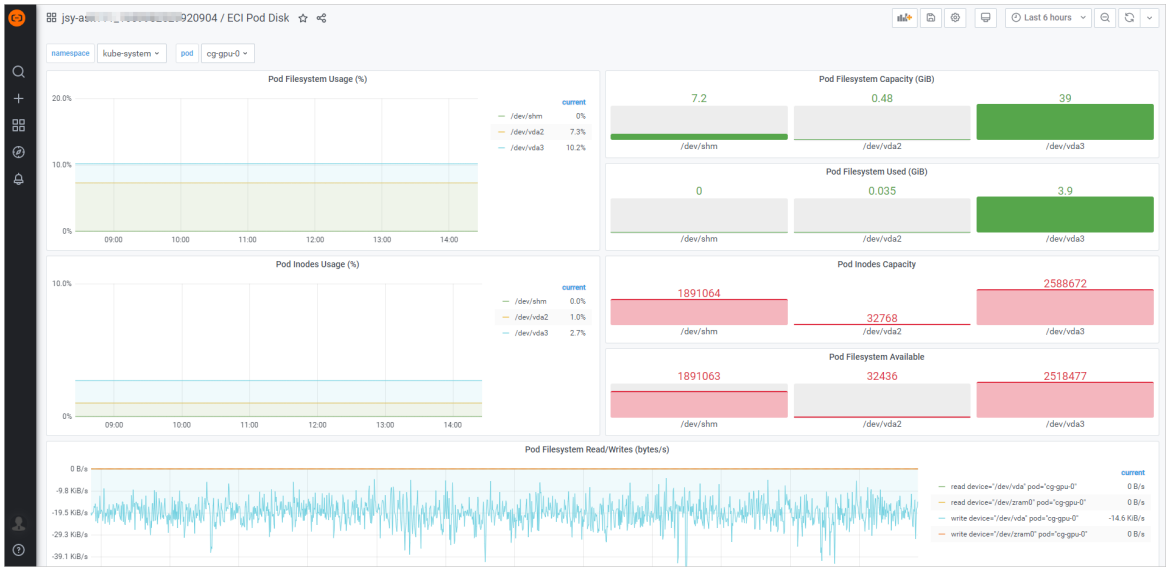
1. Log on to the Container Service console.
2. Go to the Grafana page.
   i. On the **Clusters** page, find the cluster that you want to monitor and click the cluster name.
   ii. On the **Cluster Information** page, click **Prometheus Monitoring** in the upper-right corner.
   iii. On the **Prometheus Monitoring** page, click **Open in New Window** in the upper-right corner.
3. In the left-side navigation pane, click the icon and select **Import**.
4. Click **Upload JSON file** to upload a JSON file.

   The JSON file contains dashboard configurations. For information about an example file, see Dashboard template for monitoring disks for pods.
5. Set both the folder and data source of the dashboard to the cluster that you want to monitor.

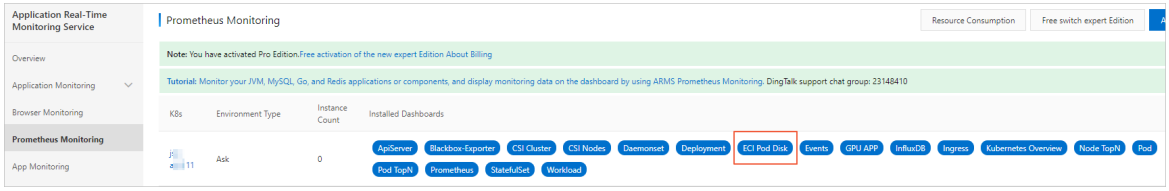


6. Click **Import**.

   After the JSON file is imported, you can view monitoring data about disks on the elastic container instances.



On the **Prometheus Monitoring** page of the ARMS console, you can view the dashboard list of the cluster and find the dashboard that you created.



7. (Optional) To view Deployment- or StatefulSet-level monitoring data, you can copy the corresponding panel from the current disk monitoring dashboard to the monitoring dashboard of the Deployment or StatefulSet within the same cluster or modify the settings of the disk monitoring dashboard.

   When you modify the disk monitoring dashboard settings, perform the following steps to filter variables:

   i. On the Grafana page of the disk monitoring dashboard, click the icon in the upper-right corner.

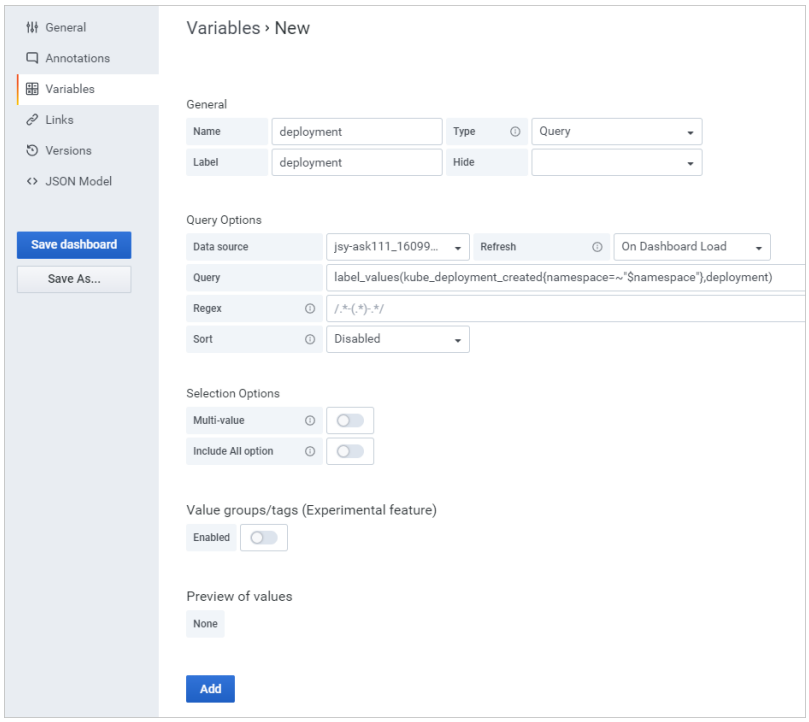
   ii. In the left-side navigation pane of the **Settings** page, click **Variables**.

iii. Click **New**, configure parameters in the General and Query Options sections, and then click **Add**.

When you add variables, set Query based on the variable types. The following table describes example Query settings for different variable types.

Type	Query
deployment	label_values(kube_deployment_created{namespace=~"$namespace"},deployment)
statefulset	label_values(kube_statefulset_created{namespace=~"$namespace"},statefulset)

The following figure shows an example on how to add a variable named deployment.



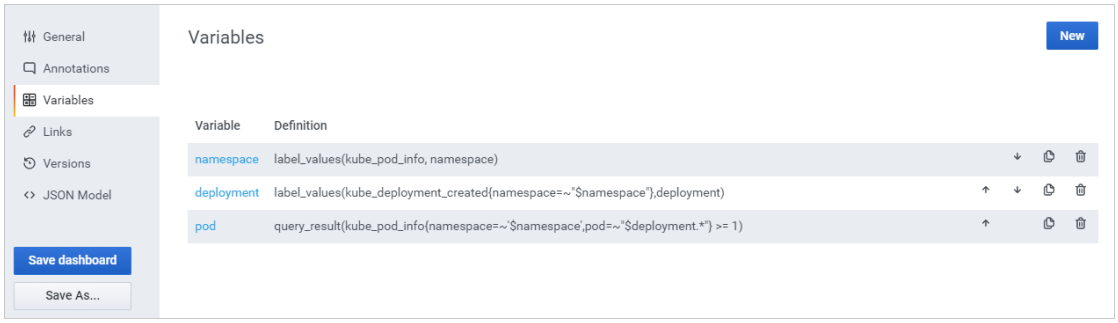
iv. Click the default variable named pod, modify Query on the **Edit** page, and then click **Update**.

Modify Query based on the type of the variable added in the previous step. The following table describes example Query settings for different variable types.

Type	Query for the pod
deployment	query_result(kube_pod_info{namespace=~'$namespace',pod=~"$deployment.*"} >= 1)
statefulset	query_result(kube_pod_info{namespace=~'$namespace',pod=~"$statefulset.*"} >= 1)

v. Adjust the order of variables until the deployment variable or a variable named statefulset is in a higher position than the pod variable.

The following figure shows that the deployment variable is in a higher position than the pod variable.



vi. Go back to the disk monitoring dashboard to check whether the filtering feature has taken effect.

The following figure shows an example of the disk monitoring dashboard where the Deployment-level filtering feature has taken effect.