

Application Operations Management

Best Practices

Issue 01
Date 2024-11-14



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: <https://www.huaweicloud.com/intl/en-us/>

Contents

1 AOM Best Practices.....	1
2 Building a Comprehensive Metric System.....	4
3 Alarm Noise Reduction.....	13
4 Unified Metric Monitoring.....	18
5 Customizing OS Images to Automatically Connect UniAgent.....	27
6 Connecting Self-Built Middleware in the CCE Container Scenario.....	31
6.1 Connecting PostgreSQL Exporter.....	31
6.2 Connecting MySQL Exporter.....	36
6.3 Connecting Kafka Exporter.....	40
6.4 Connecting Memcached Exporter.....	44
6.5 Connecting MongoDB Exporter.....	48
6.6 Connecting Elasticsearch Exporter.....	52
6.7 Connecting Redis Exporter.....	56
6.8 Connecting Other Exporters.....	61
7 Interconnecting Third-Party/IDC/Huawei Cloud Cross-Region Self-Built Prometheus with AOM Prometheus Instances.....	62

1 AOM Best Practices

This document summarizes Application Operations Management (AOM) best practices and provides solutions and operation guide to help you easily use AOM.

Table 1-1 AOM best practices

Best Practice	Description
2 Building a Comprehensive Metric System	This section describes how to build a metric system and a dashboard for all-round, multi-dimensional, and visualized monitoring of resources and applications.
3 Alarm Noise Reduction	This section describes how to set alarm noise reduction. Before sending an alarm notification, AOM processes alarms based on noise reduction rules to prevent alarm storms.
4 Unified Metric Monitoring	This section describes how to centrally monitor metric data of different accounts.
5 Customizing OS Images to Automatically Connect UniAgent	This section describes how to package images for connecting UniAgent in the Linux and Windows environments.
6 Connecting Self-Built Middleware in the CCE Container Scenario	6.1 Connecting PostgreSQL Exporter When using PostgreSQL, you need to monitor their status and locate their faults in a timely manner. The Prometheus monitoring function monitors PostgreSQL running using Exporter in the CCE container scenario. This section describes how to deploy PostgreSQL Exporter and implement alarm access.

Best Practice	Description
	<p>6.2 Connecting MySQL Exporter</p> <p>MySQL Exporter collects MySQL database metrics. Core database metrics collected through Exporter are used for alarm reporting and dashboard display. Currently, Exporter supports MySQL 5.6 or later. If the MySQL version is earlier than 5.6, some metrics may fail to be collected.</p>
	<p>6.3 Connecting Kafka Exporter</p> <p>When using Kafka, you need to monitor their running, for example, checking the cluster status and whether messages are stacked. The Prometheus monitoring function monitors Kafka running using Exporter in the CCE container scenario. This section describes how to deploy Kafka Exporter and implement alarm access.</p>
	<p>6.4 Connecting Memcached Exporter</p> <p>When using Memcached, you need to monitor their running and locate their faults in a timely manner. The Prometheus monitoring function monitors Memcached running using Exporter in the CCE container scenario. This section describes how to monitor Memcached.</p>
	<p>6.5 Connecting MongoDB Exporter</p> <p>When using MongoDB, you need to monitor MongoDB running and locate their faults in a timely manner. The Prometheus monitoring function monitors MongoDB running using Exporter in the CCE container scenario. This section describes how to deploy MongoDB Exporter and implement alarm access.</p>
	<p>6.6 Connecting Elasticsearch Exporter</p> <p>When using Elasticsearch, you need to monitor Elasticsearch running, such as the cluster and index status. The Prometheus monitoring function monitors Elasticsearch running using Exporter in the CCE container scenario. This section describes how to deploy Elasticsearch Exporter and implement alarm access.</p>
	<p>6.7 Connecting Redis Exporter</p> <p>When using Redis, you need to monitor Redis running and locate their faults in a timely manner. The Prometheus monitoring function monitors Redis running using Exporter in the CCE container scenario. This section describes how to monitor Redis.</p>

Best Practice	Description
	6.8 Connecting Other Exporters Guidance has been provided for connecting common Exporters. AOM is compatible with the native Prometheus, so you can also connect other Exporters in the community.
7 Interconnecting Third-Party/IDC/Huawei Cloud Cross-Region Self-Built Prometheus with AOM Prometheus Instances	It is common for cloud users to interconnect IDC/ third-party self-built Prometheus with AOM Prometheus instances.

2 Building a Comprehensive Metric System

This section describes how to build a metric system and a dashboard for all-round, multi-dimensional, and visualized monitoring of resources and applications.

Scenario

In the Internet era, user experience is the top priority. The page response speed, access latency, and access success rate often affect user experience. If such information cannot be obtained in a timely manner, a large number of users will be lost. O&M personnel of an online shopping mall used open-source software to collect metrics. However, these metrics are scattered and cannot be displayed centrally.

Solution

AOM implements one-stop, multi-dimensional O&M for cloud applications. In the access center, connect metrics of businesses, applications, middleware, and infrastructure. You can also customize dashboards for monitoring and set alarm rules through a unified entry to implement routine inspection and ensure normal service running.

AOM monitors metrics from multiple dimensions in different scenarios. It has a multi-layer (infrastructure, middleware, application, and business) metric system, displaying more than 1,000 types of metrics.

Table 2-1 Four-layer metric system

Category	Source	Example	How to Access
Business metrics	Device log SDKs and extracted ELB logs	UV, PV, latency, access failure rate, and access traffic	Connect Business Metrics
	Transaction monitoring or reported custom metrics	URL calls, maximum concurrency, and maximum response time	


Category	Source	Example	How to Access
Application metrics	Component performance graphs or API performance data	URL calls, average latency, error calls, and throughput	Connect Application Metrics
Middleware metrics	Native or cloud middleware data	File system capacity and file system usage	Connect Middleware Metrics
Infrastructure metrics	Container or cloud service data, such as compute, storage, network, and database data	CPU usage, memory usage, and health status	Connect Infrastructure Metrics <ul style="list-style-type: none"> • Connect Container Metrics • Connect Cloud Service Metrics

Prerequisites

- [ELB logs have been ingested to LTS.](#)
- An ECS has been bound to the environment.

Step 1: Build a Four-layer Metric System

Step 1 Connect business metrics.

1. Log in to the AOM 2.0 console.
2. In the navigation pane, choose **Access Center**.
3. In the **Business** panel on the right, click a target card.
 - Connecting ELB log metrics
 - i. The system can automatically connect the log metrics.
 - ii. Choose **Dashboard** in the navigation pane, select the created dashboard, and click  in the upper right corner of the page. On the **Log Sources** tab, enter the corresponding SQL statement to check the log metrics. For example, to check traffic metrics, enter an SQL statement and click **Search**.
 - Connecting APM transaction metrics
 - i. Install an APM probe for the workload. For details, see [Installing an APM Probe](#).
 - ii. After the installation is complete, log in to the console of the service where the probe is installed and trigger the collection of APM transaction metrics. In the example of an online shopping mall, you can add a product to the shopping cart to trigger the collection.

- iii. Log in to the AOM 2.0 console.
- iv. In the navigation pane, choose **Metric Browsing**. In the right pane, select the connected APM metrics to view.

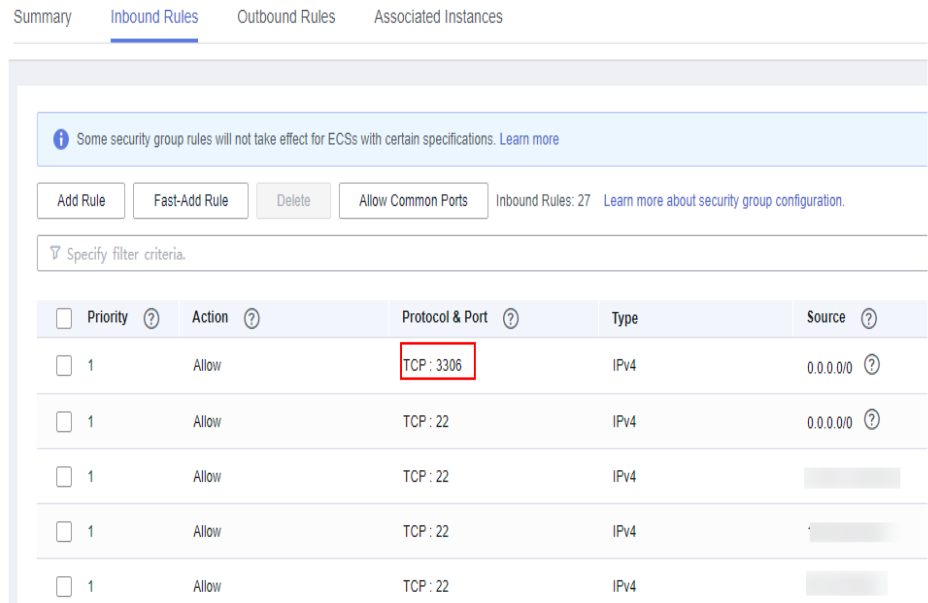
Step 2 Connect application metrics.

1. To install an APM probe for a workload, perform the following steps:
 - a. Log in to the CCE console and click a target cluster.
 - b. Choose **Workloads** in the navigation pane, and select the type of workload whose metrics are to be reported to AOM.
 - c. Click a target workload. On the **APM Settings** tab page, click **Edit** in the lower right corner.
 - d. Select the **APM 2.0** probe, set **Probe Version** to **latest-x86**, set **APM Environment** to **phoenixenv1**, and select the created application **phoenixapp1** from the **APM App** drop-down list.
 - e. Click **Save**.
2. After the installation is complete, log in to the console of the service where the probe is installed and trigger the collection of application metrics. In the example of an online shopping mall, you can add a product to the shopping cart to trigger the collection.
3. Log in to the AOM 2.0 console.
4. In the navigation pane, choose **Metric Browsing**. In the right pane, select the connected application metrics to view.

Step 3 Connect middleware metrics.

1. Upload the data to the ECS.
 - a. Download the **mysqld_exporter-0.14.0.linux-amd64.tar.gz** package from <https://prometheus.io/download/>.
 - b. Log in to the ECS as the **root** user, upload the Exporter software package to the ECS, and decompress it.
 - c. Log in to the RDS console. On the **Instances** page, click an RDS DB instance name in the instance list. On the basic information page, view the RDS security group.
 - d. Check whether port 3306 is enabled in the RDS security group.

Figure 2-1 Checking whether the RDS port is enabled



- e. Go to the decompressed folder and configure the **mysql.cnf** file on the ECS:

```
cd mysql_exporter-0.14.0.linux-amd64
vi mysql.cnf
```

For example, add the following content to the **mysql.cnf** file:

```
[client]
user=root (RDS username)
password=**** (RDS password)
host=192.168.0.198 (RDS public IP address)
port=3306 (port)
```

- f. Run the following command to start the **mysql_exporter** tool:

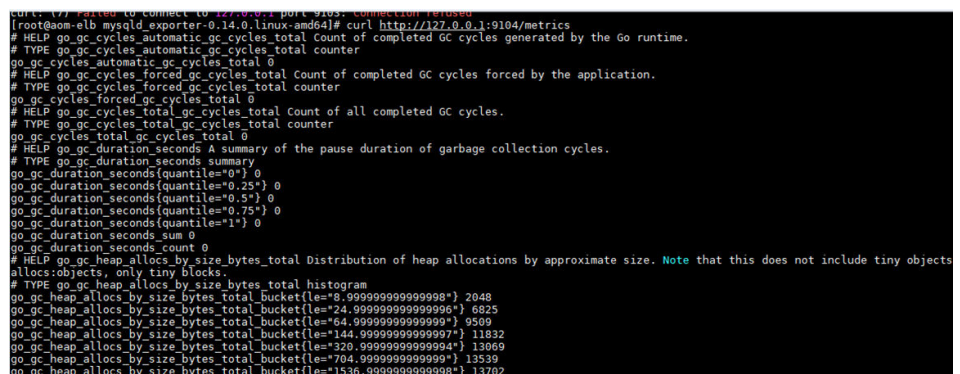
```
nohup ./mysql_exporter --config.my-cnf="mysql.cnf" --collect.global_status --
collect.global_variables &
```

- g. Run the following command to check whether the tool is started properly:

```
curl http://127.0.0.1:9104/metrics
```

If the command output shown in **Figure 2-2** is displayed, the tool is started properly.

Figure 2-2 Checking metrics



2. Connect middleware metrics using VM access mode.
 - a. Log in to the AOM 2.0 console.
 - b. On the **VM Access** page, install the UniAgent for the ECS. For details, see [Manual Installation](#).
 - c. In the navigation pane, choose **Access Center**. In the **Prometheus Middleware** panel on the right, click a target card.
 - d. In the dialog box that is displayed, configure a collection task and install Exporter. For details, see [Exporter Access in the VM Scenario](#).
 - e. Click **Create**.
3. After the connection is complete, choose **Metric Browsing** in the navigation pane on the left. In the right pane, view the connected middleware metrics.

Step 4 Connect infrastructure metrics.

1. Log in to the AOM 2.0 console.
2. In the navigation pane, choose **Access Center**.
3. In the **Prometheus Running Environments** or **Prometheus Cloud Services** panel, click a target card.
 - Select a container metric card:
For example, if you select the CCE card, the ICAgent is installed by default after you purchase a CCE cluster.
 - Select a cloud service metric card:
 - i. In the displayed dialog box, select the desired cloud service to monitor. For example, RDS or DCS.
 - ii. Click **Confirm**.
After the connection is complete, the [Cloud Service Monitoring](#) page is displayed. You can view the information (such as running status) of the selected cloud service.
4. After the connection is complete, choose **Metric Browsing** in the navigation pane on the left. In the right pane, select the connected infrastructure metrics to view.

----End

Step 2: Add a Dashboard for Unified Monitoring

Step 1 Create a metric alarm rule.

You can set threshold conditions in metric alarm rules for resource metrics. If a metric value meets the threshold condition, a threshold alarm will be generated. If no metric data is reported, an insufficient data event will be generated.

Metric alarm rules can be created in the following modes: **Select from all metrics** and **PromQL**. The following uses **Select from all metrics** as an example.

1. Log in to the AOM 2.0 console.
2. In the navigation pane, choose **Alarm Management > Alarm Rules**.
3. On the **Metric/Event Alarm Rules** tab page, click **Create**.
4. Set the basic information about the alarm rule, such as the rule name.

5. Set parameters about the alarm rule. Set **Rule Type** to **Metric alarm rule** and **Configuration Mode** to **Select from all metrics**, and select a Prometheus instance from the drop-down list.
6. Set alarm rule details.
You need to set information such as the statistical period, condition, detection rule, trigger condition, and alarm severity. The detection rule consists of the statistical mode (**Avg**, **Min**, **Max**, **Sum**, and **Samples**), determination criterion (\geq , \leq , $>$, and $<$), and threshold value. For example, if **Statistical Period** is **1 minute**, **Rule** is **Avg >1**, **Consecutive Periods** is **3**, and **Alarm Severity** is **Critical**, a critical alarm will be generated when the average metric value is greater than **1** for three consecutive periods.
7. Click **Advanced Settings** and set information such as **Check Interval** and **Alarm Clearance**.
8. Set an alarm notification policy. There are two alarm notification modes. As shown in [Figure 2-3](#), the direct alarm reporting mode is selected.
Direct alarm reporting: An alarm is directly sent when the alarm condition is met. If you select this mode, set an interval for notification and specify whether to enable an action rule.
 - a. Set the frequency for sending alarm notifications.
 - b. Specify whether to enable an alarm action rule. After an alarm action rule is enabled, the system sends notifications based on the associated SMN topic and message template.

Figure 2-3 Alarm notification

Alarm Notification

Notify When

- Alarm triggered Alarm cleared

Alarm Mode

Direct alarm reporting Alarm noise reduction

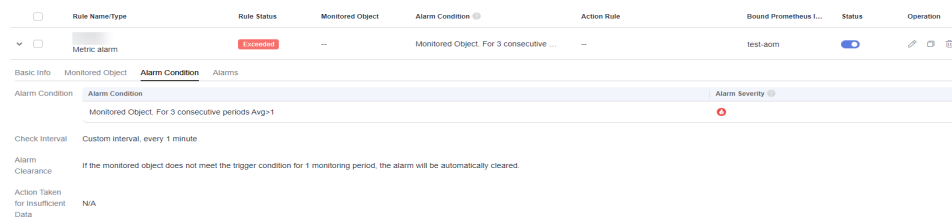
Frequency

Once

Action Rule

9. Click **Confirm**. Then, click **Back to Alarm Rule List** to view the created alarm rule.
As shown in [Figure 2-4](#), click \surd next to a rule name to view details.
In the expanded list, if a monitored object meets the configured alarm condition, a metric alarm is generated on the alarm page. To view the alarm, choose **Alarm Management > Alarm List** in the navigation pane. If a host meets the preset notification policy, the system sends an alarm notification to the specified personnel by email, SMS, or WeCom.

Figure 2-4 Alarm rule



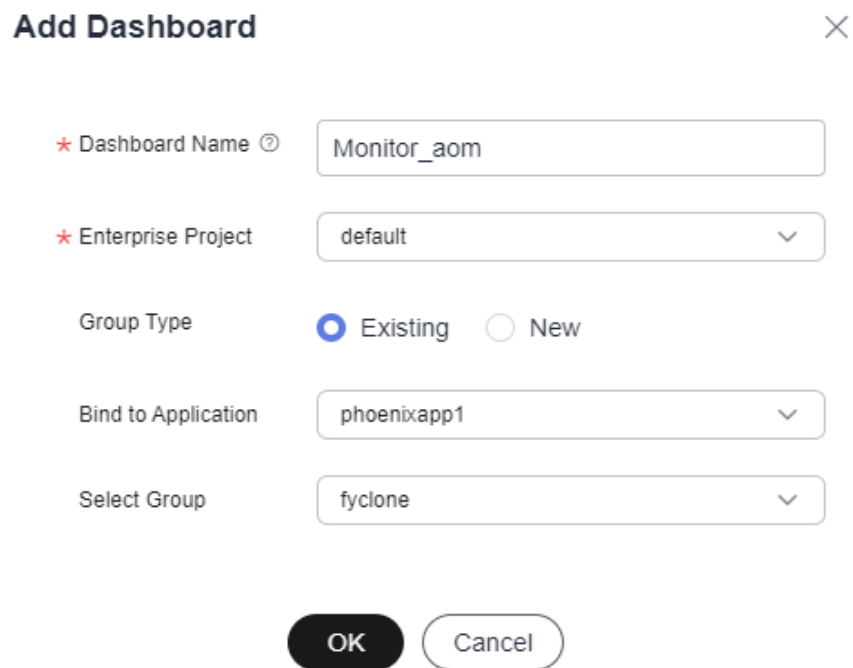
Step 2 Create a dashboard.

1. Create a dashboard.

- a. Log in to the AOM 2.0 console.
- b. In the navigation pane, choose **Dashboard**.
- c. Click **Add Dashboard** in the upper left corner of the list.
- d. In the displayed dialog box, set parameters.

Bind the dashboard to the created application so that you can monitor key metrics of the application on the **Application Monitoring** page.

Figure 2-5 Creating a dashboard




- e. Click **OK**.
2. Add a graph to the dashboard.
- a. In the dashboard list, click the created dashboard.
 - b. Go to the target dashboard page and click  in the upper right corner to add a graph to the dashboard. Select a proper graph as required.

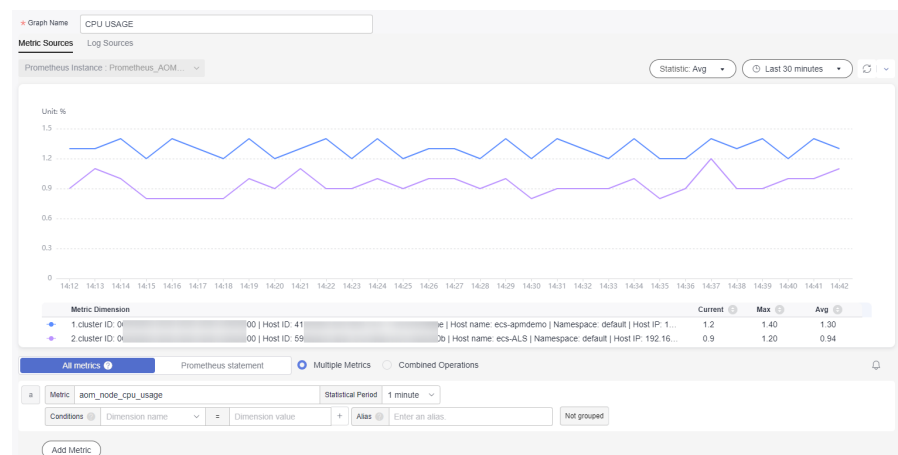
Table 2-2 Adding a graph

Graph Type	Data Source	Scenario
Metric graph	Metric data	Monitors the metrics about the business layer, application layer, and Prometheus middleware.
Log graph	Log data	Monitors business metrics or other log metrics, such as key metrics (latency, throughput, and errors) cleaned based on ELB logs.


The following describes how to add a metric graph for **CPU usage** and a log graph for **latency**.

- Add a metric graph for CPU usage.
 Select the **CPU Usage** metric. After the setting is complete, the metric graph shown in **Figure 2-6** is displayed.

Figure 2-6 Adding a metric graph



- Add a log graph for latency. Click the **Log Sources** tab and set parameters to add a log graph.
 You can directly obtain the SQL query statement from the graph.
 - In the upper right corner of the graph display area, click **Show Chart**.
 - In the **Charts** list, select required log metrics to monitor.
 - The query statement corresponding to the metric is automatically filled in the SQL statement setting area.
 After setting the parameters, click **Add to Dashboard**.

- c. You can repeat the preceding operations to add more graphs to the dashboard. Then click  to save the dashboard.

----End

3 Alarm Noise Reduction

This section describes how to set alarm noise reduction. Before sending an alarm notification, AOM processes alarms based on noise reduction rules to prevent alarm storms.

Scenario

When analyzing applications, resources, and businesses, e-commerce O&M personnel find that the number of alarms is too large and there are too many identical alarms. They need to detect faults based on the alarms and monitor applications comprehensively.

Solution

Use AOM to set alarm rules to monitor the usage of resources (such as hosts and components) in the environment in real time. When AOM or an external service is abnormal, an alarm is triggered immediately. AOM also provides the alarm noise reduction function. Before sending an alarm notification, AOM processes alarms based on noise reduction rules. This helps you identify critical problems and avoid alarm storms.

Alarm noise reduction consists of four parts: grouping, deduplication, suppression, and silence.

- You can filter different subnets of alarms and then group them according to certain conditions. Alarms in the same group are aggregated to trigger one notification.
- By using suppression rules, you can suppress or block notifications related to specific alarms. For example, when a major alarm is generated, less severe alarms can be suppressed. Another example, when a node is faulty, all other alarms of the processes or containers on this node can be suppressed.
- You can create a silence rule to shield alarm notifications in a specified period. The rule takes effect immediately after it is created.
- AOM has built-in deduplication rules. The service backend automatically deduplicates alarms. You do not need to manually create rules.

Monitoring ELB metrics at the business layer is used as an example here.

Prerequisite

An alarm action rule has been created.

Step 1: Create a Grouping Rule

When a critical or major alarm is generated, the **Monitor_host** action rule is triggered, and alarms are grouped by alarm source. To create a grouping rule, do as follows:

- Step 1** Log in to the AOM 2.0 console.
- Step 2** In the navigation pane, choose **Alarm Management > Alarm Noise Reduction**.
- Step 3** On the **Grouping Rules** tab page, click **Create** and set the rule name and grouping condition.

Figure 3-1 Creating a grouping rule

* Rule Name

* Enterprise Project

Description

Grouping Rule

Grouping Condition

Alarm Severity	event_severity	Equals To	Critic... x Ma... x	<input type="button" value="x"/>
Alarm Source	resource_provider	Equals To	A... x	<input type="button" value="x"/>

Action Rule

Combination Rule

* Combine Notifications

* Initial Wait Time Range: 0s to 10 mins.

* Batch Processing Interval Range: 5s to 30 mins.

* Repeat Interval Range: 1 min to 15 days.

Note: If Repeat Interval is set to 0, identical notifications will not be sent again.

Table 3-1 Alarm combination rule

Combine Notifications	<p>Combines grouped alarms based on specified fields. Alarms in the same group are aggregated for sending one notification.</p> <p>Notifications can be combined:</p> <ul style="list-style-type: none"> • By alarm source: Alarms triggered by the same alarm source are combined into one group for sending notifications. • By alarm source + severity: Alarms triggered by the same alarm source and of the same severity are combined into one group for sending notifications. • By alarm source + all tags: Alarms triggered by the same alarm source and with the same tag are combined into one group for sending notifications.
Initial Wait Time	<p>Interval for sending an alarm notification after alarms are combined for the first time. It is recommended that the time be set to seconds to prevent alarm storms.</p> <p>Value range: 0s to 10 minutes. Recommended: 15s.</p>
Batch Processing Interval	<p>Waiting time for sending an alarm notification after the combined alarm data changes. It is recommended that the time be set to minutes. If you want to receive alarm notifications as soon as possible, set the time to seconds.</p> <p>The change here refers to a new alarm or an alarm status change.</p> <p>Value range: 5s to 30 minutes. Recommended: 60s.</p>
Repeat Interval	<p>Waiting time for sending an alarm notification after the combined alarm data becomes duplicate. It is recommended that the time be set to hours.</p> <p>Duplication means that no new alarm is generated and no alarm status is changed while other attributes (such as titles and content) are changed.</p> <p>Value range: 0 minutes to 15 days. Recommended: 1 hour.</p>

----End

Step 2: Create a Metric Alarm Rule (Configuration Mode Set to Select from all metrics)

You can set threshold conditions in metric alarm rules for resource metrics. If a metric value meets the threshold condition, a threshold alarm will be generated. If no metric data is reported, an insufficient data event will be generated.

Metric alarm rules can be created in the following modes: **Select from all metrics** and **PromQL**. The following describes how to create an alarm rule for monitoring all metrics at the ELB business layer.

Step 1 Log in to the AOM 2.0 console.

Step 2 In the navigation pane, choose **Alarm Management > Alarm Rules**.

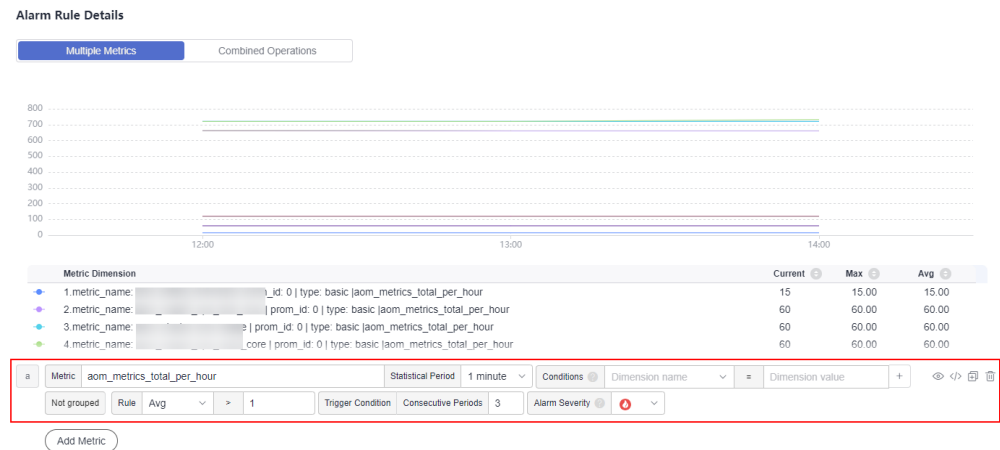
Step 3 On the **Metric/Event Alarm Rules** tab page, click **Create**.

Step 4 Set the basic information about the alarm rule, such as the rule name.

Step 5 Set the detailed information about the alarm rule.

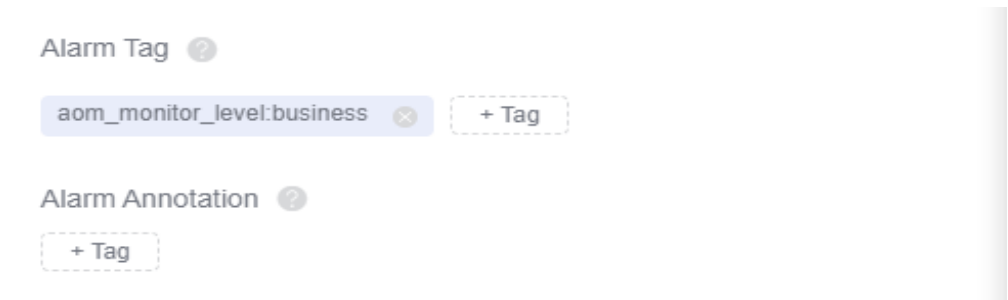
1. Set **Rule Type** to **Metric alarm rule** and **Configuration Mode** to **Select from all metrics**.
2. Set parameters such as the metric, environment, and check interval.

Figure 3-2 Setting the detailed information about the alarm rule



3. Set alarm tags and annotations to group alarms. They can be associated with alarm noise reduction policies for sending notifications. As a business-layer metric is selected in [Step 5.2](#), set **Alarm Tag** to **aom_monitor_level:business**.

Figure 3-3 Customizing tag information



NOTE

The tag of full metrics is in the format of "key:value". Generally, **key** is set to **aom_monitor_level**. **value** varies depending on the layer of metrics:

- Infrastructure metrics: **infrastructure**
- Middleware metrics: **middleware**
- Application metrics: **application**
- Business metrics: **business**

Step 6 Set an alarm notification policy. There are two alarm notification modes. In this example, the alarm noise reduction mode is selected.

Alarm noise reduction: Alarms are sent only after being processed based on noise reduction rules, preventing alarm storms.

Figure 3-4 Selecting the alarm noise reduction mode

Alarm Notification

Notify When

Alarm triggered Alarm cleared

Alarm Mode

Direct alarm reporting **Alarm noise reduction**

Grouping Rule

aom1 ↕ ↻ 📄

Step 7 Click **Confirm**. Then, click **Back to Alarm Rule List** to view the created alarm rule.

As shown in the following figure, a metric alarm rule is created. Click **↕** in front of the rule name to view its details.

Figure 3-5 Creating a metric alarm rule

Rule Name/Type	Rule Status	Monitored Object	Alarm Condition	Action Rule	Bound Prometheus L...	Status	Operation
Metric alarm	Enabled	--	Monitored Object. For 3 consecutive ...	--	Prometheus_AO...	<input checked="" type="checkbox"/>	↕ 📄 🗑️

Basic Info	Monitored Object	Alarm Condition	Alarms
Alarm Condition	Alarm Condition	Alarm Severity	
Monitored Object	Monitored Object. For 3 consecutive periods Avg>1		
Check Interval	Custom interval, every 1 minute		
Alarm Clearance	If the monitored object does not meet the trigger condition for 1 monitoring period, the alarm will be automatically cleared.		
Action Taken for Insufficient Data	N/A		

In the expanded list, if a metric value meets the configured alarm condition, a metric alarm is generated on the alarm page. To view the alarm, choose **Alarm Management > Alarm List** in the navigation pane.

If the preset notification policy is met, the system sends an alarm notification to the specified personnel by email, SMS, or WeCom.

----End

4 Unified Metric Monitoring

This section describes how to centrally monitor metric data of different accounts.

Scenario

O&M personnel of an e-commerce platform need to monitor metric data of different accounts in real time.

Solution

Create a Prometheus instance for multi-account aggregation and connect accounts, cloud services, and cloud service metrics. On the **Metric Browsing** page, you can monitor metrics of multiple member accounts and set alarm rules for them. When a metric is abnormal, an alarm is triggered immediately and a notification is sent.

Prerequisites

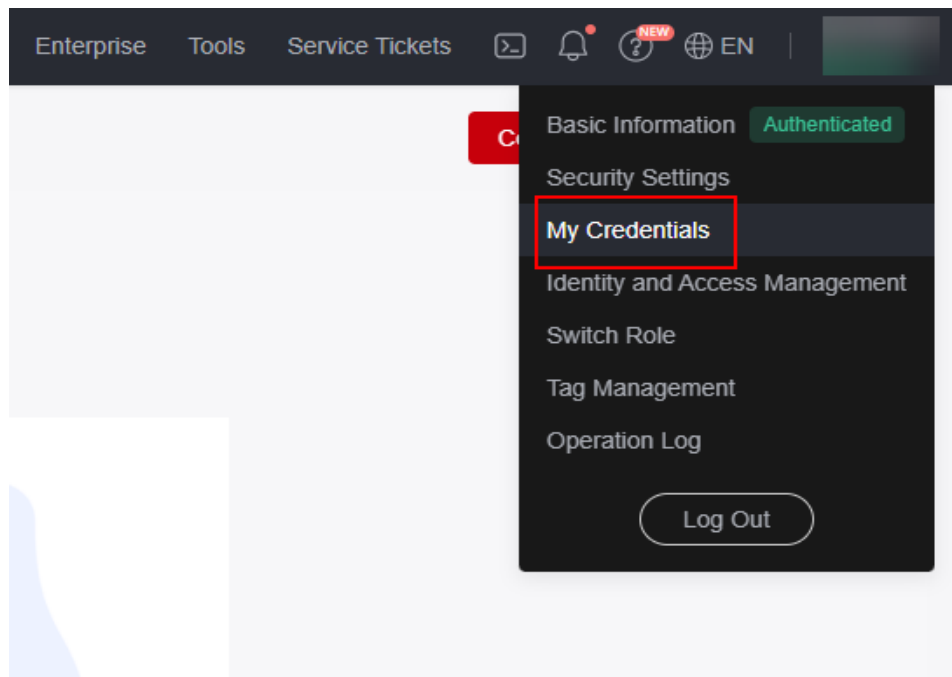
- The monitoring account and the monitored account have been added to an **organization**. The monitoring account must be an organization administrator. If not, perform **step 2** to set a delegated administrator.
- For the monitored account, metrics of the following cloud services can be aggregated: FunctionGraph, Elastic Volume Service (EVS), Cloud Backup and Recovery (CBR), Object Storage Service (OBS), Virtual Private Cloud (VPC), Elastic Load Balance (ELB), Direct Connect, NAT Gateway, Distributed Message Service (DMS), Distributed Cache Service (DCS), Relational Database Service (RDS), Document Database Service (DDS), Data Replication Service (DRS), LakeFormation, MapReduce Service (MRS), GaussDB(DWS), Cloud Search Service (CSS), and Web Application Firewall (WAF). Cloud Container Engine (CCE) and Elastic Cloud Server (ECS) metrics collected by ICAgents can also be aggregated.

Step 1: Connecting Cloud Services for a Monitored Account

The following uses **FunctionGraph** and **ECS** as examples. The procedure for connecting CCE is similar to that for connecting ECS. However, ICAgents are automatically installed by default when you purchase CCE clusters. The procedure for connecting FunctionGraph is similar to that for connecting other cloud services.

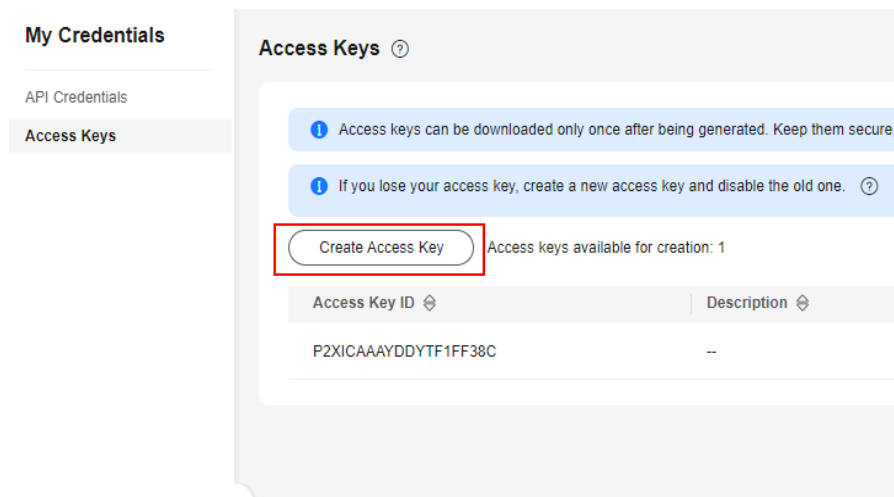
- Connecting FunctionGraph
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Access Center**.
 - c. Under **Cloud Services**, click **FunctionGraph**. In the displayed dialog box, click **Connect Now**.
- Connecting ECS
 - a. Hover over the username in the upper right corner and choose **My Credentials** from the drop-down list.

Figure 4-1 My credentials



- b. On the **My Credentials** page, click the **Access Keys** tab.
- c. Click **Create Access Key** and enter a verification code or password.

Figure 4-2 Adding an access key



- d. Click **OK** to download the generated AK/SK.
You can obtain the AK from the access key list and SK from the downloaded CSV file.
- e. Return to the AOM 2.0 console page. In the navigation pane, choose **Collection Management**.
- f. In the navigation pane, choose **UniAgent > VM Access**.
- g. On the **VM Access** page, select the hosts where ICAgents are to be installed and choose **Plug-in Batch Operation**.

Figure 4-3 Installing ICAgents

The screenshot shows a web interface for UniAgent Batch Operation. At the top, there are two dropdown menus: 'UniAgent Batch Operation' and 'Plug-in Batch Operation' (which is highlighted with a red box). To the right is a search bar with the text 'Search by host name by default.' Below this is a table with three columns: 'Host Name/IP Address', 'Access Mode', and 'UniAgent Status'. The table contains three rows of data:

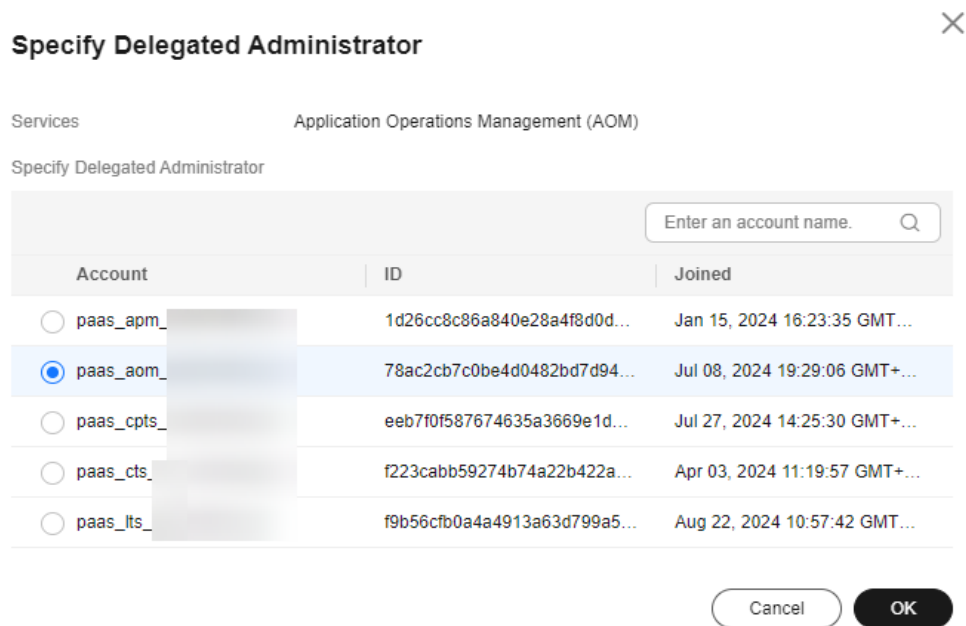
Host Name/IP Address	Access Mode	UniAgent Status
<input checked="" type="checkbox"/> ecs-apmdemo	Direct access	● Abnormal
<input type="checkbox"/> xg-uniagenttest-0002 Proxy	Direct access	● Abnormal
<input type="checkbox"/> xg-uniagenttest-0001	Direct access	● Abnormal

- h. In the displayed dialog box, set **Operation** to **Install**, **Plug-in** to **ICAgent**, and **Version** to **5.12.163**, and enter the AK/SK obtained in **d**.
- i. Click **OK** to install ICAgents.

Step 2: Enable Access for AOM and Set a Delegated Administrator (Skip this Step You Are an Organization Administrator)

- Step 1** Log in to the Organizations console as an administrator.
- Step 2** In the navigation pane, choose **Services**.
- Step 3** In the service list, locate **Application Operations Management (AOM)** and click **Enable Access** in the **Operation** column.
- Step 4** Click **Specify Delegated Administrator** in the **Operation** column of **AOM**, select the desired account, and click **OK**. As shown in **Figure 4-4**, **paas_aom** is specified as the delegated administrator.

Figure 4-4 Specifying a delegated administrator



----End

Step 3: Create an Instance for Multi-Account Aggregation

- Step 1** Log in to the AOM 2.0 console as an administrator or delegated administrator.
- Step 2** In the navigation pane, choose **Prometheus Monitoring > Instances**. On the displayed page, click **Add Prometheus Instance**.
- Step 3** Enter an instance name and select the **Prometheus for Multi-Account Aggregation** instance type.
- Step 4** Click **OK**. As shown in [Figure 4-5](#), a multi-account aggregation instance named **test-aom** is created.

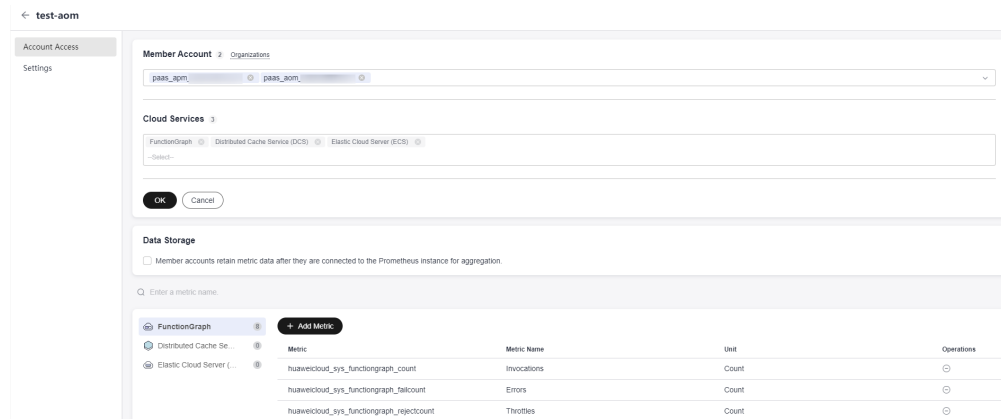
Figure 4-5 Prometheus instance list

Prometheus Instance	Instance Type	Enterprise Project
test-aom	Prometheus for Multi-Account Aggregation	default
	Prometheus for ECS	default
	Prometheus for Cloud Services	
	Prometheus for Remote Write	
	Prometheus for Cloud Services	
	Prometheus for Cloud Services	default

- Step 5** In the Prometheus instance list, click the name of the created instance. On the displayed page, select the accounts, cloud services, and cloud service metrics to connect.

For example, connect member accounts **paas_apm** and **paas_aom**. Connect cloud services such as FunctionGraph, DCS, and ECS. Click **Add Metric**. In the displayed dialog box, select desired metrics.

Figure 4-6 Connecting accounts



Wait for 2 to 3 minutes and view the connected metric data on the **Metric Browsing** page.

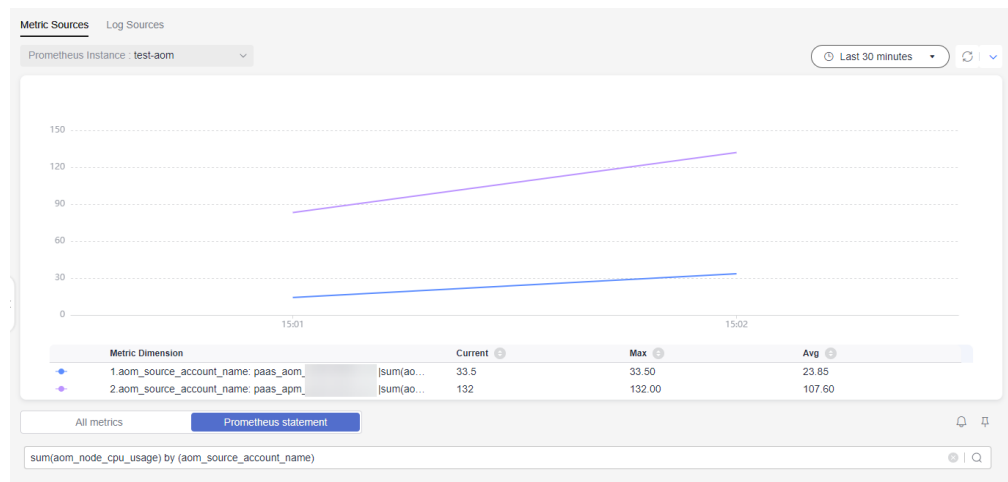
----End

Step 4: Configuring Unified Monitoring

Step 1 Check whether the metrics of the created instance are connected.

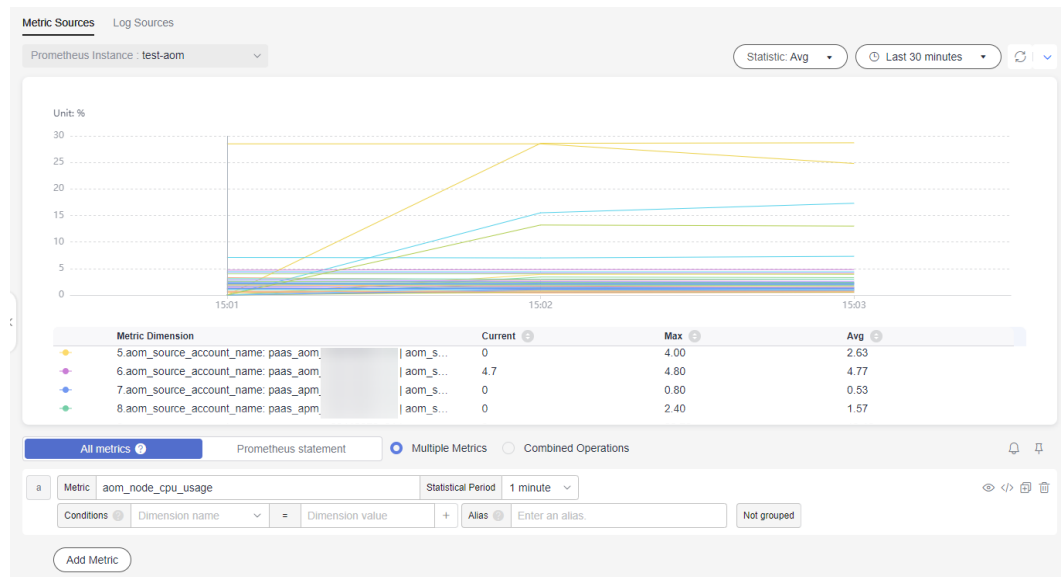
1. In the navigation pane, choose **Metric Browsing**. In the **Prometheus Instance** drop-down list, select instance **test-aom** created in [step 3](#).
2. Click **All metrics**, select a metric, and copy the metric name.
3. Click **Prometheus statement** and enter **sum(metric name) by (aom_source_account_name)** to check whether the metric is connected.


Figure 4-7 Checking metrics



Step 2 Click **All metrics** and select the metric to be monitored. As shown in [Figure 4-8](#), select the **aom_node_cpu_usage** metric so that its values and trends under the **paas_apm** and **paas_aom** accounts can be monitored in real time.

Figure 4-8 Checking metrics



Step 3 Click  in the upper right corner of the metric list to add an alarm rule for the selected metric.

1. Set the basic information about the alarm rule, such as the rule name.
2. Set the detailed information about the alarm rule.
 - a. By default, the rule type, configuration mode, and Prometheus instance in the alarm rule settings are the same as those on the **Metric Browsing** page.
 - b. Set alarm rule details. By default, the metric selected on the **Metric Browsing** page is automatically displayed.

You need to set information such as the statistical period, condition, detection rule, trigger condition, and alarm severity. The detection rule consists of the statistical mode (**Avg**, **Min**, **Max**, **Sum**, and **Samples**), determination criterion (\geq , \leq , $>$, and $<$), and threshold value. For example, if **Statistical Period** is **1 minute**, **Rule** is **Avg > 1**, **Consecutive Periods** is **3**, and **Alarm Severity** is **Critical**, a critical alarm will be generated when the average metric value is greater than **1** for three consecutive periods.

Figure 4-9 Setting an alarm rule



- c. Click **Advanced Settings** and set information such as **Check Interval** and **Alarm Clearance**.
- d. Set an alarm notification policy. There are two alarm notification modes. As shown in **Figure 4-10**, the direct alarm reporting mode is selected.
Direct alarm reporting: An alarm is directly sent when the alarm condition is met. If you select this mode, set an interval for notification and specify whether to enable an action rule.
 - i. Set the frequency for sending alarm notifications.
 - ii. Specify whether to enable an alarm action rule. After an alarm action rule is enabled, the system sends notifications based on the associated SMN topic and message template.

Figure 4-10 Alarm notification

Alarm Notification

Notify When

Alarm triggered Alarm cleared

Alarm Mode

Direct alarm reporting Alarm noise reduction

Frequency

Once

Action Rule

- e. Click **Confirm**. Then, click **Back to Alarm Rule List** to view the created alarm rule.
As shown in **Figure 4-11**, click next to a rule name to view details. In the expanded list, if a monitored object meets the configured alarm condition, a metric alarm is generated on the alarm page. To view the alarm, choose **Alarm Management > Alarm List** in the navigation pane. If a host meets the preset notification policy, the system sends an alarm notification to the specified personnel by email, SMS, or WeCom.

Figure 4-11 Alarm rule

Rule Name/Type	Rule Status	Monitored Object	Alarm Condition	Action Rule	Bound Prometheus L...	Status	Operation
Metric alarm	Excellent	--	Monitored Object. For 3 consecutive ...	--	test-aom	<input checked="" type="checkbox"/>	

Basic Info Monitored Object Alarm Condition Alarms

Alarm Condition	Alarm Condition	Alarm Severity
Monitored Object. For 3 consecutive periods Avg>1		

Check Interval Custom interval, every 1 minute

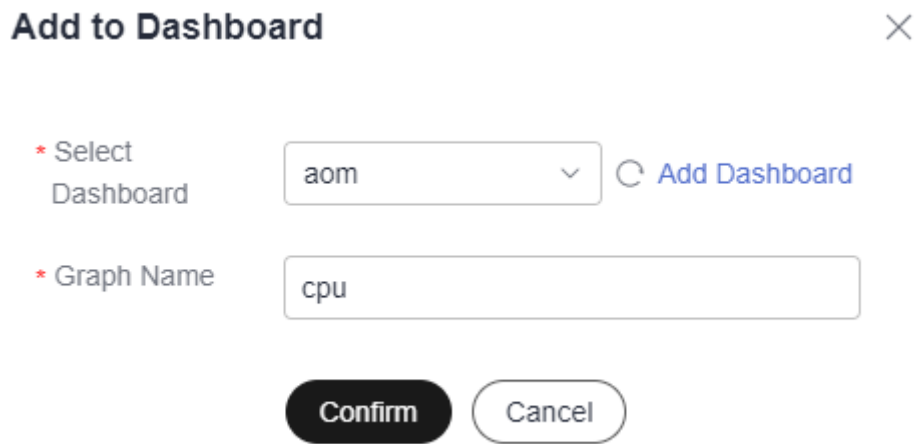
Alarm Clearance If the monitored object does not meet the trigger condition for 1 monitoring period, the alarm will be automatically cleared.

Action Taken for Insufficient Data N/A

Step 4 Click in the upper right corner of the metric list to add the graph to the dashboard.

1. Select a dashboard from the drop-down list and enter the graph name. If the dashboards in the list cannot meet your requirements, click **Add Dashboard** to add one. For details, see [Creating a Dashboard](#).

Figure 4-12 Adding the graph to a dashboard



Add to Dashboard ✕

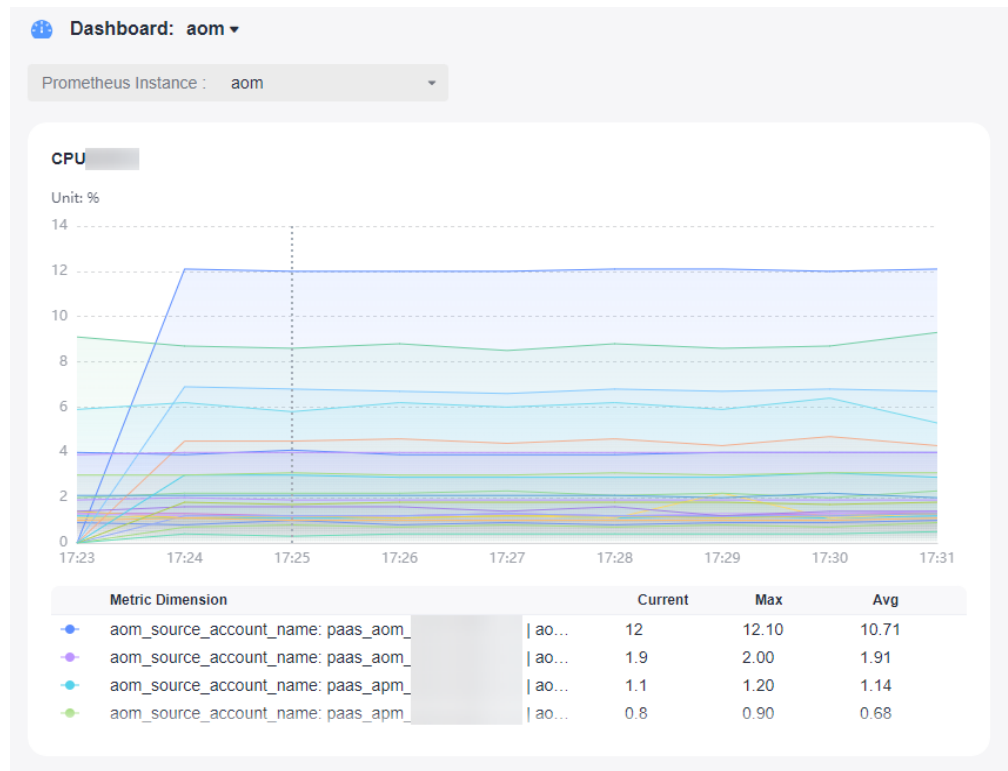
* Select Dashboard ↻ Add Dashboard

* Graph Name

Confirm Cancel

2. Click **Confirm**. The dashboard page is displayed. As shown in [Figure 4-13](#), the **CPU Usage** graph is added to the **aom** dashboard so that its values and trends under the **paas_apm** and **paas_aom** accounts can be monitored in real time.

Figure 4-13 Viewing the graph



----End

5 Customizing OS Images to Automatically Connect UniAgent

This section describes how to package images for connecting UniAgent in the Linux and Windows environments.

Overview

An image is an Elastic Compute Server (ECS) or Bare Metal Server (BMS) template that contains OS or service data and may also contain proprietary software and application software, such as database software. Images are classified into public, private, Marketplace, and shared images.

Image Management Service (IMS) provides easy-to-use, self-service image management functions. You can use a public, private, or shared image to apply for ECSs. You can also create private images from existing ECSs or using external image files.

Packaging an Image in the Linux Environment

In the Linux environment, you can package an image according to the following procedure:

Prerequisites

Ensure that no UniAgent has been installed on the Linux host where the image is to be packaged.

Procedure

Step 1 Create an ECS by referring to [ECS Getting Started](#).

Step 2 For example, in the **CN North-Beijing4** region, download the **install_uniagentd_self_OS.sh** script to the **/root** directory of the ECS:

```
wget https://aom-uniagent-cn-north-4.obs.cn-north-4.myhuaweicloud.com/install_uniagentd_self_OS.sh
      {region_id}=cn-north-4
      {obs_domain}=obs.cn-north-4.myhuaweicloud.com
```

NOTE

Download command: **wget https://aom-uniagent-*{region_id}*.*{obs_domain}*/install_uniagentd_self_OS.sh**

Step 3 In the `/etc/init.d/` directory, set the `install_uniagentd_self_OS.sh` script to automatically start upon power-on:

```
bash /root/install_uniagentd_self_OS.sh config
```

If the **AOMInstall** startup script exists in the `/etc/init.d/` directory, your setting is successful.

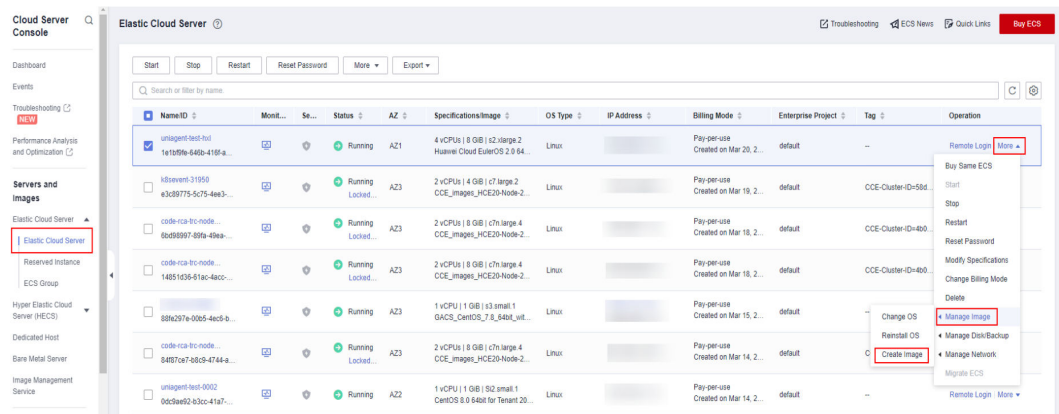
Step 4 Delete the configuration script:

```
rm -f /root/install_uniagentd_self_OS.sh
```

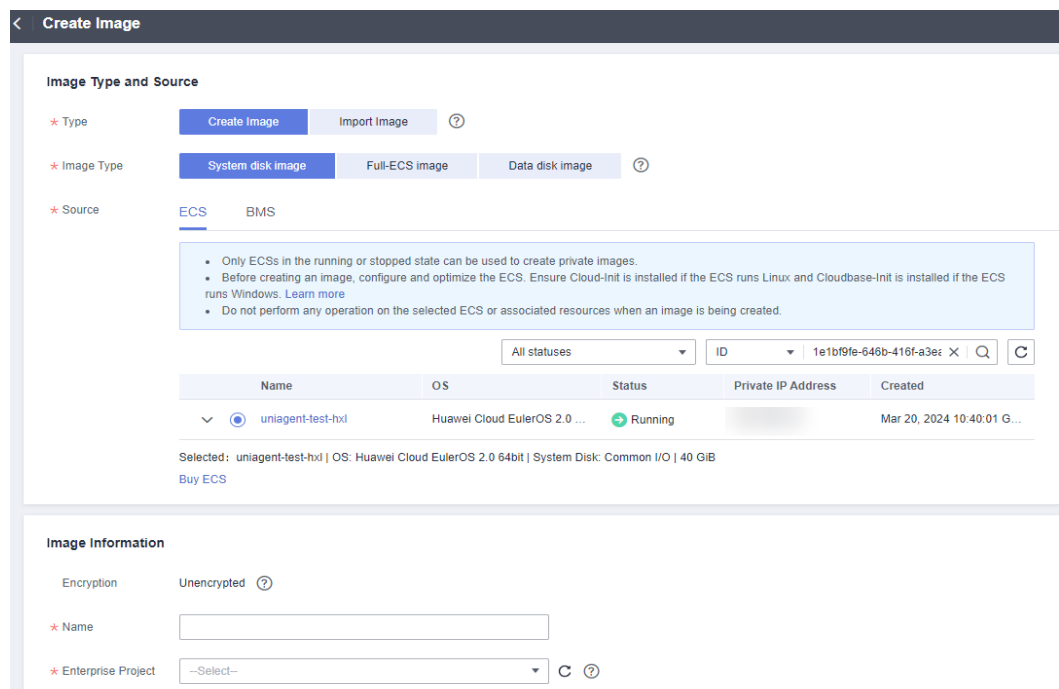
NOTE

After the preceding steps are complete, you can create an image. Do not restart the Linux host before you create an image.

Step 5 Locate the target ECS and click **Create Image** in the **Operation** column to create a private image. For details, see [Creating an Image](#).



Step 6 Configure image information as required.



----End

Packaging an Image in the Windows Environment

In the Windows environment, you can only install the UniAgent, delete some files, and then package your private image.

Step 1 Create an ECS by referring to [ECS Getting Started](#).

Step 2 On the ECS, manually install the UniAgent by referring to [Installing a UniAgent](#). Then check the UniAgent status on the UI.

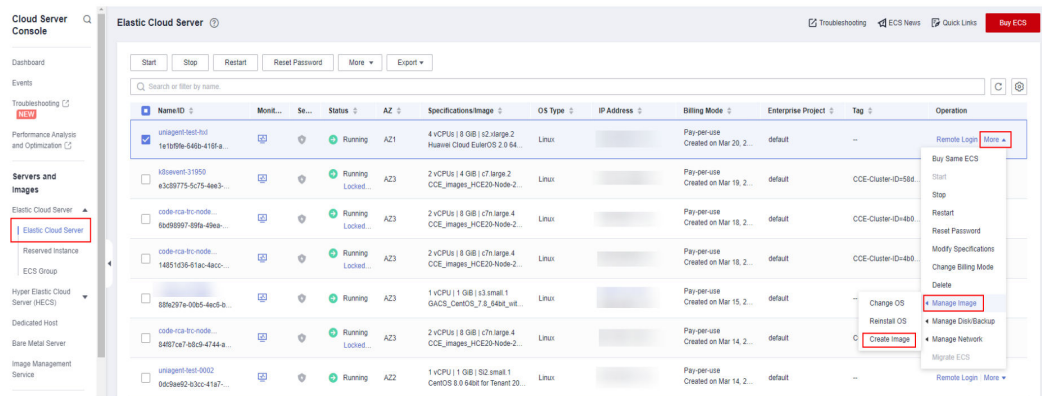
Step 3 Run the following command on the ECS after the UniAgent is installed:

```
sc stop uniagentdservice
&& del /s/q C:\uniagentd\uniagentd.sn && rd /s/q C:\uniagentd\tmp C:\uniagentd\log C:\uniagentd\libexec
&& echo -e "${ak_info}\n${sk_info}\n${master_info}" > C:\uniagentd\conf\uniagentd.conf
```

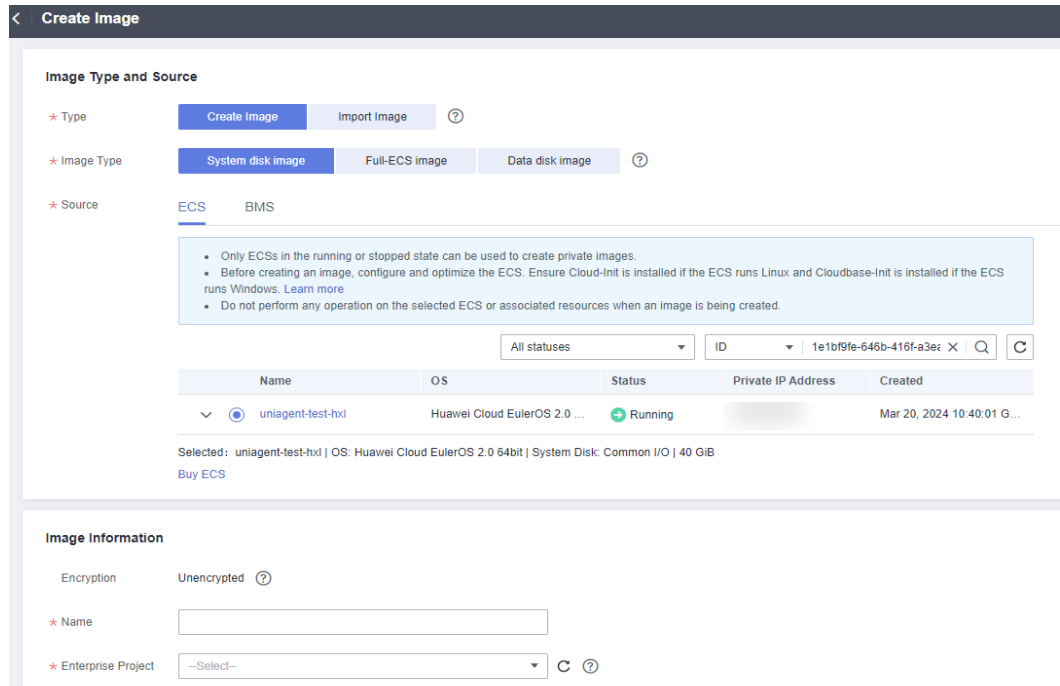
NOTE

Obtain the values of *ak_info*, *sk_info*, and *master_info* from the manual installation page and replace them based on site requirements. Each AK/SK pair corresponds to a specific project.

Step 4 Locate the target ECS and click **Create Image** in the **Operation** column to create a private image. For details, see [Creating an Image](#).



Step 5 Configure image information as required.



----End

6 Connecting Self-Built Middleware in the CCE Container Scenario

6.1 Connecting PostgreSQL Exporter

Application Scenario

When using PostgreSQL, you need to monitor their status and locate their faults in a timely manner. The Prometheus monitoring function monitors PostgreSQL running using Exporter in the CCE container scenario. This section describes how to deploy PostgreSQL Exporter and implement alarm access.

Prerequisites

- A CCE cluster has been created and PostgreSQL has been installed.
- Your service has been connected for Prometheus monitoring and a CCE cluster has also been connected. For details, see [Prometheus Instance for CCE](#).
- You have uploaded the [postgres_exporter](#) image to SoftWare Repository for Container (SWR). For details, see [Uploading an Image Through a Container Engine Client](#).

Deploying PostgreSQL Exporter

Step 1 Log in to the CCE console.

Step 2 Click the connected cluster. The cluster management page is displayed.

Step 3 Perform the following operations to deploy Exporter:

1. Use **Secret** to manage PostgreSQL passwords.

In the navigation pane, choose **Workloads**. In the upper right corner, click **Create from YAML** to configure a YAML file. In the YAML file, use Kubernetes **Secret** to manage and encrypt passwords. When starting PostgreSQL Exporter, the secret key can be directly used but the corresponding password needs to be changed as required.

YAML configuration example:

```
apiVersion: v1
kind: Secret
```

```
metadata:
  name: postgres-test
type: Opaque
stringData:
  username: postgres
  password: you-guess # PostgreSQL password.
```

2. Deploy PostgreSQL Exporter.

In the navigation pane, choose **Workloads**. In the upper right corner, click **Create from YAML** to deploy Exporter.

YAML configuration example (Change the parameters if needed):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres-test # Change the name based on requirements. You are advised to add the
  PostgreSQL instance information.
  namespace: default # Must be the same as the namespace of the PostgreSQL service.
  labels:
    app: postgres
    app.kubernetes.io/name: postgresql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
      app.kubernetes.io/name: postgresql
  template:
    metadata:
      labels:
        app: postgres
        app.kubernetes.io/name: postgresql
    spec:
      containers:
        - name: postgres-exporter
          image: swr.cn-north-4.myhuaweicloud.com/aom-exporter/postgres-exporter:v0.8.0 # postgres-
          exporter image uploaded to SWR.
          args:
            - "--web.listen-address=:9187" # Enabled port of Exporter.
            - "--log.level=debug" # Log level.
          env:
            - name: DATA_SOURCE_USER
              valueFrom:
                secretKeyRef:
                  name: postgres-test # Secret name specified in the previous step.
                  key: username # Secret key specified in the previous step.
            - name: DATA_SOURCE_PASS
              valueFrom:
                secretKeyRef:
                  name: postgres-test # Secret name specified in the previous step.
                  key: password # Secret key specified in the previous step.
            - name: DATA_SOURCE_URI
              value: "x.x.x.x:5432/postgres?sslmode=disable" # Connection information.
          ports:
            - name: http-metrics
              containerPort: 9187
```

3. Obtain metrics.

The running time of the Postgres instance cannot be obtained by running the **curl http://exporter:9187/metrics** command. To obtain this metric, customize a **queries.yaml** file.

- Create a configuration that contains **queries.yaml**.
- Mount the configuration as a volume to a directory of Exporter.
- Use the configuration through **extend.query-path**. The following shows **Secret** and **Deployment**:

```
# The following shows the queries.yaml file that contains custom metrics:
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-test-configmap
  namespace: default
data:
  queries.yaml: |
    pg_postmaster:
      query: "SELECT pg_postmaster_start_time as start_time_seconds from
pg_postmaster_start_time()"
      master: true
      metrics:
        - start_time_seconds:
            usage: "GAUGE"
            description: "Time at which postmaster started"

# The following shows the mounted Secret and ConfigMap, and defines Exporter deployment
parameters (such as the image):
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres-test
  namespace: default
  labels:
    app: postgres
    app.kubernetes.io/name: postgresql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
      app.kubernetes.io/name: postgresql
  template:
    metadata:
      labels:
        app: postgres
        app.kubernetes.io/name: postgresql
    spec:
      containers:
        - name: postgres-exporter
          image: wrouesnel/postgres_exporter:latest
          args:
            - "--web.listen-address=:9187"
            - "--extend.query-path=/etc/config/queries.yaml"
            - "--log.level=debug"
          env:
            - name: DATA_SOURCE_USER
              valueFrom:
                secretKeyRef:
                  name: postgres-test-secret
                  key: username
            - name: DATA_SOURCE_PASS
              valueFrom:
                secretKeyRef:
                  name: postgres-test-secret
                  key: password
            - name: DATA_SOURCE_URI
              value: "x.x.x.x:5432/postgres?sslmode=disable"
          ports:
            - name: http-metrics
              containerPort: 9187
          volumeMounts:
            - name: config-volume
              mountPath: /etc/config
      volumes:
        - name: config-volume
```

```

configMap:
  name: postgres-test-configmap
---
apiVersion: v1
kind: Service
metadata:
  name: postgres
spec:
  type: NodePort
  selector:
    app: postgres
  app.kubernetes.io/name: postgresql
  ports:
    - protocol: TCP
      nodePort: 30433
      port: 9187
      targetPort: 9187

```

d. Access the following address:

http://{Public IP address of any node in the cluster}:30433/metrics. You can then use the custom **queries.yaml** file to query the Postgres instance startup time.

Figure 6-1 Accessing a cluster node

```

← → ↻ :30433/metrics
# TYPE go_memstats_stack_inuse_bytes gauge
go_memstats_stack_inuse_bytes 624288
# HELP go_memstats_stack_sys_bytes Number of bytes obtained from system for stack allocator.
# TYPE go_memstats_stack_sys_bytes gauge
go_memstats_stack_sys_bytes 824288
# HELP go_memstats_sys_bytes Number of bytes obtained from system.
# TYPE go_memstats_sys_bytes gauge
go_memstats_sys_bytes 7.04812e+07
# HELP go_threads Number of OS threads created.
# TYPE go_threads gauge
go_threads 6
# HELP pg_exporter_last_scrape_duration_seconds Duration of the last scrape of metrics from PostgreSQL.
# TYPE pg_exporter_last_scrape_duration_seconds gauge
pg_exporter_last_scrape_duration_seconds 0.016062949
# HELP pg_exporter_last_scrape_error Whether the last scrape of metrics from PostgreSQL resulted in an error (1 for error, 0 for success).
# TYPE pg_exporter_last_scrape_error gauge
pg_exporter_last_scrape_error 0
# HELP pg_exporter_scrapes_total Total number of times PostgreSQL was scraped for metrics.
# TYPE pg_exporter_scrapes_total counter
pg_exporter_scrapes_total 2
# HELP pg_locks_count Number of locks
# TYPE pg_locks_count gauge
pg_locks_count{datname="aa",mode="accessexclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="aa",mode="accessshare",server="192.168.0.205:30432"} 0
pg_locks_count{datname="aa",mode="exclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="aa",mode="rowexclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="aa",mode="rowshare",server="192.168.0.205:30432"} 0
pg_locks_count{datname="aa",mode="share",server="192.168.0.205:30432"} 0
pg_locks_count{datname="aa",mode="shareexclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="postgres",mode="accessexclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="postgres",mode="accessshare",server="192.168.0.205:30432"} 1
pg_locks_count{datname="postgres",mode="exclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="postgres",mode="rowexclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="postgres",mode="rowshare",server="192.168.0.205:30432"} 0
pg_locks_count{datname="postgres",mode="share",server="192.168.0.205:30432"} 0
pg_locks_count{datname="postgres",mode="shareexclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template0",mode="accessexclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template0",mode="accessshare",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template0",mode="exclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template0",mode="rowexclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template0",mode="rowshare",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template0",mode="share",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template0",mode="shareexclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template1",mode="accessexclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template1",mode="accessshare",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template1",mode="exclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template1",mode="rowexclusive",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template1",mode="rowshare",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template1",mode="share",server="192.168.0.205:30432"} 0
pg_locks_count{datname="template1",mode="shareexclusive",server="192.168.0.205:30432"} 0
# HELP pg_settings_allow_system_table_mods Allows modifications of the structure of system tables.
# TYPE pg_settings_allow_system_table_mods gauge
pg_settings_allow_system_table_mods{server="192.168.0.205:30432"} 0
# HELP pg_settings_archive_timeout_seconds Forces a switch to the next WAL file if a new file has not been started within N seconds. [Units converted to seconds.]
# TYPE pg_settings_archive_timeout_seconds gauge

```

----End

Adding a Collection Task

Add **PodMonitor** to configure a collection rule for monitoring the service data of applications deployed in the CCE cluster.

NOTE

In the following example, metrics are collected every 30s. Therefore, you can check the reported metrics on the AOM page about 30s later.

```
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: postgres-exporter
  namespace: default
spec:
  namespaceSelector:
    matchNames:
      - default # Namespace where Exporter is located.
  podMetricsEndpoints:
    - interval: 30s
      path: /metrics
      port: http-metrics
  selector:
    matchLabels:
      app: postgres
```

Verifying that Metrics Can Be Reported to AOM

- Step 1** Log in to the AOM 2.0 console.
- Step 2** In the navigation pane on the left, choose **Prometheus Monitoring > Instances**.
- Step 3** Click the Prometheus instance connected to the CCE cluster. The instance details page is displayed.
- Step 4** On the **Metrics** tab page of the **Metric Management** page, select your target cluster.
- Step 5** Select job `{namespace}/postgres-exporter` to query metrics starting with **pg**.
----End

Setting a Dashboard and Alarm Rule on AOM

By setting a dashboard, you can monitor CCE cluster data on the same screen. By setting an alarm rule, you can detect cluster faults and implement warning in a timely manner.

- Setting a dashboard
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Dashboard**. On the displayed page, click **Add Dashboard** to add a dashboard. For details, see [Creating a Dashboard](#).
 - c. On the **Dashboard** page, select a Prometheus instance for CCE and click **Add Graph**. For details, see [Adding a Graph to a Dashboard](#).
- Setting an alarm rule
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Alarm Management > Alarm Rules**.
 - c. On the **Metric/Event Alarm Rules** tab page, click **Create** to create an alarm rule. For details, see [Creating a Metric Alarm Rule](#).

6.2 Connecting MySQL Exporter

Application Scenario

MySQL Exporter collects MySQL database metrics. Core database metrics collected through Exporter are used for alarm reporting and dashboard display. Currently, Exporter supports MySQL 5.6 or later. If the MySQL version is earlier than 5.6, some metrics may fail to be collected.

NOTE

You are advised to use CCE for unified Exporter management.

Prerequisites

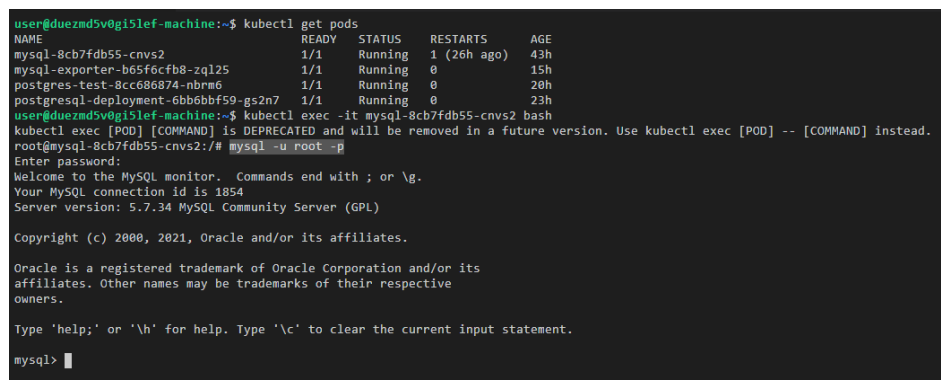
- A CCE cluster has been created and MySQL has been installed.
- Your service has been connected for Prometheus monitoring and a CCE cluster has also been connected. For details, see [Prometheus Instance for CCE](#).
- You have uploaded the [mysql_exporter](#) image to SoftWare Repository for Container (SWR). For details, see [Uploading an Image Through a Container Engine Client](#).

Database Authorization

Step 1 Log in to the cluster and run the following command:

```
kubectl exec -it ${mysql_podname} bash
mysql -u root -p
```

Figure 6-2 Executing the command



```
user@dueznd5v0g151ef-machine:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mysql-8cb7fdb55-cnvs2              1/1     Running  1 (26h ago) 43h
mysql-exporter-b65f6cfb8-zql25     1/1     Running  0           15h
postgres-test-8cc686874-nbrm6     1/1     Running  0           20h
postgresl-deployment-6bb6bbf59-gs2n7 1/1     Running  0           23h
user@dueznd5v0g151ef-machine:~$ kubectl exec -it mysql-8cb7fdb55-cnvs2 bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@mysql-8cb7fdb55-cnvs2:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1854
Server version: 5.7.34 MySQL Community Server (GPL)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Step 2 Log in to the database and run the following command:

```
CREATE USER 'exporter'@'x.x.x.x(hostip)' IDENTIFIED BY 'xxxx(password)' WITH MAX_USER_CONNECTIONS 3;
GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO 'exporter'@'x.x.x.x(hostip)';
```

Step 3 Check whether the authorization is successful.

Enter the following SQL statement to check whether there is any Exporter data. *host* indicates the IP address of the node where the MySQL database is located.

```
select user,host from mysql.user;
```

Figure 6-3 SQL statement

```
mysql> select user,host from mysql.user;
+-----+-----+
| user      | host      |
+-----+-----+
| root      | %         |
| exporter  | 192.168.0.205 |
| mysql.session | localhost |
| mysql.sys | localhost |
| root      | localhost |
+-----+-----+
5 rows in set (0.00 sec)

mysql> █
```

----End

Deploying MySQL Exporter

- Step 1** Log in to the CCE console.
- Step 2** Click the connected cluster. The cluster management page is displayed.
- Step 3** Perform the following operations to deploy Exporter:

1. Use **Secret** to manage MySQL connection strings.

In the navigation pane, choose **ConfigMaps and Secrets**. In the upper right corner, click **Create from YAML** and enter the following **.yaml** file. The password is encrypted based on Opaque requirements.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret
  namespace: default
type: Opaque
stringData:
  datasource: "user:password@tcp(ip:port)/" # MySQL connection string, which needs to be encrypted.
```

 **NOTE**

For details about how to configure a secret, see [Creating a Secret](#).

2. Deploy MySQL Exporter.

In the navigation pane, choose **Workloads**. In the upper right corner, click **Create Workload**. Then select the **Deployment** workload and select a desired namespace to deploy MySQL Exporter. YAML configuration example for deploying Exporter:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: mysql-exporter # Change the name based on service requirements. You are advised to add the MySQL instance information, for example, ckafka-2vrgx9fd-mysql-exporter.
  name: mysql-exporter # Change the name based on service requirements. You are advised to add the MySQL instance information, for example, ckafka-2vrgx9fd-mysql-exporter.
  namespace: default # Must be the same as the namespace of MySQL.
spec:
```



```
replicas: 1
selector:
  matchLabels:
    k8s-app: mysql-exporter # Change the name based on service requirements. You are advised to
add the MySQL instance information, for example, ckafka-2vrgx9fd-mysql-exporter.
template:
  metadata:
    labels:
      k8s-app: mysql-exporter # Change the name based on service requirements. You are advised to
add the MySQL instance information, for example, ckafka-2vrgx9fd-mysql-exporter.
  spec:
    containers:
      - env:
        - name: DATA_SOURCE_NAME
          valueFrom:
            secretKeyRef:
              name: mysql-secret
              key: datasource
          image: swr.cn-north-4.myhuaweicloud.com/aom-exporter/mysql-d-exporter:v0.12.1
          imagePullPolicy: IfNotPresent
          name: mysql-exporter
          ports:
            - containerPort: 9104
              name: metric-port
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
        dnsPolicy: ClusterFirst
        imagePullSecrets:
          - name: default-secret
        restartPolicy: Always
        schedulerName: default-scheduler
        securityContext: {}
        terminationGracePeriodSeconds: 30
---
apiVersion: v1
kind: Service
metadata:
  name: mysql-exporter
spec:
  type: NodePort
  selector:
    k8s-app: mysql-exporter
  ports:
    - protocol: TCP
      nodePort: 30337
      port: 9104
      targetPort: 9104
```

NOTE

For details about Exporter parameters, see [mysql-exporter](#).

3. Check whether MySQL Exporter is successfully deployed.
 - a. On the **Deployments** tab page, click the Deployment created in [Step 3.2](#). In the pod list, choose **More > View Logs** in the **Operation** column. The Exporter is successfully started and its access address is exposed.
 - b. Perform verification using one of the following methods:
 - Log in to a cluster node and run either of the following commands:
`curl http://{Cluster IP address}:9104/metrics`
`curl http://{Private IP address of any node in the cluster}:30337/metrics`
 - In the instance list, choose **More > Remote Login** in the **Operation** column and run the following command:
`curl http://localhost:9104/metric`

- Access `http://{Public IP address of any node in the cluster}:30337/metrics`.

Figure 6-4 Accessing a cluster node

```

30337/metrics
# HELP mysql_exporter_last_scrape_error Whether the last scrape of metrics from MySQL resulted in an error (1 for error, 0 for success).
# TYPE mysql_exporter_last_scrape_error gauge
mysql_exporter_last_scrape_error 0
# HELP mysql_exporter_scrapes_total Total number of times MySQL was scraped for metrics.
# TYPE mysql_exporter_scrapes_total counter
mysql_exporter_scrapes_total 34
# HELP mysql_global_status_aborted_clients Generic metric from SHOW GLOBAL STATUS.
# TYPE mysql_global_status_aborted_clients untyped
mysql_global_status_aborted_clients 0
# HELP mysql_global_status_aborted_connects Generic metric from SHOW GLOBAL STATUS.
# TYPE mysql_global_status_aborted_connects untyped
mysql_global_status_aborted_connects 20
# HELP mysql_global_status_binlog_cache_disk_use Generic metric from SHOW GLOBAL STATUS.
# TYPE mysql_global_status_binlog_cache_disk_use untyped
mysql_global_status_binlog_cache_disk_use 0
# HELP mysql_global_status_binlog_cache_use Generic metric from SHOW GLOBAL STATUS.
# TYPE mysql_global_status_binlog_cache_use untyped
mysql_global_status_binlog_cache_use 0
# HELP mysql_global_status_binlog_stat_cache_disk_use Generic metric from SHOW GLOBAL STATUS.
# TYPE mysql_global_status_binlog_stat_cache_disk_use untyped
mysql_global_status_binlog_stat_cache_disk_use 0
# HELP mysql_global_status_binlog_stat_cache_use Generic metric from SHOW GLOBAL STATUS.
# TYPE mysql_global_status_binlog_stat_cache_use untyped
mysql_global_status_binlog_stat_cache_use 0
# HELP mysql_global_status_buffer_pool_dirty_pages InnoDB buffer pool dirty pages.
# TYPE mysql_global_status_buffer_pool_dirty_pages gauge
mysql_global_status_buffer_pool_dirty_pages 0
# HELP mysql_global_status_buffer_pool_page_changes_total InnoDB buffer pool page state changes.
# TYPE mysql_global_status_buffer_pool_page_changes_total counter
mysql_global_status_buffer_pool_page_changes_total {operations='flushed'} 53
# HELP mysql_global_status_buffer_pool_pages InnoDB buffer pool pages by state.
# TYPE mysql_global_status_buffer_pool_pages gauge
mysql_global_status_buffer_pool_pages {state='data'} 327
mysql_global_status_buffer_pool_pages {state='free'} 7865
mysql_global_status_buffer_pool_pages {state='misc'} 0
# HELP mysql_global_status_bytes_received Generic metric from SHOW GLOBAL STATUS.
# TYPE mysql_global_status_bytes_received untyped
mysql_global_status_bytes_received 28608
# HELP mysql_global_status_bytes_sent Generic metric from SHOW GLOBAL STATUS.
# TYPE mysql_global_status_bytes_sent untyped
mysql_global_status_bytes_sent 1.096652e+06
# HELP mysql_global_status_commands_total Total number of executed MySQL commands.
# TYPE mysql_global_status_commands_total counter
mysql_global_status_commands_total {command='admin_commands'} 34
mysql_global_status_commands_total {command='alter_db'} 0
mysql_global_status_commands_total {command='alter_db_upgrade'} 0
mysql_global_status_commands_total {command='alter_event'} 0
mysql_global_status_commands_total {command='alter_function'} 0
mysql_global_status_commands_total {command='alter_instance'} 0
mysql_global_status_commands_total {command='alter_procedure'} 0
mysql_global_status_commands_total {command='alter_server'} 0
mysql_global_status_commands_total {command='alter_table'} 0
mysql_global_status_commands_total {command='alter_tablespace'} 0
mysql_global_status_commands_total {command='alter_user'} 0
mysql_global_status_commands_total {command='analyze'} 0
mysql_global_status_commands_total {command='assign_to_keycache'} 0
mysql_global_status_commands_total {command='begin'} 0
mysql_global_status_commands_total {command='binlog'} 0
mysql_global_status_commands_total {command='call_procedure'} 0
mysql_global_status_commands_total {command='change_db'} 1

```

----End

Collecting Service Data of the CCE Cluster

Add **PodMonitor** to configure a collection rule for monitoring the service data of applications deployed in the CCE cluster.

Configuration information:

```

apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: mysql-exporter
  namespace: default
spec:
  namespaceSelector:
    matchNames:
      - default # Namespace where Exporter is located.
  podMetricsEndpoints:
    - interval: 30s
      path: /metrics
      port: metric-port
  selector:
    matchLabels:
      k8s-app: mysql-exporter

```

 NOTE

In this example, metrics are collected every 30s. Therefore, you can check the reported metrics on the AOM page about 30s later.

Verifying that Metrics Can Be Reported to AOM

- Step 1** Log in to the AOM 2.0 console.
- Step 2** In the navigation pane on the left, choose **Prometheus Monitoring > Instances**.
- Step 3** Click the Prometheus instance connected to the CCE cluster. The instance details page is displayed.
- Step 4** On the **Metrics** tab page of the **Metric Management** page, select your target cluster.
- Step 5** Select job `{namespace}/mysql-exporter` to query custom metrics starting with **mysql**.

----End

Setting a Dashboard and Alarm Rule on AOM

By setting a dashboard, you can monitor CCE cluster data on the same screen. By setting an alarm rule, you can detect cluster faults and implement warning in a timely manner.

- Setting a dashboard
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Dashboard**. On the displayed page, click **Add Dashboard** to add a dashboard. For details, see [Creating a Dashboard](#).
 - c. On the **Dashboard** page, select a Prometheus instance for CCE and click **Add Graph**. For details, see [Adding a Graph to a Dashboard](#).
- Setting an alarm rule
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Alarm Management > Alarm Rules**.
 - c. On the **Metric/Event Alarm Rules** tab page, click **Create** to create an alarm rule. For details, see [Creating a Metric Alarm Rule](#).

6.3 Connecting Kafka Exporter

Application Scenario

When using Kafka, you need to monitor their running, for example, checking the cluster status and whether messages are stacked. The Prometheus monitoring function monitors Kafka running using Exporter in the CCE container scenario. This section describes how to deploy Kafka Exporter and implement alarm access.

 NOTE

You are advised to use CCE for unified Exporter management.

Prerequisites

- A CCE cluster has been created and Kafka has been installed.
- Your service has been connected for Prometheus monitoring and a CCE cluster has also been connected. For details, see [Prometheus Instance for CCE](#).
- You have uploaded the [kafka_exporter](#) image to SoftWare Repository for Container (SWR). For details, see [Uploading an Image Through a Container Engine Client](#).

Deploying Kafka Exporter

Step 1 Log in to the CCE console.

Step 2 Click the connected cluster. The cluster management page is displayed.

Step 3 Perform the following operations to deploy Exporter:

1. Deploy Kafka Exporter.

In the navigation pane, choose **Workloads**. In the upper right corner, click **Create Workload**. Then select the **Deployment** workload and select a desired namespace to deploy Kafka Exporter. YAML configuration example for deploying Exporter:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: kafka-exporter # Change the name based on service requirements. You are advised to add
the Kafka instance information, for example, ckafka-2vrgx9fd-kafka-exporter.
  name: kafka-exporter # Change the name based on service requirements. You are advised to add
the Kafka instance information, for example, ckafka-2vrgx9fd-kafka-exporter.
  namespace: default # Namespace of an existing cluster
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: kafka-exporter # Change the name based on service requirements. You are advised to
add the Kafka instance information, for example, ckafka-2vrgx9fd-kafka-exporter.
  template:
    metadata:
      labels:
        k8s-app: kafka-exporter # Change the name based on service requirements. You are advised to
add the Kafka instance information, for example, ckafka-2vrgx9fd-kafka-exporter.
    spec:
      containers:
        - args:
            - --kafka.server=120.46.215.4:30092 # Address of the Kafka instance
          image: swr.cn-north-4.myhuaweicloud.com/mall-swarm-demo/kafka-exporter:latest
          imagePullPolicy: IfNotPresent
          name: kafka-exporter
          ports:
            - containerPort: 9308
              name: metric-port # Required when you configure a collection task
          securityContext:
            privileged: false
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          dnsPolicy: ClusterFirst
          imagePullSecrets:
```

```
- name: default-secret
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  terminationGracePeriodSeconds: 30
---
apiVersion: v1
kind: Service
metadata:
  name: kafka-exporter
spec:
  type: NodePort
  selector:
    k8s-app: kafka-exporter
  ports:
    - protocol: TCP
      nodePort: 30091
      port: 9308
      targetPort: 9308
```

 **NOTE**

For more details about Exporter parameters, see [kafka-exporter](#).

2. Check whether Kafka Exporter is successfully deployed.
 - a. On the **Deployments** tab page, click the Deployment created in [Step 3.1](#). In the pod list, choose **More > View Logs** in the **Operation** column. The Exporter is successfully started and its access address is exposed.
 - b. Perform verification using one of the following methods:
 - Log in to a cluster node and run either of the following commands:
`curl http://{Cluster IP address}:9308/metrics`
`curl http://{Private IP address of any node in the cluster}:30091/metrics`
 - In the instance list, choose **More > Remote Login** in the **Operation** column and run the following command:
`curl http://localhost:9308/metric`
 - Access `http://{Public IP address of any node in the cluster}:30091/metrics`.

Figure 6-5 Accessing a cluster node

```

30091/metrics
go_memstats_mcache_inuse_bytes 19200
# HELP go_memstats_mcache_sys_bytes Number of bytes used for mcache structures obtained from system.
# TYPE go_memstats_mcache_sys_bytes gauge
go_memstats_mcache_sys_bytes 25768
# HELP go_memstats_mspan_inuse_bytes Number of bytes in use by mspan structures.
# TYPE go_memstats_mspan_inuse_bytes gauge
go_memstats_mspan_inuse_bytes 46240
# HELP go_memstats_mspan_sys_bytes Number of bytes used for mspan structures obtained from system.
# TYPE go_memstats_mspan_sys_bytes gauge
go_memstats_mspan_sys_bytes 49152
# HELP go_memstats_next_gc_bytes Number of heap bytes when next garbage collection will take place.
# TYPE go_memstats_next_gc_bytes gauge
go_memstats_next_gc_bytes 4.473824e+06
# HELP go_memstats_other_sys_bytes Number of bytes used for other system allocations.
# TYPE go_memstats_other_sys_bytes gauge
go_memstats_other_sys_bytes 1.074585e+06
# HELP go_memstats_stack_inuse_bytes Number of bytes in use by the stack allocator.
# TYPE go_memstats_stack_inuse_bytes gauge
go_memstats_stack_inuse_bytes 524288
# HELP go_memstats_stack_sys_bytes Number of bytes obtained from system for stack allocator.
# TYPE go_memstats_stack_sys_bytes gauge
go_memstats_stack_sys_bytes 524288
# HELP go_memstats_sys_bytes Number of bytes obtained from system.
# TYPE go_memstats_sys_bytes gauge
go_memstats_sys_bytes 1.5156485e+07
# HELP go_threads Number of OS threads created.
# TYPE go_threads gauge
go_threads 6
# HELP kafka_brokers Number of Brokers in the Kafka Cluster.
# TYPE kafka_brokers gauge
kafka_brokers 1
# HELP kafka_exporter_build_info A metric with a constant '1' value labeled by version, revision, branch, and goversion from which kafka_exporter was built.
# TYPE kafka_exporter_build_info gauge
kafka_exporter_build_info {branch="HEAD", goversion="go1.17.3", revision="15e4ad6a9ea8203135d4b974e825f22e31c750e5", version="1.4.2"} 1
# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 0.02
# HELP process_max_fds Maximum number of open file descriptors.
# TYPE process_max_fds gauge
process_max_fds 1.048576e+06
# 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 1.513472e+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 1.70253762409e+09
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 7.3420944e+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 1.8446744037709552e+19
# HELP promhttp_metric_handler_requests_in_flight Current number of scrapes being served.

```

----End

Collecting Service Data of the CCE Cluster

Add PodMonitor to configure a collection rule for monitoring the service data of applications deployed in the CCE cluster.

NOTE

In the following example, metrics are collected every 30s. Therefore, you can check the reported metrics on the AOM page about 30s later.

Configuration information:

```

apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: kafka-exporter
  namespace: default
spec:
  namespaceSelector:
    matchNames:
      - default # Namespace where Exporter is located.
  podMetricsEndpoints:
    - interval: 30s
      path: /metrics
      port: metric-port
  selector:
    matchLabels:
      k8s-app: kafka-exporter

```

Verifying that Metrics Can Be Reported to AOM

Step 1 Log in to the AOM 2.0 console.

Step 2 In the navigation pane on the left, choose **Prometheus Monitoring > Instances**.

- Step 3** Click the Prometheus instance connected to the CCE cluster. The instance details page is displayed.
- Step 4** On the **Metrics** tab page of the **Metric Management** page, select your target cluster.
- Step 5** Select job `{namespace}/kafka-exporter` to query custom metrics starting with **kafka**.
- End

Setting a Dashboard and Alarm Rule on AOM

By setting a dashboard, you can monitor CCE cluster data on the same screen. By setting an alarm rule, you can detect cluster faults and implement warning in a timely manner.

- Setting a dashboard
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Dashboard**. On the displayed page, click **Add Dashboard** to add a dashboard. For details, see [Creating a Dashboard](#).
 - c. On the **Dashboard** page, select a Prometheus instance for CCE and click **Add Graph**. For details, see [Adding a Graph to a Dashboard](#).
- Setting an alarm rule
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Alarm Management > Alarm Rules**.
 - c. On the **Metric/Event Alarm Rules** tab page, click **Create** to create an alarm rule. For details, see [Creating a Metric Alarm Rule](#).

6.4 Connecting Memcached Exporter

Application Scenario

When using Memcached, you need to monitor their running and locate their faults in a timely manner. The Prometheus monitoring function monitors Memcached running using Exporter in the CCE container scenario. This section describes how to monitor Memcached.

NOTE

You are advised to use CCE for unified Exporter management.

Prerequisites

- A CCE cluster has been created and Memcached has been installed.
- Your service has been connected for Prometheus monitoring and a CCE cluster has also been connected. For details, see [Prometheus Instance for CCE](#).
- You have uploaded the [memcached_exporter](#) image to Software Repository for Container (SWR). For details, see [Uploading an Image Through a Container Engine Client](#).

Deploying Memcached Exporter

Step 1 Log in to the CCE console.

Step 2 Click the connected cluster. The cluster management page is displayed.

Step 3 Perform the following operations to deploy Exporter:

1. Configure a secret.

In the navigation pane, choose **ConfigMaps and Secrets**. Then click **Create from YAML** in the upper right corner of the page. YAML configuration example:

```
apiVersion: v1
kind: Secret
metadata:
  name: memcached-exporter-secret
  namespace: default
type: Opaque
stringData:
  memcachedURI: 120.46.215.4:11211 # Memcached address
```

NOTE

- Format of the Memcached connection string: **http://{ip}:{port}**.
- For details about how to configure a secret, see [Creating a Secret](#).

2. Deploy Memcached Exporter.

In the navigation pane, choose **Workloads**. On the **Deployments** tab page, click **Create from YAML** in the upper right corner and then configure a YAML file to deploy Exporter.

YAML configuration example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: memcached-exporter # Change the value based on service requirements.
  name: memcached-exporter # Change the value based on service requirements.
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: memcached-exporter # Change the value based on service requirements.
  template:
    metadata:
      labels:
        k8s-app: memcached-exporter # Change the value based on service requirements.
    spec:
      containers:
        - env:
            - name: Memcached_Url
              valueFrom:
                secretKeyRef:
                  name: memcached-exporter-secret # Secret name specified in the previous step.
                  key: memcachedURI # Secret key specified in the previous step.
            - name: Memcached_ALL
              value: "true"
          image: swr.cn-east-3.myhuaweicloud.com/aom-org/bitnami/memcached-exporter:0.13.0 #
          imagePullPolicy: IfNotPresent
          name: memcached-exporter
          ports:
            - containerPort: 9150
              name: metric-port
```



```

securityContext:
  privileged: false
  terminationMessagePath: /dev/termination-log
  terminationMessagePolicy: File
dnsPolicy: ClusterFirst
imagePullSecrets:
- name: default-secret
restartPolicy: Always
schedulerName: default-scheduler
securityContext: {}
terminationGracePeriodSeconds: 30
---
apiVersion: v1
kind: Service
metadata:
  name: memcached-exporter
spec:
  type: NodePort
  selector:
    k8s-app: memcached-exporter
  ports:
    - protocol: TCP
      nodePort: 30122
      port: 9150
      targetPort: 9150
  
```

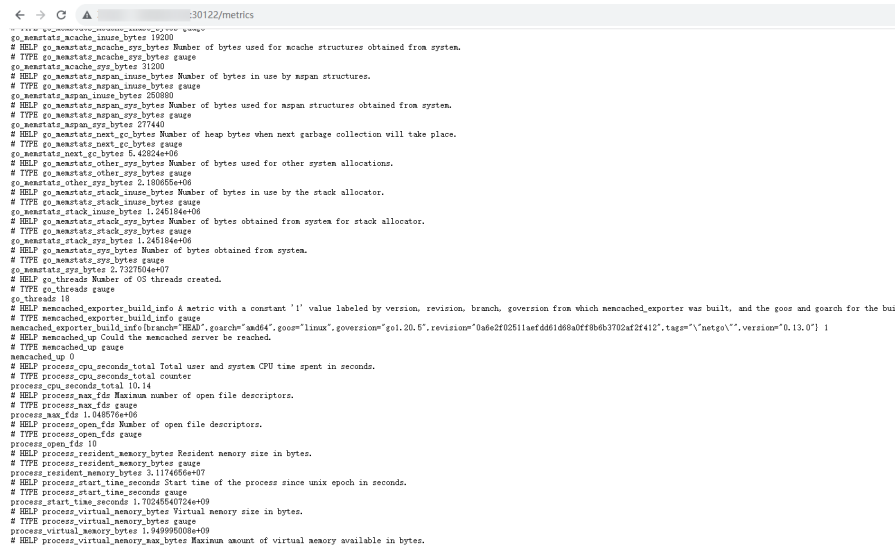
NOTE

For more details about Exporter parameters, see [memcached_exporter](#).

3. Check whether Memcached Exporter is successfully deployed.
 - a. On the **Deployments** tab page, click the Deployment created in **Step 3.2**. In the pod list, choose **More > View Logs** in the **Operation** column. The Exporter is successfully started and its access address is exposed.
 - b. Perform verification using one of the following methods:

- Log in to a cluster node and run either of the following commands:
curl http://{Cluster IP address}:9150/metrics
curl http://{Private IP address of any node in the cluster}:30122/metrics
- Access **http://{Public IP address of any node in the cluster}:30122/metrics**.

Figure 6-6 Accessing a cluster node



- In the instance list, choose **More > Remote Login** in the **Operation** column and run the following command:

```
curl http://localhost:9150/metric
```

Figure 6-7 Executing the command

```
user@ungnt6cs5eps2ff-machine:~$ curl http://localhost:9150/metrics
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 9
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.20.5"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 504008
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 504008
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 4545
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 0
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 6.74584e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 504008
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 1.753088e+06
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
```

----End

Adding a Collection Task

Add PodMonitor to configure a collection rule for monitoring the service data of applications deployed in the CCE cluster.

NOTE

In the following example, metrics are collected every 30s. Therefore, you can check the reported metrics on the AOM page about 30s later.

```
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: memcached-exporter
  namespace: default
spec:
  namespaceSelector:
    matchNames:
      - default # Namespace where Exporter is located.
  podMetricsEndpoints:
    - interval: 30s
      path: /metrics
      port: metric-port
  selector:
    matchLabels:
      k8s-app: memcached-exporter
```

Verifying that Metrics Can Be Reported to AOM

- Step 1** Log in to the AOM 2.0 console.
 - Step 2** In the navigation pane on the left, choose **Prometheus Monitoring > Instances**.
 - Step 3** Click the Prometheus instance connected to the CCE cluster. The instance details page is displayed.
 - Step 4** On the **Metrics** tab page of the **Metric Management** page, select your target cluster.
 - Step 5** Select job `{namespace}memcached-exporter` to query metrics starting with `go_memstats`.
- End

Setting a Dashboard and Alarm Rule on AOM

By setting a dashboard, you can monitor CCE cluster data on the same screen. By setting an alarm rule, you can detect cluster faults and implement warning in a timely manner.

- Setting a dashboard
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Dashboard**. On the displayed page, click **Add Dashboard** to add a dashboard. For details, see [Creating a Dashboard](#).
 - c. On the **Dashboard** page, select a Prometheus instance for CCE and click **Add Graph**. For details, see [Adding a Graph to a Dashboard](#).
- Setting an alarm rule
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Alarm Management > Alarm Rules**.
 - c. On the **Metric/Event Alarm Rules** tab page, click **Create** to create an alarm rule. For details, see [Creating a Metric Alarm Rule](#).

6.5 Connecting MongoDB Exporter

Application Scenario

When using MongoDB, you need to monitor MongoDB running and locate their faults in a timely manner. The Prometheus monitoring function monitors MongoDB running using Exporter in the CCE container scenario. This section describes how to deploy MongoDB Exporter and implement alarm access.

NOTE

You are advised to use CCE for unified Exporter management.

Prerequisites

- A CCE cluster has been created and MongoDB has been installed.

- Your service has been connected for Prometheus monitoring and a CCE cluster has also been connected. For details, see [Prometheus Instance for CCE](#).
- You have uploaded the [mongodb_exporter](#) image to SoftWare Repository for Container (SWR). For details, see [Uploading an Image Through a Container Engine Client](#).

Deploying MongoDB Exporter

Step 1 Log in to the CCE console.

Step 2 Click the connected cluster. The cluster management page is displayed.

Step 3 Perform the following operations to deploy Exporter:

1. Configure a secret.

In the navigation pane, choose **ConfigMaps and Secrets**. Then click **Create from YAML** in the upper right corner of the page. YAML configuration example:

```
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret-test
  namespace: default
type: Opaque
stringData:
  datasource: "mongodb://{user}:{passwd}@{host1}:{port1},{host2}:{port2},{host3}:{port3}/admin" #
Corresponding URI.
```

NOTE

- The password has been encrypted based on Opaque requirements.
- For details about how to configure a secret, see [Creating a Secret](#).

2. Deploy MongoDB Exporter.

In the navigation pane, choose **Workloads**. In the upper right corner, click **Create Workload**. Then select the **Deployment** workload and select a desired namespace to deploy MongoDB Exporter. YAML configuration example for deploying Exporter:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: mongodb-exporter # Change the value based on service requirements. You are advised to
add the MongoDB instance information.
  name: mongodb-exporter # Change the value based on service requirements. You are advised to add
the MongoDB instance information.
  namespace: default #Must be the same as the namespace of MongoDB.
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: mongodb-exporter # Change the value based on service requirements. You are advised
to add the MongoDB instance information.
  template:
    metadata:
      labels:
        k8s-app: mongodb-exporter # Change the value based on service requirements. You are advised
to add the MongoDB instance information.
    spec:
      containers:
        - args:
            - --collect.database # Enable collection of database metrics.
```

```
- --collect.collection # Enable collection of metric sets.
- --collect.topmetrics # Enable collection of database header metrics.
- --collect.indexusage # Enable collection of index usage statistics.
- --collect.connpoolstats # Enable collection of MongoDB connection pool statistics.
env:
- name: MONGODB_URI
  valueFrom:
    secretKeyRef:
      name: mongodb-secret-test
      key: datasource
image: swr.cn-north-4.myhuaweicloud.com/mall-swarm-demo/mongodb-exporter:0.10.0
imagePullPolicy: IfNotPresent
name: mongodb-exporter
ports:
- containerPort: 9216
  name: metric-port # Required when you configure a collection task.
securityContext:
  privileged: false
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
dnsPolicy: ClusterFirst
imagePullSecrets:
- name: default-secret
restartPolicy: Always
schedulerName: default-scheduler
securityContext: { }
terminationGracePeriodSeconds: 30
---
apiVersion: v1
kind: Service
metadata:
  name: mongodb-exporter
spec:
  type: NodePort
  selector:
    k8s-app: mongodb-exporter
  ports:
    - protocol: TCP
      nodePort: 30003
      port: 9216
      targetPort: 9216
```

NOTE

For more details about Exporter parameters, see [mongodb_exporter](#).

3. Check whether MongoDB Exporter is successfully deployed.
 - a. On the **Deployments** tab page, click the Deployment created in [Step 3.2](#). In the pod list, choose **More > View Logs** in the **Operation** column. The Exporter is successfully started and its access address is exposed.
 - b. Perform verification using one of the following methods:
 - Log in to a cluster node and run either of the following commands:
`curl http://{Cluster IP address}:9216/metrics`
`curl http://{Private IP address of any node in the cluster}:30003/metrics`
 - Access `http://{Public IP address of any node in the cluster}:30003/metrics`.

Figure 6-8 Accessing a cluster node

```

< -> ↻ 🔍 :30003/metrics

# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 8
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.11.13"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.81956e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.81956e+06
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 3124
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 3308
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 0
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 2.23436e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 1.81956e+06
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 6.3234048e+07
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 3.31776e+06
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 16998
# HELP go_memstats_heap_released_bytes Number of heap bytes released to OS.
# TYPE go_memstats_heap_released_bytes gauge
go_memstats_heap_released_bytes 0
# HELP go_memstats_heap_sys_bytes Number of heap bytes obtained from system.
# TYPE go_memstats_heap_sys_bytes gauge
go_memstats_heap_sys_bytes 6.6551808e+07
# HELP go_memstats_last_gc_time_seconds Number of seconds since 1970 of last garbage collection.
# TYPE go_memstats_last_gc_time_seconds gauge
go_memstats_last_gc_time_seconds 0
# HELP go_memstats_lookups_total Total number of pointer lookups.
# TYPE go_memstats_lookups_total counter
go_memstats_lookups_total 0

```

- In the instance list, choose **More > Remote Login** in the **Operation** column and run the following command:
curl http://localhost:9216/metric

----End

Collecting Service Data of the CCE Cluster

Add PodMonitor to configure a collection rule for monitoring the service data of applications deployed in the CCE cluster.

NOTE

In the following example, metrics are collected every 30s. Therefore, you can check the reported metrics on the AOM page about 30s later.

```

apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: mongodb-exporter
  namespace: default
spec:
  namespaceSelector:
    matchNames:
      - default # Namespace where Exporter is located.
  podMetricsEndpoints:
    - interval: 30s
      path: /metrics
      port: metric-port
      selector:
        matchLabels:
          k8s-app: mongodb-exporter

```

Verifying that Metrics Can Be Reported to AOM

- Step 1** Log in to the AOM 2.0 console.
 - Step 2** In the navigation pane on the left, choose **Prometheus Monitoring > Instances**.
 - Step 3** Click the Prometheus instance connected to the CCE cluster. The instance details page is displayed.
 - Step 4** On the **Metrics** tab page of the **Metric Management** page, select your target cluster.
 - Step 5** Select job *{namespace}/MongoDB-exporter* to query custom metrics starting with **mongodb**.
- End

Setting a Dashboard and Alarm Rule on AOM

By setting a dashboard, you can monitor CCE cluster data on the same screen. By setting an alarm rule, you can detect cluster faults and implement warning in a timely manner.

- Setting a dashboard
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Dashboard**. On the displayed page, click **Add Dashboard** to add a dashboard. For details, see [Creating a Dashboard](#).
 - c. On the **Dashboard** page, select a Prometheus instance for CCE and click **Add Graph**. For details, see [Adding a Graph to a Dashboard](#).
- Setting an alarm rule
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Alarm Management > Alarm Rules**.
 - c. On the **Metric/Event Alarm Rules** tab page, click **Create** to create an alarm rule. For details, see [Creating a Metric Alarm Rule](#).

6.6 Connecting Elasticsearch Exporter

Application Scenario

When using Elasticsearch, you need to monitor Elasticsearch running, such as the cluster and index status. The Prometheus monitoring function monitors Elasticsearch running using Exporter in the CCE container scenario. This section describes how to deploy Elasticsearch Exporter and implement alarm access.

NOTE

You are advised to use CCE for unified Exporter management.

Prerequisites

- A CCE cluster has been created and Elasticsearch has been installed.

- Your service has been connected for Prometheus monitoring and a CCE cluster has also been connected. For details, see [Prometheus Instance for CCE](#).
- You have uploaded the [elasticsearch_exporter](#) image to SoftWare Repository for Container (SWR). For details, see [Uploading an Image Through a Container Engine Client](#).

Deploying Elasticsearch Exporter

Step 1 Log in to the CCE console.

Step 2 Click the connected cluster. The cluster management page is displayed.

Step 3 Perform the following operations to deploy Exporter:

1. Configure a secret.

In the navigation pane, choose **ConfigMaps and Secrets**. Then click **Create from YAML** in the upper right corner of the page. The following shows a YAML configuration example:

```
apiVersion: v1
kind: Secret
metadata:
  name: es-secret-test
  namespace: default
type: Opaque
stringData:
  esURI: http://124.70.14.51:30920 # URI of Elasticsearch. Use the IP address of the cluster or any
  node in the cluster.
```

NOTE

- Format of the Elasticsearch connection string: `<proto>://<user>:<password>@<host>:<port>`, for example, **http://admin:pass@localhost:9200**. You can also leave the password blank, for example, **http://10.247.43.50:9200**.
 - The password has been encrypted based on Opaque requirements.
 - For details about how to configure a secret, see [Creating a Secret](#).
2. Deploy Elasticsearch Exporter.

In the navigation pane, choose **Workloads**. In the upper right corner, click **Create Workload**. Then select the **Deployment** workload and a desired namespace to deploy Elasticsearch Exporter. YAML configuration example for deploying Exporter:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: es-exporter # Change the value based on service requirements.
  name: es-exporter # Change the value based on service requirements.
  namespace: default #Select a proper namespace to deploy Exporter. If no namespace is available,
  create one.
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: es-exporter # Change the value based on service requirements.
  template:
    metadata:
      labels:
        k8s-app: es-exporter # Change the value based on service requirements.
    spec:
      containers:
```



```
- env:
  - name: ES_URI
    valueFrom:
      secretKeyRef:
        name: es-secret-test # Secret name specified in the previous step.
        key: esURI # Secret key specified in the previous step.
  - name: ES_ALL
    value: "true"
image: swr.cn-north-4.myhuaweicloud.com/mall-swarm-demo/es-exporter:1.1.0
imagePullPolicy: IfNotPresent
name: es-exporter
ports:
  - containerPort: 9114
    name: metric-port
securityContext:
  privileged: false
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
dnsPolicy: ClusterFirst
imagePullSecrets:
  - name: default-secret
restartPolicy: Always
schedulerName: default-scheduler
securityContext: {}
terminationGracePeriodSeconds: 30
---
apiVersion: v1
kind: Service
metadata:
  name: es-exporter
  name-space: default # Must be the same as the namespace where Exporter is deployed.
spec:
  type: NodePort
  selector:
    k8s-app: es-exporter
  ports:
    - protocol: TCP
      nodePort: 30921
      port: 9114
      targetPort: 9114
```

NOTE

In the preceding example, **ES_ALL** is used to collect all Elasticsearch monitoring items. You can change parameters if needed. For more details about Exporter parameters, see [elasticsearch_exporter](#).

3. Check whether Elasticsearch Exporter is successfully deployed.
 - a. On the **Deployments** tab page, click the Deployment created in [Step 3.2](#). In the pod list, choose **More > View Logs** in the **Operation** column. The Exporter is successfully started and its access address is exposed.
 - b. Perform verification using one of the following methods:
 - Log in to a cluster node and run either of the following commands:
`curl http://{Cluster IP address}:9114/metrics`
`curl http://{Private IP address of any node in the cluster}:30921/metrics`
 - Access `http://{Public IP address of any node in the cluster}:30921/metrics`.

Figure 6-9 Accessing a cluster node

```
← → 30921/metrics
# HELP elasticsearch_breaker_estimated_size_bytes Estimated size in bytes of breaker
# TYPE elasticsearch_breaker_estimated_size_bytes gauge
elasticsearch_breaker_estimated_size_bytes{accounting="cluster",docker="cluster",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 0
elasticsearch_breaker_estimated_size_bytes{accounting="in_flight_requests",cluster="docker",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 0
elasticsearch_breaker_estimated_size_bytes{accounting="request",cluster="docker",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 0
elasticsearch_breaker_estimated_size_bytes{accounting="request",cluster="docker",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 0
# HELP elasticsearch_breaker_list_size_bytes List size in bytes for breaker
# TYPE elasticsearch_breaker_list_size_bytes gauge
elasticsearch_breaker_list_size_bytes{accounting="cluster",docker="cluster",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 1.98502536e+09
elasticsearch_breaker_list_size_bytes{accounting="in_flight_requests",cluster="docker",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 4.20010214e+08
elasticsearch_breaker_list_size_bytes{accounting="request",cluster="docker",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 1.98502536e+09
elasticsearch_breaker_list_size_bytes{accounting="request",cluster="docker",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 1.01171259e+09
# HELP elasticsearch_breaker_overhead_of_circuit_breaker
# TYPE elasticsearch_breaker_overhead_of_circuit_breaker counter
elasticsearch_breaker_overhead_of_circuit_breaker{accounting="cluster",docker="cluster",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 1
elasticsearch_breaker_overhead_of_circuit_breaker{accounting="in_flight_requests",cluster="docker",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 1.53
elasticsearch_breaker_overhead_of_circuit_breaker{accounting="request",cluster="docker",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 2
elasticsearch_breaker_overhead_of_circuit_breaker{accounting="request",cluster="docker",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 1
# HELP elasticsearch_breaker_triggered_for_breaker
# TYPE elasticsearch_breaker_triggered_for_breaker counter
elasticsearch_breaker_triggered_for_breaker{accounting="cluster",docker="cluster",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 0
elasticsearch_breaker_triggered_for_breaker{accounting="in_flight_requests",cluster="docker",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 0
elasticsearch_breaker_triggered_for_breaker{accounting="request",cluster="docker",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 0
elasticsearch_breaker_triggered_for_breaker{accounting="request",cluster="docker",es_client_node="true",es_data_node="true",es_ingest_node="true",es_master_node="true",host="192.168.52.237",name="k8s-elasticsearch-98b6d44f-qy9b7"} 0
# HELP elasticsearch_cluster_health_active_primary_shards
# TYPE elasticsearch_cluster_health_active_primary_shards gauge
elasticsearch_cluster_health_active_primary_shards{cluster="docker"} 0
# HELP elasticsearch_cluster_health_active_shards
# TYPE elasticsearch_cluster_health_active_shards gauge
elasticsearch_cluster_health_active_shards Aggregate total of all shards across all indices, which includes replica shards.
# HELP elasticsearch_cluster_health_active_shards{cluster="docker"} 0
# HELP elasticsearch_cluster_health_delay_unassigned_shards
# TYPE elasticsearch_cluster_health_delay_unassigned_shards gauge
elasticsearch_cluster_health_delay_unassigned_shards{cluster="docker"} 0
# HELP elasticsearch_cluster_health_initializing_shards
# TYPE elasticsearch_cluster_health_initializing_shards gauge
elasticsearch_cluster_health_initializing_shards Count of shards that are being freshly created.
# HELP elasticsearch_cluster_health_initiating_shards
# TYPE elasticsearch_cluster_health_initiating_shards gauge
elasticsearch_cluster_health_initiating_shards{cluster="docker"} 0
# HELP elasticsearch_cluster_health_jvm_gc_failures
# TYPE elasticsearch_cluster_health_jvm_gc_failures counter
elasticsearch_cluster_health_jvm_gc_failures{cluster="docker"} 0
# HELP elasticsearch_cluster_health_jvm_gc_failures
# TYPE elasticsearch_cluster_health_jvm_gc_failures counter
elasticsearch_cluster_health_jvm_gc_failures{cluster="docker"} 0
# HELP elasticsearch_cluster_health_number_of_data_nodes
# TYPE elasticsearch_cluster_health_number_of_data_nodes gauge
elasticsearch_cluster_health_number_of_data_nodes{cluster="docker"} 1
# HELP elasticsearch_cluster_health_number_of_in_flight_fetch
# TYPE elasticsearch_cluster_health_number_of_in_flight_fetch gauge
elasticsearch_cluster_health_number_of_in_flight_fetch The number of ongoing shard info requests.
# HELP elasticsearch_cluster_health_number_of_in_flight_fetch
# TYPE elasticsearch_cluster_health_number_of_in_flight_fetch gauge
elasticsearch_cluster_health_number_of_in_flight_fetch{cluster="docker"} 0
# HELP elasticsearch_cluster_health_number_of_nodes
# TYPE elasticsearch_cluster_health_number_of_nodes gauge
elasticsearch_cluster_health_number_of_nodes{cluster="docker"} 1
# HELP elasticsearch_cluster_health_number_of_pending_tasks
# TYPE elasticsearch_cluster_health_number_of_pending_tasks gauge
elasticsearch_cluster_health_number_of_pending_tasks{cluster="docker"} 0
# HELP elasticsearch_cluster_health_number_of_pending_tasks
# TYPE elasticsearch_cluster_health_number_of_pending_tasks gauge
elasticsearch_cluster_health_number_of_pending_tasks{cluster="docker"} 0
# HELP elasticsearch_cluster_health_relocating_shards
# TYPE elasticsearch_cluster_health_relocating_shards gauge
elasticsearch_cluster_health_relocating_shards{cluster="docker"} 0
# HELP elasticsearch_cluster_health_resize_shards
# TYPE elasticsearch_cluster_health_resize_shards gauge
elasticsearch_cluster_health_resize_shards{cluster="docker"} 0
# HELP elasticsearch_cluster_health_status
# TYPE elasticsearch_cluster_health_status gauge
elasticsearch_cluster_health_status{cluster="docker",color="green"} 1
elasticsearch_cluster_health_status{cluster="docker",color="yellow"} 0
elasticsearch_cluster_health_status{cluster="docker",color="red"} 0
# HELP elasticsearch_cluster_health_task_max_waiting_in_queue_millis
# TYPE elasticsearch_cluster_health_task_max_waiting_in_queue_millis gauge
elasticsearch_cluster_health_task_max_waiting_in_queue_millis{cluster="docker"} 0
# HELP elasticsearch_cluster_health_task_max_waiting_in_queue_millis
# TYPE elasticsearch_cluster_health_task_max_waiting_in_queue_millis gauge
elasticsearch_cluster_health_task_max_waiting_in_queue_millis{cluster="docker"} 0
# HELP elasticsearch_cluster_health_total_recuper
# TYPE elasticsearch_cluster_health_total_recuper counter
elasticsearch_cluster_health_total_recuper{cluster="docker"} 0
# HELP elasticsearch_cluster_health_unassigned_shards
# TYPE elasticsearch_cluster_health_unassigned_shards gauge
elasticsearch_cluster_health_unassigned_shards{cluster="docker"} 0
# HELP elasticsearch_cluster_health_unassigned_shards
# TYPE elasticsearch_cluster_health_unassigned_shards gauge
elasticsearch_cluster_health_unassigned_shards{cluster="docker"} 0
```

- In the instance list, choose **More > Remote Login** in the **Operation** column and run the following command:
`curl http://localhost:9114/metric`

----End

Collecting Service Data of the CCE Cluster

Add PodMonitor to configure a collection rule for monitoring the service data of applications deployed in the CCE cluster.

NOTE

In the following example, metrics are collected every 30s. Therefore, you can check the reported metrics on the AOM page about 30s later.

```
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: elasticSearch-exporter
  namespace: default
spec:
  namespaceSelector: # Select the namespace where Exporter is deployed.
    matchNames:
      - default # Namespace where Exporter is located.
  podMetricsEndpoints:
    - interval: 30s # Set the metric collection period.
      path: /metrics # Enter the path corresponding to Prometheus Exporter. Default: /metrics.
      port: metric-port # Enter the name of "ports" in the YAML file corresponding to Prometheus Exporter.
      selector: # Enter the label of the target Exporter pod.
        matchLabels:
          k8s-app: elasticSearch-exporter
```

Verifying that Metrics Can Be Reported to AOM

Step 1 Log in to the AOM 2.0 console.

Step 2 In the navigation pane on the left, choose **Prometheus Monitoring > Instances**.

- Step 3** Click the Prometheus instance connected to the CCE cluster. The instance details page is displayed.
- Step 4** On the **Metrics** tab page of the **Metric Management** page, select your target cluster.
- Step 5** Select job `{namespace}/elasticsearch-exporter` to query custom metrics starting with `elasticsearch`.
- End

Setting a Dashboard and Alarm Rule on AOM

By setting a dashboard, you can monitor CCE cluster data on the same screen. By setting an alarm rule, you can detect cluster faults and implement warning in a timely manner.

- Setting a dashboard
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Dashboard**. On the displayed page, click **Add Dashboard** to add a dashboard. For details, see [Creating a Dashboard](#).
 - c. On the **Dashboard** page, select a Prometheus instance for CCE and click **Add Graph**. For details, see [Adding a Graph to a Dashboard](#).
- Setting an alarm rule
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Alarm Management > Alarm Rules**.
 - c. On the **Metric/Event Alarm Rules** tab page, click **Create** to create an alarm rule. For details, see [Creating a Metric Alarm Rule](#).

6.7 Connecting Redis Exporter

Application Scenario

When using Redis, you need to monitor Redis running and locate their faults in a timely manner. The Prometheus monitoring function monitors Redis running using Exporter in the CCE container scenario. This section describes how to monitor Redis.

NOTE

You are advised to use CCE for unified Exporter management.

Prerequisites

- A CCE cluster has been created and Redis has been installed.
- Your service has been connected for Prometheus monitoring and a CCE cluster has also been connected. For details, see [Prometheus Instance for CCE](#).
- You have uploaded the `redis_exporter` image to Software Repository for Container (SWR). For details, see [Uploading an Image Through a Container Engine Client](#).

Deploying Redis Exporter

Step 1 Log in to the CCE console.

Step 2 Click the connected cluster. The cluster management page is displayed.

Step 3 Perform the following operations to deploy Exporter:

1. In the navigation pane, choose **ConfigMaps and Secrets**. Switch to the **Secrets** tab. Then click **Create from YAML** in the upper right corner of the page. The following shows a YAML configuration example:

```
apiVersion: v1
kind: Secret
metadata:
  name: redis-secret-test
  namespace: default # Must be the same as the namespace where Exporter is deployed.
type: Opaque
stringData:
  password: redis123 # Redis password.
```

NOTE

- The password has been encrypted based on Opaque requirements.
 - For details about how to configure a secret, see [Creating a Secret](#).
2. Deploy Redis Exporter.

In the navigation pane, choose **Workloads**. On the displayed page, click the **Deployments** tab, click **Create from YAML** in the upper right corner, and select a namespace. You can deploy Exporter through the console or using a YAML file. The following shows a YAML configuration example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: redis-exporter # Change the value based on service requirements. You are advised to add
the Redis instance information, for example, crs-66e112fp-redis-exporter.
  name: redis-exporter # Change the value based on service requirements. You are advised to add the
Redis instance information, for example, crs-66e112fp-redis-exporter.
  namespace: default #Select a proper namespace to deploy Exporter. If no namespace is available,
create one.
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: redis-exporter # Change the name based on service requirements. You are advised to
add the Redis instance information, for example, crs-66e112fp-redis-exporter.
  template:
    metadata:
      labels:
        k8s-app: redis-exporter # Change the name based on service requirements. You are advised to
add the Redis instance information, for example, crs-66e112fp-redis-exporter.
    spec:
      containers:
        - env:
            - name: REDIS_ADDR
              value: 120.46.215.4:30379 # IP address:port number of Redis
            - name: REDIS_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: redis-secret-test # Secret name specified in the previous step.
                  key: password # Secret key specified in the previous step.
          image: swr.cn-north-4.myhuaweicloud.com/mall-swarm-demo/redis-exporter:v1.32.0 # Replace
the value with the address of the image you uploaded to SWR.
          imagePullPolicy: IfNotPresent
          name: redis-exporter
        ports:
```

```
- containerPort: 9121
  name: metric-port # Required when you configure a collection task.
  securityContext:
    privileged: false
  terminationMessagePath: /dev/termination-log
  terminationMessagePolicy: File
  dnsPolicy: ClusterFirst
  imagePullSecrets:
  - name: default-secret
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  terminationGracePeriodSeconds: 30
---
apiVersion: v1
kind: Service
metadata:
  name: redis-exporter
  name-space: default # Must be the same as the namespace where Exporter is deployed.
spec:
  type: NodePort
  selector:
    k8s-app: redis-exporter
  ports:
    - protocol: TCP
      nodePort: 30378
      port: 9121
      targetPort: 9121
```

 NOTE

For more details about Exporter parameters, see [redis_exporter](#).

3. Check whether Redis Exporter is successfully deployed.
 - a. On the **Deployments** tab page, click the Deployment created in [Step 3.2](#). In the pod list, choose **More > View Logs** in the **Operation** column. The Exporter is successfully started and its access address is exposed.
 - b. Perform verification using one of the following methods:
 - Log in to a cluster node and run either of the following commands:
`curl http://{Cluster IP address}:9121/metrics`
`curl http://{Private IP address of any node in the cluster}:30378/metrics`
 - Access `http://{Public IP address of any node in the cluster}:30378/metrics`.
If no data is obtained, check whether the values of "REDIS_ADDR" and "REDIS_PASSWORD" in the YAML file set during [Redis Exporter deployment](#) are correct. The following shows an example:

Figure 6-10 Accessing a cluster node

```

← → C ▲ 30378/metrics

# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 7
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.17.3"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 962888
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 962888
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 4236
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 178
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 0
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 4.067e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 962888
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 1.769472e+06
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 1.998848e+06
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 4037
# HELP go_memstats_heap_released_bytes Number of heap bytes released to OS.
# TYPE go_memstats_heap_released_bytes gauge
go_memstats_heap_released_bytes 1.769472e+06

```

- In the instance list, choose **More > Remote Login** in the **Operation** column and run the following command:
`curl http://localhost:9121/metrics`

Figure 6-11 Executing the command

```

redis-exporter      nodeport      10.241.222.95      <none>      9121:30378/ILP      56
user@anisfyg9ulitku8-machine:~$ curl http://localhost:30378/metrics
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 8
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.17.3"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.029288e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 2.029288e+06
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 4236
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 384
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 0
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 4.09784e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge

```

----End

Adding a Collection Task

Add PodMonitor to configure a collection rule for monitoring the service data of applications deployed in the CCE cluster.

NOTE

In the following example, metrics are collected every 30s. Therefore, you can check the reported metrics on the AOM page about 30s later.

```
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: redis-exporter
  namespace: default
spec:
  namespaceSelector: # Select the namespace where the target Exporter pod is located.
  matchNames:
    - default # Namespace where Exporter is located.
  podMetricsEndpoints:
    - interval: 30s # Set the metric collection period.
      path: /metrics # Enter the path corresponding to Prometheus Exporter. Default: /metrics.
      port: metric-port# Enter the name of "ports" in the YAML file corresponding to Prometheus Exporter.
      selector: # Enter the label of the target Exporter pod.
      matchLabels:
        k8s-app: redis-exporter
```

Verifying that Metrics Can Be Reported to AOM

- Step 1** Log in to the AOM 2.0 console.
- Step 2** In the navigation pane on the left, choose **Prometheus Monitoring > Instances**.
- Step 3** Click the Prometheus instance connected to the CCE cluster. The instance details page is displayed.
- Step 4** On the **Metrics** tab page of the **Metric Management** page, select your target cluster.
- Step 5** Enter **redis** in the search box to search. If metrics starting with **redis** are displayed, the metrics are successfully connected to AOM.

----End

Setting a Dashboard and Alarm Rule on AOM

By setting a dashboard, you can monitor CCE cluster data on the same screen. By setting an alarm rule, you can detect cluster faults and implement warning in a timely manner.

- Setting a dashboard
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Dashboard**. On the displayed page, click **Add Dashboard** to add a dashboard. For details, see [Creating a Dashboard](#).
 - c. On the **Dashboard** page, select a Prometheus instance for CCE and click **Add Graph**. For details, see [Adding a Graph to a Dashboard](#).
- Setting an alarm rule
 - a. Log in to the AOM 2.0 console.
 - b. In the navigation pane, choose **Alarm Management > Alarm Rules**.
 - c. On the **Metric/Event Alarm Rules** tab page, click **Create** to create an alarm rule. For details, see [Creating a Metric Alarm Rule](#).

6.8 Connecting Other Exporters

Application Scenario

Guidance has been provided for connecting common Exporters. AOM is compatible with the native Prometheus, so you can also connect other Exporters in the community.

Methods

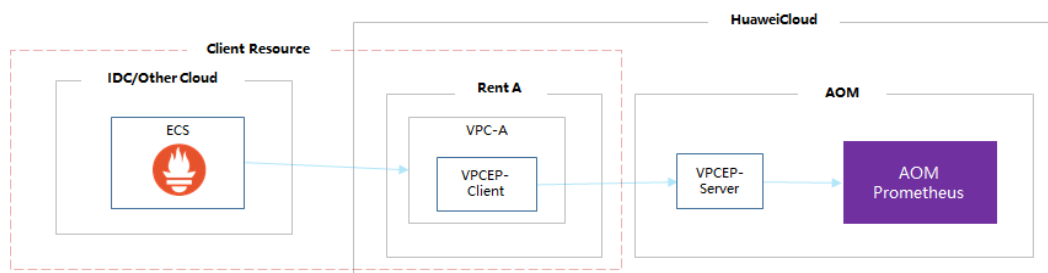
Customize dashboards or use either of the following methods to integrate basic components for monitoring:

1. [Integrating Exporters in the open-source community](#)
2. Instructions in [connecting common self-built middleware in the container scenario](#)

7 Interconnecting Third-Party/IDC/Huawei Cloud Cross-Region Self-Built Prometheus with AOM Prometheus Instances

Background

It is common for cloud users to interconnect IDC/third-party self-built Prometheus with AOM Prometheus instances.



Scenario

You need to [configure a VPC endpoint](#) first. If you have an ECS on Huawei Cloud, perform steps [2](#) and [3](#) to verify the network connectivity as required. Then, [access the AOM domain name through Direct Connect](#) to interconnect self-built Prometheus with an AOM Prometheus instance.

Precaution

This function is available only in **CN North-Beijing4** and **CN East-Qingdao**.

Step 1: Configure a VPC Endpoint

For example, in **CN North-Beijing4**, purchase a VPC endpoint according to the following procedure.

Step 1 Log in to the VPC Endpoint console.

Step 2 In the navigation pane on the left, choose **VPC Endpoint > VPC Endpoints**.

Step 3 Click **Buy VPC Endpoint** and set parameters as required.

1. **Region:** Select **CN North-Beijing4**.
2. **Service Category:** Select **Find a service by name**.
3. **VPC Endpoint Service Name:** Enter **cn-north-4.aom-access.df1ac4a2-7088-4cbe-990f-97ec3e121269** and click **Verify**. For the service names in other regions, see [Table 7-1](#).
 - a. If "Service name found" is displayed, proceed with subsequent operations.
 - b. If "Service not found" is displayed, check whether the region is the same as that of the VPC endpoint service or whether the entered service name is correct.

Table 7-1 VPC endpoint service names

Region	Service Name
CN North-Beijing4	cn-north-4.aom-access.df1ac4a2-7088-4cbe-990f-97ec3e121269
CN East-Qingdao	cn-east-5.aom-access.bf610bc3-24b5-43fa-a6ae-74d64d601817

4. **VPC/Subnet:** Set the parameters by referring to [Buying a VPC Endpoint for Accessing Interface VPC Endpoint Services](#).

 **NOTE**

The VPC must be the same as that of the purchased ECS.

Step 4 Confirm the configuration and click **Next**.

- If the configuration is correct, click **Submit**.
- If any parameter is incorrect, click **Previous** to modify it as needed, and then click **Submit**.

----End

(Optional) Step 2: Check the ECS Security Group Configuration

Verify the connectivity between AOM and ECS.

Step 1 Log in to the ECS console.

Step 2 In the navigation pane, choose **Elastic Cloud Server > Elastic Cloud Server**.

Step 3 Click the target ECS. The **Summary** tab page of the ECS is displayed.

Step 4 Check whether the VPC of the ECS is the same as the VPC selected when you purchase the endpoint in [step 1](#).

- If yes, go to the next step.
- If no, modify the configuration to ensure that they are the same.

Step 5 On the **Summary** page of the ECS, click **Security Groups**. On the **Outbound Rules** tab page of the security group, check the values of **Protocol & Port** and **Destination**.

Ensure that **Protocol & Port** is set to **All** and **Destination** is set to **0.0.0.0/0**.

----End

(Optional) Step 3: Verify Connectivity

Log in to the ECS and run the **curl** command to call the API to verify the connectivity.

- Step 1** Log in to the ECS console.
- Step 2** In the navigation pane, choose **Elastic Cloud Server > Elastic Cloud Server**.
- Step 3** Click **Remote Login** in the **Operation** column that contains the target ECS to remotely **log in to the ECS**.
- Step 4** Run the following command to access AOM. **Figure 7-1** shows how to access AOM in **CN North-Beijing4**.

```
curl aom-access.cn-north-4.myhuaweicloud.com:8443
```

Figure 7-1 Domain name and port number for accessing AOM

```
[root@aom-test-hmz ~]# curl -kv -X POST "https://aom-access.cn-north-4.myhuaweicloud.com:8443/v1/21a51c0ce8324aef8a4ba67548d9649e/ams/report/metricdata" -H "Content-Type: application/json" -H "X-Sdk-Date: 20240624T030230Z" -H "host: aom-access.cn-north-4.myhuaweicloud.com:8443" -H "authorization: SDK-HMAC-SHA256 Access=GG21B8KJQFEBNPNL6ZU, SignedHeaders=content-type;host;x-sdk-date, Signature=2d495fa876989b8d55fc60e47bef9ffe73da704306fa43ad3b744780d444e847" -d '{{"metric":{"namespace":"NOPSAS_ESC","dimensions":{"name":"instance_id","value":"instance-101"}},"values":{"unit":"percent","metric_name":"my_cpu_util","type":"int","value":35},"collect_time":1719198092000}}'
* About to connect() to aom-access.cn-north-4.myhuaweicloud.com port 8443 (#0)
* Trying 192.168.0.31...
* Connected to aom-access.cn-north-4.myhuaweicloud.com (192.168.0.31) port 8443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* NSS: client certificate not found (nickname not specified)
* SSL connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate:
*   subject: OID.1.1.1.4=684fc87a79d7471583df22e62a7da95b,OID.1.1.1.3=op_svc_mgmt_container1:797c8fc3c4dd4f6985d5f42dc6bf70bd,OID.1.1.1.2=cn-north-4:684fc87a79d7471583df22e62a7da95b:op_cfe_kubelet,OID.1.1.1.1=op_svc_mgmt_paas,CN=apm OSS3.0 CA,OU=OSS & Service Tools Dept,0="Huawei Technologies Co., Ltd",L=ShenZhen,ST=GuangDong,C=CN
*   start date: Jun 06 16:00:00 2022 GMT
*   expire date: Jun 06 16:00:00 2052 GMT
*   common name: apm OSS3.0 CA
*   issuer: CN=OSS3.0 CA,OU=OSS & Service Tools Dept,0="Huawei Technologies Co., Ltd",L=ShenZhen,ST=GuangDong,C=CN
> POST /v1/21a51c0ce8324aef8a4ba67548d9649e/ams/report/metricdata HTTP/1.1
> User-Agent: curl/7.29.0
> Accept: */*
> Content-Type: application/json
> X-Sdk-Date: 20240624T030230Z
> host: aom-access.cn-north-4.myhuaweicloud.com:8443
> authorization: SDK-HMAC-SHA256 Access=GG21B8KJQFEBNPNL6ZU, SignedHeaders=content-type;host;x-sdk-date, Signature=2d495fa876989b8d55fc60e47bef9ffe73da704306fa43ad3b744780d444e847
> Content-Length: 211
>
* upload completely sent off: 211 out of 211 bytes
< HTTP/1.1 200 OK
< Content-Length: 59
< Content-Type: application/json
< Date: Mon, 24 Jun 2024 03:03:56 GMT
<
* Connection #0 to host aom-access.cn-north-4.myhuaweicloud.com left intact
{"errorCode":"SUCSTG_AMS_2000000","errorMessage":"success"}[root@aom-test-hmz ~]#
```

----End

Step 4: Access the AOM Domain Name Through Direct Connect

A self-built host can access the VPC endpoint based on the endpoint IP address and then access the domain name interconnected with the VPC endpoint. Alternatively, configure domain name resolution on the host and then access AOM through an API. The following describes how to configure domain name resolution to access AOM.

- Step 1** Take CentOS as an example. Run the following command on the self-built host:

```
sudo vi /etc/hosts
```

Step 2 Configure domain name resolution. For example, add the following configuration:

```
192.168.0.31 aom-access.cn-north-4.myhuaweicloud.com
```

192.168.0.31 is the IP address of the VPC endpoint, and **aom-access.cn-north-4.myhuaweicloud.com** is the domain name of AOM.

----End