Alibaba Cloud Application Real-time Monitoring Service

Prometheus Monitoring

Issue: 20200511
Legal disclaimer

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

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

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

3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.
5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please contact Alibaba Cloud directly if you discover any errors in this document.
# Document conventions

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

Legal disclaimer.............................................................................................................. I
Document conventions..................................................................................................... I
1 Prometheus monitoring overview .............................................................................. 1
2 Use ARMS Prometheus Monitoring ........................................................................... 6
3 View Prometheus Monitoring metrics ........................................................................ 8
4 Preconditioning............................................................................................................ 11
   4.1 Tutorial.................................................................................................................. 11
      4.1.1 Overview....................................................................................................... 11
      4.1.2 Use ARMS Prometheus Monitoring to monitor JVM data............................ 11
      4.1.3 Use ARMS Prometheus Monitoring to monitor MySQL data....................... 19
      4.1.4 Use ARMS Prometheus Monitoring to monitor Go applications................ 24
      4.1.5 Use ARMS Prometheus Monitoring to monitor Redis data........................... 30
      4.1.6 Use ARMS Prometheus Monitoring to customize the Grafana dashboard................................................................. 35
      4.1.7 Use ARMS Prometheus Monitoring to monitor ZooKeeper......................... 39
5 Console functions....................................................................................................... 44
## 1 Prometheus monitoring overview

ARMS Prometheus fully integrates with the open-source Prometheus ecosystem, supports monitoring by a wide range of components, provides a variety of pre-available monitoring dashboards, and provides fully managed Prometheus services.

### What is Prometheus?

Prometheus is an open-source system monitoring and alarm framework inspired by Google’s Borg mon system. In 2012, former Google employees of SoundCloud created Prometheus and developed it as an open source project in the community. On 2015, the project was officially released. In 2016, Prometheus joined the Cloud Native Computing Foundation (Cloud Native Computing Foundation), Becoming a project second only to Kubernetes in popularity.

Prometheus has the following features:

- Multi-Dimensional Data Model (Key-Value pairs based on time series)
- Flexible query and aggregation language PromQL
- Local and distributed storage
- Collect time series data by using an HTTP push model
- Pushgateway (an optional Prometheus middleware) can be used to implement the Push mode.
- Discover target machines through dynamic service discovery or static configuration
- Supports multiple charts and data dashboards.

### Advantages of ARMS Prometheus monitoring and open-source Prometheus monitoring

Overall, compared with open-source Prometheus monitoring, ARMS Prometheus monitoring has the following advantages:

- More lightweight, more stable, and more accurate
- Unlimited data volume
- Fully compatible with the open-source ecosystem
- Cost saving

### More lightweight, more stable, and more accurate

- Compared with open-source Prometheus monitoring, the overall structure of ARMS Prometheus monitoring is more lightweight. You do not need to build your own
Prometheus monitoring system. Instead, you only need to install ARMS Prometheus monitoring agent PromAgent to start monitoring services.

- In terms of system stability, the open-source Prometheus monitoring generally occupies 16 GB ~ 128 GB memory, while ARMS Prometheus only uses 200 MB ~ 1 GB memory and 1 core CPU. Compared with open-source Prometheus monitoring, ARMS Prometheus monitoring is more stable.

- In terms of the accuracy of data capturing and writing, the open-source Prometheus monitoring only captures data once, and the discarding logic exists when the data is written to the storage component instantly. However, ARMS Prometheus retries capturing data five times and continuously writes the data to the storage component concurrently without the discard logic.

**Unlimited data volume**

- The maximum data collection capability of the open-source Prometheus monitoring system is millions of Metrics. The data collection capability of the ARMS Prometheus monitoring system can be scaled based on the number of K8s replicas to break down collection tasks in a balanced manner.
• The upper limit of data storage capability of the open-source Prometheus monitoring system is limited by the size of the local disk, while the storage capability of the central cloud storage service of ARMS Prometheus monitoring system is theoretically unlimited.

**Fully compatible with the open-source ecosystem**

• ARMS Prometheus is fully compatible with the clients and query languages in the open-source ecosystem of Prometheus, as well as the collection rules and usage value in the open-source ecosystem.

• ARMS Prometheus monitoring is compatible with three mainstream collection rules, including standard open source prometheus.yaml Collection rule configuration file, collection rule ServiceMonitor suitable for custom K8s monitoring, and default collection rule Annotation. Compared with open-source Prometheus monitoring, ARMS Prometheus monitoring can be used without restarting. prometheus.yaml Configuration files update collection rules dynamically. In Deployment You do not need to write multiple lines of code in the file. You only need to add the following three annotations:

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

• ARMS Prometheus is compatible with Grafana. By configuring the Prometheus http api url, you can implement multi-tenant isolation of data sources and multi-tenant isolation of the Grafana dashboard in Grafana. ARMS Prometheus is also compatible with the Explore data debugging module of Grafana.

• ARMS Prometheus is compatible with the http api module of open-source Prometheus monitoring. It supports three standard data query interfaces: query, query_range, and labelValues. You can add/userId/clusterId/regionId/ This group of IDs achieves multi-tenant isolation.
• Although ARMS Prometheus monitoring uses the existing alerting system of ARMS, it is fully compatible with the alerting rule PromQL of open-source Prometheus monitoring.

**Cost saving**

• ARMS Prometheus supports K8s monitoring by default. After you install the default K8s monitoring, ARMS Prometheus automatically creates default reporter, collection rule, Grafana dashboard, and ARMS alarms for you. Your time cost can be reduced from about 3 days when you used open-source Prometheus to monitor K8s to about 10 minutes.

• ARMS Prometheus supports open-source component monitoring. You only need to enter the AccessKeyId and AccessKeySecret of your Alibaba Cloud account, as well as the account and password for the RDS and Redis components. ARMS Prometheus automatically generates an Exporter for these components by default. and create a
default widget dashboard. Your time cost can be reduced from about 7 days when you used open source Prometheus to monitor open source components to about 3 minutes.

- ARMS Prometheus supports one-click installation, one-click removal, and debugging of Prometheus monitoring through health check. Your time cost can be reduced from about one day when you used open-source Prometheus monitoring to about three minutes.
2 Use ARMS Prometheus Monitoring

For Container Service Kubernetes clusters, you can install the Prometheus agent in Application Real-Time Monitoring Service (ARMS) with one click. Then, you can monitor multiple performance metrics of hosts and Kubernetes clusters through the predefined dashboards.

Prerequisites

- #unique_5: Create a Container Service Kubernetes cluster.
- #unique_6

Install the Prometheus agent

1. Log on to ARMS console.
2. In the left-side navigation pane, click Prometheus Monitoring.
   
   On the Prometheus monitoring page, all the Container Service Kubernetes clusters under your Alibaba Cloud account are displayed.
3. On the Prometheus monitoring page, click Installation in the Actions column. In the Tips dialog box, click OK.
   
   During agent installation, the installation progress is displayed by a progress bar and is also displayed as a percentage in the Actions column.
   
   After the Prometheus agent is installed, all installed agents are displayed in the Has installation plugins column.

   ![Note]

   If the installation fails, retry until it is successful. If the installation still fails after several attempts, contact us at our DingTalk account 23148410.

Access Prometheus dashboards

1. In the left-side navigation pane, click Prometheus Monitoring.
2. On the Prometheus monitoring page, click a link in the Has installation plugins column to access the monitoring dashboard in a new browser window.

Uninstall the Prometheus agent

To disable Prometheus Monitoring for a Kubernetes cluster, uninstall the Prometheus agent as follows:
1. In the left-side navigation pane, click **Prometheus Monitoring**.

2. On the **Prometheus monitoring** page, click **Unloading** in the **Actions** column. In the **Tips** dialog box, click **OK**.

   After the agent is uninstalled, the agent is no longer displayed in the **Has installation plugins** column.

**Next steps**

- View Prometheus Monitoring metrics

**More information**

- Prometheus monitoring overview
3 View Prometheus Monitoring metrics

The preset monitoring dashboards provided by Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS) contain the Kubernetes cluster overview, Kubernetes deployment, pods, and host details. You can view a wide range of Prometheus metrics through these dashboards, and modify the attributes such as the time range and refresh frequency of the dashboard data as needed.

Prerequisites

Install the Prometheus agent

Open a monitoring dashboard

After installing the Prometheus agent, you can open the preset monitoring dashboards on the Prometheus monitoring page.

1. Log on to the ARMS console.
2. In the left-side navigation pane, click Prometheus Monitoring.
   On the Prometheus monitoring page, all the Container Service Kubernetes clusters under your Alibaba Cloud account are displayed.
3. On the Prometheus monitoring page, click a link in the Has installation plugins column to access the monitoring dashboard in a new browser window.

View monitoring metrics

You can view monitoring metrics on the following preset monitoring dashboards:

- Kubernetes cluster overview dashboard
  The dashboard displays the following monitoring metrics:
  - Total Usage: Cluster CPU Usage, Cluster Memory Usage, and Cluster Filesystem Usage
  - CPU Usage: Pod CPU Usage and Processes CPU Usage
  - Memory Usage: Pod Memory Usage and Processes Memory Usage
  - Network Info: Network I/O Pressure, Pod Network I/O, and Processes Network I/O
• Kubernetes deployment dashboard

The dashboard displays the following monitoring metrics:

- Overview: Deployment CPU usage, Deployment Memory Usage, and Unavailable Replicas
- Detail: CPU Usage, Memory Usage, and Processes Network I/O

• Pod dashboard

The dashboard displays the following monitoring metrics:

- Pod Info: Pod IP Address, Pod Status, Pod Container, and Container restarts
- Total Usage: Pod Memory Usage and Pod CPU Usage
- CPU Usage: Pod CPU Usage and Processes CPU Usage
- Memory Usage: Pod Memory Usage and Processes Memory Usage
- Network Info: Network I/O Pressure, Pod Network I/O, and Processes Network I/O

• Host details dashboard

The dashboard displays the following monitoring metrics:

- CPU Info: CPU Cores, CPU Usage, and CPU Iowait
- Memory Info: Total Memory and Memory Usage
- Disk Info: Total Disk Space, Disk I/O Read/Write Time, and Disk Read/Write Rate
- Network Info: Network Traffic and TCP Connections

Common operations on dashboards

The following figure shows the common operations on dashboards.

1. Switch dashboards

   The drop-down list displays the name of the currently viewed dashboard and can be used to switch to another dashboard. You can enter a name in the search bar on the top to search for a dashboard, or use Filter to search for a dashboard with a specified tag from the drop-down list.

2. Set the time interval and refresh frequency

   After clicking the icon, you can select a predefined time interval for monitoring data, such as the last 5 minutes, last 12 hours, or last 30 days. You can also customize an absolute time interval by setting the start time and end time. In addition, you can set the dashboard refresh frequency.
3. Expand the time interval

When you click the expand button each time, the time interval is expanded twice, with the start time earlier and end time later equally. Assume that the selected time interval is the last 10 minutes. After you click the button, the start time is 5 minutes earlier while the end time is 5 minutes later.

4. Manually refresh the dashboard

Click this button to refresh the monitoring data of all panels on the current dashboard.

5. Filter monitoring data

Select an option from the drop-down list to filter the monitoring data displayed on the current dashboard.

**Common operations on the dashboard panel**

Click the drop-down menu at the top of the panel to perform the following operations:

- View the current panel in full screen: Click **View** or press the shortcut key V. Press the shortcut key V again or press Esc to exit the full screen.
- Share the current panel: Click **Share** or press P and S in sequence to access the Share dialog box and acquire the sharing link, embedded link, or snapshot link of the current panel.
- Acquire the JSON code of the current panel: Choose **More > Panel JSON** and then copy the JSON code in the **JSON** dialog box.
- Export data on the current panel into a CSV file: Choose **More > Export CSV**. In the **Export CSV** dialog box, set the format and export the file.
- Toggle legends: Choose **More > Toggle legend** or press P and L in sequence to toggle the legends.

**Prometheus monitoring overview**

**Install the Prometheus agent**
4 Preconditioning

4.1 Tutorial

4.1.1 Overview

The tutorial series introduces how to monitor your JVM, MySQL, Go, and Redis applications or components, and display monitoring data on the Grafana dashboard by using Prometheus Monitoring of Application Real-Time Application Service (ARMS).

Use ARMS Prometheus Monitoring to monitor JVM data
This tutorial describes how to expose JVM data by instrumentation points in applications, capture JVM data by using Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS), and display JVM data on the ARMS Prometheus Grafana dashboard, finally to monitor JVM data with ARMS Prometheus Monitoring.

Use ARMS Prometheus Monitoring to monitor MySQL data
Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS) allows you to monitor MySQL by capturing MySQL data and displaying the captured data on the ARMS Prometheus Grafana dashboard.

Use ARMS Prometheus Monitoring to monitor Go applications
Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS) allows you to monitor the running status of Go applications deployed in Alibaba Cloud Container Service Kubernetes clusters, view various monitoring data on the dashboard, and configure monitoring jobs and alerts. This tutorial describes how to expose data by using instrumentation points in Go applications, capture data by using ARMS Prometheus Monitoring, and display data on the ARMS Prometheus Grafana dashboard, to ultimately monitor Go applications with ARMS Prometheus Monitoring.

Use ARMS Prometheus Monitoring to monitor Redis data
Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS) allows you to monitor Redis by capturing Redis data and displaying the captured data on the ARMS Prometheus Grafana dashboard.

4.1.2 Use ARMS Prometheus Monitoring to monitor JVM data
This tutorial describes how to expose JVM data by instrumentation points in applications, capture JVM data by using Prometheus Monitoring of Application Real-Time Monitoring
Application Real-time Monitoring Service  Prometheus Monitoring / 4 Preconditioning

Service (ARMS), and display JVM data on the ARMS Prometheus Grafana dashboard, finally to monitor JVM data with ARMS Prometheus Monitoring.

Prerequisites

- Download the demo project
- #unique_5
- #unique_15

Context

The following figure shows the workflow.

![Workflow Diagram](image)

**Step 1: Expose JVM data by instrumentation points in applications**

Use JVM Exporter to expose JVM data in applications.

1. Add the following information to the pom.xml file:

```xml
<dependency>
   <groupId>io.prometheus</groupId>
   <artifactId>simpleclient_hotspot</artifactId>
   <version>0.6.0</version>
</dependency>
```

2. Add the JVM Exporter dependency in the location where initialization is available.

   For example, add the dependency in the `src/main/java/com/monitise/prometheus_dmo/DemoController.java` file of the demo project.

```java
@PostConstruct
public void initJvmExporter() {
   io.prometheus.client.hotspot.DefaultExports.initialize();
```

```java
management.port: 8081
endpoints.prometheus.path: prometheus-metrics
```

Enable the HTTP port.

```java
@SpringBootApplication
@EnablePrometheusEndpoint
@EnableSpringBootMetricsCollector
public class PrometheusDemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(PrometheusDemoApplication.class, args);
    }
}
```

**Step 2: Deploy the application to the Container Service Kubernetes cluster**

Deploy the application to the Container Service Kubernetes cluster so that ARMS Prometheus Monitoring can monitor and capture the JVM data.

1. Run the following commands in the `buildDockerImage.sh` file by row:

```bash
mvn clean install -DskipTests
docker build -t <name of the local temporary Docker image>:<version of the local temporary Docker image> . --no-cache
sudo docker tag <name of the local temporary Docker image>:<version of the local temporary Docker image>:<registry domain name>/<namespace>/<image name>:<image version>
sudo docker push <registry domain name>/<namespace>/<image name>:<image version>
```

Example:

```bash
mvn clean install -DskipTests
docker build -t prometheus-demo:v0 . --no-cache
sudo docker tag prometheus-demo:v0 registry.cn-hangzhou.aliyuncs.com/fuling/prometheus-demo:v0
sudo docker push registry.cn-hangzhou.aliyuncs.com/fuling/prometheus-demo:v0
```

In this step, a Docker image named prometheus-demo is created, and the image is pushed to Alibaba Cloud Docker Registry.
2. In the left-side navigation pane, choose Clusters > Clusters. On the Clusters page, find the target cluster, and click Dashboard in the Actions column.

3. In the left-side navigation pane, choose Workloads > Deployments. In the upper-right corner, click CREATE, and enter the following information in the CREATE FROM TEXT INPUT section:

```
apiVersion: extensions/v1beta1
category: Deployment
metadata:
  name: prometheus-demo
spec:
  replicas: 2
  template:
    metadata:
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/path: '/prometheus-metrics'
        prometheus.io/port: '8081'
      labels:
        app: tomcat
  spec:
    containers:
    - name: tomcat
      imagePullPolicy: Always
      image: registry.cn-hangzhou.aliyuncs.com/fuling/promethues-demo:v0
      ports:
        - containerPort: 8080
          name: tomcat-normal
        - containerPort: 8081
          name: tomcat-monitor
```

In this step, the promethues-demo Docker image created in 1 is deployed to the Container Service Kubernetes cluster.
4. In the left-side navigation pane, choose **Discovery and Load Balancing > Services**. In the upper-right corner, click **CREATE**, and enter the following information in the **CREATE FROM TEXT INPUT** section:

```yaml
apiVersion: v1
class: Service
metadata:
  labels:
    app: tomcat
name: tomcat
namespace: default
spec:
  ports:
  - name: tomcat-normal
    port: 8080
    protocol: TCP
    targetPort: 8080
  - name: tomcat-monitor
    port: 8081
    protocol: TCP
    targetPort: 8081
type: NodePort
selector:
  app: tomcat
```

**Step 3: Configure ARMS Prometheus Monitoring to capture the JVM data**

Configure ARMS Prometheus Monitoring in the ARMS console to capture the JVM data.

1. In the left-side navigation pane, click **Prometheus Monitoring**.

2. On the top of the **Prometheus monitoring** page, select the region where the Container Service Kubernetes cluster is located. Find the target cluster, and click **Installation** in the **Actions** column.

3. After the ARMS Prometheus agent is installed, find the target cluster, and click **Settings** in the **Actions** column.

4. On the **Details** tab, click **Add ServiceMonitor**. In the **Add ServiceMonitor** dialog box, enter the following information:

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: tomcat-demo
  namespace: default
spec:
  endpoints:
  - interval: 30s
    name: tomcat-monitor
    path: /prometheus-metrics
    namespaceSelector:
```

*Issue: 20200511*
Step 4: Display the JVM data on the Grafana dashboard

Import the Grafana dashboard template in the ARMS console and specify the Container Service Kubernetes cluster where the Prometheus data source is located.

1. Go to Host Dashboard.

2. In the left-side navigation pane, choose + > Import, enter 10877 in the Grafana.com Dashboard field, and click Load.
3. On the **Import** page, set the following information and click **Import**.

![Import page](image)

- a) Enter a custom dashboard name in the **Name** field.
- b) Select your Container Service Kubernetes cluster from the **Folder** drop-down list.
- c) Select your Container Service Kubernetes cluster from the **Select a Prometheus data source** drop-down list.

After the configuration is complete, the ARMS Prometheus Grafana JVM dashboard appears, as shown in the following figure.
Step 5: Create an alert

1. You can select one of the two available methods to access the Create Alert page.
   - On the **New Dashboard** page of the ARMS Prometheus Grafana dashboard, click the icon to go to the ARMS Prometheus **Create Alert** dialog box.
   - In the left-side navigation pane of the console, choose **Alerts > Alert Policies**. On the **Alert Policies** page, choose **Create Alert > Prometheus** in the upper-right corner.

2. In the **Create Alert** dialog box, enter all required information and click **Save**.
   a) Enter Alert Name, for example, network receiving pressure alert.
   b) Select the corresponding **Cluster** of the Prometheus monitoring job.
   c) Set **Type** to **grafana**.
   d) Select the specific **dashboard** and **chart** to monitor.
   e) Set alert rules.
      
      A. Select **Meet All of the Following Criteria**.
      B. Edit the alert rule. For example, an alert is triggered when the value of N is 5 and the average value of network receiving bytes (MB) is at least 3.

      **Note:**
      
      A Grafana chart may contain data of curve A, curve B, and curve C. You can select one of them to monitor.

      **Notice:**
      
      The "$" symbol in the PromQL statement can lead to an error. You must delete the "=" symbol and the parameters on both sides of it in the statement containing the "$" symbol. For example, modify `sum (rate (container_network_receive_bytes_total{instance=~"^$HostIp.*"}[1m])) to `sum (rate (container_network_receive_bytes_total[1m]))

   f) Set Notification Mode. For example, select SMS.
   g) Set Notification Receiver. In the **Contact Groups** box, click the name of a contact group. If the contact group appears in the **Selected Groups** box, the setting is successful.

What's next
After the ARMS Prometheus Grafana JVM dashboard is configured, you can view Prometheus Monitoring metrics and customize the dashboard. For more information, see the following documents.

View Prometheus Monitoring metrics

Use ARMS Prometheus Monitoring to customize the Grafana dashboard

Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS) allows you to display monitoring data on the ARMS Prometheus Grafana dashboard. You can customize a Grafana dashboard or import the dashboard from the Grafana official website. This topic describes how to customize the Grafana dashboard to display monitoring data.

Create alarms

4.1.3 Use ARMS Prometheus Monitoring to monitor MySQL data

Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS) allows you to monitor MySQL by capturing MySQL data and displaying the captured data on the ARMS Prometheus Grafana dashboard.

Prerequisites

- #unique_5
- #unique_15

Context

The following figure shows the workflow.

Step 1: Capture MySQL data through an external application

Deploy an application by using the mysqld-exporter image officially provided by Prometheus to the Container Service Kubernetes cluster to capture MySQL data.
1. In the left-side navigation pane, choose **Clusters > Clusters**. On the **Clusters** page, find the target cluster, and click **Dashboard** in the **Actions** column.

2. In the left-side navigation pane, choose **Workloads > Deployments**. In the upper-right corner, click **CREATE**, and enter the following information in the **CREATE FROM TEXT INPUT** section:

   ```yaml
   apiVersion: extensions/v1beta1
   kind: Deployment
   metadata:
     name: mysqld-exporter
   spec:
     replicas: 1
     template:
       metadata:
         labels:
           app: mysqld-exporter
       spec:
         containers:
           - name: mysqld-exporter
             imagePullPolicy: Always
             env:
               - name: DATA_SOURCE_NAME
                 value: "<yourMySQLUsername>:<yourMySQLPassword>@(<IP>:<port>)/"
             image: prom/mysqld-exporter
             ports:
               - containerPort: 9104
             name: mysqld-exporter
   ```

   In this step, the mysqld-exporter application is deployed to the Container Service Kubernetes cluster.

3. In the left-side navigation pane, choose **Discovery and Load Balancing > Services**. In the upper-right corner, click **CREATE**, and enter the following information in the **CREATE FROM TEXT INPUT** section:

   ```yaml
   apiVersion: v1
   ```
Step 2: Configure ARMS Prometheus Monitoring to receive MySQL data

Configure ARMS Prometheus Monitoring in the ARMS console to receive MySQL data captured through external applications.

1. In the left-side navigation pane, click Prometheus Monitoring.

2. On the top of the Prometheus monitoring page, select the region where the Container Service Kubernetes cluster is located. Find the target cluster, and click Installation in the Actions column.

3. After the ARMS Prometheus agent is installed, find the target cluster, and click Settings in the Actions column.

4. On the Details tab, click Add ServiceMonitor. In the Add ServiceMonitor dialog box, enter the following information:

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app: mysqld-exporter
    release: p
    name: mysqld-exporter
spec:
  endpoints:
  - interval: 30s
    # The Mysqld Grafana template ID is 7362.
    # Enter the value of the Name field for Port of Prometheus Exporter in the service.
    yaml file.
    port: mysqld-exporter
    # Enter the path exposed in Prometheus Exporter code.
    path: /metrics
    namespaceSelector:
      any: true
      # The namespace of the NGINX Demo
    selector:
      matchLabels:
```
Step 3: Display the MySQL data on the Grafana dashboard

Import the Grafana dashboard template in the ARMS console and specify the Container Service Kubernetes cluster where the Prometheus data source is located.

1. Go to Host Dashboard.
2. In the left-side navigation pane, choose + > Import, enter 7362 in the Grafana.com Dashboard field, and click Load.

3. On the Import page, set the following information and click Import.

   a) Enter a custom dashboard name in the Name field.
   b) Select your Container Service Kubernetes cluster from the Folder drop-down list.
   c) Select your Container Service Kubernetes cluster from the prometheus drop-down list.

After the configuration is complete, the ARMS Prometheus Grafana MySQL dashboard appears, as shown in the following figure.
Step 4: Create an alert

1. You can select one of the two available methods to access the Create Alert page.
   - On the New Dashboard page of the ARMS Prometheus Grafana dashboard, click the icon to go to the ARMS Prometheus Create Alert dialog box.
   - In the left-side navigation pane of the console, choose Alerts > Alert Policies. On the Alert Policies page, choose Create Alert > Prometheus in the upper-right corner.

2. In the Create Alert dialog box, enter all required information and click Save.
   a) Enter Alert Name, for example, network receiving pressure alert.
   b) Select the corresponding Cluster of the Prometheus monitoring job.
   c) Set Type to grafana.
   d) Select the specific dashboard and chart to monitor.
   e) Set alert rules.
      
      A. Select Meet All of the Following Criteria.
      B. Edit the alert rule. For example, an alert is triggered when the value of N is 5 and the average value of network receiving bytes (MB) is at least 3.

      ![Note]
      A Grafana chart may contain data of curve A, curve B, and curve C. You can select one of them to monitor.

      ![Notice]
      The "$" symbol in the PromQL statement can lead to an error. You must delete the "=" symbol and the parameters on both sides of it in the statement containing the "$" symbol. For example, modify sum (rate (container_network_receive_bytes_total{instance=~"^$HostIp.*"}[1m])) to sum (rate (container_network_receive_bytes_total[1m]))

   f) Set Notification Mode. For example, select SMS.
   g) Set Notification Receiver. In the Contact Groups box, click the name of a contact group. If the contact group appears in the Selected Groups box, the setting is successful.

What's next
After the ARMS Prometheus Grafana MySQL dashboard is configured, you can view Prometheus Monitoring metrics and customize the dashboard. For more information, see the following documents.

**Related tasks**
- View Prometheus Monitoring metrics
- Use ARMS Prometheus Monitoring to customize the Grafana dashboard

Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS) allows you to display monitoring data on the ARMS Prometheus Grafana dashboard. You can customize a Grafana dashboard or import the dashboard from the Grafana official website. This topic describes how to customize the Grafana dashboard to display monitoring data.

**Create alarms**

**4.1.4 Use ARMS Prometheus Monitoring to monitor Go applications**

Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS) allows you to monitor the running status of Go applications deployed in Alibaba Cloud Container Service Kubernetes clusters, view various monitoring data on the dashboard, and configure monitoring jobs and alerts. This tutorial describes how to expose data by using instrumentation points in Go applications, capture data by using ARMS Prometheus Monitoring, and display data on the ARMS Prometheus Grafana dashboard, to ultimately monitor Go applications with ARMS Prometheus Monitoring.

**Prerequisites**

Ensure that you have completed the following operations:

- Download the demo project
- #unique_5
- #unique_15

**Context**

The following figure shows the workflow.

**Step 1: Expose application data by instrumentation points in Go applications**

Use Prometheus Exporter in Go applications to expose application data.

1. Import the monitoring package to your Go application.

```go
import "fmt"
```
2. Bind the monitoring interface to `promhttp.Handler()`.

```go
http.Handle(path, promhttp.Handler()) //Initializes an HTTP handler.
```

### Step 2: Deploy the application to the Container Service Kubernetes cluster

Deploy the application to the Container Service Kubernetes cluster so that ARMS Prometheus Monitoring can monitor and capture the application data.

1. Run the following commands in the `buildDockerImage.sh` file by row:

```bash
mvn clean install -DskipTests
docker build -t <name of the local temporary Docker image>:<version of the local temporary Docker image> . --no-cache
sudo docker tag <name of the local temporary Docker image>:<version of the local temporary Docker image> <registry domain name>/<namespace>/<image name>:<image version>
sudo docker push <registry domain name>/<namespace>/<image name>:<image version>
```

Example:

```bash
mvn clean install -DskipTests
docker build -t promethues-arms-aliyun-go-demo:v0.1 . --no-cache
sudo docker tag promethues-arms-aliyun-go-demo:v0.1 registry.cn-hangzhou.aliyuncs.com/fuling/promethues-arms-aliyun-go-demo-amd64:dev-v0.1
sudo docker push registry.cn-hangzhou.aliyuncs.com/fuling/promethues-demo:v0.1
```

In this step, a Docker image named `promethues-arms-aliyun-go-demo` is created, and the image is pushed to Alibaba Cloud Docker Registry.

2. In the left-side navigation pane, choose **Clusters > Clusters**. On the **Clusters** page, find the target cluster, and click **Dashboard** in the **Actions** column.
3. In the left-side navigation pane, choose **Workloads > Deployments**. In the upper-right corner, click **CREATE**, and enter the following information in the **CREATE FROM TEXT INPUT** section:

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: promethues-arms-aliyun-go-demo
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: promethues-arms-aliyun-go-demo
    spec:
      containers:
        - name: promethues-arms-aliyun-go-demo
          imagePullPolicy: Always
          image: registry.cn-hangzhou.aliyuncs.com/fuling/prometheus-arms-aliyun-go-demo-amd64:dev-v0.1
          ports:
            - containerPort: 8077
              name: arms-go-demo
```

In this step, the Docker image created in 1 is deployed to the Container Service Kubernetes cluster.

4. In the left-side navigation pane, choose **Discovery and Load Balancing > Services**. In the upper-right corner, click **CREATE**, and enter the following information in the **CREATE FROM TEXT INPUT** section:

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: promethues-arms-aliyun-go-demo
  name: promethues-arms-aliyun-go-demo
spec:
  ports:
    - name: arms-go-demo
      port: 8077
      protocol: TCP
      targetPort: 8077
      type: NodePort
  selector:
    app: promethues-arms-aliyun-go-demo
```

**Step 3: Configure ARMS Prometheus Monitoring to capture the data of Go applications**

Configure ARMS Prometheus Monitoring in the ARMS console to capture the data of Go applications.

1. In the left-side navigation pane, click **Prometheus Monitoring**.
2. On the top of the Prometheus monitoring page, select the region where the Container Service Kubernetes cluster is located. Find the target cluster, and click Installation in the Actions column.

3. After the ARMS Prometheus agent is installed, find the target cluster, and click Settings in the Actions column.

4. On the Details tab, click Add ServiceMonitor. In the Add ServiceMonitor dialog box, enter the following information:

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  # Enter a unique name.
  name: promethues-arms-aliyun-go-demo
  # Enter the target namespace.
  namespace: default
spec:
  endpoints:
    - interval: 30s
      # Enter the value of the Name field for Port of Prometheus Exporter.
      port: arms-go-demo
      # Enter the value of the Path field for Prometheus Exporter.
      path: /metrics
      namespaceSelector:
        any: true
        # The namespace of the NGINX Demo.
        selector:
          matchLabels:
            app: promethues-arms-aliyun-go-demo
```

**Step 4: Display the data of Go applications on the Grafana dashboard**

Import the Grafana dashboard template in the ARMS console and specify the Container Service Kubernetes cluster where the Prometheus data source is located.

1. Go to Host Dashboard.
2. In the left-side navigation pane, choose + > Import, enter 6671 in the Grafana.com Dashboard field, and click Load.

3. On the Import page, set the following information and click Import.
   
   a) Enter a custom dashboard name in the Name field.
   b) Select your Container Service Kubernetes cluster from the Folder drop-down list.
   c) Select your Container Service Kubernetes cluster from the prometheus-apl drop-down list.

   After the configuration is complete, the ARMS Prometheus Grafana Go dashboard appears, as shown in the following figure.

**Step 5: Create an alert**

1. You can select one of the two available methods to access the Create Alert page.
   
   - On the New Dashboard page of the ARMS Prometheus Grafana dashboard, click the icon to go to the ARMS Prometheus Create Alert dialog box.
   - In the left-side navigation pane of the console, choose Alerts > Alert Policies. On the Alert Policies page, choose Create Alert > Prometheus in the upper-right corner.
2. In the **Create Alert** dialog box, enter all required information and click **Save**.
   a) Enter Alert Name, for example, network receiving pressure alert.
   b) Select the corresponding **Cluster** of the Prometheus monitoring job.
   c) Set **Type** to **grafana**.
   d) Select the specific **dashboard** and **chart** to monitor.
   e) Set alert rules.

   **A.** Select **Meet All of the Following Criteria**.
   **B.** Edit the alert rule. For example, an alert is triggered when the value of $N$ is 5 and
   the average value of network receiving bytes (MB) is at least 3.

   **Note:**
   A Grafana chart may contain data of curve A, curve B, and curve C. You can select
   one of them to monitor.

   **C.** In the **PromQL** field, edit or enter a new PromQL statement.

   **Notice:**
   The "$" symbol in the PromQL statement can lead to an error. You must delete
   the "=" symbol and the parameters on both sides of it in the statement containing
   the "$" symbol. For example, modify
   
   ```prometheus
   sum(rate(container_network_receive_bytes_total{instance=~"^$HostIp.*"}[1m]))
   ```
   
   to
   
   ```prometheus
   sum(rate(container_network_receive_bytes_total[1m]))
   ```

   f) Set Notification Mode. For example, select SMS.
   g) Set Notification Receiver. In the **Contact Groups** box, click the name of a contact
   group. If the contact group appears in the **Selected Groups** box, the setting is
   successful.

**What's next**

After the ARMS Prometheus Grafana Go dashboard is configured, you can view Prometheus
Monitoring metrics and customize the dashboard. For more information, see the following
documents.

**View Prometheus Monitoring metrics**

**Use ARMS Prometheus Monitoring to customize the Grafana dashboard**

**Related concepts**

Prometheus monitoring overview
4.1.5 Use ARMS Prometheus Monitoring to monitor Redis data

Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS) allows you to monitor Redis by capturing Redis data and displaying the captured data on the ARMS Prometheus Grafana dashboard.

Prerequisites

- #unique_5
- #unique_15

Context

The following figure shows the workflow.

Step 1: Capture Redis data through an external application

Deploy the redis-exporter application to the Container Service Kubernetes cluster to capture Redis data.

1. In the left-side navigation pane, choose Clusters > Clusters. On the Clusters page, find the target cluster, and click Dashboard in the Actions column.

2. In the left-side navigation pane, choose Workloads > Deployments. In the upper-right corner, click CREATE, and enter the following information in the CREATE FROM TEXT INPUT section:

   ![Note:]
   
   Change <Address> and <Password> to the corresponding values of Redis. You can also use the sample values provided by ARMS to experience the service:
   
   - Redis address: redis://r-bp167pgkqkh7h25coypd.redis.rds.aliyuncs.com:6379
   - Redis password: Arms1234
   
   apiVersion: extensions/v1beta1
In this step, the redis-exporter application is deployed to the Container Service Kubernetes cluster.

3. In the left-side navigation pane, choose **Discovery and Load Balancing > Services**. In the upper-right corner, click **CREATE**, and enter the following information in the **CREATE FROM TEXT INPUT** section:

```yaml
apiVersion: v1
class: Service
metadata:
  app: redis-exporter
  name: redis-exporter
spec:
  ports:
  - name: redis-exporter
    port: 9121
    protocol: TCP
    targetPort: 9121
    type: NodePort
  selector:
    app: redis-exporter
```

**Step 2: Configure ARMS Prometheus Monitoring to receive Redis data**

Configure ARMS Prometheus Monitoring in the ARMS console to receive Redis data captured by the external application.

1. In the left-side navigation pane, click **Prometheus Monitoring**.
2. On the top of the Prometheus monitoring page, select the region where the Container Service Kubernetes cluster is located. Find the target cluster, and click Installation in the Actions column.

3. After the ARMS Prometheus agent is installed, find the target cluster, and click Settings in the Actions column.

4. On the Details tab, click Add ServiceMonitor. In the Add ServiceMonitor dialog box, enter the following information:

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  # Enter a unique name.
  name: redis-exporter
  # Enter the target namespace.
  namespace: default
spec:
  endpoints:
    - interval: 30s
      # The Redis Grafana template ID is 763.
      # Enter the value of the Name field for Port of Prometheus Exporter in the service.yaml file.
      port: redis-exporter
      # Enter the path exposed in Prometheus Exporter code.
      path: /metrics
      namespaceSelector:
        any: true
        # The namespace of the NGINX Demo
        selector:
          matchLabels:
            # Enter the label field of service.yaml to locate the target service.yaml file.
            app: redis-exporter
```

**Step 3: Display the Redis data on the Grafana dashboard**

Import the Grafana dashboard template in the ARMS console and specify the Container Service Kubernetes cluster where the Prometheus data source is located.

1. Go to Host Dashboard.
2. In the left-side navigation pane, choose + > Import, enter 763 in the Grafana.com Dashboard field, and click Load.

![Import page](image)

3. On the Import page, set the following information and click Import.

   a) Enter a custom dashboard name in the Name field.
   b) Select your Container Service Kubernetes cluster from the Folder drop-down list.
   c) Select your Container Service Kubernetes cluster from the prom drop-down list.

   After the configuration is complete, the ARMS Prometheus Grafana Redis dashboard is shown in the following figure.

Step 4: Create an alert

1. You can select one of the two available methods to access the Create Alert page.

   - On the New Dashboard page of the ARMS Prometheus Grafana dashboard, click the icon to go to the ARMS Prometheus Create Alert dialog box.
   - In the left-side navigation pane of the console, choose Alerts > Alert Policies. On the Alert Policies page, choose Create Alert > Prometheus in the upper-right corner.
2. In the Create Alert dialog box, enter all required information and click Save.
   a) Enter Alert Name, for example, network receiving pressure alert.
   b) Select the corresponding Cluster of the Prometheus monitoring job.
   c) Set Type to grafana.
   d) Select the specific dashboard and chart to monitor.
   e) Set alert rules.

   A. Select Meet All of the Following Criteria.
   B. Edit the alert rule. For example, an alert is triggered when the value of N is 5 and the average value of network receiving bytes (MB) is at least 3.

   ![Note:
   A Grafana chart may contain data of curve A, curve B, and curve C. You can select one of them to monitor.

   C. In the PromQL field, edit or enter a new PromQL statement.

   ![Notice:
   The "$" symbol in the PromQL statement can lead to an error. You must delete the "=" symbol and the parameters on both sides of it in the statement containing the "$" symbol. For example, modify sum (rate (container_network_receive_bytes_total{instance=~"^$HostIp.*"}[1m])) to sum (rate (container_network_receive_bytes_total[1m]))

   f) Set Notification Mode. For example, select SMS.
   g) Set Notification Receiver. In the Contact Groups box, click the name of a contact group. If the contact group appears in the Selected Groups box, the setting is successful.

What's next

After the ARMS Prometheus Grafana Redis dashboard is configured, you can view Prometheus Monitoring metrics and customize the dashboard. For more information, see the following documents.

Related tasks
View Prometheus Monitoring metrics
Use ARMS Prometheus Monitoring to customize the Grafana dashboard
Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS) allows you to display monitoring data on the ARMS Prometheus Grafana dashboard. You can customize
Create alarms

4.1.6 Use ARMS Prometheus Monitoring to customize the Grafana dashboard

Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS) allows you to display monitoring data on the ARMS Prometheus Grafana dashboard. You can customize a Grafana dashboard or import the dashboard from the Grafana official website. This topic describes how to customize the Grafana dashboard to display monitoring data.

Prerequisites

Step 1: Deploy the application to the Container Service Kubernetes cluster

Deploy the application to the Container Service Kubernetes cluster so that ARMS Prometheus Monitoring can monitor and capture the JVM data.

1. Run the following commands in the buildDockerImage.sh file by row:

   mvn clean install -DskipTests
   docker build -t <name of the local temporary Docker image>:<version of the local temporary Docker image> . --no-cache
   sudo docker tag <name of the local temporary Docker image>:<version of the local temporary Docker image> <registry domain name>/<namespace>/<image name>:<image version>
   sudo docker push <registry domain name>/<namespace>/<image name>:<image version>

   Example:

   mvn clean install -DskipTests
docker build -t promethues-demo:v0 . --no-cache
   sudo docker tag promethues-demo:v0 registry.cn-hangzhou.aliyuncs.com/fuling/promethues-demo:v0
   sudo docker push registry.cn-hangzhou.aliyuncs.com/fuling/promethues-demo:v0

   In this step, a Docker image named promethues-demo is created, and the image is pushed to Alibaba Cloud Docker Registry.

2. In the left-side navigation pane, choose Clusters > Clusters. On the Clusters page, find the target cluster, and click Dashboard in the Actions column.
3. In the left-side navigation pane, choose Workloads > Deployments. In the upper-right corner, click CREATE, and enter the following information in the CREATE FROM TEXT INPUT section:

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>prometheus-demo</td>
</tr>
<tr>
<td>replicas</td>
<td>2</td>
</tr>
<tr>
<td>annotations</td>
<td>prometheus.io/scrape: 'true'</td>
</tr>
<tr>
<td></td>
<td>prometheus.io/path: '/prometheus-metrics'</td>
</tr>
<tr>
<td></td>
<td>prometheus.io/port: '8081'</td>
</tr>
<tr>
<td>labels</td>
<td>app: tomcat</td>
</tr>
<tr>
<td></td>
<td>name: tomcat-normal</td>
</tr>
<tr>
<td></td>
<td>name: tomcat-monitor</td>
</tr>
<tr>
<td>ports</td>
<td>containerPort: 8080</td>
</tr>
<tr>
<td></td>
<td>name: tomcat-normal</td>
</tr>
<tr>
<td></td>
<td>containerPort: 8081</td>
</tr>
</tbody>
</table>

In this step, the promethues-demo Docker image is deployed to the Container Service Kubernetes cluster.

4. In the left-side navigation pane, choose Discovery and Load Balancing > Services. In the upper-right corner, click CREATE, and enter the following information in the CREATE FROM TEXT INPUT section:

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>tomcat</td>
</tr>
<tr>
<td>namespace</td>
<td>default</td>
</tr>
<tr>
<td>ports</td>
<td>name: tomcat-normal</td>
</tr>
<tr>
<td></td>
<td>name: tomcat-monitor</td>
</tr>
<tr>
<td></td>
<td>port: 8080</td>
</tr>
<tr>
<td></td>
<td>protocol: TCP</td>
</tr>
<tr>
<td></td>
<td>targetPort: 8080</td>
</tr>
<tr>
<td></td>
<td>name: tomcat-monitor</td>
</tr>
<tr>
<td></td>
<td>port: 8081</td>
</tr>
</tbody>
</table>
```
Step 2: Install the ARMS Prometheus agent for the application

1. In the left-side navigation pane, click **Prometheus Monitoring**.

2. On the top of the **Prometheus monitoring** page, select the region where the Container Service Kubernetes cluster is located. Find the target cluster, and click **Installation** in the **Actions** column.

   **Note:**
   If the installation fails, retry until it is successful. If the installation still fails after several attempts, contact us at our DingTalk account 23148410.

Step 3: Configure the data collection rule for Prometheus Monitoring to monitor the application

After the ARMS Prometheus agent is installed, it monitors CPU information, memory information, and network information by default. If you want to monitor non-default data, such as order information, you need to configure the data collection rule for Prometheus Monitoring to monitor the application.

1. Find the target cluster, and click **Settings** in the **Actions** column.

2. You can configure the data collection rule for Prometheus Monitoring to monitor the application in the following scenarios:

   • To monitor the business data of applications deployed in the Kubernetes cluster, such as order information, you can click **Add ServiceMonitor** on the **Details** tab. In the **Add ServiceMonitor** dialog box, enter the following information:

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  # Enter a unique name.
  name: tomcat-demo
  # Enter the target namespace.
  namespace: default
spec:
  endpoints:
  - interval: 30s
    # Enter the value of the Name field for Port of Prometheus Exporter.
    port: tomcat-monitor
    # Enter the value of the Path field for Prometheus Exporter.
    path: /prometheus-metrics
namespaceSelector:
```

any: true
selector:
  matchLabels:
    # Enter the label field of service.yaml to locate the target service.yaml file.
    app: tomcat

- To monitor business data outside the Kubernetes cluster, such as the number of Redis
counters, you can click Edit prometheus.yaml on the Details tab to configure
the native prometheus.yaml file. In the Edit prometheus.yaml dialog box, enter the
following information:

global:
  scrape_interval: 15s
  evaluation_interval: 15s
  scrape_configs:
    - job_name: 'prometheus'
      static_configs:
        - targets: ['localhost:9090']

Step 4: Create a Grafana dashboard

1. Go to Host Dashboard.
2. In the left-side navigation pane, choose + > Dashboard, and click Add Query in the New
   Panel section.
3. Select a cluster from the drop-down list next to Query. On the A collapse panel, select a
   monitoring metric, for example, go_gc_duration_seconds, from the Metrics drop-down
   list.
4. Click the chart icon on the left side of the page to select the visualization type of the
   dashboard, such as a chart, table, or heatmap, and configure other parameters as
   needed.
5. Click the setting icon on the left side of the page and enter a chart name.
6. Click the bell icon on the left side of the page. Click Create Alert in the Alert section
   to configure alerts. The subsequent alert configuration page will jump to the alert
   configuration page in ARMS Prometheus Monitoring.
7. Click the save icon in the upper-right corner. In the Save As... dialog box, enter the
   dashboard name, select a cluster, and click Save to save the dashboard and chart. You
   can create multiple dashboards and charts as needed.

Step 5: Debug data and monitor complex metrics

To monitor metrics that involve complex operations, debug data in ARMS Prometheus
Monitoring to acquire the corresponding PromQL statement.

1. On the Prometheus monitoring page, click Settings in the Actions column.
2. Click **Data debugging**. On the **Explore** page of ARMS Prometheus Grafana, you can enter the PromQL statement in the **Metrics** field for debugging.

3. After successful debugging, you can repeat the preceding steps to add more dashboards or charts. For more information, see **Step 4: Create a Grafana dashboard**.

**Result**

The ARMS Prometheus Grafana dashboard after the configuration is complete is shown in the following figure.

**View Prometheus Monitoring metrics**

**Use ARMS Prometheus Monitoring to monitor JVM data**

This tutorial describes how to expose JVM data by instrumentation points in applications, capture JVM data by using Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS), and display JVM data on the ARMS Prometheus Grafana dashboard, finally to monitor JVM data with ARMS Prometheus Monitoring.

### 4.1.7 Use ARMS Prometheus Monitoring to monitor ZooKeeper

This tutorial describes how to expose data by using instrumentation points in ZooKeeper, capture data by using Create a Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS), and display data on the ARMS Prometheus Grafana dashboard, to ultimately monitor ZooKeeper with ARMS Prometheus Monitoring.

**Context**

The following figure shows the workflow.

**Step 1: Start the JMX service**

Enable the Java Management Extensions (JMX) service in ZooKeeper to acquire resource information.

1. Add `JMXPORT=8999` in line 44 in the `/opt/zk/zookeeper-3.4.10/bin/zkServer.sh` file as follows:

   ```bash
   if [ "x$JMXLOCALONLY" = "x" ]
   then
     JMXLOCALONLY=false
   fi
   
   JMXPORT=8999 ## Added here in line 44
   
   if [ "x$JMXDISABLE" = "x" ] || [ "$JMXDISABLE" = 'false' ]
   then
     echo "ZooKeeper JMX enabled by default $JMXPORT ..." >&2
   fi
   ```
2. Restart ZooKeeper.

```
/opt/zk/zookeeper-3.4.10/bin/zkServer.sh start /opt/zk/zookeeper-3.4.10/conf/zoo_sample.cfg &
```

**Step 2: Start jmx_exporter in ZooKeeper**

Start jmx_exporter to allow access to JMX information through HTTP so that ARMS Prometheus Monitoring can capture data.

1. Download `zookeeper.yaml` to the `/opt/exporter_zookeeper/` directory.
2. Add `hostPort: localhost:8999` to the first line in the downloaded file `/opt/exporter_zookeeper/zookeeper.yaml` to expose the running port of the JMX service to jmx_exporter.
3. Download the executable file of jmx_exporter to the opt/exporter_zookeeper/ directory.
4. Start jmx_exporter.

```
```

The application configuration is complete. You can run the following command to check whether jmx_exporter is running properly:

```
curl http://<IP address of the server where jmx_exporter is located>:9997/metrics
```

**Step 3: Configure ARMS Prometheus Monitoring to capture the data of ZooKeeper**

Configure ARMS Prometheus Monitoring in the ARMS console to capture the data of ZooKeeper.

1. In the left-side navigation pane, click **Prometheus Monitoring**.
2. On the top of the **Prometheus monitoring** page, select the region where the Container Service Kubernetes cluster is located, and click the name of the target cluster.
3. On the page that appears, click the **Details** tab and then click **Edit prometheus.yaml**.
4. Paste the following code to the file.

```
global:
s  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
s  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
```
Step 4: Display ZooKeeper data on the Grafana dashboard

Import the Grafana dashboard template in the ARMS console and specify the Container Service Kubernetes cluster where the Prometheus data source is located.

1. Go to Host Dashboard.

2. In the left-side navigation pane, choose + > **Import**, enter 10981 in the **Grafana.com Dashboard** field, and click **Load**.

3. On the **Import** page, set the following information and click **Import**.

   a) Enter a custom dashboard name in the **Name** field.
   b) Select your Container Service Kubernetes cluster from the **Folder** drop-down list.
   c) Select your Container Service Kubernetes cluster from the drop-down list at the bottom.

After the configuration is complete, the ARMS Prometheus Grafana ZooKeeper dashboard appears, as shown in the following figure.
Step 5: Create an alert

1. You can select one of the two available methods to access the Create Alert page.
   - On the New DashBoard page of the ARMS Prometheus Grafana dashboard, click the icon to go to the ARMS Prometheus Create Alert dialog box.
   - In the left-side navigation pane of the console, choose Alerts > Alert Policies. On the Alert Policies page, choose Create Alert > Prometheus in the upper-right corner.

2. In the Create Alert dialog box, enter all required information and click Save.
   a) Enter Alert Name, for example, network receiving pressure alert.
   b) Select the corresponding Cluster of the Prometheus monitoring job.
   c) Set Type to grafana.
   d) Select the specific dashboard and chart to monitor.
   e) Set alert rules.
      A. Select Meet All of the Following Criteria.
      B. Edit the alert rule. For example, an alert is triggered when the value of N is 5 and the average value of network receiving bytes (MB) is at least 3.

      Note:
      A Grafana chart may contain data of curve A, curve B, and curve C. You can select one of them to monitor.

      C. In the PromQL field, edit or enter a new PromQL statement.

      Notice:
      The "$" symbol in the PromQL statement can lead to an error. You must delete the "=" symbol and the parameters on both sides of it in the statement containing the "$" symbol. For example, modify sum (rate (container_network_receive_bytes_total{instance=~"^$HostIp.*"}[1m])) to sum (rate (container_network_receive_bytes_total[1m]))

      f) Set Notification Mode. For example, select SMS.
      g) Set Notification Receiver. In the Contact Groups box, click the name of a contact group. If the contact group appears in the Selected Groups box, the setting is successful.

What’s next
After the ARMS Prometheus Grafana ZooKeeper dashboard is configured, you can view Prometheus Monitoring metrics and customize the dashboard. For more information, see the following documents.

**Related tasks**

- View Prometheus Monitoring metrics
- Use ARMS Prometheus Monitoring to customize the Grafana dashboard

Prometheus Monitoring of Application Real-Time Monitoring Service (ARMS) allows you to display monitoring data on the ARMS Prometheus Grafana dashboard. You can customize a Grafana dashboard or import the dashboard from the Grafana official website. This topic describes how to customize the Grafana dashboard to display monitoring data.

Create alarms
5 Console functions