ALIBABA CLOUD

Alibaba Cloud

CloudMonitor

Best Practices

Document Version: 20201023

Alibaba Cloud

Legal disclaimer

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

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

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

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

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

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

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

Document conventions

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

Table of Contents

1.Create an alarm template 05
2.Receive alert notifications in a DingTalk group 07
3.Monitor the availability of services in a VPC 09
4.Create an alert rule for pods in Container Service for Kuber... 11
5.Query monitoring data by calling API operations 13
6.Automate O&M based on status change events of ECS insta... 17
7.Monitor resources based on tags 25

1.Create an alarm template

This topic illustrates how to use application groups and alarm templates to manage cloud resource alarm rules for various services more efficiently. These tactics are especially important for those who need to monitor and manage resources across several Alibaba Cloud products and regions and who need to modify alarm rules for these resources in a timely manner.

Purposes

- Configuring alarm rules for application groups rather than for single instances can improve efficiency by greatly reducing the time required to configure alarm rules.
  - By setting the resource range of an alarm rule to application group, your alarm rule will be effective for all resources within the target application group, and the number of resources monitored can expand as your services are scaled outward. After initial configuration, you can move specific resources into or out of the application group easily. You can also modify the alarm rule directly so to make changes effective to all instances within an application group.
  - Conversely, setting the resource range to instance will make your alarm rule effective for only one instance. Modifications to your alarm rule will also be effective for only one instance. As a consequence, supposing that you set all your alarm rules this way, as the number of your instances increase, managing alarm rules for these instances will become increasingly time consuming and difficult.
- Using alarm templates can also reduce the time required to configure alarm rules.

  The monitoring metrics and alarm thresholds of basic services, such as ECS, RDS, and SLB, are set to fixed values during alarm rule configuration. You can create alarm templates easily based on these configurations, and by creating alarm templates with your target metrics and condition thresholds, you can easily apply these templates to alarm rules you configure for an application group, making configuring rules easy even as your services scale outward. Using alarm templates also enables you to easily modify multiple alarm rules at the same time.

Procedures

The following case outlines the procedure that can be applied to the typical back-end services of an e-commerce company. This case serves to illustrate how you can create application groups and use alarm templates to easily build a service monitoring and alarming system on the cloud, even for growing service requirements.

1. Create an alarm template named "EcommerceBackendAlarmTemplate".
   i. Log on to the CloudMonitor Console.
   ii. From the left-side navigation pane, choose **Alarm Templates > Alarms**.
   iii. On the **Alarm Templates** page, click **Create Alarm Template** in the upper-right corner.
   iv. In the displayed dialog box, set the parameters in the Basic Info area.
   v. In the **View Alarm Rules** area, click **Add Alarm Rule** to add the required alarm rules to the alarm template.
   vi. Click **OK**.
2. Create an alarm contact and an alarm contact group.
   i. Log on to the CloudMonitor Console

i. Log on to the CloudMonitor Console.

ii. From the left-side navigation pane, choose **Alarms** > **Alarm Contacts**.

iii. On the **Alarm Contact Management** page, click **Create Alarm Contact** in the upper-right corner. In the displayed dialog box, enter your phone number and email.

To ensure that you can receive and verify alarm notifications in a timely manner, the system will send verification codes to your phone and email.

iv. Click the **Alarm Contact Group** tab.

v. In the upper-right corner, click **Create Alarm Contact Group**.

vi. In the displayed dialog box, enter the group name and select the contacts that you want to add to the group.

3. Create an application group and apply the alarm template. Here, we create an application group named "InventoryManagementOnlineEnvironment" and use the created alarm template "EcommerceBackendAlarmTemplate".

i. Log on to the CloudMonitor Console.

ii. In the left-side navigation pane, click **Application Groups**.

iii. On the **Application Groups** page, click **Create Group** in the upper-right corner.

iv. In the **Basic Information** area, set **Product Group Name** and **Contact Group**.

The contact group is the alarm contact group for receiving alarm notifications.

v. In the **MonitorAlarm** area, set **Select Template** and **Notification Methods** and enable **Initialize Agent Installation**.

The selected template is used to initialize alarm rules for the instances in the group. After new instances are created, a CloudMonitor agent will be automatically installed to collect monitoring data.

vi. In the **Add Instance dynamically** area, add at most three dynamic rules with the relationship of AND or OR. Then, click **Add Product** to customize dynamic rules for RDS and SLB.

Generally, the cloud resources used for inventory management are a server, database, and SLB resource. You can customize dynamic rules to add ECS instances. An ECS instance name can be matched through the condition of **Contain**, **start with**, or **end with**. The instances conforming to the dynamic rules will be added to the specified application group (including instances to be created in the future).

vii. Click **Create Application Group**.

The instances conforming to the dynamic rules are added to the created application group, which can be viewed on the **Basic Information** page of the application group.

2.Receive alert notifications in a DingTalk group

This topic describes how to configure a DingTalk chatbot to receive alert notifications in a DingTalk group.

Prerequisites

An Alibaba Cloud account is created. To create an Alibaba Cloud account, visit the account registration page.

Context

CloudMonitor allows you to receive alert notifications in a DingTalk group by following the procedure described in this topic.

To use a DingTalk chatbot to receive alert notifications, you do not need to modify any alert rules. You only need to create an alert contact with the webhook URL of the DingTalk chatbot configured or add the webhook URL of the DingTalk chatbot to an existing alert contact. When alerts are triggered based on alert rules in which the alert contact is an alert notification recipient, you can receive alert notifications in the DingTalk group where the DingTalk chatbot is created.

If you add the webhook URL of a DingTalk chatbot to an existing alert contact, the DingTalk group where the DingTalk chatbot is created will receive all the previous alert notifications that were sent to the alert contact in emails.

Create a DingTalk chatbot on DingTalk for PC

1. Start DingTalk for PC and go to the DingTalk group where you want to receive alert notifications.
2. In the upper-right corner, click the **Group Settings** icon.
3. In the Group Settings right-side pane, click **Group Assistant**. In the Group Assistant right-side pane, click **Add Robot**.
4. In the **ChatBot** dialog box, click the + icon and click **Custom**.
5. In the **Robot details** dialog box, click **Add**.
6. In the Add Robot dialog box, enter a chatbot name, for example, CloudMonitor alert notifications.
7. Select **Custom Keywords** in the Security Settings section and add the following five keywords one by one: Alibaba Cloud, Service, CloudMonitor, Monitor, and ECS. Then, click **Finished**.
8. Click **Copy** to copy the webhook URL.
9. Click **Finished**.

Add the webhook URL of the DingTalk chatbot to an alert contact

1. Log on to the CloudMonitor console.
2. In the left-side navigation pane, choose **Alarms** > **Alarm Contacts**. The **Alarm Contacts** page appears.

3. On the Alert Contacts tab, find the target alert contact and click **Edit** in the **Actions** column.
4. In the Set Alarm Contact right-side pane, enter the webhook URL in the DingTalk Robot field.

3.Monitor the availability of services in a VPC

This topic describes how to use Cloud Monitor to monitor the availability of services in a virtual private cloud (VPC).

Background information

As an increasing number of users migrate their services from the classic network to VPCs that are safer and more reliable, users need to monitor the availability of services in VPCs. This topic describes how to monitor the availability of services in a VPC, including Elastic Compute Service (ECS), ApsaraDB RDS, ApsaraDB for Redis, and Server Load Balancer (SLB).

Preparations

The following figure shows how to monitor the availability of services in a VPC.

Before you can monitor the availability of services in a VPC, you must install the Cloud Monitor agent on the ECS instances that will be used as monitoring nodes. To monitor the availability of a service in a VPC, create an availability monitoring task in the Cloud Monitor console, select a monitoring node, and specify the URL or port of the monitored target. After you create the availability monitoring task, the Cloud Monitor agent on the monitoring node sends an HTTP request or a Telnet request to the URL or port every minute. The Cloud Monitor agent collects the response time and status codes and reports the monitoring results to Cloud Monitor. Cloud Monitor displays the monitoring results in a chart and generates alerts if the connection times out or the monitoring fails.

Procedure

> **Note**
>
> - You must install the Cloud Monitor agent on the ECS instances that will be used as monitoring nodes.
> - You must create an application group and add the monitoring nodes to the group.

1. Log on to the Cloud Monitor console.
2. In the left-side navigation pane, click **Application Groups**.
3. On the **Application grouping** tab of the **Application Groups** page, click the name or ID of the application group for which you want to create an availability monitoring task.
4. In the left-side navigation pane of the page that appears, click **Availability Monitoring**.
5. On the page that appears, click **Create Configuration** in the upper-right corner.
6. In the **CreateAvailability Monitoring** dialog box, set relevant parameters.
   - To monitor whether local processes on ECS instances in a VPC respond, select the ECS instances to be monitored in the Target Server section, set the Detection Target parameter to URL or IP address, and enter the addresses in *localhost:port/path* format in the Detection Type section.

- To monitor whether an SLB instance in a VPC responds, select an ECS instance that resides in the same VPC as the SLB instance in the Target Server section, set the Detection Target parameter to URL or IP address, and enter the address of the SLB instance in the Detection Type section.
- To monitor whether an ApsaraDB RDS or ApsaraDB for Redis instance in a VPC responds to an ECS instance, add the ApsaraDB RDS or ApsaraDB for Redis instance to the application group of the ECS instance, select the ECS instance in the Target Server section, and set the Detection Target parameter to RDS-DB or KVStore for Redis.

7. Click **OK**.

   You can view the monitoring results in the monitoring chart of the task. If the connection times out or the monitoring fails, you can receive an alert notification.

   

8. Find the task and click **Monitoring Charts** in the Actions column to view the monitoring details of the task.

   

4.Create an alert rule for pods in Container Service for Kubernetes

Cloud Monitor provides multiple metrics, such as the CPU utilization, memory usage, and network conditions, to help you monitor the status of Container Service for Kubernetes clusters. After you create a Container Service for Kubernetes cluster, Cloud Monitor automatically monitors the cluster. You can log on to the Cloud Monitor console to view the detailed monitoring data. You can configure alert rules for the metrics. If a metric value exceeds the specified threshold, Cloud Monitor sends an alert notification.

Prerequisites

- A Container Service for Kubernetes cluster is created. For more information, see Create a cluster of ACK Managed Edition.
- An application group is created for the Container Service for Kubernetes cluster. For more information, see Create an application group.
- An alert contact or alert group is created. For more information, see Create an alert contact or alert group.

Context

- Monitoring data is retained for up to 31 days.
- You can view the monitoring data for up to 14 consecutive days.

Procedure

1. Create an alert template for Container Service for Kubernetes.
   i. Log on to the Cloud Monitor console.
   ii. In the left-side navigation pane, choose **Alerts** > **Alert Templates**.
   iii. On the **Alert Templates** page, click **Create Alert Template** in the upper-right corner.
   iv. In the **Create/modify alert templates** panel, enter the template name, select **Kubernetes** as the service, and configure alert rules.
   v. Click **OK**.
2. Apply the alert template to the application group that is created for your Container Service for Kubernetes cluster.
   i. In the **Create/modify alert template complete** message, click **OK**.
   ii. In the **Apply Template to Group** dialog box, select the application group and set the Muted, Effective Period, HTTP CallBack, and Option parameters.
   iii. Click **OK**.
   iv. In the **Apply Template to Group** message, click **OK**.

What's next

- If you need to send alert notifications for different Container Service for Kubernetes clusters to different alert groups, you must modify the alert group associated with each application group.

For more information, see Modify an application group.

- You can call an API operation to modify alert groups for multiple application groups at a time.

  Log on to OpenAPI Explorer and call the PutContactGroup operation to modify the alert groups.

  For more information about how to set the parameters of the PutContactGroup operation, see PutContactGroup.

5.Query monitoring data by calling API operations

This topic describes how to call API operations to query monitoring data of various Alibaba Cloud services.

Large enterprises have their own operations and maintenance (O&M) and monitoring systems. When they migrate business to Alibaba Cloud, these enterprises need to integrate monitoring data of cloud resources with their existing systems. This topic describes how to use Cloud Monitor API operations to query monitoring data of various services. This way, you can integrate monitoring data of Alibaba Cloud with your existing systems.

API operations for querying monitoring data of metrics

Cloud Monitor provides the following operations to query monitoring data of metrics:

- Operation used to query services: queries services that can be monitored by Cloud Monitor. For more information, see DescribeProjectMeta.
- Operation used to query metrics: queries metrics that are available for a monitored service. For more information, see DescribeMetricMetaList.
- Operations used to query monitoring data: query monitoring data based on services and metrics. For more information, see DescribeMetricMetaList and DescribeMetricLast.

Usage notes:

- The DescribeMetricList and DescribeMetricLast operations allow you to query data of a specific metric for all your instances. You can create multiple threads to query the monitoring data of multiple metrics at a time. Alternatively, you can create a single thread to obtain the monitoring data of multiple metrics one by one.
- The DescribeMetricList operation supports a maximum of 20 queries per second (QPS), whereas the DescribeMetricLast operation supports a maximum of 30 QPS.
- The DescribeMetricLast operation is applicable to scenarios where you need to obtain the most recent monitoring data at regular intervals. The time window automatically slides forward. For each window, the most recent record is retrieved.
- Services may take some time to report monitoring data to Cloud Monitor. The delay varies with different services. We recommend that you extend the time window by 5 to 10 minutes when you call the DescribeMetricLast operation to query the latest data.
- Cloud Monitor retains data that is obtained every few seconds for 7 days, and data that is obtained every few minutes for 31 days.
- If you want to query the aggregate data of all your instances, you do not need to specify the Dimensions parameter.

Example

The following example demonstrates how to call the DescribeMetricLast operation to query the latest monitoring data and the DescribeMetricList operation to query the monitoring data in a specified time range.

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
```

```
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.profile.DefaultProfile;
import com.google.gson.Gson;
import java.util.*;
import com.aliyuncs.cms.model.v20190101. *;

/**
 * You can call the DescribeMetricList operation to query the monitoring data of a specified instance in a specified time range.
 * The DescribeMetricList operation can query the monitoring data of multiple instances at a time.
 * To query the monitoring data of multiple instances in a specified time range, specify these instances for the query. You can specify a maximum of 10 instances at a time.
 * Query monitoring data in a specified time range.
 */
public class DescribeMetricList {

    public static void main(String[] args) {
        DefaultProfile profile = DefaultProfile.getProfile("cn-hangzhou", "<accessKeyId>", "<accessSecret>");
        IAcsClient client = new DefaultAcsClient(profile);

        DescribeMetricListRequest request = new DescribeMetricListRequest();
        // You can call the DescribeMetricMetaList and DescribeProjectMeta operations to query the namespace and metric.
        request.setNamespace("acs_ecs_dashboard");
        request.setMetricName("cpu_total");
        // The Period parameter is set to 60, which specifies that monitoring data is obtained every 60 seconds. The value of the Period parameter varies with metrics. The period of most metrics are set to 60 seconds by default.
        request.setPeriod("60");
        // The number of entries to return on each page. A maximum of 1,000 entries can be returned for each query.
        request.setLength("1000");
        // The beginning of the time range to query.
        request.setStartTime("2019-07-22 11:00:00");
        // The end of the time range to query.
        request.setEndTime("2019-07-22 12:00:00");
        // Set the Dimensions parameter to filter monitoring data. The value can be a JSON array or a JSON object.
        request.setDimensions("[{\"instanceId\":\"i-8vb******\"}]");
```

```
        try {
            DescribeMetricListResponse response = client.getAcsResponse(request);
            System.out.println(new Gson().toJson(response));
        } catch (ServerException e) {
            e.printStackTrace();
        } catch (ClientException e) {
            System.out.println("ErrCode:" + e.getErrCode());
            System.out.println("ErrMsg:" + e.getErrMsg());
            System.out.println("RequestId:" + e.getRequestId());
        }

    }
}
```

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.profile.DefaultProfile;
import com.google.gson.Gson;
import java.util.*;
import com.aliyuncs.cms.model.v20190101. *;

/**
 * Query the latest monitoring data.
 **/
public class DescribeMetricLast {

    public static void main(String[] args) {
        DefaultProfile profile = DefaultProfile.getProfile("cn-hangzhou", "<accessKeyId>", "<accessSecret
>");
        IAcsClient client = new DefaultAcsClient(profile);

        DescribeMetricLastRequest request = new DescribeMetricLastRequest();

         // You can call the DescribeMetricMetaList and DescribeProjectMeta operations to query the nam
espace and metric.
        request.setNamespace("acs_ecs_dashboard");
```

```
        request.setMetricName("cpu_total");
        // Set the Dimensions parameter to filter monitoring data. The value can be a JSON array or a JSO
N object.
        request.setDimensions("[{\"instanceId\":\"i-8vb6p*****\"}]");
        // The number of entries to return on each page. A maximum of 1,000 entries can be returned for
each query.
        request.setLength("1000");
        // The beginning of the time range to query.
        request.setStartTime("2019-07-22 11:00:00");
        // The end of the time range to query.
        request.setEndTime("2019-07-22 12:00:00");
        request.setPeriod("60");

        try {
            DescribeMetricLastResponse response = client.getAcsResponse(request);
            System.out.println(new Gson().toJson(response));
        } catch (ServerException e) {
            e.printStackTrace();
        } catch (ClientException e) {
            System.out.println("ErrCode:" + e.getErrCode());
            System.out.println("ErrMsg:" + e.getErrMsg());
            System.out.println("RequestId:" + e.getRequestId());
        }

    }
}
```

6.Automate O&M based on status change events of ECS instances

In addition to the existing system events, CloudMonitor supports the status change events for Elastic Compute Service (ECS). The status change events include interruption notification events that are applied to preemptible instances. A status change event is triggered when the status of an ECS instance changes. The status changes can be caused by operations that you perform in the ECS console and by calling API operations or using SDKs, auto scaling, overdue payment, and system exceptions.

Context

The existing system events for ECS are used to notify you of alerts that require manual operations. The status change events are not about alerts. They are common notifications that are suitable for automated audit and O&M scenarios. CloudMonitor allows you to automatically handle the status change events of ECS instances by using Function Compute or Message Service (MNS).

Before you begin

Create an MNS queue

- Create an MNS queue.
  i. Log on to the MNS console.
  ii. On the **Queues** page, select a region and click **Create Queue** in the upper-right corner.
  
  iii. In the Create Queue dialog box, enter a queue name, set relevant parameters, and click **OK**. In this example, set the queue name to ecs-cms-event.
- Create an event-triggered alert rule.
  i. Log on to the CloudMonitor console.
  ii. In the left-side navigation pane, click **Event Monitoring**.
  iii. On the **Event Monitoring** page, click the **Alarm Rules** tab. On the Alarm Rules tab, click **Create Event Alert**.
  
  iv. In the **Basic Information** section of the Create / Modify Event Alert right-side pane, enter an alert rule name. In this example, enter ecs-test-rule.
  v. In the **Event alert** section, perform the following operations:
    - Set the **Event Type** parameter to **System Event**.
    - Set the **Product Type** parameter to **ECS**.
    - Set the **Event Type** parameter to **Status Notification**.
    - Set the **Event Name** parameter as needed.

- Set the Resource Range parameter as needed. If you set the **Resource Range** parameter to **All Resources**, CloudMonitor sends alert notifications for all resource-related events. If you set the **Resource Range** parameter to **Application Groups**, CloudMonitor sends alert notifications for events related to the resources in the specified application group.

vi. In the **Alarm Type** section, perform the following operations:

- Set the **Contact Group** and **Notification Method** parameters as needed.
- Select **MNS queue** and set the **Region** and **Queue** parameters as needed. In this example, select the ecs-cms-event queue.

vii. Click **OK**.

- Install Python dependencies.

  The following code is tested in Python 3.6. You can use other programming languages, such as Java, as needed.

  Use Python Package Index (PyPI) to install the following Python dependencies:

  - aliyun-python-sdk-core-v3>=2.12.1
  - aliyun-python-sdk-ecs>=4.16.0
  - aliyun-mns>=1.1.5

Procedure

CloudMonitor sends all status change events of ECS instances to MNS. Then, you can write code to receive messages from MNS and handle the messages.

- Practice 1: Record all creation and release events of ECS instances

  You cannot query ECS instances that have been released in the ECS console. If you need to query released ECS instances, you can store status change events of all ECS instances in your own database or logs. When an ECS instance is created, a Pending event is triggered. When an ECS instance is released, a Deleted event is triggered. CloudMonitor records both types of events.

  i. Create a Conf file.

  Add the following parameters related to MNS in the Conf file:

  - `endpoint`: the endpoint for accessing MNS. You can obtain the endpoint by clicking **Get Endpoint** on the **Queues** page in the MNS console.
  - `access_key` and `access_key_secret`: the AccessKey ID and AccessKey secret used to access MNS. You can obtain the AccessKey ID and AccessKey secret in the User Management console.
  - `region_id` and `queue_name`: the region where the MNS queue resides and the name of the MNS queue. You can obtain the region ID and queue name on the **Queues** page in the MNS console.

```
class Conf:
    endpoint = 'http://<id>.mns.<region>.aliyuncs.com/'
    access_key = '<access_key>'
    access_key_secret = '<access_key_secrect>'
     = 'cn-beijing'
    queue_name = 'test'
    vsever_group_id = '<your_vserver_group_id>'
```

ii. Use the MNS SDK to develop an MNS client for receiving messages from MNS.

```
# -*- coding: utf-8 -*-
import json
from mns.mns_exception import MNSExceptionBase
import logging
from mns.account import Account
from . import Conf


class MNSClient(object):
    def __init__(self):
        self.account =  Account(Conf.endpoint, Conf.access_key, Conf.access_key_secret)
        self.queue_name = Conf.queue_name
        self.listeners = dict()

    def regist_listener(self, listener, eventname='Instance:StateChange'):
        if eventname in self.listeners.keys():
            self.listeners.get(eventname).append(listener)
        else:
            self.listeners[eventname] = [listener]

    def run(self):
        queue = self.account.get_queue(self.queue_name)
        while True:
            try:
                message = queue.receive_message(wait_seconds=5)
                event = json.loads(message.message_body)
                if event['name'] in self.listeners:
                    for listener in self.listeners.get(event['name']):
                        listener.process(event)
                queue.delete_message(receipt_handle=message.receipt_handle)
```

```
        except MNSExceptionBase as e:
            if e.type == 'QueueNotExist':
                logging.error('Queue %s not exist, please create queue before receive message.', se
lf.queue_name)
            else:
                logging.error('No Message, continue waiting')


class BasicListener(object):
    def process(self, event):
        pass
```

The preceding code is used to receive messages from MNS and delete the messages after the listener is called to consume the messages.

iii. Register a listener to consume events. The following listener generates a log entry after it receives a Pending or Deleted event.

```
 # -*- coding: utf-8 -*-
import logging
from .mns_client import BasicListener


class ListenerLog(BasicListener):
    def process(self, event):
        state = event['content']['state']
        resource_id = event['content']['resourceId']
        if state == 'Panding':
            logging.info(f'The instance {resource_id} state is {state}')
        elif state == 'Deleted':
            logging.info(f'The instance {resource_id} state is {state}')
```

Add the following code to the Main function:

```
mns_client = MNSClient()

mns_client.regist_listener(ListenerLog())

mns_client.run()
```

In the production environment, you can store the events in your database or Log Service for subsequent queries and audits.

- Practice 2: Automatically start ECS instances that are shut down

  In scenarios where ECS instances may be shut down unexpectedly, you may want to automatically start the ECS instances.

  You can reuse the MNS client developed in Practice 1 and create another listener. When the listener receives a Stopped event for an ECS instance, you can run the start command on the ECS instance to start it.

```
# -*- coding: utf-8 -*-
import logging
from aliyunsdkecs.request.v20140526 import StartInstanceRequest
from aliyunsdkcore.client import AcsClient
from .mns_client import BasicListener
from .config import Conf



class ECSClient(object):
    def __init__(self, acs_client):
        self.client = acs_client

    # Start the target ECS instance.
    def start_instance(self, instance_id):
        logging.info(f'Start instance {instance_id} ...')
        request = StartInstanceRequest.StartInstanceRequest()
        request.set_accept_format('json')
        request.set_InstanceId(instance_id)
        self.client.do_action_with_exception(request)



class ListenerStart(BasicListener):
    def __init__(self):
        acs_client = AcsClient(Conf.access_key, Conf.access_key_secret, Conf.region_id)
        self.ecs_client = ECSClient(acs_client)

    def process(self, event):
        detail = event['content']
        instance_id = detail['resourceId']
        if detail['state'] == 'Stopped':
            self.ecs_client.start_instance(instance_id)
```

In the production environment, you can listen to Starting, Running, or Stopped events after the start command is run. Then, you can perform further O&M by using a timer and a counter based on whether the ECS instance is started.

- Practice 3: Automatically remove preemptible instances from SLB before they are released

  An interruption notification event is triggered 5 minutes before a preemptible instance is released. During the 5 minutes, you can perform specific operations to prevent your services from being interrupted. For example, you can remove the target preemptible instance from a Server Load Balancer (SLB) instance.

  You can reuse the MNS client developed in Practice 1 and create another listener. When the listener receives the interruption notification event for a preemptible instance, you can call the SLB SDK to remove the preemptible instance from an SLB instance.

```
# -*- coding: utf-8 -*-
from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.request import CommonRequest
from .mns_client import BasicListener
from .config import Conf


class SLBClient(object):
    def __init__(self):
        self.client = AcsClient(Conf.access_key, Conf.access_key_secret, Conf.region_id)
        self.request = CommonRequest()
        self.request.set_method('POST')
        self.request.set_accept_format('json')
        self.request.set_version('2014-05-15')
        self.request.set_domain('slb.aliyuncs.com')
        self.request.add_query_param('RegionId', Conf.region_id)

    def remove_vserver_group_backend_servers(self, vserver_group_id, instance_id):
        self.request.set_action_name('RemoveVServerGroupBackendServers')
        self.request.add_query_param('VServerGroupId', vserver_group_id)
        self.request.add_query_param('BackendServers',
                                     "[{'ServerId':'" + instance_id + "','Port':'80','Weight':'100'}]")
        response = self.client.do_action_with_exception(self.request)
        return str(response, encoding='utf-8')


class ListenerSLB(BasicListener):
    def __init__(self, vsever_group_id):
        self.slb_caller = SLBClient()
        self.vsever_group_id = Conf.vsever_group_id

    def process(self, event):
        detail = event['content']
        instance_id = detail['instanceId']
        if detail['action'] == 'delete':
            self.slb_caller.remove_vserver_group_backend_servers(self.vsever_group_id, instance_id)
```

> **Notice**
>
> For interruption notification events, set the `event name` in the following way: `mns_client.regist_listener(ListenerSLB(Conf.vsever_group_id), 'Instance:PreemptibleInstanceInterruption')`.

In the production environment, you can apply for another preemptible instance and add it as a backend server to SLB to ensure the performance of your services.

7.Monitor resources based on tags

Large enterprises or organizations may maintain thousands of resources. If you use application groups to manage these resources, thousands of applications groups are required. Manual maintenance is time-consuming and error-prone. CloudMonitor allows you to attach tags to resources, classify and manage resources based on tags, and automate resource monitoring. Tag-based monitoring helps you reduce monitoring costs.

Prerequisites

- An Alibaba Cloud account is created and passes real-name verification. For more information, see Account registration and Real-name verification.
- Tags are attached to resources of Alibaba Cloud services based on business needs.

Context

Note the following limits when you use CloudMonitor to manage resources based on tags:

- You can only use tags to manage Elastic Compute Service (ECS), ApsaraDB for RDS, and Server Load Balancer (SLB) instances. Network interface controllers (NICs) and disks cannot be tagged.
- An application group supports a maximum of 3,000 resources for each Alibaba Cloud service. Resources are added to an application group in a random order. If the number of resources reaches the limit, your resources can no longer be added to the application group.
- You can view the monitoring charts of an application group five minutes after the application group is created
- The system automatically generates alerts based on alert rules five minutes after an application group is created.

Attach the cloudmonitor-group tag to resources

When you create an Alibaba Cloud resource, attach the cloudmonitor-group tag to the resource if you need to manage the resource in the CloudMonitor console. CloudMonitor automatically creates an application group for this tag. You can view the monitoring charts of the application group and manage the resources.

1. Attach the cloudmonitor-group tag when you create resources.
2. View the application group that is automatically created for the tag in the CloudMonitor console.

> **Note** For the automatically created application group, the **Contact Group** parameter is set to **Default Contact Group** and the Template Name parameter is set to **ASF** by default. You can modify the parameters based on business needs.

Specify tags in the CloudMonitor console

If you have attached tags other than cloudmonitor-group to Alibaba Cloud resources, you can create application groups in the CloudMonitor console based on the tags. Then, you can manage resources in the application groups based on your needs.

1. Log on to the CloudMonitor console.
2. Create an application group based on tags.

   > Note After you create the application group, wait 2 minutes and then view the generated application group.

   i. In the left-side navigation pane, select **Application Groups**.

   ii. On the **Application Groups** page, click **Create Group**.

   iii. In the **Create Group** right-side pane, set the parameters in the **Creation method**, **Basic Information**, **MonitorAlarm**, **Region**, **Match Rule**, and **Event Monitor** sections.When you create an application group, set the following parameters:

   - **Creation method**: Select **Smart tag synchronization creation**. The system automatically generates an application group name.
   - **Contact Group**: **Default Contact Group** is selected by default. You can specify an alert group based on business needs.
   - Select Template: **ASF** is selected by default. You can create a template based on business needs.
   - **Resource Tag Key** and **Tag Value**: You can specify a tag key and tag values based on business needs.
   - **Initialize Agent Installation**: If you enable this feature, the system automatically installs the CloudMonitor agent on the servers where resources reside. This feature is enabled by default.

   iv. Click **Add**.

   On the Application grouping tab, you can select **Resource tags** from the drop-down list to filter specified resources.