ALIBABA CLOUD

# Alibaba Cloud

## 云监控

## 最佳实践

文档版本：20201023

阿里云

# 法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

# 通用约定

| 格式 | 说明 | 样例 |
| --- | --- | --- |
| 危险 | 该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。 | 危险<br>重置操作将丢失用户配置数据。 |
| 警告 | 该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。 | 警告<br>重启操作将导致业务中断，恢复业务时间约十分钟。 |
| 注意 | 用于警示信息、补充说明等，是用户必须了解的内容。 | 注意<br>权重设置为0，该服务器不会再接受新请求。 |
| 说明 | 用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。 | 说明<br>您也可以通过按Ctrl+A选中全部文件。 |
| > | 多级菜单递进。 | 单击**设置**> **网络**> **设置网络类型**。 |
| **粗体** | 表示按键、菜单、页面名称等UI元素。 | 在**结果确认**页面，单击**确定**。 |
| `Courier字体` | 命令或代码。 | 执行 `cd /d C:/window` 命令，进入Windows系统文件夹。 |
| *斜体* | 表示参数、变量。 | `bae log list --instanceid`<br>*Instance_ID* |
| [] 或者 [a\|b] | 表示可选项，至多选择一个。 | `ipconfig [-all\|-t]` |
| {} 或者 {a\|b} | 表示必选项，至多选择一个。 | `switch {active\|stand}` |

# 目录

1.报警模板最佳实践 05
2.如何通过钉钉群接收报警通知 07
3.内网监控最佳实践 08
4.如何创建容器服务Kubernetes版的Pod报警规则 09
5.使用API查询监控数据 10
6.ECS状态变化事件的自动化运维最佳实践 14
7.通过标签自动监控资源 21

# 1.报警模板最佳实践

本文旨在通过一个具体案例讲解企业用户如何使用报警模板，高效管理好各业务使用的云资源的报警规则。

## 背景信息

当您的云账号下拥有很多服务器和云产品的资源时，怎样才能快速的为这些资源创建报警规则，在报警规则不合理时修改报警规则。本文将通过具体案例为您讲解大企业用户如何通过使用报警模板和应用分组，提升效率并管理好各业务使用云资源的报警规则。

## 使用报警模板的准备工作

使用报警模板前，我们先了解一下报警规则配置在应用分组和配置在单个实例上的区别以及报警模板如何提升配置规则的效率。

- 报警规则配置在应用分组和配置在单实例上的区别：
  - 创建报警规则时资源范围可以选择“实例”或者“应用分组”，如果选择“应用分组”，那么报警规则的作用范围就是整个应用分组内的所有资源。您的业务需要扩容或者缩容时，只需要将相应资源移入或移出应用分组，而不需要增加或删除报警规则。如果需要修改报警规则，也只需要修改这一条报警规则，就会在组内所有实例上生效。
  - 如果您选择将报警规则创建在实例上，那么该规则只对单一实例有效。修改报警规则时也只对单一实例生效。当实例增多时报警规则会变得难以管理。
- 报警模板极大的提升了配置报警规则的效率。
  - ECS、RDS、SLB等基础服务在配置报警时，监控项和报警阈值相对固定，为这些需要报警的指标建立模板后，新增业务时，创建好应用分组后直接将模板应用在分组上，即可一键创建报警规则。
  - 当您需要批量新增、修改、删除报警规则时，也可以修改模板后，将模板统一应用在分组上，极大的节省操作时间。

## 使用报警模板的实施步骤

### 注意事项

当您的账号下服务器和其他云产品实例非常多时，首先建议您按照业务视角为资源创建不同的应用分组，然后通过应用分组来批量管理资源。

### 操作步骤

下面我们以一个常见的电商网站后台业务为例，介绍如何使用报警模板和应用分组，快速将业务的云上监控报警体系搭建起来。

1. 首先，我们创建一个名为“电商后台模块报警模板”的报警模板。
   i. 登录云监控控制台。
   ii. 单击左侧导航栏中**报警服务**下的**报警模板**，进入**报警模板**页面。
   iii. 单击右上角的**创建报警模板**按钮，进入**创建报警模板**页面。
   iv. 配置模板**基本信息**：输入模板名称和描述。
   v. 配置**报警策略**：选择**产品类型**，添加并设置报警规则，将业务模块需要的报警策略添加到报警模板中。
   vi. 点击**确认**按钮，保存报警模板配置。
2. 创建报警联系人和报警联系组。

i. 登录云监控控制台。

ii. 单击左侧导航栏中**报警服务**下的**报警联系人**，进入**报警联系人管理**页面。

iii. 单击右上角的**新建联系人**按钮，填写手机、邮箱等信息。添加手机和邮箱时需要对手机和邮件进行验证，防止您填写了错误的信息，无法及时收到报警通知。

iv. 在**报警联系人管理**页面，单击页面上方的**报警联系组**页签，切换到报警联系组列表。

v. 单击右上角的**新建联系组**，弹出**新建联系组**页面。

vi. 填写组名并选择需要加入组中的联系人即可。

3. 创建一个名为“库存管理线上环境”的应用分组，并选择刚才创建的报警模板。

i. 登录云监控控制台。

ii. 单击左侧导航栏中的**应用分组**，进入**应用分组**页面。

iii. 单击右上角的**创建组**按钮，进入**创建应用分组**页面。

iv. 配置**基本信息**：输入**应用分组名称**，选择**联系人组**。联系人组即报警联系组，用于接收报警通知。

v. 配置**监控报警**：选择**报警模板**（用于对组内的实例初始化报警规则）和**报警级别**。启用**初始化安装监控插件**，即在新生成ECS 实例后，会对实例安装云监控插件，以便采集监控数据。

vi. 配置**动态添加实例**：库存管理这块业务使用的云资源，我们以最常见的服务器+数据库+负载均衡资源组合为例。通过制定动态匹配规则添加云服务器ECS实例，支持根据ECS实例名称进行字段的“包含”、“前缀”、“后缀”匹配，符合匹配规则的实例会加入到应当前用分组（包含后续新创建的实例）。最多可以添加三条动态匹配规则，规则之间可以是“与”、“或”的关系。点击**添加产品**，可继续制定云数据库RDS版和负载均衡的动态匹配规则。

vii. 点击**创建应用分组**按钮，完成分组的创建。进入到该应用分组详情页，即可看到符合匹配规则的实例已添加到您所创建的应用分组内。

# 2.如何通过钉钉群接收报警通知

本文为您介绍如何设置通过钉钉群接收报警通知。

## 前提条件

进行操作前，请确保您已经注册了阿里云账号。如还未注册，请先完成账号注册。

## 背景信息

云监控新增钉钉群接收报警通知的功能，您可以按照以下指引设置钉钉群接收报警通知。

已经创建的报警规则，只需要在报警联系人中增加钉钉机器人的回调地址，即可收到钉钉群报警，无需修改报警规则。

在已有的报警联系人上新增钉钉机器人后，即可通过钉钉群接收联系人之前通过邮件收到的全部报警规则。

## 创建钉钉机器人（PC版）

1. 在PC版中打开您要接收报警通知的钉钉群。
2. 单击右上角的**群设置**图标，打开**群设置**弹窗。
3. 单击**智能群助手**，单击**添加智能机器人**，打开**群机器人**弹窗。
4. 在**群机器人**弹窗中单击**自定义**，创建一个用于接收报警通知的钉钉机器人。
5. 在**机器人详情**窗口，单击**添加**，进入**添加机器人**窗口。
6. 输入机器人名字，例如云监控报警通知。
7. 安全设置勾选**自定义关键词**，逐个添加5个关键词（阿里云、云服务、监控、Monitor、ECS），单击**完成**。
8. 单击**复制**，复制webhook地址。
9. 单击**完成**。

## 在报警联系人中添加钉钉机器人

1. 登录云监控控制台。
2. 单击左侧导航栏中**报警服务**下的**报警联系人**，进入**报警联系人管理**页面。
3. 单击**编辑**，打开**设置报警联系人**窗口。
4. 在已有联系人中添加钉钉机器人的回调地址，即钉钉机器人webhook地址。

# 3.内网监控最佳实践

本文旨在通过具体案例介绍如何使用云监控实现内网监控的目的。

## 背景信息

随着越来越多的用户从经典网络迁移到更安全、更可靠的VPC网络环境，如何监控VPC内部服务是否正常响应就成为需要关注的问题。本文将通过具体案例说明如何监控VPC内ECS上的服务是否可用、VPC内ECS到RDS、Redis的连通性如何、VPC内SLB是否正常响应。

## 内网监控准备工作

内网监控的原理如下图所示。

首先需要您在服务器上安装云监控插件，然后通过控制台配置监控任务，选择已安装插件的机器作为探测源，并配置需要探测的目标URL或端口。完成配置后，作为探测源的机器会通过插件每分钟发送一个HTTP请求或Telnet请求到目标URL或端口，并将响应时间和状态码收集到云监控进行报警和图表展示。

## 内网监控的实施步骤

> 说明
> - 作为探测源的服务器需要安装云监控插件。
> - 需要创建应用分组，并将作为探测源的服务器加入到分组中。

1. 登录云监控控制台。
2. 在左侧导航栏，单击**应用分组**。
3. 在**应用分组**页面，单击目标**分组名称/分组ID**链接。
4. 在目标应用分组的左侧导航栏，单击**可用性监控**
5. 在**可用性监控**页面，单击**新建配置**。
6. 在**创建可用性监控**页面，设置可用性监控相关参数。
   - 需要监控VPC内ECS本地进程是否响应正常时，可在探测源中选中所有需要监控的ECS，在探测目标中填写*localhost:port/path*格式的地址，进行本地探测。
   - 需要监控VPC内SLB是否正常响应时，可选择与SLB在同一VPC网络内的ECS作为探测源，在探测目标中填写SLB的地址进行探测。
   - 需要监控VPC内ECS后端使用的RDS或Redis是否正常响应时，可将与ECS在同一VPC网络内的RDS或Redis添加到应用分组，并在探测源中选择相应的ECS，在探测目标中选择RDS或Redis实例。
7. 单击**确定**。

   您可以在任务对应的监控图表中查看探测结果，并在探测失败时收到报警通知。

   
8. 单击任务列表中的**监控图表**，可查看监控详情。

   

# 4.如何创建容器服务Kubernetes版的Pod报警规则

云监控通过监控容器服务Kubernetes版的CPU、内存、网络等监控项，帮助您监控其使用情况。您创建容器服务Kubernetes版后，云监控自动对其进行监控，您可以查看其监控详情。您可以对容器服务Kubernetes版的监控项设置报警规则，当监控项超过设定阈值时，云监控自动给您发送报警通知。

## 前提条件

- 请确保您已创建容器服务Kubernetes版，操作方法请参见快速创建Kubernetes托管版集群。
- 请确保您已创建容器服务Kubernetes版所需的应用分组，操作方法请参见创建应用分组。
- 请确保您已创建容器服务Kubernetes版的报警联系人或报警联系组，操作方法请参见创建报警联系人或报警联系组。

## 背景信息

- 监控数据最多保存31天。
- 您最多可连续查看14天的监控数据。

## 操作步骤

1. 创建容器服务Kubernetes版的报警模板。
   i. 登录云监控控制台。
   ii. 在左侧导航栏，单击**报警服务 > 报警模板**。
   iii. 在**报警模板**页面，单击**创建报警模板**。
   iv. 在**创建/修改报警模板**页面，输入模板名称，产品选择**容器服务-Kubernetes版**，设置容器服务Kubernetes版的报警规则。
   v. 单击**确定**。
2. 应用容器服务Kubernetes版的报警模板到应用分组。
   i. 在**创建/修改报警模板完成**对话框中，单击**确定**。
   ii. 在**应用模板到分组**页面，选择应用分组、通道沉默周期、生效时间、报警回调和模板应用方式。
   iii. 单击**确认**。
   iv. 在**应用模板到分组**对话框中，单击**确认**。

## 后续步骤

- 如果您不同的容器服务Kubernetes版的报警通知，需要发送给不同联系人组，则需要修改应用分组关联的报警联系人组。

  修改应用分组中报警联系人组的操作方法，请参见修改应用分组。
- 如果您需要修改多个容器服务Kubernetes版的应用分组的联系人组，可以通过OpenAPI工具进行修改。

  登录OpenAPI Explorer，调用PutContactGroup接口修改报警联系人组。

  PutContactGroup接口中参数的设置方法，请参见PutContactGroup。

# 5.使用API查询监控数据

本文为您介绍如何使用API查询阿里云各产品监控数据。

大型企业内部通常有自建的运维监控系统，上云过程中会面临如何将云资源监控数据与已有系统集成的问题。下面本文将为您介绍如何通过云监控接口查询各产品监控数据，从而将阿里云的监控数据与现有系统进行集成。

## 指标类监控数据查询的接口

云监控提供以下3类接口用于指标类监控数据的查询：

- 查询产品列表接口：查询云监控支持哪些产品的监控项，详情请参见DescribeProjectMeta。
- 查询监控项列表接口：查询对应产品可以获取哪些监控项，详情请参见DescribeMetricMetaList。
- 查询监控数据接口：根据产品信息和监控项信息，查询具体的监控数据，详情请参见DescribeMetricList和DescribeMetricLast。

注意事项：

- DescribeMetricList和DescribeMetricLast接口支持批量获取用户下所有实例的某个指标的数据。如果想获取多个指标，可以多个线程获取多个指标，也可以单线程循环获取多个指标。
- DescribeMetricList接口支持的最大QPS是20，DescribeMetricLast接口的最大QPS是30。
- DescribeMetricLast接口适用于需要定时全量拉取所有最新数据的情况。时间窗口自动往前滑动，每个周期都取一条最新数据。
- 监控数据会有一定的延迟，且各产品的监控数据的延迟情况不太一样，所以建议您使用DescribeMetricLast查询最新数据时，时间窗口放宽到5-10分钟。
- 秒级精度的数据保存7天，分钟级精度的数据保存31天。
- 如果您需要查询云账号所有实例的数据，则不需要指定Dimensions。

## 实战案例

通过Demo演示，为您介绍如何使用DescribeMetricLast接口查询最新的监控数据，使用DescribeMetricList接口查询指定时间段内的监控数据。

```
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.*;

/**
 * 使用DescribeMetricList接口可以查询指定时间段内，指定实例的监控数据。
 * 该查询允许指定多个实例进行批量查询。
 * 如果需要获取多个实例一段时间内的监控数据，可以在查询时指定多个实例，每次最多10个实例。
 * 查询一段时间内的监控数据
 */
```

```
 */
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();
        //namespace和metric通过DescribeMetricMetaList和DescribeProjectMeta获取
        request.setNamespace("acs_ecs_dashboard");
        request.setMetricName("cpu_total");
        //period表示要获取60s精度的监控数据。period根据每个metric有不同的定义，大部分metric都会有60s的per
iod。
        request.setPeriod("60");
        //本次查询的分页长度，每次查询最多返回1000条数据。
        request.setLength("1000");
        //查询数据的开始时间
        request.setStartTime("2019-07-22 11:00:00");
        //查询数据的结束时间
        request.setEndTime("2019-07-22 12:00:00");
        //查询的关联dimension过滤，即可以是一个JSONArray，也可以是一个JSONObject
        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.*;

/**
 * 取得最后一条监控数据
 **/
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();

        //namespace和metric通过DescribeMetricMetaList和DescribeProjectMeta获取
        request.setNamespace("acs_ecs_dashboard");
        request.setMetricName("cpu_total");
        //查询的关联dimension过滤，即可以是一个JSONArray，也可以是一个JSONObject
        request.setDimensions("[{\"instanceId\":\"i-8vb6p*****\"}]");
        //本次查询的分页长度，每次查询最多返回1000条数据。
        request.setLength("1000");
        //查询数据的开始时间
        request.setStartTime("2019-07-22 11:00:00");
        //查询数据的结束时间
        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.ECS状态变化事件的自动化运维最佳实践

阿里云ECS在已有的系统事件的基础上，通过云监控新发布了状态变化类事件和抢占型实例的中断通知事件。当ECS实例的状态发生变化时，会触发一条ECS实例状态变化事件。这种变化包括您在控制台、OpenAPI和SDK操作导致的变化，也包括弹性伸缩或欠费等原因而自动触发的变化，还包括因系统异常而触发的变化。

## 背景信息

云监控之前发布的系统事件，主要针对告警后人工介入的场景，而本次新发布的事件属于正常类的信息通知，适合自动化的审计运维场景。为了自动化处理ECS状态变化事件，云监控提供了两种主要途径：一种是通过函数计算，另一种是通过MNS消息队列。

## 自动化处理ECS状态变化事件的准备工作

## 创建消息队列

- 创建消息队列
  i. 登录MNS控制台。
  ii. 在**队列**页面，选择地域，单击右上角的**创建队列**，进入**新建队列**页面。
  iii. 输入队列名称（例如：ecs-cms-event）等信息，单击**确认**，即可完成创建消息队列。
- 创建事件报警规则
  i. 登录云监控控制台。
  ii. 在左侧导航栏，单击**事件监控**。
  iii. 在**事件监控**页面，单击**报警规则**页签，单击右上角的**创建事件报警**。
  iv. 在**基本信息**区域，填写报警规则名称，例如：ecs-test-rule。
  v. 在**事件报警规则**区域，相关参数设置如下：
    - **事件类型**选择**系统事件**。
    - **产品类型**选择**云服务器ECS**。
    - **事件类型**选择**状态通知**。
    - **事件名称**按照实际情况选择。
    - 当**资源范围**选择**全部资源**时，任何资源发生相关事件，都会按照配置发送通知；当**资源范围**选择**应用分组**时，只有指定应用分组内的资源发生相关事件，才会按照配置发送通知。
  vi. 在**报警方式**区域，相关参数设置如下：
    - **联系人组**和**通知方式**按照实际情况选择。
    - 报警数据写入方式选择**消息服务队列**，**地域**和**队列**按照实际情况选择（例如：ecs-cms-event）。
  vii. 单击**确定**。
- 安装Python依赖

  本文所有的代码均使用Python 3.6测试通过，您也可以使用Java等其他编程语言。

  请使用PyPI（Python Package Index）安装以下Python依赖：

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

## 自动化处理ECS状态变化事件的实施步骤

云监控会将云服务器ECS所有的状态变化事件投递到MNS中，再通过编写代码从MNS获取消息并进行消息处理。

- 实践一：对所有ECS的创建和释放事件进行记录

  目前ECS控制台无法查询已经释放的实例。如果您有查询需求，可以通过ECS状态变化事件将所有ECS的生命周期记录在自己的数据库或日志中。当您创建ECS时，会发送一个Pending事件，当您释放ECS时，会发送一个Deleted事件，云监控对这两种事件进行记录。

  i. 编辑一个Conf文件。

  Conf文件需包含MNS的如下信息：

  - `endpoint`：在MNS控制台的队列页面，单击获取Endpoint。
  - `access_key` 和 `access_key_secret`：在用户信息管理控制台中获取。
  - `region_id` 和 `queue_name`：在MNS控制台的队列页面，查看队列名称和所属地域。

```
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. 使用MNS的SDK编写一个MNS Client用来获取MNS消息。

```
# -*- 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
```

上述代码只对MNS消息获取的数据，调用Listener消费消息之后删除消息。

iii. 注册一个指定Listener消费事件。这个简单的Listener判断收到Pending和Deleted事件时，打印一行日志。

```
 # -*- 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}')
```

Main函数写法如下：

```
mns_client = MNSClient()


mns_client.regist_listener(ListenerLog())


mns_client.run()
```

实际生产环境下，可能需要将事件存储在数据库里，或者使用日志服务（SLS），方便后期的搜索和审计。

- 实践二：ECS关机自动重启

  在某些场景下，ECS会非预期的关机，您可能需要自动重启已经关机的ECS。

  为了实现ECS关机自动重启，您可以复用实践一里面的MNS Client，添加一个新的Listener。当收到Stopped事件时，对该ECS执行命令start。

```
# -*- 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

    # 启动ECS实例
    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)
```

在实际生产环境下，执行完start命令后，可能还需要继续接收后续的Starting、Running或Stopped等事件，再配合计时器和计数器，进行start成功或失败之后的处理。

- 实践三：抢占型实例释放前，自动从负载均衡（SLB）移除

  抢占型实例在释放之前五分钟左右，会发出释放告警事件，您可以利用这短暂的时间运行一些业务不中断的逻辑。例如，主动从SLB的后端服务器中去掉这台即将被释放的抢占型实例，而不是被动等待实例释放后SLB的自动处理。

  您复用实践一的MNS Client，添加一个新的Listener，当收到抢占型实例的释放告警时，调用SLB的SDK。

```
# -*- 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)
```

> **注意**
>
> 抢占型实例释放告警的 `event name` 与前面不同，应该是 `mns_client.regist_listener(ListenerSLB(Conf.vsever_group_id), 'Instance:PreemptibleInstanceInterruption')` 。

**在实际生产环境下，您需要再申请一台新的抢占型实例，挂载到SLB上，来保证服务能力。**

# 7.通过标签自动监控资源

对于大型企业或组织而言，可能需要维护成千上万条资源。即使对资源进行分组，也会有上千个组。人工维护耗时费力且易出错。云监控支持对资源绑定标签，并根据标签将资源分类管理，实现基于标签的自动化监控，降低您的资源监控成本。

## 前提条件

- 请确保您已完成阿里云账号注册和实名认证。
- 请确保您在阿里云其他产品上创建资源时，已根据实际业务管理需求绑定标签。

## 背景信息

云监控基于标签管理资源的限制如下：

- 目前只支持云服务器ECS（只支持实例，不支持网卡，磁盘等）、云数据库RDS和负载均衡SLB。
- 一个应用分组中每个产品最多只支持3000条资源，且资源加入分组的顺序是随机的。超出的资源不会加入分组。
- 创建应用分组约5分钟后，您可以查看分组级别的数据图表。
- 创建应用分组约5分钟后，系统自动按照报警规则上报告警。

## 资源绑定固定标签

当您在阿里云的不同产品上创建资源时，如果确认该资源需要通过云监控控制台进行管理，则需要在该资源上绑定标签：cloudmonitor-group。针对该标签，云监控都会自动创建一个应用分组。在分组中，您可以查看资源监控图表，并对资源进行管理。

1. 在阿里云产品上创建资源时，绑定标签：cloudmonitor-group。
2. 云监控根据标签在云监控控制台上自动创建一个应用分组。

> **说明** 对于自动创建的应用分组，**联系人组**默认是**云账号报警联系人**，监控模板默认是**常用基础模板**。您可以根据实际业务需求手动修改。

## 在云监控控制台上指定标签

如果您在阿里云的不同产品上创建资源且绑定了非cloudmonitor-group的标签时，可以在云监控控制台上通过手动创建应用分组来指定标签，对该标签下的资源进行定制化管理。

1. 登录云监控控制台。
2. 创建应用分组指定标签。

> **说明** 创建应用分组完成后，请您稍等2分钟再查看生成的应用分组。

   i. 在左侧导航栏选择**应用分组**。
   ii. 在**应用分组**页面，单击**创建组**。

iii. 在**创建应用分组**页面，配置**创建方式**、**基本信息**、**监控报警**、**区域**、**匹配规则**和**事件监控**。创建应用分组时，参数配置说明如下：

- **创建方式**选择**智能标签同步创建**，系统自动生成**应用分组名称**。
- **联系人组**默认是**云账号报警联系人**，您可以根据实际业务需求自定义。
- 监控报警模板默认是**常用基础模板**，您可以根据实际业务需求自定义。
- 您可以根据实际业务需求配置**资源标签键**和**资源标签值**。
- 启用**初始化安装监控插件**，系统会对本应用分组的服务器批量安装监控插件，默认处于启用状态。

iv. 单击**添加**。

通过**资源标签**过滤出指定的标签值。