ALIBABA CLOUD

阿里云

微服务引擎
微服务治理

文档版本：20220708

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 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 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 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 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` *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.应用信息

1.1. 查看应用详情

您可以通过MSE治理中心控制台查看已接入MSE治理中心的Spring Cloud和Dubbo应用详情。

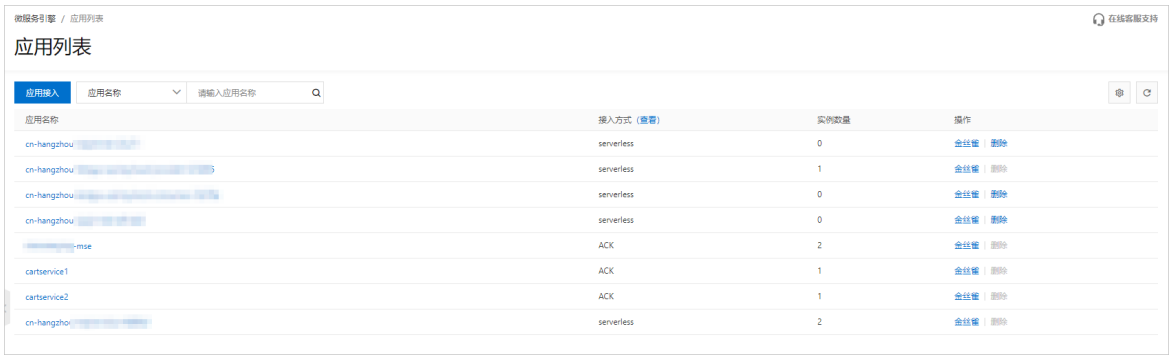
前提条件

MSE治理中心已接入微服务应用，相关内容，请参见：

- ACK微服务应用接入MSE治理中心
- ECS微服务应用接入MSE治理中心

查看微服务应用列表

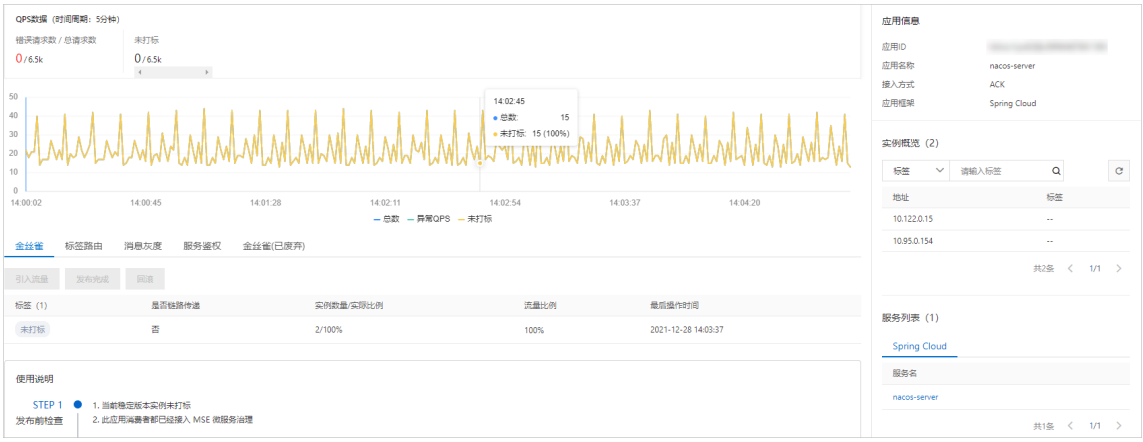
1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
3. 在顶部菜单栏选择地域。
4. 在**应用列表**页面查看已开启微服务中心的应用的相关信息，包括**应用名称**、**接入方式**和**实例数量**。



您可以单击**应用接入**或**接入方式**右侧的**查看**，查看**容器服务K8s集群**、**ECS集群**、**SAE指导**、**企业级分布式应用服务EDAS**、**服务网格**等接入方式。

查看微服务应用详情

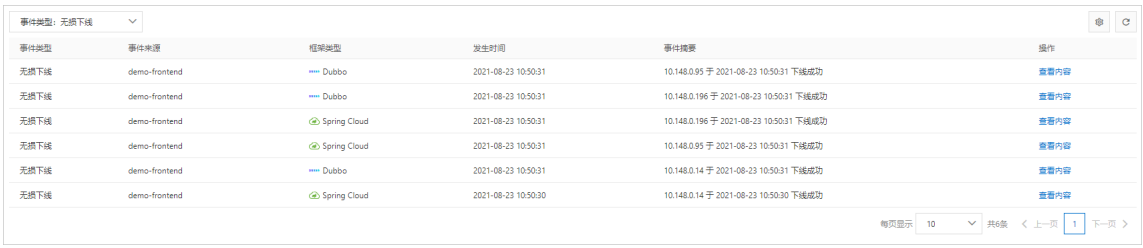
1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
3. 在顶部菜单栏选择地域。
4. 在**应用列表**页面单击目标应用名称，可查看应用详情。
5. 您可根据需求选择以下操作。
   - 在左侧导航栏单击**应用详情**，可查看应用的**QPS数据**，查看当前应用的请求数。



您可以在**应用详情**页面右侧查看应用的**基本信息**、**实例概览**和**服务列表**。
您也可以在**应用详情**页面设置**金丝雀**、**标签路由**、**消息灰度**、**无损上下线**、**服务鉴权**、**推空保护**路由等规则。具体操作，请参见：

- 配置金丝雀发布
- 配置标签路由
- 配置消息灰度
- 配置无损上线
- 配置无损下线
- 配置服务鉴权
- 配置推空保护

- 在左侧导航栏单击**接口详情**，可查看应用的接口详情，您可以单击**Spring Cloud**页签，查看Spring Cloud应用下的所有请求路径；单击**Dubbo**页签，查看Dubbo应用下的所有服务。
  选中某个接口，在右侧会显示该接口的**接口概览**和**节点详情**。
  - 单击**接口概览**页签，显示该接口的类名、方法、参数、返回值类型、QPS数据、RT数据等。
    单击右上方的**测试**，在**选择测试方法**面板中配置测试参数进行测试，具体操作，请参见服务测试。
  - 单击**节点详情**页签，在**筛选节点**下拉框中选择要查看的节点，会显示该节点的QPS数据。
- 在左侧导航栏单击**事件中心**，可查看应用事件的**事件类型**、**事件来源**、**框架类型**、**发生时间**以及**事件摘要**。其中事件类型包括**离群摘除**、**摘除恢复**、**无损下线**、**推空保护**、**手动上线**以及**手动下线**。



单击**操作**列下方的**查看内容**，可弹窗显示**事件来源**、**发生时间**、**事件摘要**、**详细内容**等。

1.2. 动态配置超时

MSE提供了动态的方法级的超时配置能力，帮助您在日常业务逻辑迭代中可以根据接口响应时间的变化快速调整，提高服务的治理能力。本文介绍如何动态配置Dubbo服务的超时。

前提条件

- 请确保相关的应用都已接入MSE治理中心，具体操作，请参见微服务治理中心入门概述。
- 请确保在MSE治理中心能查询到相应的服务信息，具体操作，请参见查询服务。

背景信息

在日常工作中会遇到各类超时配置，业务逻辑变更后，已有调用关系随着业务发展可能需要不断调整，相应服务接口响应时间的变化可能需要上线后才能确定。MSE的动态配置超时功能为Dubbo服务接口、方法提供了灵活的超时配置能力，能够帮助您快速动态调整接口超时时间，提高服务的可用性。

视频教程

操作步骤

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 服务查询**。
3. 在**服务查询**页面选择**框架：Dubbo**，然后单击具体的Dubbo服务名。
4. 在**服务详情**面板中，单击**超时配置**区域的**添加**按钮。在**添加超时配置**面板中配置相关参数，然后单击**确定**。

   超时配置参数说明如下。

参数	描述
服务方法	选择配置当前服务的方法，星号（*）表示所有服务接口。
针对消费者应用	选择针对当前服务的消费者应用，星号（*）表示所有消费者应用。
超时时间	设置调用的超时时间，超时时间应设置为大于0的整数，单位：ms。此配置优先级高于其他同级别配置。

   超时配置优先级关系参考：

   - 相较于其他配置优先级：MSE治理中心的方法级配置>客户端及其他来源的方法级配置>MSE治理中心的接口级配置>客户端及其他来源的接口级配置。
   - 相较于自身配置优先级：
     - 指定服务方法的配置>所有服务方法（即选择星号（*））。
     - 指定消费者应用的配置>所有消费者应用的配置（即选择星号（*））。
     - 服务方法和消费者应用配置相同的情况下，新建配置>旧配置。

   超时配置添加成功后，可在**服务详情**页面的**超时配置**区域列表中查看。

结果验证

选择和超时配置相关的消费者应用，触发该调用验证。

> **说明** 该调用的首次超时配置可能需要多次调用才能验证。

为使效果更直观明显，可以选择不影响业务的调用关系设置极小的阈值触发异常查看。

相关操作

超时配置记录支持**添加**和**删除**操作，具体超时阈值的修改可以通过先增加新记录再删除旧记录来实现。

1.3. 查询服务

您可以通过微服务中心MSC查询部署的Spring Cloud或Dubbo应用的服务列表和服务详情。

查看服务列表

1.
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 服务查询**。
3. 在顶部菜单栏选择地域。
4. 在**服务查询**页面，通过左上角的下拉框选择**框架：Spring Cloud**或**框架：Dubbo**来查看目标服务。

   如果服务较多，可以通过**服务名**、**IP**或**应用名**进行筛选或搜索，关键字的大小写不敏感。其中**IP**会因ECS集群和容器服务K8s集群有所不同。

   - ECS集群：IP为应用实例（ECS）的IP地址。
   - 容器服务K8s集群：IP为应用实例（Pod）的IP地址。

查看服务详情

1.
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 服务查询**。
3. 在顶部菜单栏选择地域。
4. 在**服务查询**页面单击具体服务名来查看服务的详细信息。

   服务详情页面包含**基本信息**、**服务调用关系**和**元数据**。

   - Spring Cloud的服务详情如下：
     - **基本信息**包含**服务名称**、**服务类型**和**应用名**。
     - **服务调用关系**包含**服务提供者**和**服务消费者**列表及其**IP**和**端口**信息。
     - 元数据包含**接口元数据**和**Metadata元数据**。
       - **接口元数据**：包含**所属类**、**请求方法**、**请求路径**、**方法名/描述**和**参数列表/描述**。
       - **Metadata元数据**：包含服务的元数据，还包含MSC提供的一些用于使用微服务能力的元数据。
   - Dubbo的服务详情如下：
     - **基本信息**包含**服务名称**、**版本**、**分组**、**服务类型**和**应用名**。
     - **服务调用关系**包含**服务提供者**和**服务消费者**列表及其**IP**、**端口**、**序列化方式**和**超时时间（ms）** 信息。
     - 元数据包含**Metadata元数据**和**接口元数据**。
       - **接口元数据**：包含**方法名**、**参数列表**和**返回类型**。
       - **Metadata元数据**：包含服务的元数据，还包含MSC提供的一些用于使用微服务能力的元数据。

1.4. 查询服务契约

服务契约指基于OpenAPI规范的微服务接口描述，是微服务系统运行和治理的基础。您无需在应用中引入依赖，直接部署后，便可以通过服务契约在线查看微服务接口、路径等API信息，不但能查询提供的服务，还能方便的使用服务测试等功能。

背景信息

服务契约包含了以下3个主要功能：

- API查询

查看服务提供者或消费者的重要API信息，包括方法名、参数列表、返回类型。Spring Cloud服务还支持查看请求方法、请求路径、所属类的类名等信息。

- Swagger注解解析
  作为OpenAPI规范的主要制定者，Swagger虽并非是唯一支持OpenAPI的工具，但基本也属于一种事实标准。服务契约支持Swagger注解解析，并在控制台的服务契约页面进行展示：
  - Swagger2的注解解析（例如@ApiOperation，@ApiParam，@ApiImplicitParam），解析value值在**描述**列展示。
  - OpenAPI3的注解解析（例如@Operation，@Parameter），解析description值在**描述**列展示。
- 服务测试的前置条件
  服务测试功能需要基于通过服务契约收集的服务的API信息，对服务接口或路径进行测试。

视频教程

操作步骤

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
3. 在顶部菜单栏选择地域。
4. 在**应用列表**页面单击目标应用名称。
   如果应用较多，可以通过**应用名称**进行模糊搜索。
5. 在左侧导航栏单击**接口详情**，可查看应用接口的**请求路径**和**接口描述**。
6. 在左侧单击请求路径名称，系统会自动在右侧显示该**接口概览**详情，包括**类名**、**方法**、**参数**、**返回值类型**、**QPS数据**、**RT数据**等。
   当使用Swagger注解时，会在**方法**和**参数**后面显示相应信息。

1.5. 应用配置

1.5.1. 什么是应用配置

应用配置是一个轻量级的动态配置框架，通过应用配置可以动态管理代码中的配置项，根据需求为某个应用开启或关闭部分功能，或设置某个性能指标的阈值。

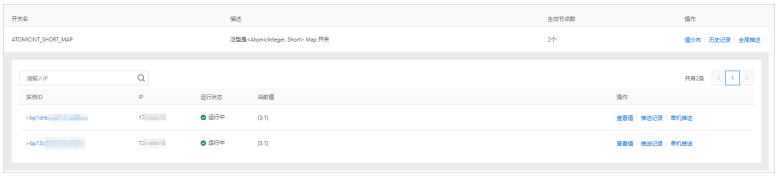
背景信息

> **说明** 目前应用配置处于灰度状态，如果您对这些功能有诉求，您可以按照ACK微服务应用接入MSE服务治理企业版操作，通过增加白名单方式体验该功能。

通常业务代码中包含许多的配置项，这些配置项用于控制各种各样的业务逻辑，例如一个bool类型的变量控制某个功能是否开启，一个list控制访问白名单或黑名单，一个String控制提示信息。开发者通常希望可以动态、实时地去查看和修改配置项，并且期望不需要编写额外的代码来管理，此时就可以利用MSE应用配置来实时修改和查看对应的配置项。与传统的配置中心不同，开发者使用MSE应用配置时，无需关注配置项的解析逻辑，只需声明对应的变量，加上MSE应用配置的注解即可在应用配置控制台对配置进行动态管理。

主要功能

- **查看应用配置**：在MSE治理中心控制台应用配置中，可以直观查看应用中包括哪些配置，具体操作步骤，请参见查看应用配置。



- **查看配置值分布**：在MSE治理中心控制台应用配置中，可以直观地查看对应配置值信息和分布信息，具体操作步骤，请参见查看应用配置。

- **设置配置推送**：在MSE治理中心控制台应用配置中，设置配置的推送值，推送成功后，业务代码里会实时生效。具体操作步骤，请参见设置配置推送。

> **说明** 应用配置还支持灰度分批推送，您可以先在一批机器验证后再全局发布，防止预期外的变更导致线上故障。

例如在大促到来的时候，可以通过配置将非核心的业务逻辑降级，减少一些非必要的资源消耗。操作流程可参考以下示例：

1. 在代码中增加核心业务配置、植入埋点和业务逻辑。
2. 在MSE治理中心控制台**应用配置**中查看业务配置的信息和值分布。
3. 在MSE治理中心控制台**应用配置**中将此配置的**推送值**设为**true**。
4. 在控制台上修改配置项，推送成功后，业务代码里会实时生效。代码中的此配置变量即变为 `true` 。即动态实时的通过应用配置控制业务逻辑。

注意事项

在有些IDE中，尤其是使用Spring Boot技术栈的时候，SwitchManager和用户自己的代码使用的是不同的ClassLoader加载的，会导致应用配置在云端修改后，在用户的工程中由于不同ClassLoader的问题取不到最新修改的值。

1.5.2. 新增功能开关

一个业务通常由多个系统、多个功能模块组成，为保证某些业务的动态性，后端程序通常会用开关来控制程序的逻辑，以达到在系统运行时切换运行逻辑的目的。本文介绍如何新增功能开关。

前提条件

您已接入新应用，详情请参见使用SDK接入和使用Spring Boot Starter接入。

通过 Java SDK 接入

通过 Java SDK 接入的应用请参见以下步骤新增功能开关。

1. 定义功能开关。

   在字段上加上 `com.taobao.csp.switchcenter.annotation.AppSwitch` 注解，字段修饰符必须为 `public static` 。
   例如以下代码：

```
public class CommonTypeSwitch {
    @AppSwitch(des = "String 类型开关", level = Level.p2)
    public static String stringSwitch = "string";
    @AppSwitch(des = "Integer 类型开关", level = Level.p1)
    public static Integer integerSwitch = 2;
    @AppSwitch(des = "Boolean 类型开关", level = Level.p4)
    public static Boolean booleanSwitch = true;
    @AppSwitch(des = "AtomicInteger 类型开关", level = Level.p1)
    public static AtomicInteger atomicIntegerSwitch = new AtomicInteger(21);
    @AppSwitch(des = "AtomicBoolean 类型开关", level = Level.p1)
    public static AtomicBoolean atomicBooleanSwitch = new AtomicBoolean(true);
    @AppSwitch(des = "AtomicLong 类型开关", level = Level.p1)
    public static AtomicLong atomicLongSwitch = new AtomicLong(4L);
    @AppSwitch(des = "泛型为 String List 类型开关", level = Level.p1)
    public static List<String> stringListSwitch = new ArrayList<String>();
    @AppSwitch(des = "泛型是<Integer, String> Map 开关", level = Level.p4)
    public static Map<Integer, String> INT_STRING_MAP = new HashMap<Integer, String>();
    @SuppressWarnings("deprecation")
    @AppSwitch(des = "Date类型开关", level = Level.p1)
    public static Date dateTypeSwitch = new Date(114, 6, 3);
    @AppSwitch(des = "BigInteger类型开关", level = Level.p1)
    public static BigInteger bigIntegerTypeSwitch = BigInteger.valueOf(38888);
    @AppSwitch(des = "BigDecimal类型开关", level = Level.p1)
    public static BigDecimal bigDecimalTypeSwitch = BigDecimal.valueOf(3.00000000001);
    @AppSwitch(des = "枚举类型开关", level = Level.p1)
    public static EnumType enumTypeSwitch = EnumType.ITEM1;
    @AppSwitch(des = "泛型为List<Integer>的LinkedList", level = Level.p1)
    public static List<List<Integer>> LIST_INT_LINKEDLIST = new LinkedList<List<Integer>>()
;
    @AppSwitch(des = "泛型为<Map<String, Integer>, Map<String, Integer>>的HashMap", level =
Level.p1)
    public static Map<Integer, Map<String, Map<String, Integer>>> MAP_MAP_HASHMAP = new Has
hMap<Integer, Map<String, Map<String, Integer>>>();
}
```

2. 调用注册方法进行注册。

```
/*
 应用调用此方法完成注册，同时请保证应用在启动的时候，调用过且知道用过一次此方法，多次调用会抛出异常。
 应用名称可不填，不填取 project.name 启动参数的值其中，常量类参数是可变参数，可注册多个常量类。
 如常量类未添加 com.taobao.csp.switchcenter.annotation.NameSpace 注解，默认使用完整类路径名作为
namespace。
*/
SwitchManager.init("appName", CommonTypeSwitch.class);
```

3. 配置启动参数。

- 非公网

```
//将 AppName 替换为自定义的应用名称。
ahas.namespace=default
project.name=AppName
```

- 公网

```
//将 AppName 替换为自定义的应用名称，将 <license> 替换为真实值。
ahas.namespace=default
project.name=AppName
ahas.license=<license>
```

> **说明** 仅公网环境接入需要 License，您可在**新应用接入**页面查看并保存 License，详情请参见查看并保存 license。

4. 重新部署您的应用。

通过 Spring Boot 接入

通过 Spring Boot 接入的应用请参见以下步骤新增功能开关。

1. 定义功能开关。
   - 在相关开关类上加上 `@Switch` 注解。
   - 在相关常量类上加 `com.alibaba.csp.ahas.switchcenter.anotation.Switch` 注解，同时在对应字段上加 `com.taobao.csp.switchcenter.annotation.AppSwitch` 注解，字段修饰符必须为 `public static` 。

```
@Switch
public class SwitchConfig {
    @AppSwitch(des = "Boolean 类型开关", level = Level.p2, callback = TestCallback.class)
    public static boolean test_switch = false;
}
```

2. 配置启动参数。

   在 application.properties 中添加以下配置项。

   - 非公网

```
#指定您要接入的特定的 AHAS 环境。
ahas.namespace=default
#自定义您的应用名称。
project.name=AppName
```

   - 公网

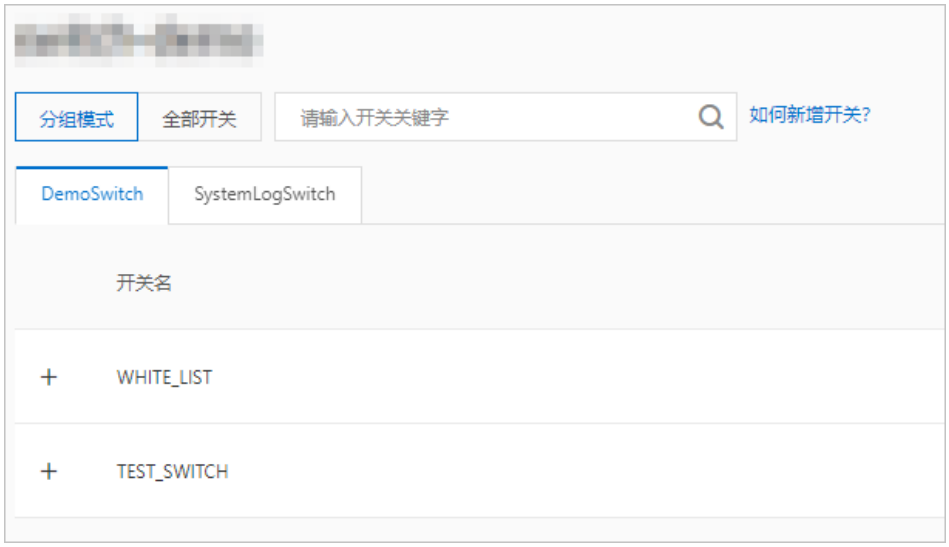
```
#指定您要接入的特定的 AHAS 环境。
ahas.namespace=default
#自定义您的应用名称。
project.name=AppName
#配置 License 信息。
ahas.license=<license>
```

> **说明** 仅公网环境接入需要 License，您可在**新应用接入**页面查看并保存 License，详情请参见查看并保存 license。

3. 重新启动您的应用。

执行结果

添加完成后，在**功能开关**页面单击目标应用的资源卡片，进入目标应用的**开关列表**页面，可查看到新增开关的相关信息。



更多信息

如果您需要自定义功能开关的分组，可在代码中添加 `com.taobao.csp.switchcenter.annotation.NameSpace` 注解；如果没有自定义，分组类别默认取 class 后面的类名，请参见以下示例。

```
package com.taobao.csp.switchcenter.example;
import com.taobao.csp.switchcenter.annotation.AppSwitch;
import com.taobao.csp.switchcenter.annotation.NameSpace;
import com.taobao.csp.switchcenter.bean.Switch.Level;
  @NameSpace(nameSpace = "customNamespace") //customNamespace为自定义分组名。
  public class PrimitiveTypeSwitch {       //PrimitiveTypeSwitch为默认分组名。
  @AppSwitch(des = "int 类型开关", level = Level.p1)
  public static int primitiveIntSwitch = 1;
  @AppSwitch(des = "doubel 类型开关", level = Level.p1)
  public static double primitiveDoubleSwitch = 1.121;
}
```

> **说明** 其中 `com.taobao.csp.switchcenter.bean.Switch.Level` 指开关的重要程度，分为p1、p2、p3、p4四个档位，p1的重要程度最高，p4的重要程度最低。

1.5.3. 管理应用配置

1.5.3.1. 查看应用配置

本文介绍如何查看应用配置的相关信息，包括配置类型、生效节点数、使用实例ID和IP等信息。

前提条件

您已成功新增应用配置，请参见新增功能开关。

操作步骤

1. 登录MSE治理中心控制台，在页面左上角选择**地域**。
2. 在控制台左侧导航栏选择**微服务治理中心 > 应用配置**，在**应用配置**页面单击目标应用**操作**列下方的**应用配置**。

   进入目标应用的**配置列表**页面，查看配置的描述、生效节点数、使用的实例ID、IP等信息。
   在**配置列表**页面，列表展现方式有**分组模式**和**全部配置**模式。
   - **分组模式**：所有的配置按照 `NameSpace` 进行分组。
   - **全部配置**模式：直接展示所有的配置。
3. 单击**操作**列的**值分布**，即可查看对应配置信息和分布信息，包括值编号、配置值等。

参数名	说明
配置名	自定义的配置名，可参见新增功能开关。
描述	配置的备注信息，记录配置的用途，即注解中引号中的内容。例如 `@AppSwitch(des = "String 类型配置")` 。
生效节点数	表示此配置生效的节点个数。
实例ID	表示此配置被引用的实例ID节点。
运行状态	运行状态指的是节点的运行状态，包括**运行中**和**已停机**。
当前值	推送时输入的推送值。

4. 单击实例后**操作**列下的**查看值**，也可查看当前节点上此配置的值信息。

1.5.3.2. 设置配置推送

无需写死的URL、接口名、阈值和读取文件用的编码、黑白名单等，您可以直接使用应用配置设置推送值，快速创建运行时能覆盖的动态配置。应用配置支持全局推送、单机推送和灰度推送。本文介绍如何使用配置推送功能。

前提条件

您已成功新增应用配置，请参见新增功能开关。

操作步骤

1. 登录MSE治理中心控制台，然后在页面左上角选择**地域**。
2. 在控制台左侧导航栏选择**微服务治理中心 > 应用配置**，在**应用配置**页面单击目标应用**操作**列下方的**应用配置**，进入目标应用的**配置列表**页面。
3. 单击**配置列表**页面**操作**列的**全局推送**或**单机推送**，在右侧弹出**配置推送**页面，在此页面中可查看配置名、namespace、配置类型等信息，也可以编辑**推送值**。

   > **说明** 配置的推送类型需在代码中定义，详情请参见变更回调。

4. 编辑完成推送值后，单击**下一步：值对比**，会显示出修改点。若还需修改，则单击**上一步：返回修改**，若修改完成，则单击**单机推送**或**全局推送**。
5. （可选）设置配置的灰度推送。

   i. 在目标配置的**操作**列单击**全局推送**，进入**配置推送**页面，在此页面中编辑推送值。

ii. 单击左下角的**灰度推送**，弹出**灰度推送设置**页面。

iii. 设置**灰度批次**，选择**是否多次暂停**。然后单击**开始灰度**。

灰度推送即分批推送，可先推送一批机器试看推送效果，防止因全量推送而引起应用故障。

- **灰度批次**：指推送的批次数，范围为2至机器总数。每批的机器数为总机器数/批次数。按机器顺序推送，同一批次内推送机器并行，多批次间按顺序推送。例如有10台机器，**灰度批次**设为3，则先推送前3台机器，再推送3台机器，最后再推送4台机器。
- **是否多次暂停**：仅第一批暂停，表示推送完第一批机器数后暂停推送，待单击**继续推送**后，再继续推送。也可以设置为每批都暂停。

1.5.3.3. 历史记录

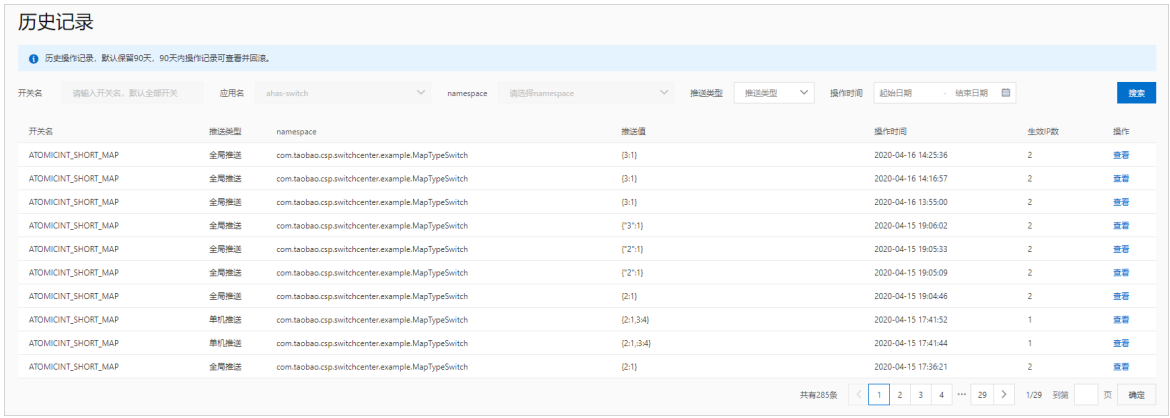
本文介绍如何查询应用配置推送的历史记录，包括推送的类型、操作时间等信息。

前提条件

您已成功新增应用配置，具体详情请参见新增功能开关。

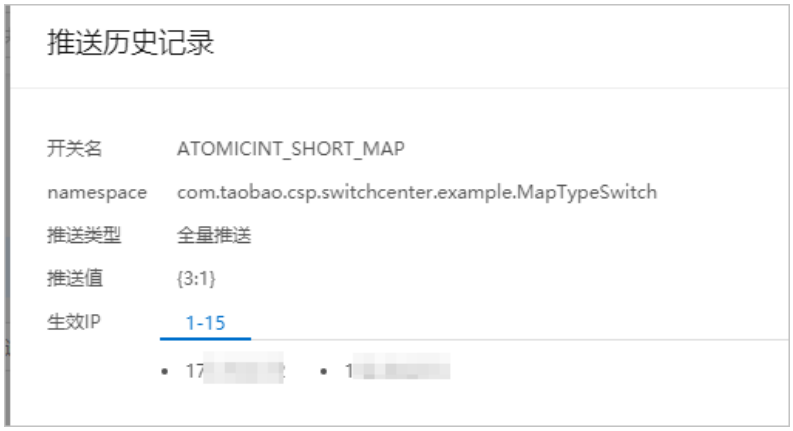
操作步骤

1. 登录 MSE治理中心控制台，然后在页面左上角选择**地域**。
2. 在控制台左侧导航栏选择**微服务治理中心 > 应用配置**，在**应用配置**页面单击目标应用**操作**列下方的**应用配置**。
   进入目标应用的**配置列表**页面。
3. 在左侧导航栏单击**历史记录**。在**历史记录**页面展示了所有配置90天内的推送记录，包括**配置名**、**推送类型**等信息。
4. 按**配置名**、**推送类型**、**操作时间**等条件过滤。



参数名	说明
推送类型	包括全局推送和单机推送。 ○ 全局推送为持久化推送，即重启服务之后，配置仍然生效。 ○ 单机推送为内存化推送，重启之后将失效。
推送值	当前配置的推送值。
操作时间	默认保留90天的操作记录，90天内的操作记录可以查看。

5. 单击**操作**列下的**查看**，可以查看此条推送记录的配置名、namespace、推送类型、推送值和生效IP。


推送历史记录
开关名
ATOMICINT_SHORT_MAP
namespace
com.taobao.csp.switchcenter.example.MapTypeSwitch
推送类型
全量推送
推送值
{3:1}
生效IP
1-15

2.流量治理

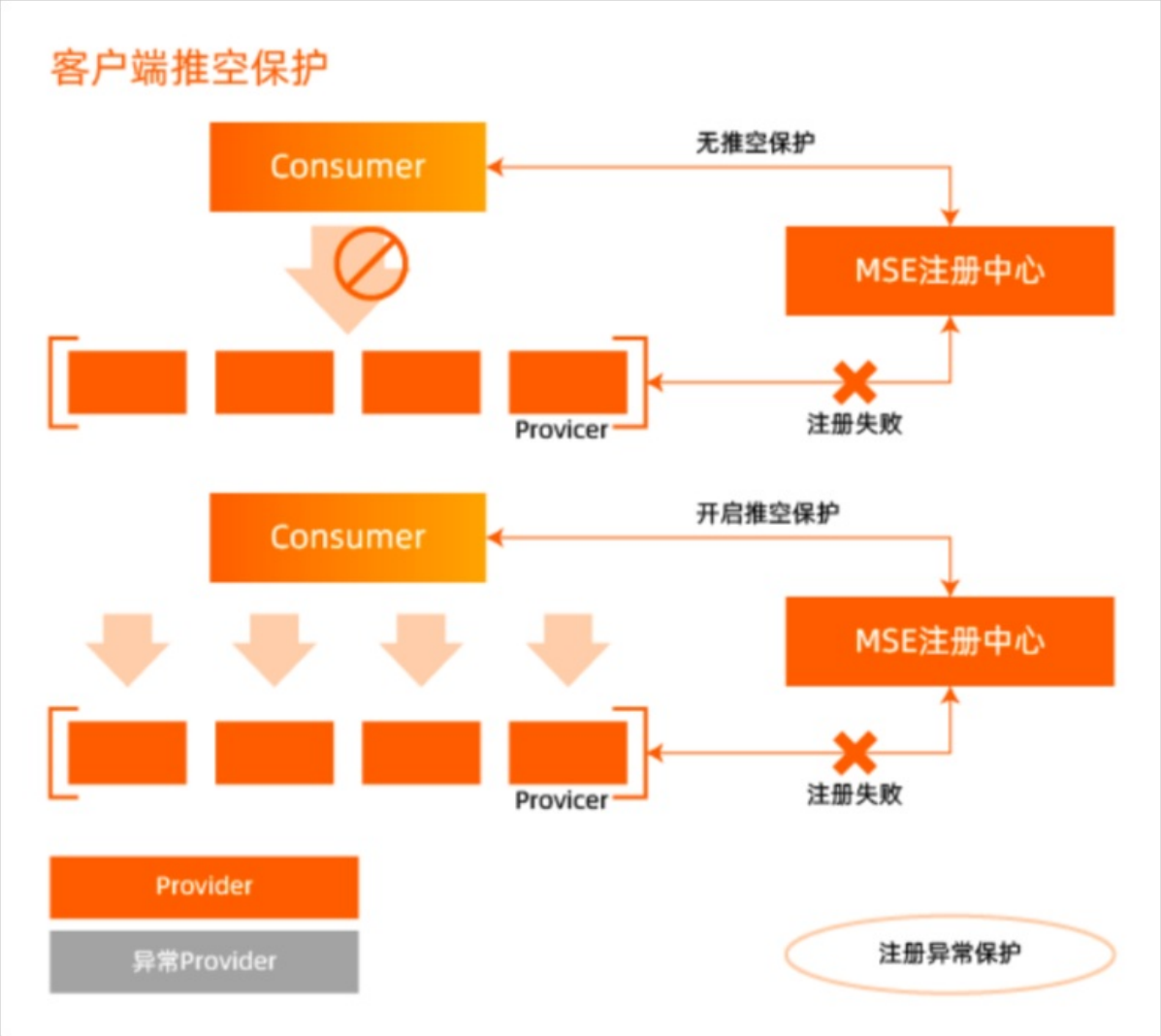
2.1. 配置推空保护

注册中心作为承担服务注册发现的核心组件，是微服务架构中必不可少的一环。本文介绍注册中心中的推空保护。

背景信息

客户端在请求注册中心订阅服务端地址列表时，在服务端注册异常的场景下，注册中心返回了空列表，此时客户端忽略该空返回的变更，从缓存中获取上一次正常的服务端地址进行服务访问。推空保护功能可以在注册中心在进行变更（变配、升降级）或遇到突发情况（例如，可用区断网断电）或其他不可预知情况下的列表订阅异常收到空的地址列表推送时，可以有效保护业务调用，增加业务可靠性。

本文演示的应用架构是由后端的微服务应用实例（Spring Cloud）构成。具体的后端调用链路有Spring Cloud Consumer调用Spring Cloud Provider，这些应用中的服务之间通过Nacos注册中心实现服务注册与发现。



说明 该功能Agent正在灰度中，如果需要使用欢迎提工单或者加入微服务引擎钉钉交流群：34754806，联系技术支持单独升级后再进行试用。

使用限制

限制项	限制值	说明
Spring Cloud版本	Spring Cloud Edgware及以上版本。	-
Dubbo版本	2.5.3 ~ 2.7.8	Dubbo 3.0+版本支持当前处于灰度中，如有场景需求，请提工单。
注册中心类型	• Nacos • Eureka • ZooKeeper	-

准备工作

- 已创建Kubernetes集群，请参见创建Kubernetes托管版集群。
- 已开通MSE微服务治理专业版，请参见开通MSE微服务治理。

1. 安装MSE微服务治理组件
   i. 在容器服务控制台左侧导航栏中，选择**市场 > 应用场景**，在搜索框中输入*ack-mse-pilot*，单击该组件。
   ii. 在详情页面单击**一键部署**，在**创建**页面选择开通该组件的集群，单击**下一步**，然后单击**确定**。
2. 为应用开启微服务治理
   i. 登录MSE治理中心控制台。在左侧导航栏选择**微服务治理中心 > 应用信息 > K8s集群列表**。在搜索框搜索目标集群，然后单击目标集群**操作**列下方的**管理**。
   ii. 在**集群详情**页面单击目标命名空间**操作**列下方的**开启微服务治理**。然后单击**确定**开启微服务治理。

部署Demo应用程序

1. 在容器服务控制台左侧导航栏，单击**集群**。在**集群列表**页面，单击目标集群名称或者目标集群右侧**操作**列下方的**详情**。
2. 在集群管理页左侧导航栏中，选择**工作负载 > 无状态**。
3. 在**无状态**页面的顶部选择**命名空间**，然后单击**使用YAML创建资源**。对模板进行相关配置，完成配置后单击**创建**。本文示例中部署sc-consumer、sc-consumer-empty和sc-provider，使用的是开源的Nacos。

```
# 开启推空保护的 sc-consumer
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sc-consumer
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sc-consumer
  template:
    metadata:
      annotations:
        msePilotCreateAppName: sc-consumer
      labels:
        app: sc-consumer
    spec:
      containers:
      - env:
        - name: JAVA_HOME
```

```
          value: /usr/lib/jvm/java-1.8-openjdk/jre
        - name: spring.cloud.nacos.discovery.server-addr
          value: nacos-server:8848
        image: registry.cn-hangzhou.aliyuncs.com/mse-demo-hz/demo:sc-consumer-0.1
        imagePullPolicy: Always
        name: sc-consumer
        ports:
        - containerPort: 18091
        livenessProbe:
          tcpSocket:
            port: 18091
          initialDelaySeconds: 10
          periodSeconds: 30
---
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-spec: slb.s1.small
    service.beta.kubernetes.io/alicloud-loadbalancer-address-type: internet
  name: sc-consumer-slb
spec:
  ports:
    - port: 80
      protocol: TCP
      targetPort: 18091
  selector:
    app: sc-consumer
  type: LoadBalancer
status:
  loadBalancer: {}
# 无推空保护的sc-consumer-empty
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sc-consumer-empty
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sc-consumer-empty
  template:
    metadata:
      annotations:
        msePilotCreateAppName: sc-consumer-empty
      labels:
        app: sc-consumer-empty
    spec:
      containers:
      - env:
        - name: JAVA_HOME
          value: /usr/lib/jvm/java-1.8-openjdk/jre
        - name: spring.cloud.nacos.discovery.server-addr
          value: nacos-server:8848
        image: registry.cn-hangzhou.aliyuncs.com/mse-demo-hz/demo:sc-consumer-0.1
```

```
        imagePullPolicy: Always
        name: sc-consumer-empty
        ports:
        - containerPort: 18091
        livenessProbe:
          tcpSocket:
            port: 18091
          initialDelaySeconds: 10
          periodSeconds: 30
---
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-spec: slb.s1.small
    service.beta.kubernetes.io/alicloud-loadbalancer-address-type: internet
  name: sc-consumer-empty-slb
spec:
  ports:
    - port: 80
      protocol: TCP
      targetPort: 18091
  selector:
    app: sc-consumer-empty
  type: LoadBalancer
status:
  loadBalancer: {}
# sc-provider
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sc-provider
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sc-provider
  strategy:
  template:
    metadata:
      annotations:
        msePilotCreateAppName: sc-provider
      labels:
        app: sc-provider
    spec:
      containers:
      - env:
        - name: JAVA_HOME
          value: /usr/lib/jvm/java-1.8-openjdk/jre
        - name: spring.cloud.nacos.discovery.server-addr
          value: nacos-server:8848
        image: registry.cn-hangzhou.aliyuncs.com/mse-demo-hz/demo:sc-provider-0.3
        imagePullPolicy: Always
        name: sc-provider
        ports:
```

```
        - containerPort: 18084
        livenessProbe:
          tcpSocket:
            port: 18084
          initialDelaySeconds: 10
          periodSeconds: 30
# Nacos Server
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nacos-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nacos-server
  template:
    metadata:
      labels:
        app: nacos-server
    spec:
      containers:
      - env:
        - name: MODE
          value: standalone
        image: nacos/nacos-server:latest
        imagePullPolicy: Always
        name: nacos-server
      dnsPolicy: ClusterFirst
      restartPolicy: Always
# Nacos Server Service 配置
---
apiVersion: v1
kind: Service
metadata:
  name: nacos-server
spec:
  ports:
  - port: 8848
    protocol: TCP
    targetPort: 8848
  selector:
    app: nacos-server
  type: ClusterIP
```

开启推空保护功能

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
4. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**，单击目标应用名称，在**应用详情**页面单击**推空保护**页签。

5. 打开**推空保护**开关按钮。

功能验证

1. 编写测试脚本 `vi curl.sh` 。

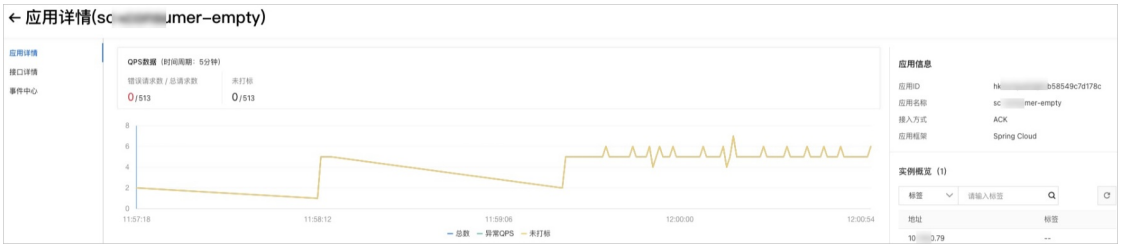
```
while :
do
        result=`curl $1 -s`
        if [[ "$result" == *"500"* ]]; then
                echo `date +%F-%T` $result
        else
                echo `date +%F-%T` $result
        fi
        sleep 0.1
done
```

2. 执行脚本，进行测试。

   i. 执行脚本 `% sh curl.sh {sc-consumer-empty-slb}:18091/user/rest` ，显示如下：

```
2022-01-19-11:58:12 Hello from [18084]10.116.0.142!
2022-01-19-11:58:12 Hello from [18084]10.116.0.142!
2022-01-19-11:58:12 Hello from [18084]10.116.0.142!
2022-01-19-11:58:13 Hello from [18084]10.116.0.142!
2022-01-19-11:58:13 Hello from [18084]10.116.0.142!
2022-01-19-11:58:13 Hello from [18084]10.116.0.142!
```

   ii. 保持脚本一直在调用，观察MSE控制台看到如下情况：

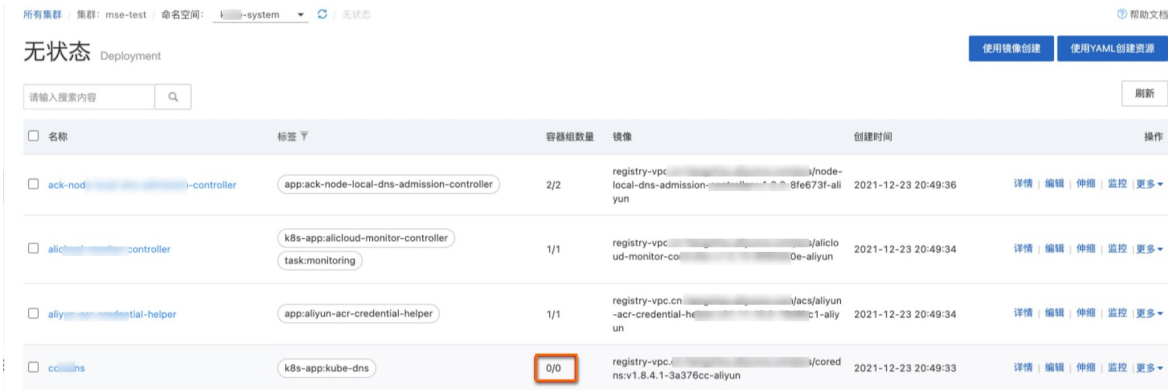


   iii. 执行脚本 `% sh curl.sh {sc-consumer-slb}:18091/user/rest` ，显示如下：

```
2022-01-19-11:58:13 Hello from [18084]10.116.0.142!
2022-01-19-11:58:13 Hello from [18084]10.116.0.142!
2022-01-19-11:58:13 Hello from [18084]10.116.0.142!
2022-01-19-11:58:14 Hello from [18084]10.116.0.142!
2022-01-19-11:58:14 Hello from [18084]10.116.0.142!
2022-01-19-11:58:14 Hello from [18084]10.116.0.142!
```

   iv. 保持脚本一直在调用，观察MSE控制台看到如下情况：

3. 将coredns组件缩容至数量0，模拟DNS网络解析异常场景。



发现实例与Nacos的连接断开且服务列表为空。

4. 模拟DNS服务恢复，将其扩容回数量2。

结果验证

在以上保持持续的业务流量过程中，可以发现sc-consumer-empty服务出现大量且持续的报错。只有重启了Provider，sc-consumer-empty才恢复正常。

```
2022-01-19-12:02:37 {"timestamp":"2022-01-19T04:02:37.597+0000","status":500,"error":"Internal
Server Error","message":"com.netflix.client.ClientException: Load balancer does not have availa
ble server for client: mse-service-provider","path":"/user/feign"}
2022-01-19-12:02:37 {"timestamp":"2022-01-19T04:02:37.799+0000","status":500,"error":"Internal
Server Error","message":"com.netflix.client.ClientException: Load balancer does not have availa
ble server for client: mse-service-provider","path":"/user/feign"}
2022-01-19-12:02:37 {"timestamp":"2022-01-19T04:02:37.993+0000","status":500,"error":"Internal
Server Error","message":"com.netflix.client.ClientException: Load balancer does not have availa
ble server for client: mse-service-provider","path":"/user/feign"}
```

相比sc-consumer-empty，sc-consumer应用全流程没有任何报错。

查看推空保护事件

1. 登录MSE治理中心控制台，在顶部菜单栏选择地域。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**，单击目标应用名称，在**应用详情**页面单击**推空保护**页签，查看**保护事件**。

2.2. 配置基于Java微服务网关的全链路灰度

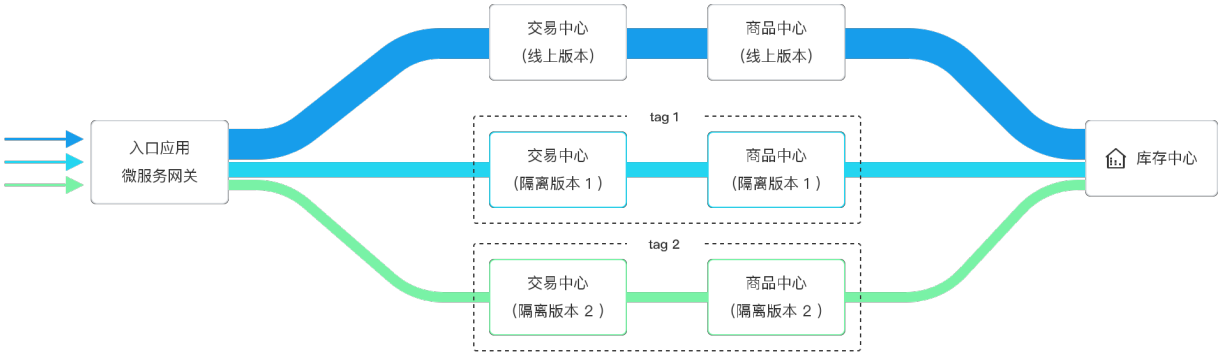
通过MSE提供的全链路灰度能力，您可以无需修改业务代码，实现全链路流量控制。本文介绍如何通过配置MSE全链路灰度功能，为Java微服务网关或者Spring Cloud应用这类入口应用实现全链路灰度。

背景信息

在微服务场景中，当您部署的Spring Cloud应用或Dubbo应用存在升级版本时，由于应用间的调用是随机的，会导致无法将具有一定特征的流量路由到应用的目标版本。全链路流量控制功能将应用的相关版本隔离成一个独立的运行环境（即泳道），通过设置泳道规则，将满足规则的请求流量路由到目标版本应用。

本文以电商架构中的下单场景为例，介绍Java微服务网关到微服务的全链路流控功能。假设应用的架构由Java微服务网关Zuul以及后端的微服务架构（Spring Cloud）组成。后端调用链路有3个：购物车（A），交易中心（B），库存中心（C），可以通过客户端或者是HTML来访问后端服务，这些服务之间通过Nacos注册中心实现服务发现。

用户下单后，流量从Java微服务网关（Spring Cloud Gateway或者Spring Cloud Zuul）进来，调用交易中心，交易中心再调用商品中心，商品中心调用下游的库存中心。交易中心和商品中心各有两个新版本（1和2）在运行，需要对这两个新版本进行灰度验证。此时通过配置微服务网关的全链路灰度规则将满足特定流控规则的请求流量路由到新版本，其余流量全部路由到线上（正式）版本。



术语说明

术语	说明
泳道	相同版本应用定义的一套隔离环境。只有满足了流控路由规则的请求流量才会路由到对应泳道里的打标应用。每条泳道，和一个标签相对应。泳道组里标签相同的多个应用节点，必定属于同一个泳道。一个应用可以属于多个泳道，一个泳道可以包含多个应用，应用和泳道是多对多的关系。
泳道组	泳道的集合。泳道组的作用主要是为了区分不同团队或不同场景。

适用场景

- 入口应用为Java微服务网关（Spring Cloud Gateway/Spring Cloud Zuul）或者Spring Cloud应用。
- 想要根据请求的Header、Cookie和Parameter等特征配置规则来实现全链路灰度。

使用限制

由于全链路灰度功能整合了标签路由功能，因此不推荐已经加入全链路流量控制的应用同时配置金丝雀发布、标签路由规则。

限制项	限制值	备注
Spring Cloud版本	Spring Cloud Edgware及以上版本。	-
Dubbo版本	2.5.3 ~ 2.7.8	Dubbo 3.0+版本支持当前处于灰度中，如有场景需求，请提交工单。
客户端类型	• Resttemplate • Spring Cloud OpenFeign	-
Java应用JDK版本	目前支持JDK 1.6、1.7、1.8版本应用接入	JDK 1.11版本当前处于灰度中，如有场景需求，请提交工单。
负载均衡类型	• Ribbon 2.0.x+ • LoadBalancer 3.0.x+	-
Spring Cloud Gateway版本	Spring Cloud Gateway 2.1.0.RELEASE+	-

限制项	限制值	备注
Spring Cloud Zuul版本	1.3.x	-
注册中心类型	• Nacos • Eureka • ZooKeeper	微服务治理能力无关注册中心，可以是MSE托管注册中心，也可以是自建注册中心。

准备工作

创建Kubernetes集群

具体创建操作，请参见创建Kubernetes托管版集群和创建Kubernetes专有版集群。

开启MSE微服务治理

1. 在MSE微服务治理开通页面，开通微服务治理专业版。关于微服务治理的计费详情，请参见价格说明。
2. 安装MSE微服务治理组件。
   i. 在容器服务ACK控制台左侧导航栏，选择**市场 > 应用市场**。
   ii. 单击**应用目录**页签，然后搜索并单击*ack-onepilot*组件。
   iii. 在**ack-onepilot**页面右上方单击**一键部署**，在**创建**面板中选择集群和命名空间，设置组件发布名称，然后单击**下一步**。

   > **说明** 推荐使用默认的命名空间ack-onepilot。

   iv. 在**参数配置**向导中确认组件参数信息，然后单击**确定**。
   安装完成后，在命名空间ack-onepilot中出现ack-onepilot应用，表示安装成功。
3. 为应用开启微服务治理。
   i. 在MSE控制台左侧导航栏选择**微服务治理中心 > K8s集群列表**。
   ii. 搜索目标集群，然后单击目标集群**操作**列下方的**管理**。
   iii. 在**集群详情**页面的命名空间列表区域，单击目标命名空间**操作**列下方的**开启微服务治理**，然后单击**确认**。

部署Demo应用程序

1. 在容器服务ACK控制台的**集群列表**页面，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。
2. 在集群管理页面左侧导航栏中，选择**工作负载 > 无状态**，然后选择**命名空间**，单击**使用YAML创建资源**。
3. 对模板进行相关配置，完成配置后单击**创建**。

   本文示例中会部署A、B、C三个应用，其中A、B应用分别部署一个基线版本和一个灰度版本；部署一个Java微服务网关Zuul，并部署一个Nacos Server应用用于实现服务发现。部署所使用的YAML文件如下，您也可以在GitHub上获取对应的源代码。

```
# 部署 Nacos Server
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nacos-server
spec:
  selector:
    matchLabels:
      app: nacos-server
```

```
  template:
    metadata:
      annotations:
      labels:
        app: nacos-server
        msePilotAutoEnable: 'off'
    spec:
      containers:
        - env:
            - name: MODE
              value: "standalone"
          image: registry.cn-shanghai.aliyuncs.com/yizhan/nacos-server:latest
          imagePullPolicy: IfNotPresent
          name: nacos-server
          ports:
            - containerPort: 8848
---
apiVersion: v1
kind: Service
metadata:
  name: nacos-server
spec:
  type: ClusterIP
  selector:
    app: nacos-server
  ports:
    - name: http
      port: 8848
      targetPort: 8848
# 部署业务应用
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-zuul
spec:
  selector:
    matchLabels:
      app: spring-cloud-zuul
  template:
    metadata:
      annotations:
        msePilotCreateAppName: spring-cloud-zuul
      labels:
        app: spring-cloud-zuul
    spec:
      containers:
        - env:
            - name: JAVA_HOME
              value: /usr/lib/jvm/java-1.8-openjdk/jre
          image: registry.cn-shanghai.aliyuncs.com/yizhan/spring-cloud-zuul:1.0.0
          imagePullPolicy: Always
          name: spring-cloud-zuul
          ports:
            - containerPort: 20000
---
apiVersion: v1
```

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-spec: slb.s1.small
    service.beta.kubernetes.io/alicloud-loadbalancer-address-type: internet
  name: zuul-slb
spec:
  ports:
    - port: 80
      protocol: TCP
      targetPort: 20000
  selector:
    app: spring-cloud-zuul
  type: LoadBalancer
status:
  loadBalancer: {}
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-a
spec:
  selector:
    matchLabels:
      app: spring-cloud-a
  template:
    metadata:
      annotations:
        msePilotCreateAppName: spring-cloud-a
      labels:
        app: spring-cloud-a
    spec:
      containers:
        - env:
            - name: JAVA_HOME
              value: /usr/lib/jvm/java-1.8-openjdk/jre
          image: registry.cn-shanghai.aliyuncs.com/yizhan/spring-cloud-a:1.0.0
          imagePullPolicy: Always
          name: spring-cloud-a
          ports:
            - containerPort: 20001
          livenessProbe:
            tcpSocket:
              port: 20001
            initialDelaySeconds: 10
            periodSeconds: 30
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-b
spec:
  selector:
    matchLabels:
      app: spring-cloud-b
  template:
```

```
    metadata:
      annotations:
        msePilotCreateAppName: spring-cloud-b
      labels:
        app: spring-cloud-b
    spec:
      containers:
        - env:
            - name: JAVA_HOME
              value: /usr/lib/jvm/java-1.8-openjdk/jre
          image: registry.cn-shanghai.aliyuncs.com/yizhan/spring-cloud-b:1.0.0
          imagePullPolicy: Always
          name: spring-cloud-b
          ports:
            - containerPort: 20002
          livenessProbe:
            tcpSocket:
              port: 20002
            initialDelaySeconds: 10
            periodSeconds: 30
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-c
spec:
  selector:
    matchLabels:
      app: spring-cloud-c
  template:
    metadata:
      annotations:
        msePilotCreateAppName: spring-cloud-c
      labels:
        app: spring-cloud-c
    spec:
      containers:
        - env:
            - name: JAVA_HOME
              value: /usr/lib/jvm/java-1.8-openjdk/jre
          image: registry.cn-shanghai.aliyuncs.com/yizhan/spring-cloud-c:1.0.0
          imagePullPolicy: Always
          name: spring-cloud-c
          ports:
            - containerPort: 20003
          livenessProbe:
            tcpSocket:
              port: 20003
            initialDelaySeconds: 10
            periodSeconds: 30
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-a-gray
spec:
```

```
  selector:
    matchLabels:
      app: spring-cloud-a-gray
  template:
    metadata:
      annotations:
        alicloud.service.tag: gray
        msePilotCreateAppName: spring-cloud-a
      labels:
        app: spring-cloud-a-gray
    spec:
      containers:
        - env:
            - name: JAVA_HOME
              value: /usr/lib/jvm/java-1.8-openjdk/jre
          image: registry.cn-shanghai.aliyuncs.com/yizhan/spring-cloud-a:1.0.0
          imagePullPolicy: Always
          name: spring-cloud-a-gray
          ports:
            - containerPort: 20001
          livenessProbe:
            tcpSocket:
              port: 20001
            initialDelaySeconds: 10
            periodSeconds: 30
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-b-gray
spec:
  selector:
    matchLabels:
      app: spring-cloud-b-gray
  template:
    metadata:
      annotations:
        alicloud.service.tag: gray
        msePilotCreateAppName: spring-cloud-b
      labels:
        app: spring-cloud-b-gray
    spec:
      containers:
        - env:
            - name: JAVA_HOME
              value: /usr/lib/jvm/java-1.8-openjdk/jre
          image: registry.cn-shanghai.aliyuncs.com/yizhan/spring-cloud-b:1.0.0
          imagePullPolicy: Always
          name: spring-cloud-b-gray
          ports:
            - containerPort: 20002
          livenessProbe:
            tcpSocket:
              port: 20002
            initialDelaySeconds: 10
            periodSeconds: 30
```

步骤一：创建泳道组

1. 登录MSE控制台， 在顶部菜单栏选择地域。
2. 在左侧导航栏选择**微服务治理中心 > 流量治理 > 全链路灰度**。
3. 单击**创建泳道组及泳道**。如果您选择的微服务空间内已经创建过泳道组，则单击**+创建泳道组**。
4. 在**创建泳道组**面板中填写**泳道组名称**并选择**Java微服务网关**，设置泳道组相关参数，然后单击**确定**。

参数	说明
泳道组名称	自定义设置泳道组的名称。支持大小写字母、数字、短划线（-）和下划线（_），长度不超过64个字符。
入口类型	目前支持Ingress及自建网关应用，以及Java微服务网关（包含Spring Cloud微服务应用）
泳道组涉及所有应用	单击**+添加灰度链路涉及应用**，选择您的入口应用或入口网关所涉及的所有相关服务。

泳道组创建完成后，在**全链路灰度**页面的**泳道组及涉及的应用**区域会出现您所创建的泳道组。请检查入口应用和所涉及的应用是否正确，如需变更泳道组信息，可在页面自行修改相关信息。

步骤二：创建泳道

1. 在**全链路灰度**页面上方选择和创建泳道组时相同的微服务空间，然后在底部单击**点击创建第一个分流泳道**。如果您选择的微服务空间内已经创建过泳道，则单击**创建泳道**。

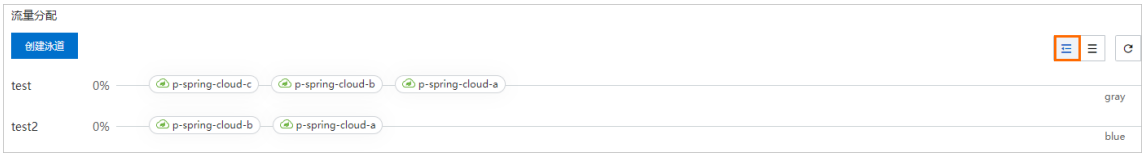
   > **说明** 加入全链路流量控制的应用，将不再支持金丝雀发布、标签路由等功能。

2. 在**创建泳道**面板中设置流控泳道相关参数，然后单击**确定**。

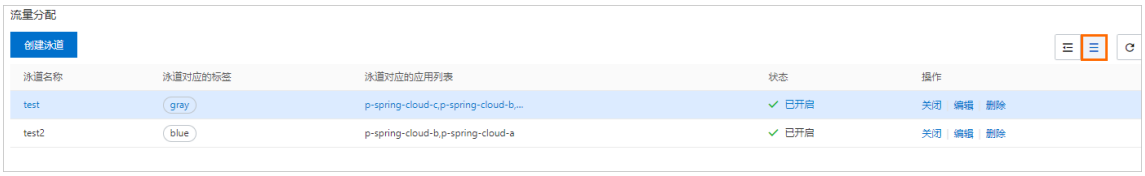
参数	说明
泳道名称	自定义设置流控泳道的名称。支持大小写字母、数字、短划线（-）和下划线（_），长度不超过64个字符。
配置应用标签	○ 配置方式：在容器ACK控制台中，在应用YAML的 `spec.template.metadata.annotations` 下增加 `alicloud.service.tag:{tag}` 。 ○ 设置标签名：如“B-gray”，并为 `spec.template.metadata.annotations` 增加如下两个key-value健值对。 ■ `msePilotCrateAppName:${AppName}` ■ `alicloud.service.tag:gray`
添加应用	当完成**创建泳道**面板中的STEP 2配置后，下拉框中会出现相应的标签列表，选择对应的标签，则会自动添加相应的应用。
路由规则	设置相应的路由规则条件。本文示例中设置的流量规则条件请求的Parmeter 为*name=xiaoming*。

完成泳道创建后，可以查看泳道详情：

- 单击 图标，您可以查看该泳道的流量比例。



- 单击图标，您可以设置该泳道上应用的状态。

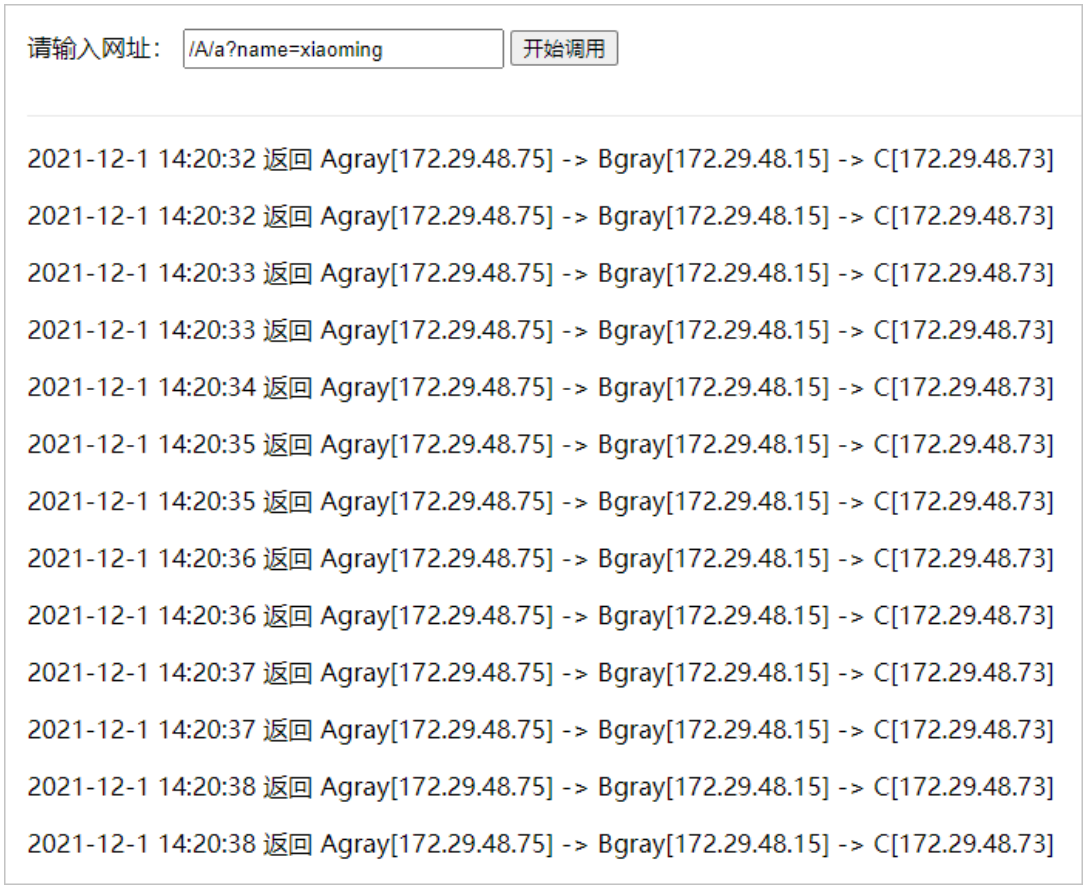


> 说明
>
> - 在**操作**列选择**开启**，表示创建的泳道将会生效，即流量会按照泳道方式进行流转，满足规则的流量会优先流向标记有当前泳道对应标签的应用版本，如果没有对应标签的应用版本则流向未打标的应用版本。
> - 在**操作**列选择**关闭**，表示关闭创建的泳道，即该应用往后的流量会流向未打标的应用版本。

步骤三：验证特征流量路由到目标应用

结果验证

1. 在容器服务ACK控制台选择目标集群进入集群详情页面，然后在左侧导航栏选择**网络 > 服务**。
2. 单击zuul-slb服务所对应的外部端点地址。
3. 在服务调用页面输入*/A/a?name=xiaoming*，然后单击**开始调用**。全链路灰度功能已经生效。



查看打标应用的流量监控图

1. 在**全链路灰度**页面单击目标泳道组页签。
2. 在**泳道组涉及应用**区域单击目标应用名称，即可在右侧出现相应的QPS监控图。

查看所有应用监控图

您除了查看单个应用的监控图外，您还可以查看泳道组内所有应用的监控图。通过对比分析所有应用的监控图，可以得到更多信息。在QPS监控图右侧单击**查看所有应用监控（流量逃逸观测）**，您可以查看该泳道所有应用的流量监控视图。您可以选择查看同一时刻调用的应用概览信息，也可以分析流量逃逸问题，判断逃逸对象。

2.3. 配置消息灰度

如果您在使用金丝雀发布、全链路灰度以及开发环境隔离等场景中需要使用到消息的灰度，那么您需要开启消息灰度的功能。目前，MSE只支持RocketMQ类型的消息灰度。

背景信息

虽然绝大多数业务场景下对于消息的灰度的要求并不像RPC的要求得这么严格，但是在以下两个场景中，还是会对消息的全链路有一定的诉求。

- 当消息的消费逻辑进行了修改时，这时候希望通过小流量的方式来验证新的消息消费逻辑的正确性，这时会对消息的灰度有诉求。
- 在消息消费时，可能会产生新的RPC调用，如果没有在消息这一环去遵循之前设定好的全链路流量控制的规则，会导致通过消息产生的这部分流量“逃逸”，从而导致全链路灰度的规则遭到破坏，导致出现不符合预期的情况。

使用说明

- 使用此功能您无需修改应用的代码和配置。
- 消息类型目前只支持RocketMQ，包含开源版本和阿里云商业版。
  - 如果您使用开源RocketMQ，则RocketMQ Server和RocketMQ Client都需要使用4.5.0及以上版本。
  - 如果您使用阿里云RocketMQ，需要使用铂金版，且Ons Client使用1.8.0.Final及以上版本。
- 消息的生产者和消息的消费者，需要同时开启消息灰度，消息的灰度功能才能生效。
- 开启消息灰度后，MSE会修改消息的Consumer Group。例如原来的Consumer Group为`group1`，环境标签为gray，开启消息灰度后，则`group`会被修改成`group1_gray`，如果您使用的是阿里云RocketMQ，请提前创建好`group`。
- 默认使用SQL92的过滤方式，如果您使用的开源RocketMQ，开源的RocketMQ Server端需要支持SQL92过滤，且在服务端开启此功能（即在*broker.conf*中配置`enablePropertyFilter=true`）。

> **说明** 如果您的应用场景不满足支持SQL92过滤的条件，那么可以使用通过FilterMessageHook在消费者过滤的方式，此方式需要在所有的应用中配置环境变量 `profiler.micro.service.mq.server.gray.enable=false` 。因为消费者过滤的方式会在每个环境都处理全量的消息，对消息的生产者和消费者压力都比较大，不推荐在生产中使用此模式。

开启消息灰度

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
4. 在**应用列表**页面单击目标应用名称。在**应用详情**页面单击**消息灰度**页签。
5. 在**未打标环境忽略的标签**右侧单**编辑**，打开**开启消息灰度**右侧的开关，然后单击**确定**。

   如果您不希望未打标环境消费其他环境生产出来的消息，请在**未打标环境忽略的标签**中选择需要忽略的标签。

> **说明**
> - 应用在开启消息灰度后，需要**重启**才能生效。
> - **未打标环境忽略的标签**支持动态生效，不需要重启应用。
> - 当消息的生产者和消费者都开启消息灰度，并且都重启生效之后。消息消费者的行为如下：
>   - 未打标的环境节点默认会消费所有环境生产出来的消息。
>   - 打标环境节点只消费相同标签环境生产出来的消息。

2.4. 配置基于Ingress网关的全链路灰度

通过Ingress-nginx提供的全链路灰度能力，可以在不需要修改任何您的业务代码的情况下，实现全链路流量控制。本文介绍通过Ingress-nginx实现全链路灰度功能。

前提条件

- 已创建Kubernetes集群，请参见创建Kubernetes托管版集群。
- 已开通MSE微服务治理专业版，请参见开通MSE微服务治理。

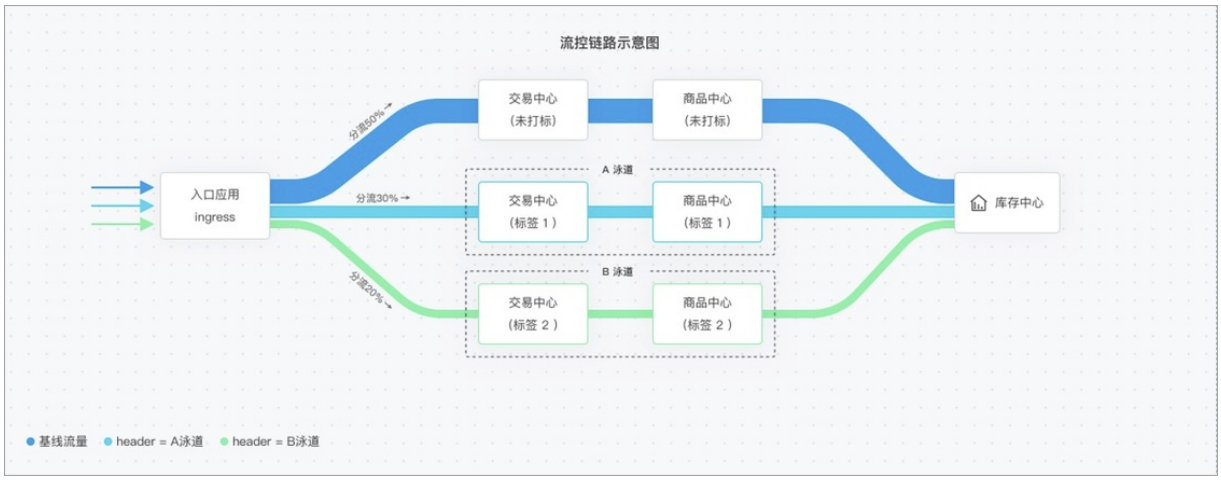
背景信息

在微服务场景中，当您部署的Spring Cloud应用或Dubbo应用存在升级版本时，由于应用间的调用是随机的，会导致无法将具有一定特征的流量路由到应用的目标版本。全链路流量控制功能将应用的相关版本隔离成一个独立的运行环境（即泳道），通过设置Ingress路由规则，将满足规则的请求流量路由到目标版本应用。

本文以电商架构中的下单场景为例介绍从Ingress网关到微服务的全链路流控功能。假设应用的架构由Ingress-niginx网关以及后端的微服务架构（Spring Cloud）组成，后端调用链路有3个：购物车（A），交易中心（B），库存中心（C），可以通过客户端或者是HTML来访问后端服务，这些服务之间通过Nacos注册中心实现服务发现。

客户下单后流量从Ingress网关进来，调用交易中心，交易中心再调用商品中心，商品中心调用下游的库存中心。

交易中心和商品中心各有两个新版本（1和2）在运行，需要对这两个新版本进行灰度验证。此时通过Ingress网关将满足特定流控规则的请求流量路由到新版本，其余流量全部路由到线上（正式）版本。



使用限制

由于全链路灰度功能整合了标签路由功能，因此不推荐已经加入全链路流量控制的应用同时配置金丝雀发布、标签路由规则。

限制项	限制值	备注
Spring Cloud版本	Spring Cloud Edgware及以上版本。	-
Dubbo版本	2.5.3 ~ 2.7.8	Dubbo 3.0+版本支持当前处于灰度中，如有场景需求，请提交工单。
客户端类型	• Resttemplate • Spring Cloud OpenFeign	-
Java应用JDK版本	目前支持JDK 1.6、1.7、1.8版本应用接入	JDK 1.11版本当前处于灰度中，如有场景需求，请提交工单。
负载均衡类型	• Ribbon 2.0.x+ • LoadBalancer 3.0.x+	-
Spring Cloud Gateway版本	Spring Cloud Gateway 2.1.0.RELEASE+	-
Spring Cloud Zuul版本	1.3.x	-

限制项	限制值	备注
注册中心类型	• Nacos • Eureka • ZooKeeper	微服务治理能力无关注册中心，可以是MSE托管注册中心，也可以是自建注册中心。

名词解释

- **泳道**
  为相同版本应用定义的一套隔离环境。只有满足了流控路由规则的请求流量才会路由到对应泳道里的打标应用。一个应用可以属于多个泳道，一个泳道可以包含多个应用，应用和泳道是多对多的关系。
- **泳道组**
  泳道的集合。泳道组的作用主要是为了区分不同团队或不同场景。

准备工作

1. 登录ack-ingress-nginx应用市场，在页面右上方单击**一键部署**，在**创建**面板中选择**集群**和**命名空间**，设置组件**发布名称**，然后单击**下一步**。

   > **说明** 推荐使用默认命名空间kube-system。

2. 在**参数配置**向导中确认组件参数信息，然后单击**确定**。
   安装完成后，在命名空间kube-system中出现ack-ingress-nginx-default应用，表示安装成功。

1. 开通微服务治理专业版：
   i. 单击开通MSE微服务治理。
   ii. 微服务治理**版本**选择**专业版**，选中**服务协议**，然后单击**立即开通**。
   关于微服务治理的计费详情，请参见价格说明。
2. 安装MSE微服务治理组件：
   i. 在容器服务控制台左侧导航栏中，选择**市场 > 应用市场**。
   ii. 在**应用市场**页面单击**应用目录**页签，然后搜索并单击*ack-onepilot*组件。
   iii. 在ack-onepilot页面右上方单击**一键部署**，在**创建**面板中选择**集群**和**命名空间**，设置组件**发布名称**，然后单击**下一步**。

   > **说明** 推荐使用默认的命名空间ack-onepilot。

   iv. 在**参数配置**向导中确认组件参数信息，然后单击**确定**。
   安装完成后，在命名空间ack-onepilot中出现ack-onepilot应用，表示安装成功。
3. 为应用开启微服务治理：
   i. 登录MSE治理中心控制台。
   ii. 在左侧导航栏选择**微服务治理中心 > 应用信息 > K8s集群列表**。
   iii. 在**K8s集群列表**页面搜索目标集群，单击图标，然后单击目标集群**操作**列下方的**管理**。
   iv. 在**集群详情**页面命名空间列表区域，单击目标命名空间**操作**列下方的**开启微服务治理**。
   v. 在**开启微服务治理**对话框中单击**确认**。

1. 在容器服务控制台左侧导航栏中，单击**集群**。
2. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。

3. 在集群管理页左侧导航栏中，选择**工作负载 > 无状态**。
4. 在**无状态**页面选择**命名空间**，然后单击**使用YAML创建资源**。
5. 对模板进行相关配置，完成配置后单击**创建**。

   本文示例中部署A、B、C三个应用，每个应用分别部署一个基线版本和一个灰度版本；并部署一个Nacos server应用，用于实现服务发现。

   - A应用
     - 基线（base）版本YAML:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-a
spec:
  replicas: 2
  selector:
    matchLabels:
      app: spring-cloud-a
  template:
    metadata:
      annotations:
        msePilotCreateAppName: spring-cloud-a
      labels:
        app: spring-cloud-a
    spec:
      containers:
      - env:
        - name: JAVA_HOME
          value: /usr/lib/jvm/java-1.8-openjdk/jre
        image: registry.cn-shanghai.aliyuncs.com/yizhan/spring-cloud-a:0.1-SNAPSHOT
        imagePullPolicy: Always
        name: spring-cloud-a
        ports:
        - containerPort: 20001
        livenessProbe:
          tcpSocket:
            port: 20001
          initialDelaySeconds: 10
          periodSeconds: 30
```

- 灰度（gray）版本YAML:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-a-new
spec:
  replicas: 2
  selector:
    matchLabels:
      app: spring-cloud-a-new
  strategy:
  template:
    metadata:
      annotations:
        alicloud.service.tag: gray
        msePilotCreateAppName: spring-cloud-a
      labels:
        app: spring-cloud-a-new
    spec:
      containers:
      - env:
        - name: JAVA_HOME
          value: /usr/lib/jvm/java-1.8-openjdk/jre
        image: registry.cn-shanghai.aliyuncs.com/yizhan/spring-cloud-a:0.1-SNAPSHOT
        imagePullPolicy: Always
        name: spring-cloud-a-new
        ports:
        - containerPort: 20001
        livenessProbe:
          tcpSocket:
            port: 20001
          initialDelaySeconds: 10
          periodSeconds: 30
```

- B应用

- 基线（base）版本YAML:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-b
spec:
  replicas: 2
  selector:
    matchLabels:
      app: spring-cloud-b
  strategy:
  template:
    metadata:
      annotations:
        msePilotCreateAppName: spring-cloud-b
      labels:
        app: spring-cloud-b
    spec:
      containers:
      - env:
        - name: JAVA_HOME
          value: /usr/lib/jvm/java-1.8-openjdk/jre
        image: registry.cn-shanghai.aliyuncs.com/yizhan/spring-cloud-b:0.1-SNAPSHOT
        imagePullPolicy: Always
        name: spring-cloud-b
        ports:
        - containerPort: 8080
        livenessProbe:
          tcpSocket:
            port: 20002
          initialDelaySeconds: 10
          periodSeconds: 30
```

- 灰度（gray）版本YAML:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-b-new
spec:
  replicas: 2
  selector:
    matchLabels:
      app: spring-cloud-b-new
  template:
    metadata:
      annotations:
        alicloud.service.tag: gray
        msePilotCreateAppName: spring-cloud-b
      labels:
        app: spring-cloud-b-new
    spec:
      containers:
      - env:
        - name: JAVA_HOME
          value: /usr/lib/jvm/java-1.8-openjdk/jre
        image: registry.cn-shanghai.aliyuncs.com/yizhan/spring-cloud-b:0.1-SNAPSHOT
        imagePullPolicy: Always
        name: spring-cloud-b-new
        ports:
        - containerPort: 8080
        livenessProbe:
          tcpSocket:
            port: 20002
          initialDelaySeconds: 10
          periodSeconds: 30
```

- C应用

- 基线（base）版本YAML:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-c
spec:
  replicas: 2
  selector:
    matchLabels:
      app: spring-cloud-c
  template:
    metadata:
      annotations:
        msePilotCreateAppName: spring-cloud-c
      labels:
        app: spring-cloud-c
    spec:
      containers:
      - env:
        - name: JAVA_HOME
          value: /usr/lib/jvm/java-1.8-openjdk/jre
        image: registry.cn-shanghai.aliyuncs.com/yizhan/spring-cloud-c:0.1-SNAPSHOT
        imagePullPolicy: Always
        name: spring-cloud-c
        ports:
        - containerPort: 8080
        livenessProbe:
          tcpSocket:
            port: 20003
          initialDelaySeconds: 10
          periodSeconds: 30
```

- 灰度（gray）版本YAML：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-cloud-c-new
spec:
  replicas: 2
  selector:
    matchLabels:
      app: spring-cloud-c-new
  template:
    metadata:
      annotations:
        alicloud.service.tag: gray
        msePilotCreateAppName: spring-cloud-c
      labels:
        app: spring-cloud-c-new
    spec:
      containers:
      - env:
        - name: JAVA_HOME
          value: /usr/lib/jvm/java-1.8-openjdk/jre
        image: registry.cn-shanghai.aliyuncs.com/yizhan/spring-cloud-c:0.1-SNAPSHOT
        imagePullPolicy: IfNotPresent
        name: spring-cloud-c-new
        ports:
        - containerPort: 8080
        livenessProbe:
          tcpSocket:
            port: 20003
          initialDelaySeconds: 10
          periodSeconds: 30
```

  - Nacos Server应用YAML：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nacos-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nacos-server
  template:
    metadata:
      labels:
        app: nacos-server
    spec:
      containers:
      - env:
        - name: MODE
          value: standalone
        image: nacos/nacos-server:latest
        imagePullPolicy: Always
        name: nacos-server
      dnsPolicy: ClusterFirst
      restartPolicy: Always
# Nacos Server Service 配置
---
apiVersion: v1
kind: Service
metadata:
  name: nacos-server
spec:
  ports:
  - port: 8848
    protocol: TCP
    targetPort: 8848
  selector:
    app: nacos-server
  type: ClusterIP
```

6. 针对入口应用A，配置两个K8s Service。
   i. 在集群详情页面左侧导航栏选择**网络 > 服务**。
   ii. 在**服务**页面选择**命名空间**，然后单击**使用YAML创建资源**。

iii. 对模板进行相关配置，完成配置后单击**创建**。

- spring-cloud-a-base对应A的base版本：

```
apiVersion: v1
kind: Service
metadata:
  name: spring-cloud-a-base
spec:
  ports:
    - name: http
      port: 20001
      protocol: TCP
      targetPort: 20001
  selector:
    app: spring-cloud-a
```

- spring-cloud-a-gray对应A的gray版本：

```
apiVersion: v1
kind: Service
metadata:
  name: spring-cloud-a-gray
spec:
  ports:
    - name: http
      port: 20001
      protocol: TCP
      targetPort: 20001
  selector:
    app: spring-cloud-a-new
```

安装Ingress-nginx组件

开启MSE微服务治理

部署Demo应用程序

步骤一：创建泳道组

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量治理 > 全链路灰度**。
4. 在**全链路灰度**页面，单击**创建泳道组及泳道**。如果您选择的微服务空间内已经创建过泳道组，则单击**+创建泳道组**。
5. 在**创建泳道组**面板中设置泳道组相关参数，然后单击**确定**。

   泳道组配置参数说明

参数	描述
泳道组名称	自定义设置泳道组的名称。支持大小写字母、数字、短划线（-）和下划线（_），长度不超过64个字符。
入口类型	选择**ingress/自建网关**。

参数	描述
泳道组涉及所有应用	单击**+添加灰度链路涉及应用**，选择您的入口应用或入口网关所涉及的所有相关服务。

泳道组创建完成后，在**全链路灰度**页面的**泳道组涉及应用**区域出现您所创建的泳道组。请检查入口应用和所涉及的应用是否正确，如需变更泳道组信息，请单击右侧的图标并修改相关信息。

步骤二：创建泳道

1. 在**全链路灰度**页面上方选择创建和泳道组时相同的微服务空间，然后底部单击**点击创建第一个分流泳道**。如果您选择的微服务空间内已经创建过泳道，则单击**创建泳道**。

   > **注意** 加入全链路流量控制的应用，将不再支持金丝雀发布、标签路由等功能。

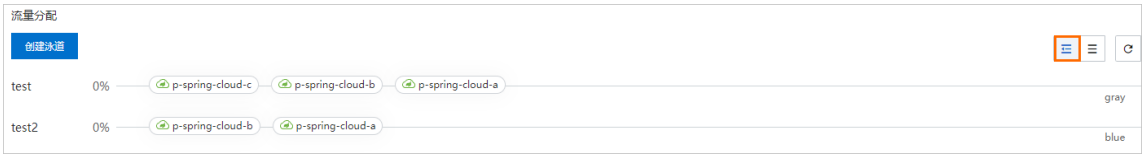
2. 在**创建泳道**面板中设置流控泳道相关参数，然后单击**确定**。

   > **注意** 如果您的网关应用是Ingress网关，需要去容器控制台配置Ingress路由规则。

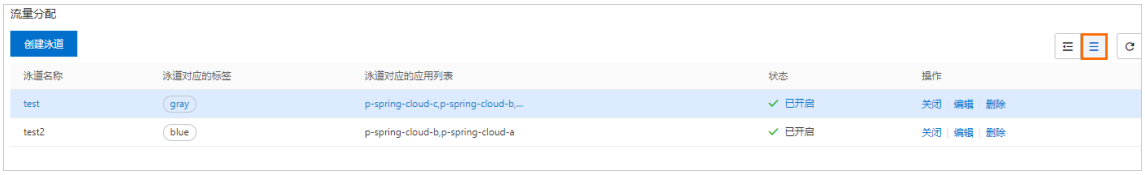
   泳道配置参数说明

参数	描述
泳道名称	自定义设置流控泳道的名称。支持大小写字母、数字、短划线（-）和下划线（_），长度不超过64个字符。
配置应用标签	配置方式：在容器ACK控制台中去应用YAML的`spec.template.metadata.annotations`下增加`alicloud.service.tag:{tag}`。
添加应用	当STPE 2配置好后，单击刷新按钮，下拉框中会出现相应的标签列表，选择对应的标签，就会自动添加相应的应用。

3. 在**全链路灰度**页面的**流量分配**面板查看涌道，有以下两种展现形式：
   - 单击图标，您可以查看该泳道的流量比例。



   - 单击图标，您可以设置该泳道上应用的状态。



操作列的**开启**或**关闭**有如下含义：

- 开启：创建的泳道将会生效，即流量会按照泳道方式进行流转，会优先流向标记有当前泳道对应标签的应用版本，如果没有对应标签的应用版本则流向未打标的应用版本。
- 关闭：关闭创建的泳道，即该应用往后的流量会流向未打标的应用版本。

4. 配置流量入口的Ingress规则，访问 `www.base.com` 路由到A应用的base版本，访问 `www.gray.com` 路由到A应用的gray版本。

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: spring-cloud-a-base
spec:
  rules:
  - host: www.base.com
    http:
      paths:
      - backend:
          serviceName: spring-cloud-a-base
          servicePort: 20001
        path: /
---
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: spring-cloud-a-gray
spec:
  rules:
  - host: www.gray.com
    http:
      paths:
      - backend:
          serviceName: spring-cloud-a-gray
          servicePort: 20001
        path: /
```

验证特征流量路由到目标应用

- **结果验证**
  - 访问 `www.base.com` 路由到基线环境
    - Curl命令：

      ```
      curl -H"Host:www.base.com" http://106.14.XX.XX/a
      ```

    - 返回结果：

      ```
      A[172.18.XX.XX] -> B[172.18.XX.XX] -> C[172.18.XX.XX]%
      ```

  - 访问 `www.gray.com` 路由到灰度环境
    - Curl命令：

      ```
      curl -H"Host:www.gray.com" http://106.14.XX.XX/a
      ```

    - 返回结果：

      ```
      Agray[172.18.XX.XX] -> Bgray[172.18.XX.XX] -> Cgray[172.18.XX.XX]%
      ```

- **查看打标应用的流量监控图**
  i. 在**全链路灰度**页面单击目标泳道组页签。
  ii. 在**泳道组涉及应用**区域单击目标应用名称，即可在右侧出现相应的**QPS监控图**。
- **查看所有应用监控图**
  您除了查看单个应用的监控图外，您还可以查看泳道组内所有应用的监控图。通过比对分析所有应用的监控图，可以分析出更多有用信息。
  在**QPS监控图**右侧单击**查看所有应用监控（流量逃逸观测）**，您可以查看该泳道所有应用的流量监控视图。
  - 您可以查看同一时刻，调用的应用概览信息。
  - 您可以分析流量逃逸问题，判断逃逸对象。

2.5. 配置金丝雀发布

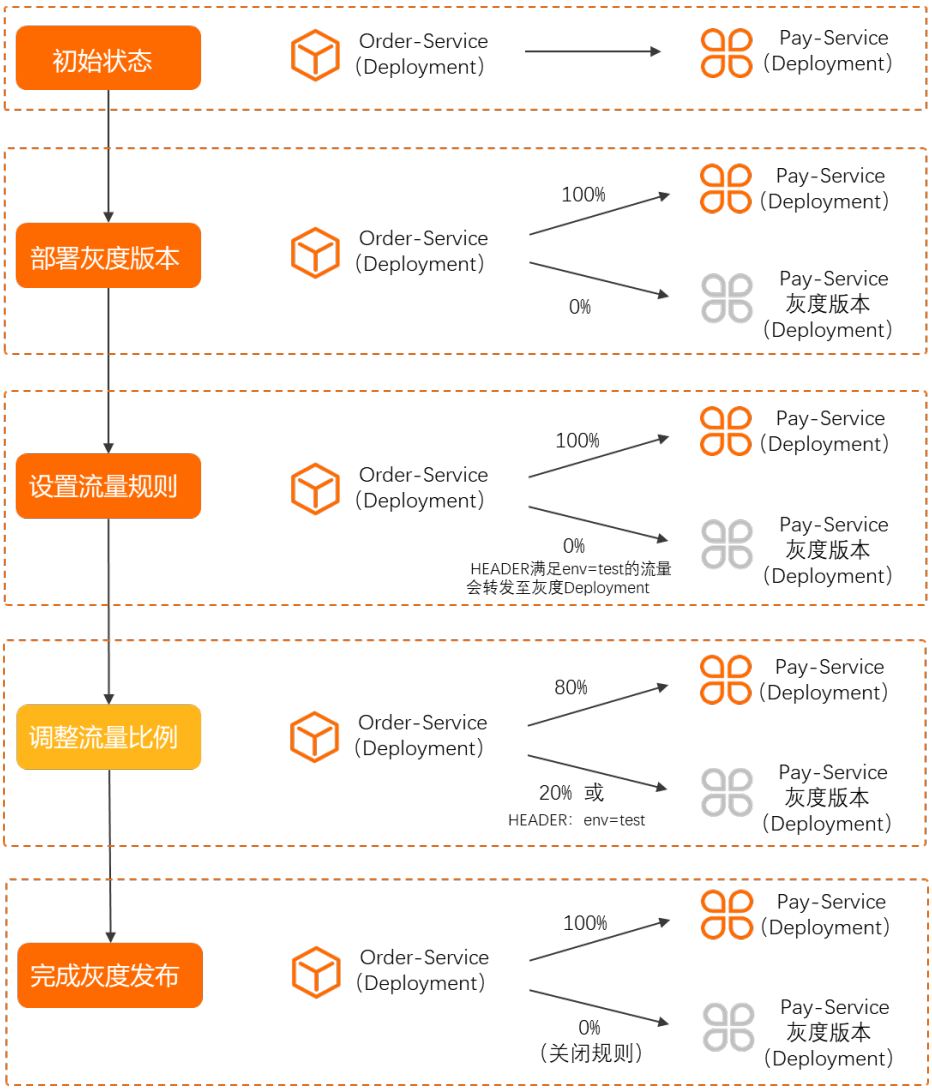
通过ECS方式接入的应用，以及部署在阿里云容器服务ACK集群中的Spring Cloud或Dubbo微服务应用，为了确保其升级的安全性，可以使用金丝雀发布（即灰度发布）进行小规模验证，验证通过后再全量升级。

前提条件

已在ACK集群中安装MSE治理中心组件，并为ACK授予MSE治理中心的访问权限。具体操作，请参见ACK微服务应用接入MSE治理中心。

背景信息

金丝雀发布的过程如下：



1. **初始状态**：假设有2个服务Order-Service和Pay-Service，Order-Service作为Consumer服务会去调用Pay-Service提供的服务。在您没有接入MSE治理中心之前，Order-Service和Pay-Service都对应一个Deployment应用，并且没有设置任何标签。
2. **部署灰度版本**：为Pay-Service部署灰度版本，此时Pay-Service新建一个Deployment应用表示灰度版本，并设置标签。具体操作，请参见在ACK中为应用接入MSE微服务治理并设置标签。

   > **说明** 默认情况下，MSE治理中心会让100%的流量转发至没有打标签的Deployment应用。

3. **设置流量规则**：为Pay-Service设置流量规则：HTTP HEADER中env=test。此时满足该规则的流量会转发至灰度Deployment应用，其他流量转发至正常Deployment应用。
4. **调整流量比例**：为灰度版本调整流量比例，让更多的流量转发至灰度Deployment应用。

   > **注意** 满足流量规则的流量还是会转发至灰度Deployment应用，不满足流量规则的流量有20%的概率转发至灰度Deployment应用。

5. **完成灰度发布**：灰度验证完毕之后，将稳定版本的Deployment应用更新为最新的镜像，单击**发布完成**，此时100%的流量转发至稳定版本Deployment应用。

   > **说明** 建议您将灰度的Deployment应用副本数改为0，无需重复创建Deployment应用。

使用限制

金丝雀发布功能只适合在单个应用的金丝雀发布场景下使用，若有更复杂的场景，请使用标签路由。具体操作，请参见配置标签路由。

操作步骤

本文主要介绍的是配置流量规则的步骤，其他步骤请根据金丝雀发布页面的提示完成。

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
4. 选择目标应用，单击目标应用名称或者**操作**列的**金丝雀**。在左侧导航栏单击**应用详情**，然后单击**金丝雀**页签。
5. 在**应用详情**页面下方单击**使用说明**，根据提示的步骤部署新版本的应用，确保校验都检查通过。

   您可以直接在**使用说明**的步骤3中单击**引入流量**，配置流量规则。

6. 单击**金丝雀**页签下方的**引入流量**，在**引入流量**面板中根据实际情况选择**按比例灰度**或**按内容灰度**，配置流量规则参数，然后单击**确定**。
   - 按比例灰度参数说明

参数	描述
标签	显示应用的标签。
是否链路传递	显示当前实例是否支持链路传递。
实例数量/实际比例	显示应用的实例数量和实例当前所占的流量比例。
流量比例	设置流量百分比。
最后操作时间	显示当前应用最新的操作时间。

> **注意** 流量规则验证成功后，再调大灰度版本流量比例，建议逐渐调大灰度版本的流量比例。

   - 按内容灰度参数说明

参数	描述
框架类型	根据实际应用自动生成相应的框架类型。

参数	描述
Path	选择服务路径，也可单击右侧的**切换为自定义输入**手动输入路径。
条件模式	包含**同时满足下列条件**或**满足下列任一条件**，根据实际需求选择。
条件列表	设置条件参数，当有多个条件规则时，可通过单击**添加新的规则条件**添加。 可以分别设置Cookie、Header、Parameter和Body Content四种类型的参数。例如： ■ Cookie：`hello = "world" 或 "world2"` ■ Parameter：`name=newversion`
是否链路传递	如果需要使用全链路流控，请打开**是否链路传递**开关。如需使用，请参见配置标签路由。

路由规则设置完成后，访问的请求带上规则里的参数，这时流量会去访问灰度Deployment应用。

7. （可选）如果在灰度的过程中，需要更新金丝雀灰度的规则，单击**引入流量**，更新流量规则，然后单击**确定**。

> **注意**
> - 如果您只需要通过流量百分比进行验证，可以通过**按比例灰度**配置应用实例的流量比例。
> - 如果您需要进行精细化的验证，可以同时配置**按比例灰度**和**按内容灰度**，将流量比例规则和内容规则进行结合。

执行结果

金丝雀验证成功：单击**发布完成**按钮，未打标版本的流量比例会被调整为100%，配置的流量规则会被暂时关闭。此时所有的流量都会被转发到未打标的Deployment应用。

> **说明** 发布完成后，建议您将灰度的Deployment应用副本数设置为0，无需重复创建Deployment应用。

金丝雀验证失败：单击**回滚**按钮，灰度版本的流量比例会被调整为0%，配置的流量规则会被清除。此时所有的流量都会被转发到正常Deployment应用，然后可以删除灰度版本的应用。

2.6. 配置标签路由

标签路由通过标签将一个或多个服务的提供者划分到同一个分组，从而约束流量只在指定分组中流转，实现流量隔离的目的。标签路由可以作为蓝绿发布、灰度发布等场景的能力基础。

前提条件

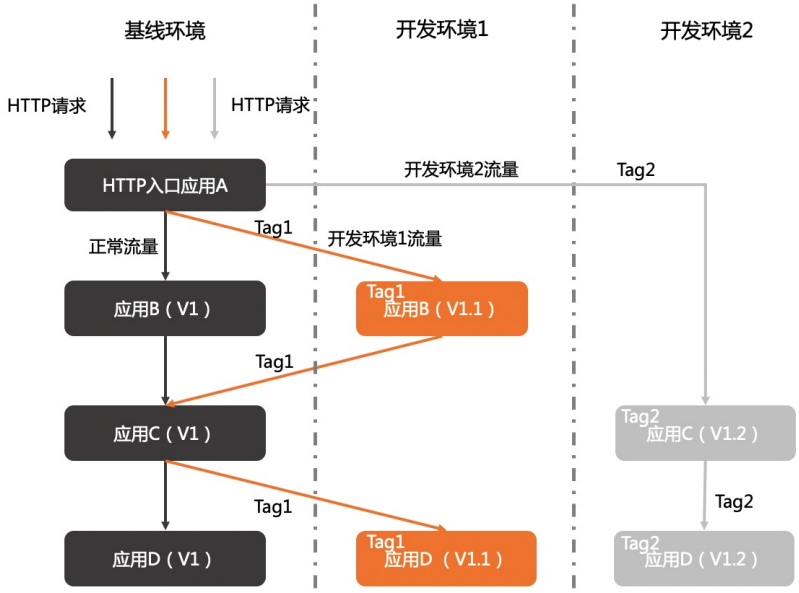
应用的ACK集群中已经安装MSE微服务治理组件，且已经为ACK授予MSE微服务治理的访问权限。具体操作，请参见ACK微服务应用接入MSE治理中心微服务治理。

使用限制

限制项	限制值	说明
Spring Cloud版本	Spring Cloud Edgware及以上版本。	该内容主要针对微服务治理中心，您的应用需要接入MSE服务治理中心。
Dubbo版本	2.5.3 ~ 2.7.8	该内容主要针对微服务治理中心，您的应用需要接入MSE服务治理中心。
客户端类型	• RestTemplate • FeignClient	该内容主要针对微服务治理中心，您的应用需要接入MSE服务治理中心。
Java应用JDK版本	目前支持JDK 1.6、1.7和1.8版本应用接入。	该内容主要针对微服务治理中心，您的应用需要接入MSE服务治理中心。
负载均衡类型	• Ribbon • LoadBalancer	该内容主要针对微服务治理中心，您的应用需要接入MSE服务治理中心。
Spring Cloud Gateway版本	Spring Cloud Gateway 2.1.0.RELEASE+	该内容主要针对微服务治理中心，您的应用需要接入MSE服务治理中心。
注册中心类型	• Nacos • Eureka • ZooKeeper	该内容主要针对微服务治理中心，您的应用需要接入MSE服务治理中心。

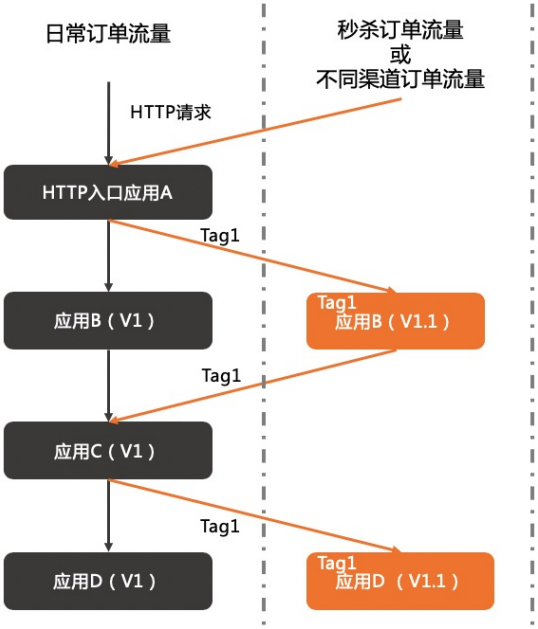
应用场景

- 多版本开发测试
  多个版本并行开发时，需要为每个版本准备一套开发环境。如果版本较多，开发环境成本会非常大。流量隔离方案可以在多版本开发测试时大幅度降低资源成本。
  使用基于标签路由的全链路流量隔离机制，可以将特定的流量路由到指定的开发环境。例如在开发环境1中只修改应用B和应用D，则为这两个应用在开发环境1中的版本创建Tag1标签，并配置对应的路由规则。入口应用A调用B时，会判断流量是否满足路由规则。如果满足，路由到开发环境1中应用B的V1.1版本；如果不满足，路由到基线环境中的应用B的V1版本。应用C调用D的时候同样根据流量决定路由到D的V1版本或V1.1版本。

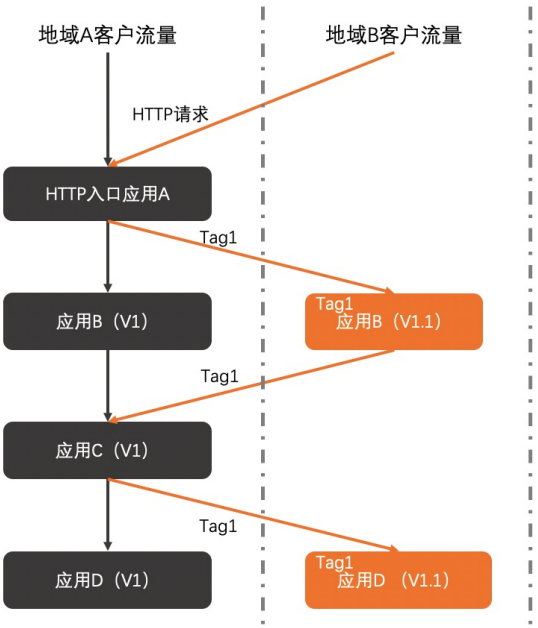


- 相同应用的多版本间流量隔离

如果一个应用有多个版本在线上同时运行，部署在不同环境中，如日常环境和特殊环境，则可以使用标签路由对不同环境中的不同版本进行流量隔离，将秒杀订单流量或不同渠道订单流量路由到特殊环境，将正常的流量路由到日常环境。即使特殊环境异常，本应进入特殊环境的流量也不会进入日常环境，不影响日常环境的使用。

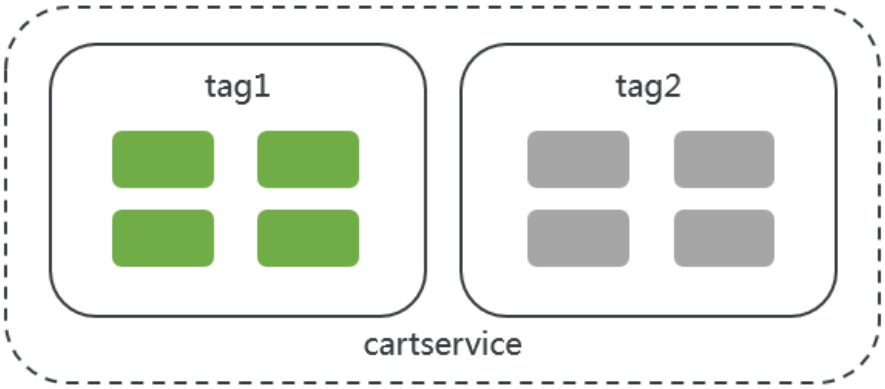


- A/B Testing
  线上有多个应用版本同时运行，期望对不同版本的应用进行A/B Testing，则可以使用标签路由的全链路流量控制将地域A（如杭州）的客户流量路由到V1版本，地域B（如上海）的客户流量路由到V1.1版本，对不同版本进行验证，从而降低新产品或新特性的发布风险，为产品创新提供保障。



操作场景

本文以cartservice为例，分别给应用打上tag1和tag2两个标签，将应用划分为2个分组，每个分组各包含四个节点。



本文介绍如何为cartservice创建标签路由，主要包含以下两步：

1. 在ACK中为应用接入MSE微服务治理并设置标签。
2. 在MSE微服务治理中为应用创建标签路由。

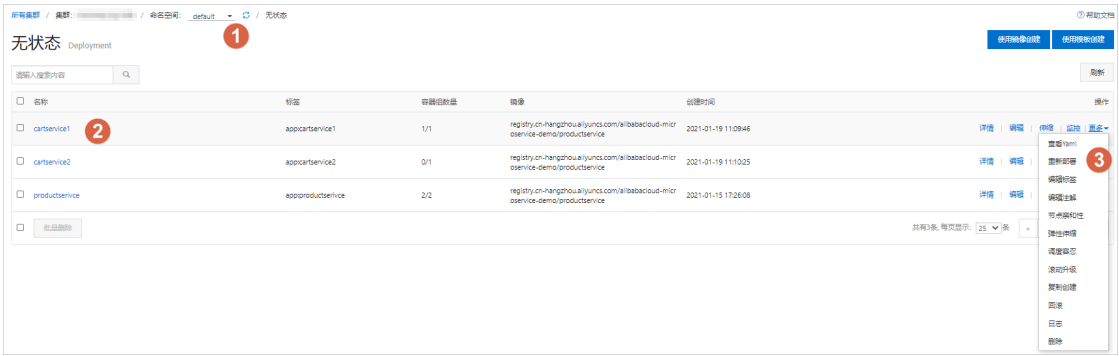
在ACK中为应用接入MSE微服务治理并设置标签

1. 登录容器服务控制台。
2. 在左侧导航栏单击**集群**，在**集群列表**中单击目标集群名称或**详情**。
3. 在**集群信息**页面左侧导航栏选择**工作负载 > 无状态**。
4. 在**无状态（Deployment）**页面右上角单击**使用镜像创建**。
5. 在**创建应用**页面分别创建两个无状态应用cartservice1和cartservice2。

   创建无状态应用的详细步骤，请参见创建无状态工作负载Deployment。
   本文仅介绍为cartservice1和cartservice2创建标签路由相关的参数。

参数	说明
集群	选择已经安装了MSE微服务治理的集群。
副本数量	按示例，设置为4。
镜像名称	在选择镜像时，单击**搜索**，选择**华东1（杭州）**地域，搜索并选择alibabacloud-microservice-demo/cartservice镜像。

6. 为应用cartservice1和cartservice2接入MSE微服务治理，并设置标签。
   i. 返回**无状态（Deployment）**页面，选择**命名空间**，找到创建的应用（cartservice1和cartservice2），在**操作**列单击**更多**，在列表中单击**查看Yaml**。



ii. 在**编辑Yaml**对话框分别为cartservice1和cartservice2在**spec > template > metadata**下添加以下`annotations`，接入MSE微服务治理。

```
annotations:
  msePilotAutoEnable: "on"
  msePilotCreateAppName: "<your-deployment-name>"
```

> **说明** 需要将`<your-deployment-name>`替换为您在MSE微服务治理实际使用的应用名称，即cartservice。

iii. 在**编辑YAML**对话框为cartservice1和cartservice2分别添加标签`alicloud.service.tag: tag1`和`alicloud.service.tag: tag2`。

```
15 spec:
16   progressDeadlineSeconds: 600
17   replicas: 4
18   revisionHistoryLimit: 10
19   selector:
20     matchLabels:
21       app: cartservice2
22   strategy:
23     rollingUpdate:
24       maxSurge: 25%
25       maxUnavailable: 25%
26     type: RollingUpdate
27   template:
28     metadata:
29       annotations:
30         alicloud.service.tag: tag1
31         msePilotAutoEnable: 'on'
32         msePilotCreateAppName: cartservice
33       labels:
34         app: cartservice1
```

```
15 spec:
16   progressDeadlineSeconds: 600
17   replicas: 4
18   revisionHistoryLimit: 10
19   selector:
20     matchLabels:
21       app: cartservice2
22   strategy:
23     rollingUpdate:
24       maxSurge: 25%
25       maxUnavailable: 25%
26     type: RollingUpdate
27   template:
28     metadata:
29       annotations:
30         alicloud.service.tag: tag2
31         msePilotAutoEnable: 'on'
32         msePilotCreateAppName: cartservice
33       labels:
34         app: cartservice2
```

在MSE微服务治理中为应用创建标签路由

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
3. 在**应用列表**页面单击在ACK创建的应用名称**cartservice**。
4. 在cartservice的**应用详情**页面单击**标签路由**页签查看实例标签。
5. 在**标签路由**列表右侧单击**流量分配**，设置各标签的流量比例，然后单击**保存**。
6. 在**标签路由**列表选择标签，然后单击**流量规则**下方的**添加**在**创建标签路由**配置参数，最后单击**确定**即可完成标签路由的创建。

> **说明** 标签路由优先采用流量规则里的路由条件，如果满足该流量规则，则会去该规则对应标签的Pod，否则按照流量比例去对应标签的Pod。

标签路由参数说明：

参数	说明
路由名称	标签路由规则名称，例如`test-springcloud`。
应用	显示应用名称。
标签	显示在ACK中为应用设置的标签。

参数	说明
应用实例	显示cartservice应用中设置了该标签的应用实例的IP及端口。
是否链路传递	如果需要使用全链路流控，请打开**是否链路传递**开关。
流量规则	
框架类型	包含**Spring Cloud**和**Dubbo**，根据应用实际框架选择。 ○ Spring Cloud：仅支持设置URL的Path，例如 `/getIp` 。 ○ Dubbo：支持选择服务和接口。
条件模式	包含**同时满足下列条件**和**满足下列任一条件**，根据实际需求选择。
条件列表	可以分别设置Parameter、Cookie和Header三种类型的参数。例如： ○ Parameter： `name=xiaoming` ○ Cookie： `hello = "world" 或 "world2"`

结果验证

本文仅通过一个示例介绍如何为应用创建标签路由，您可以为应用参照配置，然后根据实际业务需求进行验证。

2.7. 配置无损上线

对于任何一个线上应用来说，发布、扩容、缩容、重启等操作不可避免，MSE提供的一套无损上下线方案针对应用启动和下线中的多个阶段都提供了相应的保护能力，具体功能包含服务预热、服务延迟注册以及无损滚动发布等。

前提条件

ACK微服务应用接入MSE治理中心微服务治理

注意事项

- 对于Dubbo应用，当前仅Dubbo 2.7相关版本支持服务预热能力，Dubbo 2.6和Dubbo 3.0版本不支持。
- 对于Spring Cloud应用，当前仅支持利用Nacos、ZooKeeper以及Eureka这3种类型注册中心构建的应用进行服务预热。
- Spring Cloud服务预热功能是基于Spring Cloud框架默认的ZoneAwareLoadBalancer负载均衡类实现的，如果应用本身修改了该配置，会导致服务预热功能失效。
- 网关应用一般不通过注册中心调用，而是通过直接对外暴露API的方式调用，因此MSE当前所支持的小流量预热功能对该类应用不生效。另一方面，网关应用本身很少变更而且没有太多业务逻辑，因此也不太需要服务预热功能。

本文演示Demo是一个服务消费者spring-cloud-zuul持续调用到服务提供者spring-cloud-a的Spring Cloud应用，通过K8s的HPA定时对spring-cloud-a应用进行伸缩来模拟应用的上线与下线。

功能入口

1. 登录MSE治理中心控制台。

2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量治理 > 无损上下线**。
4. 在**应用列表**列单击目标应用名称，右侧显示该应用的无损上线配置以及相关的可观测数据。
   - 应用列表区域，显示当前应用列表信息。
   - 应用无损上线配置区域，您可以对目标应用配置无损上线规则。
   - 可观测数据概览区域，您可以查看应用事件统计信息，以及观测无损上下线事件，支持通过应用结束时间过滤。

> **注意** **无损上下线**页面中的可观测数据概览区域提供了无损上下线事件可视化功能，服务提供者和服务消费者需满足下列条件，才能保证数据的准确性：
>
> - 开通MSE微服务治理。
> - 打开应用无损上下线配置总开关。
>
> 如果不满足上述所有条件属于非正常使用，不能保证展示的数据准确。

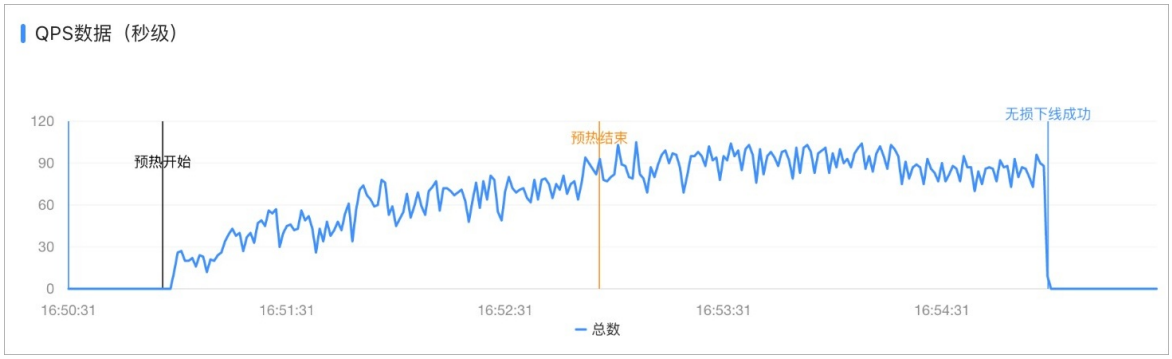
小流量预热服务

在较大流量下，刚启动的冷系统直接处理大量请求可能由于应用内部资源初始化不彻底从而出现请求阻塞、报错等问题。此时通过服务预热功能，在应用刚启动阶段通过小流量帮助应用在处理大量请求前完成初始化，可有效解决该类上线异常问题。

1. 在**无损上下线**页面的**应用无损上线配置**区域，单击右侧的**编辑**按钮。
2. 在**应用无损上线配置**对话框中配置**预热时长**和**预热曲线**，完成配置后单击**确定**。
   - **预热时长**：应用实例下一次启动的预热时间，默认预热时长为120秒。服务预热时长设置范围为0~86400秒（即24小时）。
   - **预热曲线**：默认为2（适合于一般预热场景），表示在预热周期内服务提供者的流量接收曲线形状呈2次曲线形状。预热曲线设置范围为0~20。
     相同预热时间，预热曲线值越大，表示预热开始将接收的流量越小，临近预热结束时接收的流量增幅越大。

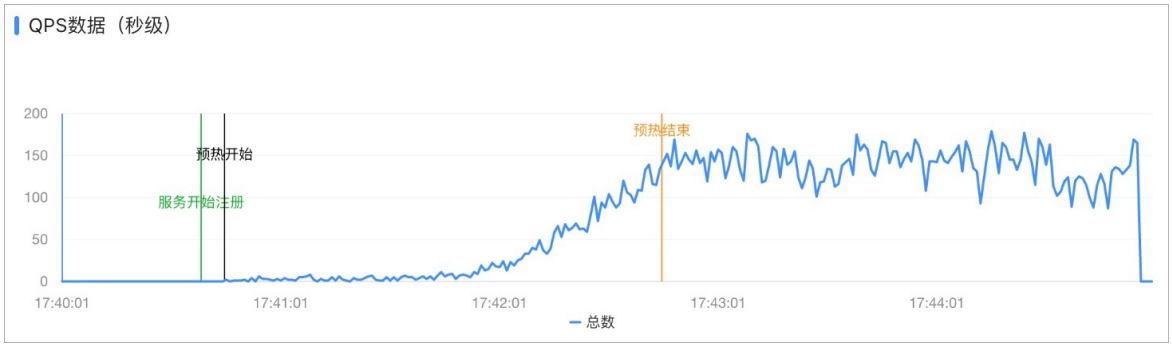
> **说明** 建议您在首次使用服务预热功能时，选择默认值。如果在使用默认值预热服务的过程中发现预热效果不明显，出现流量损失，可以通过调节该参数进行优化。

3. 打开**应用无损上线配置**右侧的总开关。
   服务预热开启后，待预热的应用将在预热周期内通过小流量实现应用启动过程的预热初始化。
   下图**预热时长**为120秒，**预热曲线**为2次的预热效果图：



> **说明** 该测试Demo是定时伸缩模拟应用启动，因此除了预热过程，还包含应用下线的过程。

下图**预热时长**为120秒，**预热曲线**为5次的预热效果图：



如上图所示，相比于2次预热过程，5次预热过程刚启动的这段时间（即17:41:01~17:42:01），QPS一直保持在一个较低值，以满足需要较长时间进行预热的复杂应用的预热需求。

延迟注册服务

对于初始化过程繁琐的应用，由于注册通常与应用初始化过程同步进行，从而出现应用还未完全初始化就已经被注册到注册中心供外部消费者调用，此时直接调用可能会导致请求报错。通过设置延迟注册，可让应用在充分初始化后再注册到注册中心对外提供服务。

1. 在**无损上下线**页面的**应用无损上线配置**区域，单击右侧的**编辑**按钮。
2. 在**应用无损上线配置**对话框中配置**延迟注册时间**，单位为秒，设置范围为0~86400秒（即24小时），完成配置后单击**确定**。
3. 打开**应用无损上线配置**右侧的总开关。
   该应用在下次重启时，延迟注册即可生效。您可以通过应用启动日志中的注册日志来查看延迟注册结果是否生效。

2.8. 配置无损下线

对于任何一个线上应用，在服务更新部署过程中，需要尽量保证客户端无感知，即从应用停止到重启恢复服务这个阶段不能影响正常的业务请求。在应用执行部署、停止、回滚、缩容、重置时，需要通过无损下线的配置来保证应用正常关闭。

> **注意**
> MSE治理中心暂不支持如下应用无损下线：
> - 暂不支持消费端为Spring Cloud LoadBalancer负载均衡应用的下游应用的无损下线。
> - 暂不支持提供端应用为WebFlux或者其他非SpringMVC应用的无损下线。
> - 暂不支持消费端为非微服务应用的下游提供者应用的无损下线。
> - 暂不支持非Java应用体系的无损下线。

视频教程

无损下线功能演示Demo链接：alibabacloud-microservice-demo。

为什么需要无损下线

应用从停止到恢复服务期间很难保证不影响正常运行的消费者的业务请求。理想条件下，在整个服务没有请求时进行更新是安全可靠的。但实际情况下，无法保证在服务下线的同时没有任何调用请求。

传统的解决方式是通过将应用更新流程划分为手工摘除流量、停应用、更新重启三个步骤，由人工操作实现客户端对更新无感知。

如果在容器或框架级别提供某种自动化机制，自动摘除流量并确保处理完已到达的请求，不仅能保证业务不受更新影响，还可以极大地提升更新应用时的运维效率，这种机制就是无损下线。

MSE治理中心无损下线的优势

- 对于开源Spring Cloud可以通过shutdownHook、Spring Boot Actuator和Ribbon实现无损下线，不仅有一定的开发工作量，而且部分注册中心会导致短暂的流量损失。
- 对于开源Dubbo可以通过shutdownHook和QoS实现无损下线，不仅有一定的开发工作量，而且对Dubbo有版本要求，还有一些遗留问题，最终影响正常使用。

MSE治理中心将无损下线的流程整合在发布流程中，对应用进行停止、部署、回滚、缩容、重置等操作时，无损下线会自动执行。相对于开源的Spring Cloud和Dubbo方案，MSE治理中心无损下线分别具有以下优势。

- 相对于开源的Spring Cloud方案，MSE治理中心无损下线具有以下优势：

分类	开源Spring Cloud	MSE治理中心
版本	使用ServiceRegistryEndpoint，需要依赖Actuator组件，且需要升级到适配的版本。	无需任何操作，无侵入地支持Spring Cloud Dalston及以上版本。
注册中心和流量损失	依赖注册中心，有些注册中心会导致流量损失。 ○ ZooKeeper不存在流量损失。 ○ Eureka存在3s流量损失。 ○ Nacos存在客户端缓存，会造成最长10s的流量损失。	无需依赖任何注册中心，对于任何注册均不存在流量损失。
场景	ECS场景需要结合变更详情；K8s场景可以配合prestop接口，但是prestop接口只能配置一个动作。	ECS和K8s全部覆盖，且不影响对应用的任何操作与配置。
客户端缓存	需要权衡利弊配置合理的Ribbon缓存的刷新时间，过长会导致下线有流量损失，过短会影响性能。	增强Ribbon下线刷新机制，通过反应式响应方式主动刷新Ribbon缓存，您无需关心缓存刷新。

- 相对于开源的Dubbo方案，MSE治理中心无损下线具有以下优势：

> **说明** 配置无损下线需要服务的消费者与提供者均接入MSE服务治理。

分类	开源Dubbo	MSE治理中心
注册中心的 unexport 未采用原子变量导致并发问题	2.5.3及之前版本不支持	支持
反向通知	2.5.3及之前版本不支持	支持
客户端等待在途请求	2.5.3及之前版本不支持	支持
qos offline	2.5.8-2.6.2及之后的版本支持	支持
Dubbo 与 Spring ShutDownHook 触发问题	仅Apache Dubbo 2.7.3+支持	支持
下线事件	仅Apache Dubbo 2.7.3+支持	支持
服务端等待	不支持	支持
提前sendReadOnly	不支持	支持

如何验证无损下线是否生效

您可以直接根据实际业务验证应用的无损下线是否已经生效。另外，MSE治理中心也提供了两个应用Demo，您可以使用这两个Demo在容器服务K8s集群中验证无损下线。您可以通过以下任一种方式验证无损下线：

无损下线验证方式一：

1. 下载应用Demo（Provider和Consumer）。
2. 将应用Demo部署到容器服务K8s集群。相关操作，请参见创建无状态工作负载Deployment。

   其中，Provider的实例个数为2，Consumer的实例个数为1。
3. 查看应用调用现状。
   i. 登录部署Consumer的Pod，执行以下命令，该命令会不停地访问服务端的服务。

   ```
   #!/usr/bin/env bash
   while true
   do
       echo `curl -s -XGET http://localhost:18091/user/rest`
   done
   ```

   ii. 查看调用请求的响应。

   ```
   [root@sc-consumer-group-1-1-65fdddf668-s8ssk admin]# sh a.sh
   Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
   Hello from [18084]172.20.0.221! 2020-03-23 10:44:23
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:23
   Hello from [18084]172.20.0.221! 2020-03-23 10:44:23
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:23
   Hello from [18084]172.20.0.223! 2020-03-23 10:44:23
   ```

   从响应中可以看到， Consumer随机访问Provider的两个实例（IP为172.20.0.221和172.20.0.223）。

   > **注意** 调用请求的响应窗口不要关闭，后续仍然会用到。
4. 将Provider的实例缩容到1，模拟实例重启的场景。
5. 再次查看调用请求的响应结果，验证无损下线。

```
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
```

一直观察客户端请求情况，可以看到无损下线的情况，同时观察客户端日志，不存在任何相关问题，客户端完全无感知。

从响应中可以看到，Consumer会固定访问Provider剩余的一个实例（IP为172.20.0.221），而不会发生调用异常，避免影响Consumer。

无损下线验证方式二：

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量治理 > 无损上下线**。
3. 在**应用列表**列单击目标应用名称，右侧显示该应用的无损上线配置以及相关的可观测数据。
   - 应用列表区域，显示当前应用列表信息。
   - 应用无损上线配置区域，您可以对目标应用配置无损上线规则。
   - 可观测数据概览区域，您可以查看应用事件统计信息，以及观测无损上下线事件，支持通过应用结束时间过滤。

> **注意** **无损上下线**页面中的可观测数据概览区域提供了无损上下线事件可视化功能，服务提供者和服务消费者需满足下列条件，才能保证数据的准确性：
>
> - 开通MSE微服务治理。
> - 打开应用无损上下线配置总开关。
>
> 如果不满足上述所有条件属于非正常使用，不能保证展示的数据准确。

如上图所示，在可观测数据概览区域，可见图中实例最后无损下线成功。

2.9. 无损滚动发布

K8s的滚动发布是将一次完整的发布过程分成多个批次，每次发布一个批次，成功后，再发布下一个批次，最终完成所有批次的发布。MSE通过提供确保在通过就绪检查前完成服务注册以及确保在就绪检查前完成服务预热等无损上线能力，帮助您在应用的滚动发布过程中，始终保证有可用的服务实例（副本）在运行，从而实现应用的无损滚动发布。

前提条件

ACK微服务应用接入MSE治理中心微服务治理

注意事项

本文演示Demo是一个服务消费者spring-cloud-zuul持续调用到服务提供者spring-cloud-a的Spring Cloud应用，通过K8s的HPA定时对spring-cloud-a应用进行伸缩来模拟应用的上线与下线。小流量服务预热相关功能基于微服务应用场景设计实现，在使用就绪检查关联服务预热功能前请确认应用已满足下列条件：

- 对于Dubbo应用，当前仅Dubbo 2.7相关版本支持服务预热能力，Dubbo 2.6和Dubbo 3.0版本不支持。
- 对于Spring Cloud应用，当前仅支持利用Nacos、ZooKeeper以及Eureka这3种类型注册中心构建的应用进行服务预热。
- Spring Cloud服务预热功能是基于Spring Cloud框架默认的ZoneAwareLoadBalancer负载均衡类实现的，如果应用本身修改了该配置，会导致服务预热功能失效。
- 网关应用一般不通过注册中心调用，而是通过直接对外暴露API的方式调用，因此MSE当前所支持的小流量预热功能对该类应用不生效。另一方面，网关应用本身很少变更而且没有太多业务逻辑，因此也不太需要服务预热功能。

> **说明** 无损滚动发布功能目前正在公测中，如果您需要使用该功能，请提交工单或者加入微服务引擎钉钉交流群：34754806，联系开发同学升级应用并试用。

功能入口

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量治理 > 无损上下线**。
4. 在**应用列表**列单击目标应用名称，右侧显示该应用的无损上线配置以及相关的可观测数据。
   - 应用列表区域，显示当前应用列表信息。
   - 应用无损上线配置区域，您可以对目标应用配置无损上线规则。
   - 可观测数据概览区域，您可以查看应用事件统计信息，以及观测无损上下线事件，支持通过应用结束时间过滤。

> **注意** **无损上下线**页面中的可观测数据概览区域提供了无损上下线事件可视化功能，服务提供者和服务消费者需满足下列条件，才能保证数据的准确性：
>
> - 开通MSE微服务治理。
> - 打开应用无损上下线配置总开关。
>
> 如果不满足上述所有条件属于非正常使用，不能保证展示的数据准确。

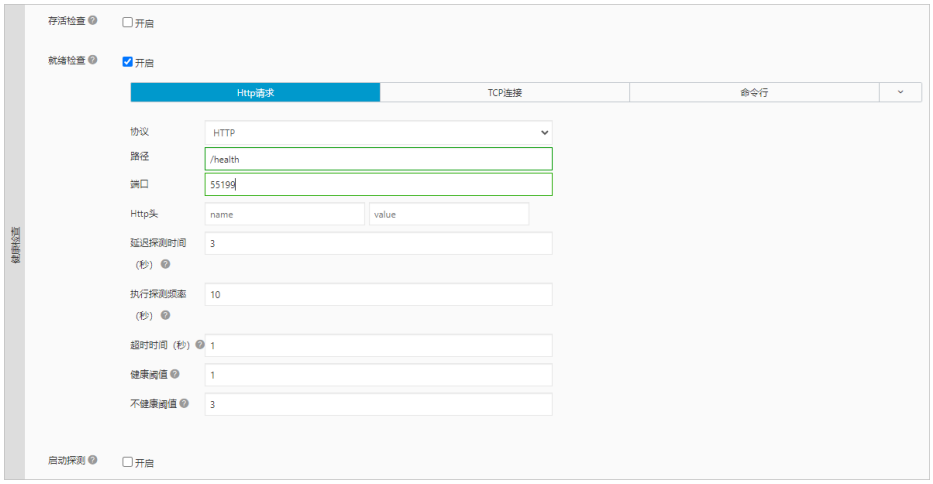
通过就绪检查前完成服务注册

K8s提供就绪检查机制，对实例在就绪前进行健康检查，但K8s并不能感知到微服务应用什么时候就绪，通常情况下，认为端口可连接应用即处于就绪态，这样会造成刚启动的服务可能未完全注册到注册中心，老应用实例就被下线，导致消费端调用异常。

微服务生命周期关联K8s就绪检查功能开启后，可通过Agent无侵入为应用提供一个检测其是否完成注册的端口，当应用注册完成返回200帮助K8s判定应用已就绪，未完成注册返回500帮助K8s判定应用未就绪。

1. 在**无损上下线**页面的**应用无损上线配置**区域，单击右侧的**编辑**按钮。
2. 在**应用无损上线配置**对话框中，单击**无损滚动发布**右侧的展开图标，开启**通过就绪检查前完成服务注册**开关，然后单击**确定**。

3. 打开**应用无损上线配置**右侧的总开关。
4. 在容器服务控制台中，阿里云容器ACK服务对应应用配置的中**健康检查**区域，选中**就绪检查**右侧的**开启**，配置如下参数，然后单击**更新**。
   - **路径**：*/health*。
   - **端口**：*55199*。
   - 其他参数默认即可。

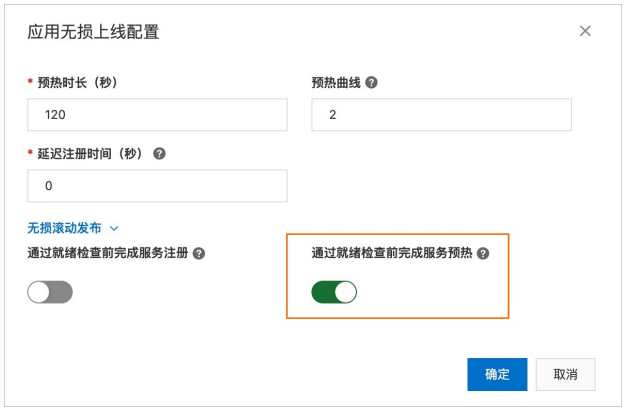


该应用在下次重启时，该配置即可生效。

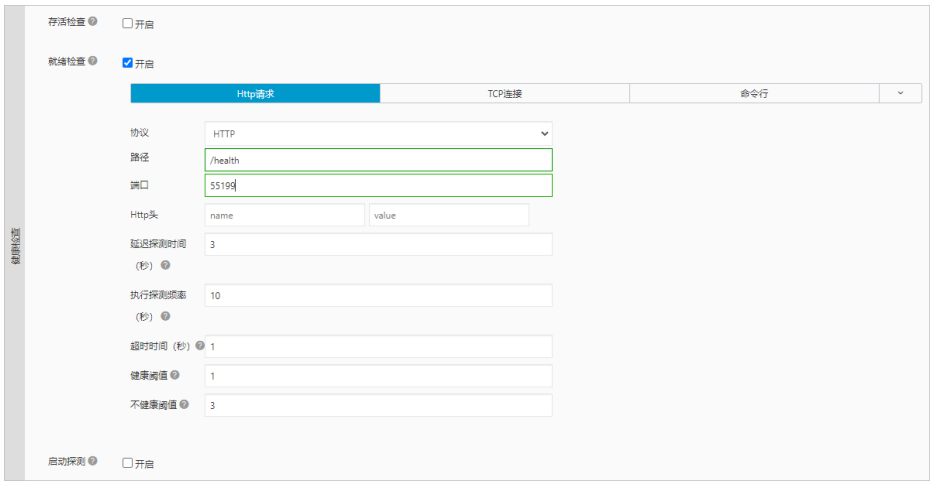
通过就绪检查前完成服务预热

除了可以单独使用服务预热，还可以将其关联K8s就绪检查机制，当应用服务预热完成后，K8s才将应用置为就绪状态。

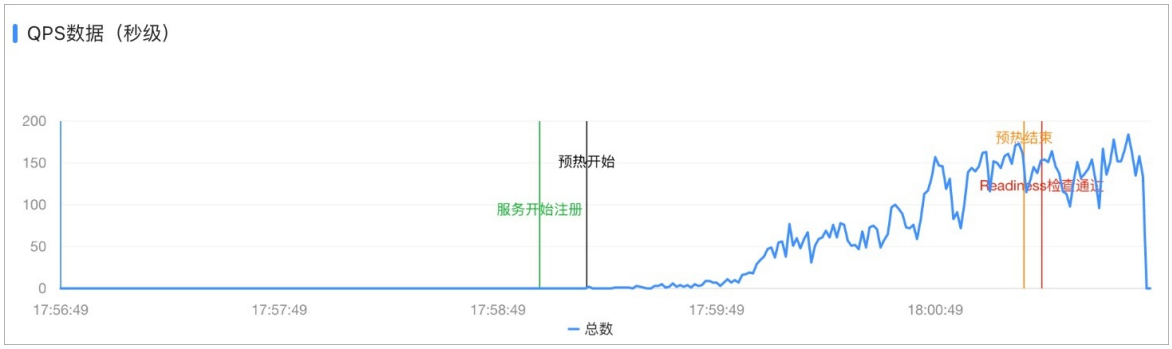
1. 在**无损上下线**页面的**应用无损上线配置**区域，单击右侧的**编辑**按钮。
2. 在**应用无损上线配置**对话框中，单击**无损滚动发布**右侧的展开图标，开启**通过就绪检查前完成服务预热**开关，然后单击**确定**。



3. 打开**应用无损上线配置**右侧的总开关。
4. 在容器服务控制台中，阿里云容器ACK服务对应应用配置的中**健康检查**区域，选中**就绪检查**右侧的**开启**，配置如下参数，然后单击**更新**。
   - **路径**：*/health*。
   - **端口**：*55199*。
   - 其他参数默认即可。



该应用在下次重启时，该配置即可生效。具体效果如下图所示（就绪检查事件在服务预热结束之后发出）：



2.10. 服务实例隔离与诊断

本文介绍服务实例隔离与诊断。

前提条件

使用服务实例隔离与诊断功能，需要满足以下条件：

- 当前仅支持Java语言相关的Dubbo 2.6.x，2.7.x和Spring Cloud E及以上版本应用。
- Spring Cloud应用不支持设置为spring.cloud.xxxx.discovery.fail-fast=false的应用上下线状态判断。
- 生成异常实例的内存快照需要将实例接入阿里云应用实例监控服务ARMS，具体请参见接入指南。

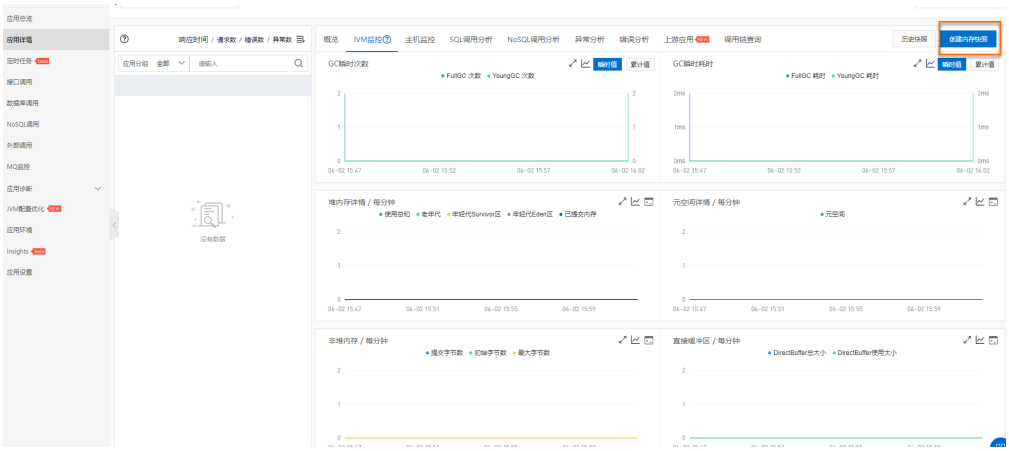
背景信息

在线上微服务场景中，当服务提供者的某些实例出现异常时，一方面，需要避免服务消费者访问到异常实例，另一方面，需要保留异常现场，便于后续的问题排查。治理中心服务实例隔离与诊断功能可帮助您及时将异常实例从注册中心摘除，然后结合阿里云应用实时监控服务ARMS所提供的内存快照生成能力，及时生成异常实例的线上环境内存快照，帮助您进行后续问题分析与诊断。服务实例隔离与诊断功能能很好地帮助您应对线上突发的事故（比如内存泄露），提升微服务系统整体稳定性。

操作步骤

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
3. 单击目标应用，进入**应用详情**页面。单击**实例列表**页签，选择异常实例，单击**操作**下方的**服务下线**，并在**微服务下线**弹框中单击确定，即可将实例从注册中心移除。
   实例从注册中心移除后，若该实例已无新请求，可通过阿里云应用监控服务ARMS提供的创建内存快照功能，给异常实例创建内存快照，以便后续进一步的问题排查。
   i. 在**实例列表**面板的**操作**列下方单击**去创建内存快照**，在**提示**弹框单击**确定**。

ii. 在**应用详情**页面单击右上角**创建内存快照**，然后在弹框页面**创建内存快照**中单击**保存**给异常实例创建内存快照。



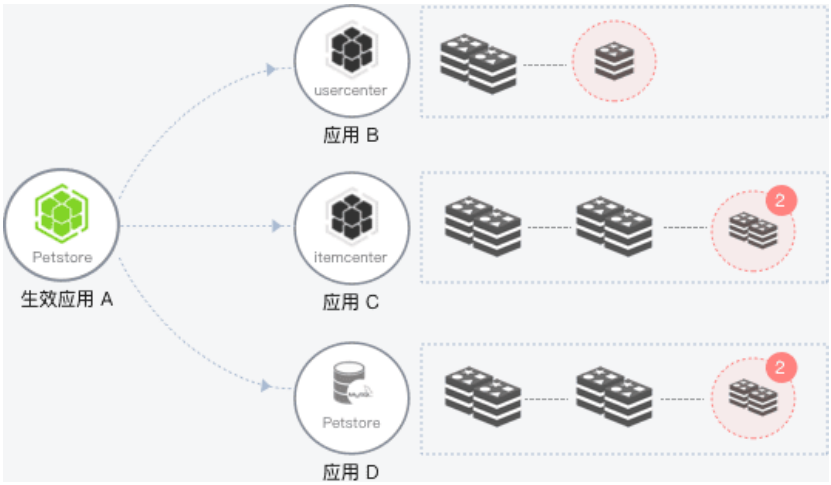
2.11. 摘除离群实例

在微服务架构中，当服务提供者的应用实例出现异常，而服务消费者无法感知时会影响服务的正常调用，并影响消费者的服务性能甚至可用性。离群实例摘除功能会检测应用实例的可用性并进行动态调整，以保证服务成功调用，从而提升业务的稳定性和服务质量。

背景信息

在下图的示例场景中，某个系统包含4个应用，A、B、C和D，其中应用A会分别调用应用B、C和D。当应用B、C或D的某些实例异常时（如图中红色圆圈所示，应用B有一个异常实例，C和D有2个异常实例），如果应用A无法感知，会导致部分调用失败；如果B、C、D的异常实例较多，有可能影响应用A的性能甚至服务可用性。

为了保护应用A的服务性能和可用性，可以为应用A配置离群实例摘除。配置后，即可监控B、C、D应用的实例状态并进行动态调整（摘除或添加），以保证服务成功调用。



离群实例摘除流程如下：

1. 当应用B、C或D的某个实例异常时，系统能够检测到，并根据配置的**摘除实例比例上限**判断是否将对应的实例从应用中摘除。
2. 摘除实例后，A的调用请求不再被分发到B、C、D的异常实例上。
3. 按配置的**恢复检测单位时间**开始检测异常实例是否恢复。
4. 检测间隔随检测次数按**恢复检测单位时间**（默认为0.5分钟）线性增加，当达到设置的**未恢复累计次数上限**后，会按最长时间间隔持续检测异常实例是否恢复。
5. 当检测到实例恢复后，将实例重新添加到应用的实例列表中，处理调用请求。同时，将检测间隔重置为**恢复**

检测单位时间，例如0.5分钟。

> 说明
> - 当提供者应用的异常实例数量过多（超过摘除实例比例上限）时，仅按照设置的比例摘除。
> - 当提供者应用中仅剩最后一个可用实例时，即使错误率超过配置的阈值，也不会摘除该实例。

视频教程

创建离群实例摘除策略

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量治理 > 离群实例摘除**。
3. 在**离群实例摘除**页面顶部菜单栏单击**创建离群实例摘除策略**。并在在**创建离群实例摘除**面板中配置相关参数，然后单击确定。
4. 在**创建离群实例摘除**面板中配置相关参数，然后单击**确定**。

   离群实例摘除策略参数说明如下。

参数	描述
策略名称	自定义策略名称，最长64个字符。
被调用服务所用框架	根据需要选择Spring Cloud或者Dubbo。
选择生效应用	选择生效应用，单击>，被选择的应用迁移至**已选应用**。 **说明** 选择生效应用后，该应用调用的所有应用的异常实例会被摘除。摘除期间，生效应用的调用请求将不再被分发到异常实例。
错误率域值	被调用的应用中某个应用实例的错误率高于设置的域值后，将摘除该实例。默认值为50%。例如该实例在统计时间窗口内被调用10次，有6次调用失败，错误率为60%，超过了配置的错误率域值（50%），则从应用中移除该实例。
高级配置	
异常类型	包含**网络异常**和**网络异常 + 业务异常（HTTP 5xx）**，根据实际业务需求选择。
QPS下限	QPS按照统计时间窗口进行计算，Dubbo 2.7版本的应用的统计时间窗口为15秒，其它Dubbo版本和Spring Cloud应用的统计时间窗口为10秒。当在统计时间窗口（例如15秒）内应用的QPS达到设置的下限后开始进行错误率统计分析。
摘除实例比例上限	摘除的异常实例比例上限，即达到阈值后，不再摘除异常实例。摘除异常实例数向下取整，例如应用实例总数为6，摘除实例比例设置为60%，摘除实例比例数为6 * 60% = 3.6，则按策略最多摘除的实例数为3。若计算结果小于1，则不会摘除实例。

参数	描述
恢复检测单位时间	在异常实例被摘除后，不断按单位时间线性累加的时间作为检测间隔，去检测异常实例是否恢复正常，单位为ms。默认为30000 ms，即0.5分钟。
未恢复累计次数上限	持续对异常实例进行检测，检测间隔随检测次数按恢复检测单位时间线性增加，当达到设置的检测次数上限后，会按最长时间间隔持续检测异常实例是否恢复。例如**恢复检测单位时间**设置30000 ms，**未恢复累计次数上限**设置为20，在第20次检测异常实例仍未恢复后，则会按10分钟（20 x 30000 ms）为间隔执行后续的检测。如果检测到实例已经恢复，则会将检测间隔重置为初始的时间间隔，即一次**恢复检测单位时间**。 **说明** **未恢复累计次数上限**不建议配置太大。配置太大会导致最长检测间隔时间较长，如果实例在检测间隔早期恢复，仍需等到检测间隔到时再进行检测，导致期间实例资源被浪费，未能及时处理业务调用请求。
默认状态	默认开启离群实例摘除策略。

结果验证

返回**离群实例摘除**页面，查看刚创建的策略是否已显示在策略列表中。

后续步骤

在策略列表的**操作**列单击**编辑**或**删除**，可以编辑或删除离群实例摘除策略。

3.流量防护

3.1. 应用防护

3.1.1. 什么是应用防护

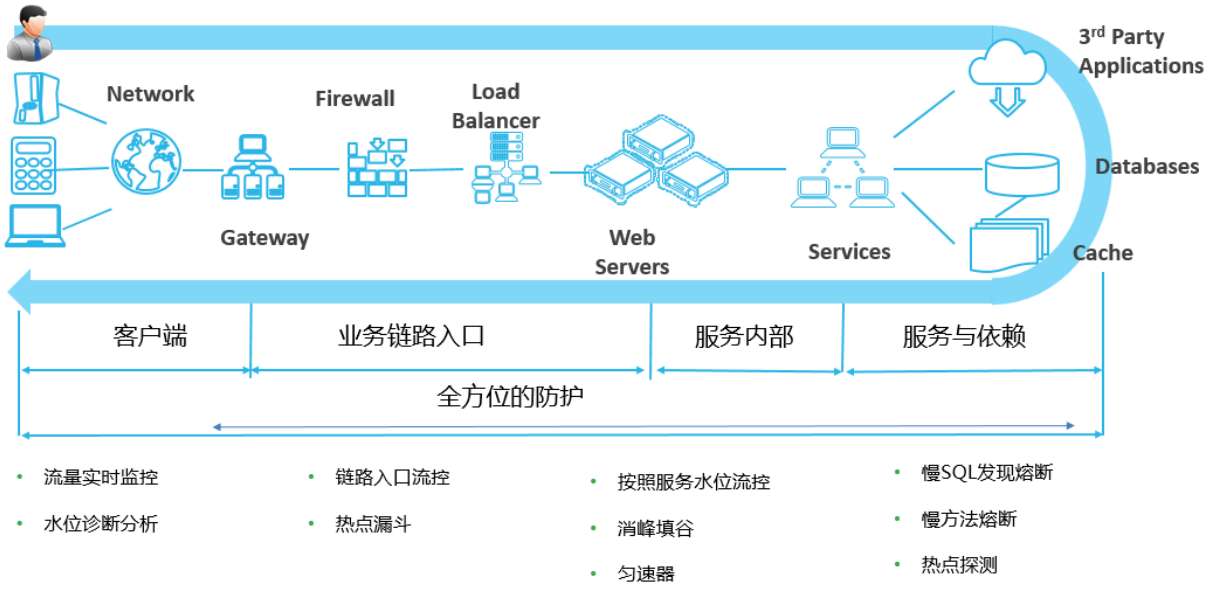
应用防护以流量为切入点，从流量控制、熔断降级、系统负载保护等多个维度来保障业务的稳定性，提供更专业稳定的流量防护手段、秒级的流量水位分布分析功能，是阿里巴巴双十一技术体系中的核心组件，同时也是开源框架Sentinel的商业化产品。

使用场景

> 注意 目前应用防护处于灰度状态，如果您对这些功能有诉求，您可以按照ACK微服务应用接入MSE服务治理企业版操作，通过增加白名单方式体验该功能。

应用防护广泛用于秒杀场景、消息削峰填谷、集群流量控制、实时熔断等场景中，从多个维度保障您的业务稳定性。

在一个常见的分布式应用中，如下图所示。一个请求先通过终端到达Gateway，再经过防火墙和网络负载均衡，其中还包括调用下游的其它服务和第三方应用，才能到达前端网络服务。应用防护在不同的层次以流量为切面提供秒级实时的流量分析（例如在客户端层提供流量实时监控和水位诊断分析功能），帮助运维人员采取针对性的防护措施，全方位地保护应用的稳定性。



功能特性

- 秒级流量分析功能，动态规则实时推送。
- 专业多样化的防护手段：
  - 入口流量控制：按照服务容量进行流量控制，常用于应用入口，例如：Gateway、前端应用、服务提供方等。
  - 热点隔离：将热点和普通流量隔离出来，避免无效热点抢占正常流量的容量。
  - 对依赖方隔离或降级：对应用和应用之间、应用内部采用隔离或降级手段，将不稳定的依赖的对应用的影响减至最小，从而保证应用的稳定性。
  - 系统防护：应用防护可以根据系统的能力（例如Load、CPU使用率等）来动态调节入口的流量，保证系统稳定性。

- 实时的单机监控能力，强大的聚合监控和历史监控查询能力。

参考文档

关于Sentinel的详细介绍，请参见Sentinel介绍。

3.1.2. 应用防护规则适用场景

微服务的稳定性一直是您非常关注的话题。随着业务从单体架构向分布式架构演进以及部署方式的变化，服务之间的依赖关系变得越来越复杂，业务系统也面临着巨大的高可用挑战。MSE应用防护就是一款借助流量控制、熔断降级等模块，来提高应用高可用能力的产品。本文介绍各个应用防护规则以及适用的场景。

不稳定场景

在生产环境中您可能遇到过以下不稳定的情况：

- 大促时瞬间洪峰流量使得系统超出最大负载、Load飙高、系统崩溃导致用户无法下单。
- “黑马”热点商品击穿缓存、数据库被打垮、挤占正常流量。
- 调用端被不稳定第三方服务拖垮、线程池被占满、调用堆积，导致整个调用链路卡死。

这些不稳定的场景可能会导致严重后果，但很多时候开发者容易忽视这些与流量、依赖相关的高可用防护。MSE应用防护功能就可以预防这些不稳定因素带来的影响，针对流量进行高可用的防护，从而保障服务“稳如磐石”。

核心场景

应用防护各规则说明和适用的核心场景如下表所示。

应用防护规则	描述	核心场景	说明文档
流量控制规则	通过MSE配置QPS模式的流控规则，当每秒的请求量超过设定的阈值时，会自动拒绝多余的请求。	适用于需要限制突发的流量，在尽可能处理请求的同时来保障服务不被击垮的场景。	配置流控规则
集群流控规则	控制某个服务调用整个集群的实时调用量，解决因流量不均导致总体限流效果不佳的问题。	• 单机流量不均导致的单机限流效果不佳 • 集群小流量流控 • 有业务含义的流量控制（分钟小时级）	配置集群流控规则

应用防护规则	描述	核心场景	说明文档
隔离规则	控制某些调用的并发数（即正在进行的数目），防止过多的慢调用挤占正常的调用。	在调用第三方服务时，防止过多的慢调用挤占正常调用的资源，避免服务不可用。	配置隔离规则
熔断规则	对不稳定的弱依赖调用进行自动熔断降级，暂时切断不稳定调用，避免局部不稳定因素导致整体的雪崩。	避免局部不稳定因素（某个慢调用、异常服务）导致整体的雪崩，例如切断某个RT高的第三方服务调用，或针对某个ID的慢SQL访问进行熔断。	配置熔断规则
热点防护规则	自动识别热点参数并控制每个热点值的访问频次或并发量，可以有效地防止过“热”的参数访问挤占正常的调用资源。	适用于针对某些热点数据中访问频次最高的Top数据进行控制的场景，例如针对一段时间内最频繁购买的商品ID进行控制，防止突发热点商品击穿缓存而导致大量请求到数据库的情形。	配置热点规则
系统自适应保护规则	结合系统指标和服务容量，自适应动态调整流量。	用作全局的兜底防护规则。	自适应流控

流量控制规则

场景说明

流量是随机、不可预测的，可能就在某一时间会出现流量洪峰，例如双十一零点的场景。然而系统的容量总是有限的，如果突如其来的流量超过了系统的承受能力，就可能会导致请求处理堆栈、堆积的请求处理缓慢、CPU/Load飙高，最后导致系统崩溃。因此，您需要针对这种突发的流量来进行限制，在尽可能处理请求的同时来保障服务不被击垮，这就是流量控制。流量控制的场景是非常通用的，适用于脉冲流量类等场景。

流控规则说明

通常在Web入口或服务提供方（Service Provider）的场景下，需要保护服务提供方自身不被流量洪峰打垮。此时通常根据服务提供方的服务能力进行流量控制，或针对特定的服务调用方进行限制。您可以结合前期压测评估核心接口的承受能力，通过MSE配置QPS模式的流控规则，当每秒的请求量超过设定的阈值时，会自动拒绝多余的请求。关于流控规则的更多信息，请参见配置流控规则。

集群流控规则说明

同时MSE也提供集群流控（分布式限流）的能力，可以控制某个服务调用整个集群的实时调用量，解决因流量不均导致总体限流效果不佳的问题。关于集群流控的更多信息，请参见配置集群流控规则。

如果您的业务符合以下场景，建议结合集群流控来保障服务稳定性：

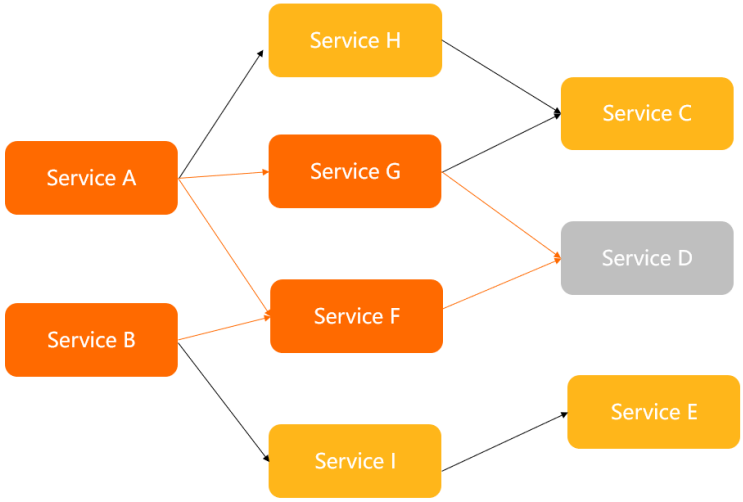
- 单机流量不均：由于负载不均衡等原因导致每台机器的流量不均，此时使用单机流控可能会出现没有达到请求总量，某些机器就开始限流的情况。
- 集群小流量流控：某些高可用防护场景下，需要将服务调用QPS限制到很小的量，此时平均到每台机器的QPS可能小于1，无法通过单机流控进行精确控制。例如限制总QPS为50，但节点数为100个，平均到每个节点QPS为0.5。
- 有业务含义的流量控制：例如限制某个API每个用户每分钟调用不超过10次。

并发控制与熔断规则

场景说明

一个服务常常会调用别的模块，可能是另外一个远程服务、数据库，或者第三方API等。例如，支付的时候，可能需要远程调用银联提供的API；查询某个商品的价格，可能需要进行数据库查询。然而，这个被依赖服务的稳定性是不能保证的。如果依赖的服务出现了不稳定的情况，请求的响应时间变长，则调用服务的方法的响应时间也会变长，线程会产生堆积，最终可能耗尽业务自身的线程池，服务本身也变得不可用。

微服务的架构示例图如下：



现代微服务架构都是分布式的，由许多微服务组成。不同服务之间相互调用，组成复杂的调用链路。以上的问题在链路调用中会产生放大的效果。复杂链路上的某一环不稳定，就可能会层层级联，最终导致整个链路都不可用。

规则说明

MSE提供以下能力避免慢调用等不稳定因素造成服务不可用：

- 并发控制：作为一种轻量级隔离的手段，控制某些调用的并发数（即正在进行的数目），防止过多的慢调用挤占正常的调用。更多信息，请参见配置隔离规则。
- 熔断：对不稳定的弱依赖调用进行自动熔断降级，暂时切断不稳定调用，避免局部不稳定因素导致整体的雪崩。更多信息，请参见配置熔断规则。
  MSE熔断规则基于熔断器模式的思想，在服务出现不稳定因素（例如响应时间变长、错误率上升）的时候暂时切断服务的调用，等待一段时间再进行尝试。一方面防止给不稳定服务“雪上加霜”，另一方面保护服务的调用方不被拖垮。MSE应用防护支持两种熔断策略：基于响应时间（慢调用比例）和基于错误（异常比例），可以有效地针对各种不稳定的场景进行防护。

> **说明**
> - 熔断器模式一般适用于弱依赖调用，即降级后不影响业务主流程，开发者需要设计好降级后的回退逻辑（fallback）和返回值。
> - 即使服务调用方引入了熔断降级机制，开发者还是需要在HTTP或RPC客户端配置请求超时时间，来做一个兜底的防护。

热点防护规则

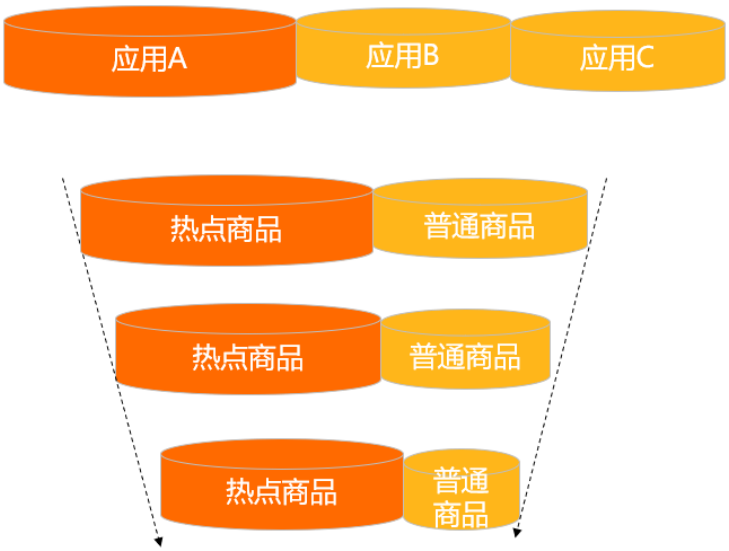
场景说明

为了防止被大流量打垮，您通常会对核心接口配置限流规则，但有的场景下配置普通的流控规则是不够的。

例如大促峰值的时候，会有不少“热点”商品，这些热点商品的瞬时访问量非常高。通常您可以事先预测一波热点商品，并对这些商品信息进行缓存“预热”，以便在出现大量访问时可以快速返回而不会都经过数据库。但每次大促都会涌现出一些“黑马”商品，这些“黑马”商品是无法事先预测的，没有被预热。当这些“黑马”商品访问量激增时，大量的请求会击穿缓存，直接经过数据库，导致数据库访问缓慢，挤占正常商品请求的资源池，最后导致系统崩溃。

规则说明

此时，利用MSE的热点参数流控能力，自动识别热点参数并控制每个热点值的访问频次或并发量，可以有效地防止过“热”的参数访问挤占正常的调用资源。更多信息，请参见配置热点规则。



系统自适应保护规则

场景说明

当您无法事先准确评估某个接口的容量，甚至无法预知核心接口的流量特征（如是否有脉冲情况）时，靠事先配置的规则可能无法有效地保护当前服务节点。当某些情况下机器的Load和CPU usage等突然开始飙高，但您却无法快速确认原因，也来不及处理异常时，您其实需要做的是快速止损，先通过自动化的兜底防护手段，将濒临崩溃的微服务“拉”回来。

规则说明

针对这些情况，MSE提供了一种系统自适应保护规则，结合系统指标和服务容量，自适应动态调整流量。MSE自适应流控结合系统的Load、CPU使用率以及服务的入口QPS、响应时间和并发量等几个维度的监控指标，通过一定的流控策略，让系统的入口流量和系统的负载达到一个平衡，让系统尽可能运行在最大吞吐量，同时保证系统整体的稳定性。更多信息，请参见自适应流控。

系统规则可以作为整个服务的一个兜底防护策略，保障服务运行，对CPU密集型的场景会有较好的效果。

3.1.3. 支持组件列表

MSE为了简化应用的接入流程，对主流框架进行了适配。本文将列出MSE支持的第三方组件和框架列表。

组件	支持版本	支持该组件的Java Agent版本
Dubbo	Agent：2.7.x，2.6.x	All
Web Servlet	Agent：3.0+	All
Spring Boot	1.3.x+	All

组件	支持版本	支持该组件的Java Agent版本
Spring MVC	4.x+	All
Spring Cloud Gateway	2.x	1.5.0+
Zuul 1.x	1.3.x	1.5.0+
GRPC-Java	1.13+	1.7.0
Jetty	8.x+	Servlet 3.0+ 支持：all
Tomcat	7.x+	Servlet 3.0+ 支持：all
WebLogic	10.3	Servlet 3.0+ 支持：all Servlet 2.x支持：1.6.0+
HttpClient 3	3.x+	待支持
HttpClient 4	4.x+	待支持
JDK HTTP	1.7.x+	待支持
OKHttp	2.x+	待支持
MyBatis	3.x+	1.8.0+
MySQL JDBC	5.0.x+	1.6.0+
Oracle JDBC	12.x	1.6.0+
PostgreSql JDBC	9.4+	待支持
SQLServer JDBC	6.4+	待支持
Redis Client（Jedis）	待支持	1.7.0
MemCached	2.8+	1.7.0
MongoDB	3.7+	待支持
RocketMQ （callback模式）	4.x	1.7.0
RabbitMQ	3.7+	1.7.0
SOFARPC	5.x	待支持

说明

- Spring Boot/Spring Cloud Web应用只需要引入 `spring-boot-starter-MSE-sentinel-client` 依赖即可接入。
- Spring Cloud Gateway网关需要引入 `spring-cloud-gateway-starter-MSE-sentinel` 依赖；Zuul 1.x网关需要引入 `spring-cloud-zuul-starter-MSE-sentinel` 依赖，无需引入其它依赖。

3.1.4. 配置规则

3.1.4.1. 配置流控规则

配置流控规则的原理是监控应用或服务流量的QPS指标，当指标达到设定的阈值时立即拦截流量，避免应用被瞬时的流量高峰冲垮，从而保障应用高可用性。本文介绍如何配置管理流控规则，以及三种常用场景的流控配置规则。

前提条件

- 开通企业版。相关内容，请参见微服务治理升级为企业版。
- MSE治理中心已接入微服务应用，相关内容，请参见：
  - ACK微服务应用接入MSE治理中心
  - ECS微服务应用接入MSE治理中心

背景信息

流量控制在网络传输中是一个常用的概念，常用于调整网络包的发送数据。系统需处理的请求是随机不可控的，而系统的处理能力是有限的，因此就需要根据系统的处理能力对流量进行控制。

功能入口

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 选择以下任意一种方法进入**新建流控规则**页面：
   - 在左侧导航栏单击**应用概览**，然后单击页面下方目标接口**操作**列中的**流控**。
   - 在左侧导航栏单击**接口详情**，在**接口详情**页面单击资源卡片右上角 ＋ 或 ⚙ 的图标，然后在**管理规则**对话框中单击**新增流控规则**。
   - 在左侧导航栏单击**规则设置**，在**流控规则**页签的左上角单击**新增流控规则**。
6. 在**新建流控规则**对话框中配置规则信息，参数说明请参见更多信息。
7. 单击**新建**。

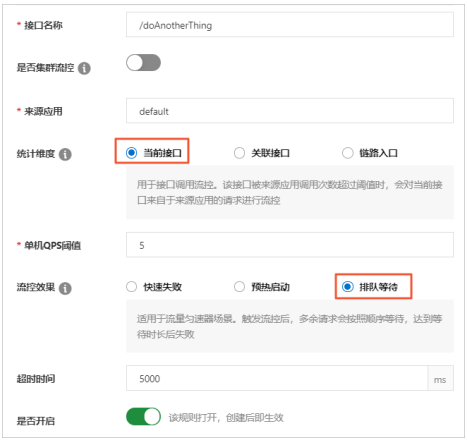
常用场景1：削峰填谷，使流量匀速通过

请求流量具有波峰波谷的特点，流控的原理是将前面的峰值流量延迟（排队时长）到后面再处理，既能最大化满足所有请求，又能保证用户体验。详情请参见削峰填谷。

在**新建流控规则**对话框中配置以下规则信息：

- **统计维度**选择**当前接口**。
- **流控效果**选择**排队等待**。
- 配置匀速模式下请求**单机QPS阈值**为5。
- **等待时长**为5s。

系统则每200 ms处理一条请求，多余的处理任务将排队；同时设置了等待时长为5s，则预计排队时长超过5s的处理任务将快速失败，直接返回默认流控信息，如文本、静态页面等。



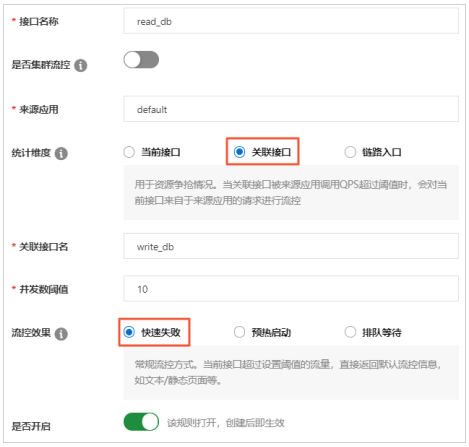
常用场景2：当资源争抢时，需留足资源给优先级高的接口

`read_db` 和 `write_db` 这两个资源分别代表数据库读写。为保证提交的数据不丢失，`write_db` 接口优先级更高。当写库操作过于频繁时，读数据的请求会被限流。详情请参见关联限流。

在**新建流控规则**对话框中配置以下规则信息：

- **统计维度**选择**关联接口**。
- **流控效果**选择**快速失败**。
- **并发数阈值**为10。

当 `write_db` 资源的QPS超过10之后，`read_db` 会被限流以保证留足资源给 `write_db`，避免 `write_db` 数据丢失。



常用场景3：预热启动避免大流量冲击

流控的原理是在流量入口处控制流量，让通过的流量缓慢增加，在一定时间内逐渐增加到阈值上限，以便系统可以预热。最适合突发流量的场景。详情请参见Warm Up（冷启动）。

在**新建流控规则**对话框中配置以下规则信息：

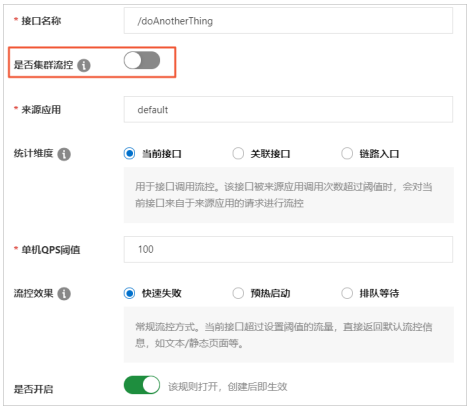
- **统计维度**选择**链路入口**。
- **流控效果**选择**预热启动**。
- **单机QPS阈值**为60。
- **预热时间**为2s。

预热流控方式下，默认会从设置的QPS阈值的1/3开始慢慢往上增加至QPS设置值。本示例中，当入口的QPS超过20（即60÷3）时，会在预热的2s内缓慢增长至60。

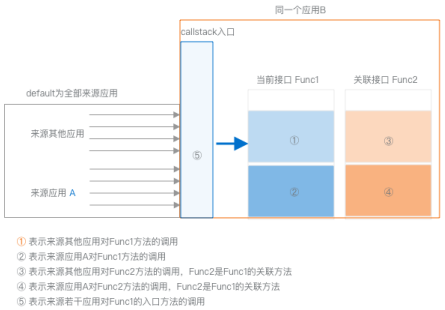
更多信息

新建流控规则页面参数说明如下：

- 单机模式：关闭集群流控的状态。



参数	描述
接口名称	待流控的资源名称。

参数	描述
来源应用	该规则针对的来源应用，默认来源应用设为 `default` ，代表不区分来源应用。 ○ 若是Dubbo服务，请填写对应调用方的Dubbo applicationName，注意调用方也需要接入Sentinel。 ○ 若是Web服务，请参见扩展接口。 ○ 若是通过注解、自定义埋点，请参见常用类及其方法，ContextUtil只在调用链入口生效，即首个埋点生效。 ○ 当前账号下已经接入了流量防护的其他应用，填写 `default` 即表示不特殊区分。 **注意** 来源数目不宜过多，200个以内内存使用量可控，过多则会影响性能。 
统计维度	选择资源调用关系进行流控。 ○ **当前接口**：直接控制来自**来源应用**中调用来源的访问流量，如果**来源应用**为 `default` 则不区分调用来源。通常应用于流量匀速通过的场景，详情请参见常用场景1。 ○ **关联接口**：控制当前资源的关联资源的流量。通常应用于资源争抢时，留足资源给优先级高接口的场景，详情请参见常用场景2。 ○ **链路入口**：控制该资源所在的调用链路的入口流量。选择**链路入口**后需要继续配置**入口资源**，即该调用链路入口的上下文名称。通常应用于预热启动避免大流量冲击的场景，详情请参见常用场景3。
单机QPS 阈值	触发对流控接口的统计维度对象的QPS阈值。

参数	描述
流控效果	选择流控方式来处理被拦截的流量。 ○ **快速失败**：达到阈值时，立即拦截请求。按照应用系统设置中的适配模块配置信息，进行内容返回。 ○ **预热启动**：需设置具体的预热时间。详情请参见Warm Up（冷启动）。 如果系统在此之前长期处于空闲的状态，当流量突然增大的时候，该方式会让处理请求的速率缓慢增加，经过设置的预热时间以后，到达系统处理请求速率的设定值。默认会从设置的QPS阈值的1/3开始慢慢往上增加至设置的QPS值，多余请求会按照快速失败处理。 ○ **排队等待**：请求匀速通过，允许排队等待，通常用于请求调用削峰填谷等场景。需设置具体的超时时间，达到超时时间后请求会快速失败。 详情请参见削峰填谷。
是否开启	打开开关表示启用该规则，关闭开关表示禁用该规则。开关修改之后会立即生效。

- 集群流控模式：开启集群流控的状态。

参数	描述
是否集群流控	开启集群流控，对集群内此资源的调用总量进行限制。
接口集群总QPS	该接口预估的集群最大QPS，表示最大流量，用于为Token Server自动分配提供参考，当流量超出该值的请求会退化到单机模式。
阈值模式	可选择设置**集群阈值**或**单机QPS阈值**。 ○ **集群阈值**：设置的阈值等同于整个集群的总阈值。 ○ **单机QPS阈值**：设置的阈值等同于单机能够承受的限额，Token Server会根据连接数来计算总的阈值。

参数	描述
失败退化策略	当出现连接失败、通信失败或Token Server不可用等情况时，流控规则是退化到单机限流的模式或是直接通过忽略失败情况。 ○ **退化到单机限流**：当出现失败的情况时，退化到根据的单机阈值来进行流控。需要设置**退化单机阈值**，代表单机的兜底阈值。 ○ **直接通过**：当出现失败的情况时，忽略失败情形，直接通过。

3.1.4.2. 配置隔离规则

隔离规则通过控制接口或依赖的并发线程数，来保证系统的稳定性。通常适用于应用内部或下游依赖出现不稳定的场景，例如慢SQL、下游应用响应时间变长等。本文介绍如何配置和管理隔离规则。

前提条件

- 开通企业版。相关内容，请参见微服务治理升级为企业版。
- MSE治理中心已接入微服务应用，相关内容，请参见：
  - ACK微服务应用接入MSE治理中心
  - ECS微服务应用接入MSE治理中心

背景信息

当强依赖的方法或接口出现不稳定的时候，可以通过配置并发线程数来限制不稳定的强依赖并发数，起到隔离异常的效果。若运行该请求的响应时间变长，会导致线程的并发数变大。当并发数超过阈值以后，AHAS将拒绝多余的请求，直到堆积的任务完成，并发线程数变少。达到将异常隔离，减小不稳定性的效果。

如何设定并发线程数阈值，可参见以下内容：

- 并发线程数 = 期望QPS*响应时间+冗余量。
- 例如预期的SQL执行时间为20毫秒，预期该请求每秒有20个，并发最大时候是6个，建议并发线程数按照以下逻辑设置：Max（20/1000*20，6）= 6 ，再加上冗余量2 ，则建议并发数阈值设置为8。
- 设置好后，当这个SQL发生死锁或者有性能问题，SQL运行特别慢成为慢SQL时，即使请求不断的进来，也仅仅会占用8个线程，不会因为持续进来的请求（请求也无法在短时间内退出），从而耗光进程的活跃线程。
- 当这个SQL恢复正常后，并发数会迅速减少。当并发数减少至低于预设的阈值时，系统就不会拒绝请求，应用的处理能力也快速的恢复。通过这样的方式，起到了根据响应时间自动调节的效果，隔离了不稳定的应用。

隔离规则配置通常用于强依赖隔离场景，详情请参见强依赖隔离。

功能入口

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 选择以下任意一种方法新建隔离规则：
   - 在左侧导航栏单击**应用概览**，然后单击页面下方目标接口**操作**列中的**隔离**。
   - 在左侧导航栏单击**接口详情**，在**接口详情**页面单击资源卡片右上角 ＋ 图标，在**新建规则**对话框中，选择**新增隔离规则**页签。

> **说明** 如果接口中已存在规则，单击 图标，在**隔离规则**页签中创建隔离规则。

- 在左侧导航栏单击**规则管理**，单击**隔离规则**页签，在**隔离规则**页签，单击**新增隔离规则**。

6. 在**新建隔离规则**对话框中配置规则信息：
   i. 在**选择防护场景**页面，修改接口名称，然后单击**下一步**。
   ii. 在**配置防护规则**页面，配置防护规则，然后单击**下一步**。

   > **说明** 若需对隔离防护规则进行编辑，则直接进入配置防护规则。

   iii. 在**配置限流行为**页面，新增和选择关联规则，然后单击**下一步 > 新增**。
   iv. 在**管理规则**页面，状态栏下单击**开启**。
   v. 在**温馨提示**页面，单击**确定**，开启已配置的防护规则。

> **说明** 参数说明具体详见更多信息。

常用场景1 保障自身资源充足

当运行该请求的响应时间变长，会导致线程的并发数变大。当并发数超过阈值以后，AHAS将拒绝多余的请求，直到堆积的任务完成，并发线程数变少。达到将异常隔离，减小不稳定性的效果。例如某个SQL执行时间为20毫秒，预期该请求每秒有20个。

在**新建隔离规则**对话框中配置以下规则信息：

- 填写**接口名称**和**来源应用**。
- **统计维度**选择**当前接口**。
- **并发数阈值**为10。

设置完成后，当这个SQL发生死锁或者存在性能问题时，该SQL运行变慢，成为慢SQL，此时即使请求不断进来，也仅仅会占用10个线程，不会因为持续进来的请求（请求也无法在短时间内退出），从而耗光进程的活跃线程。当这个SQL恢复正常后，并发数会迅速减少。当并发数减少至低于预设的阈值时，系统就不会拒绝请求，应用的处理能力也快速的恢复。通过这样的方式，起到了根据响应时间自动调节的效果，隔离了不稳定的应用。

常用场景2 有一定相关联性的接口

当关联接口被来源应用调用QPS超过阈值时，会对当前接口来源应用的请求进行限流，有一定的相关性的方法来配置规则。例如 `read_db` 和 `write_db` 这两个资源分别代表数据库读写，`write_db` 接口优先级更高。

为保证读写资源争抢时，`write_db` 的接口可以留足资源，可在**新建隔离规则**对话框中配置以下规则信息：

- **接口名称**为 `write_db` 。
- **统计维度**选择**关联接口**。

- **关联接口名**为`read_db`。
- **并发数阈值**为10。

这样在`read_db`接口被调用QPS超过10次后，会对`write_db`接口来自于来源应用的请求进行隔离限流，保证`write_db`的足够资源。

常用场景3 针对入口链路来配置隔离规则

从入口处将资源进行分别隔离，以保障更高优先级入口。当callstack入口被来源应用调用QPS数超过阈值时，会对当前接口来自于来源应用的请求进行隔离流控。

在**新建隔离规则**对话框中配置以下规则信息：

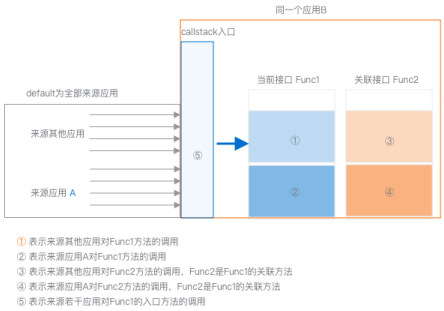
- 填写**接口名称**和**来源应用**。
- **统计维度**选择**链路入口**。
- **并发数阈值**设置为10。

当**callstack入口**的接口被调用超过10次，当前接口`user_test`会对来自于来源应用的请求进行隔离流控。

更多信息

新建隔离规则页面参数解释如下：

参数	描述
接口名称	待隔离的资源名称。

参数	描述
来源应用	该规则针对的来源应用，默认来源应用设为 `default`，表示不区分来源应用。  • 若是Dubbo服务，请填写对应调用方的Dubbo applicationName，注意调用方也需要接入Sentinel。 • 若是Web服务，请参见扩展接口。 • 若是通过注解、自定义埋点，请参见常用类及其方法，ContextUtil只在调用链入口生效，即首个埋点生效。 • 当前账号下已经接入了流量防护的其他应用，填写 `default` 即表示不特殊区分。 **注意** 来源数目不宜过多，200个以内内存使用量可控，过多则会影响性能。
统计维度	选择资源调用关系进行隔离流控。 • **当前接口**：直接控制来自**来源应用**中调用来源的访问流量，如果**来源应用**为 `default` 则不区分调用来源，通常应用于保障自身资源充足的场景，请参见常用场景1 保障自身资源充足。 • **关联接口**：控制当前资源的关联资源的流量。通常应用于资源争抢时，留足资源给优先级高接口的场景，请参见常用场景2 有一定相关联性的接口。 • **链路入口**：控制该资源所在的调用链路的入口流量。选择**链路入口**后需要继续配置**callstack入口**，即该调用链路入口的上下文名称。通常应用于接口有多入口资源的场景，请参见常用场景3 针对入口链路来配置隔离规则。
并发数阈值	资源的并发线程数（即该资源正在执行的线程数）阈值。

3.1.4.3. 配置熔断规则

MSE的熔断规则可以监控应用内部或者下游依赖的响应时间或异常比例，当达到指定的阈值时立即降低下游依赖的优先级。在指定的时间内，系统不会调用该不稳定的资源，避免应用受到影响，从而保障应用高可用性。当指定时间过后，再重新恢复对该资源的调用。

前提条件

- 开通企业版。相关内容，请参见微服务治理升级为企业版。
- MSE治理中心已接入微服务应用，相关内容，请参见：
  - ACK微服务应用接入MSE治理中心
  - ECS微服务应用接入MSE治理中心

背景信息

除了流量控制以外，对调用链路中不稳定的方法或者下游依赖进行熔断也是重要措施之一。由于调用关系的复杂性，如果调用链路中的某一环节出现了错误，会导致这个请求失败，甚至会放大不稳定性，导致整个链路无法正常服务。熔断功能会在调用链路中某个方法出现不稳定时（例如某方法出现Timeout或异常比例升高），对这个方法的调用进行限制，让请求快速失败，避免此错误影响整个链路。关于熔断的相关信息，请参见熔断结构图。

熔断规则配置通常用于弱依赖降级场景，更多信息，请参见弱依赖降级。

功能入口

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 选择以下任意一种方法进入熔断规则的设置页面：
   - 在左侧导航栏单击**应用概览**，然后单击页面下方目标接口**操作**列中的**熔断**。
   - 在左侧导航栏单击**接口详情**，在**接口详情**页面单击资源卡片右上角 ＋ 或 ⚙ 的图标，在**管理规则**对话框中单击目标方案的页签，然后单击**熔断规则**页签，单击**新增熔断规则**。
   - 在左侧导航栏单击**规则设置**，单击目标方案的页签，然后单击**熔断规则**页签，在页面右上角单击**新增熔断规则**。
6. 在**新增熔断规则**或**新增规则**的对话框中配置规则信息。

> **说明** 慢调用比例、统计窗口时长、最小请求数目等配置项需要Java SDK版本≥1.6.0或Java Agent版本≥1.7.5。

常用场景1：慢调用熔断示例

例如调用第三方服务，但响应时间太慢，会影响当前接口，所以对其进行熔断操作。

在**新增熔断规则**或**新增规则**对话框中配置以下示例规则信息。

参数	示例值	描述
接口名称	test	接口名称。
统计窗口时长	1	统计时长为1秒。
阈值类型	慢调用比例	选择以慢调用比例作为阈值。
慢调用RT	1000	超过1000 ms则判定为慢请求。
降级阈值	80%	触发熔断的慢调用比例阈值为80%。
熔断时长	10	熔断时长有10秒。

参数	示例值	描述
最小请求数目	10	触发熔断的最小请求数目为10。
熔断恢复策略	单次探测恢复	经过熔断时长后，熔断器会对接下来的一个请求进行探测，若该请求符合预期（不为慢调用或没有异常），则结束熔断；否则重新回到熔断阶段。

规则开启后，在统计时长1秒内，当请求数目大于10，并且慢调用的比例大于80%的时候，则在接下来10秒的熔断时长内，请求都会快速失败。经过10秒后熔断器会进入探测恢复状态，若接下来的一个请求响应时间小于设置的1000 ms则结束熔断，若大于1000 ms则会再次被熔断。

常用场景2：异常熔断示例

例如第三方内容展示时，系统会出现异常，当异常比例较高时，可以对其进行熔断操作，以保证更好的用户体验。

在**新增熔断规则**或**新增规则**对话框中配置以下示例规则信息。

参数	示例值	描述
接口名称	test	接口名称。
统计窗口时长	1	统计时长为1秒。
阈值类型	异常比例	选择以异常比例作为阈值。
降级阈值	80%	触发熔断的异常比例阈值为80%。
熔断时长	10	熔断时长有10秒。
最小请求数目	10	触发熔断的最小请求数目为10。

参数	示例值	描述
熔断恢复策略	单次探测恢复	经过熔断时长后，熔断器会对接下来的一个请求进行探测，若该请求符合预期（不为慢调用或没有异常），则结束熔断；否则重新回到熔断阶段。

规则开启后，在统计时长1秒内，当请求数目大于10，并且异常的比例大于80%的时候，则在接下来10秒的熔断时长内，请求都会快速失败。经过10秒后熔断器会进入探测恢复状态，若接下来的一个请求没有异常则结束熔断，否则会再次熔断。

更多信息

新增熔断规则或**新增规则**对话框的参数说明如下。

参数	描述
接口名称	适用该规则的应用资源。
统计窗口时长	统计的时间窗口长度，取值范围为1秒~120分钟。
最小请求数目	触发熔断的最小请求数目，若当前统计窗口内的请求数小于此值，即使达到熔断条件规则也不会触发。

参数	描述
阈值类型	选择以**慢调用比例**或**异常比例**作为阈值。 • 选择以**慢调用比例**作为阈值，需要设置允许的**慢调用RT**（即最大的响应时间），请求的响应时间大于该值则统计为慢调用。 在**降级阈值**中设置触发熔断的慢调用比例。规则开启后，在单位统计时长内请求数目大于设置的**最小请求数目**，并且慢调用的比例大于阈值，则接下来的**熔断时长**内请求会自动被熔断。经过熔断时长后熔断器会进入探测恢复状态，若接下来的一个请求响应时间小于设置的**慢调用RT**则结束熔断，若大于设置的**慢调用RT**则会再次被熔断。 • 选择以**异常比例**作为阈值，需要在**降级阈值**中设置触发熔断的异常比例。 规则开启后，在单位统计时长内业务异常数目大于设置的**最小请求数目**，并且异常的比例大于阈值，则接下来的**熔断时长**内请求会自动被熔断。
熔断时长	即熔断触发后持续的时间。资源进入熔断状态后，在配置的熔断时长内，请求都会快速失败。

熔断恢复策略	熔断器进入恢复阶段（半开启状态）的恢复策略。
	• **单次探测恢复**：经过熔断时长后，熔断器会对接下来的一个请求进行探测，若该请求符合预期（不为慢调用或没有异常），则结束熔断；否则重新回到熔断阶段。
	• **渐进式恢复**：需要设置**恢复阶段数**和**每步最小通过数目**。
	  ○ 经过熔断时长后，熔断器按照设定的恢复阶段数进行渐进式恢复，若该阶段内请求达到一定量即每步最小通过数目，则触发检查。检查的请求若都未超过阈值，则逐步提高允许通过的请求比例，直到请求完全恢复；若某一步的指标超出阈值，则重新回到熔断阶段。
	  ○ 请求比例T=100/恢复阶段数N，则第一阶段请求比例为T，第二阶段为2T……直到100%。
	  ○ 例如恢复阶段数为3，每步最小通过数目为5，则三个阶段分别按照33%、67%和100%的比例放入请求，当每阶段的请求数目大于等于5时进行检查，若请求的指标未超阈值则进入下一恢复阶段，直至完全恢复。
	说明 渐进式恢复策略功能需要Java SDK版本≥1.6.2。

3.1.4.4. 配置主动降级规则

MSE主动降级规则可以指定对某些接口进行降级，被降级的接口会触发自定义的降级行为（如返回指定内容）而不会执行原有的逻辑。本文介绍如何新增主动降级规则。

前提条件

- 开通企业版。相关内容，请参见微服务治理升级为企业版。
- MSE治理中心已接入微服务应用，相关内容，请参见：
  - ACK微服务应用接入MSE治理中心
  - ECS微服务应用接入MSE治理中心

注意事项

主动降级规则支持的场景说明如下：

- 主动降级规则仅支持MSE Sentinel Java SDK或Agent 1.8.4及以上版本。
- 降级规则中的行为配置目前仅对MSE自带的Web埋点生效（Servlet、Spring Web、Spring Cloud Gateway适配），其它埋点类型仍会按照原有的Fallback逻辑进行处理。更多信息，请参见配置触发规则后的逻辑。
- 若有在代码中注册Block Handler的方式自定义Fallback逻辑，则控制台配置的主动降级规则行为配置不生效。
- 主动降级规则暂不支持其他多语言SDK方式接入。

新增主动降级规则

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 选择以下任意一种方法进入设置主动降级规则的对话框。
   - 在左侧导航栏单击**接口详情**，在**接口详情**页面单击资源卡片右上角 ＋ 或 ⚙ 的图标，在**管理规则**对话框中单击目标方案的页签，然后单击**降级规则**页签，在页面右上角单击**新增主动降级规则**。
   - 在左侧导航栏单击**规则设置**，单击目标方案的页签，然后单击**降级规则**页签，在页面右上角单击**新增主动降级规则**。
6. 在设置主动降级规则的对话框，完成以下配置，然后单击**新建**。

参数	描述	示例值
接口名称	适用该规则的资源名称，需要与监控页面上的资源名（埋点传入的资源名）保持一致。	/hello
降级行为选择	表示开启该降级规则后，该接口调用的行为。 **默认行为**对应**应用设置**页面**基础设置**中的**模块适配设置**Web的配置。具体操作，请参见设置适配模块。 **说明** 目前降级行为只对MSE自带的Web埋点生效。	默认行为

若需要新增行为，单击**新增行为**，完成以下配置，然后单击**新增**。更多信息，请参见配置Web行为。

参数	描述	示例值
行为名称	该行为的名称。长度不超过128个字符，同个应用内名称不能重复。	测试行为

参数	描述	示例值
针对的资源类型	目前仅支持Web类型。	Web
Web限流处理策略	定义Web接口访问触发某种规则后的行为表现。目前支持以下两种策略： ○ 自定义返回：需设置HTTP返回状态码、返回内容的格式和返回的内容。表示Web接口访问触发规则后返回自定义的内容。 ○ 跳转到指定页面：需设置指定跳转的URL。表示Web接口访问触发规则后系统会跳转指定的页面URL。	自定义返回
HTTP返回状态码	默认429。当Web限流处理策略为自定义返回时，需要填写。	429
返回content-type	设置返回内容的格式为普通文本（TEXT）或JSON。当Web限流处理策略为自定义返回时，需要填写。	JSON字符串
HTTP返回文本	输入当Web接口访问触发规则后返回的内容。当Web限流处理策略为自定义返回时，需要填写。	{"message": "blocked oops"}
跳转地址	输入当Web接口访问触发规则后系统会跳转的页面URL。当Web限流处理策略为跳转到指定页面时，需要填写。 **说明** 跳转的本质是返回302状态码。对于后端服务直接渲染返回的页面，跳转是有效的；对于前端通过AJAX请求到后端服务后，再解析后端返回到前端展示的页面，跳转无效。	http://MSE.console.aliyun.com

设置完成的主动降级规则会展示在**规则设置**页面主动降级规则列表中。

3.1.4.5. 自适应流控

系统支持自适应流控或手动设置系统规则，自适应流控是根据系统的CPU使用率自动动态地调整应用程序的入口流；系统规则是从整体维度手动设置规则，对应用入口流量进行控制。目的都是为了让系统的入口流量和系统的负载达到一个平衡，保证系统在最大吞吐量状态下稳定运行。

前提条件

- 开通企业版。相关内容，请参见微服务治理升级为企业版。
- MSE治理中心已接入微服务应用，相关内容，请参见：
  - ACK微服务应用接入MSE治理中心

- ECS微服务应用接入MSE治理中心

背景信息

系统支持开启自适应流控或手动设置系统规则：

- 自适应流控：当开启自适应流控，系统会根据CPU使用率动态调节应用的入口流量，在尽可能保证吞吐量的同时保证高负载下系统稳定。
- 系统规则：当关闭自适应流控，需要您手动设置系统规则。系统规则从整体维度对应用入口流量进行控制，结合应用的负载、CPU使用率、总体平均RT、入口QPS和并发线程数等几个维度的监控指标，让系统的入口流量和系统的负载达到一个平衡，保证系统在最大吞吐量状态下稳定运行。

系统保护规则是应用整体维度的，而不是资源维度的，并且仅对入口流量生效。入口流量指的是进入应用的流量，例如Web服务或Dubbo服务端接收的请求。通过自定义埋点接入和通过注解接入的应用，入口流量为 `EntryType.IN` 的逻辑被调用时产生的流量。配置系统规则的对象必须是入口流量，且默认接口为Dubbo或HSF才生效，如默认埋点的Dubbo、Servelet等，或者通过修改 `EntryType.IN` 来让系统规则对该方法生效。

系统规则支持以下的模式：

- Load（仅对Linux、Unix-like机器生效）：当系统Load 1超过阈值且系统当前的并发线程数超过系统容量时才会触发系统保护。
- CPU使用率：当系统CPU使用率超过阈值（0.0~1.0）即触发系统保护。
- RT：当单台机器上所有入口流量的平均RT达到阈值即触发系统保护。
- 线程数：当单台机器上所有入口流量的并发线程数达到阈值即触发系统保护。
- 入口QPS：当单台机器上所有入口流量的QPS达到阈值即触发系统保护。

对于一个应用来说，每种相同的系统保护规则最多只能存在一条，即一个应用最多配置五条系统保护规则。配置系统规则的原理请参见系统防护。

新建系统规则

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 在左侧导航栏单击**规则设置**，然后单击**自适应流控**页签。
6. 在页面左上角，关闭**自适应流控**，在对话框中单击**确定关闭**。
7. 在**自适应流控**页签右上角单击**新建系统保护规则**。
8. 在**新建系统保护规则**对话框中，配置规则信息。

参数	描述	使用场景说明
CPU使用率	当系统CPU使用率超过阈值即触发系统保护，阈值设置范围为0.0~1.0（代表0%~100%）。	适用于设置基础资源水位的场景，比如需要保证一定的冗余水位。但系统水位不宜过高，需要留部分水位。
Load	当系统的Load1超过阈值，且系统当前的并发线程数超过系统容量时才会触发系统保护。系统容量由系统的maxQps * minRt计算得出。	适用于设置基础资源水位的场景，比如需要保证一定的冗余水位。但系统水位不宜过高，需要留部分水位。
线程数	当单台机器上所有入口流量的并发线程数达到阈值即触发系统保护。	适用于设置基础资源水位的场景，比如需要保证一定的冗余水位。但系统水位不宜过高，需要留部分水位。

参数	描述	使用场景说明
入口平均RT	当单台机器上所有入口流量的平均RT达到阈值即触发系统保护，单位是毫秒。	适用于衡量入口请求的场景。
入口总QPS	当单台机器上所有入口流量的QPS达到阈值即触发系统保护。	适用于衡量入口请求的场景。

9. 单击**新增**。

3.1.4.6. 配置热点规则

为应用配置热点规则后，MSE将分析统计参数，即资源调用过程中的调用次数较高的参数，并根据配置的热点规则对包含热点参数的资源调用进行限流，保护系统稳定性。本文介绍如何为应用配置热点规则。

背景信息

热点即经常被访问的数据。例如在以下场景中需要统计某个热点数据中访问频次最高的Top数据，并对其访问进行限制。

- 针对一段时间内最频繁购买的商品ID进行限制，防止击穿缓存而导致大量请求到数据库的情形。
- 针对一段时间内频繁访问的用户ID进行限制，防止恶意刷单。

MSE利用LRU（Least Recently Used）策略统计最近最常访问的热点参数，结合令牌桶算法来进行参数级别的流控。

功能入口

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 在左侧导航栏单击**规则设置**，然后单击**热点规则**页签。
6. 在**热点规则**页签右上角单击**新增热点限流规则**。
7. 在**新增热点限流规则**对话框中，配置规则信息。
8. 单击**新增**。
9. （可选）（可选）若需对某些特殊的热点参数值单独配置阈值，则在规则列表中单击目标规则**操作**列的**添加例外项**，然后在**热点规则例外项**对话框中单击**添加**，并填写单独配置的参数信息，然后单击**确定**。

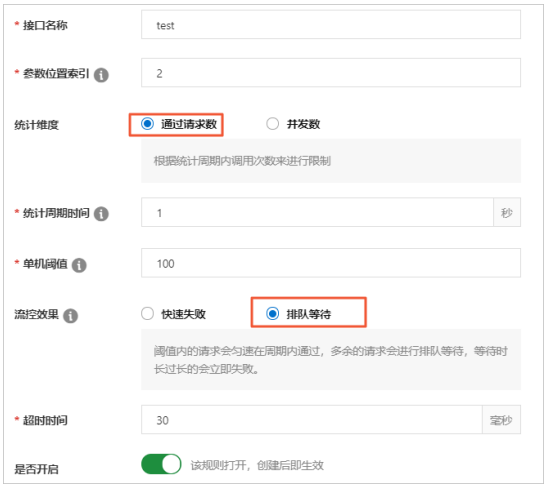
> **说明** 热点规则例外项仅支持基本类型和字符串类型。由于一条规则只对应一个参数索引位置，因此例外项类型需要保持统一。例如在资源R的参数索引0处配置一条热点规则，阈值为5QPS。若在这条规则新增两条例外项：参数Sentinel阈值为100，参数MSE阈值为200。则实际生效的时候，其它热点参数会限制每秒钟访问不超过5次，而Sentinel和MSE这两个参数值作为例外项，会按照对应的例外项阈值生效。

常用场景1 秒杀场景

秒杀等抢购商品的时候，由于流量较大会导致系统响应不及时，甚至系统崩溃。为保证系统的稳定性，可配置热点规则，超过一定量的阈值后，系统会让购买热点商品的流量排队等待。

例如购买同一商品，1s内调用超过100次请求后，则其余请求进行等待。在**新建热点规则**对话框中配置以下规则信息。

- 填写**接口名称**。
- **统计维度**选择**通过请求数**。
- **统计周期时间**设置为1s，**单机阈值**设置为100。
- **流控效果**选择**排队等待**。
- **超时时间**设置为30 ms。



1s内调用此接口超过100次，多余的请求要进行排队等待，等待时长超过30 ms的请求就会立即失败。

常用场景2 调用请求频繁，占用较多系统资源

例如秒杀的时候还需要修改下单地址，当调用修改请求较多的时候，会占用了写数据库较多资源，则可以对其进行热点快速失败的处理，稍后再修改。在**新建热点规则**对话框中配置以下规则信息。

- 填写**接口名称**。
- **统计维度**选择**并发数**。
- **统计周期时间**设置为1s，**单机阈值**设置为100。
- **流控效果**选择**快速失败**。

表示1s内只能最多处理100条修改请求，其余超出的请求都会快速失败。

更多信息

新建热点规则页面参数说明如下：

参数	描述
接口名称	适用该规则的资源名称，与埋点传入的资源名保持一致。

参数	描述
参数位置索引	埋点传入参数的索引位置。对应 `SphU.entry(xxx,args)` 中的参数索引位置。例如 `SphU.entry(resourceName,EntryType.IN,1,paramA,paramB)` 埋点中，`paramA` 的参数索引是0，`paramB` 的参数索引是1。
统计维度	可选择通过请求数或并发线程数。 • **通过请求数**：限制一段时间内的调用次数。 • **并发数**：限制该资源调用的并发数。
统计周期时间	统计窗口时间长度（单位为秒）。例如统计窗口时长为10s，QPS阈值为5代表限制10s内每个热点参数访问不超过5次。
单机阈值	是作用于每个热点参数的阈值。
流控效果	当**统计维度**为**通过请求数**时，可以选择流控效果来处理被拦截的流量。 • **快速失败**：达到阈值时，立即拦截请求。该模式下可以额外设置一个**缓冲请求数**，即针对突发请求额外允许的请求数目。 • **排队等待**：请求匀速通过，允许排队等待，通常用于消息队列削峰填谷等场景。需设置具体的超时时间，排队时会计算预计的排队时长，若超过最大超时时间则请求会直接被拒绝。例如，**单机阈值**配置为5，则代表请求每200 ms才能通过一个，多出的请求将排队等待通过。**超时时间**配置1000 ms，则当前排队请求超过5个（>1000 ms）时，新到来的请求将会直接被拒绝。

3.1.4.7. 查看热点监控详情

热点指的是频繁被访问的数据。通过热点监控统计某个热点数据中访问频次最高的数据，并对这些Top数据的访问进行相关限制。本文介绍配置热点规则后，如何查看热点监控详情。

前提条件

配置热点规则。

操作步骤

在热点规则配置完成后，即可查看热点监控详情。

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 在左侧导航栏单击**热点详情**。
   在左侧**热点规则**页签中可查看最近10秒被触发的热点规则信息。

6. 单击被触发的热点规则，在左侧的**热点概览**页签可查看该热点的概览信息。

   热点概览字段通过词云图的方式实时展示，热点字段大小根据热点被阻塞的流量设置，被阻塞流量越多，热点字段越大。热点概览生命周期与热点规则相同。

7. 单击**热点详情**页签，可查看该热点的详细信息。

   热点详情中将会保留最近5分钟的热点字段信息。

8. 单击**热点概览**或者**热点详情**页签中的任一热点字段，在**节点详情**页签中可查看该热点分节点限流的情况。

   如下图所示，单击**热点概览**页签中的热点字段**foo9263**，即可查看该几点分节点的限流信息。

3.1.5. 配置行为

背景信息

1.

3.1.5.1. 配置Web行为

Web行为可以在Web类型埋点资源触发了某种规则后，返回对应的自定义处理行为，例如，某个Web接口触发流控规则后返回Blocked by Sentinel的提示文本。本文介绍如何新增、修改、删除和关联Web行为。

背景信息

配置行为主要是配置Fallback行为。Fallback行为定义某个埋点资源触发了某种规则（如流控、熔断、降级）后的处理行为。目前Fallback行为仅支持Web和RPC两种资源类型。本文介绍Fallback行为中Web行为的配置方法。

前提条件

- 开通企业版。相关内容，请参见微服务治理升级为企业版。
- MSE治理中心已接入微服务应用，相关内容，请参见：
  - ACK微服务应用接入MSE治理中心
  - ECS微服务应用接入MSE治理中心

新增行为

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 在左侧导航栏选择**应用管理**，然后单击**行为管理**页签。
6. 单击**新增行为**，在**新增行为**对话框中完成以下配置，然后单击**新建**。

参数	描述	示例值
行为名称	该行为的名称。长度不超过128个字符，同个应用内名称不能重复。	测试行为
针对的资源类型	包括Web和RPC两种类型，此处选择Web类型。	Web
Web限流处理策略	定义Web接口访问触发某种规则后的行为表现。目前支持以下两种策略： - 自定义返回：需设置HTTP返回状态码、返回内容的格式和返回的内容。表示Web接口访问触发规则后返回自定义的内容。 - 跳转到指定页面：需设置指定跳转的URL。表示Web接口访问触发规则后系统会跳转指定的页面URL。	自定义返回
HTTP返回状态码	默认为429。当Web限流处理策略为自定义返回时，需要填写。	429
返回content-type	设置返回内容的格式为普通文本（TEXT）或JSON。当Web限流处理策略为自定义返回时，需要填写。	JSON字符串
HTTP返回文本	输入当Web接口访问触发规则后返回的内容。当Web限流处理策略为自定义返回时，需要填写。	{"message": "blocked oops"}

参数	描述	示例值
跳转地址	输入当Web接口访问触发规则后系统会跳转的页面URL。当Web限流处理策略为跳转到指定页面时，需要填写。 **说明** 跳转的本质是返回302状态码。对于后端服务直接渲染返回的页面，跳转是有效的；对于前端通过AJAX请求到后端服务后，再解析后端返回到前端展示的页面，跳转无效。	http://MSE.console.aliyun.com

新增的行为会显示在**应用管理**页面的**行为管理**页签中。

修改或删除行为

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 在左侧导航栏选择**应用管理**，然后单击**行为管理**页签。
   在行为列表页，您可以查看各个行为的具体描述，修改或删除行为。

关联行为

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 选择以下任意一种方式进入**新增流控防护规则**页面：
   - 在左侧导航栏单击**接口详情**后，在**WEB服务**页签下单击目标接口卡片右上角的图标进入**管理规则**页面，然后单击**新增流控规则**。
   - 在左侧导航栏单击**规则管理**，然后在**流控规则**页签下单击**新增流控规则**。
5. 完成选择防护场景和配置防护规则后，在**配置限流行为**区域，完成下列设置。
   i. 选择**接口类型**为**Web**。
   ii. 在**关联行为**的下拉列表中选择目标行为进行关联，或单击**新增行为**来创建新的行为进行关联。

   说明
   - 如果您不需要自定义限流后的Fallback行为，则选择默认行为即可，默认接口类型为空。
   - 新增规则时，若当前接口已有绑定行为，后续若绑定新的行为则会覆盖接口下已有的行为。
   - 选择Fallback类型时，若不选择接口类型，则绑定默认行为；选择接口类型并绑定相应类型行为后，不可修改。
6. 单击**下一步**后，单击**新增**。

3.1.5.2. 配置RPC行为

RPC行为可以在RPC类型埋点资源触发了某种规则后，返回相应的自定义的处理行为，例如某个RPC接口触发流控规则后返回自定义的接口返回值。本文介绍如何新增、修改、删除和关联RPC行为。

前提条件

- RPC Fallback行为仅支持MSE Sentinel Java SDK 1.9.5及以上版本以及Java Agent 1.10.1及以上版本。
- RPC行为在MSE SDK 1.9.5~1.10.3版本仅支持在Dubbo类型接口下生效，在MSE SDK 1.10.4及以上版本支持Dubbo、HSF、SOFARPC类型接口。
- RPC行为暂不支持其他多语言SDK方式接入。

背景信息

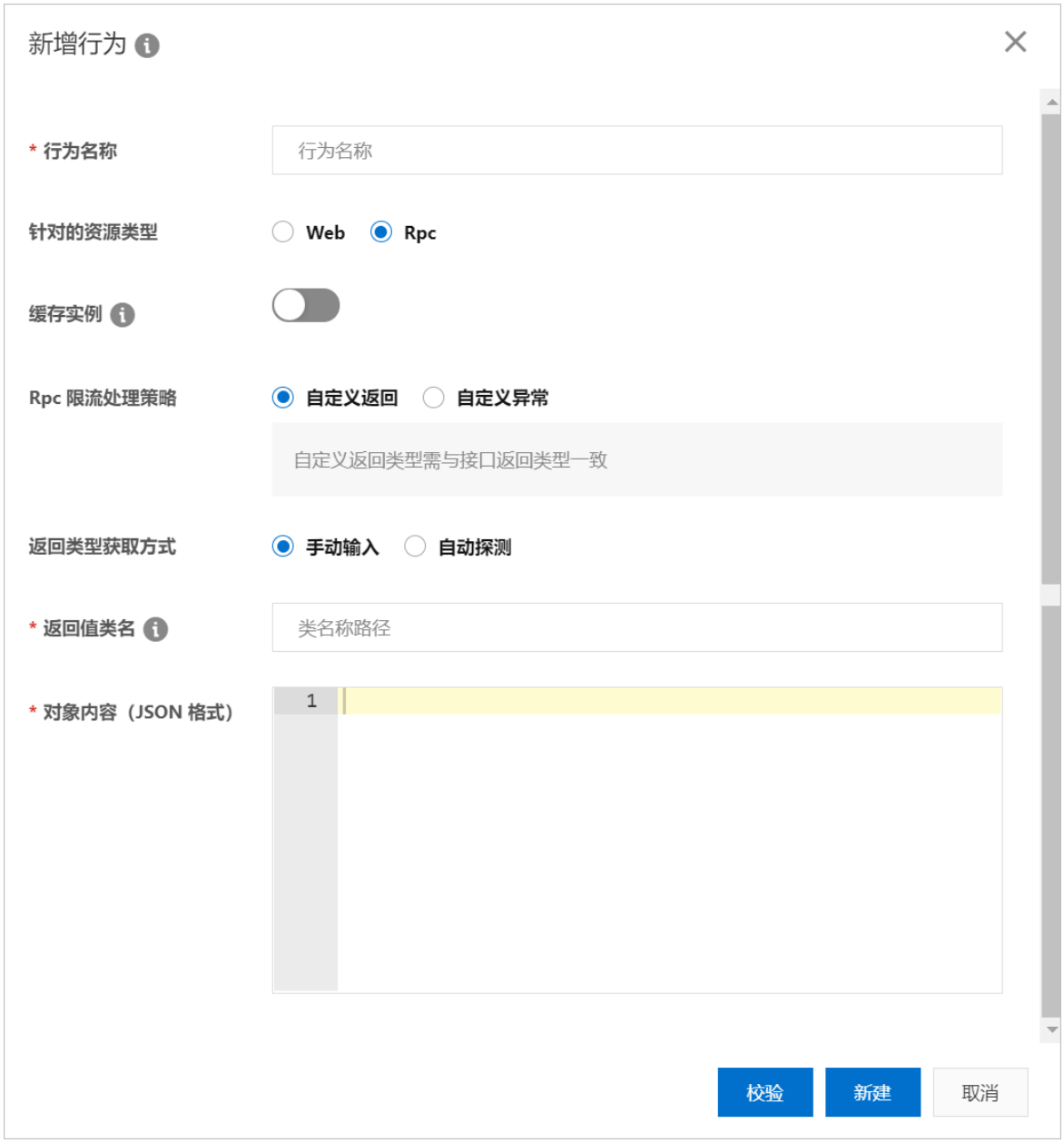
配置行为主要是配置Fallback行为。Fallback行为定义某个埋点资源触发了某种规则（如流控、熔断、降级）后的处理行为。目前Fallback行为仅支持Web和RPC两种资源类型。本文介绍Fallback行为中RPC行为的配置方法。

在MSE SDK 1.10.4及以上版本，RPC行为支持返回值包含未确定类型的泛型、自动探测接口方法返回类型。

> 注意
>
> - 返回值类名中不允许存在接口、抽象类，请使用相应具体继承类型。
> - 返回值若包含未确定类型的泛型，需要在类名中确定其相应的具化类型。
> - 自动探测模式仅支持Dubbo、HSF类型接口。

新增行为

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 在左侧导航栏选择**应用管理**，然后单击**行为管理**页签。
5. 单击**新增行为**，在**新增行为**对话框中完成以下配置。



参数	描述	示例值
行为名称	该行为的名称。长度不超过128个字符，同个应用内名称不能重复。	测试行为
针对的资源类型	包括Web和RPC两种类型，此处选择RPC类型。	RPC
缓存实例	是否缓存返回值。若开启，则会缓存生成的Fallback对象，在该行为触发时，均复用同一对象。	开启

参数	描述	示例值
RPC限流处理策略	定义RPC接口访问触发某种规则后的行为表现。目前支持以下两种策略： ○ **自定义返回**：自定义返回结果。需设置返回类型和返回的内容，表示RPC接口访问触发规则后返回自定义的实体类。 ○ **自定义异常**：抛出自定义异常。需设置异常的类名和异常文本，表示RPC接口访问触发规则后系统会返回指定的异常信息。	自定义返回
返回类型获取方式	填充返回值类名的方式，分别为： ○ **手动输入**：选择手动输入模式，需要填写自定义返回值的全限定类名（返回值类名），然后单击**校验**进行有效性校验，校验通过后可以创建行为。 ○ **自动探测**：选择自动探测模式，需要选择创建RPC行为的接口方法（行为关联方法），控制台会根据选择的行为关联方法，自动填充方法返回值类名，然后填写完成对象内容（JSON格式）后可以创建行为。 MSE SDK 1.10.4及以上版本支持使用自动探测功能。 **注意** 如果当前接口方法返回值包含未具化的参数类型，需要手动填写泛型的具体类型，通过校验后即可以在控制台创建行为。	手动输入
行为关联方法	当返回类型获取方式为**自动探测**时，需要选择当前RPC行为关联的接口方法，方法返回值类型即为当前自定义返回值的类型。	com.alibaba.demo.RpcResult
返回值类名	选择RPC限流处理策略为**自定义返回**时，需填写的类名称路径。 **说明** **自定义返回**目前不支持对象类型中包含未确定类型的泛型，如Map<K, V>、List<T>等。	com.alibaba.demo.OrderService:getOrder(long)
对象内容（JSON格式）	选择RPC限流处理策略为**自定义返回**时，填写当RPC接口访问触发规则时返回结果的对象内容（JSON格式）。	{"id": "123", "name": "test"}

参数	描述	示例值
异常类名	选择RPC限流处理策略为**自定义异常**时，需填写的异常类名称路径。	java.lang.RuntimeException
异常信息文本	选择RPC限流处理策略为**自定义异常**时，填写RPC接口访问触发规则后抛出自定义异常的文本信息。 **说明** 当前异常仅支持包含String类型构建函数的异常类。	"Operation failed"

6. 单击**校验**。

   说明 RPC行为的数据结构需要与客户端相应接口的数据结构保持一致。因此，新增RPC行为时，需完成数据类型的校验才能完成创建。校验失败时可以根据相应异常信息提示进行修改。

7. 单击**新建**。
   新增的行为会显示在**应用管理**页面的**行为管理**页签中。

修改或删除行为

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 在左侧导航栏选择**应用管理**，然后单击**行为管理**页签。
   在行为列表页，您可以查看各个行为的具体描述，修改或删除行为。

关联行为

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 选择以下任意一种方式进入**新增流控防护规则**页面：
   - 在左侧导航栏单击**接口详情**后，在**RPC服务**页签下单击目标接口卡片右上角的图标进入**管理规则**页面，然后单击**新增流控规则**。
   - 在左侧导航栏单击**规则管理**，然后在**流控规则**页签下单击**新增流控规则**。
5. 完成选择防护场景和配置防护规则后，在**配置限流行为**区域，完成下列设置。
   i. 选择**接口类型**为**Rpc**。
   ii. 在**关联行为**的下拉列表中选择目标行为进行关联，或单击**新增行为**来创建新的行为进行关联。

> 说明
> - 如果您不需要自定义限流后的Fallback行为，则选择默认行为即可，默认接口类型为空。
> - 新增规则时，若当前接口已有绑定行为，后续若绑定新的行为则会覆盖接口下已有的行为。
> - 选择Fallback类型时，若不选择接口类型，则绑定默认行为；选择接口类型并绑定相应类型行为后，不可修改。

6. 单击**下一步**后，单击**新增**。

3.1.6. 场景防护

3.1.6.1. Web场景防护

MSE的Web场景防护功能面向提供Web服务的应用，针对访问请求中的一些参数项进行精细化的流量控制。对于使用了主流Web框架（Servlet容器、Spring Web、Spring Boot）的应用，MSE实现了API粒度的请求参数解析，通过配置Web流控规则，可以对请求中IP、Host、Header、URL Param等参数维度的资源调用进行流量控制，保护业务与系统的稳定性。本文介绍如何为应用配置Web场景防护。

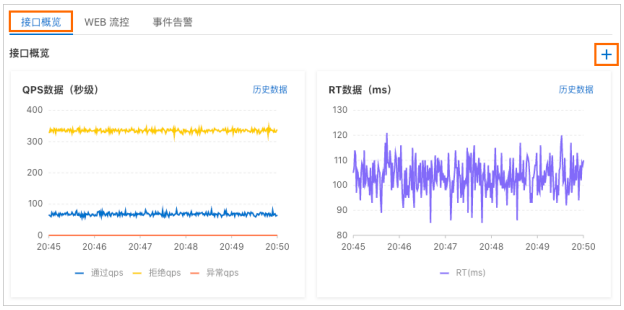
背景信息

在提供Web服务的场景下，除了API维度的限流降级防护，针对访问请求来源IP、访问请求Param参数等资源调用的限流防护是各种业务场景下能更好保证业务应用正常运行的手段。在一些大流量的Web业务场景下，可能不单是对当前接口进行限制，而需要针对当前访问频次最高的来源IP或访问频次最高的商品ID，有针对性地对其访问进行限制，比如：

- 对一段时间内最频繁购买的商品ID进行限制，以防击穿缓存而导致大量请求到数据库的情况。
- 对一段时间内频繁大量访问的来源IP进行限制，防止利用虚假信息恶意刷单。

操作步骤

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 在左侧导航栏选择**场景防护 > WEB场景**。
5. 选择以下任意一种方式进入Web防护的设置页面：
   - 在**接口概览**页签右侧单击+图标。



   - 单击**WEB流控**页签，然后单击**新增Web防护规则**。
6. 在**新增Web防护规则**对话框中完成规则配置。

参数	说明
选择防护场景	
API	选择需配置规则的接口API。
防护类型	默认为**WEB防护**。
配置防护规则	
参数属性	针对所选API的参数属性进行流量控制： ○ **Client IP**：请求端的IP地址。 **说明** 若请求经过代理，会优先尝试从X-Forwarded-For请求头中获取IP信息，如果其IP信息存在，将会作为实际请求端IP地址。 ○ **Remote Host**：请求端的Host Header。 ○ **Header**：根据指定的HTTP Header进行解析，要填写某个具体的Header Key，则该规则针对这个Header Key下面的热点值分别进行限制。选择**Header**后，可选择配置请求属性值的匹配策略，只有匹配该模式的请求属性值会纳入统计和流控。 ○ **URL参数**：根据指定的HTTP请求参数进行解析，需要填写对应的参数名称。选择**URL参数**后，可选择配置请求属性值的匹配策略，只有匹配该模式的请求属性值会纳入统计和流控。
（可选）匹配模式和匹配串	若选择参数属性为**Header**或**URL参数**，可打开**属性值匹配**开关，并设置**匹配模式**和**匹配串**。 **匹配模式**： ○ **精确**：严格按照给定的匹配串来匹配值。 ○ **子串**：若请求属性值包含该子串则匹配成功，比如若子串设置匹配ab，则aba和cabc都可以匹配，而cba则不能匹配。 ○ **正则**：按给定的正则表达式匹配串进行匹配。
阈值类型	默认为**请求数**。
阈值	触发对流控接口的统计维度对象的QPS阈值。设置时，需选择统计时间间隔，支持秒、分钟、小时和天4种维度。 例如，若阈值填写为*10*，统计间隔选择**分**，则表示每分钟对应的请求数目不超过10个。

参数		说明
高级选项	流控方式	○ **快速失败**：当阈值类型为QPS时，被拦截的流量将快速失败，即达到阈值时，立即拦截请求。 **说明** 被拦截拒绝掉的请求，将返回行为管理中配置的自定义信息，若未配置会返回默认行为，即429错误码加上默认文本信息。 ○ **匀速排队**：当阈值类型为QPS时，被拦截的请求将匀速通过，允许排队等待。需设置具体的超时时间，预计达到超时时间的请求会立即失败，而不会排队。例如，QPS配置为10，则代表请求每100 ms才能通过一个，多出的请求将排队等待通过。超时时间代表最大排队时间，超出最大排队时间的请求将会直接被拒绝。 **说明** 匀速排队时，QPS不要超过1000（请求间隔1 ms）。
	Burst size	当**流控方式**选择为**快速失败**时，可以额外设置一个Burst Size，即针对突发请求额外允许的请求数目。
	超时时间	当**流控方式**选择为**匀速排队**时，需设置具体的超时时间，单位为ms。例如，QPS配置为5，则代表请求每200 ms才能通过一个，多出的请求将排队等待通过。超时时间代表最大排队时间，超出最大排队时间的请求将会直接被拒绝。
是否开启		○ 开启：Web防护规则创建后即生效。 ○ 关闭：Web防护规则创建后不生效。
配置限流行为		
接口类型		默认为**Web**。
关联行为		○ **默认行为**：默认为此选项。 **说明** ■ 如您无需自定义限流后处理行为，选择默认行为即可。 ■ **默认行为**对应**应用基础设置 > Web fallback 行为**里面配置的全局行为。更多信息，请参见应用基础设置.。 ■ 默认行为为返回429错误码加上默认的文本信息。 ○ **新增行为**：新增自定义限流后处理行为，创建完成后您可在**关联行为**选择框中下拉选择。更多详细信息，请参见配置Web行为。 **新增Web行为参数说明** ❯ ○ **取消行为**：取消接口与关联行为的关联。

7. 操作完成后，单击**新增**。
   您可在**WEB流控**页签查看该条防护规则。
8. 在**WEB流控**页签，选择目标Web防护规则，单击**批量开启**。
   规则开启后，当前资源会按照配置的Web流控规则进行流量控制。

Web防护规则常用场景

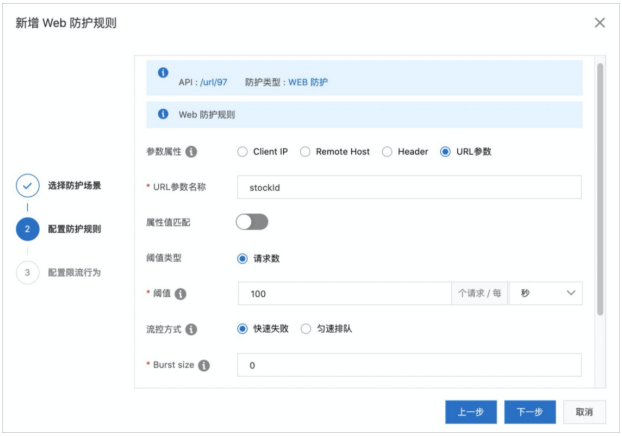
在秒杀等抢购商品的场景下，由于流量较大，可能会导致系统响应不及时甚至崩溃。为保证系统稳定，可配置热点规则，超过一定阈值后，系统会让购买热点商品的流量排队等待。

例如购买同一商品，希望1秒内同个商品超过100次请求后，对多余的请求进行拒绝，可在**新建Web防护规则**对话框中配置以下规则信息，表示1秒内，针对热点商品ID进行下单的请求，每个单独商品ID每秒最多只能允许100个请求，其余超出的当前商品下单请求都会被拒绝，返回自定义的信息。

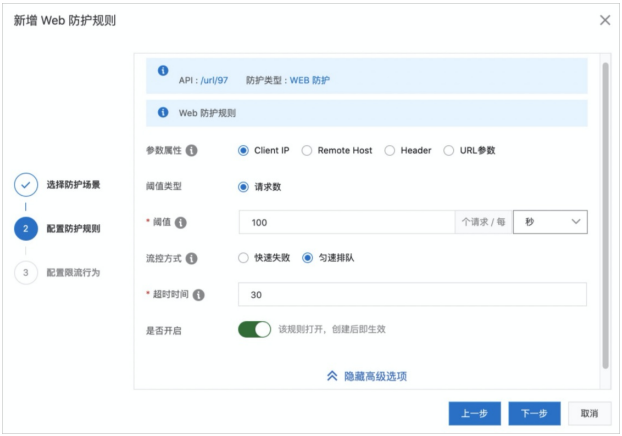
- **参数属性**输入*URL参数*。

> **说明** 在参数属性中，选择当前热点商品ID所对应的参数字段，例如，假设URL参数中存在一个stockId字段对应请求的商品ID，那么参数属性可以选择URL参数，并在参数名称中填写属性在请求中对应的字段名称。

- **URL参数名称**输入stockId。
- **阈值**输入*100*个请求/每**秒**。
- **流控方式**选择**快速失败**。



例如在促销活动中，某些恶意刷单请求较多的时候，会占用较多商品库存或服务器资源。这种情况下可以针对其IP来源进行排队等待的处理，使访问请求匀速通过，防止过量的请求对服务稳定性产生影响。在**新建Web防护规则**对话框中配置以下规则信息，表示每个不同来源IP调用此接口的请求会以每10 ms一个的速度匀速通过（1s/100=10 ms），后续多余的请求要进行排队等待。排队中的请求如果等待时长超过30 ms就会立即失败。



参数	说明
参数属性	选择Client IP。

参数	说明
阈值类型	默认选择*请求数*。
阈值	输入*100*个请求/每**秒**。
流控方式	选择**匀速排队**。
超时时间	输入*30*。
是否开启	选择*开启*。

场景二：防止恶意刷单

3.1.7. 集群流控

3.1.7.1. 配置集群流控规则

集群流控可以控制某个服务调用整个集群的实时调用量，可以解决因流量不均匀导致总体限流效果不佳的问题。集群流控可以精确地控制整个集群的调用总量，结合单机限流兜底，更好地发挥流量防护的效果。本文主要介绍设置集群流控的操作步骤。

背景信息

集群流控通常适用于以下场景：

- 单机流量不均：由于负载不均衡等原因导致每台机器的流量不均，这时使用单机流控可能会出现没有达到请求总量，某些机器就开始限流的情况。
- 集群小流量流控：某些高可用防护场景下需要将服务调用QPS限制到很小的量，此时平均到每台机器的QPS可能小于1，无法通过单机流控进行精确控制。例如希望限制总QPS为50，但节点数有100个。
- 有业务含义的流量控制：例如限制某个API每个用户每分钟调用不超过10次。

> **说明** Token Client与Server的通信会带来一定的网络开销，响应时间可能会上升2 ms~5 ms左右。

步骤一：选择档位创建集群

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 在应用防护管理页左侧导航栏单击**集群流控**。
5. 在**集群流控资源配置**区域内，选择**集群类型**为**生产**，然后滑动指针选择集群流控的总配置量级，单击**创建**，然后在对话框中单击**确认**。

   总配置量级即最大QPS，表示需要流控的接口所能承载的预估的最大QPS，代表可能到来的最大流量。

   > **说明** 实际流量（无论是否被流控）超出配置的最大QPS后，流控策略会退化到单机模式。为保证流控效果，阈值之和上限为配置最大QPS的95%，例如最大QPS选择100000，则所有规则阈值之和最大值为95000。

   选定总配置量级档位并创建集群后，系统会自动为该应用分配集群的Token Server。
6. （可选）单击**Token Client设置**区域**操作**列的**编辑**，设置**Token请求超时时间**，然后单击**确定**。

在某些场景下，集群流控Client与Token Server之间的网络通信时延较高，需要调整超时时间。

> **说明**
> Token请求超时时间单位为ms，取值范围为(0,10000]，一般不建议超过20 ms。公网环境网络延时较高，建议设置超时时长约为50 ms，但不建议超过80 ms。

步骤二：设置集群流控规则

1. 在应用防护管理页左侧导航栏单击**规则管理**，选择目标方案页签，单击**流控规则**页签，然后单击**新增流控规则**。
2. 在**新增流控规则**对话框，开启**是否集群流控**，并设置相关参数。

参数	描述	示例
接口名称	设置接口名称。	function_9
是否集群流控	开启此开关，即对集群内此资源的调用总量进行限制。	开启
是否开启	开启此开关，规则即生效；关闭此开关，规则不生效。	开启
集群阈值	表示该接口的限流阈值。	100
统计窗口时长	集群流控统计的时间窗口长度，取值范围为1秒~24小时。	1秒
失败退化策略	当出现连接失败、通信失败或Token Server不可用等情况时，流控规则是退化到单机限流的模式或是直接通过忽略失败情况： ○ 退化到单机限流：当出现通信失败的情况时，退化到设置的单机阈值来进行流控。需要在规则中配置单机退化阈值，代表单机的兜底阈值。 ○ 直接通过：当出现通信失败的情况时，请求直接通过。	退化到单机限流
退化阈值自动调整	开启后会自动调整退化阈值，默认关闭。 **说明** 此功能需要SDK版本≥1.8.6支持。	关闭
退化单机阈值	代表单机的兜底阈值，当失败退化策略选择退化到单机限流时，需要设置此选项。	10

参数	描述	示例
自动调整增量值	当开启退化阈值自动调整时，需要设置自动调整的增量。这是在根据接口阈值与应用机器数量计算出的单机均摊流量基础上，用来提供保护退化阈值的一个增量。即单机均摊流量加上增量值为实际生效退化阈值。	2

3. 单击**新建**，完成规则创建。
   创建规则完成后，可以在**规则设置**页面查看到创建的集群流控规则，**阈值模式**为**集群总体**。

3.1.7.2. 查看集群详情

完成集群流控配置后，您可以在**集群详情**页面查看集群整体以及集群下各接口的数据详情，包括集群限流和分节点限流统计详情、限流比、接口流量环比、Token Client请求耗时和Token Client响应类型。

前提条件

完成集群流控规则配置。具体操作，请参见配置集群流控规则。

功能入口

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 在左侧导航栏，选择**集群流控 > 集群详情**。

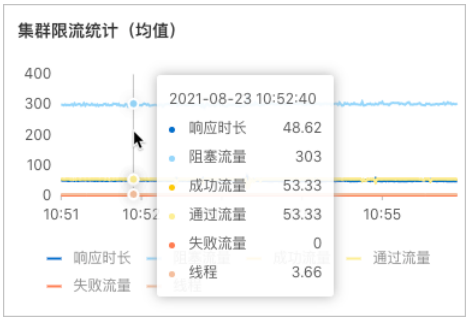
查看数据详情

目标集群详情页面显示了集群的具体情况。您可以在**集群名称**区域单击**全部接口**或目标接口名称，选择查看集群整体详情或各接口详情。

> **说明** 所有图表仅展示最近5分钟的数据。所展示数据每10秒钟刷新一次。

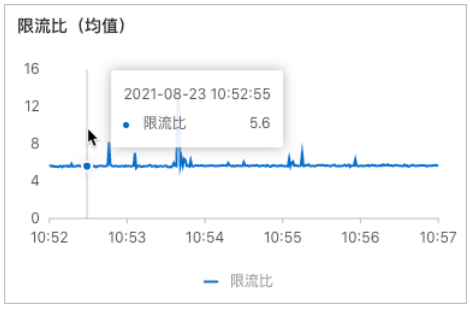
该图表展示统计时间段内集群限流的总体情况，其指标包括响应时长、阻塞流量、成功流量、通过流量、失败流量和线程。

您可以将鼠标悬浮在图上，查看指定时刻下指标的具体数据。您还可以单击面板上的指标名称（例如响应时长），打开或关闭该指标在图表中的可见性。



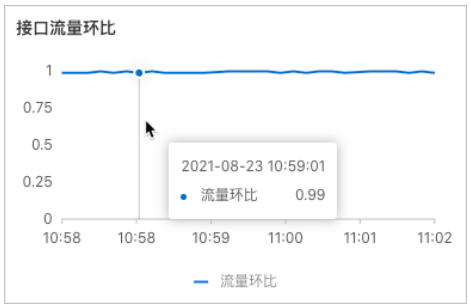
该图表展示被限流流量与通过流量的比值。若比值较大，说明过多流量被阻塞。此时，您可以根据实际需求调整限流规则。

您可以将鼠标悬浮在图上，查看指定时刻下指标的具体数据。

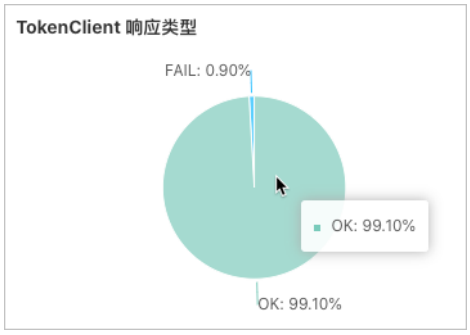


该图表通过计算当前10秒与前10秒的外部流量的比值，展示外部流量随时间变化的情况。

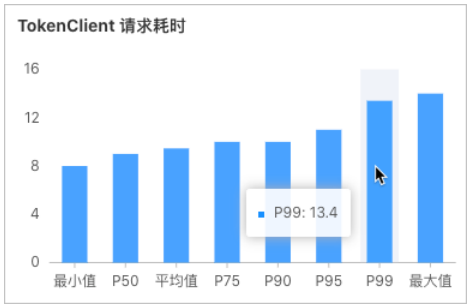
您可以将鼠标悬浮在图上，查看指定时刻下指标的具体数据。



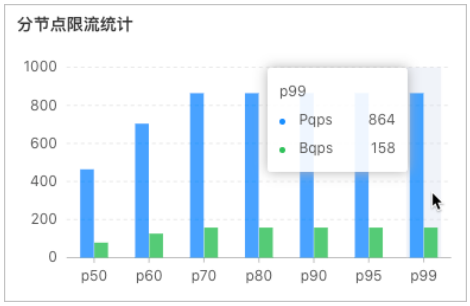
该图表展示Token Client响应成功或失败的比例。



该图表展示Token Client请求的实际耗时情况。您可以参考P99或最大值显示的实际值来调整集群配置中所设置的Token请求超时时间。



该图表展示流量在不同节点的分布情况，以检查负载均衡是否符合预期。PQPS为通过流量，BQPS为阻塞流量。



> **说明** 您可以滑动**选择查询时刻**区域的指针，查看**Token Client响应类型**、**Token Client请求耗时**和**分节点限流统计**图表在指定时刻下的具体数据。

分节点限流统计

Token Client响应类型

Token Client请求耗时

3.1.8. 管理应用

3.1.8.1. 应用概览

将应用接入MSE应用防护后，MSE将监控各应用、接口、机器的实时数据，从而评估系统的整体表现，并为流控降级规则提供重要依据。本文介绍应用概览页的主要功能。

功能入口

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。

功能介绍

应用概览页面会动态刷新，展示应用的限流指标详情，以TOP形式罗列请求、流控、响应时间、异常事件等信息。

- **限流指标**：统计了应用QPS、RT、CPU等数据。

> **说明** **应用概览**中涉及到的QPS、响应时间均为应用入口接口的统计，不包括应用内部方法调用的统计。

- **QPS数据**：展示了近5分钟通过请求数、流控请求数和异常请求数的时序图。
  - 单击**历史数据**，可以查看QPS更详细的历史数据，最多可以查看7天内的历史数据。
  - 单击图例，可以隐藏或展示该指标的时序图。
- **防护事件**：展示了该应用在流量防护期间触发的防护事件。
  - 单击**详情**，查看该防护事件的详情。
  - 单击**防护事件查询**，可查看更多的防护事件。
- **RT**：响应时间，单位为ms。展示了近5分钟响应时间的时序图。单击**历史数据**，可以查看RT更详细的历史数据，最多可以查看7天内的历史数据。
- **CPU**：用户CPU使用率。展示了近5分钟CPU使用率。单击**历史数据**，可以查看CPU更详细的历史数据，最多可以查看7天内的历史数据。
- **状态码统计**：展示了近5分钟不同时间HTTP各个状态码的数量。
  - 单击**历史数据**，可以自定义查看任意时间段的状态数据和对比数据，最多查看7天历史数据。
  - 单击图例，可以隐藏或展示该指标的时序图。

- **TOP列表**：包括通过QPS、流控QPS、平均RT的TOP接口列表，以及CPU的TOP节点列表。

- TOP接口列表会动态刷新，按**通过QPS**由大到小排列。
- 单击接口名称或该区域右上角的**查看全部**，进入**接口详情**页面，查看所有接口的QPS、CPU、Load等详细信息。
- 单击**CPU TOP**列表的节点名，进入**机器监控**页面，查看所有节点的应用指标。
- 单击目标接口**操作**列的**流控**、**隔离**或**降级**，可为该资源配置相应规则。

- **异常数据**：统计了该应用近5分钟的异常情况，为您排查系统问题提供有效信息。

> **说明** 该区域默认展示最近一次统计情况。

- 单击**异常统计数据**柱形图中的柱形，则在右侧**异常TOP**列表中，展示该异常的具体接口、类型、次数。
- 单击**异常统计数据**右上角的**历史数据**，可以查看任意时间段发生的异常次数，最多可以查看7天内的历史数据。
- 单击**异常TOP**列表**操作**列的**查看**，进入到**接口详情**页面，查看该接口的异常详情。
- 您在**异常统计数据**区域查看到异常数据后，可在应用本地日志中查看异常记录。

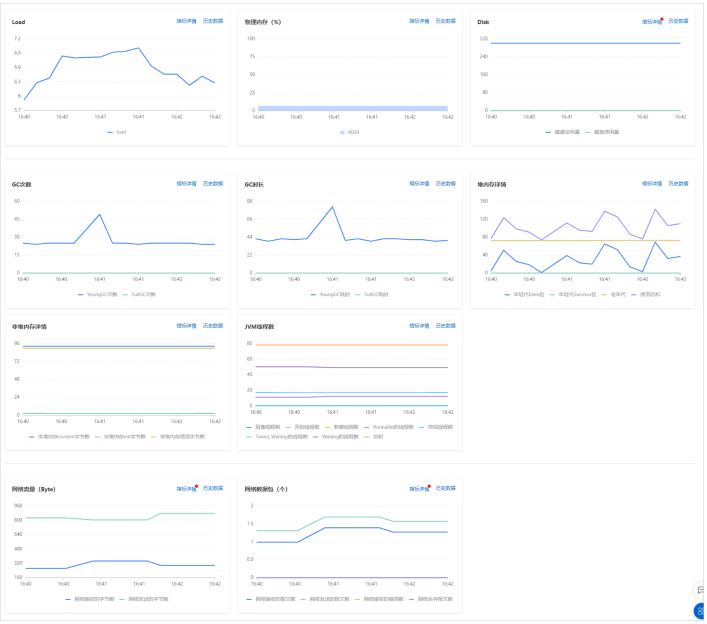
> **说明** 控制台查看以秒为单位的异常统计情况，异常详细内容您可以在本地异常记录文件中查看。

  - 异常记录的文件地址：与*metrics.log*同个目录。
  - 异常记录的文件名称：
    - *{appName}-MSE_exceptions.log.{日期，精确到天}*
    - *{appName}-MSE_exceptions.log.{日期，精确到天}.idx*
  - 异常记录格式：`{时间戳}|{时间，精确到毫秒}|{异常接口}|{异常详情}`

  样例如下所示：

```
1619749703000|2021-04-30 10:28:23.062|/exception1|com.alibaba.csp.sentinel.demo.exception.TestException: test exception_1 15
1619749703000|2021-04-30 10:28:23.065|/exception|com.alibaba.csp.sentinel.demo.exception.AnotherException: test another exception 71
1619749705000|2021-04-30 10:28:25.065|/exception1|com.alibaba.csp.sentinel.demo.exception.TestException: test exception_1 38
```

- **系统资源指标**：展示了系统的Load、物理内存、Disk等指标的时序图。



- 单击各指标时序图中的**指标详情**，查看各指标的节点分位图、TOP列表等详细信息。
- 单击各指标时序图中的**历史数据**，可以查看各个指标更详细的历史数据，最多可以查看7天内的历史数据。
- 鼠标悬浮在图中某一点，可查看该时刻该指标的具体数据。

3.1.8.2. 接口详情

在接口详情页面，主要展示该应用所有接口的通过QPS、限流QPS、异常QPS指标、RT、并发数据等，还可以在此页面为接口管理流控规则。本文介绍接口详情页的主要功能。

功能入口

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 在左侧导航栏选择**接口详情**。

功能介绍

接口详情页面展示了该应用的所有接口的详细信息，包括统计的QPS、RT、并发等数据。

您还可以在此页面进行以下操作：

- （图标①）在页面右上角选择**展示模式**，默认**详情展示**。
  - **详情展示**：以时序图和时序列表的形式展现接口的通过QPS、限流QPS、RT等信息。
  - **统计展示**：以列表的形式展现某一天接口的指标占比、通过总请求数、拒绝总请求数等信息。
- （图标②）在页面右上角可以选择**回放时间**，查看接口的历史数据。

  > **说明** 最多保留7天的历史数据。

- （图标③）在顶部菜单栏中单击各类型的页签可以过滤对应类型的接口，包括**WEB**、**RPC**等。
- （图标④）在接口列表区域，单击接口名称，可以具体查看该接口QPS数据时序图、RT数据时序图、并发数据时序图以及防护事件等，以及该接口在不同节点上的流量情况。
- （图标⑤）在时序图区域，可以选择要展示或隐藏的指标，还可以选择接口指标的展现形式。
  - **节点对比**：各接口以卡片的形式展现各接口的数据。
  - **集群统计**：以QPS、RT、并发各数据的统计维度展现接口的数据。

  单击右上角的图标，可以导出接口详情的数据，包括某一时间段某些接口秒级的QPS数据、RT数据等。一次最多可导出5个接口的数据，也可以选择导出PDF格式或者CSV格式。

  > **说明** 模式的切换仅在全部接口场景下支持。

- （图标⑥）在时序图区域，还可以对各接口设置流控规则等操作。
  - 单击图标，可以将该接口添加至流量大盘，便于在流量大盘中观测系统整体流量，具体操作，请参见创建流量大盘。
  - 单击或图标，进入**管理规则**或**新增规则**对话框，可以新增或删除流控、隔离和降级规则，也可以编辑已有的规则或开启关闭规则。具体操作，请参见配置流控规则、配置隔离规则和配置熔断规则。

- 单击 图标，可以查看该接口指标的历史数据。

  > **说明** 最多保留7天的历史数据。

- 单击 图标，可以导出该接口详情的数据，包括某一时间段秒级的QPS数据、RT数据等，也可以选择导出PDF格式或者CSV格式。
- 单击接口名称后，会在右侧**接口概览**页展示该接口对应的各数据时序图。
  - 单击**节点详情**页签，筛选查看不同接口的数据。
  - 单击**状态统计**页签，展示该接口的HTTP状态码、错误数状态码时序图等。

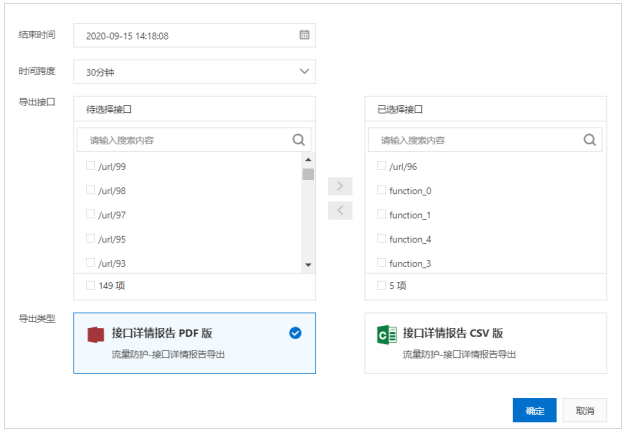
    > **说明** 错误码默认为非2xx和3xx的状态码。

  - 单击**异常统计**页签，查看该应用近5分钟的异常情况。
    - 单击**异常统计数据**柱形图中的柱形，则在右侧**异常TOP**列表中，展示该异常的具体接口、类型、次数。
    - 单击**异常统计数据**右上角的**历史数据**，可以查看任意时间段发生的异常次数，最多可以查看7天内的历史数据。

导出接口数据

MSE可以导出各个接口详情的数据，包括某一时间段接口秒级的QPS数据、RT数据等，便于您统计分析数据。

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 在左侧导航栏选择**接口详情**。
5. 您可以通过以下任意一种方式导出数据。
   - 在**接口详情**页时序图区域的右上角单击 图标，可以导出多个接口的数据。
   - 在**接口列表**区域，单击接口名称，在时序图区域的右上角单击 图标，可以导出该接口的数据。
6. 在**导出接口详情数据**对话框中，选择时间、导出接口以及导出类型，单击**确定**。



> **说明** 目前支持一次性最多导出5个接口的数据。

3.1.8.3. 机器监控

在机器监控页面，主要展示了所有节点的通过QPS、限流QPS、异常QPS、RT、并发等指标，还可以在此页面为接口管理流控规则。一个节点对应一个JVM进程，当多个JVM接入单机后，即展示为多个节点。本文介绍机器监控页的主要功能。

功能入口

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 在左侧导航栏中选择**机器监控**。

功能介绍

机器监控页面展示了应用的所有节点详细信息以及这些节点的QPS、CPU、LOAD时序图。

您可以在此页面进行以下操作：

- （图标①）在页面右上角选择**回放时间**，查看该时间前5分钟内的历史数据。

  > **说明** 最多保留7天的历史数据。

- （图标②）在**节点名称**区域，罗列了全部节点和对应的通过QPS、限流QPS、异常QPS、RT等信息。单击节点名称可以查看对应的各数据时序图。
- （图标③）在时序图区域，可以进行以下操作：
  - 在时序图区域，可以选择要展示或隐藏的指标，还可以选择接口指标的展现形式。
    - **节点对比**：各接口以卡片的形式展现各接口的数据。
    - **集群统计**：以QPS、RT、并发各数据的统计维度展现接口的数据。

    > **说明** 模式的切换仅在全部接口场景下支持。

  - 在**节点概览**页签，单击**QPS**、**CPU**、**LOAD**等页签，可以分别查看全部节点相关指标的时序图，还可以选择要展示或隐藏的指标。
  - 在**JVM监控**页签，单击**GC次数**、**GC耗时**等页签，可以查看全部节点JVM的数据。
  - 单击 图标，可以查看该接口指标的历史数据。

    > **说明** 最多保留7天的历史数据。

- 单击节点名称后，会在右侧**节点概览**页展示该节点对应的各数据时序图。
  - 单击**JVM监控**页签，查看该接口的GC次数、GC耗时等时序图。
  - 单击**接口详情**页签，筛选查看不同接口的数据。
  - 单击**callstack信息**页签，查看所有接口的信息，并可以设置该接口的限流规则、查看历史数据。
    - **平铺展示**：不区分调用链路关系，平铺展示接口的运行情况。
    - **树状展示**：根据接口的调用链路关系，展示树状结构。
    - 单击目标接口**操作**列中的**流控**、**隔离**或**降级**，可以快速管理限流规则。
    - 单击目标接口**操作**列中的**更多 > 查看监控**，可查看该接口指标的历史数据和数据对比情况。
  - 单击**异常统计**页签，查看该应用近5分钟的异常情况。
    - 单击**异常统计数据**柱形图中的柱形，则在右侧**异常TOP**列表中，展示该异常的具体接口、类型、次数。
    - 单击**异常统计数据**右上角的**历史数据**，可以查看任意时间段发生的异常次数，最多可以查看7天内的历史数据。

3.1.8.4. 规则管理

在规则管理页面，主要展示了流控、隔离、降级等规则下包含的接口信息，还可以通过此页面管理各个接口的限流规则。本文介绍规则管理页面的主要功能。

功能入口

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 在左侧导航栏中选择**规则管理**。

功能介绍

在规则管理页面以不同类型的规则为维度，展现了各个规则下包含的接口信息，包括接口的来源、阈值模式、流控效果等，还可以在此页面管理接口的流控、隔离、降级等规则。

您可以在此页面进行以下操作：

- 单击**新建方案**，可以新增不同的规则的方案组合。
- 单击各个页签，查看各个规则下包含的接口名称、来源应用、规则状态等信息。
- 在各规则页面，单击**新增XX规则**，可以快速创建流控、隔离等规则。
  具体操作，请参见以下文档：
  - 配置流控规则
  - 配置隔离规则
  - 配置熔断规则
- 单击目标接口**操作**列的**编辑**或**删除**，可以快速管理限流规则。
- 选择目标接口**操作**列的**更多 > 操作日志**，查看该接口相关的操作日志。
- 单击目标接口**操作**列的**更多 > 历史事件**，可以查看该接口各指标的历史数据。

> **说明** 最多保留7天的历史数据。

3.1.8.5. 管理基本信息

在应用管理页面可以查看所有资源相关的接入节点、操作日志、事件中心的详情，本文介绍如何管理接入节点、事件和操作日志。

管理接入节点

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 在左侧导航栏中选择**应用管理**，然后单击**接入节点**页签。
   在节点列表，展示了各个节点的名称、IP地址、健康状态等信息。

管理事件中心

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 在左侧导航栏选择**应用管理**，然后单击**事件中心**页签。

   在事件中心列表，罗列了相关的防护事件，包括事件的级别、类型、起始时间等信息，其中级别包含了WARNING和ERROR。
5. 单击**操作**列的**查看详情**，查看该接口的历史监控数据。

管理操作日志

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 在左侧导航栏选择**应用管理**，然后单击**操作日志**页签。

   在操作日志列表，罗列了相关的操作日志，包括具体的操作内容、类别、操作人ID等信息。
5. 在左上角的搜索框里输入资源名，搜索目标资源的相关操作日志。

3.1.8.6. 事件中心

在事件中心页面，主要展示了各个防护事件的详细信息，包括级别、类型、起始时间等。

操作步骤

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 在左侧导航栏选择**事件中心**，进入**事件中心**页面。

   在**事件中心**页面，罗列了相关的防护事件，包括事件的级别、类型、起始时间等信息，其中级别包含了WARNING和ERROR。
5. 单击**操作**列的**查看详情**，查看该接口的历史监控数据。

> **说明** 高级防护最多保留7天的历史数据，入门级防护保留半小时的历史数据。

3.1.8.7. 应用基础设置

在应用管理页面的基础设置中，您可以切换防护模式来提升系统防护能力，配置Web应用触发流控等规则后的处理逻辑，以及设置簇点数目限制、来源数目限制等信息。本文介绍如何进行应用的基础设置。

功能入口

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
4. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
5. 在左侧导航栏单击**应用管理**，然后单击**基础设置**页签。

设置适配模块

通过设置适配模块功能，可以动态配置Web类型应用触发流控等规则后的处理逻辑。具体操作步骤如下：

1. 在**模块适配设置**区域单击目标应用**操作**列的**修改**。
2. 在**模块适配设置修改**对话框中配置参数。

参数	描述	示例值
Web fallback行为	定义Web接口访问触发某种规则后的行为表现。目前支持以下两种策略： ○ 返回指定内容：需设置HTTP状态码、返回内容的格式和返回的内容。表示Web接口访问触发规则后返回自定义的内容。 ○ 跳转到指定页面：需设置指定跳转的URL。表示Web接口访问触发规则后系统会跳转指定的页面URL。	返回指定内容
HTTP状态码	默认429。当Web限流处理策略为自定义返回时，需要填写。	429
返回content-type	设置返回内容的格式为普通文本（TEXT）或JSON。当Web限流处理策略为自定义返回时，需要填写。	JSON字符串
HTTP返回文本	输入当Web接口访问触发规则后返回的内容。当Web限流处理策略为自定义返回时，需要填写。	{"message": "blocked oops"}

参数	描述	示例值
跳转URL	输入当Web接口访问触发规则后系统会跳转的页面URL。当Web限流处理策略为跳转到指定页面时，需要填写。 **说明** 跳转的本质是返回302状态码。对于后端服务直接渲染返回的页面，跳转是有效的；对于前端通过AJAX请求到后端服务后，再解析后端返回到前端展示的页面，跳转无效。	http:/mse.console.aliyun.com

3. 单击**确定**。

说明

- 适配模块配置的配置项会覆盖JVM参数传入的相关配置项。
- 适配模块配置仅针对默认的Web流控处理逻辑生效。若您注册了自定义的UrlBlockHandler，则适配模块配置无效。

设置通用配置

通过通用设置功能可修改应用的簇点数目限制、来源数目限制、入口数目限制和最大统计RT等配置。具体步骤如下：

1. 在**通用设置**区域**操作**列单击**修改**。
2. 在对话框中按需配置**簇点数目限制**、**来源数目限制**、**入口数目限制**和**最大统计RT**。

配置项	说明
簇点数目限制	限制埋点资源数，默认6000。建议设置限制不超过6000，当实际资源数超过6000时，会导致占用内存较大。
来源数目限制	限制每个资源下来源节点数，用于按调用来源限流。
入口数目限制	限制入口context数，用于链路限流。
最大统计RT	限制统计的最大RT（ms），默认为4900 ms。当统计数超出设置上限时，则按照设置上限记录统计数。

3. 单击**确定**。

3.1.9. 告警管理

3.1.9.1. 管理告警联系人

告警被触发时会向您指定的联系人分组发送通知，而在创建联系人分组之前必须先创建联系人。创建联系人时，您可以指定联系人用于接收通知的手机号码和邮箱地址，也可以提供用于自动发送告警通知的钉钉机器人地址。

前提条件

：设置钉钉机器人告警：如需将钉钉机器人添加为联系人，则需要先获取钉钉机器人的地址。

创建联系人

1. 登录MSE管理控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**注册配置中心 > 告警管理 > 联系人管理**。
4. 选择**联系人**页签，单击右上角的**新建联系人**。
5. 在**新建联系人**对话框中编辑联系人信息，然后单击**确认**。
   - 如需添加联系人，请编辑联系人**姓名**、**手机号码**和**邮箱**。

     > **说明**
     > - 手机号码和邮箱必须至少填写一项。
     > - 每个手机号码或邮箱只能用于一个联系人。
     > - 最多支持添加100个联系人。

   - 如需添加钉钉机器人，请填写钉钉机器人地址。

     > **说明** 获取钉钉机器人地址的方法参见设置钉钉机器人告警。

   - 如需接收系统通知，请勾选**是否接收系统通知**。

创建联系人分组

1. 登录MSE管理控制台。
2. 在左侧导航栏选择**注册配置中心 > 告警管理 > 联系人管理**。
3. 选择**联系人组**页签，单击右上角的**新建联系组**。
4. 在**新建联系组**对话框中填写**组名**，选择**报警联系人**，并单击**确认**。

   > **说明** 如果**报警联系人**列表中没有选项，则您需要先创建联系人。

后续操作

- 如需搜索联系人，请在**联系人**页签上，从搜索下拉框中选择**姓名**、**手机号码**或Email，然后在搜索框中输入联系人姓名、手机号码或邮箱的全部或部分字符，并单击**搜索**。
- 如需编辑联系人，请单击联系人右侧**操作**列中的**编辑**，在**编辑联系人**对话框中编辑信息，并单击**确认**。
- 如需删除单个联系人，请单击联系人右侧**操作**列中的**删除**，并在弹出的对话框中单击**删除**。
- 如需删除多个联系人，请勾选目标联系人，单击**批量删除**，并在弹出的对话框中单击**确认**。
- 如需搜索联系组，请在**联系人组**页签的搜索框中输入联系人分组名称的全部或部分字符，并单击Q图标。

  > **注意** 英文搜索关键字区分大小写。

- 如需编辑联系组，请单击联系人分组右侧的图标，并在**编辑联系组**对话框中编辑相关信息。
- 如需查看联系组中的联系人信息，请单击联系人分组右侧的下箭头图标来展开联系组。

> **说明** 您可以在展开模式下移除联系组中的联系人。如需移除，请单击目标联系人**操作**列中的**移除**。

- 如需删除联系组，请单击联系人分组右侧的 × 图标。

> **注意** 删除联系组之前，请确保没有正在运行的监控任务，否则可能导致告警等功能失效。

相关文档

- 管理告警规则
- 设置钉钉机器人告警

3.1.9.2. 管理告警规则

MSE提供了应用监控告警功能，可在满足告警条件时通过邮件、短信、钉钉等渠道实时告警，帮助您主动发现异常。在告警管理模块中，您可以管理账号下自定义监控报警规则，并查询告警事件和告警通知的历史记录。

前提条件

创建联系人分组

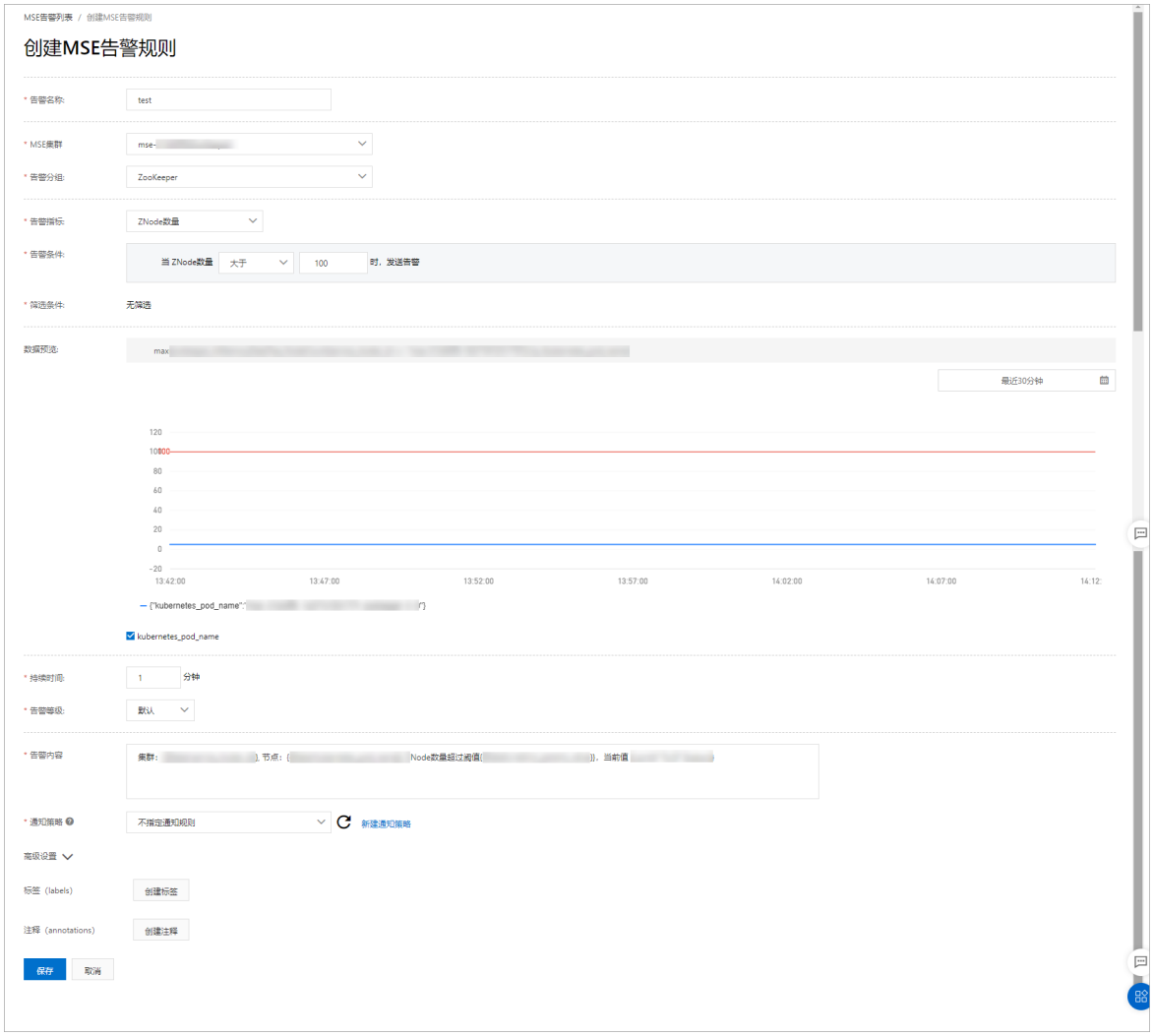
背景信息

报警控件本质是数据集的数据展示方式，所以在创建报警控件的同时，会创建一个数据集来存储报警控件的底层数据。

> **说明** 新建报警大约在10分钟内生效，报警判断会存在1分钟~3分钟的延时。

创建告警规则

1. 登录MSE管理控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**注册配置中心 > 告警管理 > 告警管理策略**。
4. 在**MSE告警列表**页面右上方单击**创建MSE告警规则**。
5. 在**创建MSE告警规则**页面配置告警相关参数，完成后单击**保存**。



告警参数说明

参数	描述
告警名称	填写告警规则名称。
MSE集群	选择集群。集群名称后的（）里显示该集群的注册配置中心类型，目前仅支持Nacos和ZooKeeper。
告警分组	选择告警指标的分组，目前支持Nacos和ZooKeeper。
告警指标	选择告警指标，不同告警分组所支持的告警指标也不同，请根据实际需求进行选择。
告警条件	设置触发告警的条件，例如：当服务数大于100时，发送告警。
筛选条件	默认**无筛选**，无需设置。
数据概览	当设置完告警条件后，在空白处单击鼠标左键，系统会自动弹出当前告警规则的预览数据。您可自定义事件周期进行筛选，当鼠标悬浮在图表上，可显示该时刻下的数据。

参数	描述
持续时间	设置满足告警条件的持续时间，当满足告警条件的时间达到设置的时间时，将会触发告警。
告警等级	设置告警的等级，默认告警等级为**默认**，告警严重程度从默认、P4、P3、P2、P1逐级上升。

告警内容	设置触发该告警时，所显示的告警内容。您可以使用Go template语法在告警内容中自定义告警参数变量，例如：
	告警集群的ID：
	`{{$label.service_cluster_id}}`
	告警的集群节点：
	`{{$label.kubernetes_pod_name}}`
	设置的阈值：
	`{{$labels.metrics_params_value}}`
	触发告警的实际值：`{{ printf "%.2f" $value }}`
	同时，告警内容也会根据告警指标自动进行调整。

通知策略	当告警触发时，告警中心会根据配置的通知策略对产生的告警事件进行分派、处理并发送通知。
高级设置	单击∧图标，设置告警规则标签和注释。
（可选）标签	单击**创建标签**，设置告警规则的标签，设置的标签可用作分派规则的选项。
（可选）注释	单击**创建注释**，设置**键**为*message*，设置**值**为*{{变量名}}告警信息*。设置完成后的格式为：`message:{{$labels.pod_name}}`**重启**`}}`**告警信息**，例如：`message:{{$labels.pod_name}}`**重启**。

后续操作

您在监控中创建的告警规则均会显示在**MSE告警列表**页面中。

1. 在**MSE告警列表**页面选中告警规则，在**操作**列中按需对目标报警规则采取以下操作。
   - 如需编辑告警规则，请单击**编辑**，并在**编辑MSE告警规则**页面中修改告警规则，然后单击**保存**。
   - 如需**启用**或**停止**告警规则，请单击**启动**或**停止**，并在提示对话框中单击**确认**。
   - 如需删除报警规则，请单击**删除**，并提示对话框中单击**确认**。
   - 如需查看告警历史，请单击**告警历史**，在**事件列表**页面中查看到历史告警信息。
2. （可选）如果您想要批量管理多条告警规则，可选中多条告警规则，然后执行以下操作。
   - 如需批量启动多条告警规则，请单击**批量启动告警**，并在提示对话框中单击**确认**。
   - 如需批量停止多条告警规则，请单击**批量停止告警**，并在提示对话框中单击**确认**。

- 如需批量删除多条告警规则，请单击**批量删除告警**，并在提示对话框中单击**确认**。

相关文档

- 管理告警联系人
- 设置钉钉机器人告警

3.1.9.3. 设置钉钉机器人告警

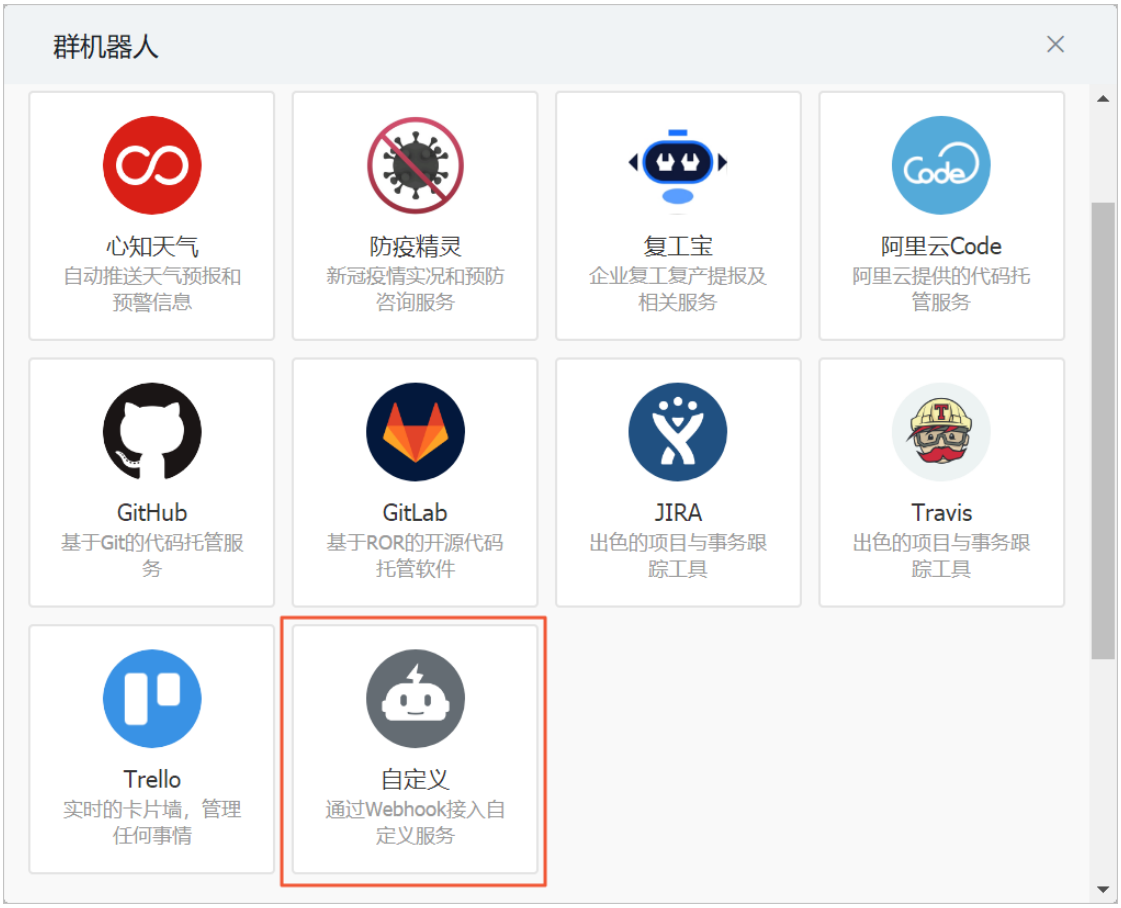
MSE告警支持钉钉群接收告警通知的功能。设置钉钉机器人告警后，您可以通过指定钉钉群接收告警通知。本文将介绍设置钉钉机器人告警的操作步骤。

操作步骤

1. 获取钉钉机器人地址。
   i. 在PC版钉钉上打开您想要添加告警机器人的钉钉群，并单击右上角的群设置图标。
   ii. 在群设置弹框中单击**智能群助手**。

iii. 在**智能群助手**页面单击**添加机器人**区域的 **+** 按钮。

iv. 在**群机器人**页面单击**添加机器人**右侧的 **+** 按钮，然后选择添加**自定义**机器人。



v. 在**机器人详情**页面单击**添加**。

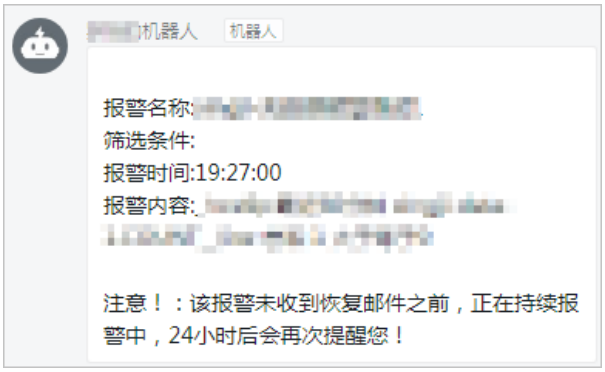
vi. 在**添加机器人**对话框中编辑机器人头像和名字，勾选**我已阅读并同意《自定义机器人服务及免责条款》**，然后单击**完成**。

若您想接收服务巡检的告警，需要在**安全设置**中选中**自定义关键词**，输入*微服务线上监控预警*。

vii. 在**添加机器人**对话框中复制生成的机器人地址。

2. 在控制台上添加钉钉机器人为联系人。具体操作，请参见创建联系人。
3. 创建一个联系组，并选择上一步创建的联系人为告警联系人。具体操作，请参见创建联系人分组。
4. 设置告警规则。
   - 若您未创建告警任务，请先创建告警，并选择通知方式为**钉钉机器人**，设置通知对象为第3步创建的联系组。具体操作，请参见管理告警规则。
   - 若您已创建告警任务，则需管理告警，选择通知方式为**钉钉机器人**，设置通知对象为第3步创建的联系组。具体操作，请参见管理告警规则。

操作至此，您已成功设置一个钉钉机器人告警。当告警触发时，您将在设置接收告警的钉钉群中收到告警通知。



3.1.10. 创建流量大盘

若您需要关注多个系统的整体流量情况，可以通过创建流量大盘功能来实现。流量大盘可展示不同应用、不同接口的通过QPS、拒绝QPS、异常QPS、RT和并发数等信息，给您带来更好的监控体验。

前提条件

接入应用防护或网关防护。

- 接入应用防护，请参见前提条件。
- 接入网关防护。

背景信息

流量大盘可以提高运维效率，例如在常见电商场景中，购物网站后端需要多个系统支撑，多个系统之间调用密切。若需查看下单操作的表现情况，可以将下单操作相关接口添加至大盘中，即可方便地查看各接口流量信息。运维人员无需关注各个系统，提高运维效率。

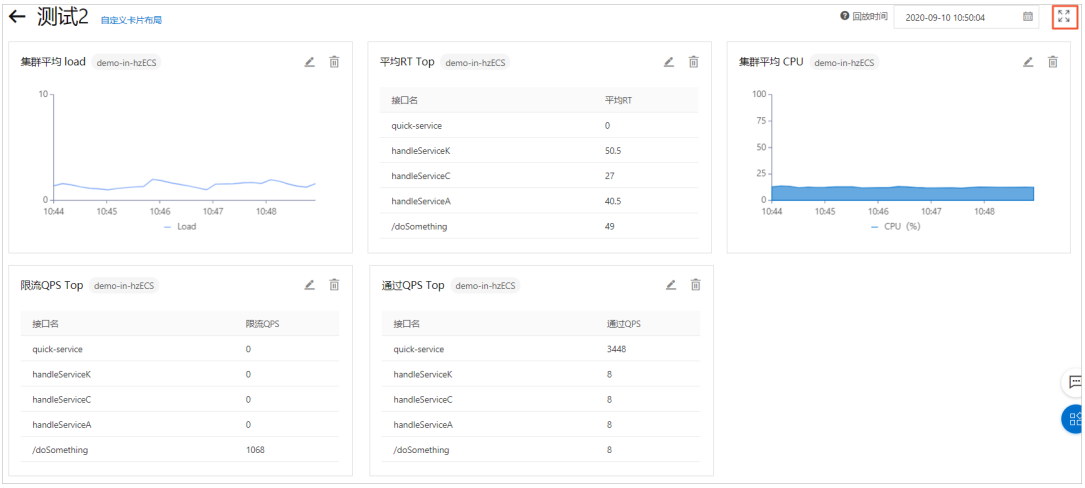
接入应用防护和接入网关防护的应用均可添加至流量大盘。

操作步骤

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 应用防护**。
3. 在**应用防护**页面，单击目标应用操作列的**应用防护**。
4. 在左侧导航栏选择**流量防护 > 流量大盘**。
5. 在**流量大盘**页面单击**创建大盘**。
6. 在**创建大盘**对话框中完成以下操作：
   i. 输入**大盘名**。
   ii. 选择配置流量大盘的应用。
   展示的应用包括接入应用防护和网关防护的应用。
   iii. 在双选框左侧面板中选择需查看的接口，然后单击>。
   所选择的接口将出现在双选框右侧面板中。
   iv. 单击**确认**。

执行结果

- 操作结束后，添加的大盘将显示在**流量大盘**页面。
- 单击目标流量大盘右上角的编辑图标，可对大盘进行编辑操作。
- 单击目标流量大盘，即可展示该大盘所添加接口的通过QPS、拒绝QPS、异常QPS、RT和并发数等信息。
- 在页面右上角单击展开图标，即可在全屏模式查看流量大盘。



3.1.11. SDK 使用手册

3.1.11.1. SDK参考概述

通过SDK接入方式将应用接入MSE应用防护后，如果默认的配置无法满足您的业务需求，可以使用MSE应用防护的SDK可以配置对代码块的流控、配置应用被流控降级后的行为以及配置扩展接口等。

配置对代码块的流控

MSE是围绕着资源来工作的，只要通过应用防护SDK定义的代码，即资源，就能够被应用防护保护起来。可通过定义资源来实现对代码块的流控。定义资源后，在MSE控制台配置相应的规则即可生效。可通过以下几种方式来定义资源：

- 注解方式定义资源
- 抛出异常的方式定义资源
- 返回布尔值方式定义资源
- 异步调用支持
- 主流框架的默认适配

定义资源详情请参见定义资源。

配置触发规则后的逻辑

当应用触发流控、降级或系统规则时，默认抛出 `BlockException` 异常类的子类（触发流控规则，则抛出流控异常 `FlowException` ；触发降级规则，则抛出降级异常 `DegradeException` ）。

若默认配置不能满足您的需求，可通过以下几种方式配置应用触发流控降级规则后的逻辑。

- 注解方式：适用于使用自定义埋点的Spring Boot应用接入、自定义埋点接入和注解接入等方式。
- Web Servlet Filter：适用于使用HTTP埋点的Spring Boot应用接入和Web应用接入等方式。
- Dubbo Adapter：适用于Dubbo应用接入方式。

配置触发规则后的逻辑详情请参见配置触发规则后的逻辑。

常用类、方法及扩展接口

MSE应用防护提供了常用类及其扩展方法和扩展接口，详情请参见常用类及其方法和扩展接口。

更多信息

- MSE提供了样例工程来帮助您体验应用防护功能，详情请参见样例工程。
- 可以通过查看日志，来快速查看单机运行情况，排查问题。日志详情请参见重要日志。

3.1.11.2. 定义资源

若您需要对特定的方法或者代码块进行流控，可以使用定义资源来实现。MSE提供了5种定义资源的方法，定义资源后，在MSE控制台为应用配置相应规则即可生效。

背景信息

MSE是围绕着资源来工作的。编码的时，只需关注如何定义资源，即哪些方法或代码块可能需要保护，而无需关注这个资源要如何保护。可通过定义资源来实现对代码块的流控，定义资源方式如下：

- 注解方式定义资源
- 抛出异常的方式定义资源
- 返回布尔值方式定义资源
- 异步调用支持
- 主流框架的默认适配

方式一：注解方式定义资源

通过 `@SentinelResource` 注解定义资源并配置 `blockHandler` 和 `fallback` 函数来进行限流之后的处理。

示例：

```
// 原本的业务方法.
@SentinelResource(blockHandler = "blockHandlerForGetUser")
public User getUserById(String id) {
    throw new RuntimeException("getUserById command failed");
}
// blockHandler 函数，原方法调用被限流/降级/系统保护的时候调用
public User blockHandlerForGetUser(String id, BlockException ex) {
    return new User("admin");
}
```

> **说明** `blockHandler` 函数会在方法触发限流、降级或系统保护规则的时候调用，而 `fallback` 函数仅会在原方法被降级时作为 fallback 方法，其它时候不会被调用。

更多信息请参见 Sentinel 注解支持文档。

方式二：抛出异常的方式定义资源

使用抛出异常的方式定义资源后，当资源发生了限流之后会抛出 `BlockException` 。您可以按需捕捉异常，并进行限流之后的逻辑处理。示例代码如下：

```
Entry entry = null;
// 务必保证finally会被执行
try {
  // 资源名可使用任意有业务语义的字符串
  entry = SphU.entry("自定义资源名");
  // 被保护的业务逻辑
  // do something...
} catch (BlockException ex) {
  // 资源访问阻止，被限流或被降级
  // 进行相应的处理操作
} finally {
  if (entry != null) {
    entry.exit();
  }
}
```

> **注意** `SphU.entry(xxx)` 需要与 `entry.exit()` 方法需匹配调用，否则会导致调用链记录异常，抛出 `ErrorEntryFreeException` 异常

方式三：返回布尔值方式定义资源

使用返回布尔值方式定义资源方式后，当资源发生了限流之后会返回 `false` 。可以根据返回值，进行限流之后的逻辑处理。示例代码如下：

```
// 资源名可使用任意有业务语义的字符串
  if (SphO.entry("自定义资源名")) {
    // 务必保证finally会被执行
    try {
      /**
      * 被保护的业务逻辑
      */
    } finally {
      SphO.exit();
    }
  } else {
    // 资源访问阻止，被限流或被降级
    // 进行相应的处理操作
  }
```

方式四：异步调用支持

MSE 支持异步调用链路的统计。在异步调用中，需要通过 `SphU.asyncEntry(xxx)` 方法定义资源，并通常需要在异步的回调函数中调用 `exit` 方法。示例如下：

```
try {
    AsyncEntry entry = SphU.asyncEntry(resourceName);
    // 异步调用
    doAsync(userId, result -> {
        try {
            // 在此处处理异步调用的结果
        } finally {
            // 在回调结束后 exit
            entry.exit();
        }
    });
} catch (BlockException ex) {
    // Request blocked
    // Handle the exception (e.g. retry or fallback)
}
```

`SphU.asyncEntry(xxx)` 不会影响当前调用线程的 Context，因此以下两个 entry 在调用链上是平级关系（处于同一层），而不是嵌套关系：

```
// 调用链类似于：
// -parent
// ---asyncResource
// ---syncResource
asyncEntry = SphU.asyncEntry(asyncResource);
entry = SphU.entry(normalResource);
```

若在异步回调中需要嵌套其它的资源调用（无论是 `entry` 还是 `asyncEntry`），只需要借助 Sentinel 提供的上下文切换功能，在对应的地方通过 `ContextUtil.runOnContext(context, f)` 进行 Context 变换，将对应资源调用处的 Context 切换为生成的异步 Context，即可维持正确的调用链路关系。示例如下：

```
public void handleResult(String result) {
    Entry entry = null;
    try {
        entry = SphU.entry("handleResultForAsync");
        // Handle your result here
    } catch (BlockException ex) {
        // Blocked for the result handler
    } finally {
        if (entry != null) {
            entry.exit();
        }
    }
}
public void someAsync() {
    try {
        AsyncEntry entry = SphU.asyncEntry(resourceName);
        // Asynchronous invocation
        doAsync(userId, result -> {
            // 在异步回调中进行上下文变换，通过 AsyncEntry 的 getAsyncContext 方法获取异步 Context
            ContextUtil.runOnContext(entry.getAsyncContext(), () -> {
                try {
                    // 此处嵌套正常的资源调用
                    handleResult(result);
                } finally {
                    entry.exit();
                }
            });
        });
    } catch (BlockException ex) {
        // Request blocked
        // Handle the exception (e.g. retry or fallback)
    }
}
```

此时的调用链如下：

```
-parent
---asyncInvocation
-----handleResultForAsync
```

普通资源与异步资源之间嵌套示例请参见AsyncEntryDemo.java。

方式五：主流框架的默认适配

为了减少开发的复杂程度，MSE对主流框架进行了适配，详情请参见支持组件列表。使用时只需要引入对应的依赖，它们框架的方法和服务都会自动被定义为资源，无需修改现有代码。

3.1.11.3. 配置触发规则后的逻辑

若默认配置不能满足您的需求时，您可以自定义应用触发流控、降级或系统规则后的逻辑。本文将介绍适用于SDK应用的逻辑配置方法。

背景信息

当应用触发流控、降级或系统规则时，默认抛出 `BlockException` 异常类的子类（触发流控规则，则抛出流控异常 `FlowException` ；触发降级规则，则抛出降级异常 `DegradeException` ）。

默认配置中，可以通过 `BlockException.isBlockException(Throwable t);` 方法判断是否为流控降级异常。

若默认配置不能满足您的需求，可通过以下几种方式配置应用触发流控、降级或系统规则后的逻辑：

- 注解方式：适用于使用自定义埋点的Spring Boot应用接入、自定义埋点接入和注解接入等方式。
- Web Servlet Filter：适用于使用HTTP埋点的Spring Boot应用接入和Web应用接入等方式。
- Dubbo Adapter：适用于Dubbo应用接入方式。

注解方式

若通过注解方式定义的资源，需要在 `@SentinelResource` 注解上给方法配置 `fallback` 函数以及 `blockHandler` 函数，来进行逻辑处理。

示例：

```
public class TestService {
    // 对应的 `handleException` 函数需要位于 `ExceptionUtil` 类中，并且必须为static函数。
    @SentinelResource(value = "test", blockHandler = "handleException", blockHandlerClass = {Ex
ceptionUtil.class})
    public void test() {
        System.out.println("Test");
    }
    // 原函数。
    @SentinelResource(value = "hello", blockHandler = "exceptionHandler", fallback = "helloFall
back")
    public String hello(long s) {
        return String.format("Hello at %d", s);
    }
    // Fallback函数，函数签名与原函数一致或加一个Throwable类型的参数。
    public String helloFallback(long s) {
        return String.format("Halooooo %d", s);
    }
    // Block异常处理函数，参数最后多一个BlockException，其余与原函数一致。
    public String exceptionHandler(long s, BlockException ex) {
        // Do some log here.
        ex.printStackTrace();
        return "Oops, error occurred at " + s;
    }
}
```

相关配置：

- `value` ：资源名称（必需项）。
- `entryType` ：entry类型（可选项），默认为 `EntryType.OUT` 。
- `blockHandler` / `blockHandlerClass` ： `blockHandler` 对应处理 `BlockException` 的函数名称，为可选项。blockHandler函数访问范围需要是 `public` ，返回类型需要与原方法相匹配，参数类型需要和原方法相匹配并且最后加一个额外的类型为 `BlockException` 的参数。blockHandler函数默认需要和原方法在同一个类中。若希望使用其它类的函数，则可以指定 `blockHandlerClass` 为对应的类的 `Class` 对象，对应的函数必需为static函数，否则无法解析。
- `fallback` ：fallback函数名称（可选项）。用于在抛出异常的时候提供fallback处理逻辑。fallback函数可以针对除了 `exceptionsToIgnore` 里排除掉的异常类型之外的异常进行处理。fallback函数签名和位置要求：

- 返回值类型必须与原函数返回值类型一致。
- 方法参数列表需要和原函数一致，或者可以额外多一个 `Throwable` 类型的参数用于接收对应的异常。
- fallback函数默认需要和原方法在同一个类中。若需使用其它类的函数，则可以指定 `fallbackClass` 为对应的类的 `Class` 对象，注意对应的函数必需为static函数，否则无法解析。

- `defaultFallback` （since 1.6.0）：默认的fallback函数名称（可选项），通常用于通用的fallback逻辑。默认fallback函数可以针对所有类型的异常（除了 `exceptionsToIgnore` 里面排除掉的异常类型）进行处理。若同时配置了fallback和defaultFallback，则只有fallback会生效。defaultFallback函数签名要求：
  - 返回值类型必须与原函数返回值类型一致。
  - 方法参数列表需要为空，或者可以额外多一个 `Throwable` 类型的参数用于接收对应的异常。
  - defaultFallback函数默认需要和原方法在同一个类中。若希望使用其他类的函数，则可以指定 `fallbackClass` 为对应的类的 `Class` 对象，注意对应的函数必需为static函数，否则无法解析。

> **说明** 若blockHandler和fallback都进行了配置，则被限流降级而抛出 `BlockException` 时只会进入 `blockHandler` 处理逻辑。若未配置 `blockHandler` 、 `fallback` 和 `defaultFallback` ，则被限流降级时会将 `BlockException` 直接抛出。

Web Servlet Filter

默认情况下，当请求被限流时会返回默认的提示页面，提示信息为： `Blocked by Sentinel (flow limiting)` 。

您可以通过以下两种方式来设定自定义的跳转URL，当请求触发限流、降级或系统规则时会自动跳转至设定好的URL。

- 通过 `WebServletConfig.setBlockPage(blockPage)` 方法设定自定义的跳转URL。
- 通过UrlBlockHandler接口编写定制化的限流处理逻辑，然后将其注册至WebCallbackManager中来实现，详情可参见Web Servlet Filter扩展接口。

Dubbo Adapter

Sentinel Dubbo Adapter支持配置全局的fallback函数，可以在Dubbo服务触发限流、降级或系统规则时进行相应的fallback处理。可通过以下两种方式来实现：

- 自定义DubboFallback接口，然后通过 `DubboFallbackRegistry` 注册。默认情况会直接将 `BlockException` 包装后抛出。
- 配合Dubbo的fallback机制来实现。

3.1.11.4. 常用类及其方法

本文为您介绍流控降级常用类和方法。

流控降级异常类BlockException

在Sentinel中所有流控降级相关的异常都是异常类 `BlockException` 的子类：

- 流控异常： `FlowException`
- 降级异常： `DegradeException`
- 系统保护异常： `SystemException`
- 热点参数限流异常： `ParamFlowException`

您可以通过以下方法判断是否为流控降级异常：

```
BlockException.isBlockException(Throwable t);
```

资源定义类SphU / SphO

`SphU` 和 `SphO` 是两个常用的用于资源定义的工具类。其中 `SphU` 以try-catch的形式定义资源，而 `SphO` 以if-else的形式定义资源。

`SphU` 类包含以下几组静态方法：

传入资源名定义资源：

- `public static Entry entry(String name) throws BlockException`
- `public static Entry entry(String name, int batchCount) throws BlockException`
- `public static Entry entry(String name, EntryType type) throws BlockException`
- `public static Entry entry(String name, EntryType type, int batchCount) throws BlockException`
- `public static Entry entry(String name, EntryType type, int batchCount, Object... args) throws BlockException`

其中资源名为传入的 `name` 。

传入Method对象定义方法资源：

- `public static Entry entry(Method method) throws BlockException`
- `public static Entry entry(Method method, int batchCount) throws BlockException`
- `public static Entry entry(Method method, EntryType type) throws BlockException`
- `public static Entry entry(Method method, EntryType type, int count) throws BlockException`
- `public static Entry entry(Method method, EntryType type, int count, Object... args) throws BlockException`

其中资源名将从传入的 `Method` 对象解析，格式为 `类名:方法签名` ，如 `com.alibaba.csp.sentinel.demo.DemoService:foo(java.lang.String)` 。

异步资源定义：

- `public static AsyncEntry asyncEntry(String name) throws BlockException`
- `public static AsyncEntry asyncEntry(String name, EntryType type) throws BlockException`
- `public static AsyncEntry asyncEntry(String name, EntryType type, int batchCount, Object... args) throws BlockException`

其中的参数解释如下。

参数名	类型	解释	默认值
entryType	`EntryType`	资源调用的流量类型，是入口流量（ `IN` ）、出口流量（ `OUT` ）、内部调用（ `INTERNAL` ）。注意系统自适应保护只对 `IN` 类型生效。	`EntryType.OUT`
resourceType	`Int`	资源调用分类，如Web/RPC/DB_SQL。	`COMMON` (0)
batchCount	`Int`	本次资源调用请求的Token数目（即算作几次调用）。	1

参数名	类型	解释	默认值
args	`Object[]`	传入的参数，用于热点参数限流。	无

返回值类型：

- 普通的资源定义返回 `Entry` 对象，代表本次资源的调用。
- 异步资源定义返回 `AsyncEntry` 对象，代表本次异步资源的调用。

更多使用请参见定义资源。

托管资源定义类SentinelWrapper

> **说明** SentinelWrapper在Java SDK 1.8.0及以上版本引入。

`SentinelWrapper` 用于定义托管执行的资源埋点。 `SentinelWrapper` 与 `SphU/SphO` 的不同在于 `SentinelWrapper` 需要您提供要执行的函数，并托管执行（与@SentinelResource注解方式类似），可以支持自动重试、超时熔断等机制。 `SentinelWrapper` 的主要函数有两个：

- `execute` ：在出现异常（包括被限流）时会直接抛出异常。
- `executeWithFallback` ：可以接受一个 `fallback` 函数来处理异常，返回正常的结果。

托管资源定义类SentinelWrapper如下：

- `public static <R> R execute(Callable<R> func, String resource, EntryType trafficType, int resourceType) throws Exception`
- `public static <R> R execute(Callable<R> func, String resource, EntryType trafficType, int resourceType, Object[] args) throws Exception`
- `public static <R> R executeWithFallback(Callable<R> func, CheckedFunction<Throwable, R> fallbackFunction, String resource, EntryType trafficType) throws Exception`
- `public static <R> R executeWithFallback(Callable<R> func, CheckedFunction<Throwable, R> fallbackFunction, String resource, EntryType trafficType, int resourceType) throws Exception`
- `public static <R> R executeWithFallback(Callable<R> func, CheckedFunction<Throwable, R> fallbackFunction, String resource, EntryType trafficType, int resourceType, Object[] args) throws Exception`

参数名	类型	解释	默认值
func	`Callable<R>`	用户要执行的函数，结果会体现在返回值上。	无（必传）
fallbackFunction	`CheckedFunction<Throwable, R>`	fallback函数，当出现异常时通过这个函数生成fallback结果。	无
entryType	`EntryType`	资源调用的流量类型，是入口流量（ `IN` ）、出口流量（ `OUT` ）、内部调用（ `INTERNAL` ）。注意系统自适应保护只对 `IN` 类型生效。	`EntryType.OUT`

参数名	类型	解释	默认值
resourceType	`Int`	资源调用分类，如Web/RPC/DB_SQL。	`COMMON`(0)
args	`Object[]`	传入的参数，用于热点参数限流。	无

Entry

`Entry`对象代表某一次资源调用，通过`SphU`或`SphO`定义资源后会返回此对象。主要方法：

- `public void exit() throws ErrorEntryFreeException`：表示资源调用结束，需要与`entry`方法成对出现。

异常描述：

- `ErrorEntryFreeException`：当前资源调用exit与entry不匹配会抛出此异常。资源的entry与exit必须成对出现。

业务异常记录类Tracer

业务异常记录类Tracer用于记录业务异常。相关方法：

- `public static void trace(Throwable e)`：记录业务异常（非`BlockException`异常）。
- `public static void trace(Throwable e, int count)`：记录业务异常，异常数目为传入的`count`。

如果用户通过`SphU`或`SphO`手动定义资源，则Sentinel不能感知上层业务的异常，需要手动调用`Tracer.trace(ex)`来记录业务异常，否则对应的异常不会统计到Sentinel异常计数中。

注解方式定义资源支持自动统计业务异常，无需手动调用`Tracer.trace(ex)`来记录业务异常。Web Servlet适配、Dubbo适配也会自动统计业务异常，无需手动统计。

上下文工具类ContextUtil

相关方法：

标识进入调用链入口（上下文）：

以下静态方法用于标识调用链路入口，用于区分不同的调用链路：

- `public static Context enter(String contextName)`
- `public static Context enter(String contextName, String origin)`

其中`contextName`代表调用链路入口名称（上下文名称），`origin`代表调用来源名称。默认调用来源为空。返回值类型为`Context`，即生成的调用链路上下文对象。

> **说明** `ContextUtil.enter(xxx)`方法仅在调用链路入口处生效，即仅在当前线程的初次调用生效，后面再调用不会覆盖当前线程的调用链路，直到exit。`Context`存于ThreadLocal中，因此切换线程时可能会丢掉，如果需要跨线程使用可以结合`runOnContext`方法使用。

流控规则中若选择“流控方式”为“链路”方式，则入口资源名即为上面的`contextName`。

退出调用链（清空上下文）：

- `public static void exit()`：该方法用于退出调用链，清理当前线程的上下文。

获取当前线程的调用链上下文：

- `public static Context getContext()`：获取当前线程的调用链路上下文对象。

在某个调用链上下文中执行代码：

- `public static void runOnContext(Context context, Runnable f)`：常用于异步调用链路中context的变换。

3.1.11.5. 扩展接口

本文为您介绍Web Servlet Filter、Dubbo Adapter等扩展接口。

Web Servlet Filter

- **自定义限流页面/处理逻辑**
  默认情况下，当请求被限流时会返回默认的提示页面。可通过三种方式设置自定义的跳转URL：
  - 方式一：`WebServletConfig.setBlockPage(blockPage)` 方法
    示例：

    ```
    //设置全局生效，被流控的所有页面都会跳转到这里。
    WebServletConfig.setBlockPage("https://www.example.test/");
    ```

  - 方式二：JVM `-Dcsp.sentinel.web.servlet.block.page=xxx`
    示例：

    ```
    //设置全局生效，被流控的所有页面都会跳转到这里。
    -Dcsp.sentinel.web.servlet.block.page=https://www.fallback.page.com/
    ```

- 方式三：更灵活的方式是定义 `UrlBlockHandler` 接口限流处理逻辑，并将其注册至 `WebCallbackManager` 。
  自定义处理逻辑的示例：

```
// 全局设置一次即可，比如在某个全局的init()方法里加入。
WebCallbackManager.setUrlBlockHandler(new UrlBlockHandler() {
    @Override
    public void blocked(HttpServletRequest request, HttpServletResponse response, BlockExce
ption ex)
        throws IOException {
        // request里包含了此次请求所有的信息，可以从其中解析出URL、请求参数等。
        logger.info("blocked: " + request.getPathInfo());
        // response表示响应对象，直接向其中写fallback结果即可。
        response.sendRedirect("https://www.fallback.page.com/"); // 将请求重定向到fallback地
址
    }
});
```

  返回Status 500的示例：

```
// 全局设置一次即可，比如在某个全局的init()方法里加入。
WebCallbackManager.setUrlBlockHandler(new UrlBlockHandler() {
    @Override
    public void blocked(HttpServletRequest request, HttpServletResponse response, BlockExce
ption ex)
        throws IOException {
        // request里包含了此次请求所有的信息，可以从其中解析出URL、请求参数等。
        logger.info("blocked: " + request.getPathInfo());
        // response表示响应对象，直接向其中写fallback结果即可。
        response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        response.getWriter().println("flow control");
    }
});
```

- **URL 资源清洗**
  Sentinel Web Servlet Filter会将每个到来的不同的URL都作为不同的资源处理，因此对于REST风格的API，需要自行实现 `UrlCleaner` 接口清洗资源，例如，将满足 `/foo/:id` 的URL都归到 `/foo/*` 资源下，然后将其注册至 `WebCallbackManager` 中。否则会导致资源数量过多，超出资源数量阈值（6000）时多出的资源的规则将**不会生效**。
  示例如下：

```
@PostConstruct
public void init() {
     // 全局注册一次即可
    WebCallbackManager.setUrlCleaner(new UrlCleaner() {
        @Override
        public String clean(String originUrl) {
             // 对originUrl进行变换，得到归一化后的URL
            if (originUrl == null || originUrl.isEmpty()) {
                return originUrl;
            }
             // 比如将满足/foo/:id的URL都归到/foo/*
            if (originUrl.startsWith("/foo/")) {
                return "/foo/*";
            }
            return originUrl;
        }
    });
}
```

- **解析请求来源**
  若希望对HTTP请求按照来源限流，则可以自己实现 `RequestOriginParser` 接口从HTTP请求中解析origin并注册至 `WebCallbackManager` 中，示例如下：

```
WebCallbackManager.setRequestOriginParser(new RequestOriginParser() {
    @Override
    public String parseOrigin(HttpServletRequest request) {
        return request.getRemoteAddr();
    }
});
```

Dubbo Adapter

Sentinel Dubbo Adapter支持配置全局的Fallback函数，可以在Dubbo服务被限流、降级或负载保护的时候进行相应的Fallback处理。用户只需要实现自定义的 `DubboFallback` 接口，并通过 `DubboFallbackRegistry` 注册即可。默认情况会直接将 `BlockException` 包装后抛出。

其它扩展接口

Sentinel提供多样化的SPI接口用于提供扩展的能力。用户可以在用同一个 `sentinel-core` 的基础上自行扩展接口实现，从而可以方便地根据业务需求给Sentinel添加自定义的逻辑。目前Sentinel提供如下的扩展点：

- 初始化过程扩展：提供 `InitFunc` SPI接口，可以添加自定义的一些初始化逻辑，如动态规则源注册等。
- Slot Chain扩展：用于给Sentinel功能链添加自定义的功能并自由编排。
- 指标统计扩展（StatisticSlot Callback）：用于扩展StatisticSlot指标统计相关的逻辑。
- Transport扩展：提供 `CommandHandler` 、 `CommandCenter` 等接口，用于对心跳发送、监控API Server进行扩展。

3.1.11.6. 样例工程

通过试用Sentinel样例工程，您能够更快捷地了解AHAS应用防护功能。

- sentinel-demo-basic：Sentinel的基本使用示例，包括流控、降级、系统保护和异步调用资源定义等demo。
- sentinel-demo-annotation-spring-aop：Sentinel使用注解的示例。
- sentinel-demo-dubbo：在Dubbo应用中接入Sentinel的示例。
- sentinel-demo-rocketmq：在RocketMQ中接入Sentinel的示例。

您也可以查看应用防护快速入门或更多的Sentinel样例工程集锦，熟悉AHAS应用防护（Sentinel）服务。

3.1.11.7. 重要日志

您可以通过查看日志快速查看单机运行情况，从而排查问题。本文列出了MSE提供的重要日志，适用于所有接入MSE应用防护的应用。

文件	路径	说明
秒级监控日志	`${user_home}/logs/csp/${app_name}-${pid}-metrics.log`	资源都会产生秒级日志，可以用来查看资源的运行情况。
*status.log日志	`/** * 写入格式： * String resource; "xx" + "_status" * long passQps; statistics[1] ---- 2xx * long blockQps; statistics[3] ---- 4xx * long successQps; statistics[0] ---- 200 * long exceptionQps; statistics[4] ---- 5xx * long rt; statistics[2] ---- 3xx * 清理工作由MetricAggregatorTask执行 */`	每分钟刷新一次，记录REST请求状态。
拦截详情日志	`{user_home}/logs/csp/sentinel-block.log`	规则生效之后，请求被拦截，就会产生对应的日志。
业务日志	`{user_home}/logs/csp/sentinel-record.log.${当天的日志}`	记录了规则的推送、接收、处理；资源调用情况，排查问题的时候会非常有帮助。
上报日志	`{user_home}/logs/csp/command-center.log.${当天的日志}`	应用和Dashboard发生通讯的记录，用于排查和Dashboard的通信问题。

您在使用SDK时会生成*status.log文件，如下图所示。

```
[redacted] csp % ls -l
total 160
-rw-r--r--  1 karl  staff  10882  1 13 12:48 KarlApp-metrics.log.2021-01-13
-rw-r--r--  1 karl  staff   1168  1 13 12:48 KarlApp-metrics.log.2021-01-13.idx
-rw-r--r--  1 karl  staff     65  1 13 12:47 KarlApp-status_metrics.log.2021-01-13
-rw-r--r--  1 karl  staff      0  1 13 12:47 KarlApp-status_metrics.log.2021-01-13.idx
-rw-r--r--  1 karl  staff   8745  1 13 12:48 agw-perf.log.2021-01-13.0
```

秒级监控日志

所有的资源都会产生秒级日志，它在 `${user_home}/logs/csp/${app_name}-${pid}-metrics.log` 里。格式如下：

```
1532415661000|2018-07-24 15:01:01|sayHello(java.lang.String)|12|3|4|2|295|10
```

1. `1532415661000` ：时间戳。
2. `2018-07-24 15:01:01` ：格式化之后的时间戳。
3. `sayHello(java.lang.String)` ：资源名。
4. `12` ：表示到来的数量，即此刻通过Sentinel规则check的数量（passed QPS）。
5. `3` ：实际该资源被拦截的数量（blocked QPS）。

6. `4`：每秒结束的资源个数（完成调用），包括正常结束和异常结束的情况（exit QPS）。
7. `2`：异常的数量。
8. `295`：资源的平均响应时间（RT）。
9. `10`：并发数。

拦截详情日志

无论限流，降级还是系统保护，它们的秒级拦截详情日志都记录在`{user_home}/logs/csp/sentinel-block.log`文件里，格式如下。

```
2014-06-20 16:35:10|1|sayHello(java.lang.String,long),FlowException,default,origin|61,0
2014-06-20 16:35:11|1|sayHello(java.lang.String,long),FlowException,default,origin|1,0
```

1. `2014-06-20 16:35:10`：时间戳。
2. `1`：序号。
3. `sayHello(java.lang.String,long)`：资源描述符。
4. `XXXException`：表示被限制的种类。`FlowException`表示被限流，`DegradeException`表示被降级，`SystemException`表示被系统保护。
5. `default`规则上配置的限制应用。
6. `origin`：实际被限制的来源应用，可能为空字符串。
7. `61,0`：61代表这一秒内限流降级发生的次数，0无含义（可忽略）。

业务日志

业务日志在`{user_home}/logs/csp/sentinel-record.log.xxx`中，包含规则的推送、接收、处理等记录。

上报日志

每一次和Dashboard的通信，都会记录在`{user_home}/logs/csp/command-center.log.${date}`日志中。可以用来排查Dashboard能否成功与机器连通等问题。

3.1.12. 参考信息

3.1.12.1. 应用防护原则概述

您在使用该功能前，请先了解以下原则。

MSE应用防护原则

- MSE是围绕着资源来工作的。
- 编码时，只需要关心如何定义资源，即哪些方法、代码块需要保护，而不需要关注如何保护这个资源。
- 通过添加规则来保护资源，规则添加即时生效。

规则配置原则

- 按照应用处理能力进行流控：
  - 按服务提供方流控原则，详情请参见服务提供方或消费方流控。
  - 削峰填谷原则，详情请参见削峰填谷。
  - 冷启动原则，详情请参见Warm Up（冷启动）。
  - 联动控制原则，详情请参见关联限流。

- 强依赖隔离原则，详情请参见强依赖隔离。
- 弱依赖降级原则，详情请参见弱依赖降级。
- 系统保护原则，详情请参见系统防护。

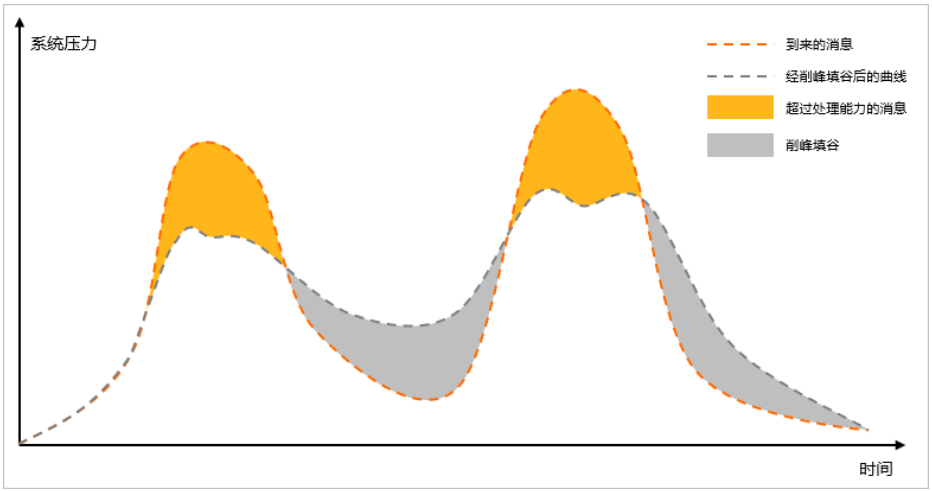
3.1.12.2. 应用防护方法

3.1.12.2.1. 削峰填谷

当消费端请求骤增时，可以为其配置排队等待的流控规则，以稳定的速度逐步处理这些请求，起到削峰填谷的效果，从而避免流量骤增造成系统负载过高。

背景信息

在实际应用中，收到的请求是没有规律的。例如：某应用的处理请求的能力是每秒10个。在某一秒，突然到来了30个请求，而接下来两秒，都没有请求到达。在这种情况下，如果直接拒绝20个请求，应用在接下来的两秒就会空闲。所以，需要把骤增的请求平均到一段时间内，让系统负载保持在请求处理水位之内，同时尽可能地处理更多请求。



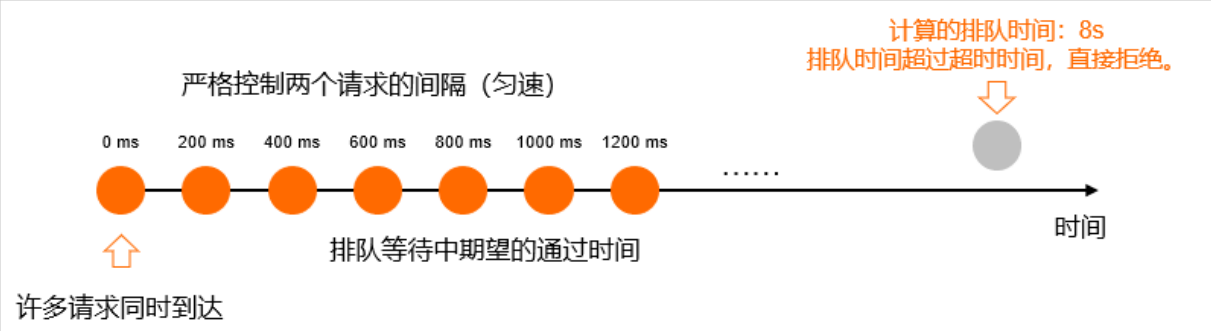
上图中，黄色的部分代表超出消息处理能力的部分。把黄色部分的消息平均到之后的空闲时间去处理，这样既可以保证系统负载处在一个稳定的水位，又可以尽可能地处理更多消息。通过配置流控规则，可以达到消息匀速处理的效果。

功能原理

AHAS流控降级的排队等待功能，可以把骤增的大量请求匀速分配，以固定的间隔时间让请求通过，起到“削峰填谷”的效果，从而避免流量骤增造成系统负载过高的情况。堆积的请求将会被排队处理，当请求的预计排队时间超过最大超时时长时，AHAS将拒绝这部分超时的请求。

例如：配置匀速模式下请求QPS为5，则每200 ms处理一条请求，多余的处理任务将排队；同时设置了超时时间为5s，则预计排队时长超过5s的处理任务将被直接拒绝。具体操作步骤，请参见新建流控规则。

示意图如下：



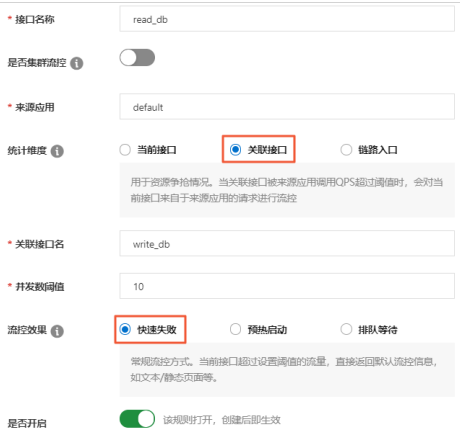
3.1.12.2.2. 关联限流

使用关联限流策略，可以避免具有关联关系的资源之间过度的争抢，造成的资源不可用问题。
当两个资源之间具有资源争抢或者依赖关系的时候，这两个资源便具有了关联。例如对数据库同一个字段的读操作和写操作存在争抢，读的速度过高会影响写得速度，写的速度过高会影响读的速度。如果放任读写操作争抢资源，则争抢本身带来的开销会降低整体的吞吐量。可使用关联限流来避免具有关联关系的资源之间过度的争抢。

示例

`read_db` 和 `write_db` 这两个资源分别代表数据库读写。给 `read_db` 设置以下规则来达到写优先的目的。具体操作步骤请参见配置流控规则常用场景。

当写库操作过于频繁时，读数据的请求会被限流。 `read_db` 会在 `write_db` 资源的QPS超过10之后，调用被拒绝。



3.1.12.2.3. Warm Up（冷启动）

对于长期处于低水位状态的系统，可以使用 Warm Up（冷启动）功能来避免流量骤增导致水位瞬间升高系统不可用的情况。

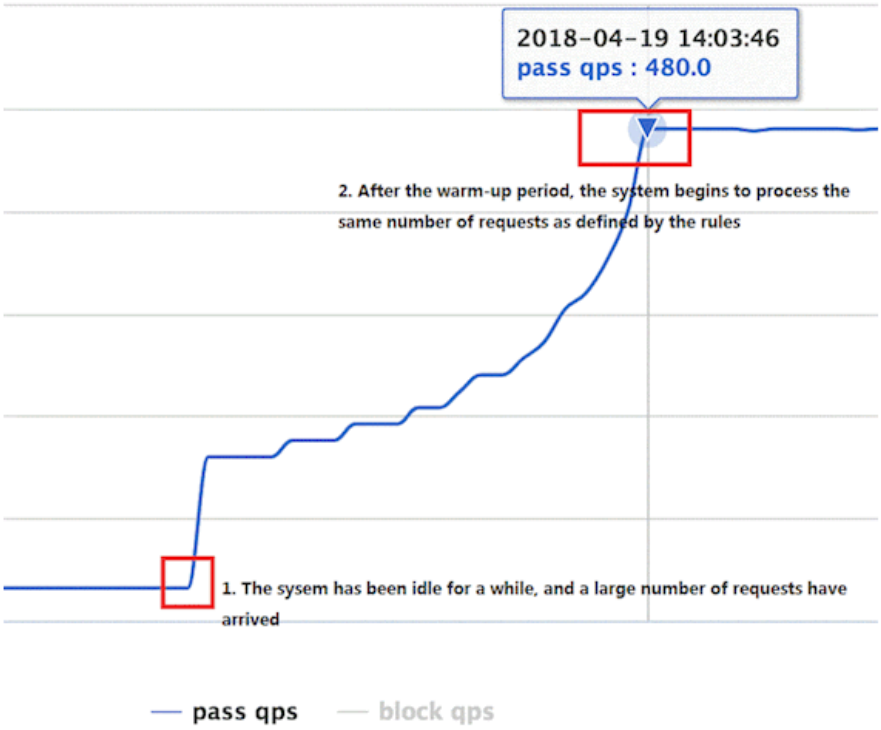
功能原理

Warm Up，即冷启动 / 预热的方式。当系统长期处于低水位的情况下，若流量突然增加，可能会把系统水位瞬间拉高把系统压垮。通过配置冷启动规则，可以让通过的流量缓慢增加，在一定时间内逐渐增加到阈值上限，给冷系统一个预热的时间，避免冷系统被压垮。

冷启动，参考了 Guava 的算法，通过随时调整斜率，把流量在指定的时间之类缓慢调整到特定的阈值。

示例

若对系统设置**流控模式**为直接，**流控方式**为 Warm Up，**预热时间**为 200 的流控规则，具体操作步骤，参见新建流控规则。设置规则后，可以看到流量的增长趋势如下图所示：



3.1.12.2.4. 服务提供方或消费方流控

AHAS 应用防护可以根据服务提供方的能力和服务消费方的分配能力进行流量控制。其中服务提供方（Service Provider）是指对外提供请求的服务或应用；服务消费方（Service Consumer）是指调用该服务的下游应用。

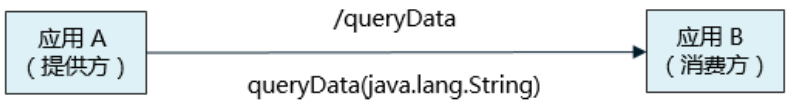
根据服务提供方限流

为了保护服务提供方不被激增的流量拖垮影响稳定性，您可以为其配置 QPS 模式的流控规则。当每秒的请求量超过设定的阈值时，AHAS 将拒绝多余的请求。提供方限流可以分为服务接口限流和服务方法限流。

- 服务接口限流：适用于整个服务接口的 QPS 不超过一定数值的情况。例如：为对应服务接口资源配置 QPS 阈值。
- 服务方法限流：适用于服务的某个方法的 QPS 不超过一定数值的情况。例如：为对应服务方法资源配置 QPS 阈值。

示例：

若应用 A 为服务提供方，其 Web 接口 `/queryData` 对应的方法是 `queryData(java.lang.String)`。假设应用 A 最多每秒只能承受 10 次调用。若调用次数超过，则应用 A 宕机。



针对此类场景，可以对应用 A 按服务方法粒度设置流控规则，限制 `queryData(java.lang.String)` 方法每秒最多只能被调用 10 次，具体操作步骤，参见新建流控规则。若超过阈值，消费方将会收到一个 `BlockException` 异常，并且快速返回。

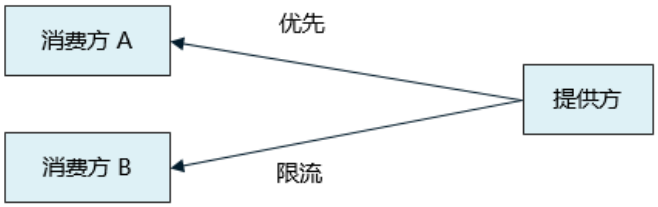
根据服务消费方限流

根据消费方限流是指根据调用方的需求来分配服务提供方的处理能力。

> **说明** 若限流规则未设置调用方（default），则该限流规则对所有调用方生效。若限流规则设置了调用方，则限流规则仅对指定调用方生效。

示例：

有两个服务消费者 A 和 B 都向服务提供方发起调用请求，我们希望优先服务消费方 A，而只对来自消费方 B 的请求进行限流。通过对消费方 B 设置限流规则（limitApp）来实现这个目的。



对于默认框架例如 Dubbo，AHAS 会自动解析 Dubbo 消费方的 `application name` 作为调用方名称（origin），在进行资源保护的时候都会带上调用方名称。而对于非默认框架，只需要在 Sentinel 原有的代码中加入 `ContextUtil.enter(resourceName, origin)` 和 `ContextUtil.exit()` 即可。示例代码如下：

```
Entry entry = null;
//这里代表消费者 A 的调用
ContextUtil.enter(queryData, "A");
try {
  // 资源名可使用任意有业务语义的字符串
  entry = SphU.entry("queryData");
  // 被保护的业务逻辑
  // do something...
} catch (BlockException e1) {
  // 资源访问阻止，被限流或被降级
  // 进行相应的处理操作
} finally {
  if (entry != null) {
    entry.exit();
  }
}
//调用结束
ContextUtil.exit();
```

> **说明** `ContextUtil.enter(xxx)` 方法仅在初次调用链路入口时才生效，可参考相关 API 文档。

更多设置

限流规则中的 `limitApp`（针对来源）支持以下三种选项，分别对应不同的场景：

1. `default`：适用于不区分消费者的场景。来自任何调用者的请求都将进行限流统计。如果这个资源名的调用总和超过了这条规则定义的阈值，则触发限流。
2. `{some_origin_name}`：适用于对特定的消费者限流的场景。只有来自这个调用者的请求才会进行流量控制。
   例如，资源 A 配置了一条针对消费者 caller1 的规则，那么当且仅当来自消费者 caller1 对资源 A 的请求才会触发流量控制。

3. `other`：表示针对除 `{some_origin_name}` 以外的其余调用方的流量进行流量控制。这个场景适用于资源对大部分消费者都有一个通用的阈值，对特定消费者有不一样的阈值的场景。
   例如，资源 A 可以对大部分消费者可以每秒提供 10 个请求，但是对于消费者 caller1 是个例外，对 caller1，每秒可以提供 200 个请求。需配置两条规则，说明如下：
   - 为消费者 caller1 配置一条 `limitApp` 为 caller1 的限流规则，这条规则每秒的最大请求量设置为 200。
   - 同时，配置一条 `limitApp` 为 other 的规则，这条规则每秒的最大请求量设置为 10，那么任意来自非 caller1 对该 resource 的调用，都不能超过 10。

> **说明** 同一个资源名可以配置多条规则，规则的生效顺序为：{some_origin_name} > other > default。

3.1.12.2.5. 弱依赖降级

当弱依赖的第三方应用出错不会影响整体流程，则称之为弱依赖。对于弱依赖不稳定时，需要配置降级规则来保护系统稳定性。

背景信息

在实际业务中，应用通常会调用依赖方（远程服务、数据库、第三方 API 等）来完成服务。例如，支付的时候需要远程调用银联提供的 API。然而依赖方的稳定性是不能保证的。若依赖方出现不稳定的情况，则请求和调用依赖方的方法的响应时间变长，线程产生堆积，最终可能耗尽自身的线程数，导致应用本身不可用。

在复杂链路中，若某一环不稳定，就可能会层层渲染，最终导致整个链路都不可用。

针对以上情况，可以使用 AHAS 应用防护功能对依赖方配置降级规则来保证系统稳定性。

功能原理

若应用依赖于多个下游服务（弱依赖），当下游服务调用过慢，则会严重影响当前服务的调用。为调用端配置基于平均响应时间或错误率的降级规则后，当调用链路中某个下游服务调用的平均响应时间或错误率超过阈值，AHAS 就会对此调用进行降级操作，拒绝多余的调用，保护应用不被调用端短板影响。

配置降级规则具体操作步骤，请参见配置熔断规则。

同时可以配合 Fallback 功能使用，在被降级的时候提供相应的处理逻辑。

3.1.12.2.6. 强依赖隔离

若依赖的第三方应用或组件，或者应用自身的内部方法出错会影响而整体流程，则称之为强依赖。对于强依赖，需要配置隔离原则来保护系统稳定性。

功能原理

当强依赖出现不稳定的时候，可以通过配置并发线程数隔离原则来限制不稳定的强依赖并发数，隔离强依赖。配置并发线程数隔离原则后，无需再进行线程池隔离，AHAS会控制资源的线程数。当请求数超过阈值时，AHAS将拒绝多余的请求，直到堆积的线程处理完成，以此来达到信号量隔离的效果。

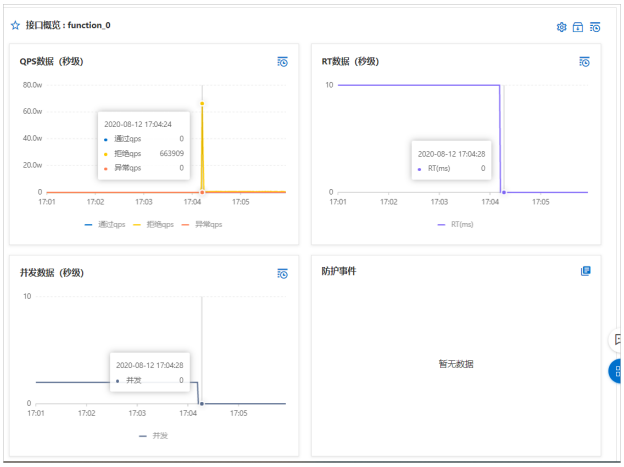
线程数目超出时，设置能够有效地防止自己被慢调用所影响。快速失败

示例

若有应用A的一个接口 `function_0` 不稳定，出现慢SQL。

针对此类场景，可以为调用设置隔离规则（将阈值类型选择为线程数，并将阈值设置为0），具体操作步骤请参见配置隔离规则。

配置规则后，`function_0` 方法被限流，各SQL调用情况如下图所示。



3.1.12.2.7. 系统防护

系统防护即从整体维度对应用入口流量进行控制，结合应用的 Load、总体平均 RT、入口 QPS 和线程数等几个维度的监控指标，让系统的入口流量和系统的负载达到一个平衡，让系统尽可能跑在最大吞吐量的同时保证系统整体的稳定性。

背景信息

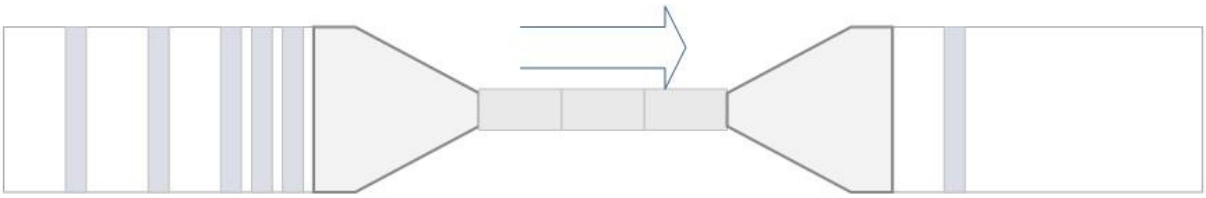
长期以来，系统自适应保护的思路是根据硬指标即系统的负载来做系统过载保护。即当系统负载高于某个阈值，就禁止或者减少流量的进入；若负载恢复，则恢复流量的进入。这样会造成两个不可避免的问题：

- 若根据负载的情况来调节流量的通过率，则会产生延迟。若当前通过率的调整会导致负载增大，那么至少要过 1 秒之后才能被观测到；同理，若当前通过率调整会使负载降低，也需要 1 秒之后才能继续调整。这种方法会浪费系统的处理能力。导致我们看到的负载曲线产生锯齿。
- 通过率恢复慢。在下游应用不可靠，应用响应时间很长，从而导致负载很高的场景中，若下游应用恢复时，应用响应时间也会随之减短，此时通过率理应会大幅度增大。但由于此时负载仍然很高，所以通过率的恢复慢。

为解决上述问题，AHAS 应用流控降级在系统自适应保护的做法是：用每分钟的负载作为启动控制流量，使用请求的响应时间以及当前系统正在处理的请求速率来决定通过的流量。旨在在系统不被拖垮的情况下，提高系统的吞吐率。

功能原理

我们把系统处理请求的过程想象为一个水管，到来的请求是往这个水管灌水，当系统处理顺畅的时候，请求不需要排队，直接从水管中穿过，这个请求的RT是最短的；反之，当请求堆积的时候，那么处理请求的时间则会变为：排队时间 + 最短处理时间。



若用 T 来表示水管内部的水量，用 RT 来表示请求的处理时间，用 P 来表示进来的请求数，那么一个请求从进入水管道到从水管出来，这个水管会存在 `P * RT` 个请求。即当 `T ≈ QPS * Avg(RT)` 的时候，可以认为系统的处理能力和允许进入的请求个数达到了平衡，系统的负载不会继续增加。当入口的流量是水管出来的流量的最大的值的时候，水管的处理能力达到最大利用。

系统规则

系统保护规则是应用整体维度的，而不是资源维度的，并且仅对入口流量生效。入口流量指的是进入应用的流量（EntryType.IN），例如 Web 服务或 Dubbo 服务端接收的请求，都属于入口流量。系统规则支持四种阈值类型：

- Load（仅对 Linux/Unix-like 机器生效）：当系统 load1 超过阈值且系统当前的并发线程数超过系统容量时才会触发系统保护。
- RT：当单台机器上所有入口流量的平均 RT 达到阈值即触发系统保护。
- 线程数：当单台机器上所有入口流量的并发线程数达到阈值即触发系统保护。
- 入口 QPS：当单台机器上所有入口流量的 QPS 达到阈值即触发系统保护。

为应用配置系统规则，具体操作步骤请参见新建系统规则。

3.1.12.3. 性能基准

本文列出了MSE应用流控降级在特定CPU、OS、Java版本的测试环境下的基准表现。

测试环境

基准的测试环境：

- CPU：Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz （32 cores）
- OS：Linux 2.6.32-220.23.2.ali927.el5.x86_64
- Java版本：

```
java version "1.8.0_77"
Java(TM) SE Runtime Environment (build 1.8.0_77-b03)
Java HotSpot(TM) 64-Bit Server VM (build 25.77-b03, mixed mode)
```

吞吐量对比

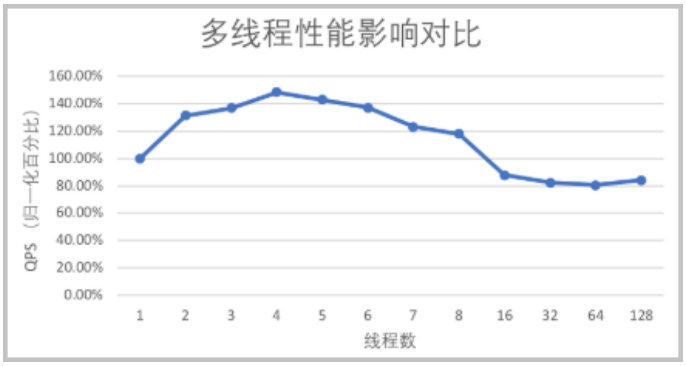
所有吞吐量测试都基于JMH编写。

- 单线程吞吐量：
  JVM参数：`-Xmn256m -Xmx1024m -XX:+UseConcMarkSweepGC`
  通过执行CPU密集型操作（如小数组排序）模拟不同QPS下的情况，来测试单线程模式下接入MSE应用流控降级与不接入MSE应用流控降级的对比。测试逻辑见SentinelEntryBenchmark。相关结果如下：

数组长度	Baseline（QPS）	With Sentinel（QPS）	性能损耗
length=25	604589.916	401687.765	33.56%
length=50	221307.617	192407.832	13.06%
length=100	97673.228	91535.146	6.28%
length=200	43742.960	42711.129	2.36%
length=500	15332.924	15171.024	1.06%
length=1000	7012.848	6984.036	0.41%

  - 当单机QPS为20万以上时，MSE应用流控降级带来的性能损耗比较大。这种情况业务场景（如缓存读取操作）本身的耗时非常小，而MSE应用流控降级的统计、检查操作会消耗一定的时间。
  - 单机QPS在5万以下时，MSE应用流控降级性带来的能损耗比较小，适用于大多数场景。

- 多线程吞吐量影响：
  在相同逻辑（对length=25的数组进行shuffle并排序）的情况下，测试不同线程数下接入MSE应用流控降级的性能表现：



内存占用情况

测试场景：6000个资源循环执行（即单机的极端场景，目前最多支持6000个Entry）。

- 单线程循环执行：内存占用约185MB。
- 8线程循环执行：内存占用约1GB（若系统持续高并发持续很，将导致底层的LongAdder内存占用很高）。

3.2. Java网关防护

3.2.1. 什么是Java网关防护

MSE可以对网关进行流量控制，从流量入口处拦截骤增的流量，防止下游服务被压垮。

> **注意** 目前Java 网关防护处于灰度状态，如果您对这些功能有诉求，您可以按照ACK微服务应用接入MSE服务治理企业版操作，通过增加白名单方式体验该功能。

网关防护的主要功能如下：

- 针对路由配置中的某个路由进行流量控制，或者自定义一组API进行流量控制。
- 针对请求的客户端IP、Header或者URL参数进行流控。
- 限制某个API的调用频率，支持秒、分钟、小时、天等多个时间维度。

关于网关防护的规则配置有如下功能：

- 如何查看网关所有接口的QPS、RT等数据，请参见接口详情。
- 如何查看所有节点的QPS、RT等数据，请参见机器监控。
- 如何创建网关流控规则，请参见API流控规则。
- 如何自定义API，请参见API管理。

3.2.2. 控制台操作

3.2.2.1. 接口详情

在接口详情页面，主要展示该应用所有接口的通过QPS、限流QPS、异常QPS指标、RT 、并发数据等，还可以管理网关接口的流控规则。本文介绍网关防护的接口详情页的主要功能。

功能入口

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 网关防护**。
4. 在**网关防护**页面单击目标应用卡片。

5. 在左侧导航栏选择**接口详情**。

功能介绍

接口详情页面展示了该网关的所有接口的详细信息，包括统计的QPS、RT、并发等数据。

您还可以在此页面进行以下操作：

- （图标①）在页面右上角选择**展示模式**，默认**详情展示**。
  - **详情展示**：以时序图和时序列表的形式展现接口的通过QPS、限流QPS、RT等信息。
  - **统计展示**：以列表的形式展现某一天接口的指标占比、通过总请求数、拒绝总请求数等信息。
- （图标②）在页面右上角可以选择**回放时间**，查看接口的历史数据。

  > **说明** 高级防护最多保留7天的历史数据，入门级防护仅保留半小时的历史数据。

- （图标③）在接口列表区域，单击接口名称，可以具体查看该接口QPS数据时序图、RT数据时序图、并发数据时序图以及防护事件等，以及该接口在不同节点上的流量情况。
- （图标④）在时序图区域，可以选择要展示或隐藏的指标，还可以选择接口指标的展现形式。
  - **卡片模式**：各接口以卡片的形式展现各接口的数据。
  - **概览模式**：以QPS、RT、并发各数据的统计维度展现接口的数据。

  > **说明** 模式的切换仅在全部接口场景下支持。

- （图标⑤）在时序图区域，还可以对各接口设置流控规则等操作。
  - 单击 图标，可以将该接口添加至流量大盘。
  - 单击 或 图标，进入管理规则对话框，可以新增或删除流控规则，也可以编辑已有的规则或开启关闭规则。
  - 单击 图标，可以查看该接口指标的历史数据。

  > **说明** 高级防护最多保留7天的历史数据，入门级防护仅保留半小时的历史数据。

3.2.2.2. 机器监控

在机器监控页面，主要展示了所有节点的通过QPS、限流QPS、异常QPS、RT、并发等指标，还可以在此页面为接口管理流控规则。本文介绍机器监控页的主要功能。

功能入口

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 网关防护**。
4. 在**网关防护**页面单击目标应用卡片。
5. 在左侧导航栏选择**接口详情**。

功能介绍

机器监控页面展示了应用的所有节点详细信息以及这些节点的QPS、CPU、LOAD时序图。

您可以在此页面进行以下操作：

- （图标①）在页面右上角选择**回放时间**，查看历史数据。

  > **说明** 高级防护最多保留7天的历史数据，入门级防护仅保留半小时的历史数据。

- （图标②）在**节点名称**区域，罗列了全部节点和对应的通过QPS、限流QPS、异常QPS、RT等信息。单击节点名称可以查看对应的各数据时序图。
- （图标③）在时序图区域，可以进行以下操作：
  - 单击**QPS**、**CPU**、**LOAD**页签，可以分别查看全部节点相关指标的时序图，还可以选择要展示或隐藏的指标。
  - 单击 图标，可以查看该接口指标的历史数据。

    > **说明** 高级防护最多保留7天的历史数据，入门级防护仅保留半小时的历史数据。

  - 单击节点名称后，会在右侧**节点概览**页展示该节点对应的各数据时序图。可以单击**分接口详情**页签，筛选查看不同接口的数据。还可以单击**callstack信息**页签，查看所有接口的信息，并可以设置该接口的限流规则、查看历史数据。
    - 平铺视图：不区分调用链路关系，平铺展示接口的运行情况。
    - 树状视图：根据接口的调用链路关系，展示树状结构。
  - 单击目标接口**操作**列中的**流控**、**隔离**或**降级**，可以快速管理限流规则。
  - 单击目标接口**操作**列中的**查看监控**，可查看该接口指标的历史数据。

3.2.2.3. API管理

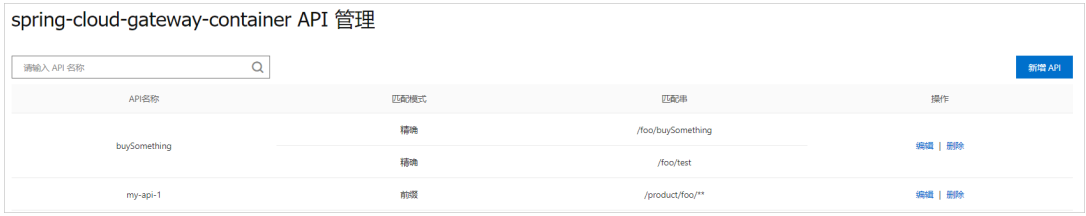
在网关防护中，您可以创建API分组，并自定义每个API下面的URL路径匹配规则。网关防护可以针对自定义的API分组进行流量控制。本文介绍如何在网关防护中管理API。

新建自定义API

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > Java网关防护**。
4. 在**网关防护**页面单击目标应用卡片。
5. 在左侧导航栏选择**API管理**，单击页面右上角的**新增API**。
6. 在**新建自定义API**对话框中，填写API名称。

   > **说明** 该名称需要全局唯一，并且不能与路由配置文件中的路由ID重复。

7. 填写URL路径匹配规则，先选择**匹配模式**，再根据匹配模式的要求填写**匹配串**。
   - **匹配模式**分为以下三类：
     - **精确模式**：严格按照给定的匹配串来匹配URL路径。示例：`/foo` 代表严格按照 `/foo` 这个路径来匹配。
     - **前缀模式**：按照给定的匹配串来进行前缀匹配，匹配串需符合Spring Web风格。示例：`/foo/**` 代表匹配以 `/foo/` 开头的所有URL，像 `/foo/22` 这种URL都可以匹配。
     - **正则模式**：按照给定的正则表达式匹配串来进行匹配。
   - **匹配串**：根据匹配模式的要求填写匹配串。
8. 单击**+新增匹配规则**，可添加多个URL路径匹配规则。
9. 单击**新增**，完成自定义API的创建。
   新增的API将出现在**API管理**页面。



相关操作

新增API后，您可以编辑、删除API。

- **编辑API**
  i. 在**API管理**页面，在目标API的**操作**列，单击**编辑**。
  ii. 在**编辑自定义API** 对话框中，修改URL匹配规则，也可以新增URL匹配规则。
- **删除API**
  i. 在**API管理**页面，在目标API的**操作**列，单击**删除**。
  ii. 在提示框中，单击**确定**，将该API分组删除。

3.2.2.4. 集群流控

相较于普通的单机流控，集群流控可以精确控制集群内某个服务的实时调用总量。在网关防护中采用集群流控，用户可无需关心负载均衡状况和网关数量，只需配置总阈值即可完成操作。本文主要介绍设置集群流控的操作步骤。

前提条件

- 开通企业版。相关内容，请参见微服务治理升级为企业版。
- MSE治理中心已接入微服务应用，相关内容，请参见：
  - ACK微服务应用接入MSE治理中心
  - ECS微服务应用接入MSE治理中心

步骤一：选择档位创建集群

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > 网关防护**。
4. 在**网关防护**页面单击目标应用卡片。
5. 在网关防护管理页左侧导航栏，单击**集群流控**。
6. 在**集群流控资源配置**区域内，单击右下角的**编辑**。
7. 选择**集群类型**为**生产**，滑动指针选择集群流控的总配置量级，单击**保存**，然后在对话框中单击**确认**。

   总配置量级即最大QPS，表示需要流控的接口所能承载的预估的最大QPS，代表可能到来的最大流量。

   > **说明** 实际流量（无论是否被流控）超出配置的最大QPS后，流控策略会退化到单机模式。为保证流控效果，阈值之和上限为配置最大QPS的95%，例如最大QPS选择100000，则所有规则阈值之和最大值为95000。

   选定总配置量级档位并创建集群后，系统会自动为该应用分配集群的Token Server。
8. （可选）单击**Token Client设置**区域**操作**列的**编辑**，设置**Token请求超时时间**，然后单击**确定**。

   在某些场景下，集群流控Client与Token Server之间的网络通信时延较高，需要调整超时时间。

   > **说明**
   > - AHAS Sentinel Client 1.6.0及以上版本支持设置Token Client。
   > - Token请求超时时间单位为ms，取值范围为(0,10000]，一般不建议超过20 ms。公网环境网络延时较高，建议设置超时时长约为50 ms，但不建议超过80 ms。

步骤二：设置集群流控规则

1. 在网关防护管理页左侧导航栏，单击**规则管理**。
2. 单击**集群流控规则**页签，然后单击**新增集群流控规则**。
3. 在**新增集群流控规则**对话框，设置相关参数。

参数	描述	示例
接口名称	设置接口名称，为对应网关的Route ID或自定义API分组名称。	httpbin_route
是否开启	开启此开关，规则即生效；关闭此开关，规则不生效。	开启

参数	描述	示例
集群阈值	表示该接口的限流阈值。	100
统计窗口时长	集群流控统计的时间窗口长度，取值范围为1秒~24小时。	1秒
失败退化策略	当出现连接失败、通信失败或Token Server不可用等情况时，流控规则是退化到单机限流的模式或是直接通过忽略失败情况： ○ 退化到单机限流：当出现通信失败的情况时，退化到设置的单机阈值来进行流控。需要在规则中配置单机退化阈值，代表单机的兜底阈值。 ○ 直接通过：当出现通信失败的情况时，请求直接通过。	退化到单机限流
退化阈值自动调整	开启后会自动调整退化阈值，默认关闭。 **说明** 此功能需要SDK版本≥1.8.6。	关闭
退化单机阈值	代表单机的兜底阈值，当失败退化策略选择退化到单机限流时，需要设置此选项。 **说明** 只有在没有开启**退化阈值自动调整**的情况下，才需要手动填写**退化单机阈值**。若开启**退化阈值自动调整**，您无需填写**退化单机阈值**，而需设置**自动调整增量值**。	10
自动调整增量值	当开启退化阈值自动调整时，需要设置自动调整的增量。这是在根据接口阈值与应用机器数量计算出的单机均摊流量基础上，用来提供保护退化阈值的一个增量。即单机均摊流量加上增量值为实际生效退化阈值。	2

4. 单击**新建**，完成规则创建。
创建规则完成后，可以在**规则设置**页面查看到创建的集群流控规则，**阈值模式**为**集群总体**。

3.2.2.5. API流控规则

为网关应用配置网关流控规则后，MSE将从流量入口处拦截激增的流量，防止下游服务被压垮。本文将介绍如何为已接入MSE的网关应用配置网关流控规则。

新建网关流控规则

1. 登录MSE治理中心控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏选择**微服务治理中心 > 流量防护 > Java网关防护**。
4. 在**网关防护**页面单击目标应用卡片。
5. 单击目标网关应用卡片，然后任选一种方式进入**API流控规则**的配置页面：
   - 在**接口详情**页面，单击API资源卡片右上角的加号图标。
   - 在左侧导航栏中单击**API流控规则**，然后在页面右上角单击**新增流控规则**。
6. 在**新增流控规则**对话框中，配置流控规则。

参数	描述
API	选择适用该规则的自定义API，或者手动输入路由配置文件中的Route ID。
针对请求属性	○ 关闭**针对请求属性**开关：不针对请求属性（如Client IP，URL参数等）进行限流，直接针对该API的所有请求进行流量控制。 ○ 开启**针对请求属性**开关：针对该API的某个请求属性进行限流，可以选择参数属性。 可以根据以下属性进行流量控制： ■ **Client IP**：请求端的IP地址。 ■ **Remote Host**：请求端的Host Header。 ■ **Header**：根据指定的HTTP Header进行解析，匹配对应的Header Key。选择**Header**后，可以配置请求属性值的匹配策略，只有匹配该模式的请求属性值会纳入统计和流控。 ■ **URL参数**：根据指定的HTTP URL参数进行解析，需要填写对应的参数名称。选择**URL参数**后，可以配置请求属性值的匹配策略，只有匹配该模式的请求属性值会纳入统计和流控。 ○ 匹配模式 ■ **精确**：严格按照给定的匹配串来匹配值。 ■ **子串**：若请求属性值包含该子串则匹配成功，如子串匹配 `ab` ，则 `aba` 和 `cabc` 都可以匹配，而 `cba` 则不能匹配。 ■ **正则**：按照给定的正则表达式匹配串来进行匹配。

参数	描述
阈值类型	○ **QPS**：应用或服务流量的QPS指标。选择QPS后，还需设置QPS阈值和统计间隔（支持秒、分钟、小时、天4种维度）。 例如，QPS阈值填写10，统计间隔选择**分**，则代表每分钟对应的请求数目不超过10个。 ○ **线程数**：资源的并发线程数，即该资源正在执行的线程数。 **说明** 开启**针对请求属性**开关后，暂时不支持线程数作为阈值类型。
流控方式	○ **快速失败**：当**阈值类型**为**QPS**时，被拦截的流量将快速失败。即达到阈值时，立即拦截请求。 ○ **匀速排队**：当**阈值类型**为**QPS**时，被拦截的请求将匀速通过，允许排队等待。 需设置具体的超时时间，预计达到超时时间的请求会立即失败，而不会排队。 例如，QPS配置为10，则代表请求每100 ms才能通过一个，多出的请求将排队等待通过。超时时间代表最大排队时间，超出最大排队时间的请求将会直接被拒绝。 **说明** 匀速排队时，QPS不要超过1000（请求间隔1 ms）。
Burst size	当流控方式为**快速失败**时，可以额外设置一个Burst Size，即针对突发请求额外允许的请求数目。
超时时间	当流控方式为**匀速排队**时，需设置具体的超时时间，达到超时时间后请求会失败。例如，QPS配置为5，则代表请求每200 ms才能通过一个，多出的请求将排队等待通过。超时时间代表最大排队时间，超出最大排队时间的请求将会直接被拒绝。

7. 单击**新增**。
   新增的规则将出现在**API流控规则**页面。

管理流控规则

在**流控规则**页面，您可以启用、禁用、编辑或删除流控规则。

- 单流控规则启用或禁用：
  在**流控规则**页面，找到目标资源下对应的流控规则，单击状态**状态**栏的启用开关，可快速启用或禁用该规则。
- 多流控规则批量启用或禁用：
  在**流控规则**页面，勾选多个流控规则，单击**批量启用**或**批量禁用**，可快速启用或禁用多个规则。
- 编辑规则：
  在**流控规则**页面，找到目标资源下对应的流控规则，单击**操作**栏的**编辑**，可修改该规则的相关信息。
- 删除规则：
  在**流控规则**页面，找到目标资源下对应的流控规则，单击**操作**栏的删除**删除**。

3.2.3. SDK使用手册

3.2.3.1. 触发网关防护规则后的限流策略

若默认配置不能满足您的需求时，您可以自定义应用触发流控、降级或系统规则后的逻辑。本文将介绍适用于SDK接入方式的逻辑配置方法。

Spring Cloud Gateway

若您的网关是Spring Cloud Gateway，则默认的限流处理逻辑是返回默认的流控文本 `Blocked by Sentinel`，返回 `status code` 为 `429 Too Many Requests`。您可以通过以下Spring配置项来配置限流后的处理策略。

- `spring.cloud.sentinel.scg.fallback.mode`：限流处理策略，目前支持跳转 `redirect` 和自定义返回 `response` 两种策略。
- `spring.cloud.sentinel.scg.fallback.redirect`：限流之后的跳转URL，仅在mode=redirect的时候生效。
- `spring.cloud.sentinel.scg.fallback.response-body`：限流之后的返回内容，仅在mode=response的时候生效。
- `spring.cloud.sentinel.scg.fallback.response-status`：限流之后的返回 `status code`，仅在mode=response的时候生效。

除此之外，您也可以在GatewayCallbackManager上通过setBlockHandler注册函数实现自定义的逻辑处理被限流的请求，对应接口为 `BlockRequestHandler`，编写逻辑可参考 `DefaultBlockRequestHandler` 默认实现类。

> **说明** YAML文件请注意转成YAML配置的形式。

Zuul 1.x

若您的网关是Zuul 1.x，则默认的限流处理逻辑是返回默认的流控文本，返回 `status code` 为 `429 Too Many Requests`。

您可以通过注册回调的方式定制处理异常，示例如下。

```
// 自定义FallbackProvider。
public class MyBlockFallbackProvider implements ZuulBlockFallbackProvider {
    @Override
    public String getRoute() {
        // 对应的route或API group。
        return "book-service";
    }
    @Override
        public BlockResponse fallbackResponse(String route, Throwable cause) {
            if (cause instanceof BlockException) { // 流控、降级、系统保护异常。
                return new BlockResponse(429, "Blocked by AHAS Sentinel", route);
            } else {
                return new BlockResponse(500, "System Error", route);
            }
        }
 }
 // 注册FallbackProvider。
 ZuulBlockFallbackManager.registerProvider(new MyBlockFallbackProvider());
```

4.多语言服务治理

4.1. 查看应用详情

您可以通过MSE治理中心控制台查看已接入MSE治理中心的多语言应用详情。

前提条件

服务网格微服务应用接入MSE治理中心

查看微服务应用列表

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
3. 在**应用列表**页面查看已开启微服务中心的应用的相关信息，包括**应用名称**、**接入方式**和**实例数量**。

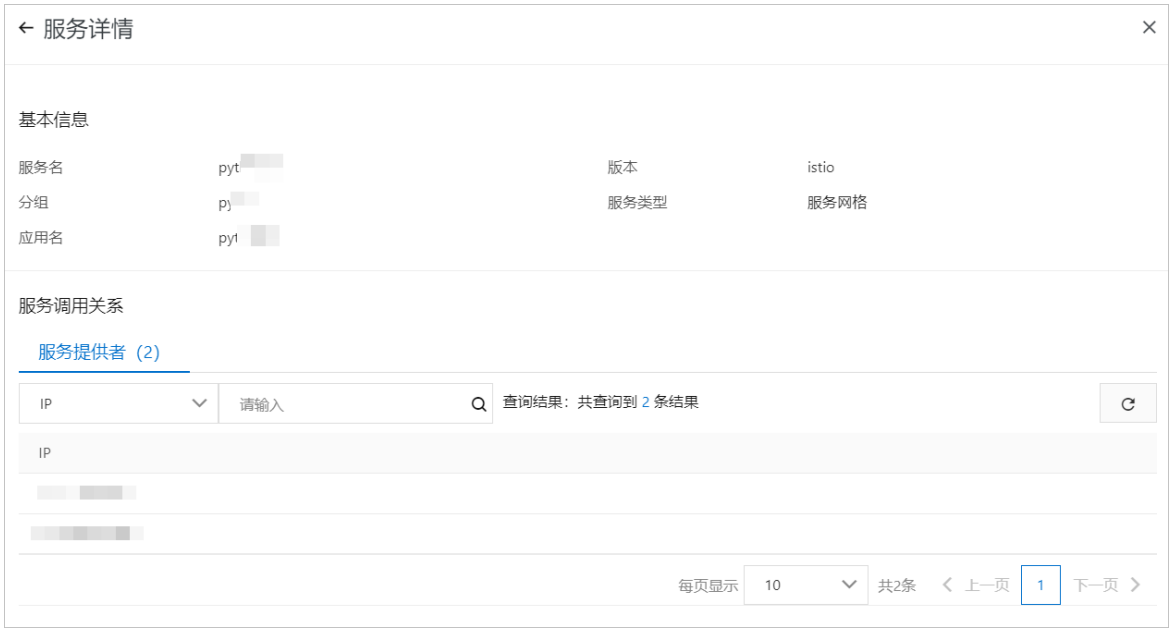
> **说明** 接入方式为ASM的应用即为多语言应用。

4.2. 查询服务

您可以通过MSE治理中心查询部署的Istio应用的服务列表和服务详情。

查询服务列表

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 服务查询**。
3. 在**请选择**文本框中选择**框架：Istio**，然后在服务列表中单击具体服务名，可以在**服务详情**页面查看服务的详细信息。
4. 在**服务详情**页面查看服务的详细信息。



4.3. 配置标签路由

标签路由通过标签将一个或多个服务的提供者划分到同一个分组，从而约束流量只在指定分组中流转，实现流量隔离的目的。标签路由可以作为蓝绿发布、灰度发布等场景的能力基础。本文介绍MSE控制台上新版本和旧版本的标签路由配置。

创建标签路由（新版本）

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量配置 > 标签路由**。
3. 单击**标签路由**右侧的**新版本**，在**标签路由**页面找到需要打标签的应用，单击目标标签**流量规则**下方的**添加**。
4. 在**创建标签路由**面板中配置相关参数，然后单击**确定**。

   路由标签参数说明如下。

参数	描述
路由名称	标签路由规则名称，例如 `test-Istio` 。
应用	显示所选择的应用名称。
标签	显示所选择的路由标签。
应用实例	显示应用实例的IP。
是否链路传递	如果需要使用全链路流控，请打开是否链路传递开关。
流量规则	
框架类型	选择**服务网格**。
Path	输入HTTP相对路径。
条件模式	包含**同时满足下列条件**和**满足下列任一条件**，根据实际需求选择。
条件列表	支持Header类型的参数设置。

5. 配置好流量规则后，在**标签路由**页面单击目标应用右侧的**流量分配**。
6. 设置各标签路由的流量比例，然后单击**保存**。

> **注意** 各流量比例配置总和为100%。

创建标签路由（旧版本）

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 流量配置 > 标签路由**。

3. 单击**标签路由**右侧的**旧版本**。
4. 在左侧导航栏选择**微服务治理中心 > 流量配置 > 路由标签**，单击**标签路由**右侧的**旧版本**，在**标签路由**页面单击**创建标签路由**。
5. 在**创建标签路由**面板中配置相关参数，然后单击**确定**。

   路由标签参数说明如下。

参数	描述
路由名称	标签路由规则名称，例如 `test-Istio` 。
描述	规则描述。
应用	选择您的应用名称。
流量类型	目前默认支持比例路由。
框架类型	支持服务网格。
流量比例	支持手动配置各标签比例。 **注意** 各流量比例配置总和为100%。

结果验证

本文仅通过一个示例介绍如何为应用创建标签路由，您可以为应用参照配置，然后根据实际业务需求进行验证。

4.4. 配置服务鉴权

当您的某个微服务应用有安全要求，不希望其他所有应用都能调用时，可以对调用该应用的其他应用进行鉴权，仅允许匹配鉴权规则的应用调用。

创建服务鉴权规则

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 安全治理 > 服务鉴权**。
3. 在**服务鉴权**页面单击**创建规则**。
4. 在**创建规则**面板中设置服务鉴权参数，然后单击**确定**。

   服务鉴权规则参数说明：

参数	说明
规则名称	鉴权规则名称，支持大小写字母、数字、下划线（_）和短划线（-），长度不超过64个字符。
被调用方类型	根据实际情况选择*应用*或*K8s Namespace*。 ○ **被调用方（应用）**：当**被调用方类型**选择**应用**时，选择被调用的应用。 ○ **被调用方（K8s Namespace）**：当**被调用方类型**选择**K8s Namespace**时，选择被调用的应用集群和所在的命名空间。

参数	说明
被调用方框架	被调用的应用所使用的框架，选择**服务网格**。
添加所有接口规则（适用于被调用方为应用的类型） 注意 所有接口的通用规则仅支持添加一次。	
被调用方接口	默认为**所有Path**，不可设置。
鉴权方式	默认为**黑名单（拒绝调用）**不可设置。
调用方	需要鉴权的调用方应用，可以单击**添加调用方**设置多个需要鉴权的调用方应用。
添加指定接口规则 注意 指定接口添加的规则不是追加，而是覆盖针对所有接口的通用规则，请谨慎配置。	
被调用方Path	指定被调用应用的Path。
鉴权方式	默认为**黑名单（拒绝调用）**不可设置。
调用方	需要鉴权的调用方应用，可以单击**添加调用方**设置多个需要鉴权的调用方应用。
默认状态	规则的启用开关。 ○ 打开：创建后即启用，默认打开。 ○ 关闭：创建后不启用，如果需要启用，请在**服务鉴权**页面规则的**操作**列单击**开启**。

结果验证
服务鉴权规则配置完成且开启后，请根据实际业务验证服务鉴权规则是否生效。

相关操作
服务鉴权规则创建完成后，您还可以编辑规则、根据规则的不同状态关闭规则或开启规则。当不再需要服务鉴权时，删除规则。

4.5. 微服务测试

4.5.1. 测试多语言服务

在日常开发中，开发人员或测试人员需要临时调用线上服务来调试已经部署的服务或查询线上数据。服务测试功能可以让您在控制台填写调用参数、发起服务调用，并得到服务调用的结果。

前提条件
- 在使用服务测试前，请确保您的应用已接入MSE治理中心。具体操作，请参见微服务治理中心入门概述。
- 在使用服务测试前，您需要完成RAM授权。具体操作，请参见为RAM用户授予MSE微服务治理中心的操作权限。

- 如果您使用RAM用户测试服务，请先在RAM中配置服务测试相关权限。具体操作，请参见权限配置示例。

操作步骤

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务测试**。
3. 在顶部菜单栏选择**地域**，在框架类型列表中选择**框架：Istio**，然后单击目标服务名称或**操作**列的**测试**按钮。
4. 在**选择测试方法**面板中设置测试相关参数，然后单击**执行**。

   测试服务参数说明如下。

参数	描述
调用IP	要测试服务的实例IP。如果部署了多个实例，在列表中选择其中一个IP，进行测试，只能单选。
Path	请求的接口URL，以 `/` 开头，例如 `/reviews/2` 。
请求方法	该所属类的请求方法，如果包含多个请求方法，在列表中选择其中一种方法，只能单选。
测试参数	在测试方法的参数区域，根据服务的代码设置方法的具体参数。

执行结果

在**结果**区域查看测试是否成功，测试结果一般会有以下几种情况：

- 结果成功，并显示调用服务的响应结果。
- 结果失败，并显示调用服务的失败响应信息。请根据响应信息，排查服务的端口、网络及代码本身的问题。

4.5.2. 压测多语言服务

在日常开发中，开发人员或测试人员需要评估服务的性能是否符合预期，避免因功能迭代导致服务性能下降而引发故障。服务压测功能可以让您低成本地评估服务性能，做到1分钟创建压测场景，5分钟获取性能指标。

前提条件

在使用服务测试前，请确保您的应用已接入MSE治理中心。具体操作，请参见微服务治理中心入门概述。

背景信息

在大促活动中，应该准备多少实例资源才能满足大促吞吐量的要求，降低因大促活动带来的访问量暴增进而引发系统宕机的风险。此时需要合理地评估服务性能，避免流量冲击引发的故障，并降低运营使用成本。

创建压测场景

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务压测**。
3. 在顶部菜单栏选择**地域**，然后单击**创建场景**。
4. 在**创建场景**面板中设置**场景配置**和**压力配置**相关参数，然后单击**确定**。

   场景配置页签相关参数说明如下。

参数	描述
场景名称	自定义压测场景名称，例如test-Istio。
应用	选择需要压测的应用。
框架类型	选择**服务网格**框架。
Path	输入HTTP相对路径，例如*/getIp*。
基本信息	设置请求方式，包括GET/POST/PUT/DELETE。 **说明** GET和DELETE只支持修改URL的Path路径。POST和PUT支持ContentType及参数编写格式。
请求头	设置请求头参数信息。关于多语言微服务支持的ContentType类型，请参见多语言参考示例。
参数模式	选择请求参数输入模式。 ○ *参数填写*：填写参数内容。关于参数填写格式，请参见参数文件URL。 ○ *参数文件路径*：填写参数文件地址。参数文件路径必须支持公网可访问。 ○ *参数文件上传*：单击**上传文件**，选择参数文件进行上传。
超时时间	设置超时时长，单位：毫秒。
直连服务	开启后选择**服务地址**，可直接连接该服务。
打印日志	开启可自动打印日志信息，但会影响到服务压测性能，建议正常压测时关闭。

压力配置页签相关参数说明如下。

参数	描述
压测模式	压测模式有两种：并发模式（虚拟用户模式）、TPS模式（Transaction Per Second，吞吐量模式）。 ○ **并发模式**：指虚拟并发用户数，从业务角度，也可以理解为同时在线的用户数。 ○ **TPS模式**：指系统每秒处理的事务数量。

参数	描述
流量模型	流量模型包括固定压力、阶梯压力和脉冲压力。 ○ **固定压力**：以配置的固定并发值进行施压，并可设置预热时长。 ○ **阶梯压力**：设置最大值、最小值、预热时间等信息，在预热递增期间，从最小值开始按照阶梯逐步递增，达到最大并发后按照最大并发持续施压。不可指定循环次数。 ○ **脉冲压力**：设置峰值、谷值以及持续时间等信息，施压流量以峰值、峰谷的锯齿波的形式进行施压。
压测时长（分钟）	指压测总时长，公测期间最大压测时长60分钟。
预热时长（分钟）	施压前的预热时间，若设置为0，则表示无需预热。

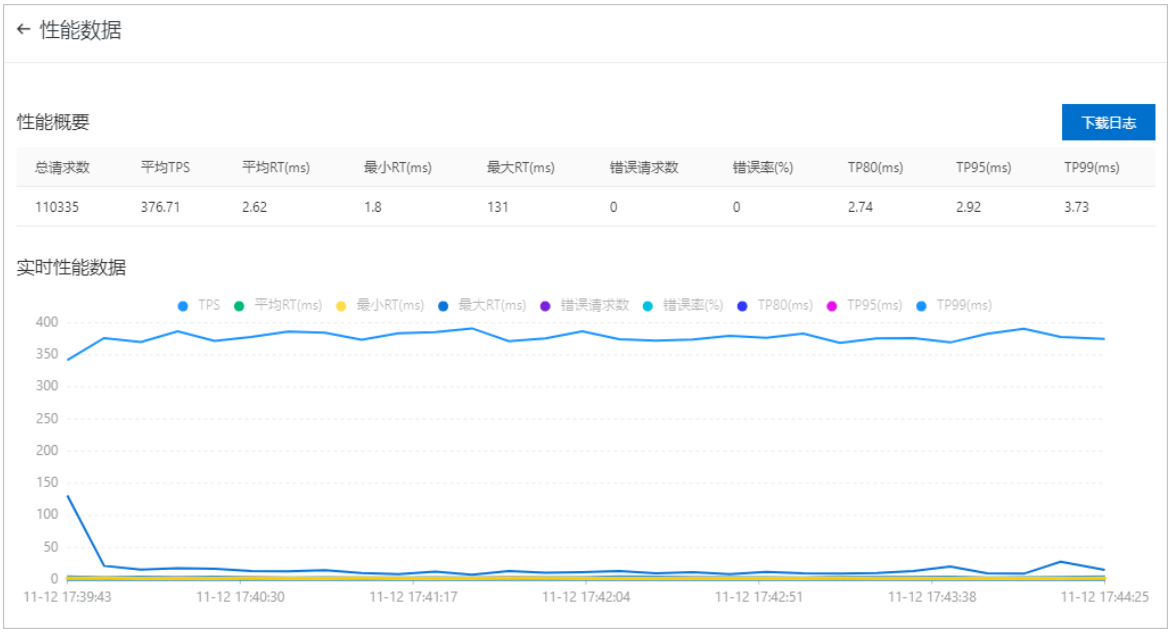
执行结果

压测场景创建成功后，您可查看**服务压测**列表查看相关信息，包括**平均TPS**、**平均响应时间**、**错误率**等。

您可在压测场景**详情**页面选择**重启**、**停止**或**编辑场景**，也可执行**删除**等操作。

查看压测报告

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务压测**。
3. 在服务压测列表中单击**操作**列的**详情**，查看**场景配置**和**运行记录**。
4. 在**运行记录**区域单击**操作**列的**详情**，查看实时性能数据。



> **说明** **性能数据**是每10秒的所有施压机数据统计，具体根据压测总时间长度会有所变化。单击图上方的图例，可以显示或隐藏某些数据曲线。

参数	说明
总请求数	整个压测过程中，共发起的请求个数。
平均TPS	压测周期内，所有压力机发出的平均TPS值，TPS=调用总次数/总运行时间。
平均RT（ms）	所有压力机发出平均响应时间。
最小RT（ms）	所有压力机中最小的一次响应时间。
最大RT（ms）	所有压力机中最大的一次响应时间。
错误请求数	所有压力机中错误请求数之和。
错误率（%）	所有压力机中的平均错误率。
TP80（ms）	所有压力机中80分位（P80）的平均值。
TP95（ms）	所有压力机中95分位（P95）的平均值。
TP99（ms）	所有压力机中99分位（P99）的平均值。

5. 单击**下载日志**，可获取压测过程中的日志。

多语言参考示例

ContentType	参数编写格式
application/x-www-form-urlencoded	[{"name": "cart"},{"age": 20}]
application/json（默认）	{"name": "cart", "age": 20}

参数文件URL：提供一个公网可下载的文件地址

平台会把该参数文件分发到每一个施压机，应用每一次调用参数就在该文件中按顺序读取一行。文件里面也支持动态函数参数。

参数文件填写格式如下：

- 方法参数类型：填写的内容是一个字符串类型的JSON数组，数组的每一位代表对应位置的参数类型。除了Java基本类型，其余类型需要填写完整的类路径。
- 方法参数：填写的内容是一个字符串类型的JSON数组，数组的每一位代表对应位置的参数。

参数编写格式	参数文件内容编写格式
application/x-www-form-urlencoded	[{"name": "cart"},{"age": 20}] [{"name": "cart"},{"age": 20}] [{"name": "cart"},{"age": 20}]
application/json	{"name": "cart", "age": 20} {"name": "cart", "age": 20} {"name": "cart", "age": 20}

4.5.3. 压测多语言服务（新版控制台）

在日常开发中，开发人员或测试人员需要评估服务的性能是否符合预期，避免因功能迭代导致服务性能下降而引发故障。服务压测功能可以让您低成本地评估服务性能，做到1分钟创建压测场景，5分钟获取性能指标。

前提条件

在使用服务测试前，请确保您的应用已接入MSE治理中心。具体操作，请参见微服务治理中心入门概述。

背景信息

在大促活动中，应该准备多少实例资源才能满足大促吞吐量的要求，降低因大促活动带来的访问量暴增进而引发系统宕机的风险。此时需要合理地评估服务性能，避免流量冲击引发的故障，并降低运营使用成本。

创建压测场景

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务压测**。
3. 在顶部菜单栏选择地域，然后单击**创建场景**。
4. 在**创建场景**面板中设置**场景配置**、**压力配置**和**数据配置**等相关参数。
   i. 在**场景配置**页签单击右侧的 ⌄ 图标，设置如下相关参数。

参数	描述
场景名称	自定义压测场景名称，例如test-Istio。
应用	选择需要压测的应用。
框架类型	选择**服务网格**框架，系统会自动识别应用类型。
Path	输入HTTP相对路径，例如*/getIp*。
基本信息	设置**请求方法**和**ContentType**。其中请求方法包括GET/POST/PUT/DELETE，ContentType包括x-www-form-urlencoded和raw，不同的ContentType提供不同可视化的参数输入方式： ■ **x-www-form-urlencoded**：表单输入，传递的参数格式为[{"name": "cart"},{"age": 20}]。 ■ **raw**：默认为application/json JSON格式输入，传递的参数格式为{"name": "cart", "age": 20}。其他格式输入，传递的参数格式按输入文本的传输。 **说明** GET和DELETE只支持修改URL的Path路径。POST和PUT支持ContentType及参数编写格式。 关于多语言微服务支持的ContentType类型，请参见多语言参考示例。
请求头	设置请求头参数信息。
断言（选填）	输入**检查对象**和**检查内容**，选择**检查条件**。
出参提取（选填）	输入**出参名**和**出参提取表达式**。
打印日志	开启可自动打印日志信息，但会影响到服务压测性能，建议正常压测时关闭。

ii. 在**压力配置**页签设置如下相关参数。

参数	描述
压测模式	压测模式有两种：并发模式（虚拟用户模式）、TPS模式（Transaction Per Second，吞吐量模式）。 ■ **并发模式**：指虚拟并发用户数，从业务角度，也可以理解为同时在线的用户数。 ■ **TPS模式**：指系统每秒处理的事务数量。
流量模型	流量模型包括固定压力、阶梯压力和脉冲压力。 ■ **固定压力**：以配置的固定并发值进行施压，并可设置预热时长。 ■ **阶梯压力**：设置最大值、最小值、预热时间等信息，在预热递增期间，从最小值开始按照阶梯逐步递增，达到最大并发后按照最大并发持续施压。不可指定循环次数。 ■ **脉冲压力**：设置峰值、谷值以及持续时间等信息，施压流量以峰值、峰谷的锯齿波的形式进行施压。
压测时长（分钟）	指压测总时长，公测期间最大压测时长60分钟。
预热时长（分钟）	施压前的预热时间，若设置为0，则表示无需预热。

iii. 在**数据配置**页签单击**添加CSV数据**，在**CSV数据设置**对话框中设置如下相关参数，然后单击**确定**。

参数	描述
名称	自定义参数名称。
注释	自定义参数说明。
变量名称	输入CSV文件中的各个参数名称，多个变量名称之间使用英文逗号（,）进行分隔。
CSV文件	单击**上传文件**，选择本地CSV格式的参数文件进行上传。

> **说明**
> ■ MSE会把CSV参数文件分发到每一个施压机，应用调用会按参数文件中的顺序自动读取参数，您只需在CSV文件中设置参数顺序即可。
> ■ 参数文件仅支持.csv格式，且文件首行表示参数名称，实际压测请求调用时，默认自动忽略首行。

5. 在**场景配置**的**基本信息**配置区域，通过*${XXX}*格式引用定义的参数变量名称，然后单击**确定**。

> **说明** XXX为**数据配置**中添加的**变量名称**，需要用*${}*格式进行引用，例如：*${param}*。

创建多步骤串联的压测场景

> **说明** 一个压测场景可以包含多个压测步骤，当后续的压测步骤依赖先前的压测步骤的输出时，需要使用参数传递。

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务压测**。
3. 在顶部菜单栏选择地域，单击目标用例右侧**操作**列的**详情**。
4. 在**场景详情**面板中单击**编辑场景**，然后在**编辑场景**面板中选择**场景配置**页签，执行以下操作：
   i. 单击目标压测步骤右侧的**访问一次**，弹出**单步骤调试结果**。
   您可查看此次请求**接口信息**、**请求入参**和**请求出参**。
   ii. 在**单步骤调试结果**面板中单击**出参提取助手**，弹出**出参提取助手**窗口，选择需要提取的出参参数进行复制，然后单击**确定**。
   弹出提示内容：*${XXX}*参数已写入剪切板，请直接粘贴。
   iii. 在**出参提取（选填）**下方的**出参提取表达式**中粘贴所选择的出参表达式，并自定义**出参名**。
   iv. 单击**添加下一步骤**添加多个压测步骤。
   v. 在该压测步骤的**基本信息**区域引用变量值，格式为*${XXX}*。

   > **说明** XXX为前序步骤的**出参提取**中自定义的**出参名**，需要用*${}*格式进行引用。

5. 在**编辑场景**面板中选择**压力配置**页签，配置压力数据，然后单击**确定**。
   多步骤串联的压测场景配置完成。

执行结果

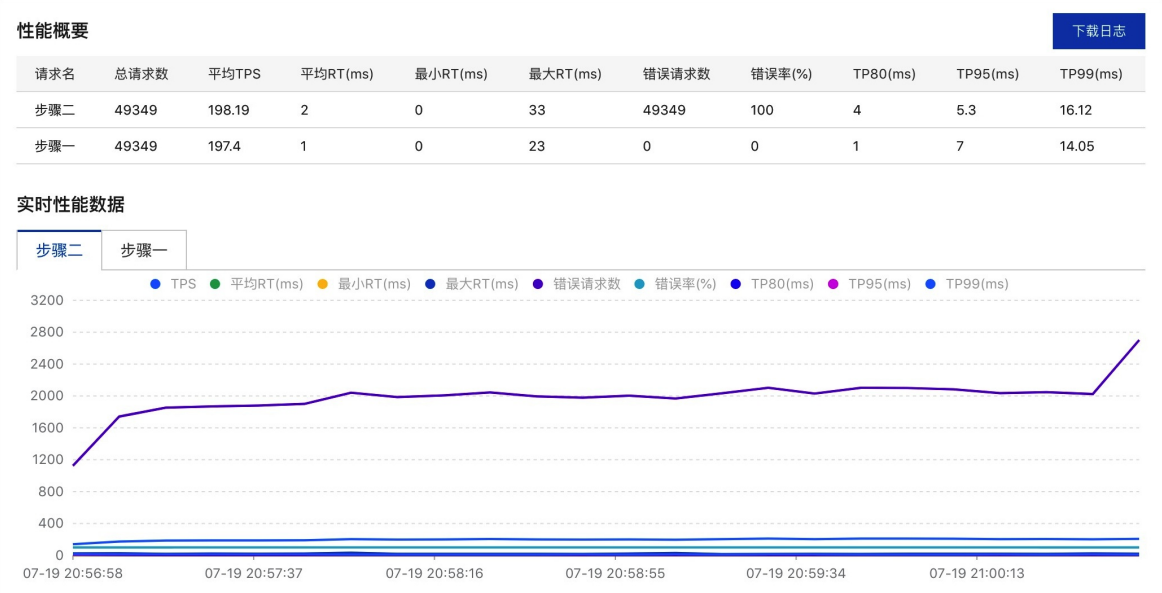
压测场景创建成功后，您可查看**服务压测**列表查看相关信息，包括**平均TPS**、**平均响应时间**、**错误率**等。

您还可以执行以下相关操作管理压测场景：

- 查看压测详情：在**服务压测**列表页面，单击**操作**列的**详情**，在压测**场景详情**页面执行**启动**、**停止**或**编辑场景**等操作。
- 复制压测场景：在**服务压测**列表页面，单击**操作**列的**复制**，可生成一个新的压测场景。
- 删除压测场景：在**服务压测**列表页面，单击**操作**列的**删除**，可删除该压测场景。

查看压测报告

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务压测**。
3. 在顶部菜单栏选择地域，在服务压测列表中单击**操作**列的**详情**，查看**场景配置**和**运行记录**。
4. 在**运行记录**区域单击**操作**列的**详情**，查看实时性能数据。



> **说明** **性能数据**是每10秒的所有施压机数据统计，具体根据压测总时间长度会有所变化。单击图上方的图例，可以显示或隐藏某些数据曲线。

参数	说明
总请求数	整个压测过程中，共发起的请求个数。
平均TPS	压测周期内，所有压力机发出的平均TPS值，TPS=调用总次数/总运行时间。
平均RT（ms）	所有压力机发出平均响应时间。
最小RT（ms）	所有压力机中最小的一次响应时间。
最大RT（ms）	所有压力机中最大的一次响应时间。
错误请求数	所有压力机中错误请求数之和。
错误率	所有压力机中的平均错误率。
TP80（ms）	所有压力机中80分位（P80）的平均值。
TP95（ms）	所有压力机中95分位（P95）的平均值。
TP99（ms）	所有压力机中99分位（P99）的平均值。

5. 单击**下载日志**，可获取压测过程中的日志。

多语言参考示例

ContentType	参数编写格式
x-www-form-urlencoded	在表单中以key-value对的方式填入，传递的参数格式：[{"name": "cart"},{"age": 20}]。

ContentType	参数编写格式
raw	• JSON(application/json)：JSON字符串，如：{"name": "cart", "age": 20}。 • XML(application/xml)：Application/XML类型的XML字符串。 • XML(text/html)：TEXT/XML类型的XML字符串。 • HTML(text/html)：HTML字符串。 • JavaScript(application/javascript)：JavaScript字符串。 • Text(text/plain)：纯文本格式的编码形式（TEXT/XML/HTML）。

4.5.4. 巡检多语言服务

目前，随着云原生技术的推广和普及，微服务化已成为趋势。但线上微服务接口可靠性却并不完善，无法实时感知异常，存在较大风险。本文介绍微服务巡检平台的相关操作，帮助您随时了解API或微服务接口的运行情况，降低服务风险。

背景信息

云原生时代下应用微服务化是趋势，但企业如何保障线上微服务的可靠性，主动感知线上微服务异常，降低业务风险呢？微服务巡检帮助您对线上服务进行7*24小时的秒级探测，实时了解服务的健康度，且当服务异常时及时告警，尽快恢复，降低损失。

创建巡检服务

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务巡检**。
3. 在顶部菜单栏选择**地域**，然后在**服务巡检**页面单击**创建巡检任务**。
4. 在**创建巡检任务**面板中设置相关参数，然后单击**确定**。

参数	描述
服务巡检任务名称	自定义服务巡检任务名称。
应用	选择需要巡检的应用。
框架类型	支持Spring Cloud、Dubbo和服务网格框架。系统会根据所选应用自动识别其框架，也可以手动选择**服务网格**。
Path	输入HTTP相对路径，例如*/getIp*。
基本信息	设置请求方式，包括GET/POST/PUT/DELETE。 **说明** GET和DELETE只支持修改URL的Path路径。POST和PUT支持ContentType及参数编写格式。
请求头	设置请求头参数信息。关于Spring Cloud微服务支持的ContentType类型，请参见多语言参考示例。

参数	描述
断言信息包含	设置接口返回值信息。如果返回值含有一个特征，如返回值含有123，则格式为"123"；如果返回值含有多个特征，如同时含有123，abc，则格式为["123","abc"]。
巡检周期	设置巡检周期，单位秒/分钟，可自定义选择。
报警触发条件	当接口巡检异常时，告警触发的频率。
报警接收管理	接收告警的联系人组。在左侧列中选中需要接手告警的联系人组，并单击>，添加到右侧列表中。
报警通知方式	报警通知方式包含**钉钉**、**短信**和**邮件**。

服务巡检任务创建成功后，返回**服务巡检**列表，查看相关信息，包括**巡检次数**、**可用率**、**平均响应时间**等。

相关操作

您还可以执行以下操作管理服务巡检。

- 任务运行：在服务巡检列表页面，单击**操作**列的**启动**，可重新启动该服务巡检任务。
- 更新配置：在服务巡检列表页面，单击**操作**列的**详情**，可重新编辑服务巡检任务。
- 暂停服务：在服务巡检列表页面，单击**操作**列的**暂停**，可暂停该服务巡检任务。
- 查看失败记录：在服务巡检列表页面，单击**操作**列的**失败记录**，可查看该服务巡检的监控详情。

多语言参考示例

ContentType	参数编写格式
application/x-www-form-urlencoded	[{"name": "cart"},{"age": 20}]
application/json（默认）	{"name": "cart", "age": 20}

4.5.5. 自动化回归多语言服务测试用例

自动化回归功能基于服务契约信息快速编排被测服务、管理自动化测试用例，帮助您高效管理、回归业务测试场景，完成业务快速验证和交付。

前提条件

在使用服务测试前，请确保您的应用已接入MSE治理中心。具体操作，请参见微服务治理中心入门概述。

创建多语言测试用例

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选择**地域**，然后单击**创建用例**。
4. 在**创建用例**页面单击测试步骤右侧的 ，然后设置相关参数信息。

参数	描述
用例名称	自定义测试用例名称。
步骤名称	自定义测试步骤名称。
应用	选择需要测试的应用。
框架类型	选择**服务网格**框架。
Path	输入HTTP相对路径，例如*/getIp*。
基本信息	设置请求方法和ContentType。请求方法包括GET/POST/PUT/DELETE，ContentType包括x-www-form-urlencoded和raw，不同得ContentType提供不同可视化的参数输入方式。 ○ **x-www-form-urlencoded**：表单输入，传递的参数格式为[{"name": "cart"},{"age": 20}]。 ○ **raw**：默认为application/json JSON格式输入，传递的参数格式为{"name": "cart", "age": 20}。其他格式输入，传递的参数格式按输入文本的传输。 关于多语言微服务支持的ContentType类型，请参见多语言参考示例。
请求头	设置请求头参数信息。
断言（选填）	输入**检查对象**和**检查内容**，选择**检查条件**。
出参提取（选填）	输入**出参名**和**解析表达式**。
（可选）**高级设置**	
用例描述	自定义测试用例描述。
加入用例集	选择需要加入的用例集。若没有用例集，可单击右侧的**创建用例集**进行创建。

5. 单击右侧的**访问一次**，弹出**单步骤调试结果**，查看此次请求入参和请求出参。
6. 单击**出参提取助手**，弹出**出参提取助手**对话框，再单击需要提取的出参名，复制该参数。
7. 在**断言（选填）**下方的**检查对象**中粘贴所复制的参数，选择**检查条件**，输入**检查内容**。
8. 在**出参提取（选填）**下方的**出参提取表达式**中粘贴所复制的参数，并自定义**出参名**。
9. 单击右上方的**保存配置**即可。
   您可在用例列表中查看创建的测试用例。

创建多步骤串联的测试用例

> **说明** 一个测试用例可以包含多个测试步骤，当后序的测试步骤依赖前序的测试步骤的输出时，需要使用参数传递。

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选中**地域**，单击目标用例右侧**操作**列的**详情**。

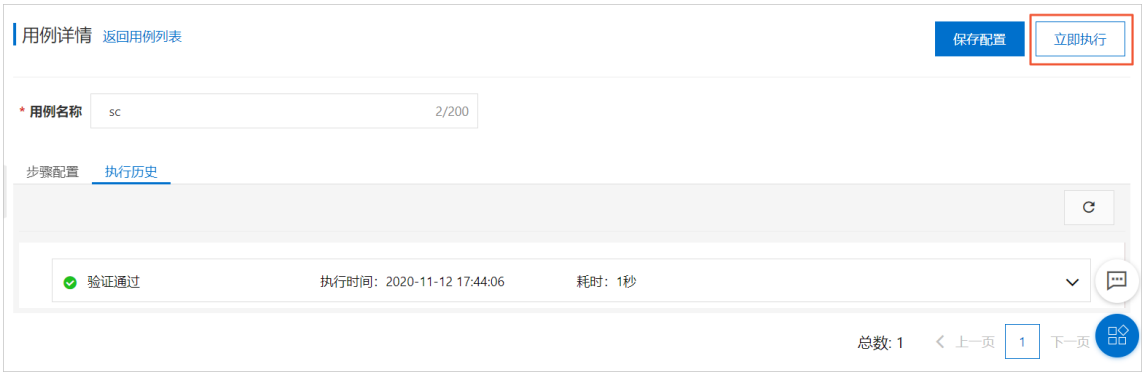
4. 在**用例详情**页面单击右侧的**访问一次**，弹出**单步骤调试结果**，查看此次请求入参和出参。
5. 单击**出参提取助手**，弹出**出参提取助手**窗口，选择需要提取的出参参数进行复制。
6. 在**出参提取（选填）**下方的**出参提取表达式**中粘贴所选择的出参表达式，并自定义**出参名**。
7. 单击**添加下一步**增加多个测试步骤。
8. 在该测试步骤的**基本信息**区域，ConetntType选中raw，在JSON格式化中输入引用变量名**${XXX}**。

   > **说明** XXX为前序步骤的**出参提取**中设置的**出参名**，需要用${}格式进行引用。

9. 单击右上方的**保存配置**，再单击**执行用例**。

执行测试用例

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选择**地域**。
4. 您可选择以下两种方式执行测试用例。
   - 在**用例列表**页面，单击目标用例右侧**操作**列的**执行**。
   - 在**用例列表**页面，单击目标用例右侧**操作**列的**详情**，在**用例详情**页面单击**立即执行**。



您可在**执行历史**页签中查看详细执行结果。

相关操作

您还可以执行以下操作管理测试用例。

- 复制测试用例：在**自动化回归**列表页面，单击**操作**列的**复制**，可生成一条新的测试用例。
- 删除测试用例：在**自动化回归**列表页面，单击**操作**列的**删除**，可删除该测试用例。

多语言参考示例

ContentType	参数编写格式
x-www-form-urlencoded	在表单中以key-value对的方式填入，传递的参数格式：[{"name": "cart"},{"age": 20}]

ContentType	参数编写格式
raw	• JSON(application/json)：JSON字符串，如：{"name": "cart", "age": 20}。 • XML(application/xml)：Application/XML类型的XML字符串。 • XML(text/html)：TEXT/XML类型的XML字符串。 • HTML(text/html)：HTML字符串。 • JavaScript(application/javascript)：JavaScript字符串。 • Text(text/plain)：纯文本格式的编码形式（TEXT/XML/HTML）。

4.5.6. 自动化回归多语言服务测试用例集

自动化回归测试用例集功能通过关联测试用例，帮助您快速完成业务验证和交付。

前提条件

在使用服务测试前，请确保您的应用已接入MSE治理中心。具体操作，请参见微服务治理中心入门概述。

创建多语言测试用例集

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 微服务测试 > 自动化回归（用例集）**。
3. 在顶部菜单栏选择**地域**，然后单击**创建用例集**。
4. 在**创建用例集**面板中输入用例集名称，单击**确定**。
   您可在**自动化回归（用例集）**页面查看所创建的用例集。

关联多语言测试用例

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 微服务测试 > 自动化回归（用例集）**。
3. 在顶部菜单栏选中**地域**，单击目标用例集右侧**操作**列的**详情**。
4. 在**用例集详情**页面单击**关联用例**。

5. 在**关联用例**面板中选中关联的用例，单击**确定**。
   您可在**用例列表**中查看所关联的用例。
6. （可选）若您想取消关联用例，在**用例列表**中单击**操作**列的**取消关联**。

执行测试用例集

1. 登录MSE治理中心控制台。

2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例集）**。
3. 在顶部菜单栏选择地域。
4. 您可选择以下两种方式执行测试用例集。
   - 在用例集列表页面，单击目标用例集右侧**操作**列的**执行**。
   - 在用例集列表页面，单击目标用例集右侧**操作**列的**详情**，在**用例集详情**页面单击**执行用例集**。

   您可在**执行历史**页签中查看详细执行结果。

相关操作

您还可以执行以下操作管理测试用例集。

- 复制测试用例：在**用例集详情**页面的**用例列表**页签中，单击**操作**列的**复制**，可生成一条新的测试用例。
- 删除测试用例集：在**自动化回归（用例集）**列表页面，单击**操作**列的**删除**，可删除该测试用例集。

4.6. 金丝雀发布

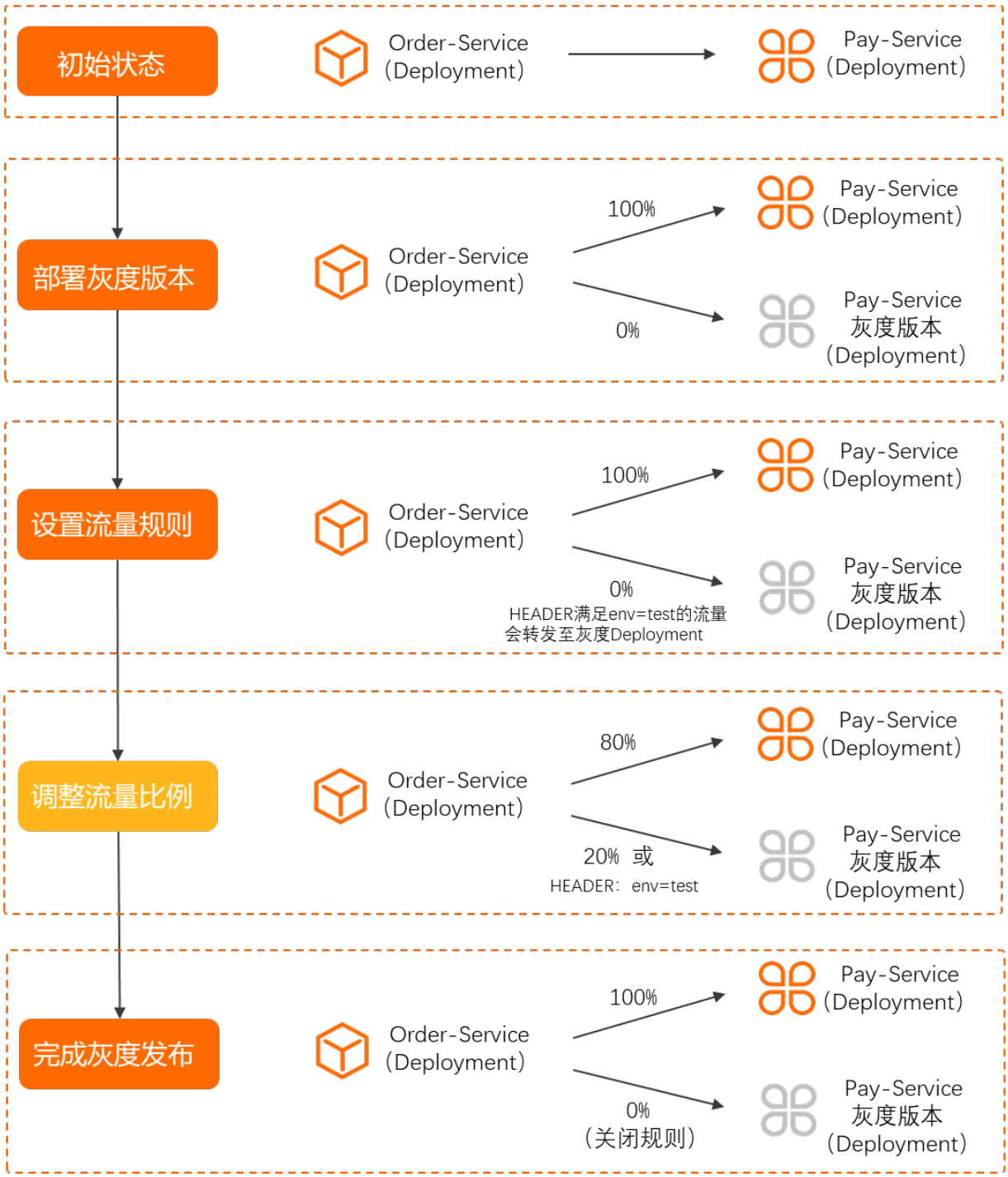
部署在阿里云容器服务ACK集群、ASK集群、自建并注册ACK集群中的多语言微服务应用，为了确保其升级的安全性，可以使用金丝雀发布（即灰度发布）进行小规模验证，验证通过后再全量升级。

前提条件

已为ACK授予MSE治理中心的访问权限，并在ASM中安装MSE治理中心组件。具体操作，请参见服务网格微服务应用接入MSE治理中心。

背景信息

金丝雀发布的过程如下：



1. **初始状态**：例如存在2个服务Order-Service和Pay-Service，Order-Service作为Consumer服务会去调用Pay-Service提供的服务。在您没有接入MSE治理中心之前，Order-Service和Pay-Service都对应一个Deployment应用，为Pay-Service设置版本号，即Deployment的Pod配置中添加label标签，如 `version:v1` 。
2. **部署灰度版本**：为Pay-Service部署灰度版本，此时Pay-Service新建一个Deployment应用表示灰度版本，并设置标签，如 `version:v2` 。
3. **设置流量规则**：为Pay-Service设置流量规则：HTTP HEADER中 `env=test` 。此时满足该规则的流量会转发至灰度Deployment应用，其他流量转发至正常Deployment应用。
4. **调整流量比例**：为灰度版本调整流量比例，让更多的流量转发至灰度Deployment应用。

> **注意** 满足流量规则的流量还是会转发至灰度Deployment应用，不满足流量规则的流量有20%的概率转发至灰度Deployment应用。

5. **完成灰度发布**：灰度验证完毕，此时100%的流量转发至灰度Deployment应用。

> **说明** 建议您删除正常的Deployment应用或者将正常的Deployment应用副本数改为0。

使用限制

金丝雀发布功能只适合在两个标签的场景下使用，若有多个标签的场景，请使用标签路由。具体操作，请参见配置标签路由。

操作步骤

本文主要介绍的是配置流量规则的步骤，其他步骤请根据实际业务需求完成。

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
3. 选择具体应用，单击**操作**列的**金丝雀**。

   或者单击具体应用名称，在**应用详情**页面选择**服务治理**区域，单击**金丝雀**页签。
4. 单击**应用实例**右侧的**编辑**，在**修改金丝雀路由**面板中的**流量规则**区域配置流量规则相关参数，然后单击**确定**。

   流量规则配置相关参数说明如下。

参数	描述
框架类型	根据实际应用自动生成相应的框架类型。
Path	输入HTTP相对服务路径。
条件模式	包含**同时满足下列条件**或**满足下列任一条件**，根据实际需求选择。
条件列表	设置条件参数，当有多个条件规则时，可通过单击**添加新的规则条件**添加。 多语言微服务应用仅支持设置Header类型的参数。

   路由规则设置完成后，访问的请求带上规则里的参数，这时流量会去访问灰度Deployment应用。
5. 在**应用实例**区域中的**流量比例**列配置流量百分比，然后单击**确定**。

> **注意** 流量规则验证成功后，再调大灰度版本流量比例，建议逐渐调大灰度版本的流量比例。

执行结果

金丝雀验证成功：单击**发布完成**按钮，未打标版本的流量比例会被调整为100%，配置的流量规则会被暂时关闭。此时所有的流量都会被转发到未打标的Deployment应用。

> **说明** 发布完成后，建议您将灰度的Deployment应用副本数设置为0，无需重复创建Deployment应用。

金丝雀验证失败：单击**回滚**按钮，灰度版本的流量比例会被调整为0%，配置的流量规则会被清除。此时所有的流量都会被转发到正常Deployment应用，然后可以删除灰度版本的应用。

4.7. 配置负载均衡

负载均衡是服务治理中至关重要的能力，MSE为多语言服务提供了多种类型的负载均衡能力，本文介绍如何为多语言应用配置负载均衡规则。

创建负载均衡规则

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
3. 在顶部菜单栏选择地域，然后单击具体应用的名称。
4. 在**应用详情**页面选择**负载均衡**页签，然后单击**暂无数据，立即创建**。
5. 在**创建**面板中配置相关参数，然后单击**确定**。

   负载均衡规则的参数说明：

参数	描述
规则名称	负载均衡规则的名称。支持大小写字母、数字、下划线（_）和短划线（-），不超过64个字符。
描述	负载均衡规则的描述。
类型	支持**简单**和**一致性哈希**类型。
若负载均衡规则选择**简单**类型，则配置以下参数：	
配置	流量调度机制支持随机、轮询和最小连接数。 ○ **随机**：随机调度流量到所有的实例。 ○ **轮询**：按顺序调度流量到所有实例。 ○ **最小连接数**：优先调度流量到连接数较少的实例。
若负载均衡规则选择**一致性哈希**类型，则配置以下参数：	

参数	描述
哈希类型	支持源地址哈希、Header哈希、Cookie哈希和Query参数哈希。 ○ **源地址哈希**：根据源地址中的内容获取哈希。 **源地址**：您可选择是否将流量按照请求源IP地址的哈希值进行调度。 ○ **Header哈希**：将以HTTP请求中的Header参数计算哈希，哈希相同的请求将会转发至同一个实例进行处理。 **Header**：您需要输入Header中对应的参数的Key的值。 ○ **Cookie哈希**：将以HTTP请求中的所有Cookie计算哈希，哈希相同的请求将会转发至同一个实例进行处理。 ■ **Cookie名称**：输入Cookie名称。支持大小写字母、数字、下划线（_）和短划线（-），不超过64个字符。 ■ **Cookie路径**：输入Cookie路径。 ■ **Cookie过期时间**：输入Cookie过期时间。 ○ **Query参数哈希**：将以HTTP请求中的Query参数计算哈希，哈希相同的请求将会转发至同一个实例进行处理。 **Query参数**：您需要输入Query参数。
默认状态	负载均衡规则的启用开关。 ○ 打开：创建后即启用，默认打开。 ○ 关闭：创建后不启用，如果需要启用，请在负载均衡页面目标规则的**操作**列单击**开启**。

负载均衡规则配置完成且开启后，请根据实际业务验证负载均衡规则是否生效。

相关操作

负载均衡规则创建完成后，您还可以**编辑**规则以及根据规则的不同状态**关闭**规则或**开启**规则。当不再需要负载均衡时，**删除**规则。

4.8. 配置故障注入

故障注入是一种模拟应用异常行为的技术，通过给应用注入特定故障，可以检测该应用的消费者处理异常情况的能力，从而提高系统的健壮性。本文介绍如何为多语言应用配置服务故障。

创建故障注入规则

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
3. 在顶部菜单栏选择地域，然后单击具体应用的名称。
4. 在**应用详情**页面选择**故障注入**页签，然后单击**创建规则**。
5. 在**创建故障注入规则**面板中配置相关参数，然后单击**确定**。

   故障注入规则的参数说明：

参数	描述
规则名称	故障注入规则的名称。例如：fault-example。
标签	选择服务的路由标签。 **说明** **默认**指所有未配置标签路由的标签集合。
框架类型	应用的框架类型，默认为**服务网格**。
流量来源	请求的发起方，即消费者应用，包括**全部应用**和**特定应用**。 **说明** 当选择特定应用时，只有请求的发起方在指定的应用集合中，才可能会根据设置的百分比触发故障。
特定应用	当流量来源选择特定应用时，设置目标应用。
故障类型	故障注入规则支持的故障类型，包括**异常类**和**延迟类**。
百分比	设置该应用的请求注入故障的百分比。
异常状态码	当故障类型选择**异常类**，设置触发异常类故障时返回的状态码，状态码有效范围为：200~599。
固定延迟时间	当故障类型选择**延迟类**，设置触发延迟类故障时延迟的时间，请求将在设置的延迟时间后继续发送，单位：毫秒。
默认状态	故障注入规则的启用开关。 ○ 打开：创建后即启用，默认打开。 ○ 关闭：创建后不启用，如果需要启用，请在故障注入页面目标规则的**操作**列单击**开启**。

故障注入规则配置完成且开启后，请根据实际业务验证故障注入规则是否生效。

相关操作

故障注入规则创建完成后，您还可以**编辑**规则以及根据规则的不同状态**关闭**规则或**开启**规则。当不再需要故障注入时，**删除**规则。

4.9. 配置服务超时

服务超时机制可以在请求的处理时间超过设置的时间时直接返回错误结果，减少消费者应用的等待时间。您可以在业务代码中通过硬编码的方式配置服务超时逻辑，但这种方式缺少灵活性，且与业务代码耦合，MSE治理中心可以在不修改业务代码的前提下为多语言应用提供配置服务超时的能力。

创建服务超时规则

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。

3. 在顶部菜单栏选择地域，然后单击具体应用的名称。
4. 在**应用详情**页面选择**服务超时**页签，然后单击**创建规则**。
5. 在**创建服务超时规则**面板中配置相关参数，然后单击**确定**。

   服务超时规则的参数说明：

参数	描述
规则名称	服务超时规则的名称。例如：timeout-example。
标签	选择服务的路由标签。 **说明** **默认**指所有未配置标签路由的标签集合。
框架类型	应用的框架类型，默认为**服务网格**。
流量来源	请求的发起方，即消费者应用，包括**全部应用**和**特定应用**。 **说明** 当选择特定应用时，只有请求的发起方在指定的应用集合中，才会触发超时响应逻辑。
特定应用	当流量来源选择特定应用时，设置目标应用。
超时响应时间	如果应用的处理时间超过了设定的超时响应时间，则直接返回超时错误，单位：毫秒。
默认状态	服务超时规则的启用开关。 ○ 打开：创建后即启用，默认打开。 ○ 关闭：创建后不启用，如果需要启用，请在服务超时页面目标规则的**操作**列单击**开启**。

   服务超时规则配置完成且开启后，请根据实际业务验证服务超时规则是否生效。

相关操作

服务超时规则创建完成后，您还可以**编辑**规则、根据规则的不同状态**关闭**规则或**开启**规则。当不再需要服务超时时，**删除**规则。

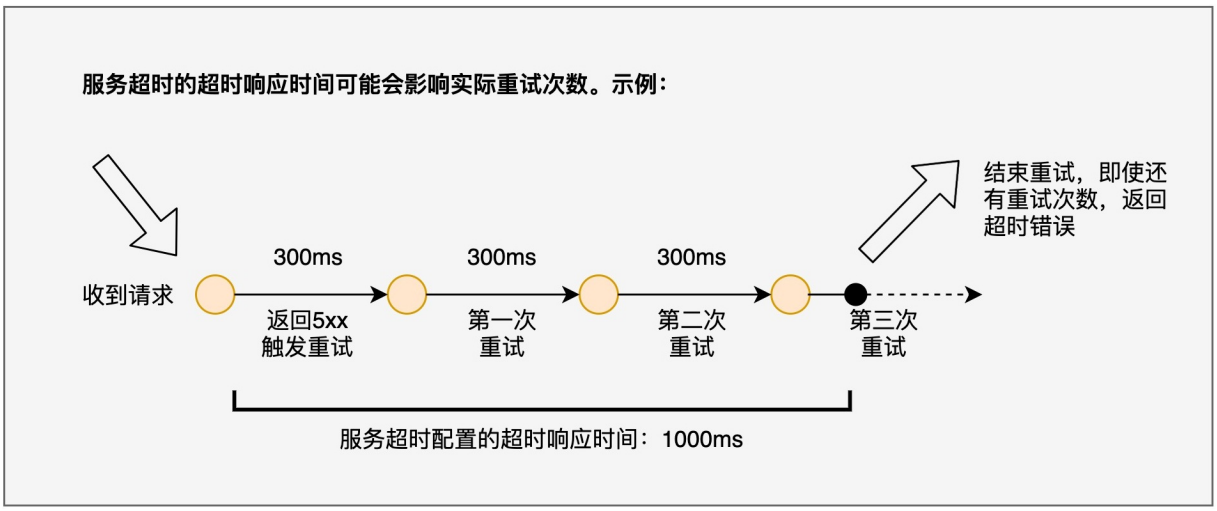
4.10. 配置服务重试

服务重试机制可以在应用暂时不可达或应用内部出现偶发性错误时重新发送请求，通过多次尝试来获取正确的响应信息，提高系统的健壮性。

注意事项说明

同时配置服务超时和服务重试规则，服务超时的超时响应时间可能会影响实际的重试次数。

举例说明，假设在服务超时页面设置的超时响应时间为1000ms，在服务重试页面设置了触发条件为5xx的重试规则，重试次数为5，再假设应用处理一条请求耗时300ms，则此时的实际运行情况如下图所示，应用会在处理第三次重试时因为时间耗尽而结束重试，直接返回超时错误。



创建服务重试规则

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
3. 在顶部菜单栏选择地域，然后单击具体应用的名称。
4. 在**应用详情**页面选择**服务重试**页签，然后单击**创建规则**。
5. 在**创建服务重试规则**面板中配置相关参数，然后单击**确定**。

   服务重试规则的参数说明：

参数	描述
规则名称	服务超时规则的名称。例如：retry-example。
标签	选择服务的路由标签。 **说明** **默认**指所有未配置标签路由的标签集合。
框架类型	应用的框架类型，默认为**服务网格**。
流量来源	请求的发起方，即消费者应用，包括**全部应用**和**特定应用**。 **说明** 当选择特定应用时，只有请求的发起方在指定的应用集合中，才会触发服务重试逻辑。
特定应用	当流量来源选择特定应用时，设置目标应用。

参数	描述
触发条件	选择触发重试的条件。 ○ 5xx：应用返回500-599状态码则进行重试。5xx包含以下四种类型的触发条件： ■ gateway-error：与5xx类似，只针对502，503，504状态码有效。 ■ reset：如果应用没有任何响应则进行重试。 ■ connect-failure：连接失败（例如超时）后进行重试。 ■ refused-stream：服务重置流时进行重试。 ○ retriable-4xx：应用返回4xx状态码时进行重试，目前仅支持409。
重试次数	设置请求触发重试时的最大重试次数。
每次重试的超时响应时间	如果应用的处理时间超过了设定的超时响应时间，则本次重试直接返回超时错误，单位：毫秒。 **说明** 如果仍有重试次数，仍可以继续发送请求。
默认状态	服务重试规则的启用开关。 ○ 打开：创建后即启用，默认打开。 ○ 关闭：创建后不启用，如果需要启用，请在服务重试页面目标规则的**操作**列单击**开启**。

服务重试规则配置完成且开启后，请根据实际业务验证服务重试规则是否生效。

相关操作

服务重试规则创建完成后，您还可以**编辑**规则、根据规则的不同状态**关闭**规则或**开启**规则。当不再需要服务重试时，**删除**规则。

4.11. 配置同AZ路由

同AZ路由，也叫作同可用区优先路由，是一种负载均衡策略，能够使流量尽可能的在同一个可用区内流转，而不是单纯的使用默认提供的轮询方式进行负载均衡。

开启同AZ路由模式

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
3. 在顶部菜单栏选择地域，然后单击具体应用的名称。
4. 在**应用详情**页面选择**同AZ路由**页签。
5. 单击**编辑**，输入阈值并开启同AZ路由状态，然后单击**确定**。

同AZ路由的参数说明：

参数	描述
域值	单击右侧的**编辑**，设置域值。当前应用在某个可用区部署的实例占比超过此阈值时，才会开启该可用区的同AZ路由功能。即消费者向该应用发起请求时，会优先将请求路由到与消费者同一个可用区的实例上。
状态	同AZ路由的启用开关。 ○ 打开：设置域值后即启用，默认打开。 ○ 关闭：设置域值后不启用。 **说明** 请确保当前应用和消费者应用在各个可用区的实例分布尽量均匀。

同AZ路由模式配置完成且开启后，请根据实际业务验证同AZ路由模式是否生效。

5.开发测试治理

5.1. 测试服务

在日常开发中，开发人员或测试人员需要临时调用线上服务来调试已经部署的服务或查询线上数据。服务测试功能可以让您在控制台填写调用参数、发起服务调用，并得到服务调用的结果。

前提条件

- 在使用服务测试前，请确保您的应用已接入MSE治理中心。具体操作，请参见微服务治理中心入门概述。
- 在使用服务测试前，您需要完成RAM授权。具体操作，请参见为RAM用户授予MSE微服务治理中心的操作权限。
- 如果您使用RAM用户测试服务，请先在RAM中配置服务测试相关权限。具体操作，请参见权限配置示例。

视频教程

操作步骤

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务测试**。
3. 在顶部菜单栏选择**地域**，在框架类型中根据需要选择**框架：Spring Cloud**或者**框架：Dubbo**，然后单击目标服务名称或**操作**列的**测试**按钮。
4. 在**选择测试方法**面板中设置测试相关参数，然后单击**执行**。

   测试服务参数说明如下：

参数	描述
调用IP	要测试服务的实例IP。如果部署了多个实例，在列表中选择其中一个IP，进行测试，只能单选。
Path	请求的接口URL，以 `/` 开头，例如 `/_mse_/readiness?` 。
请求方法	该所属类的请求方法，如果包含多个请求方法，在列表中选择其中一种方法，只能单选。
测试参数	在测试方法的参数区域，根据服务的代码设置方法的具体参数。

执行结果

在**结果**区域查看测试是否成功，测试结果一般会有以下几种情况：

- 结果成功，并显示调用服务的响应结果。
- 结果失败，并显示调用服务的失败响应信息。请根据响应信息，排查服务的端口、网络及代码本身的问题。

5.2. 压测服务

在日常开发中，开发人员或测试人员需要评估服务的性能是否符合预期，避免因功能迭代导致服务性能下降而引发故障。服务压测功能可以让您低成本地评估服务性能，做到1分钟创建压测场景，5分钟获取性能指标。

前提条件

在使用服务测试前，请确保您的应用已接入MSE治理中心。具体操作，请参见微服务治理中心入门概述。

背景信息

在大促活动中，应该准备多少实例资源才能满足大促吞吐量的要求，降低因大促活动带来的访问量暴增进而引发系统宕机的风险。此时需要合理地评估服务性能，避免流量冲击引发的故障，并降低运营使用成本。

创建压测场景

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务压测**。
3. 在顶部菜单栏选择地域，然后单击**创建场景**。
4. 在**创建场景**面板中设置**场景配置**、**压力配置**和**数据配置**等相关参数。
   i. 在**场景配置**页签单击右侧的 图标，设置如下相关参数。

参数	描述
场景名称	自定义压测场景名称，例如test-springcloud。
步骤名称	自定义压测步骤名称。
应用	选择需要压测的应用。
框架类型	根据需要选择Spring Cloud或者Dubbo框架，系统会自动识别应用类型。
Path	选择HTTP相对路径，例如*/getIp*，或单击**切换为自定义输入**，手动输入HTTP相对路径。 **说明** 此参数只有在框架类型为Spring Cloud才需要配置。
服务	选择应用的服务。 **说明** 此参数只有在框架类型为Dubbo才需要配置。
方法	选择服务的方法。 **说明** 此参数只有在框架类型为Dubbo才需要配置。

参数	描述
基本信息	■ 对于Spring Cloud框架： 设置**请求方法**和**ContentType**。其中请求方法包括GET/POST/PUT/DELETE，ContentType包括x-www-form-urlencoded和raw，不同的ContentType提供不同可视化的参数输入方式： ■ **x-www-form-urlencoded**：表单输入，传递的参数格式为[{"name": "cart"},{"age": 20}]。 ■ **raw**：默认为application/json JSON格式输入，传递的参数格式为{"name": "cart", "age": 20}。其他格式输入，传递的参数格式按输入文本的传输。 ■ 对于Dubbo框架：支持JSON格式的参数输入方式，默认入参为[]。 ■ 关于Spring Cloud微服务支持的ContentType类型，请参见Spring Cloud参考示例。 ■ 关于Dubbo微服务的方法参数类型及配置方式，请参见Dubbo参数示例。 **说明** GET和DELETE只支持修改URL的Path路径。POST和PUT支持ContentType及参数编写格式。
请求头	设置请求头参数信息。 **说明** 此参数只有在框架类型为Spring Cloud才需要配置。
断言（选填）	输入**检查对象**和**检查内容**，选择**检查条件**。
出参提取（选填）	输入**出参名**和**出参提取表达式**。
打印日志	开启可自动打印日志信息，但会影响到服务压测性能，建议正常压测时关闭。

ii. 在**压力配置**页签设置如下相关参数。

参数	描述
压测模式	压测模式有两种：并发模式（虚拟用户模式）、TPS模式（Transaction Per Second，吞吐量模式）。 ■ **并发模式**：指虚拟并发用户数，从业务角度，也可以理解为同时在线的用户数。 ■ **TPS模式**：指系统每秒处理的事务数量。
流量模型	流量模型包括固定压力、阶梯压力和脉冲压力。 ■ **固定压力**：以配置的固定并发值进行施压，并可设置预热时长。 ■ **阶梯压力**：设置最大值、最小值、预热时间等信息，在预热递增期间，从最小值开始按照阶梯逐步递增，达到最大并发后按照最大并发持续施压。不可指定循环次数。 ■ **脉冲压力**：设置峰值、谷值以及持续时间等信息，施压流量以峰值、峰谷的锯齿波的形式进行施压。
压测时长（分钟）	指压测总时长，公测期间最大压测时长60分钟。
预热时长（分钟）	施压前的预热时间，若设置为0，则表示无需预热。

iii. 在**数据配置**页签单击**添加CSV数据**，在**CSV数据设置**对话框中设置如下相关参数，然后单击**确定**。

参数	描述
名称	自定义参数名称。
注释	自定义参数说明。
变量名称	输入CSV文件中的各个参数名称，多个变量名称之间使用英文逗号（,）进行分隔。
CSV文件	单击**上传文件**，选择本地CSV格式的参数文件进行上传。

> **说明**
> ■ MSE会把CSV参数文件分发到每一个施压机，应用调用会按参数文件中的顺序自动读取参数，您只需在CSV文件中设置参数顺序即可。
> ■ 参数文件仅支持.csv格式，且文件首行表示参数名称，实际压测请求调用时，默认自动忽略首行。

5. 在**场景配置**的**基本信息**配置区域，通过*${XXX}*格式引用定义的参数变量名称，然后单击**确定**。

> **说明** XXX为**数据配置**中添加的**变量名称**，需要用*${}*格式进行引用，例如：*${param}*。

创建多步骤串联的压测场景

> **说明** 一个压测场景可以包含多个压测步骤，当后续的压测步骤依赖先前的压测步骤的输出时，需要使用参数传递。

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务压测**。
3. 在顶部菜单栏选择地域，单击目标用例右侧**操作**列的**详情**。
4. 在**场景详情**面板中单击**编辑场景**，然后在**编辑场景**面板中选择**场景配置**页签，执行以下操作：
   i. 单击目标压测步骤右侧的**访问一次**，弹出**单步骤调试结果**。
   您可查看此次请求**接口信息**、**请求入参**和**请求出参**。
   ii. 在**单步骤调试结果**面板中单击**出参提取助手**，弹出**出参提取助手**窗口，选择需要提取的出参参数进行复制，然后单击**确定**。
   弹出提示内容：*${XXX}*参数已写入剪切板，请直接粘贴。
   iii. 在**出参提取（选填）**下方的**出参提取表达式**中粘贴所选择的出参表达式，并自定义**出参名**。
   iv. 单击**添加下一步骤**添加多个压测步骤。
   v. 在该压测步骤的**基本信息**区域引用变量值，格式为*${XXX}*。

   > **说明** XXX为前序步骤的**出参提取**中自定义的**出参名**，需要用*${}*格式进行引用。

5. 在**编辑场景**面板中选择**压力配置**页签，配置压力数据，然后单击**确定**。
   多步骤串联的压测场景配置完成。

执行结果

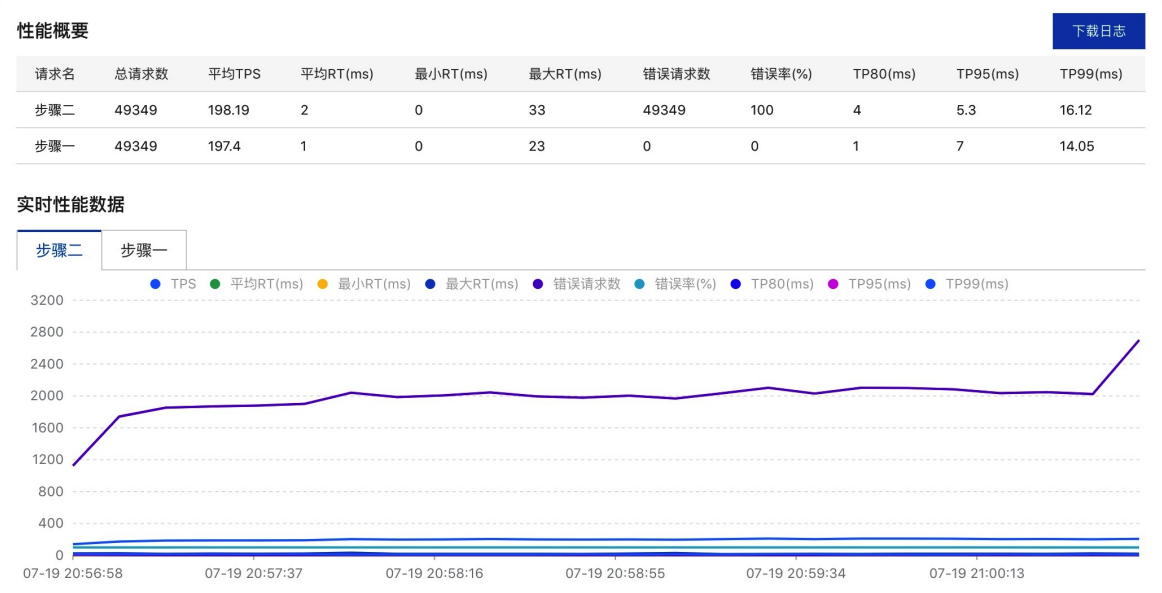
压测场景创建成功后，您可查看**服务压测**列表查看相关信息，包括**平均TPS**、**平均响应时间**、**错误率**等。

您还可以执行以下相关操作管理压测场景：

- 查看压测详情：在**服务压测**列表页面，单击**操作**列的**详情**，在压测**场景详情**页面执行**启动**、**停止**或**编辑场景**等操作。
- 复制压测场景：在**服务压测**列表页面，单击**操作**列的**复制**，可生成一个新的压测场景。
- 删除压测场景：在**服务压测**列表页面，单击**操作**列的**删除**，可删除该压测场景。

查看压测报告

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务压测**。
3. 在顶部菜单栏选择地域，在服务压测列表中单击**操作**列的**详情**，查看**场景配置**和**运行记录**。
4. 在**运行记录**区域单击**操作**列的**详情**，查看实时性能数据。



> **说明** **性能数据**是每10秒的所有施压机数据统计，具体根据压测总时间长度会有所变化。单击图上方的图例，可以显示或隐藏某些数据曲线。

参数	说明
总请求数	整个压测过程中，共发起的请求个数。
平均TPS	压测周期内，所有压力机发出的平均TPS值，TPS=调用总次数/总运行时间。
平均RT（ms）	所有压力机发出平均响应时间。
最小RT（ms）	所有压力机中最小的一次响应时间。
最大RT（ms）	所有压力机中最大的一次响应时间。
错误请求数	所有压力机中错误请求数之和。
错误率	所有压力机中的平均错误率。
TP80（ms）	所有压力机中80分位（P80）的平均值。
TP95（ms）	所有压力机中95分位（P95）的平均值。
TP99（ms）	所有压力机中99分位（P99）的平均值。

5. 单击**下载日志**，可获取压测过程中的日志。

附录一：Spring Cloud参考示例

ContentType	参数编写格式
x-www-form-urlencoded	在表单中以key-value对的方式填入，传递的参数格式：[{"name": "cart"},{"age": 20}]。

ContentType	参数编写格式
raw	• JSON(application/json)：JSON字符串，如：{"name": "cart", "age": 20}。 • XML(application/xml)：Application/XML类型的XML字符串。 • XML(text/html)：TEXT/XML类型的XML字符串。 • HTML(text/html)：HTML字符串。 • JavaScript(application/javascript)：JavaScript字符串。 • Text(text/plain)：纯文本格式的编码形式（TEXT/XML/HTML）。

附录二：Dubbo参考示例

方法	参数类型填写方式	参数填写方式
String sayHello(String name);	["java.lang.String"]	["hello, dubbo"]
String helloBean(HelloBean helloBean);	["com.alibaba.dubbo.api.DemoService"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloBean(HelloBean helloBean1, HelloBean helloBean2);	["com.alibaba.dubbo.api.DemoService","com.alibaba.pts.dubbo.api.DemoService"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"},{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloMap(Map helloMap);	["java.util.Map"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloMap(Map helloMap1, Map helloMap2);	["java.util.Map", "java.util.Map"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"},{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloList(List helloList);	["java.util.List"]	[[1]]

方法	参数类型填写方式	参数填写方式
String helloList(List helloList1, List helloList2);	["java.util.List","java.util.List"]	[[1],[1,2]]
String helloString(String helloString);	["java.lang.String"]	[[1],[1,2],[1,3]]
String helloString(String helloString1, String helloString2);	["java.lang.String","java.lang.String"]	["hello, dubbo", "hello, dubbo"]
String helloInt(int helloInt);	["int"]	["hello, dubbo", "hello, dubbo"]
String helloInt(int helloInt1, int helloInt2);	["int","int"]	["1","2"]
String helloBoolean(boolean helloBoolean);	["boolean"]	["true"]
String helloBoolean(boolean helloBoolean1, boolean helloBoolean2);	["boolean","boolean"]	["true","false"]
String helloVoid();	[]	[]

5.3. 自动化回归服务测试用例

自动化回归功能基于服务契约信息快速编排被测服务、管理自动化测试用例，帮助您高效管理、回归业务测试场景，完成业务快速验证和交付。

前提条件

在使用服务测试前，请确保您的应用已接入MSE治理中心。具体操作，请参见微服务治理中心入门概述。

视频教程

创建测试用例

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选择**地域**，然后单击**创建用例**。
4. 在**创建用例**页面单击测试步骤右侧的下拉框，然后设置相关参数信息。

参数	描述
用例名称	自定义测试用例名称。
步骤名称	自定义测试步骤名称。
应用	选择需要测试的应用。
框架类型	根据需要选择Spring Cloud或Dubbo框架。

参数	描述
Path	设置HTTP相对路径，例如*/getIp*。 **说明** 此参数只有在框架类型为Spring Cloud才需要配置。
服务	选择应用的服务。 **说明** 此参数只有在框架类型为Dubbo才需要配置。
方法	选择服务的方法。 **说明** 此参数只有在框架类型为Dubbo才需要配置。
基本信息	○ 对于Spring Cloud框架： 设置**请求方法**和**ContentType**。其中请求方法包括GET/POST/PUT/DELETE，ContentType包括x-www-form-urlencoded和raw，不同的ContentType提供不同可视化的参数输入方式： ■ **x-www-form-urlencoded**：表单输入，传递的参数格式为[{"name": "cart"},{"age": 20}]。 ■ **raw**：默认为application/json JSON格式输入，传递的参数格式为{"name": "cart", "age": 20}。其他格式输入，传递的参数格式按输入文本的传输。 ○ 对于Dubbo框架：支持JSON格式的参数输入方式，默认入参为[]。 ○ 关于Spring Cloud微服务支持的ContentType类型，请参见附录一：Spring Cloud参考示例。 ○ 关于Dubbo微服务的方法参数类型及配置方式，请参见。Dubbo参考示例
请求头	设置请求头参数信息。 **说明** 此参数只有在框架类型为Spring Cloud才需要配置。
断言（选填）	输入**检查对象**和**检查内容**，选择**检查条件**。
出参提取（选填）	输入**出参名**和**解析表达式**。
（可选）**高级设置**	
用例描述	自定义测试用例描述。

参数	描述
加入用例集	选择需要加入的用例集。若没有用例集，可单击右侧的**创建用例集**进行创建。

5. 单击右侧的**访问一次**，弹出**单步骤调试结果**，查看此次请求入参和请求出参。
6. 单击**出参提取助手**，弹出**出参提取助手**对话框，再单击需要提取的出参名，复制该参数。
7. 在**断言（选填）**下方的**检查对象**中粘贴所复制的参数，选择**检查条件**，输入**检查内容**。
8. 在**出参提取（选填）**下方的**出参提取表达式**中粘贴所复制的参数，并自定义**出参名**。
9. 单击右上方的**保存配置**。
   您可在用例列表中查看创建的测试用例。

创建多步骤串联的测试用例

> **说明** 一个测试用例可以包含多个测试步骤，当后序的测试步骤依赖前序的测试步骤的输出时，需要使用参数传递。

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选择**地域**，单击目标用例右侧**操作**列的**详情**。
4. 在**用例详情**页面单击右侧的**访问一次**，弹出**单步骤调试结果**，查看此次请求入参和出参。
5. 单击**出参提取助手**，弹出**出参提取助手**窗口，选择需要提取的出参参数进行复制。
6. 在**出参提取（选填）**下方的**出参提取表达式**中粘贴所选择的出参表达式，并自定义**出参名**。
7. 单击**添加下一步**增加多个测试步骤。
8. 在该测试步骤的**基本信息**区域，ConetntType选择raw，在**JSON格式化**中输入引用变量名**${xxx}**。

   > **说明** xxx为前序步骤的**出参提取**中设置的**出参名**，需要用 `${ }` 格式进行引用。

9. 单击右上方的**保存配置**，再单击**立即执行**。

创建包含子用例的多步骤串联的测试用例

一个测试用例可以包含多个测试步骤，假设某个用例作为子用例，需要被其他用例引用时，可以通过多步骤串联的方式引入同一个用例集中的其他测试用例，后序步骤可以使用前序步骤中的出参提取变量，做到更好的用例复用。

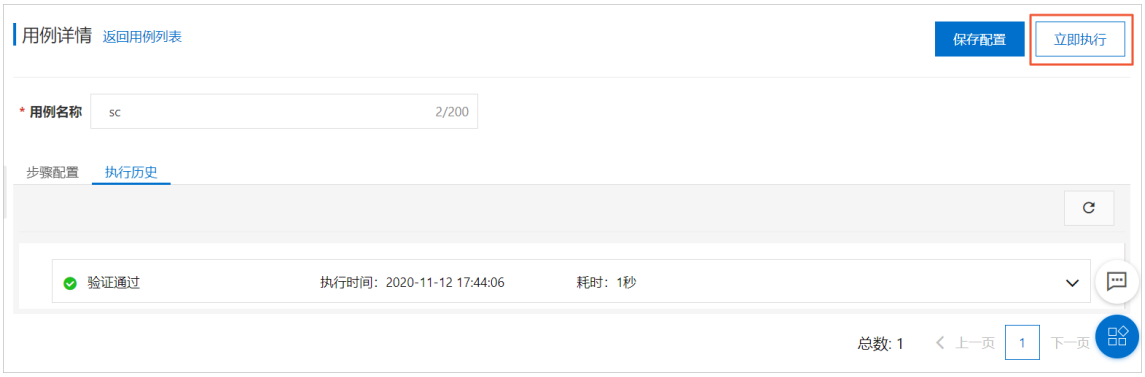
1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选择**地域**，单击目标用例右侧**操作**列的**详情**。
4. 在**用例详情**页面单击**添加测试步骤**右侧的下拉箭头，然后单击**添加子用例**。
5. 在**步骤配置**列表中单击测试步骤右侧的下拉箭头，在**选择子用例**下拉框中选择对应的子用例。

   > **说明**
   > - 如果**选择子用例**无可选项，需要先在**用例详情**页面的**高级设置**区域中将此用例加入用例集。所选择的子用例为同一**用例集**下的其他用例。
   > - 如果无用例集或者加入用例集后仍无可选子用例，请先检查用例集中是否有测试用例。

6. 单击右上方的**保存配置**，然后单击**变量列表**。
   在**变量列表**面板中会出现**子用例变量**，后序测试步骤可直接引用子用例的出参提取变量。
7. 继续添加其他的测试步骤，单击右上方的**保存配置**，然后单击**立即执行**。
   您可以在**执行历史**页签中查看引用子用例变量的后序步骤中，参数被合适的替换执行。

执行测试用例

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选择**地域**。
4. 您可选择以下两种方式执行测试用例。
   - 在**用例列表**页面，单击目标用例右侧**操作**列的**执行**。
   - 在**用例列表**页面，单击目标用例右侧**操作**列的**详情**，在**用例详情**页面单击**立即执行**。



您可在**执行历史**页签中查看详细执行结果。

相关操作

您还可以执行以下操作管理测试用例。

- 复制测试用例：在**自动化回归**列表页面，单击**操作**列的**复制**，可生成一条新的测试用例。
- 删除测试用例：在**自动化回归**列表页面，单击**操作**列的**删除**，可删除该测试用例。

附录一：Spring Cloud参考示例

ContentType	参数编写格式
x-www-form-urlencoded	在表单中以key-value对的方式填入，传递的参数格式：`[{"name": "cart"},{"age": 20}]`。
raw	• JSON(application/json)：JSON字符串，如：`{"name": "cart", "age": 20}`。 • XML(application/xml)：Application/XML类型的XML字符串。 • XML(text/html)：TEXT/XML类型的XML字符串。 • HTML(text/html)：HTML字符串。 • JavaScript(application/javascript)：JavaScript字符串。 • Text(text/plain)：纯文本格式的编码形式（TEXT/XML/HTML）。

附录二：Dubbo参考示例

方法	参数类型填写方式	参数填写方式
String sayHello(String name);	["java.lang.String"]	["hello, dubbo"]
String helloBean(HelloBean helloBean);	["com.alibaba.dubbo.api.DemoService"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloBean(HelloBean helloBean1, HelloBean helloBean2);	["com.alibaba.dubbo.api.DemoService","com.alibaba.pts.dubbo.api.DemoService"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"},{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloMap(Map helloMap);	["java.util.Map"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloMap(Map helloMap1, Map helloMap2);	["java.util.Map", "java.util.Map"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"},{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloList(List helloList);	["java.util.List"]	[[1]]
String helloList(List helloList1, List helloList2);	["java.util.List","java.util.List"]	[[1],[1,2]]
String helloString(String helloString);	["java.lang.String"]	[[1],[1,2],[1,3]]
String helloString(String helloString1, String helloString2);	["java.lang.String","java.lang.String"]	["hello, dubbo", "hello, dubbo"]
String helloInt(int helloInt);	["int"]	["hello, dubbo", "hello, dubbo"]
String helloInt(int helloInt1, int helloInt2);	["int","int"]	["1","2"]

方法	参数类型填写方式	参数填写方式
String helloBoolean(boolean helloBoolean);	["boolean"]	["true"]
String helloBoolean(boolean helloBoolean1, boolean helloBoolean2);	["boolean","boolean"]	["true","false"]
String helloVoid();	[]	[]

5.4. 自动化回归变量使用方法

本文介绍自动化回归提供的变量类型以及变量作为接口参数的使用方法。

背景信息

在测试用例编排过程中，经常会遇到参数的传递和共享，并且产生不必要参数的复制与粘贴，微服务测试自动化回归中提供丰富的变量来实现测试请求的动态可变性。

查看变量类型

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选择地域，然后单击目标用例**操作**列的**详情**。
4. 在**用例详情**页面单击右上方的**变量列表**，在**变量列表**面板中查看变量类型及使用方法。

> **说明** 当变量重名时，变量优先级为：**出参提取变量>环境变量>集合变量>自定义全局变量**。

变量类型	适用范围	说明
出参提取变量	当前测试用例	当创建多个步骤的测试用例时，将前面测试步骤的出参提取（当前测试步骤请求的返回值中截取需要的内容，可提取多个）作为变量，在后续测试步骤的请求中作为参数使用。 一个测试用例中出参提取的出参名不允许重复。
环境变量	所有测试用例	被测服务有多个环境时，先在测试环境完成测试，再部署到线上进行回归，但测试用例不会发生变化。 不同环境的被测域名不同，例如测试环境变量test定义一个base_uri，在线上环境变量online也定义一个base_uri，测试用例请求URL中都引用${base_uri}，切换环境即可运行不同环境的用例。 一个环境内变量名唯一，环境与环境之间可以定义重复的变量。

变量类型	适用范围	说明
集合变量	测试集下的测试用例	归属的测试集内测试用例共享的变量，非测试集内的测试用例无法引用声明的集合变量。 测试集内变量名唯一，不同测试集之间可以定义重复的变量。
自定义全局变量	所有测试用例	同一云账号下的全局变量名唯一，即当前登录用户不能设置两个同名的变量。
子用例变量		

设置出参提取变量

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选择地域，然后单击目标用例**操作**列的**详情**。
4. 在**用例详情**页面单击右侧的**访问一次**，在**单步骤调试结果**面板中查看此次请求入参和出参，然后单击**出参提取助手**。
5. 在**出参提取助手**对话框选择需要提取的出参的参数进行复制，然后单击**确定**。
6. 在**步骤配置**右侧单击 图标，展开自动化回归参数。
7. 单击**出参提取（选填）**页签，在**出参提取表达式**中粘贴所选择的出参表达式，并自定义**出参名**。

> **说明** 在后续测试步骤的参数以*${出参名}*，例如*${code}*格式进行引用。

设置环境变量

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选择地域，然后单击目标用例**操作**列的**详情**。
4. 新增环境变量操作如下：
   i. 在**用例详情**页面单击右上方的**变量列表**。
   ii. 在**变量列表**面板中单击**环境变量**右侧的 图标。
   iii. 在**新增环境变量**对话框中单击**添加环境变量**，设置区分环境可访问的变量，您可新增环境**变量名**和**变量值**及**备注**等，然后单击**确定**。

   > **说明** 变量名仅支持以字母开头，包含下划线（_）、短划线（-）、字母和数字。
5. 编辑环境变量操作如下：
   i. 在**用例详情**页面单击右上角的**环境变量**下拉框中对应环境变量名右侧的 图标。

ii. 在**编辑环境变量**对话框中设置环境变量，您可编辑环境**变量名**和**变量值**，或在**操作**列删除该变量，然后单击**确定**。

> **说明** 变量名仅支持以字母开头，包含下划线（_）、短划线（-）、字母和数字。

iii. （可选）在**用例详情**页面，单击右上角的**变量列表**，单击**环境变量**右侧的 + 图标，也可对环境变量进行修改和删除操作。

设置集合变量

集合变量需要先将测试用例加入用例集后，在用例集中设置。

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例集）**。
3. 在顶部菜单栏选择地域，然后单击目标用例集**操作**列的**详情**。
4. 在**用例集详情**页面单击**变量设置**页签。
5. 在**集合变量**区域设置集合**变量名**和**变量值**，您也可在**操作**列单击 图标删除该变量，然后单击**保存用例集**。

> **说明** 变量名仅支持以字母开头，包含下划线（_）、短划线（-）、字母和数字。

设置自定义全局变量

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选择地域，然后单击目标用例集**操作**列的**详情**。
4. 在**用例详情**页面单击右上方的**变量列表**。
5. 在**变量列表**面板中单击**自定义全局变量**右侧的 图标。
6. 在**编辑自定义全局变量**对话框中设置自定义全局**变量名**和**变量值**，您也可在**操作**列单击 图标删除该变量，然后单击**确定**。

> **说明** 变量名仅支持以字母开头，包含下划线（_）、短划线（-）、字母和数字。

出参提取变量设置为全局变量

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选择地域，然后单击目标用例集**操作**列的**详情**。
4. 在**步骤配置**右侧单击 图标，展开自动化回归参数。
5. 单击**出参提取（选填）**页签，设置**出参名**和**出参提取表达式**，然后在**操作**列选中**设置为全局变量**。
6. 单击页面右上方的**保存配置**，然后单击**变量列表**。
   您可在**变量列表**面板中的**出参提取变量设置为全局变量**区域看到设置的变量信息。

5.5. 自动化回归服务测试用例集

自动化回归测试用例集功能通过关联测试用例，帮助您快速完成业务验证和交付。

前提条件

在使用服务测试前，请确保您的应用已接入MSE治理中心。具体操作，请参见微服务治理中心入门概述。

创建测试用例

创建测试用例集

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例集）**。
3. 在顶部菜单栏选择**地域**，然后单击**创建用例集**。
4. 在**创建用例集**面板中输入用例集名称，单击**确定**即可创建用例集。

关联测试用例

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例集）**。
3. 在顶部菜单栏选中**地域**，单击目标用例集右侧**操作**列的**详情**。
4. 在**用例集详情**页面单击**关联用例**。选中关联的用例单击**确定**即可关联用例。
5. （可选）若您想取消关联用例，在**用例列表**中单击**操作**列的**取消关联**。

执行测试用例集

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例集）**。
3. 在顶部菜单栏选择地域。
4. 您可选择以下两种方式执行测试用例集。
   - 在用例集列表页面，单击目标用例集右侧**操作**列的**执行**。
   - 在用例集列表页面，单击目标用例集右侧**操作**列的**详情**，在**用例集详情**页面单击**执行用例集**。

   您可在**执行历史**页签中查看详细执行结果。

相关操作

您还可以执行以下操作管理测试用例集。

- 复制测试用例：在**用例集详情**页面的**用例列表**页签中，单击**操作**列的**复制**，可生成一条新的测试用例。
- 删除测试用例集：在**自动化回归（用例集）**列表页面，单击**操作**列的**删除**，可删除该测试用例集。

5.6. 自动化回归的脚本化编排

自动化回归脚本化编排支持将测试用例的UI编排转换成脚本化编排，还可以将测试用例加入用例集后，在用例集中将加入的所有用例导出成JSON格式的脚本文件，然后使用本地编辑器对脚本进行修改后再导入到当前用例集或其他用例集中，方便您对用例进行迁移和管理。

前提条件

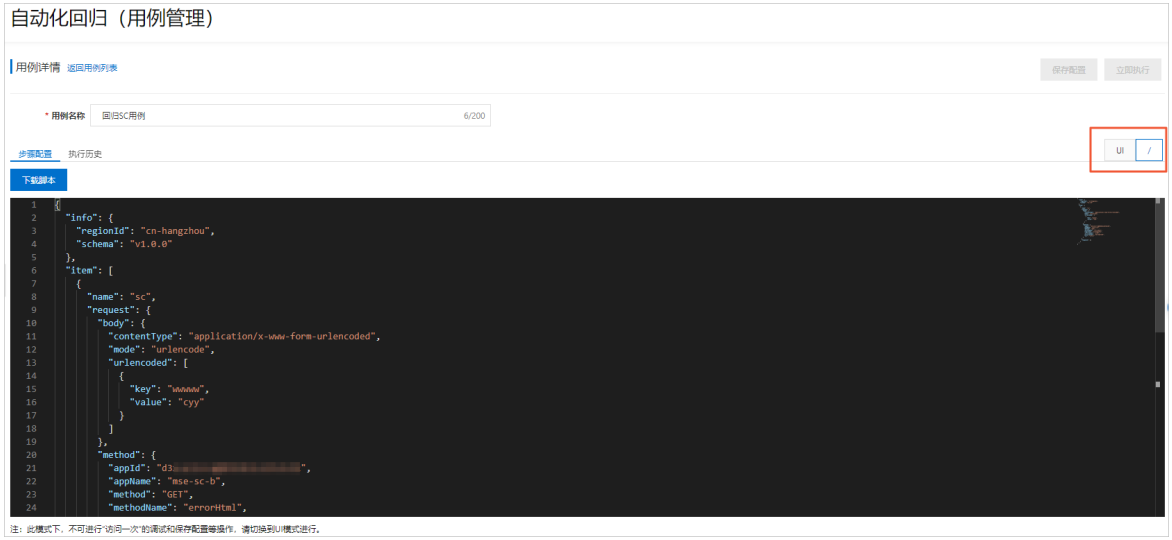
- 在使用服务测试前，请确保您的应用已接入MSE治理中心。具体操作，请参见微服务治理中心入门概述。
- 创建测试用例。

操作步骤

1. 登录MSE治理中心控制台。

2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例管理）**。
3. 在顶部菜单栏选择地域，然后单击目标用例**操作**列下方的**详情**。
4. 在**用例详情**页面右侧单击/图标，切换到脚本模式；单击*UI*，切换到UI模式。

   关于用例管理的脚本参数说明，请参见用例集管理脚本参数说明。



> **说明** 脚本模式下，不可进行**访问一次**的调试和**保存配置**等操作，请切换到UI模式进行。

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 自动化回归（用例集）**。
3. 在顶部菜单栏选择地域，然后单击目标用例**操作**列下方的**详情**。
4. 在**用例集详情**页面的**用例列表**页签单击**导出脚本**，在**导出脚本**对话框中单击**导出**。

   关于用例集管理的脚本参数说明，请参见用例集管理脚本参数说明。

   成功导出JSON格式的脚本文件。
5. 使用本地编辑器对脚本进行修改，然后在**用例集详情**页面右侧单击**导入脚本**，在**导入脚本**对话框中选择**相同用例**执行的操作，并单击**上传文件**，选择并上传本地的脚本文件，然后单击**导入**。

相同用例是指该用例集下相同名称的用例，执行的操作说明如下：

- **终止导入**：此次操作选中的所有用例都不会被同步到目标微服务空间。
- **跳过**：跳过重复的用例，继续克隆其他用例。
- **覆盖**：用此次选择的用例覆盖目标微服务空间中已有的相同用例。

脚本化编排用例管理

脚本化编排用例集管理

用例管理脚本模式参数说明

自动化回归功能支持将UI模式切换成脚本模式修改测试用例，再将脚本模式切换到UI模式进行调试和保存，脚本中有关请求的基础信息解析为测试步骤的API。

脚本模式相关的字段解析说明如下。

脚本字段	字段解析
info.namespaceId	微服务空间。
info.regionId	地域。
info.schema	脚本版本号。
item[]	表示多个测试步骤。
item[0].name	测试步骤的名称。
item[0].request.method	请求方法。
regionId	地域，支持Spring Cloud和Dubbo服务。
appId	应用ID，支持Spring Cloud和Dubbo服务。
appName	应用名称，支持Spring Cloud和Dubbo服务。
serviceType	服务类型，支持Spring Cloud和Dubbo服务。
serviceName	服务名称，支持Spring Cloud和Dubbo服务。
methodName	方法名称，支持Spring Cloud和Dubbo服务。
methodTypes	方法类型，仅支持Dubbo服务。
group	组别，仅支持Dubbo服务。
version	版本号，仅支持Dubbo服务。
method	请求方法，仅支持Spring Cloud服务。
uri	请求路径，仅支持Spring Cloud服务。
item[0].request.body	参数基本信息。

脚本字段	字段解析

contentType
mode
urlencoded
raw
	根据框架类型和ContentType区分，其中mode有urlencoded和raw两种模式：
	• `"mode":"urlencoded"`：当框架类型为Spring Cloud服务时，可选此模式。入参信息填在对应的`urlencoded`中，以`key`和`value`的方式传入，例如：
	`"urlencoded":[ { "key":"aa", "value":"11" } ]`
	• `"mode":"raw"`：Spring Cloud服务可选此模式，Dubbo仅支持此类型，入参信息填在对应`raw`中，以字符串方式传入，例如：
	`"raw":"[\"11234\"]"`

item[0].request.header	请求头。
key	请求头的Key，仅支持Spring Cloud服务。
value	请求头对应的Value值，仅支持Spring Cloud服务。
item[0].request.checkpoints	断言。
point	检查对象，为`${}`括起来的`JsonPath`，例如`${response.name}`。
checkers.operate	检查条件。
checkers.expect	检查内容，即预期值。
item[0].request.exports	出参提取。
key	出参提取的出参名。
value	对应的出参提取表达式，为`${}`括起来的`JsonPath`。

说明

- 脚本中的info信息仅做展示，转换成UI模式时以实际页面选择为准。
- JSON脚本中，配置`item[0] item[1] item[2]……`表示有多个测试步骤。
- 从脚本模式转换成UI模式时，将对脚本格式和`request.method`中的内容进行正确性校验，若不正确，则不允许转换。其中`request.method.method`为Spring Cloud框架类型基本信息中的请求方法，支持GET、POST、PUT和DELETE。

其中 `request.body.contentType` 支持以下7种类型。

contentType	描述
`application/x-www-form-urlencoded`	对应 `mode` 为 `urlencoded` 。
`application/json`	对应 `mode` 为 `raw` 。
`text/plain`	
`application/javascript`	
`text/html`	
`text/xml`	
`application/xml`	

> **说明** Dubbo服务仅支持 `contentType=application/json` , `mode=raw` 的模式。

其中 `request.checkpoints.checkers.operate` 检查条件枚举如下。

checkers.operate	描述
EQUAL_NUMERIC	等于（数字）
NOT_EQUAL_NUMERIC	不等于（数字）
GREATER_OR_EQUAL_NUMERIC	大于等于（数字）
LESS_OR_EQUAL_NUMERIC	小于等于（数字）
GREATER_NUMERIC	大于（数字）
LESS_NUMERIC	小于（数字）
EQUAL_STRING	等于（字符串、区分大小写）
NOT_EQUAL_STRING	不等于（字符串、区分大小写）
EQUAL_STRING_IGNORE	等于（字符串、不区分大小写）
NOT_EQUAL_STRING_IGNORE	不等于（字符串、不区分大小写）
CONTAIN_STRING	包含（字符串）
NOT_CONTAIN_STRING	不包含（字符串）
TIME_EARLY	时间早于
TIME_LATE	时间晚于

checkers.operate	描述
IS_NULL	为空，表示没有该字段。 **说明** `checkers.operate` 为该字段时，无需设置 `checkers.expect` 。
IS_NOT_NULL	不为空，表示有该字段。 **说明** `checkers.operate` 为该字段时，无需设置 `checkers.expect` 。
JSON_CONTAIN	JSON对象中包含此值
JSON_NOT_CONTAIN	JSON对象中不包含此值
JSON_ARRAY_NULL	JSON数组是否为空数组
JSON_ARRAY_NOT_NULL	JSON数组是否为非空数组
JSON_ARRAY_SIZE_EQUAL	JSON数组长度等于
JSON_ARRAY_SIZE_GREATER	JSON数组长度大于
JSON_ARRAY_SIZE_GREATER_OR_EQUAL	JSON数组长度大于等于
JSON_ARRAY_SIZE_LESS	JSON数组长度小于
JSON_ARRAY_SIZE_LESS_OR_EQUAL	JSON数组长度小于等于
IS_JSON_OBJECT	是否为JSON对象类型 **说明** `checkers.operate` 为该字段时，无需设置 `checkers.expect` 。
IS_JSON_ARRAY	是否为JSON数组类型 **说明** `checkers.operate` 为该字段时，无需设置 `checkers.expect` 。
REGEX_COMPARE	正则表达式

用例集管理脚本参数说明

自动化回归用例集功能支持将加入用例集中的所有用例导出成脚本，然后使用本地用编辑器对脚本进行修改后导入到当前用例集或其他用例集中。

导出的脚本文件为JSON格式，字段解析和测试用例的脚本模式相比多了一层 `item` ，内容如下：

脚本字段	字段解析
info.namespaceId	微服务空间
info.regionId	地域
info.schema	脚本版本号
item[]	多个测试用例
item[0].name	测试用例的名称
item[0].item[0].name	第1个用例的第一个步骤的名称
item[0].item[1].name	第1个用例的第二个步骤的名称

说明

- 脚本中的info信息仅做展示，导入脚本时以实际页面选择为准。
- JSON脚本中，配置 `item[0] item[1] item[2]……` 表示有多个测试用例。
- 导入脚本时将对脚本格式和 `request.method` 中的内容进行正确性校验。若导入失败，则返回导入的总数、成功数、失败数和具体的失败原因。
- 用例集下相同名称的用例将被识别为相同用例。若遇到相同用例时，可选择导入规则为终止导入、跳过或覆盖。

5.7. 巡检服务

目前，随着云原生技术的推广和普及，微服务化已成为趋势。但线上微服务接口可靠性却并不完善，无法实时感知异常，存在较大风险。本文介绍微服务巡检平台的相关操作，帮助您随时了解API或微服务接口的运行情况，降低服务风险。

背景信息

云原生时代下应用微服务化是趋势，但企业如何保障线上微服务的可靠性，主动感知线上微服务异常，降低业务风险呢？微服务巡检帮助您对线上服务进行7*24小时的秒级探测，实时了解服务的健康度，且当服务异常时及时告警，尽快恢复，降低损失。

视频教程

创建巡检服务

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务巡检**。
3. 在顶部菜单栏选择**地域**，然后在**服务巡检**页面单击**创建巡检任务**。
4. 在**创建巡检任务**面板中设置相关参数，然后单击**确定**。

参数	描述
服务巡检任务名称	自定义服务巡检任务名称。
应用	选择需要巡检的应用。

参数	描述
框架类型	支持Spring Cloud、Dubbo和服务网格框架。系统会根据所选应用自动识别其框架，也可以手动选择框架类型。
Path	选择HTTP相对路径，例如*/getIp*。或单击**切换为自定义输入**，手动输入HTTP相对路径。 **说明** 此参数只有框架类型为Spring Cloud时，才需要配置。
服务	选定应用中需要巡检的目标服务。 **说明** 此参数只有框架类型为Dubbo时，才需要配置。
方法	选定服务中需要巡检的方法。 **说明** 此参数只有框架类型为Dubbo时，才需要配置。
基本信息	设置请求方式，包括GET/POST/PUT/DELETE。 **说明** GET和DELETE只支持修改URL的Path路径。POST和PUT支持ContentType及参数编写格式。
请求参数	设置请求参数。关于Dubbo服务的方法参数类型及配置方式，请参见Dubbo参考示例。
请求头	设置请求头参数信息。关于Spring Cloud微服务支持的ContentType类型，请参见Spring Cloud参考示例。
断言信息包含	设置接口返回值信息。如果返回值含有一个特征，如返回值含有123，则格式为"123"；如果返回值含有多个特征，如同时含有123，abc，则格式为["123","abc"]。
巡检周期	设置巡检周期，单位秒/分钟，可自定义选择。
报警触发条件	当接口巡检异常时，告警触发的频率。
报警接收管理	接收告警的联系人组。在左侧列中选中需要接手告警的联系人组，并单击>，添加到右侧列表中。
报警通知方式	报警通知方式包含**钉钉**、**短信**和**邮件**。

服务巡检任务创建成功后，返回**服务巡检**列表，查看相关信息，包括**巡检次数**、**可用率**、**平均响应时间**等。

相关操作

您还可以执行以下操作管理服务巡检。

- 任务运行：在服务巡检列表页面，单击**操作**列的**启动**，可重新启动该服务巡检任务。
- 更新配置：在服务巡检列表页面，单击**操作**列的**详情**，可重新编辑服务巡检任务。
- 暂停服务：在服务巡检列表页面，单击**操作**列的**暂停**，可暂停该服务巡检任务。
- 查看失败记录：在服务巡检列表页面，单击**操作**列的**失败记录**，可查看该服务巡检的监控详情。

Spring Cloud参考示例

ContentType	参数编写格式
application/x-www-form-urlencoded	[{"name": "cart"},{"age": 20}]
application/json（默认）	{"name": "cart", "age": 20}

Dubbo参考示例

方法	参数类型填写方式	参数填写方式
String sayHello(String name);	["java.lang.String"]	["hello, dubbo"]
String helloBean(HelloBean helloBean);	["com.alibaba.dubbo.api.DemoService"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloBean(HelloBean helloBean1, HelloBean helloBean2);	["com.alibaba.dubbo.api.DemoService","com.alibaba.pts.dubbo.api.DemoService"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"},{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloMap(Map helloMap);	["java.util.Map"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloMap(Map helloMap1, Map helloMap2);	["java.util.Map", "java.util.Map"]	[{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"},{"booleanValue":true,"helloSubValue":{"booleanValue":false,"intValue":2,"stringValue":"subbean"},"intValue":1,"stringValue":"bean"}]
String helloList(List helloList);	["java.util.List"]	[[1]]

方法	参数类型填写方式	参数填写方式
String helloList(List helloList1, List helloList2);	["java.util.List","java.util.List"]	[[1],[1,2]]
String helloString(String helloString);	["java.lang.String"]	[[1],[1,2],[1,3]]
String helloString(String helloString1, String helloString2);	["java.lang.String","java.lang.String"]	["hello, dubbo", "hello, dubbo"]
String helloInt(int helloInt);	["int"]	["hello, dubbo", "hello, dubbo"]
String helloInt(int helloInt1, int helloInt2);	["int","int"]	["1","2"]
String helloBoolean(boolean helloBoolean);	["boolean"]	["true"]
String helloBoolean(boolean helloBoolean1, boolean helloBoolean2);	["boolean","boolean"]	["true","false"]
String helloVoid();	[]	[]

5.8. 智能流量测试服务

智能流量测试功能通过录制微服务应用接口的流量，并自动生成对应的自动化回归测试用例和服务压测场景，帮助您模拟真实请求进行服务压测并以零编码成本完成接口自动化回归。本文介绍如何录制Spring Cloud服务的流量和如何将录制流量自动生成服务压测场景，Dubbo服务同样适用。

前提条件

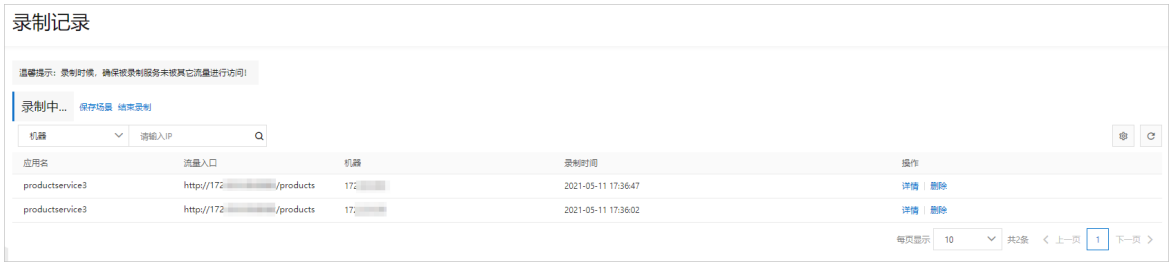
在使用智能流量测试前，请确保您的应用已接入MSE治理中心。具体操作，请参见微服务治理中心入门概述。

背景信息

在微服务测试过程中，开发人员很难编写应用的API测试用例，通常期望能通过控制台操作，快速生成应用的API测试用例和性能测试用例。流量测试功能可以帮助您低成本、低门槛的录制应用的API流量，并支持自动生成多条流量的压测参数和压测场景，帮助您轻松完成自动化测试和性能测试场景编写。

录制自动化回归流量

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 智能流量测试**。
3. 在顶部菜单栏选择**地域**，然后在**应用名**文本框中输入应用名称，单击图标。
4. 在应用列表中单击目标应用**操作**列的**自动化回归录制**。
5. 在**录制流量**对话框中选择路径，然后单击**确认**。
6. 在**录制记录**页面，您可查看当前流量信息，包括应用名、流量入口、机器、录制时间等。



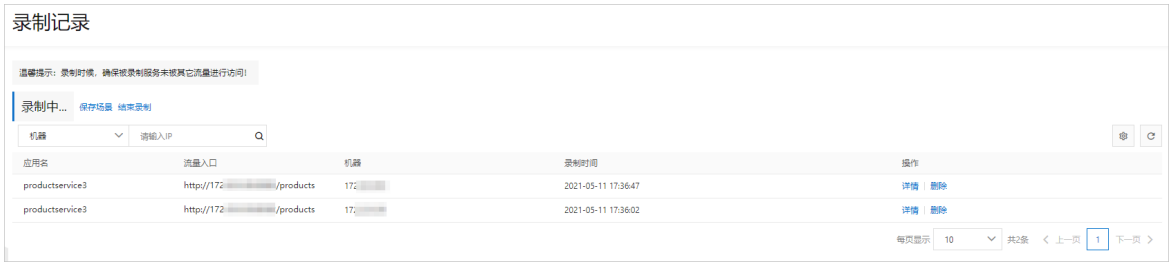
当前流量正在**录制中**，您可在**录制记录**页面执行以下操作：

- 单击目标录制流量**操作**列下的**详情**，可在**流量详情**面板中查看请求信息、响应信息等。
- 单击目标录制流量**操作**列下的**删除**，可删除该流量数据。

7. （可选）在**录制记录**页面单击左上角**保存场景**，在**保存操作场景**对话框中输入场景名，单击**确定**。
   当前流量录制结果会自动保存至管理页面，请参见管理流量录制场景。
8. （可选）在**录制记录**页面单击左上角**结束录制**。
   自动返回**智能流量测试**页面。

录制服务压测流量

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 智能流量测试**。
3. 在顶部菜单栏选择**地域**，然后在**应用名**文本框中输入应用名称，单击图标。
4. 在应用列表中单击目标应用**操作**列的**服务压测录制**。
5. 在**录制的请求路径**对话框中选择路径，然后单击**确认**。
6. 在**录制记录**页面，您可查看当前流量信息，包括应用名、流量入口、机器、录制时间等。



当前流量正在**录制中**，您可在**录制记录**页面执行以下操作：

- 单击目标录制流量**操作**列下的**详情**，可在**流量详情**面板中查看请求信息、响应信息等。
- 单击目标录制流量**操作**列下的**删除**，可删除该流量数据。

7. （可选）在**录制记录**页面单击左上角**保存场景**，在**保存操作场景**对话框中输入场景名，单击**确定**。
   当前流量录制结果会自动保存至管理页面，请参见管理流量录制场景。
8. （可选）在**录制记录**页面单击左上角**结束录制**。
   自动返回**智能流量测试**页面。

管理流量录制场景

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 智能流量测试**。
3. 在顶部菜单栏选择**地域**，然后在**应用名**文本框中输入应用名称，单击图标。
4. 在应用列表中单击目标应用**操作**列的**管理**。

5. 在**流量管理**页面，您可查看保存的流量信息，包括流量场景、场景类别等。

您可在**流量管理**页面执行以下操作：

- 单击目标流量**操作**列下的**详情**，可在**场景对应流量列表**面板中查看流量数据。
- 单击目标流量**操作**列下的**删除**，可删除该流量数据。

生成服务压测场景

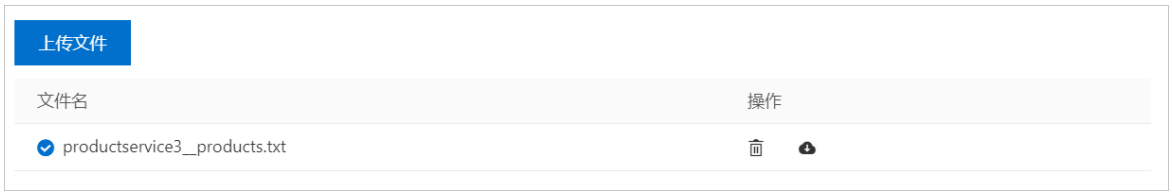
1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 智能流量测试**。
3. 在顶部菜单栏选择**地域**，然后在**应用名**文本框中输入应用名称，单击图标。
4. 在应用列表中单击目标应用**操作**列的**管理**。
5. 在**流量管理**页面，选中流量场景，单击**生成服务压测场景**，在**生成压测场景**对话框中单击**确认**。

控制台自动跳转至服务压测**场景详情**页面。

6. 在服务压测**场景详情**面板中单击**编辑场景**。

   关于服务压测配置的相关内容，请参见压测服务。

7. 在**场景配置**页签的配置文件区域可选择上传或下载文件。



- 单击**上传文件**，可上传更新后的参数文件。
- 单击图标，可下载参数文件进行查看和编辑。

生成自动化回归场景

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 智能流量测试**。

3. 在顶部菜单栏选择**地域**，然后在**应用名**文本框中输入应用名称，单击图标。
4. 在应用列表中单击目标应用**操作**列的**管理**。
5. 在**流量管理**页面，选中需要生产压测的场景，单击**批量生成自动化回归用例**，在**生成自动化回归场景**对话框中单击**确认**。
   自动生成对应的自动化回归测试用例，控制台自动跳转至服务**自动化回归（用例管理）**页面。
6. 在**用例来源**下拉框中选择**智能流量测试**，单击目标自动化回归用例**操作**列下方的**详情**。
7. 在**用例详情**页面，选择**步骤配置**页签，单击展开图标。
   关于服务自动化回归测试用例的相关内容，请参见自动化回归服务测试用例。
8. 在**步骤配置**中单击**断言（选填）**页签，查看和修改用例断言内容。
9. 单击右侧的**断言规则配置**，您可在**断言规则配置**面板中配置断言规则。

> **说明** 测试用例可以直接执行回归测试或者加入用例集进行回归测试。相关内容，请参见自动化回归服务测试用例集。

5.9. 配置服务Mock

您可以通过MSE创建Mock服务，系统自动根据请求参数返回不同的结果，并且随机生成返回数据，能够真实地模拟后端服务，支持系统联调。例如部署了2个应用：生产者Provider和消费者Consumer，Consumer依赖了Provider的接口，由于Provider的代码还没准备就绪，可以选择Consumer应用创建Mock规则，模拟Provider的接口返回值。

前提条件

在使用服务Mock前，请确保您的应用已接入MSE治理中心。具体操作，请参见微服务治理中心入门概述。

创建服务Mock规则

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 开发测试治理 > 服务Mock**。
3. 在顶部菜单栏选择**地域**，在**服务Mock**页面单击**创建服务Mock**。
4. 在**创建服务Mock**面板中填入相关参数，然后单击**确定**。
   创建服务Mock参数说明如下。

参数	描述
规则名称	输入服务Mock规则名称，支持大小写字母、数字、下划线（_）和短划线（-），长度不超过64个字符。
描述	输入Mock规则描述信息。
应用	选择需要Mock的应用。
Mock规则列表	单击**添加规则**，展开输入Mock规则。 **说明** 您可以同时添加多个Mock规则，最先创建的规则优先级最高。

参数	描述
框架类型	包含**Spring Cloud**框架和**Dubbo**框架，根据实际应用选择框架类型。 ○ 若您选择**Spring Cloud**框架，设置**服务路径**和**请求方法**，例如*/getIp*和GET。 ○ 若您选择**Dubbo**框架，设置**服务方法**。
条件模式	选择服务Mock规则的条件策略，包括**同时满足下列条件**和**满足下列任一条件**，请根据实际需求进行选择。
条件列表	单击**添加新的规则条件**，设置规则条件。 ○ 若您选择**Spring Cloud**应用，支持以下JSON格式的参数输入方式： ■ Parameter ■ Header ■ Cookie ■ Body ○ 若您选择**Dubbo**应用，支持以下JSON格式的参数输入方式，其中默认入参为[]： ■ RpcContext ■ Parameter
Mock策略	默认支持返回自定义JSON数据策略。
返回数据	自定义Mock返回值。例如：`{"name": "123","age":"123"}`。
返回延迟	自定义请求的响应时间，单位：ms。
默认状态	规则的启用开关。 ○ 打开：创建后即启用，默认打开。 ○ 关闭：创建后不启用，如果需要启用，请在**服务Mock**页面规则的**操作**列单击**开启**规则。

服务Mock规则配置完成且开启后，请根据实际业务验证服务Mock规则是否生效。

相关操作

服务Mock规则创建完成后，在**服务Mock**页面您还可以**编辑**规则、根据规则的不同状态**关闭**规则或**开启**规则。当不再需要服务Mock时，**删除**规则。

创建Mock服务后，系统自动根据请求参数返回不同的结果。支持随机生成返回数据，能够真实地模拟后端服务，支持系统联调，用于系统间第三方依赖Mock。例如部署了2个应用：生产者Provider和消费者Consumer，Consumer依赖了Provider的接口，由于Provider的代码还没准备就绪，可以选择Consumer应用创建Mock规则，模拟Provider的接口返回值。

6.安全治理

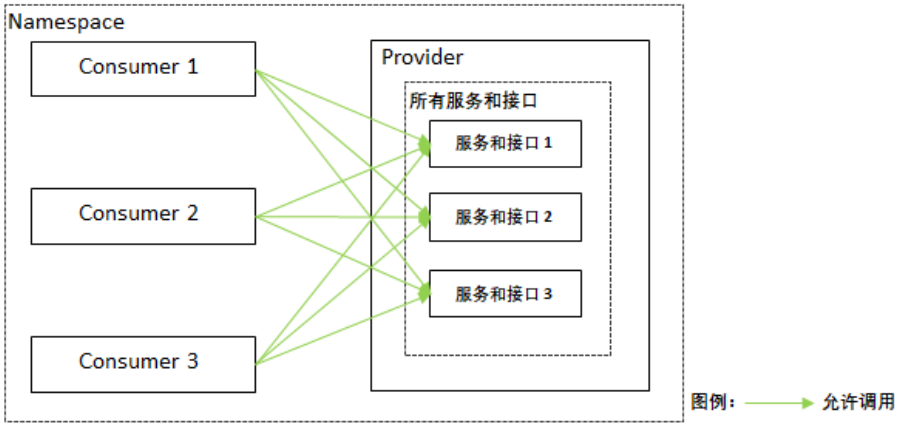
6.1. 配置服务鉴权

当您的某个微服务应用有安全要求，不希望其他所有应用都能调用时，可以对调用该应用的其他应用进行鉴权，仅允许匹配鉴权规则的应用调用。

背景信息

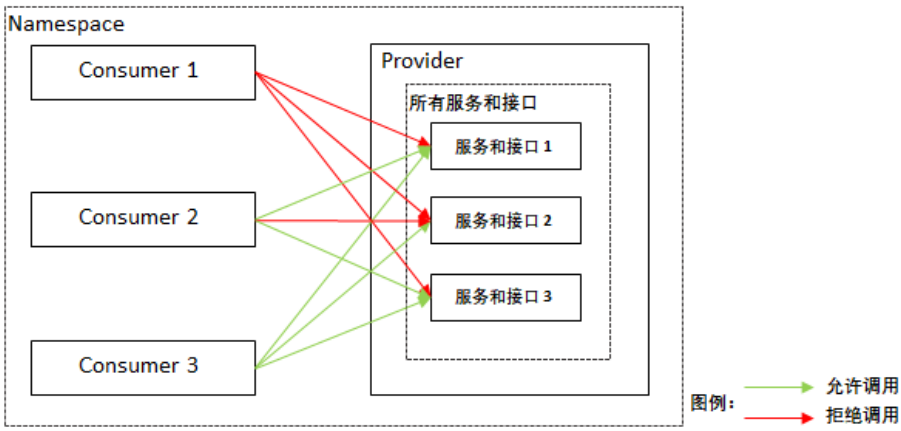
下面以一个示例介绍Spring Cloud服务鉴权的使用场景。Dubbo服务鉴权同样适用。

- 未配置服务鉴权
  Consumer 1、2、3和Provider在同一个命名空间内，Consumer 1、2和3默认可以调用Provider的所有Path（Path 1、2和3）。



- 配置服务鉴权
  - 设置所有Path的鉴权
    可以对Provider的所有Path设置鉴权规则，例如Provider所有Path的鉴权规则设置为拒绝Consumer 1调用（黑名单），则允许Consumer 2、3调用（白名单）。
  - 设置指定Path的鉴权
    在设置所有Path的鉴权基础上，还可以设置Consumer指定Path的鉴权规则，例如按所有Path的鉴权方式，Consumer 2、3可以访问Provider的所有Path，但Provider的Path2涉及一些核心业务或数据，不希望Consumer 2调用，可以将Path 2对Consumer 2的鉴权方式设置为黑名单（拒绝调用），则Consumer 2只能访问Provider的Path 1和Path 3。

  设置完鉴权规则的调用关系如下图所示。



视频教程

创建服务鉴权规则

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 安全治理 > 服务鉴权**。
3. 在**服务鉴权**页面单击**创建规则**。
4. 在**创建规则**页面设置服务鉴权参数，然后单击**确定**。

   服务鉴权规则参数说明：

参数	说明
规则名称	鉴权规则名称，支持大小写字母、数字、下划线（_）和短划线（-），长度不超过64个字符。
被调用方类型	根据实际情况选择*应用*或*K8s Namespace*。
被调用方（应用）	当**被调用方类型**选择**应用**时，选择被调用的应用。
被调用方（K8s Namespace）	当**被调用方类型**选择**K8s Namespace**时，选择被调用的应用集群和所在的命名空间。
被调用方框架	被调用的应用所使用的框架，根据需要选择**Spring Cloud**或者**Dubbo**。
添加所有接口规则 **注意** 所有接口的通用规则仅支持添加一次。	
被调用方接口	默认为**所有Path**，不可设置。 **说明** 此参数仅适用于Spring Cloud。
被调用方Path	默认为**所有服务/所有接口**，不可设置。 **说明** 此参数仅适用于Dubbo。
鉴权方式	服务鉴权的方式，包含**白名单（允许调用）**和**黑名单（拒绝调用）**，请根据实际鉴权需求选择。
调用方	需要鉴权的调用方应用，可以单击**添加调用方**设置多个需要鉴权的调用方应用。
添加指定接口规则 **注意** 指定接口添加的规则不是追加，而是覆盖针对所有接口的通用规则，请谨慎配置。	

参数	说明
被调用方Path	指定被调用应用的Path。 **说明** 此参数仅适用于Spring Cloud。
被调用方接口	指定被调用应用的服务和接口。 **说明** 此参数仅适用于Dubbo。
鉴权方式	服务鉴权的方式，包含**白名单（允许调用）**和**黑名单（拒绝调用）**，请根据实际鉴权需求选择。
调用方	需要鉴权的调用方应用，可以单击**添加调用方**设置多个需要鉴权的调用方应用。
默认状态	规则的启用开关。 ○ 打开：创建后即启用，默认打开。 ○ 关闭：创建后不启用，如果需要启用，请在**服务鉴权**页面规则的**操作**列单击**开启**。

结果验证

服务鉴权规则配置完成且开启后，请根据实际业务验证服务鉴权规则是否生效。

相关操作

服务鉴权规则创建完成后，您还可以编辑规则、根据规则的不同状态关闭规则或开启规则。当不再需要服务鉴权时，删除规则。

7.系统设置

7.1. 升级MSE微服务治理组件

为了您能够使用最新的MSE服务治理功能，建议您及时升级MSE微服务治理组件。原微服务治理组件为ack-mse-pilot，现在的微服务治理组件为ack-onepilot，本文介绍如何升级ack-mse-pilot组件至ack-onepilot组件。

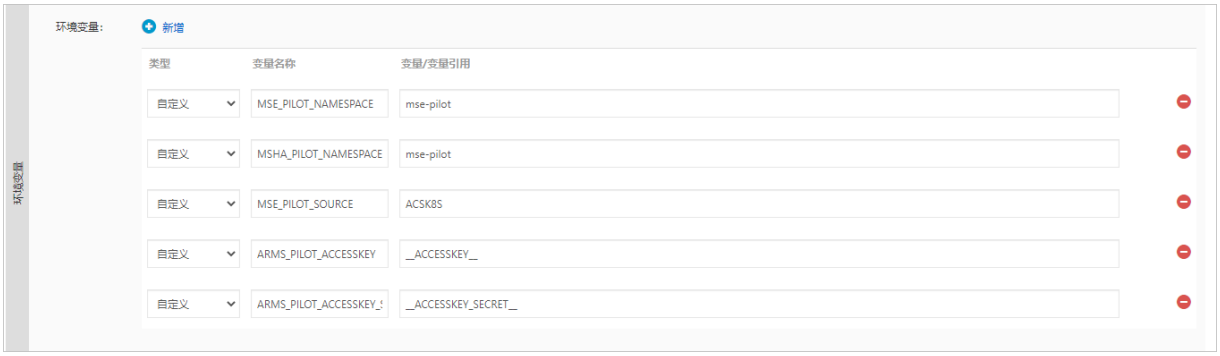
背景信息

MSE微服务治理组件ack-mse-pilot升级为ack-onepilot需要以下操作：

1. 备份ack-mse-pilot组件的环境变量。
2. 安装ack-onepilot组件，必要时设置从ack-mse-pilot组件备份出来的环境变量，请参见安装ack-onepilot组件。
3. 卸载ack-mse-pilot，请参见卸载ack-mse-pilot组件。

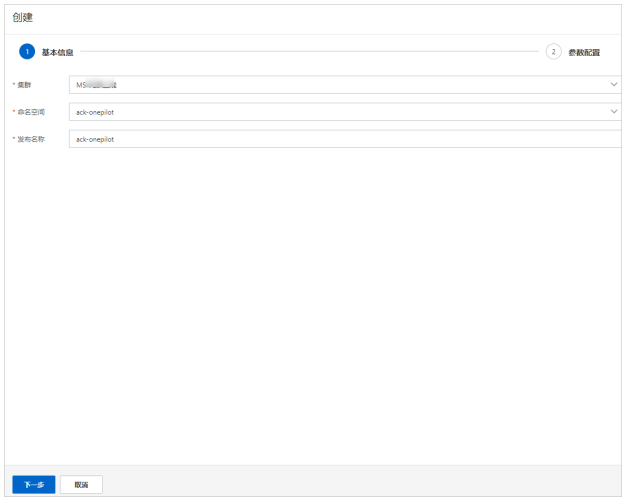
安装ack-onepilot组件

建议您在升级ack-mse-pilot组件至ack-onepilot组件之前，先记录该组件下的**mse-pilot-ack-mse-pilot**应用的全部环境变量，便于升级后验证。



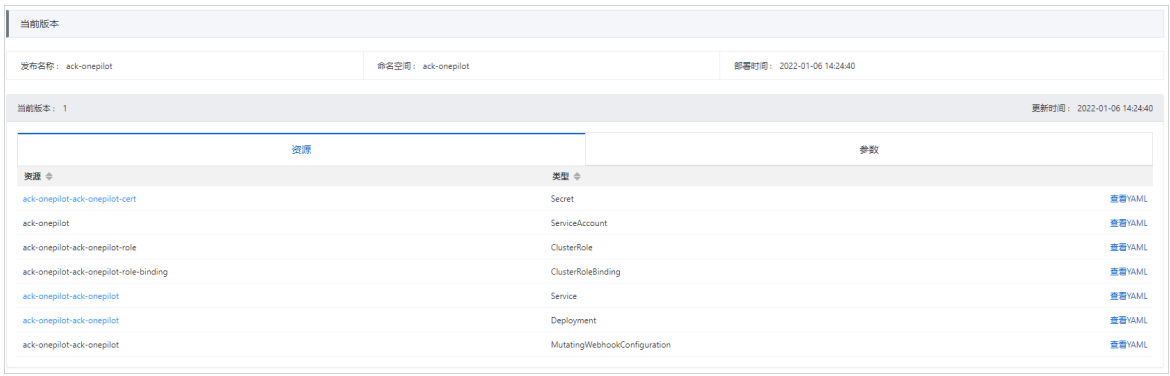
安装ack-onepilot组件操作如下：

1. 登录容器服务控制台。
2. 在左侧导航栏单击**市场 > 应用目录**。
3. 在**应用目录**页面搜索并单击*ack-onepilot*。
4. 在**ack-onepilot**页面右上方单击**一键部署**，在**创建**面板中选择**集群**和**命名空间**，设置组件**发布名称**，然后单击**下一步**。



> **说明** 建议使用默认的命名空间ack-onepilot。

5. 在**参数配置**向导中确认组件参数信息，然后单击**确定**。
   安装MSE微服务治理组件大约需要2分钟，请耐心等待。
   创建成功后，会自动跳转到目标集群的**发布**页面，检查安装结果。如果出现以下页面，展示相关资源，则说明安装成功。



卸载ack-mse-pilot组件

ack-onepilot组件安装完成后，请卸载ack-mse-pilot组件。

1. 在集群管理页左侧导航栏中，选择**应用 > Helm**。
2. 单击**mse-pilot**应用**操作**列下方的**删除**。
3. 在确认**删除应用**对话框中单击**确定**。

7.2. 关闭MSE微服务治理

MSE微服务治理商业化后，若您不再继续使用MSE服务治理中心，建议您及时关闭MSE微服务治理，避免对您的应用造成影响。本文介绍如何关闭MSE微服务治理。

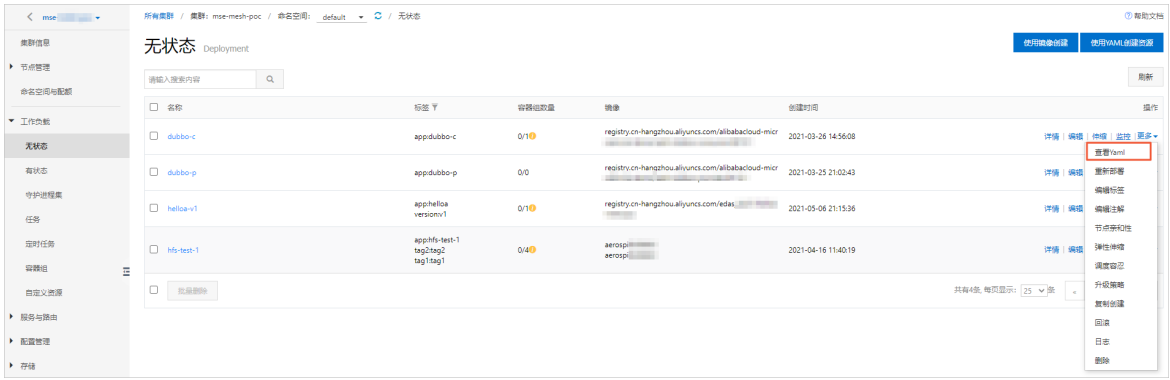
关闭流程

关闭MSE微服务治理包含以下步骤：

1. 为已有应用关闭MSE微服务治理
2. 在ACK中卸载MSE治理中心组件
3. 在MSE中删除微服务应用

为应用关闭MSE微服务治理

1. 登录容器服务控制台。
2. 在左侧导航栏单击**集群**，然后在**集群列表**页面单击目标集群的集群名称。
3. 在集群信息左侧导航栏选择**工作负载 > 无状态**。
4. 在**无状态**页面左上角选择**命名空间**，并在目标应用的**操作**列中单击**更多**，在列表中单击**查看Yaml**。



5. 在**编辑YAML**对话框的**spec > template > metadata**中找到annotations，删除 `msePilotAutoEnable: "on"` 或者改为 `msePilotAutoEnable: "off"` ，然后单击**更新**。

```
annotations:
  msePilotAutoEnable: "off"
  msePilotCreateAppName: "<your-deployment-name>"
```

在ACK中卸载MSE治理中心组件

1. 在集群信息页面左侧导航栏选择**应用 > Helm**。
2. 在**Helm**页面单击**mse-pilot**组件**操作**列下方的**删除**。
3. 在**删除应用**对话框中单击**确定**即可卸载MSE治理中心组件。

在MSE中删除微服务应用

1. 登录MSE治理中心控制台。
2. 在左侧导航栏选择**微服务治理中心 > 应用信息 > 应用列表**。
3. 在**应用列表**页面单击目标应用**操作**列下方的**删除**。
4. 在确认删除对话框中单击**确认**。

执行结果

完成上述步骤后，您就为部署在容器服务Kubernetes版ACK中的应用关闭了MSE微服务治理能力。