



21天微服务实战营-Day9

华为云DevCloud & ServiceStage服务联合出品



Day9 CSE实战之服务治理

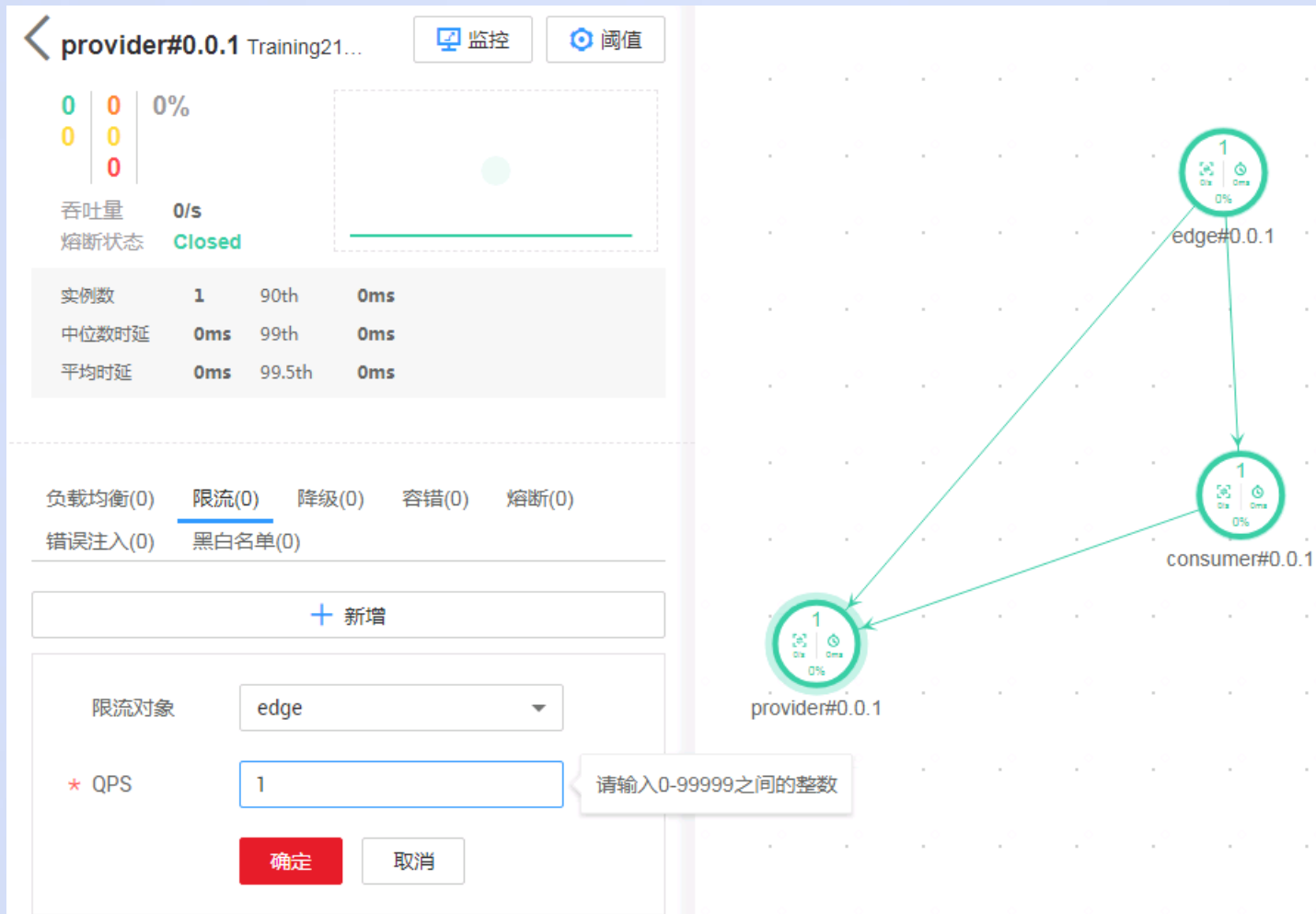
大纲

- 限流策略
- 服务熔断
- 服务降级
- 灰度发布

限流策略

- CSEJavaSDK支持客户端和服务端的限流策略，限制每秒钟最大请求数
- 支持default、微服务、schema（ 契约 ）、operation（ 接口 ）限流四个粒度的限流
- 治理页面支持的是服务端微服务级限流配置
- 限流策略是实例级限流，例如，配置provider服务的服务端流控为1000QPS，如果有两个provider服务实例，则他们可以接收共计2000QPS的流量

限流策略

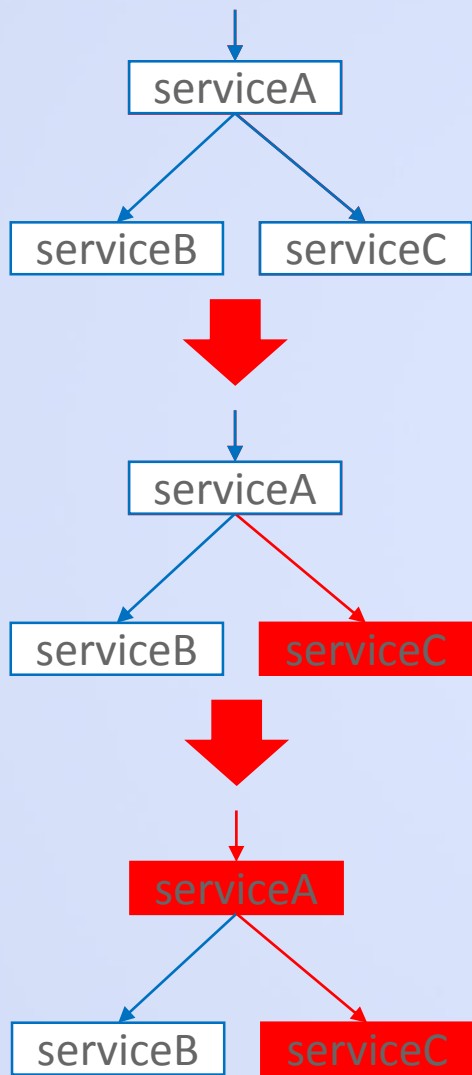


在provider服务上配置限流策略为对edge服务限流至1QPS，通过edge服务连续调用consumer和provider服务。

观察provider服务中打印的日志，可以看到每秒钟provider只处理一次来自edge服务的请求，其他请求都以429状态码返回的。而对于来自consumer的请求，provider正常处理并返回200状态码。

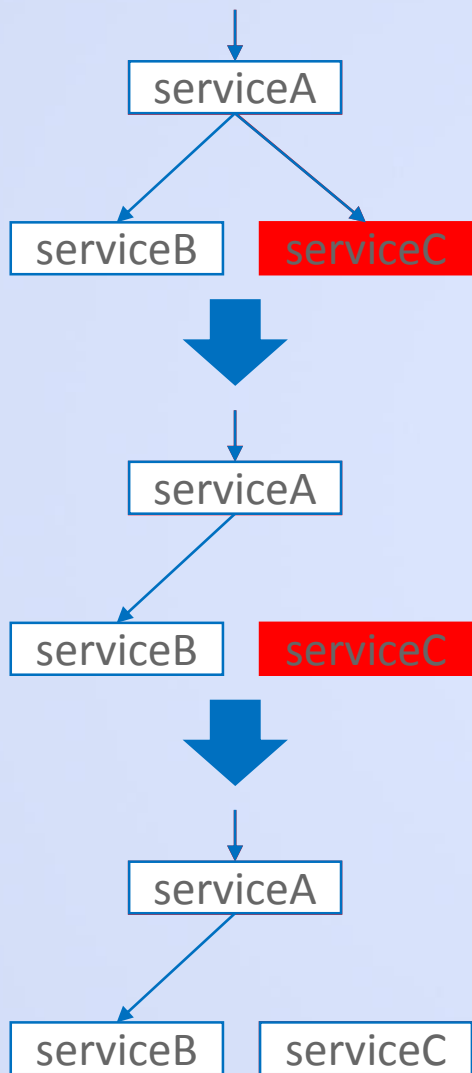
```
09:44:37 CST /provider/v0/hello/Alice 200 13 2 5c75eb8566ff9bf7
09:44:38 CST /provider/v0/hello/Bob 200 11 1 5c75eb86c817a894 01
09:44:38 CST /provider/v0/hello/Alice 200 13 1 5c75eb866c170711
09:44:38 CST /provider/v0/hello/Bob 429 41 1 5c75eb86281fa768 01
09:44:38 CST /provider/v0/hello/Alice 200 13 1 5c75eb868440acd4
09:44:38 CST /provider/v0/hello/Bob 429 41 1 5c75eb860ea48971 01
09:44:38 CST /provider/v0/hello/Alice 200 13 2 5c75eb867246b63d
09:44:38 CST /provider/v0/hello/Bob 429 41 0 5c75eb86255404de 01
09:44:38 CST /provider/v0/hello/Alice 200 13 2 5c75eb86f17d2d98
09:44:38 CST /provider/v0/hello/Bob 429 41 0 5c75eb86f6becc4a4 01
09:44:38 CST /provider/v0/hello/Alice 200 13 1 5c75eb86a680379a
09:44:38 CST /provider/v0/hello/Bob 429 41 1 5c75eb8686971aac 01
09:44:39 CST /provider/v0/hello/Alice 200 13 1 5c75eb8762f30c07
09:44:39 CST /provider/v0/hello/Bob 200 11 1 5c75eb87567287cd 01
09:44:39 CST /provider/v0/hello/Alice 200 13 1 5c75eb877cd27728
09:44:39 CST /provider/v0/hello/Bob 429 41 1 5c75eb8715293b46 01
09:44:39 CST /provider/v0/hello/Alice 200 13 1 5c75eb87d02ef8c3
09:44:39 CST /provider/v0/hello/Bob 429 41 1 5c75eb87a133c166 01
```

服务熔断

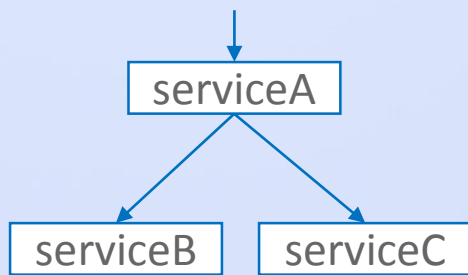


- 对于一个分布式系统，如果某个请求的调用链中的某个服务出现故障，响应变慢，会导致整个链路的响应变慢，请求堆积。
- 当这种情况变得越来越严重的时候，占用的资源会越来越多，到达系统瓶颈，造成整个系统崩溃，所有请求都不可用。

服务熔断



- 熔断可以将问题服务隔离开，令请求可以快速返回
 - 待问题服务变为正常状态后，再从熔断状态中恢复过来
- 通过这种机制，我们可以临时断开次要业务路径，保障系统整体的可用性。



服务熔断——手动熔断

edge#0.0.1 Training21Days-...

监控

阈值

0 | 0 | 0%

0 | 0 | 0%

0 | 0 | 0%

吞吐量 0/s

熔断状态 Closed

实例数	1	90th	0ms
中位数时延	0ms	99th	0ms
平均时延	0ms	99.5th	0ms

负载均衡(0)

限流(0)

降级(0)

容错(0)

熔断(0)

错误注入(0)

黑白名单(0)

+ 新增

熔断对象

consumer

helloConsumer.sayHello

触发条件

☒ 手动熔断

☐ 取消熔断

☐ 自动熔断

确定

取消



熔断可以手动开启，也可以自动开启。其触发方式不同，但效果相同。

这里我们选择edge服务，手工熔断其调用consumer服务的sayHello方法的路径。

服务熔断——手动熔断

edge#0.0.1 Training21Days-...

监控

阈值

0 | 0 | 0%

0 | 0 | 0%

0 | 0 | 0%

吞吐量 0/s

熔断状态 Closed

实例数	1	90th	0ms
中位数时延	0ms	99th	0ms
平均时延	0ms	99.5th	0ms

负载均衡(0)

限流(0)

降级(0)

容错(0)

熔断(0)

错误注入(0)

黑白名单(0)

+ 新增

熔断对象

consumer

helloConsumer.sayHello

触发条件

☒ 手动熔断 ☐ 取消熔断

☐ 自动熔断

确定

取消



熔断可以手动开启，也可以自动开启。其触发方式不同，但效果相同。

这里我们选择edge服务，手工熔断edge服务调用consumer服务sayHello方法的路径。

服务熔断——手动熔断

通过edge服务调用consumer服务的sayHello方法和greeting方法，可以看到sayHello方法已经调不通了，但greeting方法仍然能够调通

The screenshot displays two sequential API requests in a REST client interface.

Request 1:

- Method: GET
- URL: 127.0.0.1:8000/rest/consumer/v0/hello?name=Alice
- Status: 490 Cse Internal Bad Request
- Time: 53 ms
- Size: 146 B
- Body (JSON):

```
{  "message": "Cse Internal Bad Request"}
```

Request 2:

- Method: POST
- URL: 127.0.0.1:8000/rest/consumer/v0/greeting
- Status: 200 OK
- Time: 17 ms
- Size: 151 B
- Body (JSON):

```
{  "msg": "Hello, Ms.Wilson",  "timestamp": "2019-02-27T04:50:30.911Z"}
```

服务熔断——自动熔断

edge#0.0.1 Training21Days...

监控 阈值

0%
吞吐量 0/s
熔断状态 Closed

实例数	1	90th	0ms
中位数时延	0ms	99th	0ms
平均时延	0ms	99.5th	0ms

负载均衡(0) 限流(0) 降级(0) 容错(0) 熔断(0)

错误注入(0) 黑白名单(0)

+ 新增

熔断对象
所有微服务
所有方法

触发条件
☐ 手动熔断 ☐ 取消熔断
☒ 自动熔断

* 熔断时间窗
(ms): 30,000

* 失败率(%): 50

* 窗口请求数: 3

确定 取消

我们也可以在页面上配置自动熔断规则。

默认的自动熔断规则需要在10秒内存在20次以上的调用，错误率达到50%时触发熔断，在我们的实验中较难触发。

这里我们将熔断规则改为：

- 熔断效果维持30秒
- 错误率达到50%时触发熔断
- 10秒内有3个以上的请求时开始判断错误率

服务熔断——自动熔断

微服务
edge

所属应用
Training21Days-HelloWo...

状态
● 在线

实例
正常 1 异常 0

创建配置

导入

全部导出

所有作用域

配置项或值

作用域	配置项	值	操作
edge@Training21Days-HelloWorld#0.0.1	cse.circuitBreaker.Consumer.requestVolumeThreshold	3	编辑 删除
edge@Training21Days-HelloWorld#0.0.1	cse.circuitBreaker.Consumer.errorThresholdPercentage	50	编辑 删除
edge@Training21Days-HelloWorld#0.0.1	cse.circuitBreaker.Consumer.sleepWindowInMilliseconds	30000	编辑 删除
edge@Training21Days-HelloWorld#0.0.1	cse.circuitBreaker.Consumer.forceOpen	false	编辑 删除
edge@Training21Days-HelloWorld#0.0.1	cse.circuitBreaker.Consumer.forceClosed	false	编辑 删除
edge@Training21Days-HelloWorld#0.0.1	cse.circuitBreaker.Consumer.enabled	true	编辑 删除
edge@Training21Days-HelloWorld	cse.loadbalance.isolation.enabled	false	编辑 删除

为了避免实例隔离机制对熔断现象的干扰，我们这里将实例隔离关闭。

TIPS：这里对熔断机制、隔离机制的调整是为了更方便地展示熔断的现象，并不是推荐的配置。如果您对于微服务熔断能力还不熟悉，建议先维持默认配置，在实践中根据实际需求再进行调整。

服务熔断——自动熔断

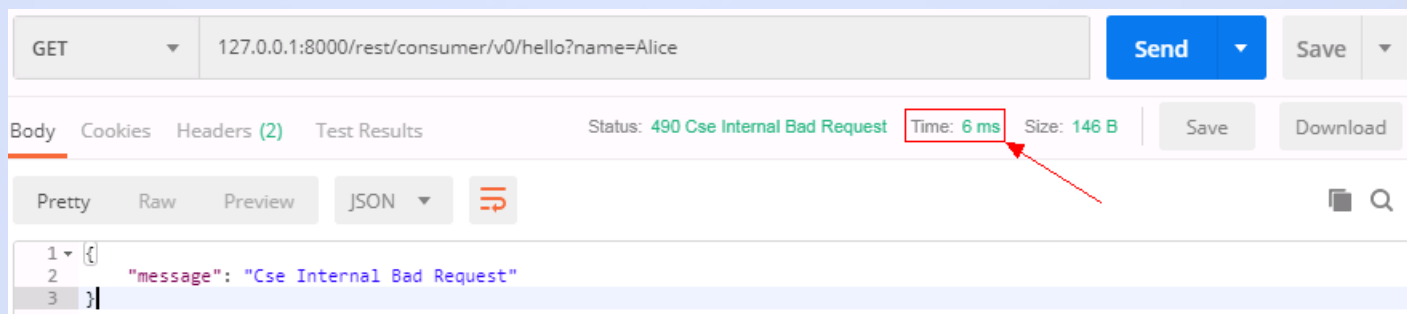
```
private DynamicLongProperty helloDelay = DynamicPropertyFactory.getInstance().getLongProperty(propName: "delay.sayHello", defaultValue: 0);

@Path("/hello")
@GET
public String sayHello(@QueryParam("name") String name) {
    // RPC 调用方式体验与本地调用相同
    if (helloDelay.get() > 0) {
        try {
            Thread.sleep(helloDelay.get());
        } catch (InterruptedException e) {
            LOGGER.error("sayHello sleeping is interrupted!", e);
        }
    }
    return helloService.sayHello(name);
}
```

我们在consumer服务的sayHello方法里增加线程sleep的逻辑

启动provider、consumer、edge服务后，在配置中心设置delay.sayHello=40000，多次发请求通过edge调用consumer的sayHello方法，触发edge服务到consumer服务sayHello方法请求路径的熔断。

可以看到熔断发生后，edge调用consumer的sayHello方法不会等待30秒超时后再返回错误，而是立即就返回错误了



服务熔断——自动熔断



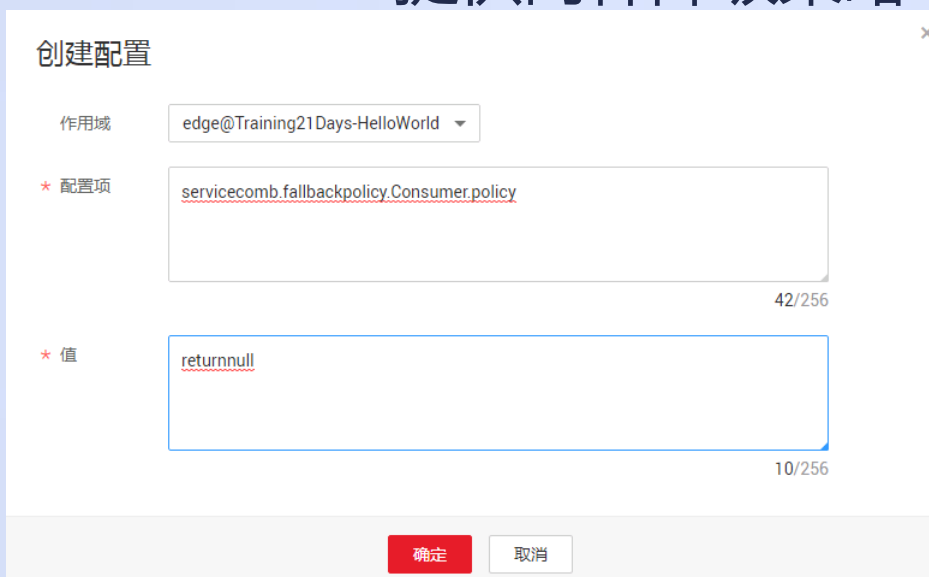
- 熔断发生后，edge服务的熔断状态变为“Open”（可能会延迟几秒，微服务实例上报状态到monitor有时间间隔）
- 熔断期间，通过edge调用consumer服务的方法仍然是通的，只有consumer服务的sayHello方法被熔断
- 修改consumer服务的delay.sayHello配置项为0，等熔断时间窗过后，再次调用consumer的sayHello方法，edge的熔断状态变回“Closed”
- 注意，edge的熔断时间窗结束后，需要等到edge调用consumer的sayHello方法成功，熔断状态才会结束。否则页面上仍然会一直显示熔断状态为“Open”

服务降级

这里讲的降级策略是指服务调用出错时的处理策略，可以对应参考ServiceComb文档资料中的[降级策略](#)的容错配置。

CSEJavaSDK提供两种降级策略，即抛出异常和返回null，默认为抛出异常

为了让降级的效果更明显，我们将降级策略修改为returnnull



创建配置

作用域: edge@Training21Days-HelloWorld

* 配置项: servicecomb.fallbackpolicy.Consumer.policy (42/256)

* 值: returnnull (10/256)

确定 取消

服务降级



← 请求超时

可以看到，当edge调用consumer超时时，返回的响应不再是490错误，而是200，消息体为空
熔断发生后，edge服务返回的也是空消息体



← 触发熔断

灰度发布

- CSEJavaSDK支持灰度发布功能，可以实现版本平滑过渡升级
- 支持按百分比引流和按请求参数特征引流两种方式

```
// for microservice version 0.0.1
// @RequestMapping(path = "/hello/{name}", method = RequestMethod.GET)
// public String sayHello(@PathVariable(value = "name") String name) {
//     return sayHelloPrefix.getValue() + name;
// }

// for microservice version 0.0.2
@RequestMapping(path = "/hello/{name}", method = RequestMethod.GET)
public String sayHello(@PathVariable(value = "name") String name) {
    return sayHelloPrefix.getValue() + name + "." + generateGreeting();
}

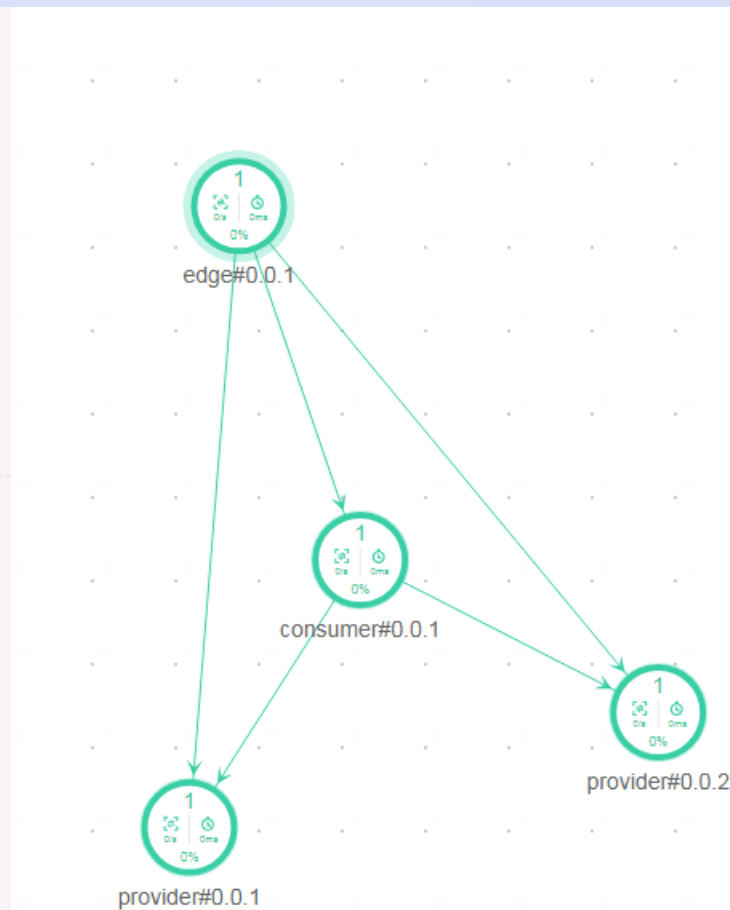
private String generateGreeting() {
    Calendar calendar = new GregorianCalendar();
    int hourOfDay = calendar.get(Calendar.HOUR_OF_DAY);
    if (hourOfDay < 12) {
        return "Good morning.";
    }
    if (hourOfDay < 18) {
        return "Good afternoon.";
    }
    if (hourOfDay < 22) {
        return "Good evening.";
    }
    return "Good night.";
}
```

为了验证灰度发布的效果，我们开发一个0.0.2版本的provider服务，它的sayHello方法会根据时间返回不同的问候信息

修改sayHello方法后，升级provider服务为0.0.2版本来启动provider服务

```
APPLICATION_ID: Training21Days-HelloWorld
service_description:
  name: provider
  #version: 0.0.1
  version: 0.0.2
```


灰度发布



将新旧版本的provider服务各启动一个实例，在治理页面看到的微服务情况如左图所示

此时通过edge连续调用sayHello方法，应该是新旧两个版本的应答交替返回，即两个版本的provider服务实例都是可用的服务实例，在轮询策略下依次被调用

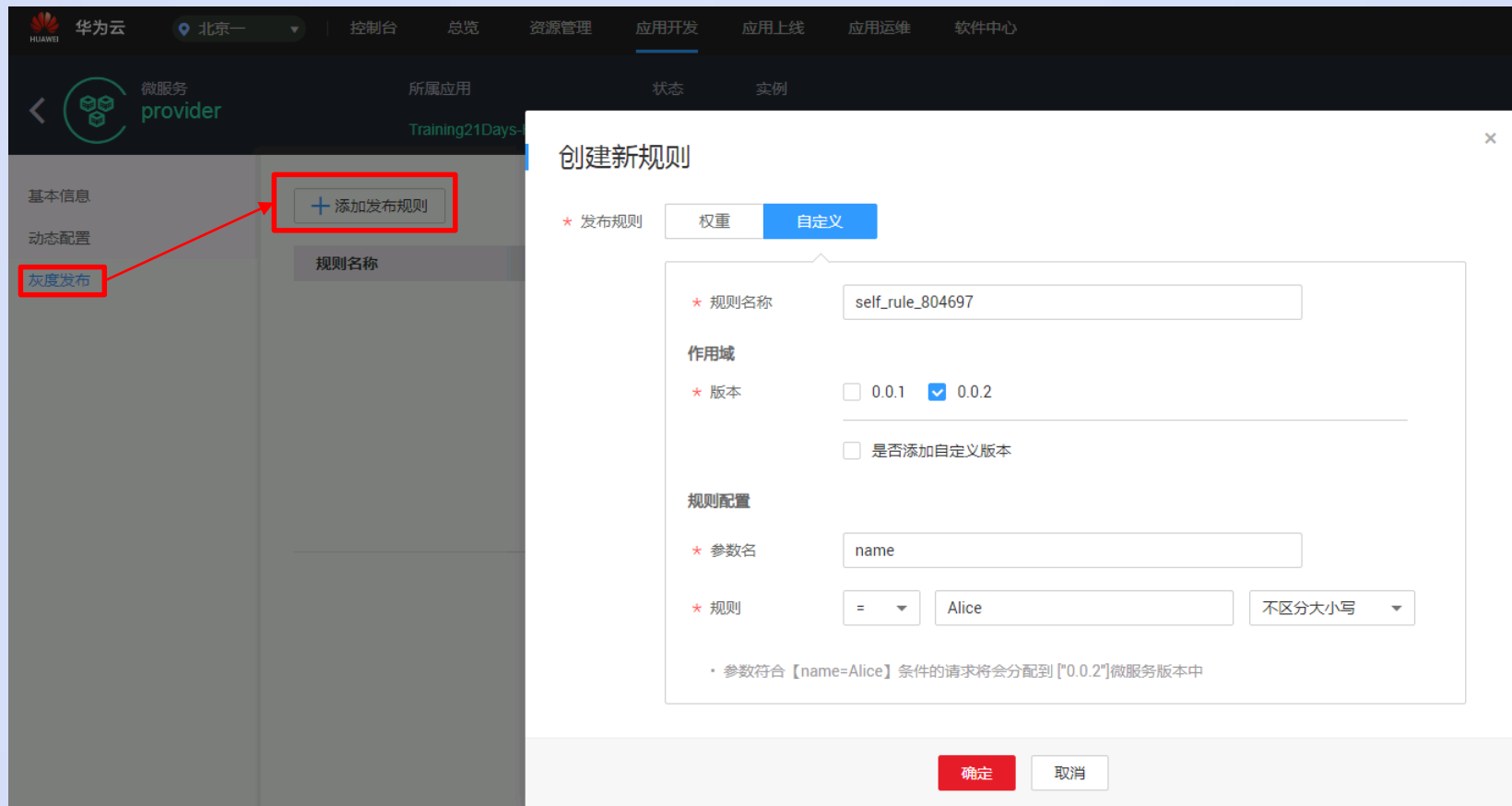
127.0.0.1:8000/rest/consumer/v0/hello?name=Alice

"Hello Alice. Good evening."

127.0.0.1:8000/rest/consumer/v0/hello?name=Alice

"Hello Alice"

灰度发布



进入provider服务的详情页面->灰度发布页面，点击添加发布规则，添加如左图所示的自定义灰度规则

灰度发布

The image displays three sequential screenshots of a REST client interface, showing the process of testing a hello service with different names.

First Screenshot:

- Method: GET
- URL: 127.0.0.1:8000/rest/consumer/v0/hello?name=Alice
- Status: 200 OK
- Time: 8 ms
- Size: 114 B
- Response Body: "Hello Alice. Good evening."

Second Screenshot:

- Method: GET
- URL: 127.0.0.1:8000/rest/consumer/v0/hello?name=Bob
- Status: 200 OK
- Time: 32 ms
- Size: 97 B
- Response Body: "Hello Bob"

Third Screenshot:

- Method: GET
- URL: 127.0.0.1:8000/rest/consumer/v0/hello?name=Cara
- Status: 200 OK
- Time: 15 ms
- Size: 98 B
- Response Body: "Hello Cara"

再次调用sayHello方法，可以看到当name=Alice时，请求始终是由0.0.2版本的provider服务处理的；而name为其他参数时，请求都是由0.0.1版本的provider服务处理的。

Thank You

