HUAWEI

21天微服务实战营-Day8

华为云DevCloud & ServiceStage服务联合出品

Day8 CSE实战之负载均衡

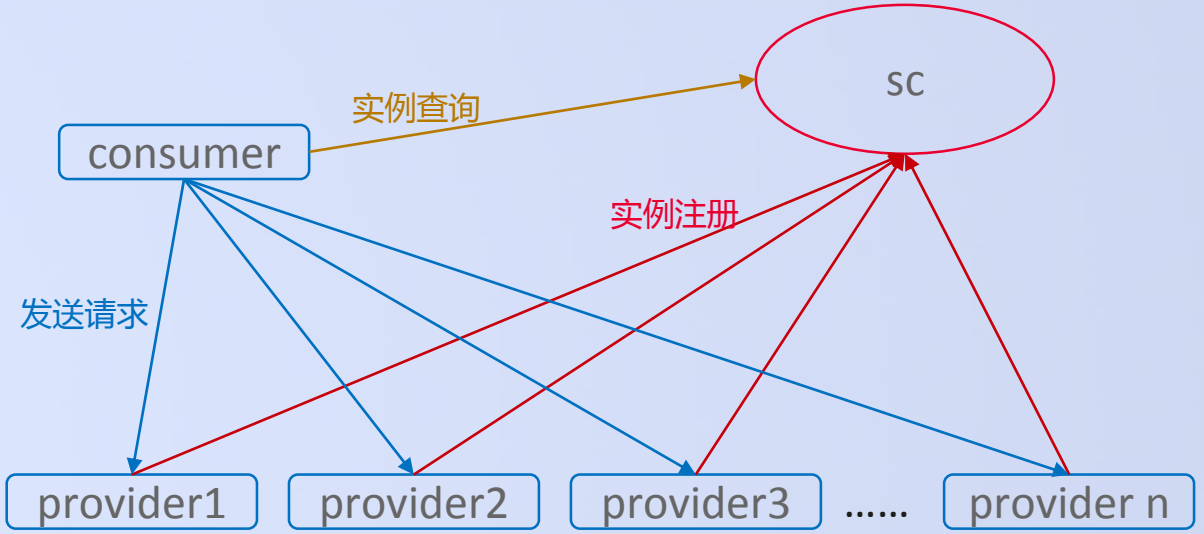
大纲

- 负载均衡策略
- 请求重试机制
- 实例隔离机制

负载均衡策略

CSEJavaSDK的负载均衡机制是客户端负载均衡：

- 微服务实例启动时会将自己的实例信息（包括IP、端口号等）注册到sc，并且通过心跳机制维持本实例的在线状态。
- consumer定时去服务中心查询provider的实例（第一次查询发生在consumer第一次调用provider的时候，之后默认30秒去sc查一次），并缓存在本地。
- consumer调用provider时会通过负载均衡机制从缓存的provider实例列表中选取一个作为本次请求发送的地址。
- 内置的负载均衡策略有RoundRobin、Random、WeightedResponse、SessionStickiness，其中默认使用的是RoundRobin。



负载均衡策略

启动一个consumer实例，三个provider实例，调用consumer的greeting方法，可以看到consumer默认是使用RoundRobin策略调用三个provider的。连续调用consumer6次，三个provider的日志记录分别是：

```
2019-02-25 11:38:30,347 [INFO] 192.168.0.45 - - Mon, 25 Feb 2019 11:38:30 CST /provider/v0/greeting 200 65 8 5c73633634a31c51
2019-02-25 11:38:33,615 [INFO] 192.168.0.45 - - Mon, 25 Feb 2019 11:38:33 CST /provider/v0/greeting 200 65 3 5c73633902d84f8a
```

实例1

```
2019-02-25 11:38:31,495 [INFO] 192.168.0.45 - - Mon, 25 Feb 2019 11:38:31 CST /provider/v0/greeting 200 65 3 5c7363376ed08941
2019-02-25 11:38:34,674 [INFO] 192.168.0.45 - - Mon, 25 Feb 2019 11:38:34 CST /provider/v0/greeting 200 65 3 5c73633a652f94c0
```

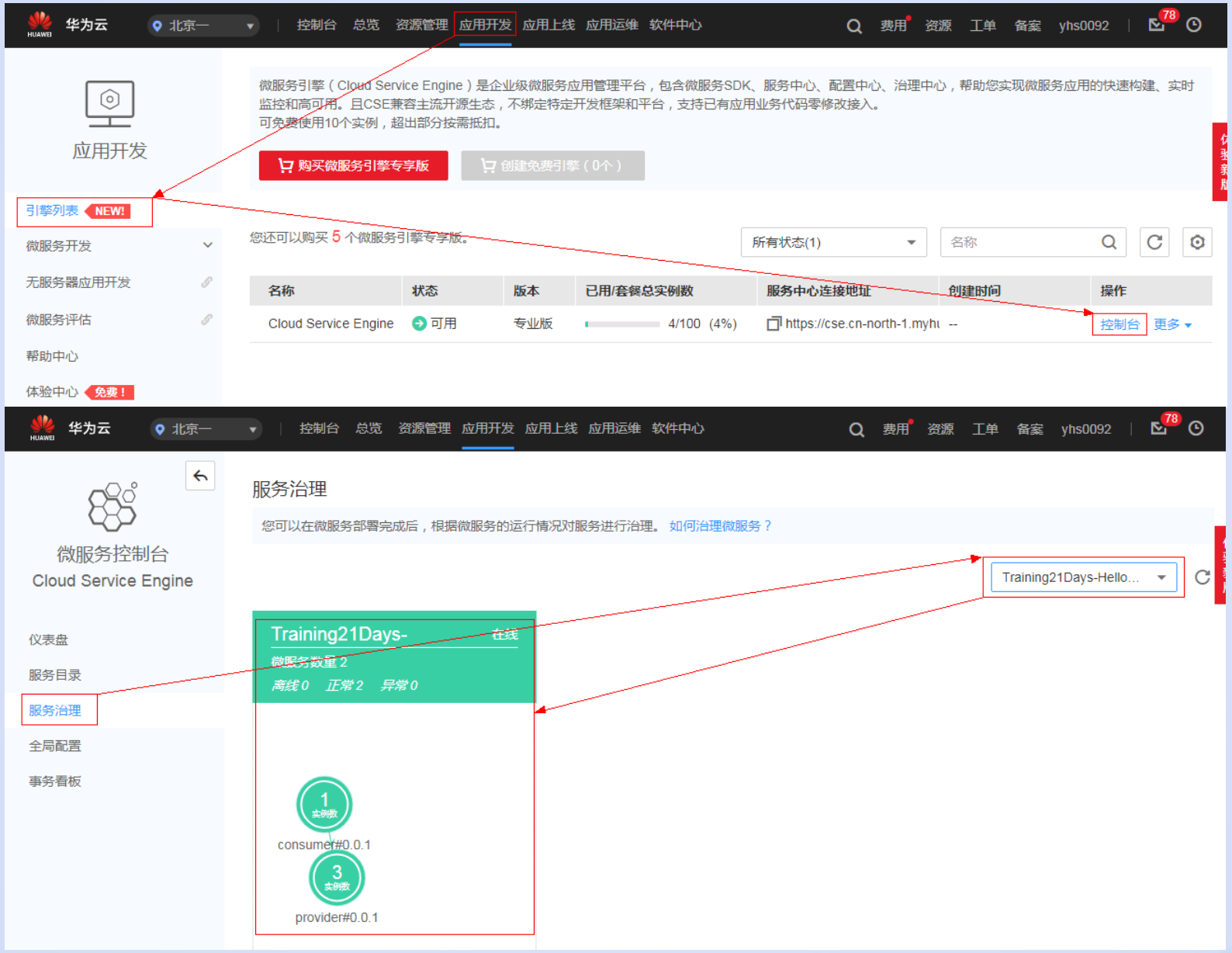
实例2

```
2019-02-25 11:38:32,625 [INFO] 192.168.0.45 - - Mon, 25 Feb 2019 11:38:32 CST /provider/v0/greeting 200 65 2 5c736338a0353b2e
2019-02-25 11:38:35,697 [INFO] 192.168.0.45 - - Mon, 25 Feb 2019 11:38:35 CST /provider/v0/greeting 200 65 3 5c73633b85f29c50
```

实例3

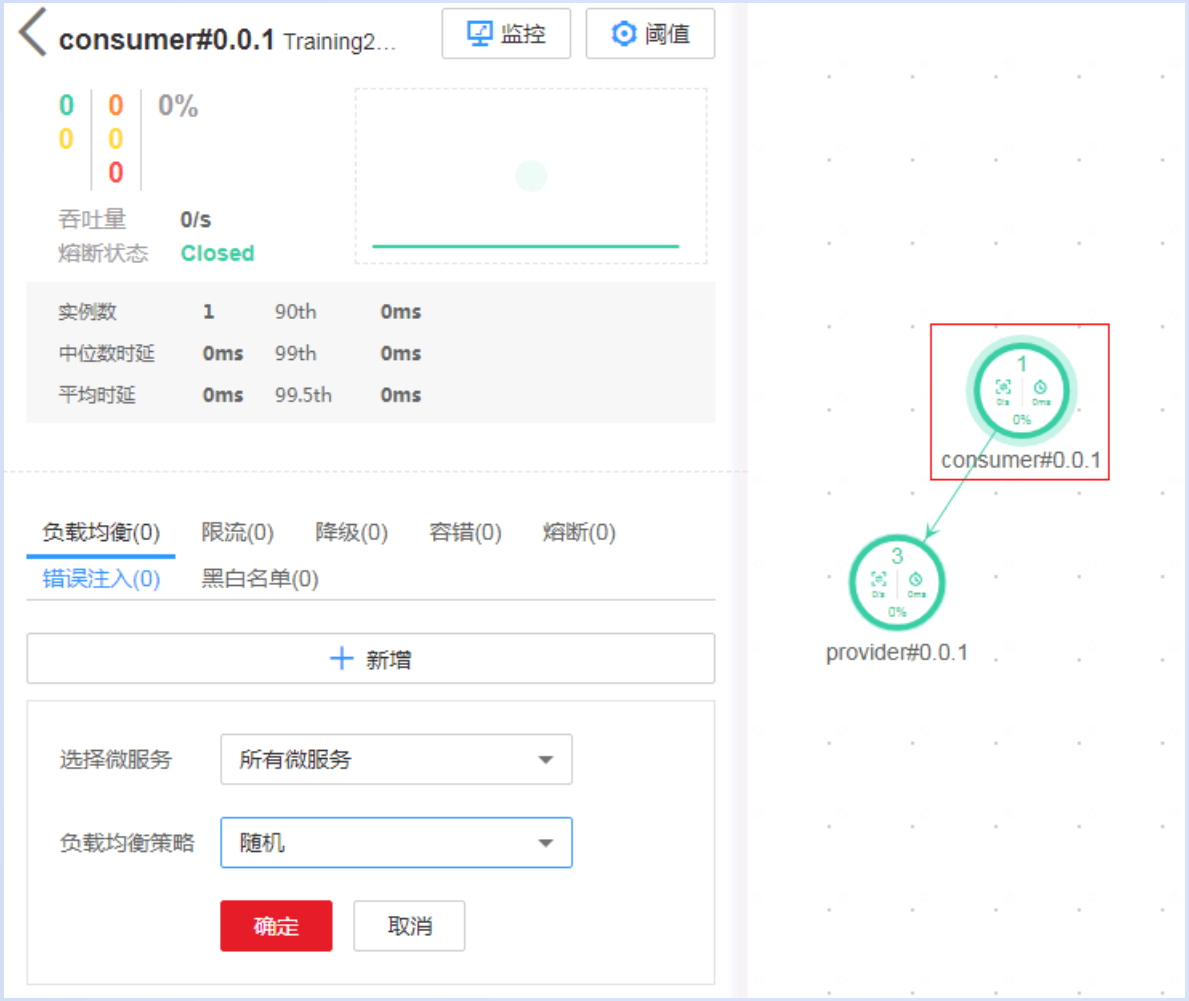
从时间顺序上可以看出，consumer服务的6次请求是由三个provider实例轮流处理的。

负载均衡策略



登陆ServiceStage页面，依次点击“应用开发”->“引擎列表”->“控制台”->“服务治理”，在应用选择下拉框里选择“Training21Days-HelloWorld”，进入该应用的治理页面

负载均衡策略



在服务治理页面的右边点击选择consumer服务，将其负载均衡策略修改为“随机”

重新调用consumer的greeting方法，可以看到三个provider实例接收greeting请求的分布情况变为随机的了

请求重试机制

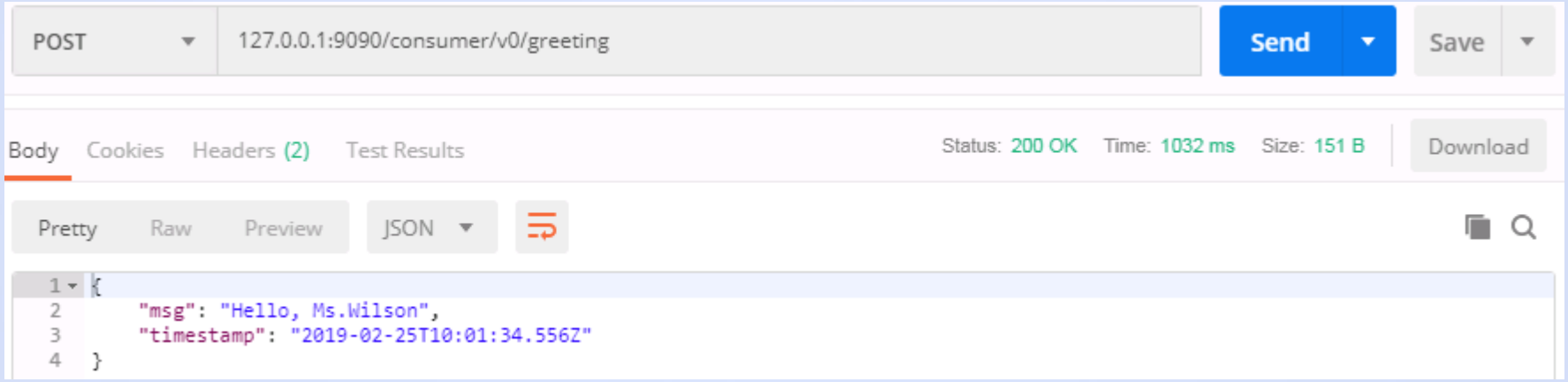
当遭遇网络连接超时、实例下线等问题时，consumer服务默认会尝试选择下一个provider服务实例进行调用，来尽量保证业务调用的成功。

- 默认的重试机制配置是retryOnSame=0,retryOnNext=1，即在同一个provider实例上重试零次，选择下一个provider实例重试一次
- 并不是所有的错误都会让consumer端进行重试，具体判断逻辑请参考ServiceComb-Java-Chassis中的DefaultRetryExtensionsFactory类
- 默认的重试机制配置在cse-solution-service-engine包的microservice.yaml文件中

请求重试机制

我们可以模拟一下实例意外停止的场景来测试重试机制。

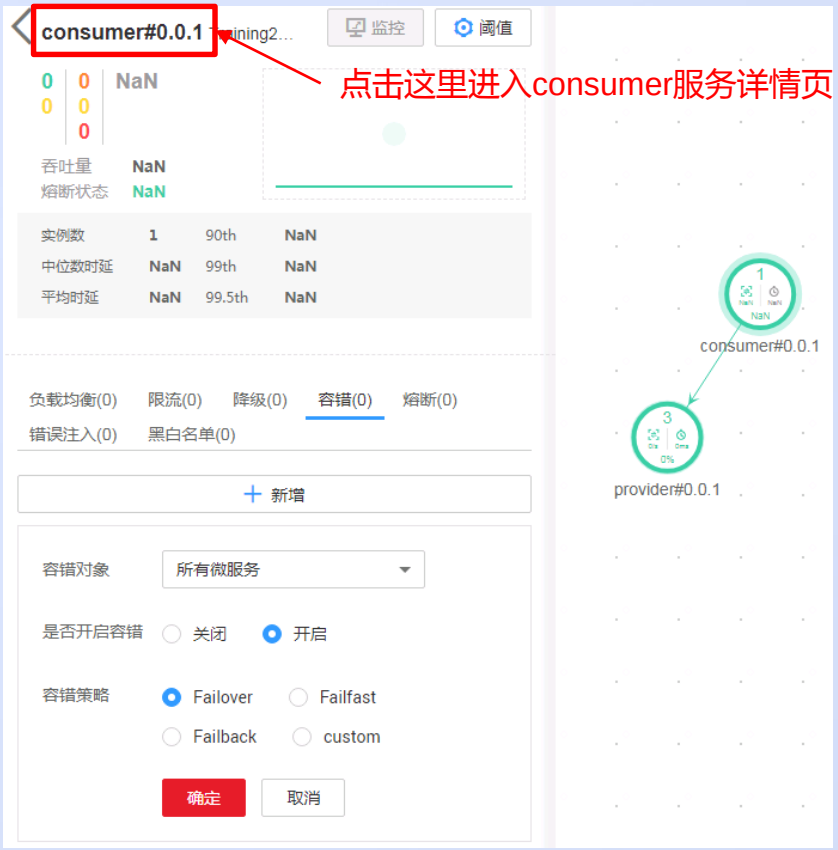
强制停止一个provider实例（注意不要是正常退出，否则触发实例优雅停机的话我们的可用时间窗口太短），然后调用consumer的greeting方法：



可以看到，虽然有的请求花费的时间长一点，但是仍然会调用成功的。如果观察consumer服务的日志，可以发现有时候consumer会选取那个被强制关闭的provider实例，但是在抛出 java.net.ConnectException: Connection refused 异常后，consumer会向另一个provider实例发送请求并调用成功。

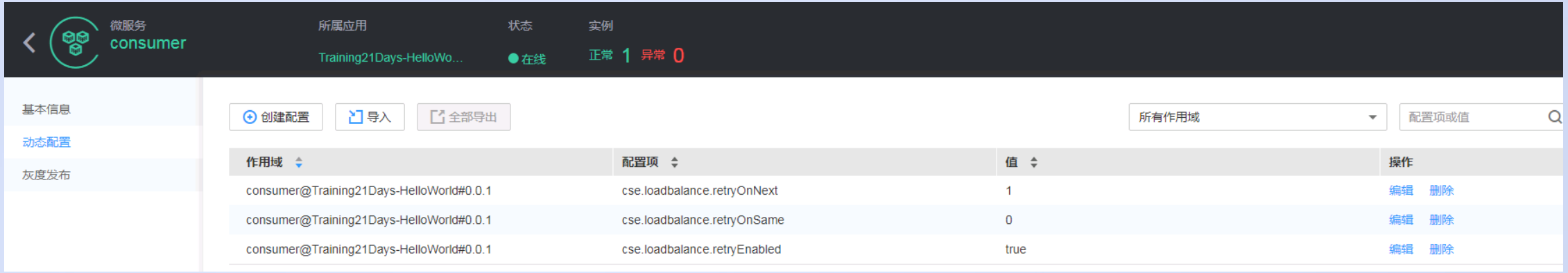
请求重试机制

在治理页面上我们可以动态地调整重试机制的配置，治理页面的重试机制名为容错。大家可以在页面上选择一下容错策略为Failover，前往consumer服务的详情页面查看配置项



请求重试机制

从配置项页面可以看到，Failover策略就是默认的tryOnSame=0，tryOnNext=1策略。



Tips: 大家可以在页面上尝试一下其他的几种重试策略。CSE的大部分治理策略都是通过配置项管理的，在治理页面上进行治理操作后，可以到这里查看对应的配置项。

实例隔离机制

如果consumer在调用某个provider实例时频繁报错，则consumer会将该provider实例隔离，等待一段时间后再尝试向其发送请求。

- 默认隔离规则是连续5次调用某实例失败后隔离该实例
- 默认隔离60秒后将被隔离实例放回可选provider实例列表中，但当次请求是否被路由到该实例仍取决于路由策略的选择
- 如果尝试对被隔离实例进行调用时失败了，则该实例立即重新进入隔离状态，等待下一个60秒重试机会；调用成功则从隔离状态恢复
- 并非所有的调用失败都会计入实例隔离的判断机制里

实例隔离机制

我们仍然启动一个consumer实例、三个provider实例，先调用consumer的方法，测试consumer会正常地将请求路由到三个provider实例上去。

接着强制关闭一个provider实例，连续调用consumer服务，可以看到consumer在调用强制关闭的provider实例几次之后，将这个实例隔离并且不再调用了。

```
Caused by: java.net.ConnectException: Connection refused: no further information
    ... 11 more
2019-02-26 11:52:20,410 [WARN] bizkeeper command CONSUMER rest provider.hello.greeting failed due to InvocationException: code=490;msg=CommonExceptionData [message=Cse Internal Bad Request] org.apache.servicecomb.bizkeeper.BizkeeperCommand.
2019-02-26 11:52:20,410 [WARN] bizkeeper execution error, info=cause:InvocationException,message:InvocationException: code=490;msg=CommonExceptionData [message=Cse Internal Bad Request];cause:AnnotatedConnectException,message:Connection refu
2019-02-26 11:52:20,410 [ERROR] service CONSUMER rest provider.hello.greeting, call error, msg is cause:InvocationException,message:InvocationException: code=490;msg=CommonExceptionData [message=Cse Internal Bad Request];cause:CseException,
 You can add fallback logic by catching this exception.
 info: operation=provider.hello.greeting.;cause:InvocationException,message:InvocationException: code=490;msg=CommonExceptionData [message=Cse Internal Bad Request];cause:AnnotatedConnectException,message:Connection refused: no further infor
2019-02-26 11:52:20,410 [ERROR] Invoke server failed. Operation CONSUMER rest provider.hello.greeting; server rest://192.168.0.45:8080; 0-0 msg cause:InvocationException,message:InvocationException: code=490;msg=CommonExceptionData [message=
 You can add fallback logic by catching this exception.
 info: operation=provider.hello.greeting.;cause:InvocationException,message:InvocationException: code=490;msg=CommonExceptionData [message=Cse Internal Bad Request];cause:AnnotatedConnectException,message:Connection refused: no further infor
2019-02-26 11:52:20,415 [ERROR] Invoke server success. Operation CONSUMER rest provider.hello.greeting; server rest://192.168.0.45:8081 org.apache.servicecomb.loadbalance.LoadbalanceHandler$3.onExecutionSuccess(LoadbalanceHandler.java:306)
2019-02-26 11:52:20,416 [INFO] 127.0.0.1 - - Tue, 26 Feb 2019 11:52:19 CST /consumer/v0/greeting 200 65 1011 5c74b7f3a20b5de3 org.apache.servicecomb.transport.rest.vertx.accesslog.impl.AccessLogHandler.lambda$handle$0(AccessLogHandler.java:
2019-02-26 11:52:21,251 [WARN] Isolate service provider's instance bda7e921396311e99b640255ac105554. org.apache.servicecomb.loadbalance.filter.IsolationDiscoveryFilter.allowVisit(IsolationDiscoveryFilter.java:142)
2019-02-26 11:52:21,257 [INFO] 127.0.0.1 - - Tue, 26 Feb 2019 11:52:21 CST /consumer/v0/greeting 200 65 7 5c74b7f532df3540 org.apache.servicecomb.transport.rest.vertx.accesslog.impl.AccessLogHandler.lambda$handle$0(AccessLogHandler.java:42)
2019-02-26 11:52:22,401 [INFO] 127.0.0.1 - - Tue, 26 Feb 2019 11:52:22 CST /consumer/v0/greeting 200 65 6 5c74b7f6803019a3 org.apache.servicecomb.transport.rest.vertx.accesslog.impl.AccessLogHandler.lambda$handle$0(AccessLogHandler.java:42)
2019-02-26 11:52:23,426 [INFO] 127.0.0.1 - - Tue, 26 Feb 2019 11:52:23 CST /consumer/v0/greeting 200 65 6 5c74b7f767922a84 org.apache.servicecomb.transport.rest.vertx.accesslog.impl.AccessLogHandler.lambda$handle$0(AccessLogHandler.java:42)
2019-02-26 11:52:24,250 [INFO] 127.0.0.1 - - Tue, 26 Feb 2019 11:52:24 CST /consumer/v0/greeting 200 65 6 5c74b7f88988d822 org.apache.servicecomb.transport.rest.vertx.accesslog.impl.AccessLogHandler.lambda$handle$0(AccessLogHandler.java:42)
2019-02-26 11:52:25,245 [INFO] 127.0.0.1 - - Tue, 26 Feb 2019 11:52:25 CST /consumer/v0/greeting 200 65 7 5c74b7f972e44db0 org.apache.servicecomb.transport.rest.vertx.accesslog.impl.AccessLogHandler.lambda$handle$0(AccessLogHandler.java:42)
2019-02-26 11:52:26,132 [INFO] 127.0.0.1 - - Tue, 26 Feb 2019 11:52:26 CST /consumer/v0/greeting 200 65 6 5c74b7fa27645f90 org.apache.servicecomb.transport.rest.vertx.accesslog.impl.AccessLogHandler.lambda$handle$0(AccessLogHandler.java:42)
```

实例隔离触发机制除了连续调用失败，还有错误率阈值。但在某些问题场景下，出错实例会积累很高的错误率。此时如果要让每隔60秒的尝试调用机制将错误率慢慢拉低到阈值以下，来解除实例的隔离状态，会花费相当长的时间。因此我们推荐大家使用默认的连续调用失败机制判断实例隔离，方便问题实例的快速隔离和正常实例的快速恢复。

小结

- CSEJavaSDK的负载均衡策略是客户端负载均衡，默认策略是轮询
- 重试策略可以在provider端实例意外退出、网络断连时保证consumer端业务调用不出错。默认的重试策略是tryOnNext=1,tryOnSame=0
- 实例隔离机制可以快速将问题实例从consumer端的实例缓存列表中排除，减小业务调用受问题实例影响的概率
- 今天的课程所讲的内容都在ServiceComb开源文档中有说明，推荐阅读：https://docs.servicecomb.io/java-chassis/zh_CN/references-handlers/loadbalance.html

Thank You