

1. Nacos简介

2018年6月, Eureka 2.0宣布闭源(但是1.X版本仍然为活跃项目), 同年7月份, 阿里Nacos宣布开源. 并快速成为国内最受关注开源产品. 作为Eureka的替代, Nacos已经成为了国内开发者的首选, 目前Nacos Star 已经突破28K(Eureka 12K)

谁在使用Nacos

请在 [谁在使用Nacos](#) 上提供信息来帮助Nacos做的更好。



Nacos (Dynamic Naming and Configuration Service)

在最初开源时, Nacos选择进行内部三个产品合并统一开源(Configserver 非持久注册中心, VIPServer 持久化注册中心, Diamond 配置中心). 定位为: 一个更易于构建云原生应用的动态服务发现, 配置管理和服务管理平台. 所以Nacos是一个注册中心组件, 但它又不仅仅是注册中心组件.

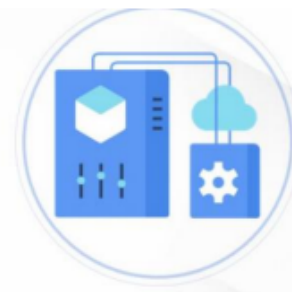
截至目前, Nacos几乎支持了所有的主流语言, 比如 Java, Go, C++, Nodejs, Python, Scala等



服务发现与管理



动态DNS服务



动态配置服务

官网: <https://nacos.io/>

仓库: <https://github.com/alibaba/nacos>





2. Nacos安装

学习阶段采用单机安装即可. 以下内容都是单机版

2.1 下载安装包


目前官方推荐的稳定版本为2.2.3, 咱们课程中也是用2.2.3

下载地址 <https://github.com/alibaba/nacos/releases/tag/2.2.3>

▼ Assets 4		
 nacos-server-2.2.3.tar.gz	142 MB	May 25
 nacos-server-2.2.3.zip	142 MB	May 25
 Source code (zip)		May 25
 Source code (tar.gz)		May 25

其他版本下载链接: 下载链接: <https://github.com/alibaba/nacos/releases>

Nacos安装包随课件提供, 也可以自己从网站下载.






名称	修改日期
 nacos-server-2.2.3.zip	2023/12/25 11:45

2.2 Windows

2.2.1 解压

把压缩包解压到任意非中文的目录下

此电脑 > Data (D:) > soft > nacos-server-2.2.3 >

名称	修改日期
 bin	2023/5/25 15:29
 conf	2023/5/25 15:08
 target	2023/5/25 15:29
 LICENSE	2023/3/6 17:48
 NOTICE	2020/5/14 10:03

目录介绍:

bin: Nacos启停脚本

- `startup.cmd` : windows平台的启动脚本
- `startup.sh` : Linux平台的启动脚本
- `shutdown.cmd` : windows平台的停止脚本
- `shutdown.sh` : Linux平台的停止脚本

conf: Nacos配置文件

target: 存放 Nacos 应用的 jar 包

2.2.2 修改单机模式

Nacos 默认启动方式为集群, 启动前需要修改配置为单机模式.

1. 使用记事本打开 `startup.cmd`
2. Line 26左右, 修改启动模式

```
1 set MODE="cluster"
```

改为

```
1 set MODE="standalone"
```

如下图所示:

```
1 set MODE="cluster"
2 set FUNCTION_MODE="all"
3 set SERVER=nacos-server
4 set MODE_INDEX=-1
5 set FUNCTION_MODE_INDEX=-1
6 set SERVER_INDEX=-1
7 set EMBEDDED_STORAGE_INDEX=-1
8 set EMBEDDED_STORAGE=""
```

```
1 set MODE="standalone"
2 set FUNCTION_MODE="all"
3 set SERVER=nacos-server
4 set MODE_INDEX=-1
5 set FUNCTION_MODE_INDEX=-1
6 set SERVER_INDEX=-1
7 set EMBEDDED_STORAGE_INDEX=-1
8 set EMBEDDED_STORAGE=""
```

2.2.3 启动Nacos

启动非常简单, 进入bin目录下, 双击 `startup.cmd` 即可

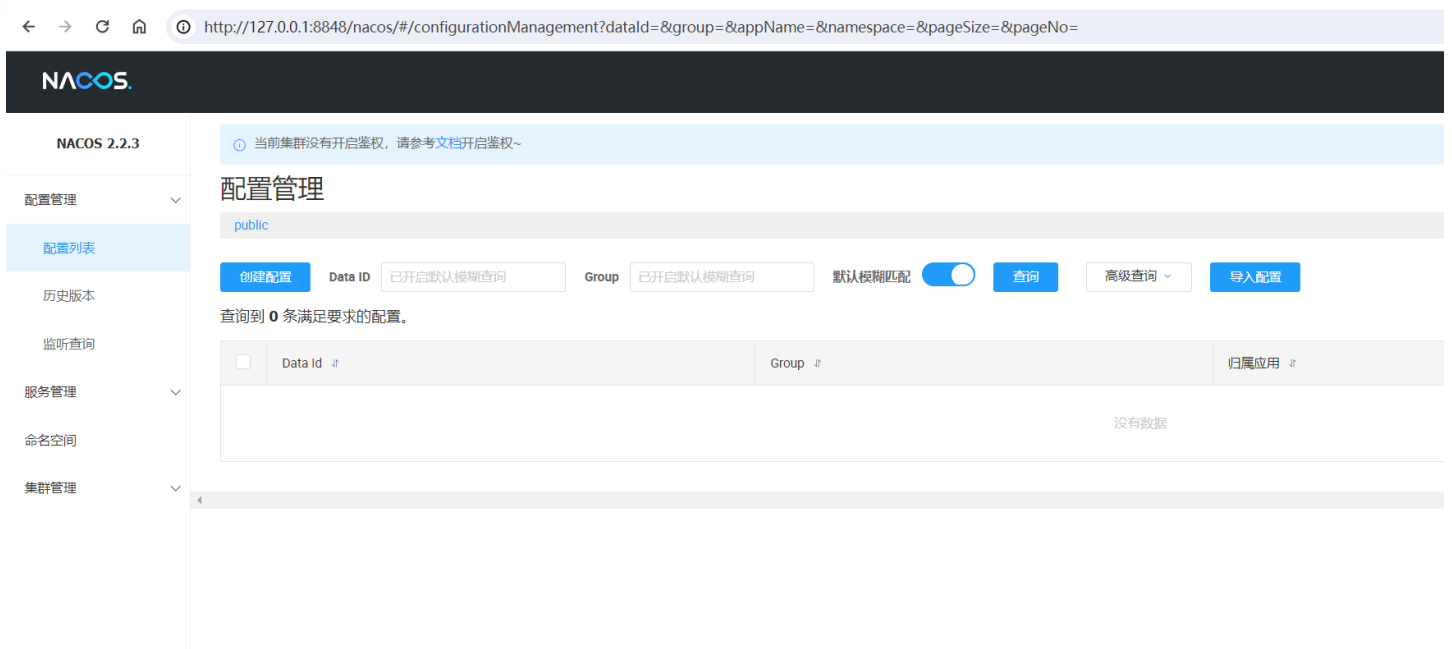
```
C:\Windows\system32\cmd.exe
nacos is starting with standalone

Nacos 2.2.3
Running in stand alone mode, All function modules
Port: 8848
Pid: 21096
Console: http://192.168.31.231:8848/nacos/index.html
https://nacos.io

2023-12-25 12:09:37,327 INFO Tomcat initialized with port(s): 8848 (http)
2023-12-25 12:09:37,462 INFO Root WebApplicationContext: initialization completed in 2570 ms
2023-12-25 12:09:40,651 INFO Adding welcome page: class path resource [static/index.html]
2023-12-25 12:09:41,053 WARN You are asking Spring Security to ignore Ant [pattern='/**']. This is not recommended -- please use permitAll via HttpSecurity#authorizeHttpRequests instead.
2023-12-25 12:09:41,054 INFO Will not secure Ant [pattern='/**']
2023-12-25 12:09:41,071 INFO Will secure any request with [org.springframework.security.web.context.request.async.WebAsyncManagerIntegrationFilter@16dalabc, org.springframework.security.web.context.SecurityContextPersistenceFilter@173f1614, org.springframework.security.web.header.HeaderWriterFilter@6ec63f8, org.springframework.security.web.csrf.CsrfFilter@76cf841, org.springframework.security.web.authentication.logout.LogoutFilter@2b4d4327, org.springframework.security.web.savedrequest.RequestCacheAwareFilter@7645f03e, org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@54b2fc58, org.springframework.security.web.authentication.AnonymousAuthenticationFilter@671ea6ff, org.springframework.security.web.session.SessionManagementFilter@40bb4f87, org.springframework.security.web.access.ExceptionTranslationFilter@297c9a9b]
2023-12-25 12:09:41,098 INFO Exposing 1 endpoint(s) beneath base path '/actuator'
2023-12-25 12:09:41,142 INFO Tomcat started on port(s): 8848 (http) with context path '/nacos'
2023-12-25 12:09:41,168 INFO Nacos started successfully in stand alone mode. use embedded storage
```

访问Nacos主页, 出现以下界面, 表示Nacos启动成功

<http://127.0.0.1:8848/nacos>



2.2.4 常见问题

Nacos启动后, 目录下会多一个logs的文件夹

报错日志在: logs/nacos.log

集群模式启动

报错日志

```
1 Caused by: java.net.UnknownHostException: jmenv.tbssite.net
2     at java.base/sun.nio.ch.NioSocketImpl.connect(NioSocketImpl.java:572)
3     at java.base/java.net.Socket.connect(Socket.java:633)
4     at java.base/sun.net.NetworkClient.doConnect(NetworkClient.java:178)
5     at
  java.base/sun.net.www.http.HttpClient.openServer(HttpClient.java:534)
6     at
  java.base/sun.net.www.http.HttpClient.openServer(HttpClient.java:639)
7     at java.base/sun.net.www.http.HttpClient.<init>(HttpClient.java:282)
8     at java.base/sun.net.www.http.HttpClient.New(HttpClient.java:387)
9     at java.base/sun.net.www.http.HttpClient.New(HttpClient.java:409)
10    at
  java.base/sun.net.www.protocol.http.HttpURLConnection.getNewHttpClient(HttpURLC
  onnection.java:1309)
11    at
  java.base/sun.net.www.protocol.http.HttpURLConnection.plainConnect0(HttpURLConn
  ection.java:1242)
12    at
  java.base/sun.net.www.protocol.http.HttpURLConnection.plainConnect(HttpURLConne
  ction.java:1128)
13    at
  java.base/sun.net.www.protocol.http.HttpURLConnection.connect(HttpURLConnection
  .java:1057)
14    at
  com.alibaba.nacos.common.http.client.request.JdkHttpClientRequest.execute(JdkHt
  tpClientRequest.java:114)
15    at
  com.alibaba.nacos.common.http.client.NacosRestTemplate.execute(NacosRestTemplat
  e.java:482)
16    at
  com.alibaba.nacos.common.http.client.NacosRestTemplate.get(NacosRestTemplate.ja
  va:72)
17    at
  com.alibaba.nacos.core.cluster.lookup.AddressServerMemberLookup.syncFromAddress
  Url(AddressServerMemberLookup.java:175)
18    at
  com.alibaba.nacos.core.cluster.lookup.AddressServerMemberLookup.run(AddressServ
  erMemberLookup.java:143)
19    ... 126 common frames omitted
20 2023-12-25 12:14:54,260 WARN [HttpClientBeanHolder] Start destroying common
  HttpClient
21
22 2023-12-25 12:14:54,260 WARN [ThreadPoolManager] Start destroying ThreadPool
```

Nacos默认是集群（cluster）启动，将其设置为单机（standalone），设置方式参考 上面章节

端口号冲突

Nacos 默认端口号是8848, 如果该端口号被其他应用占用, 启动会报错:

```
1 Caused by: java.net.BindException: Address already in use: bind
2       at java.base/sun.nio.ch.Net.bind0(Native Method)
3       at java.base/sun.nio.ch.Net.bind(Net.java:555)
4       at
  java.base/sun.nio.ch.ServerSocketChannelImpl.netBind(ServerSocketChannelImpl.java:337)
5       at
  java.base/sun.nio.ch.ServerSocketChannelImpl.bind(ServerSocketChannelImpl.java:294)
6       at
  io.grpc.netty.shaded.io.netty.channel.socket.nio.NioServerSocketChannel.doBind(NioServerSocketChannel.java:141)
7       at
  io.grpc.netty.shaded.io.netty.channel.AbstractChannel$AbstractUnsafe.bind(AbstractChannel.java:562)
8       at
  io.grpc.netty.shaded.io.netty.channel.DefaultChannelPipeline$HeadContext.bind(DefaultChannelPipeline.java:1334)
9       at
  io.grpc.netty.shaded.io.netty.channel.AbstractChannelHandlerContext.invokeBind(AbstractChannelHandlerContext.java:506)
10      at
  io.grpc.netty.shaded.io.netty.channel.AbstractChannelHandlerContext.bind(AbstractChannelHandlerContext.java:491)
11      at
  io.grpc.netty.shaded.io.netty.channel.DefaultChannelPipeline.bind(DefaultChannelPipeline.java:973)
12      at
  io.grpc.netty.shaded.io.netty.channel.AbstractChannel.bind(AbstractChannel.java:260)
13      at
  io.grpc.netty.shaded.io.netty.bootstrap.AbstractBootstrap$2.run(AbstractBootstrap.java:356)
14      at
  io.grpc.netty.shaded.io.netty.util.concurrent.AbstractEventExecutor.runTask(AbstractEventExecutor.java:174)
15      at
  io.grpc.netty.shaded.io.netty.util.concurrent.AbstractEventExecutor.safeExecute(AbstractEventExecutor.java:167)
16      at
  io.grpc.netty.shaded.io.netty.util.concurrent.SingleThreadEventExecutor.runAllTasks(SingleThreadEventExecutor.java:470)
```

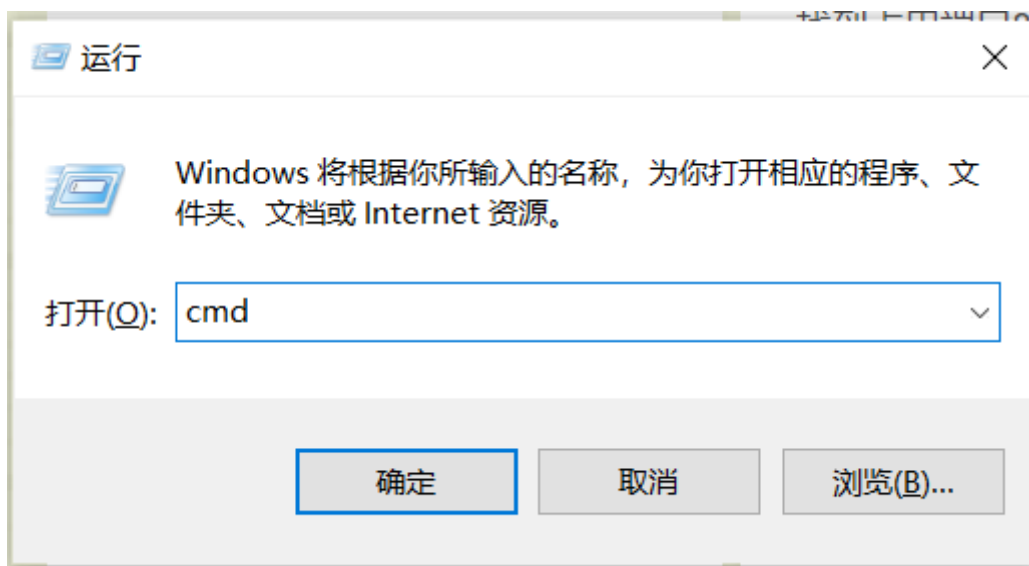
```
17         at
    io.grpc.netty.shaded.io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:503)
18         at
    io.grpc.netty.shaded.io.netty.util.concurrent.SingleThreadEventExecutor$4.run(SingleThreadEventExecutor.java:997)
19         at
    io.grpc.netty.shaded.io.netty.util.internal.ThreadExecutorMap$2.run(ThreadExecutorMap.java:74)
20         at
    io.grpc.netty.shaded.io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)
21         at java.base/java.lang.Thread.run(Thread.java:842)
22 2023-12-25 12:22:05,267 WARN [HttpClientBeanHolder] Start destroying common HttpClient
23
```

解决方式有以下两种, 任选其一:

1. 关闭该进程

a. 打开cmd

Win + R, 弹出命令提示符, 输入cmd



b. 查找进程

输入命令

```
1 netstat -ano|findstr "8848"
```

```
命令提示符
Microsoft Windows [版本 10.0.19045.3803]
(c) Microsoft Corporation。保留所有权利。

C:\Users\lucf>netstat -ano|findstr "8848"
TCP    0.0.0.0:8848          0.0.0.0:0          LISTENING        4968
TCP    [::]:8848           [::]:0             LISTENING        4968

C:\Users\lucf>
```

c. 杀掉进程

```
1 taskkill /pid 4968 -f
```

```
命令提示符
Microsoft Windows [版本 10.0.19045.3803]
(c) Microsoft Corporation。保留所有权利。

C:\Users\lucf>netstat -ano|findstr "8848"
TCP    0.0.0.0:8848          0.0.0.0:0          LISTENING        4968
TCP    [::]:8848           [::]:0             LISTENING        4968

C:\Users\lucf>taskkill /pid 4968 -f
成功: 已终止 PID 为 4968 的进程。

C:\Users\lucf>
```

2. 修改Nacos端口号

修改文件: \${Nacos目录}/conf/application.properties

23行左右

```
#***** Spring Boot Related Configurations *****#
### Default web context path:
server.servlet.contextPath=/nacos
### Include message field
server.error.include-message=ALWAYS
### Default web server port:
server.port=8848
```

修改8848为期望的端口号即可。

2.3 Linux

2.3.1 准备安装包

上传提前下载好的安装包到服务器上某个目录, 比如 `/usr/local/src`

解压安装包

```
1 unzip nacos-server-2.2.3.zip
```

如果第一次使用, 未安装unzip命令, 需要安装一下

```
1 apt-get install unzip
```

解压后目录如下:

```
1 root@hcscs-ecs-0bb1:/usr/local/src/nacos# pwd
2 /usr/local/src/nacos
3 root@hcscs-ecs-0bb1:/usr/local/src/nacos# ll
4 total 44
5 drwxr-xr-x 5 root root 4096 May 25 2023 ./
6 drwxr-xr-x 3 root root 4096 Dec 25 15:07 ../
7 drwxr-xr-x 2 root root 4096 May 25 2023 bin/
8 drwxr-xr-x 2 root root 4096 May 25 2023 conf/
9 -rw-r--r-- 1 root root 16583 Mar 6 2023 LICENSE
10 -rw-r--r-- 1 root root 1305 May 14 2020 NOTICE
11 drwxr-xr-x 2 root root 4096 May 25 2023 target/
12
```

和windows一样

2.3.2 单机模式启动

进入nacos/bin目录, 输入命令:

```
1 bash startup.sh -m standalone
```

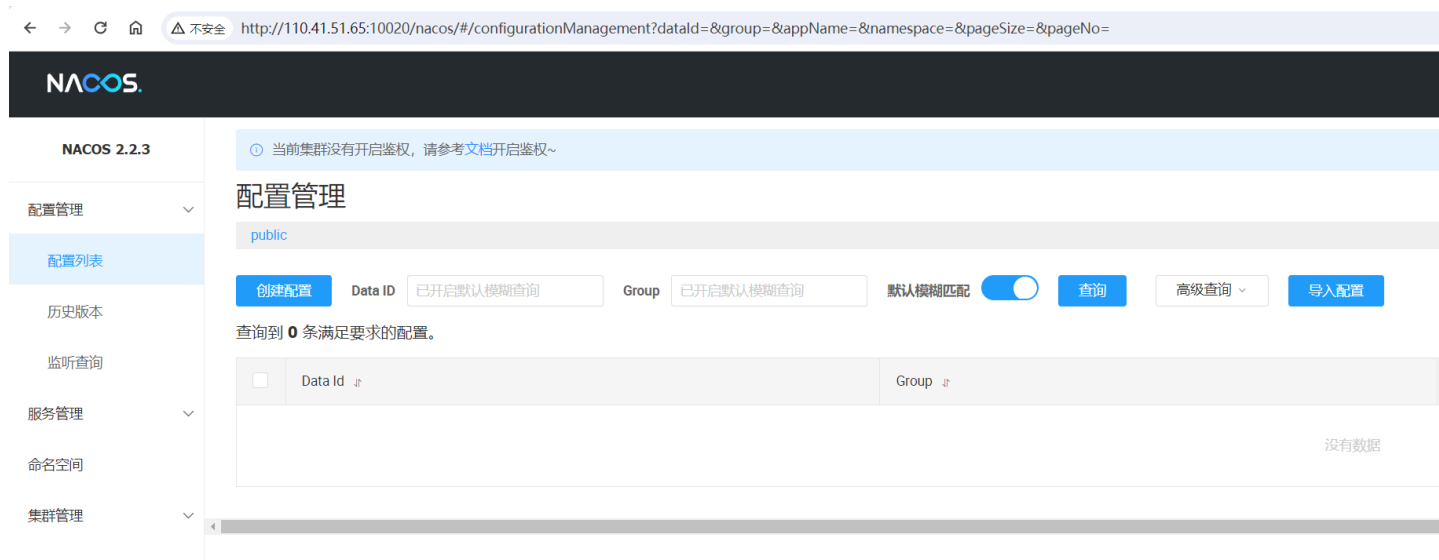
上述命令为Ubuntu系统的命令

nacos安装前需要先安装JDK

CenterOS使用: `sh startup.sh -m standalone`

```
1 root@hcss-ecs-0bb1:/usr/local/src/nacos/bin# bash startup.sh -m standalone
2 /usr/lib/jvm/java-17-openjdk-amd64/bin/java -Xms512m -Xmx512m -Xmn256m -
  Dnacos.standalone=true -Dnacos.member.list= -
  Xlog:gc*:file=/usr/local/src/nacos/logs/nacos_gc.log:time,tags:filecount=10,file
  esize=100m -
  Dloader.path=/usr/local/src/nacos/plugins,/usr/local/src/nacos/plugins/health,/
  usr/local/src/nacos/plugins/cmdb,/usr/local/src/nacos/plugins/selector -
  Dnacos.home=/usr/local/src/nacos -jar /usr/local/src/nacos/target/nacos-
  server.jar --spring.config.additional-
  location=file:/usr/local/src/nacos/conf/ --
  logging.config=/usr/local/src/nacos/conf/nacos-logback.xml --server.max-http-
  header-size=524288
3 nacos is starting with standalone
4 nacos is starting, you can check the /usr/local/src/nacos/logs/start.out
5
```

启动成功后, 访问Nacos链接: <http://IP:port/nacos>



10020为修改后的端口号, 需要在服务器上开放对应的端口号

另外, 再开放 `Nacos端口号 +1000` 和 `Nacos端口号+1001` 的端口

比如端口号为10020, 则需要开放端口号为: 10020, 11020, 11021

端口号为8848, 则需要开放端口号为: 9848, 9849

2.3.3 常见问题

参考Windows常见问题

3. Nacos快速上手

Nacos是Spring Cloud Alibaba的组件, Spring Cloud Alibaba遵循Spring Cloud中定义的服务注册, 服务发现规范. 因此使用Nacos和使用Eureka对于微服务来说, 并没有太大区别.

主要差异在于:

- Eureka需要自己搭建一个服务, Nacos不用自己搭建服务, 组件已经准备好了, 只需启动即可.
- 对应依赖和配置不同

操作参考: <https://github.com/alibaba/spring-cloud-alibaba/wiki/Nacos-discovery>


3.1 服务注册/服务发现

Nacos的服务注册和服务发现代码一样

引入Spring Cloud Alibaba依赖

在父工程的pom文件中的 `<dependencyManagement>` 中引入Spring Cloud Alibaba的依赖:

```
1 <properties>
2     <spring-cloud-alibaba.version>2022.0.0.0-RC2</spring-cloud-alibaba.version>
3 </properties>
4
5 <dependency>
6     <groupId>com.alibaba.cloud</groupId>
7     <artifactId>spring-cloud-alibaba-dependencies</artifactId>
8     <version>${spring-cloud-alibaba.version}</version>
9     <type>pom</type>
10    <scope>import</scope>
11 </dependency>
```

 注意: Spring Boot 和Spring Cloud的版本是有一定对应关系的. Spring Cloud Alibaba也遵循Spring Cloud 的标准, 在引入依赖时, 一定要确认各个版本的对应关系.

Spring Cloud Alibaba 和Spring Cloud版本对应关系, 参考官方文档:

<https://sca.aliyun.com/zh-cn/docs/2022.0.0.0/overview/version-explain/>

版本在一定范围内可以自由选择.

引入Nacos 依赖

在order-service和product-service中引入nacos依赖

```
1 <dependency>
```

```
2     <groupId>com.alibaba.cloud</groupId>
3     <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
4 </dependency>
```

引入Load Balance依赖

```
1 <dependency>
2     <groupId>org.springframework.cloud</groupId>
3     <artifactId>spring-cloud-loadbalancer</artifactId>
4 </dependency>
```

3.2 配置Nacos地址

课堂中直接使用服务器Nacos地址

配置项	Key	默认值	说明
服务端地址	spring.cloud.nacos.discovery. server-addr	无	Nacos Server 启动监听的ip地址和端口

```
1 spring:
2   application:
3     name: product-service
4   cloud:
5     nacos:
6       discovery:
7         server-addr: 110.41.51.65:10020
```

也可以改成本地Nacos的IP: 127.0.0.1:8848

3.3 远程调用

1. 修改IP为项目名

```
1 public OrderInfo selectOrderById(Integer orderId) {
2     OrderInfo orderInfo = orderMapper.selectOrderById(orderId);
3     String url = "http://product-service/product/" + orderInfo.getProductId();
4     ProductInfo productInfo = restTemplate.getForObject(url,
        ProductInfo.class);
```

```
5     orderInfo.setProductInfo(productInfo);
6     return orderInfo;
7 }
```

2. 为restTemplate添加负载均衡注解 @LoadBalanced

```
1 @Configuration
2 public class BeanConfig {
3     @Bean
4     @LoadBalanced
5     public RestTemplate restTemplate(){
6         return new RestTemplate();
7     }
8 }
```

3.4 启动服务

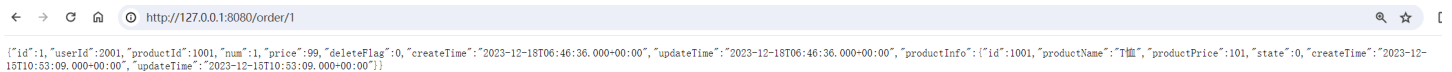
启动两个服务, 观察Nacos的管理界面, 发现order-service 和product-service 都注册在Nacos上了



The screenshot shows the Nacos 2.2.3 web interface. The left sidebar has a red box around '服务列表' (Service List). The main content area shows a table of services. The table has columns: 服务名 (Service Name), 分组名称 (Group Name), 集群数目 (Cluster Count), 实例数 (Instance Count), 健康实例数 (Healthy Instance Count), 触发保护阈值 (Trigger Protection Threshold), and 操作 (Operations). Two services are listed: 'product-service' and 'order-service', both in the 'DEFAULT_GROUP' with 1 instance each. The '操作' column contains links for '详情' (Details), '示例代码' (Sample Code), '订阅者' (Subscribers), and '删除' (Delete).

服务名	分组名称	集群数目	实例数	健康实例数	触发保护阈值	操作
product-service	DEFAULT_GROUP	1	1	1	false	详情 示例代码 订阅者 删除
order-service	DEFAULT_GROUP	1	1	1	false	详情 示例代码 订阅者 删除

测试接口: <http://127.0.0.1:8080/order/1>



3.5 启动多个服务, 测试负载均衡

启动三个product-service服务



观察Nacos控制台



多次访问接口, 观察日志

<http://127.0.0.1:8080/order/1>

3.6 常见问题

java.net.UnknownHostException

```
1 2023-12-25T19:04:23.803+08:00 ERROR 25892 --- [nio-8080-exec-1] o.a.c.c.C.[.[.
  [/.][dispatcherServlet] : Servlet.service() for servlet [dispatcherServlet]
  in context with path [] threw exception [Request processing failed:
  org.springframework.web.client.ResourceAccessException: I/O error on GET
  request for "http://product-service/product/1001": product-service] with root
  cause
2
3 java.net.UnknownHostException: product-service
4     at java.base/sun.nio.ch.NioSocketImpl.connect(NioSocketImpl.java:572) ~
  [na:na]
5     at java.base/java.net.Socket.connect(Socket.java:633) ~[na:na]
6     at java.base/java.net.Socket.connect(Socket.java:583) ~[na:na]
7     at java.base/sun.net.NetworkClient.doConnect(NetworkClient.java:183) ~
  [na:na]
8     at
  java.base/sun.net.www.http.HttpClient.openServer(HttpClient.java:534) ~[na:na]
```

```
9
10 // ...
11
```

检查是否添加 LoadBalance 依赖

```
1 <dependency>
2   <groupId>org.springframework.cloud</groupId>
3   <artifactId>spring-cloud-loadbalancer</artifactId>
4 </dependency>
```

服务注册失败

可能没有报错日志, 或者报错日志如下(与版本有关)

```
1 Parameter 0 of method inetIPv6Utils in
  com.alibaba.cloud.nacos.util.UtilIPv6AutoConfiguration required a bean of type
  'org.springframework.cloud.commons.util.InetUtilsProperties' that could not be
  found.
2
3 The injection point has the following annotations:
4   -
    @org.springframework.beans.factory.annotation.Autowired(required=true)
5
6
7 Action:
8
9 Consider defining a bean of type
  'org.springframework.cloud.commons.util.InetUtilsProperties' in your
  configuration.
```

检查Spring Cloud Alibaba版本是否正确

参考: [版本发布说明 | Spring Cloud Alibaba](#)

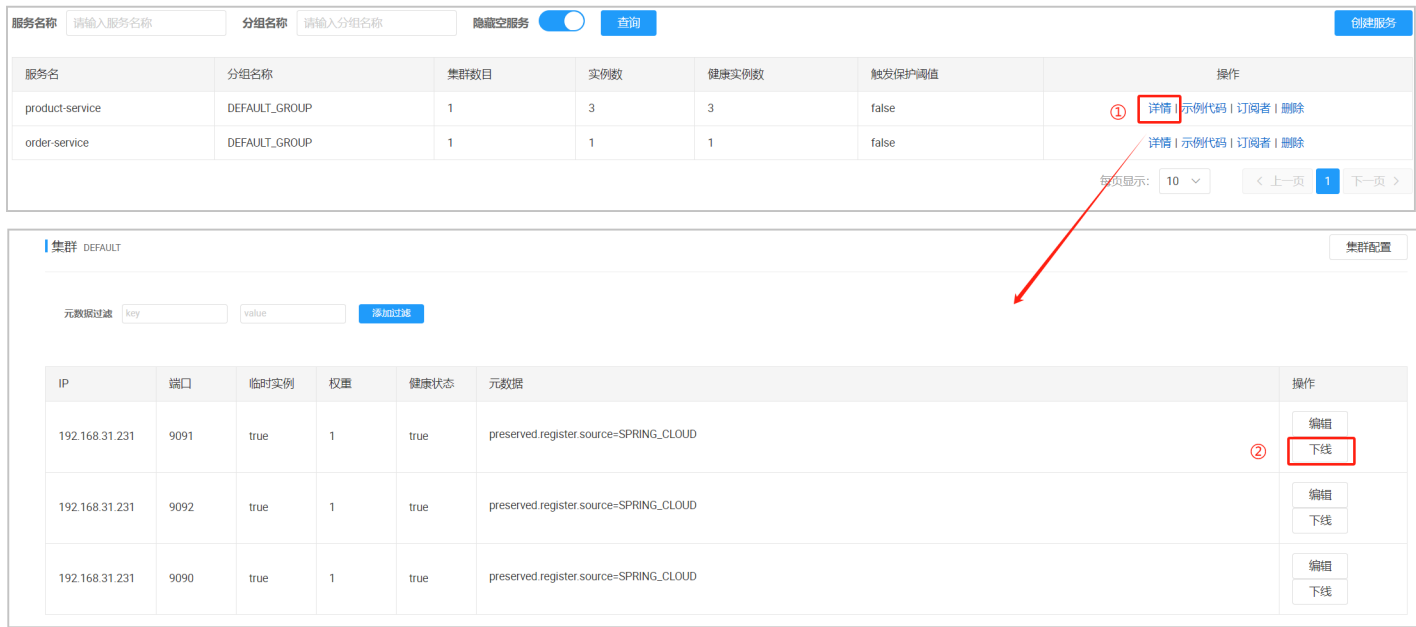
4. Nacos负载均衡

生产环境相对是比较恶劣的, 我们需要对服务的流量进行更加精细的控制. Nacos支持多种负载均衡策略, 包括权重, 同机房, 同地域, 同环境等.

4.1 服务下线

当某一个节点上接口的性能较差时, 我们可以第一时间对该节点进行下线.

操作步骤: 服务详情 -> 下线



点击下线后, 再次请求接口, 会发现该服务没有请求进来了

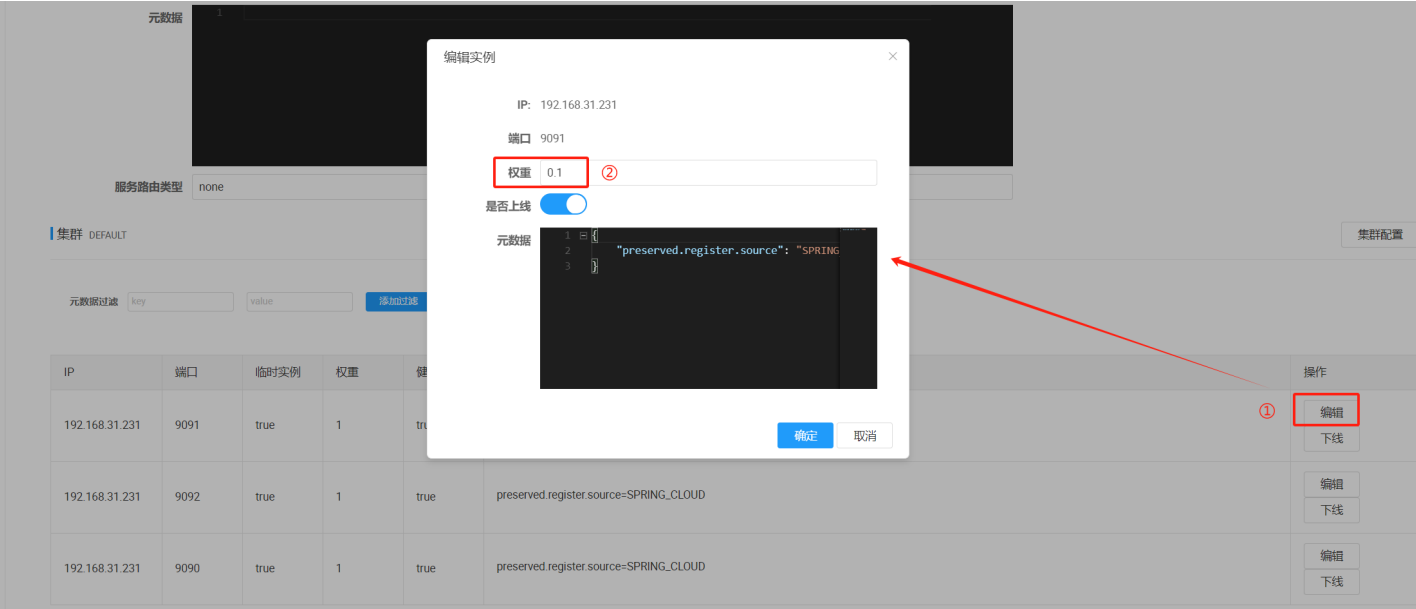
再次单击上线, 该节点会继续收到请求.

4.2 权重配置

除了下线之外, 我们也可以配置这个节点的流量权重

4.2.1 配置权重

操作步骤: 找到对应节点 -> 编辑 -> 在弹出的窗口修改权重值



每个节点默认权重为1, 修改为0.1.

4.2.2 开启Nacos负载均衡策略

由于Spring Cloud LoadBalance组件自身有负载均衡配置方式, 所以不支持Nacos的权重属性配置. 我们需要开启Nacos的负载均衡策略, 让权重配置生效

参考: [如何解决MSE Nacos上修改服务实例的权重不生效问题_微服务引擎\(MSE\)-阿里云帮助中心](#)

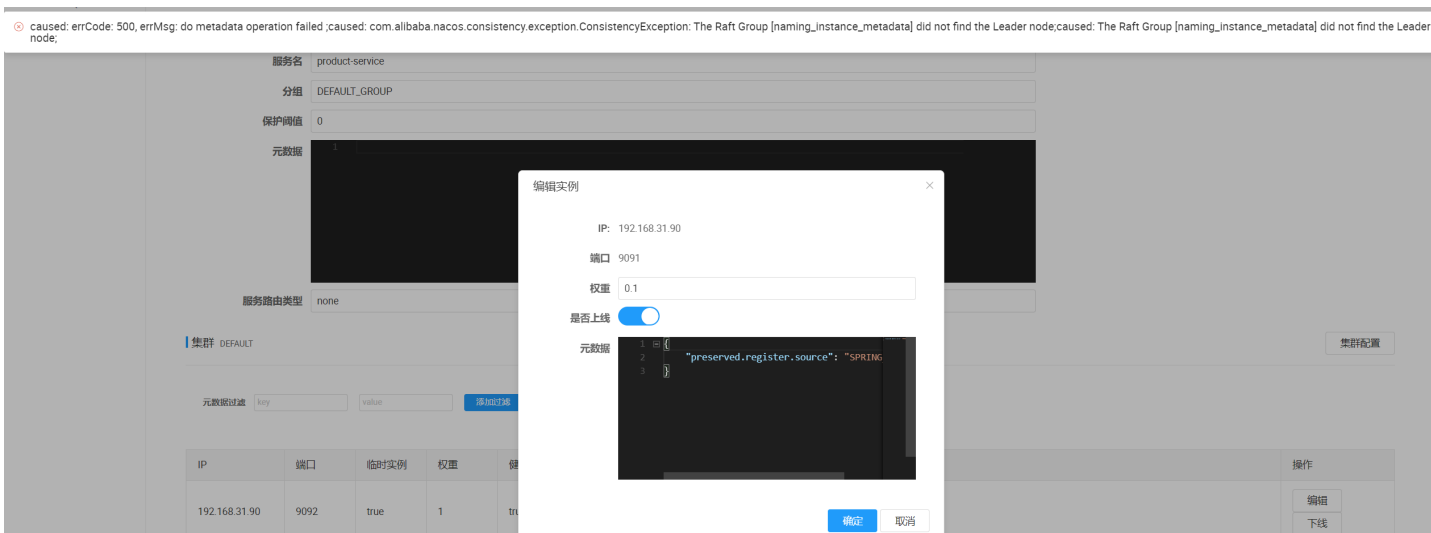
```
1 #开启nacos的负载均衡策略
2 spring.cloud.loadbalancer.nacos.enabled=true
3
4 spring:
5   cloud:
6     loadbalancer:
7       nacos:
8         enabled: true
```

4.2.3 测试权重配置

启动服务, 访问多次接口, 观察结果, 会发现9091端口号的实例接收的请求明显比另外两个实例少
整体流量生效, 局部流量不是严格按照设置的比例进行分配的

4.2.4 常见问题

修改权重时, 可能会报错:



报错信息: caused: errCode: 500, errMsg: do metadata operation failed ;caused: com.alibaba.nacos.consistency.exception.ConsistencyException: The Raft Group [naming_instance_metadata] did not find the Leader node;caused: The Raft Group [naming_instance_metadata] did not find the Leader node;

原因: Nacos 采用 raft 算法来计算 Leader, 并且会记录前一次启动的集群地址, 当服务器 IP 改变时会导致 raft 记录的集群地址失效, 导致选 Leader 出现问题. (网络环境发生变化时, IP地址也会发生变化)

解决办法: 删除 Nacos 根目录下 data 文件夹下的 protocol 文件夹即可.

4.3 同集群优先访问

Nacos把同一个机房内的实例, 划分为一个集群. 所以同集群优先访问, 在一定程度上也可以理解为同机房优先访问.

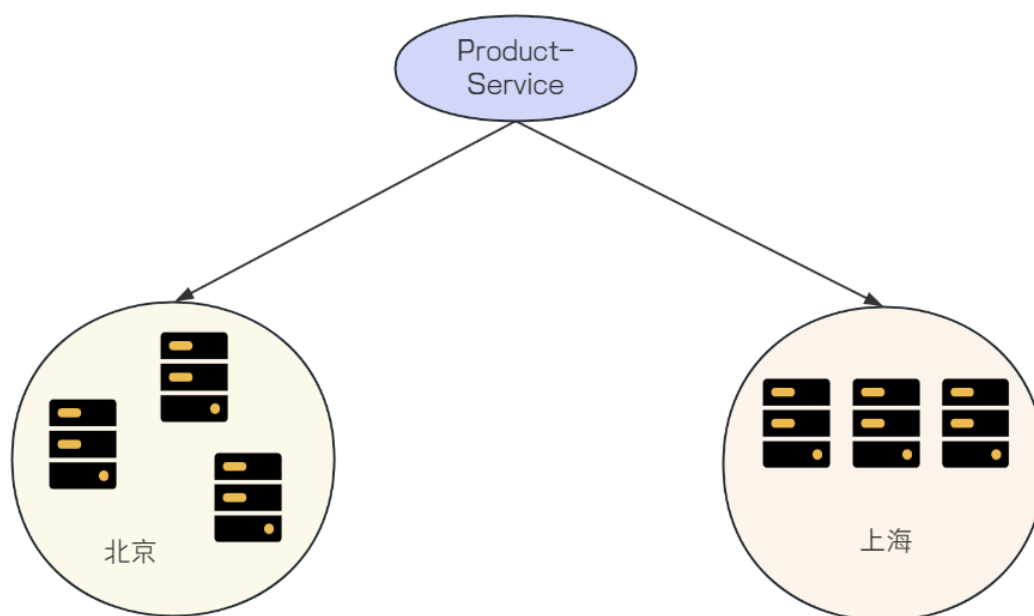
微服务架构中, 一个服务通常有多个实例共同提供服务, 这些实例可以部署在不同的机器上, 这些机器可以分布在不同的机房, 比如product-service:

实例1: 分布在上海机房

实例2: 分布在上海机房

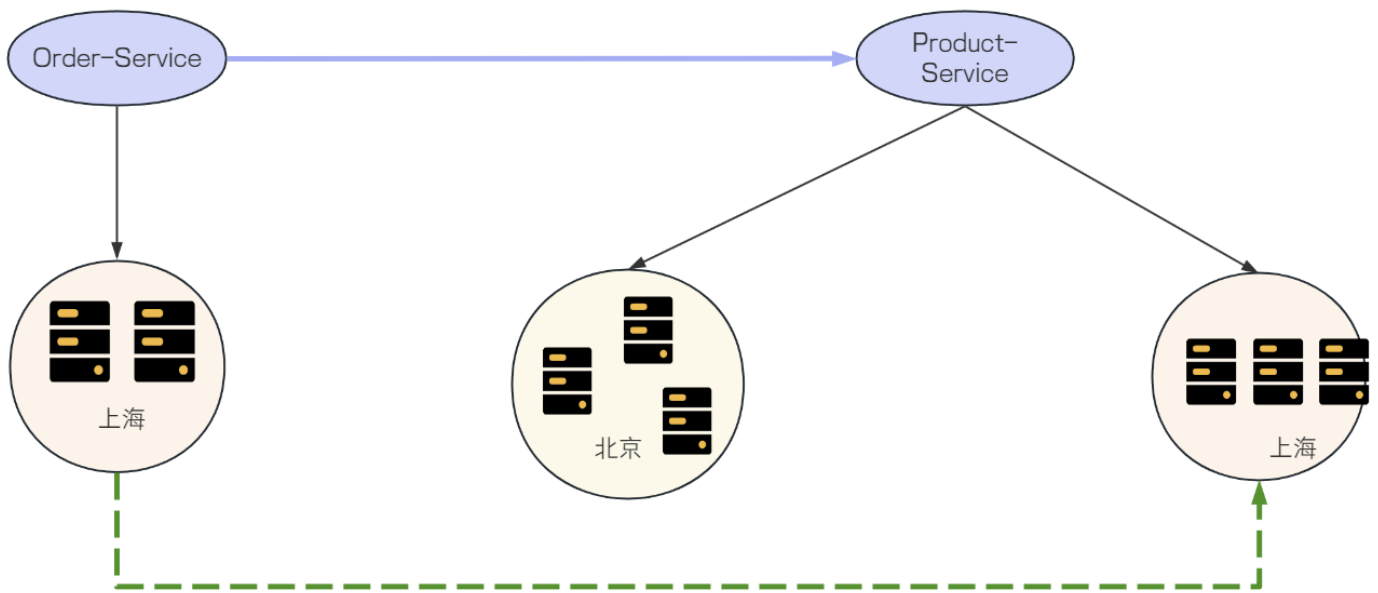
实例3: 分布在北京机房

实例4: 分布在北京机房



微服务访问时, 应尽量访问同机房的实例. 当本机房内实例不可用时, 才访问其他机房的实例.

比如order-service 在上海机房, product-service 在北京和上海机房都有实例, 那我们希望可以优先访问上海机房, 如果上海机房没有实例, 或者实例不可用, 再访问北京机房的实例. 通常情况下, 因为同一个机房的机器属于一个局域网, 局域网访问速度更快一点.



4.3.1 给实例配置集群名称

1. 为product-service配置集群名称

```

1 spring:
2   cloud:
3     nacos:
4       discovery:
5         server-addr: 110.41.51.65:10020
6         cluster-name: SH #集群名称: 上海集群

```

重启服务, 观察Nacos控制台, SH集群下多了一个实例

集群 SH						集群配置
元数据过滤 <input type="text" value="key"/> <input type="text" value="value"/> <button>添加过滤</button>						
IP	端口	临时实例	权重	健康状态	元数据	操作
192.168.31.90	9090	true	1	true	preserved.register.source=SPRING_CLOUD	<div>编辑</div> <div>下线</div>

集群 DEFAULT						集群配置
元数据过滤 <input type="text" value="key"/> <input type="text" value="value"/> <button>添加过滤</button>						
IP	端口	临时实例	权重	健康状态	元数据	操作
192.168.31.90	9092	true	1	true	preserved.register.source=SPRING_CLOUD	<div>编辑</div> <div>下线</div>
192.168.31.90	9091	true	10	true	preserved.register.source=SPRING_CLOUD	<div>编辑</div> <div>下线</div>

复制product-service启动配置, 添加VM Option

设置9091端口号的实例, 机房为BJ

```
1 -Dserver.port=9091 -Dspring.cloud.nacos.discovery.cluster-name=BJ
```

设置9092端口号的实例, 机房为BJ

```
1 -Dserver.port=9091 -Dspring.cloud.nacos.discovery.cluster-name=BJ
```

观察Nacos, BJ集群下多了一个实例

集群 SH

集群配置

元数据过滤 添加过滤

IP	端口	临时实例	权重	健康状态	元数据	操作
192.168.31.90	9090	true	1	true	preserved.register.source=SPRING_CLOUD	<div>编辑</div> <div>下线</div>

集群 BJ

集群配置

元数据过滤 添加过滤

IP	端口	临时实例	权重	健康状态	元数据	操作
192.168.31.90	9092	true	1	true	preserved.register.source=SPRING_CLOUD	<div>编辑</div> <div>下线</div>
192.168.31.90	9091	true	1	true	preserved.register.source=SPRING_CLOUD	<div>编辑</div> <div>下线</div>

2. 为order-service配置集群名称: SH

```
1 spring:
2   cloud:
3     nacos:
4       discovery:
5         server-addr: 110.41.51.65:10020
6         cluster-name: SH    #集群名称: 上海集群
```

4.3.2 开启Nacos负载均衡策略

同权重配置

```
1 #开启nacos的负载均衡策略
2 spring.cloud.loadbalancer.nacos.enabled=true
3
4 spring:
```

```
5   cloud:
6     loadbalancer:
7       nacos:
8         enabled: true
```

4.3.3 测试

启动服务

1. 对接口访问多次, 观察日志, 会发现只有9090端口的实例收到了请求(同集群)
2. 把9090端口的实例进行下线(SH集群), 再次访问接口, 观察日志, 发现9091端口和9092端口的实例收到了请求

5. Nacos 健康检查

5.1 两种健康检查机制

Nacos作为注册中心, 需要感知服务的健康状态, 才能为服务调用方提供良好的服务.

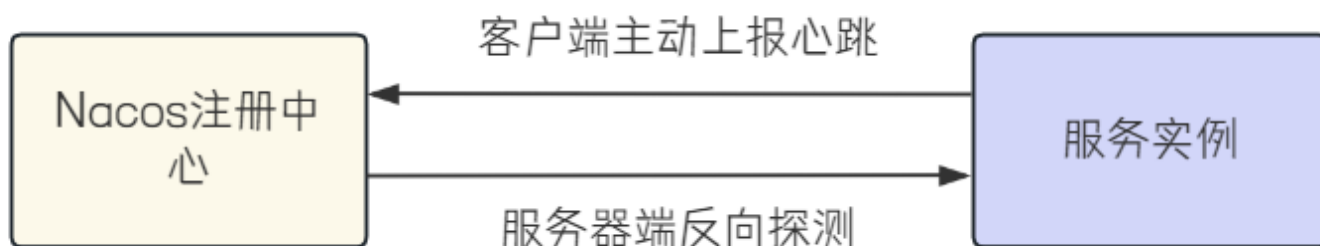
Nacos 中提供了两种健康检查机制:

客户端主动上报机制:

- 客户端通过心跳上报方式告知服务端(nacos注册中心)健康状态, 默认心跳间隔5秒;
- nacos会在超过15秒未收到心跳后将实例设置为不健康状态, 超过30秒将实例删除

服务器端反向探测机制:

- nacos主动探知客户端健康状态, 默认间隔为20秒.
- 健康检查失败后实例会被标记为不健康, 不会被立即删除.



比如领导管理员工的工作

1. 员工主动汇报: 员工每天主动汇报自己工作进度
2. 领导主动问询: 领导每周向员工了解工作进度

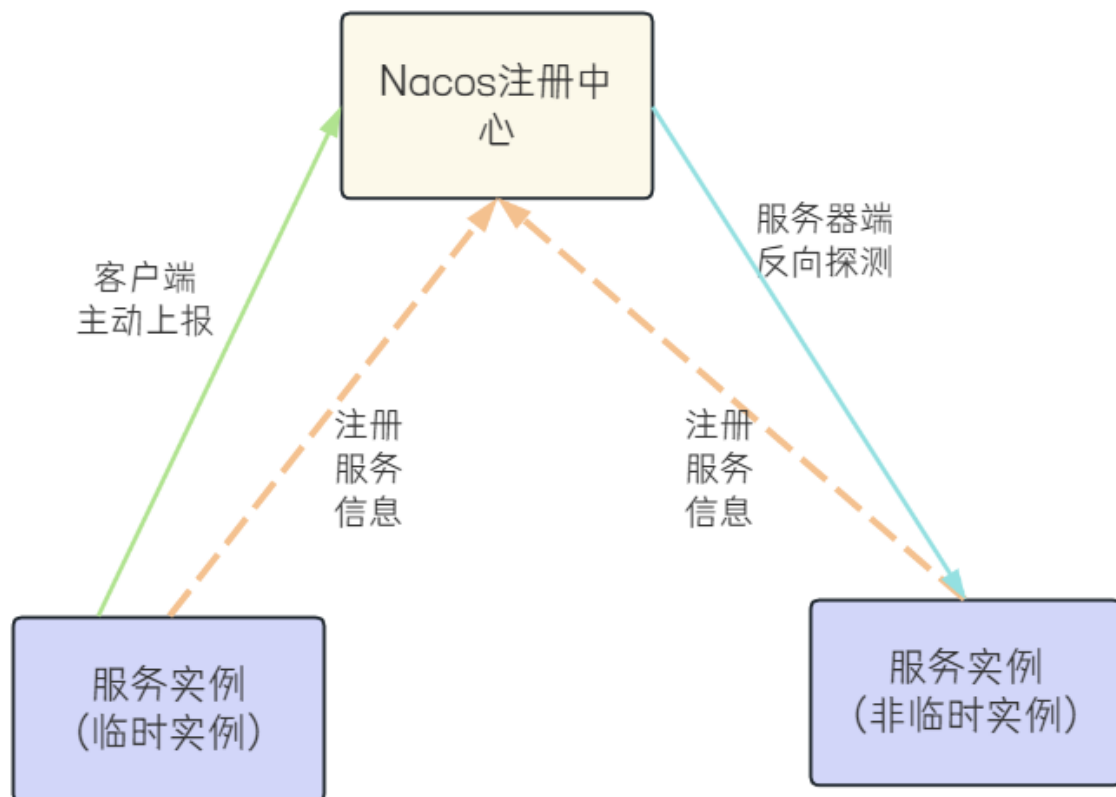
Nacos 中的健康检查机制不能主动设置, 健康检查机制是和 Nacos 的**服务实例类型**强相关的.

5.2 Nacos服务实例类型

Nacos的服务实例(注册的节点)分为临时实例和非临时实例。

- 临时实例: 如果实例宕机超过一定时间, 会从服务列表剔除, 默认类型。
- 非临时实例: 如果实例宕机, 不会从服务列表剔除, 也可以叫永久实例

Nacos对临时实例, 采取的是 客户端主动上报机制, 对非临时实例, 采取服务器端反向探测机制。



配置一个服务实例为永久实例

```
1 spring:
2   cloud:
3     nacos:
4       discovery:
5         ephemeral: false # 设置为非临时实例
```

重启服务, 观察Nacos控制台

IP	端口	临时实例	权重	健康状态	元数据
192.168.31.107	9091	false	1	true	preserved.register.source=SPRING_CLOUD
192.168.31.107	9092	false	1	true	preserved.register.source=SPRING_CLOUD

停止服务, 再观察控制台

IP	端口	临时实例	权重	健康状态	元数据
192.168.31.107	9091	false	1	false	preserved.register.source=SPRING_CLOUD
192.168.31.107	9092	false	1	false	preserved.register.source=SPRING_CLOUD

节点依然不会消失.

5.3 常见问题

5.3.1 Nacos服务实例类型不允许改变

设置服务实例类型, 重新启动Nacos可能会报错

报错信息参考:

```
1 Caused by: com.alibaba.nacos.api.exception.NacosException: failed to req
  API:/nacos/v1/ns/instance after all servers([110.41.51.65:10020]) tried:
  caused: errCode: 400, errMsg: Current service DEFAULT_GROUP@@product-service is
  ephemeral service, can't register persistent instance. ;
2       at
  com.alibaba.nacos.client.naming.remote.http.NamingHttpClientProxy.reqApi(Naming
  HttpClientProxy.java:410) ~[nacos-client-2.2.1.jar:na]
3       at
  com.alibaba.nacos.client.naming.remote.http.NamingHttpClientProxy.reqApi(Naming
  HttpClientProxy.java:351) ~[nacos-client-2.2.1.jar:na]
4       at
  com.alibaba.nacos.client.naming.remote.http.NamingHttpClientProxy.reqApi(Naming
  HttpClientProxy.java:346) ~[nacos-client-2.2.1.jar:na]
5       // ...
6
7
```

原因: Nacos会记录每个服务实例的IP和端口号, 当发现IP和端口都没有发生变化时, Nacos不允许一个服务实例类型发生变化, 比如从临时实例,变为非临时实例, 或者从非临时实例, 变成临时实例.

解决办法:

- 1. 停掉nacos
- 2. 删除nacos 目录下 /data/protocol/raft 信息, 里面会保存应用实例的元数据信息.

5.3.2 服务正常, Nacos健康检查失败

现象:

服务正常, 但是Nacos显示健康状态为false

http://127.0.0.1:9090/product/1001

{“id”:1001,“productName”:“T恤”,“productPrice”:101,“state”:0,“createTime”:“2023-12-15T10:53:09.000+00:00”,“updateTime”:“2023-12-15T10:53:09.000+00:00”}

IP	端口	临时实例	权重	健康状态	元数据
192.168.31.107	9090	false	1	false	preserved.register.source=SPRING_CLOUD

原因和解决办法:

参考:

[如何解决Nacos持久化实例HTTP/TCP的健康检查不通过问题_微服务引擎\(MSE\)-阿里云帮助中心](#)

6. Nacos环境隔离

企业开发中, 一个服务会分为开发环境, 测试环境和生产环境.

- 1. 开发环境: 开发人员用于开发的服务器, 是最基础的环境. 一般日志级别设置较低, 可能会开启一些调试信息.
- 2. 测试环境: 测试人员用来进行测试的服务器, 是开发环境到生产环境的过渡环境.
- 3. 生产环境: 正式提供对外服务的环境, 通常关掉调试信息.

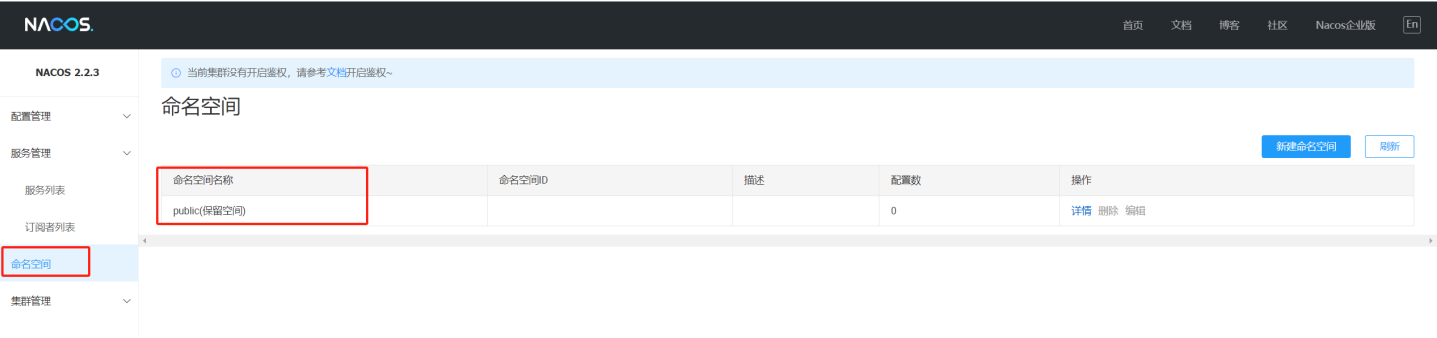
通常情况下, 这几个环境是不能互相通信的. Nacos提供了namespace(命名空间)来实现环境的隔离. 不同的namespace的服务不可见.

6.1 创建Namespace

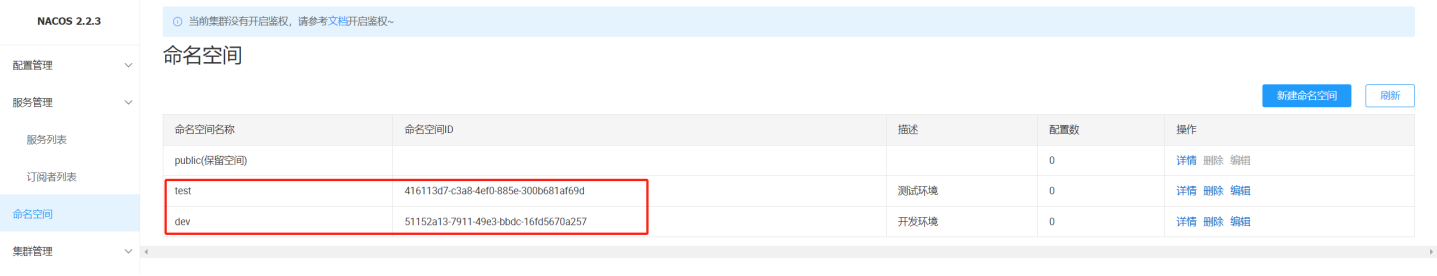
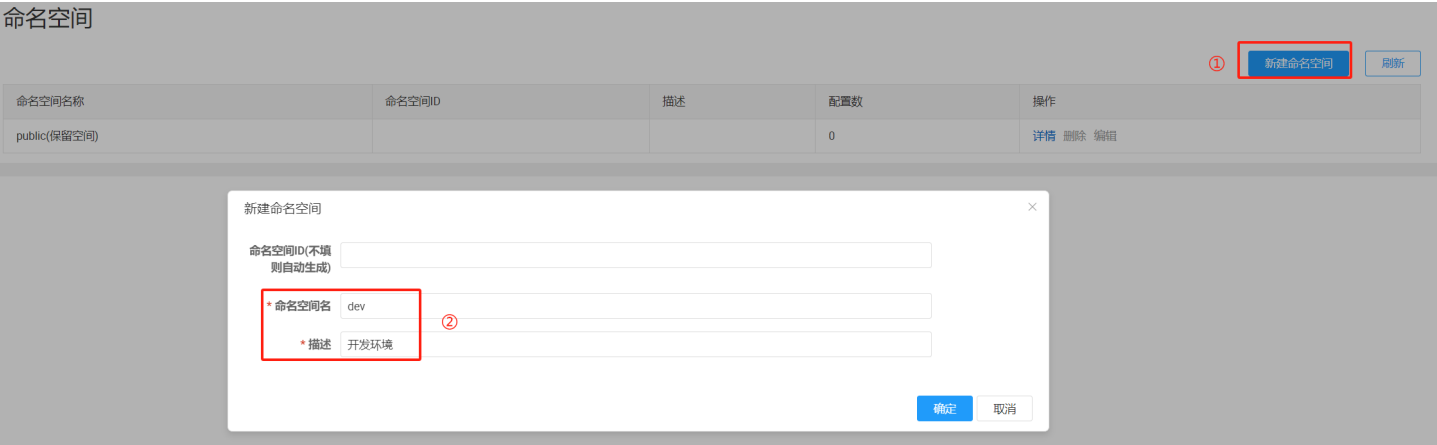
默认情况下, 所有服务都在同一个namespace, 名为public



点击左侧命名空间, 可以对namespace进行操作



新增命名空间



6.2 配置namespace

namespace创建完成后, 对服务进行配置

配置项	Key	默认值	说明
命名空间	spring.cloud.nacos.discovery.namespace	无	常用场景之一是不同环境的注册的区分离，例如开发测试环境和生产环境的资源（如配

置、服务) 隔离等。

修改order-service的命名空间

```
1 spring:
2   cloud:
3     nacos:
4       discovery:
5         namespace: 51152a13-7911-49e3-bbdc-16fd5670a257
```

6.3 测试远程调用

1. 启动服务, 观察Nacos控制台

public 命名空间下只有product-service服务



order-service在dev命名空间下



2. 访问接口, 测试远程调用

发现服务报错:

```
1 2023-12-27T18:23:27.058+08:00 ERROR 30408 --- [nio-8080-exec-1] o.a.c.c.C.[.[.
  [/.][dispatcherServlet] : Servlet.service() for servlet [dispatcherServlet]
  in context with path [] threw exception [Request processing failed:
  java.lang.IllegalStateException: No instances available for product-service]
  with root cause
2
3 java.lang.IllegalStateException: No instances available for product-service
```

```

4      at
  org.springframework.cloud.loadbalancer.blocking.client.BlockingLoadBalancerClient.execute(BlockingLoadBalancerClient.java:78) ~[spring-cloud-loadbalancer-4.0.3.jar:4.0.3]
5      at
  org.springframework.cloud.client.loadbalancer.LoadBalancerInterceptor.intercept(LoadBalancerInterceptor.java:56) ~[spring-cloud-commons-4.0.3.jar:4.0.3]
6      at
  org.springframework.http.client.InterceptingClientHttpRequest$InterceptingRequestExecution.execute(InterceptingClientHttpRequest.java:87) ~[spring-web-6.0.14.jar:6.0.14]
7      at
  org.springframework.http.client.InterceptingClientHttpRequest.executeInternal(InterceptingClientHttpRequest.java:71) ~[spring-web-6.0.14.jar:6.0.14]
8      at
  org.springframework.http.client.AbstractBufferingClientHttpRequest.executeInternal(AbstractBufferingClientHttpRequest.java:48) ~[spring-web-6.0.14.jar:6.0.14]
9      at
  org.springframework.http.client.AbstractClientHttpRequest.execute(AbstractClientHttpRequest.java:66) ~[spring-web-6.0.14.jar:6.0.14]
10     at
  org.springframework.web.client.RestTemplate.doExecute(RestTemplate.java:862) ~[spring-web-6.0.14.jar:6.0.14]
11 // ...

```

3. 修改product-service的其中一个实例, 命名空间改为dev

```

1 spring:
2   cloud:
3     nacos:
4       discovery:
5         namespace: 51152a13-7911-49e3-bbdc-16fd5670a257

```

4. 启动服务

观察Nacos控制台

NACOS 2.2.3

当前集群没有开启鉴权, 请参考[文档](#)开启鉴权~

服务列表 命名空间ID: 51152a13-7911-49e3-bbdc-16fd5670a257

public | test | **dev**

服务名称: 请输入服务名称 分组名称: 请输入分组名称 隐藏空服务: ☒ 查询 创建服务

服务名	分组名称	集群数目	实例数	健康实例数	触发保护阈值	操作
order-service	DEFAULT_GROUP	1	1	1	false	详情 示例代码 订阅者 删除
product-service	DEFAULT_GROUP	1	1	1	false	详情 示例代码 订阅者 删除

每页显示: 10 < 上一页 1 下一页 >

再次访问接口 <http://127.0.0.1:8080/order/3>, 发现远程调用成功

```
<  →  ↻  ↺  http://127.0.0.1:8080/order/3  🔍  ☆  □

[{"id":3,"userId":2001,"productId":1003,"num":1,"price":40,"deleteFlag":0,"createTime":"2023-12-18T06:46:36.000+00:00","updateTime":"2023-12-18T06:46:36.000+00:00","productInfo":{"id":1003,"productName":"短裤","productPrice":44,"state":0,"createTime":"2023-12-15T10:53:09.000+00:00","updateTime":"2023-12-15T10:53:09.000+00:00"}}
```

7. Nacos配置中心

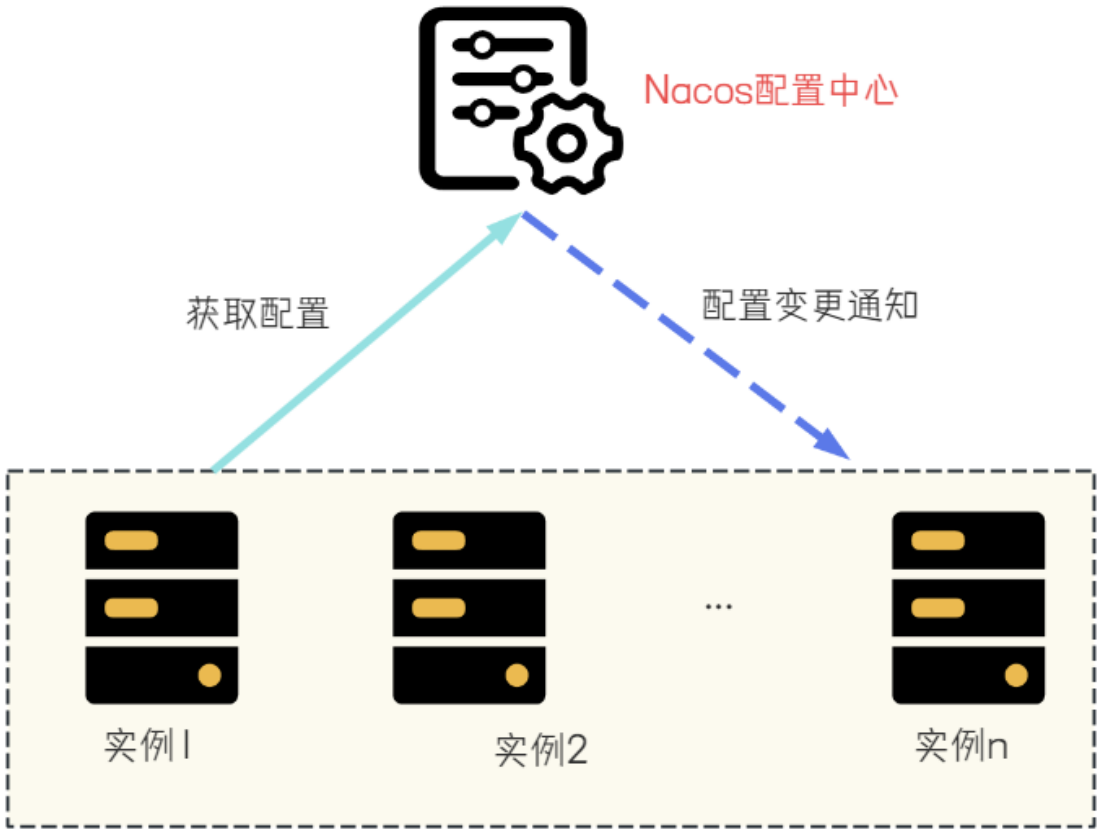
除了注册中心和负载均衡之外, Nacos还是一个配置中心, 具备配置管理的功能。
Namespace 的常用场景之一是不同环境的配置区分隔离. 例如开发测试环境和生产环境的配置隔离.

7.1 为什么需要配置中心

当前项目的配置都在代码中, 会存在以下问题:

- 1. 配置文件修改时, 服务需要重新部署. 微服务架构中, 一个服务可能有成百个实例, 挨个部署比较麻烦, 且容易出错.
- 2. 多人开发时, 配置文件可能需要经常修改, 使用同一个配置文件容易冲突.

配置中心就是对这些配置项进行统一管理. 通过配置中心, 可以集中查看, 修改和删除配置, 无需再逐个修改配置文件. 提高效率的同时, 也降低了出错的风险.



- 1. 服务启动时, 从配置中心读取配置项的内容, 进行初始化.
- 2. 配置项修改时, 通知微服务, 实现配置的更新加载.

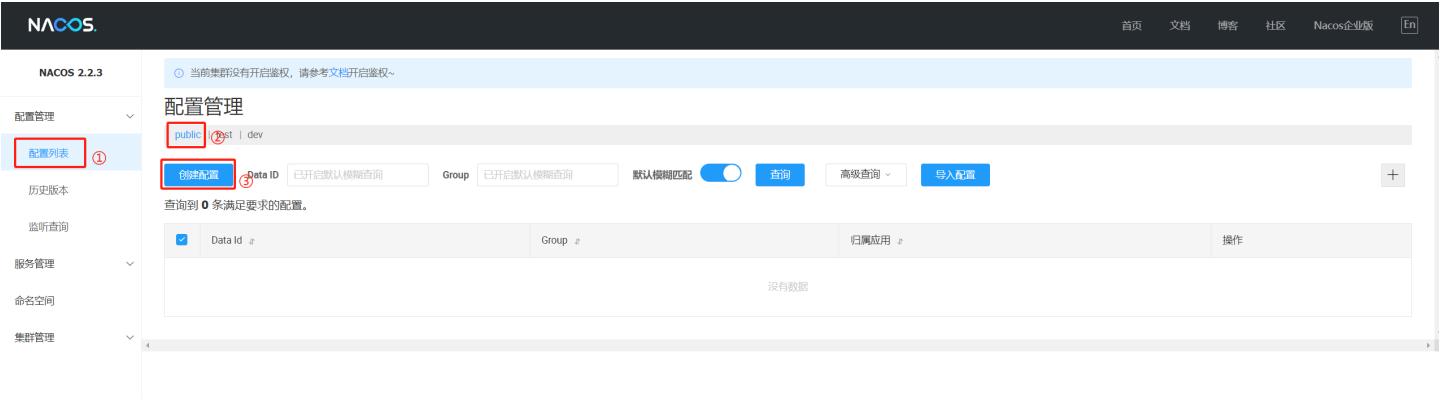
7.2 快速上手


通过以下操作, 我们先来感受下Nacos 配置中心的使用

参考文档: [Nacos Spring Cloud 快速开始](#) [Nacos config](#)

7.2.1 添加配置

在Nacos控制台添加配置项



 注意: 配置管理的命名空间和服务列表的命名空间是隔离的, 两个是分别设置的. 默认是public 也就是**服务管理命名空间配置 ≠ 配置管理的命名空间**

新建配置项

新建配置

* 命名空间

* Data ID ①

* Group

[更多高级选项](#)

描述

配置格式 ☐ TEXT ☐ JSON ☐ XML ☐ YAML ☐ HTML ☒ Properties ②

* 配置内容 ③

```
1 nacos.test.num=5
```

[发布](#) [返回](#)

配置内容:

```
1 nacos.test.num=5
```

说明:

1. Data ID 设置为项目名称
2. 配置内容的数据格式, 目前只支持 `properties` 和 `yaml` 类型
3. 设置配置内容

7.2.2 获取配置

1. 引入Nacos Config依赖

```
1 <dependency>
2     <groupId>com.alibaba.cloud</groupId>
3     <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
4 </dependency>
5 <!-- SpringCloud 2020.*之后版本需要引入bootstrap-->
6 <dependency>
7     <groupId>org.springframework.cloud</groupId>
8     <artifactId>spring-cloud-starter-bootstrap</artifactId>
9 </dependency>
```

2. 配置bootstrap.properties

微服务启动前, 需要先获取nacos中配置, 并与application.yml配置合并. 在微服务运行之前, Nacos要求必须使用 bootstrap.properties 配置文件来配置Nacos Server 地址.

```
1 spring.application.name=product-service
2 spring.cloud.nacos.config.server-addr=110.41.51.65:10020
```

或者使用bootstrap.yml

```
1 spring:
2   application:
3     name: product-service
4   cloud:
5     nacos:
6       config:
7         server-addr: 110.41.51.65:10020
```

`spring.application.name` 需要和nacos配置管理的Data ID一致

`spring.cloud.nacos.config.server-addr` 为Nacos Server的地址



配置中心和注册中心的配置是隔离的

Nacos 配置中心: `spring.cloud.nacos.config.server-addr`

Nacos 注册中心: `spring.cloud.nacos.discovery.server-addr`

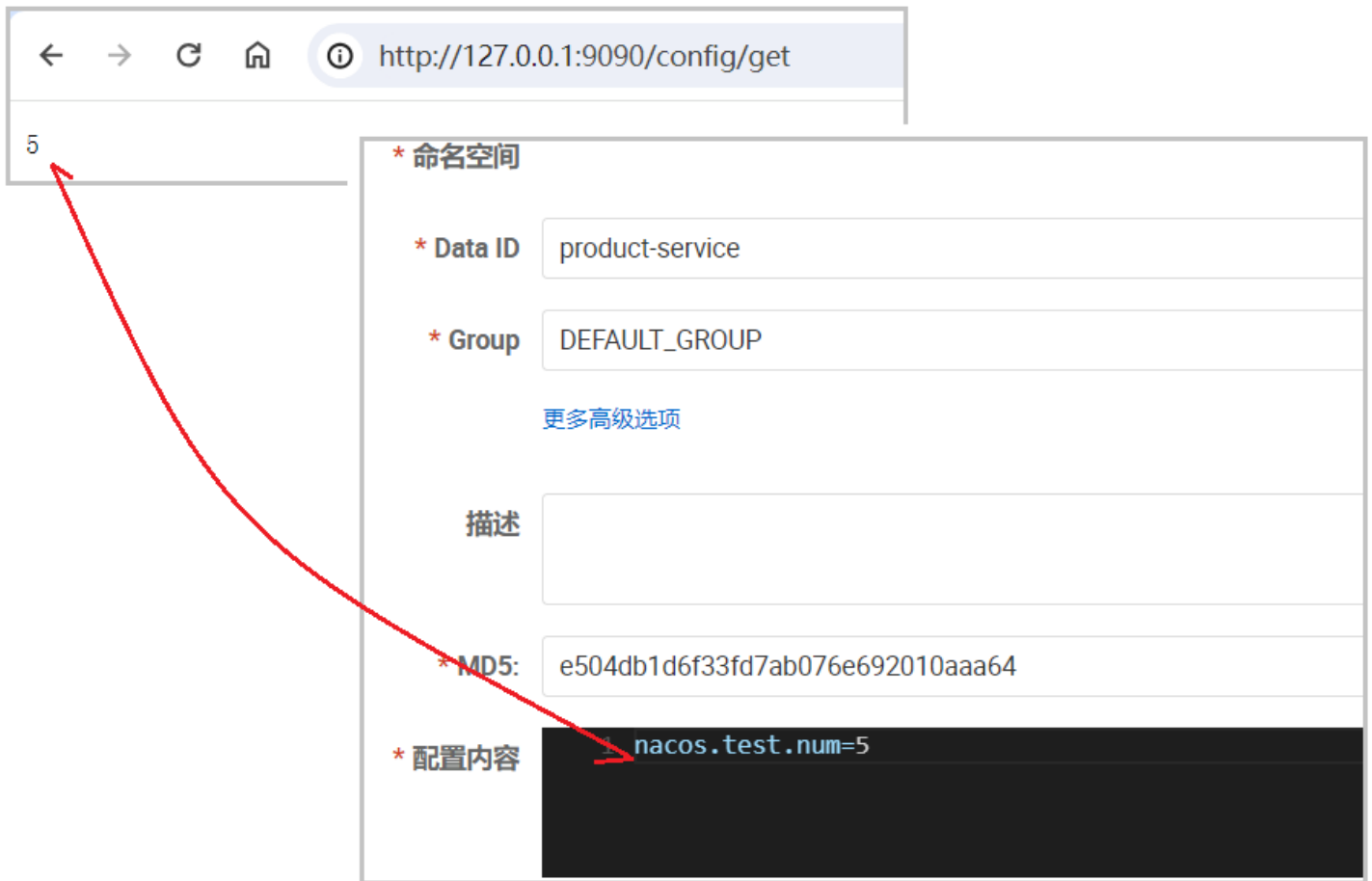
3. 编写程序

```
1 import org.springframework.beans.factory.annotation.Value;
2 import org.springframework.cloud.context.config.annotation.RefreshScope;
3 import org.springframework.web.bind.annotation.RequestMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RequestMapping("/config")
7 @RefreshScope
8 @RestController
9 public class NacosController {
10     @Value("${nacos.test.num:0}")
11     private Integer nacosNum;
12
13     @RequestMapping("/get")
14     public Integer get() {
15         return nacosNum;
16     }
17 }
```

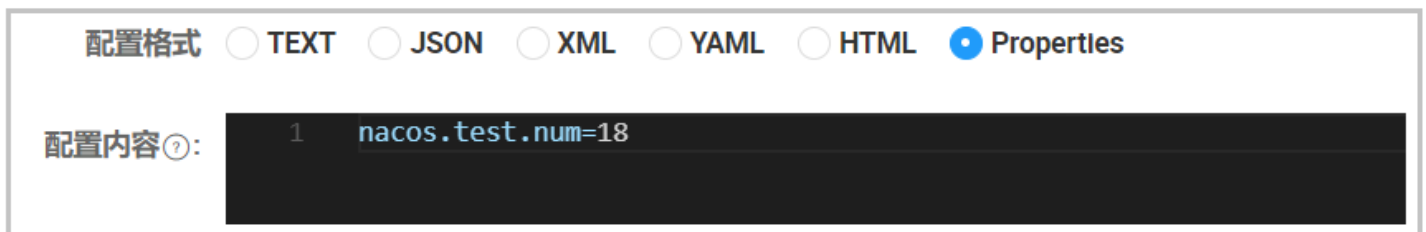
- @Value 读取配置
- @RefreshScope 配置进行热更新

4. 测试

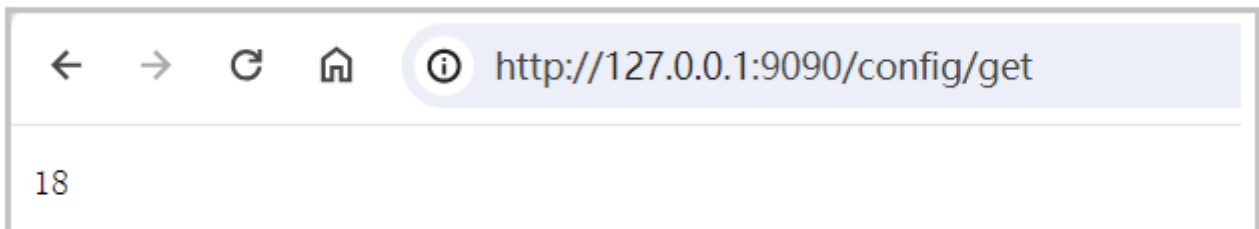
启动程序, 访问接口: <http://127.0.0.1:9090/config/get>



在Nacos控制台修改 `nacos.test.num`



再次访问接口:



7.2.3 常见问题

1. 读取不到配置项

可能原因:

- 配置错误: 检查配置Data ID, 配置格式, 配置空间等
- 未引入依赖

2. No spring.config.import property has been defined

启动报错日志:

```
1 *****
2 APPLICATION FAILED TO START
3 *****
4
5 Description:
6
7 No spring.config.import property has been defined
8
9 Action:
10
11 Add a spring.config.import=nacos: property to your configuration.
12     If configuration is not required add
13     spring.config.import=optional:nacos: instead.
14     To disable this check, set spring.cloud.nacos.config.import-
15     check.enabled=false.
```

原因: `bootstrap.properties` 是系统级的资源配置文件, 用于程序执行更加早期配置信息读取. 但是SpringCloud 2020.* 之后的版本把bootstrap禁用了, 导致在读取文件的时候读取不到而报错, 所以需要重新导入bootstrap 包进来就可以了

3. Nacos Server地址配置错误

报错信息如下:

```
1 2023-12-28T17:12:53.070+08:00 ERROR 14356 --- [t.remote.worker]
   c.a.n.c.remote.client.grpc.GrpcClient : Server check fail, please check
   server 127.0.0.1 ,port 11020 is available , error ={}
2
3 java.util.concurrent.ExecutionException:
   com.alibaba.nacos.shaded.io.grpc.StatusRuntimeException: UNAVAILABLE: io
   exception
4     at
   com.alibaba.nacos.shaded.com.google.common.util.concurrent.AbstractFuture.ge
   tDoneValue(AbstractFuture.java:566) ~[nacos-client-2.2.1.jar:na]
5     at
   com.alibaba.nacos.shaded.com.google.common.util.concurrent.AbstractFuture.ge
   t(AbstractFuture.java:445) ~[nacos-client-2.2.1.jar:na]
```

```

6         // ...
7
8 Caused by: com.alibaba.nacos.shaded.io.grpc.StatusRuntimeException:
    UNAVAILABLE: io exception
9         at
    com.alibaba.nacos.shaded.io.grpc.Status.asRuntimeException(Status.java:539)
    ~[nacos-client-2.2.1.jar:na]
10        at
    com.alibaba.nacos.shaded.io.grpc.stub.ClientCalls$UnaryStreamToFuture.onClose
    (ClientCalls.java:544) ~[nacos-client-2.2.1.jar:na]
11        at
    com.alibaba.nacos.shaded.io.grpc.internal.DelayedClientCall$DelayedListener$
    3.run(DelayedClientCall.java:471) ~[nacos-client-2.2.1.jar:na]
12
13        // ...
14 Caused by:
    com.alibaba.nacos.shaded.io.grpc.netty.shaded.io.netty.channel.AbstractChann
    el$AnnotatedConnectException: Connection refused: no further information:
    /127.0.0.1:11020
15 Caused by: java.net.ConnectException: Connection refused: no further
    information
16        at java.base/sun.nio.ch.Net.pollConnect(Native Method) ~[na:na]
17        at java.base/sun.nio.ch.Net.pollConnectNow(Net.java:672) ~[na:na]
18        at
    java.base/sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:
    946) ~[na:na]
19
20

```

7.3 配置中心详解

7.3.1 设置命名空间

Nacos配置管理的命名空间和服务列表的命名空间是分别设置的. 默认是public

Nacos命名空间配置依然在bootstrap.properties中进行配置

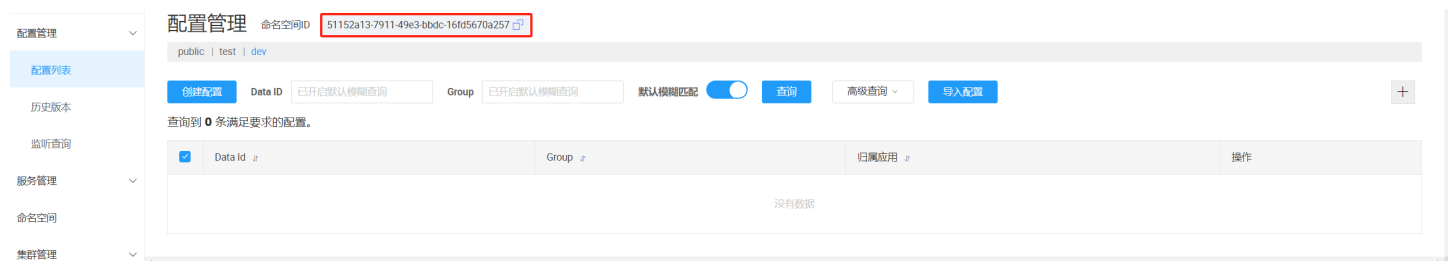
```
1 spring.cloud.nacos.config.namespace=51152a13-7911-49e3-bbdc-16fd5670a257
```

对应bootstrap.yml配置

```
1 spring:
2   cloud:
```

```
3      nacos:
4      config:
5      namespace: 51152a13-7911-49e3-bbdc-16fd5670a257
```

value对应 命名空间的ID, 如下图所示:

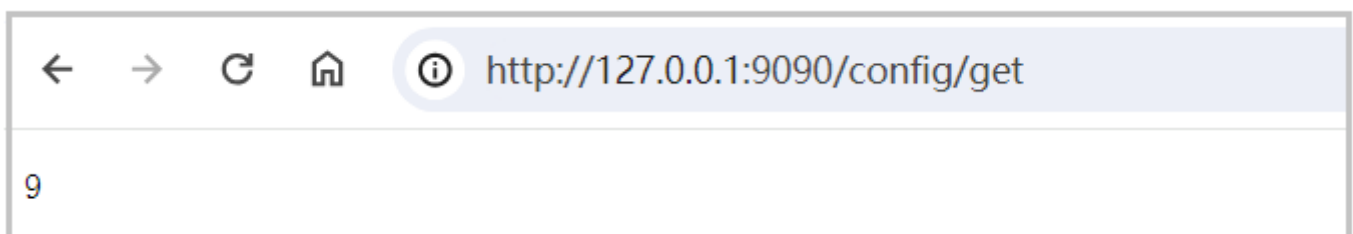


如果设置命名空间后, 项目启动时, 会从该命名空间下找对应的配置项。



重新访问接口, 观察结果:

<http://127.0.0.1:9090/config/get>



7.3.2 Data Id

Data Id 格式介绍


在 Nacos Spring Cloud 中, `dataId` 的完整格式如下:

```
1 ${prefix}-${spring.profiles.active}.${file-extension}
```

- `prefix` 默认为 `spring.application.name` 的值, 也可以通过配置项 `spring.cloud.nacos.config.prefix` 来配置.
- `spring.profiles.active` 即为当前环境对应的 profile. 当 `spring.profiles.active` 为空时, 对应的连接符 `-` 也将不存在, dataId 的拼接格式变成 `${prefix}.${file-extension}`
- `file-extension` 为配置内容的数据格式, 可以通过配置项 `spring.cloud.nacos.config.file-extension` 来配置. 目前只支持 `properties` 和 `yaml` 类型. 默认为 `properties`.

微服务启动时, 会从Nacos读取多个配置文件:

1. `${prefix}-${spring.profiles.active}.${file-extension}` 如: `product-service-dev.properties`
2. `${prefix}.${file-extension}` , 如: `product-service.properties`
3. `${prefix}` 如 `product-service`

 `${spring.application.name}`, `${spring.profiles.active}` 等通过配置文件来指定时, 必须放在 `bootstrap.properties` 文件中

观察日志

在 `bootstrap.yml` 中添加 `spring.profiles.active` 值

```
1 spring:
2   profiles:
3     active: dev
```

启动服务, 观察日志

```
1 2023-12-28T18:48:08.614+08:00 INFO 36672 --- [          main]
   c.a.n.client.config.impl.ClientWorker    : [fixed-51152a13-7911-49e3-bbdc-
   16fd5670a257-110.41.51.65_10020] [subscribe] product-service-
   dev.properties+DEFAULT_GROUP+51152a13-7911-49e3-bbdc-16fd5670a257
2 2023-12-28T18:48:08.624+08:00 INFO 36672 --- [          main]
   c.a.nacos.client.config.impl.CacheData   : [fixed-51152a13-7911-49e3-bbdc-
   16fd5670a257-110.41.51.65_10020] [add-listener] ok, tenant=51152a13-7911-49e3-
```

```
bbdc-16fd5670a257, dataId=product-service-dev.properties, group=DEFAULT_GROUP,
cnt=1
3 2023-12-28T18:48:08.624+08:00 INFO 36672 --- [          main]
c.a.c.n.refresh.NacosContextRefresher : [Nacos Config] Listening config:
dataId=product-service-dev.properties, group=DEFAULT_GROUP
4 2023-12-28T18:48:08.625+08:00 INFO 36672 --- [          main]
c.a.n.client.config.impl.ClientWorker : [fixed-51152a13-7911-49e3-bbdc-
16fd5670a257-110.41.51.65_10020] [subscribe] product-
service.properties+DEFAULT_GROUP+51152a13-7911-49e3-bbdc-16fd5670a257
5 2023-12-28T18:48:08.625+08:00 INFO 36672 --- [          main]
c.a.nacos.client.config.impl.CacheData : [fixed-51152a13-7911-49e3-bbdc-
16fd5670a257-110.41.51.65_10020] [add-listener] ok, tenant=51152a13-7911-49e3-
bbdc-16fd5670a257, dataId=product-service.properties, group=DEFAULT_GROUP,
cnt=1
6 2023-12-28T18:48:08.625+08:00 INFO 36672 --- [          main]
c.a.c.n.refresh.NacosContextRefresher : [Nacos Config] Listening config:
dataId=product-service.properties, group=DEFAULT_GROUP
7 2023-12-28T18:48:08.626+08:00 INFO 36672 --- [          main]
c.a.n.client.config.impl.ClientWorker : [fixed-51152a13-7911-49e3-bbdc-
16fd5670a257-110.41.51.65_10020] [subscribe] product-
service+DEFAULT_GROUP+51152a13-7911-49e3-bbdc-16fd5670a257
8 2023-12-28T18:48:08.627+08:00 INFO 36672 --- [          main]
c.a.nacos.client.config.impl.CacheData : [fixed-51152a13-7911-49e3-bbdc-
16fd5670a257-110.41.51.65_10020] [add-listener] ok, tenant=51152a13-7911-49e3-
bbdc-16fd5670a257, dataId=product-service, group=DEFAULT_GROUP, cnt=1
9 2023-12-28T18:48:08.627+08:00 INFO 36672 --- [          main]
c.a.c.n.refresh.NacosContextRefresher : [Nacos Config] Listening config:
dataId=product-service, group=DEFAULT_GROUP
10
```

三个文件的优先级为: **product-service-dev.properties** > **product-service.properties** > **product-service**

测试

bootstrap.yml配置如下:

```
1 spring:
2   application:
3     name: product-service
4   profiles:
5     active: dev
6   cloud:
7     nacos:
```

```
8      config:
9          server-addr: 110.41.51.65:10020
10         namespace: 51152a13-7911-49e3-bbdc-16fd5670a257
```

配置项如下:

The screenshot shows the Nacos configuration management interface. At the top, the namespace is set to 51152a13-7911-49e3-bbdc-16fd5670a257. Below the search bar, there are three configurations listed:

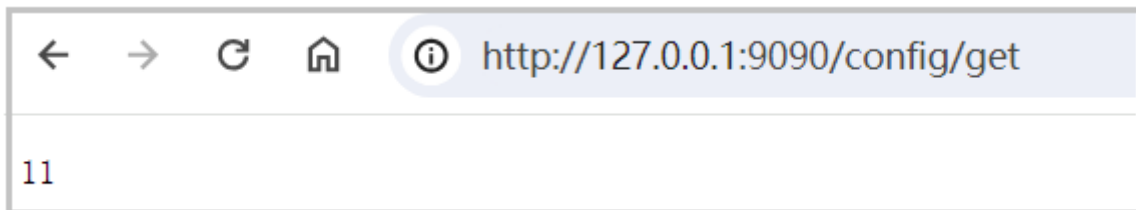
Data Id	Group
product-service	DEFAULT_GROUP
product-service.properties	DEFAULT_GROUP
product-service-dev.properties	DEFAULT_GROUP

Red arrows point from each configuration name to its corresponding detail view on the right:

- product-service: Configuration content is `1 nacos.test.num=9`
- product-service.properties: Configuration content is `1 nacos.test.num=10`
- product-service-dev.properties: Configuration content is `1 nacos.test.num=11`

访问接口: <http://127.0.0.1:9090/config/get>

服务获取到了 `product-service-dev.properties` 的值



删除 `product-service-dev.properties` 配置, 再次访问接口



注意:

1. bootstrap.yml 设置的配置格式必须和nacos控制台配置的数据格式保持一致.
2. 不设置配置格式(`spring.cloud.nacos.config.file-extension`)时, 默认为**properties**

8. 服务部署

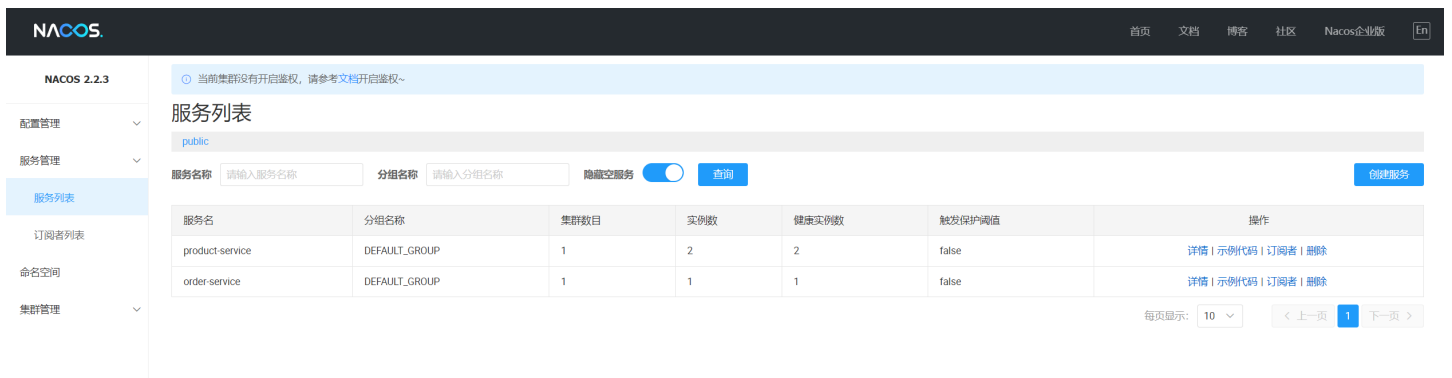
1. 修改数据库, Nacos等相关配置
2. 对两个服务进行打包
3. 上传jar到Linux服务器
4. 启动Nacos

启动前最好把data数据删除掉.

5. 启动服务

```
1 #后台启动order-service, 并设置输出日志到logs/order.log
2 nohup java -jar order-service.jar >logs/order.log &
3
4 #后台启动product-service, 并设置输出日志到logs/order.log
5 nohup java -jar product-service.jar >logs/product-9090.log &
6
7 #启动实例, 指定端口号为9091
8 nohup java -jar product-service.jar --server.port=9091 >logs/product-9091.log &
```

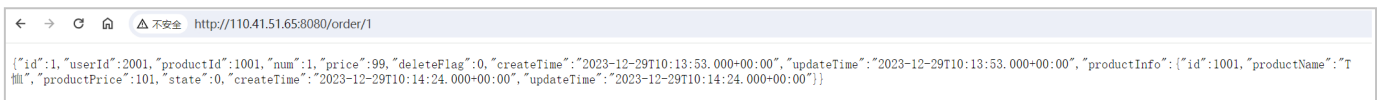
观察Nacos控制台



6. 测试

访问接口: <http://110.41.51.65:8080/order/1>

观察远程调用的结果:



9. Nacos与Eureka的区别

共同点:

- 都支持服务注册和服务拉取

区别:

1. 功能

Nacos除了服务发现和注册之外, 还提供了配置中心, 流量管理和DNS服务等功能.

2. CAP理论

Eureka遵循AP原则, Nacos可以切换AP和CP模式,默认AP.

Nacos 根据配置识别CP或者AP模式. 如果注册Nacos的Client的节点是临时节点, 那么Nacos对这个Client节点的效果就是AP, 反之是CP. AP和CP可以同时混合存在.

3. 服务发现

Eureka: 基于拉模式. Eureka Client会定期从Server拉取服务信息, 有缓存, 默认每30秒拉取一次.

Nacos: 基于推送模式. 服务列表有变化时实时推送给订阅者, 服务端和客户端保持心跳连接.

服务启动:

```
1 11:02:05.685+08:00 INFO 3544 --- [110.41.51.65-45]
  com.alibaba.nacos.common.remote.client : [e41cbcc4-e986-428c-bd30-
  f1e4820e4084] Receive server push request, request =
  NotifySubscriberRequest, requestId = 2
2 11:02:05.686+08:00 INFO 3544 --- [110.41.51.65-45]
  com.alibaba.nacos.client.naming          : new ips(1) service:
  DEFAULT_GROUP@@product-service ->
  [{"ip":"192.168.31.107","port":9092,"weight":1.0,"healthy":true,"enabled":tr
  ue,"ephemeral":true,"clusterName":"BJ","serviceName":"DEFAULT_GROUP@@product
  -service","metadata":
  {"preserved.register.source":"SPRING_CLOUD"},"instanceHeartBeatTimeout":1500
  0,"instanceHeartBeatInterval":5000,"ipDeleteTimeout":30000}]
3 11:02:05.687+08:00 INFO 3544 --- [110.41.51.65-45]
  com.alibaba.nacos.client.naming          : current ips:(2) service:
  DEFAULT_GROUP@@product-service ->
  [{"ip":"192.168.31.107","port":9090,"weight":1.0,"healthy":true,"enabled":tr
  ue,"ephemeral":true,"clusterName":"SH","serviceName":"DEFAULT_GROUP@@product
  -service","metadata":
  {"preserved.register.source":"SPRING_CLOUD"},"instanceHeartBeatTimeout":1500
  0,"instanceHeartBeatInterval":5000,"ipDeleteTimeout":30000},
  {"ip":"192.168.31.107","port":9092,"weight":1.0,"healthy":true,"enabled":tru
  e,"ephemeral":true,"clusterName":"BJ","serviceName":"DEFAULT_GROUP@@product-
  service","metadata":
  {"preserved.register.source":"SPRING_CLOUD"},"instanceHeartBeatTimeout":1500
  0,"instanceHeartBeatInterval":5000,"ipDeleteTimeout":30000}]
4 11:02:05.724+08:00 INFO 3544 --- [110.41.51.65-45]
  com.alibaba.nacos.common.remote.client : [e41cbcc4-e986-428c-bd30-
  f1e4820e4084] Ack server push request, request = NotifySubscriberRequest,
  requestId = 2
```


服务停止:

```
1 2023-12-29T11:02:58.839+08:00 INFO 3544 --- [110.41.51.65-59]
  com.alibaba.nacos.common.remote.client : [e41cbcc4-e986-428c-bd30-
  f1e4820e4084] Receive server push request, request =
  NotifySubscriberRequest, requestId = 3
2 2023-12-29T11:02:58.839+08:00 INFO 3544 --- [110.41.51.65-59]
  com.alibaba.nacos.client.naming          : removed ips(1) service:
  DEFAULT_GROUP@@product-service ->
  [{"ip":"192.168.31.107","port":9092,"weight":1.0,"healthy":true,"enabled":tr
  ue,"ephemeral":true,"clusterName":"BJ","serviceName":"DEFAULT_GROUP@@product
  -service","metadata":
  {"preserved.register.source":"SPRING_CLOUD"},"instanceHeartBeatTimeOut":1500
  0,"instanceHeartBeatInterval":5000,"ipDeleteTimeout":30000}]
3 2023-12-29T11:02:58.840+08:00 INFO 3544 --- [110.41.51.65-59]
  com.alibaba.nacos.client.naming          : current ips:(1) service:
  DEFAULT_GROUP@@product-service ->
  [{"ip":"192.168.31.107","port":9090,"weight":1.0,"healthy":true,"enabled":tr
  ue,"ephemeral":true,"clusterName":"SH","serviceName":"DEFAULT_GROUP@@product
  -service","metadata":
  {"preserved.register.source":"SPRING_CLOUD"},"instanceHeartBeatTimeOut":1500
  0,"instanceHeartBeatInterval":5000,"ipDeleteTimeout":30000}]
4 2023-12-29T11:02:58.842+08:00 INFO 3544 --- [110.41.51.65-59]
  com.alibaba.nacos.common.remote.client : [e41cbcc4-e986-428c-bd30-
  f1e4820e4084] Ack server push request, request = NotifySubscriberRequest,
  requestId = 3
```