1. Nacos简介

1.1. 为什么叫Nacos

前四个字母分别为Naming和Configuration的前两个字母，最后的s为Service

1.2. 是什么

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中心

Nacos：Dynamic Naming and Configuration Service

Nacos就是注册中心+配置中心的组合 => 等价于: Nacos = Eureka+Config+Bus

1.3. 能干嘛

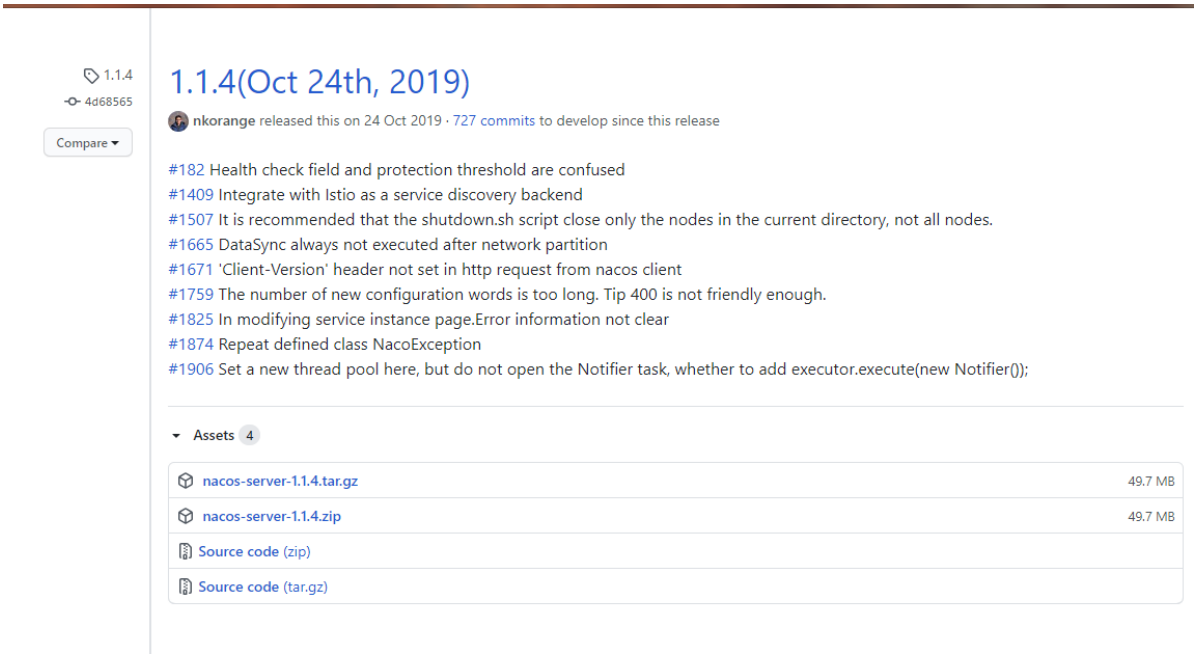
替代Eureka做服务注册中心

替代Config做服务配置中心

1.4. 下载

官网:

> https://github.com/alibaba/Nacos



官方下载地址

> https://github.com/alibaba/nacos/releases

官网文档:

> https://nacos.io/zh-cn/index.html
>
> https://spring-cloud-alibaba-group.github.io/github-pages/greenwich/spring-cloud-alibaba.html#_spring_cloud_alibaba_nacos_discovery

1.5. 各种注册中心比较

服务注册与发现框架	CAP 模型	控制台管理	社区活跃度
Eureka	AP	支持	低 (2.x 版本闭源)
Zookeeper	CP	不支持	中
Consul	CP	支持	高
Nacos	AP	支持	高

据说 Nacos在阿里巴巴内部有超过10万的实例运行，已经过了类似双十一等各种大型流量的考验

2. 安装并运行Nacos

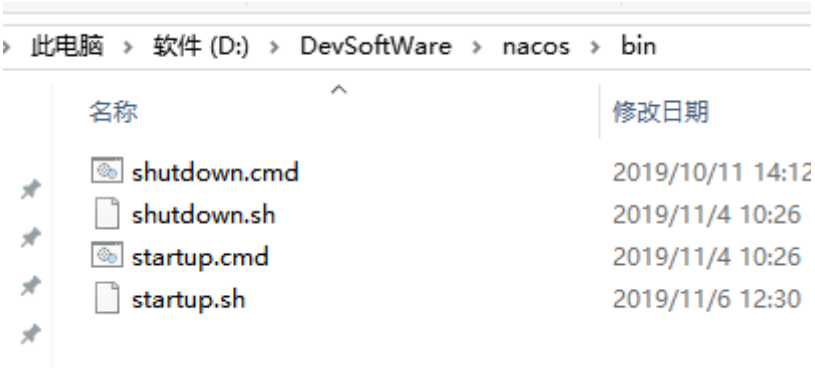
2.1. 下载/安装

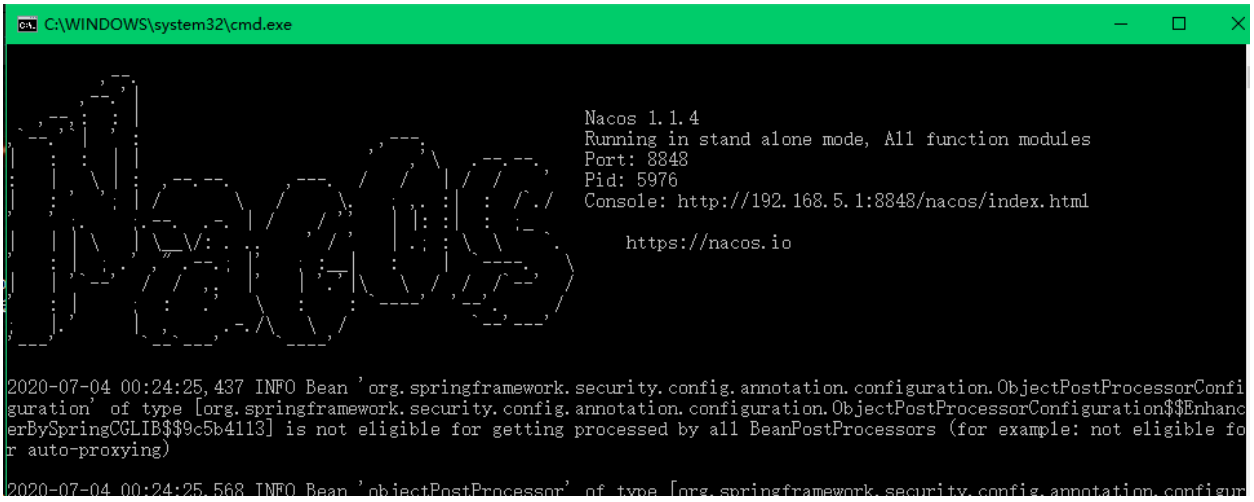
官网: https://github.com/alibaba/nacos/releases/tag/1.1.4

> 这里使用的Nacos版本与视频课程中的版本一致, 均为1.1.4

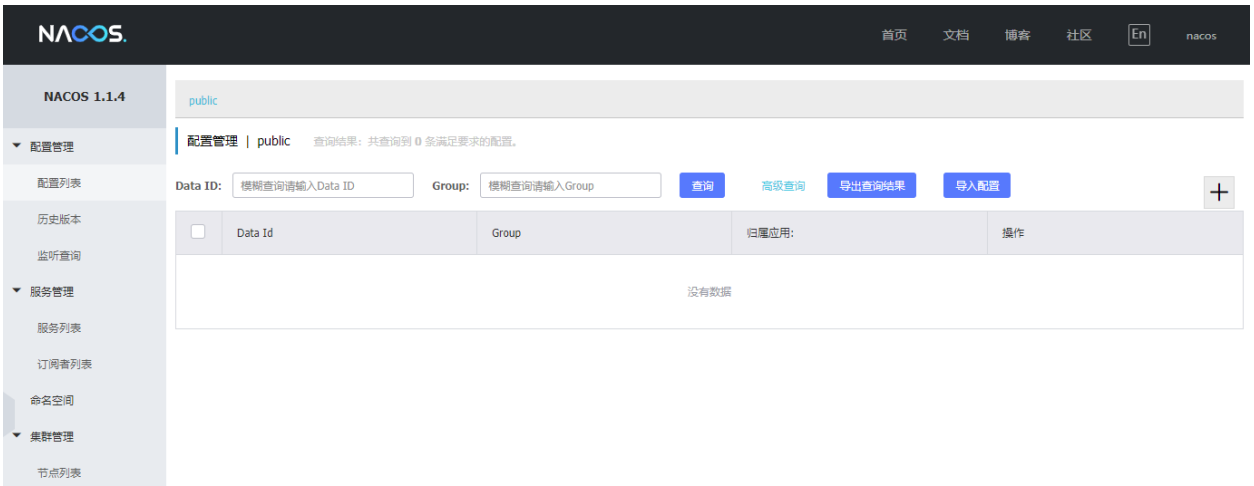
2.2. 使用

解压安装包，直接运行bin目录下的startup.cmd





命令运行成功后直接访问http://localhost:8848/nacos



默认账号密码都是nacos

3. Nacos作为服务注册中心演示

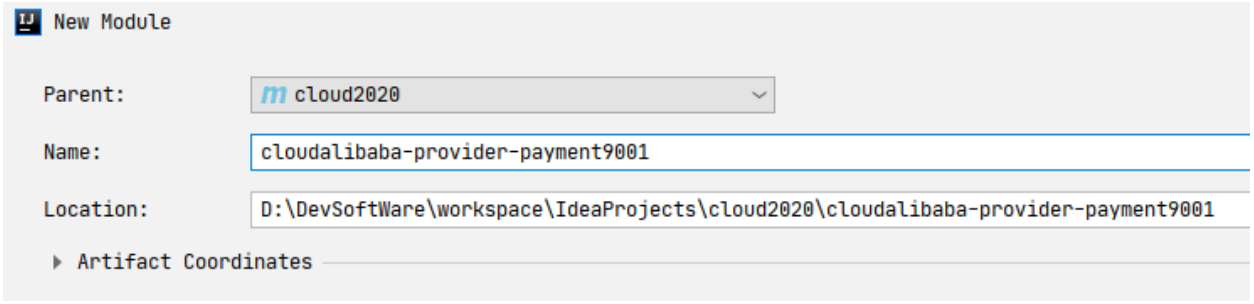
3.1. 官方文档

> https://spring-cloud-alibaba-group.github.io/github-pages/greenwich/spring-cloud-alibaba.html

3.2. 基于Nacos的服务提供者

3.2.1. 新建Module

新建cloudalibaba-provider-payment9001



3.2.2. POM

3.2.2.1. 父模块POM

修改内容:

```
            <!--spring cloud alibaba 2.1.0.RELEASE-->
            <dependency>
                <groupId>com.alibaba.cloud</groupId>
                <artifactId>spring-cloud-alibaba-dependencies</artifactId>
                <version>2.1.0.RELEASE</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
```

完整内容:

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/200
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.
    <modelVersion>4.0.0</modelVersion>

    <groupId>cn.sitedev.springcloud</groupId>
```

```
    <artifactId>cloud2020</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>pom</packaging>

    <modules>
        <module>cloud-provider-payment8001</module>
        <module>cloud-provider-payment8002</module>
        <module>cloud-consumer-order80</module>
        <module>cloud-api-commons</module>
        <module>cloud-eureka-server7001</module>
        <module>cloud-eureka-server7002</module>
        <module>cloud-provider-payment8004</module>
        <module>cloud-consumerzk-order80</module>
        <module>cloud-providerconsul-payment8006</module>
        <module>cloud-consumerconsul-order80</module>
        <module>cloud-consumer-feign-order80</module>
        <module>cloud-provider-hystrix-payment8001</module>
        <module>cloud-consumer-feign-hystrix-order80</module>
        <module>cloud-consumer-hystrix-dashboard9001</module>
        <module>cloud-gateway-gateway9527</module>
        <module>cloud-config-center-3344</module>
        <module>cloud-config-client-3355</module>
        <module>cloud-config-client-3366</module>
        <module>cloud-stream-rabbitmq-provider8801</module>
        <module>cloud-stream-rabbitmq-consumer8802</module>
        <module>cloud-stream-rabbitmq-consumer8803</module>
        <module>cloudalibaba-provider-payment9001</module>
    </modules>

    <!--统一管理jar包版本-->
    <;properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <junit.version>4.12</junit.version>
        <log4j.version>1.2.17</log4j.version>
        <lombok.version>1.16.18</lombok.version>
        <mysql.version>5.1.47</mysql.version>
        <druid.version>1.1.16</druid.version>
        <spring.boot.version>2.2.2.RELEASE</spring.boot.version>
        <spring.cloud.version>Hoxton.SR1</spring.cloud.version>
        <spring.cloud.alibaba.version>2.1.0.RELEASE</spring.cloud.alibaba.version>
        <mybatis.spring.boot.version>1.3.0</mybatis.spring.boot.version>
```

```
    </properties>

    <!--子模块继承后,提供作用:锁定版本+子module不用groupId和version-->
    <dependencyManagement>
        <dependencies>
            <!--springboot 2.2.2-->
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-dependencies</artifactId>
                <version>${spring.boot.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <!--Spring cloud Hoxton.SR1-->
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring.cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <!--Spring cloud alibaba 2.1.0.RELEASE-->
            <dependency>
                <groupId>com.alibaba.cloud</groupId>
                <artifactId>spring-cloud-alibaba-dependencies</artifactId>
                <version>${spring.cloud.alibaba.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>${junit.version}</version>
                <scope>test</scope>
            </dependency>
            <dependency>
                <groupId>log4j</groupId>
                <artifactId>log4j</artifactId>
                <version>${log4j.version}</version>
            </dependency>
            <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
```

```
                <version>${mysql.version}</version>
            </dependency>
            <dependency>
                <groupId>com.alibaba</groupId>
                <artifactId>druid</artifactId>
                <version>${druid.version}</version>
            </dependency>
            <dependency>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <version>${lombok.version}</version>
            </dependency>
            <dependency>
                <groupId>org.mybatis.spring.boot</groupId>
                <artifactId>mybatis-spring-boot-starter</artifactId>
                <version>${mybatis.spring.boot.version}</version>
            </dependency>
            <!--spring cloud alibaba 2.1.0.RELEASE-->
            <dependency>
                <groupId>com.alibaba.cloud</groupId>
                <artifactId>spring-cloud-alibaba-dependencies</artifactId>
                <version>2.1.0.RELEASE</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>${spring.boot.version}</version>
                <configuration>
                    <fork>true</fork>
                    <addResources>true</addResources>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <!--第三方maven私服-->
```

```
    <repositories>
        <repository>
            <id>nexus-aliyun</id>
            <name>Nexus aliyun</name>
            <url>http://maven.aliyun.com/nexus/content/groups/public</url>
            <releases>
                <enabled>true</enabled>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>
</project>
```

3.2.2.2. 本模块POM

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.o
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloudalibaba-provider-payment9001</artifactId>
    <dependencies>
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
```

```
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.62</version>
        </dependency>
    </dependencies>
</project>
```

3.2.3. YML

```
server:
  port: 9001
spring:
  application:
    name: nacos-payment-provider
  cloud:
    nacos:
      discovery:
        server-addr: localhost:8848 #配置Nacos地址

management:
  endpoints:
    web:
      exposure:
```

```
        include: '*'
```

3.2.4. 主启动

```
package cn.sitedev.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@EnableDiscoveryClient
@SpringBootApplication
public class PaymentMain9001 {
    public static void main(String[] args) {
        SpringApplication.run(PaymentMain9001.class, args);
    }
}
```

3.2.5. 业务类

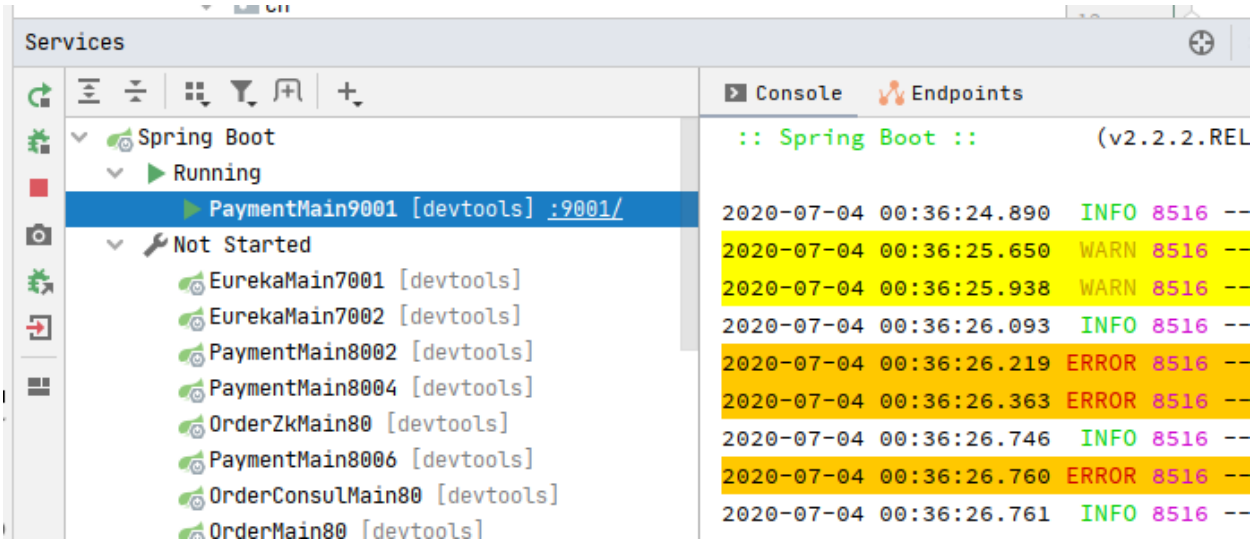
3.2.5.1. Controller

```
package cn.sitedev.springcloud.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;


@RestController
public class PaymentController {
    @Value("${server.port}")
    private String serverPort;

    @GetMapping(value = "/payment/nacos/{id}")
    public String getPayment(@PathVariable("id") Integer id) {
        return "nacos registry, serverPort: " + serverPort + "\t id" + id;
    }
```

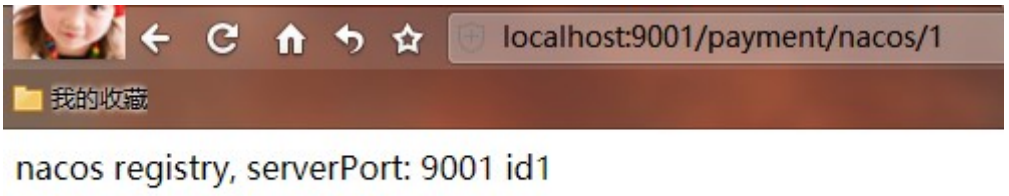
```
18 }
```

3.2.6. 测试

启动cloudalibaba-provider-payment9001应用



浏览器访问http://localhost:9001/payment/nacos/1



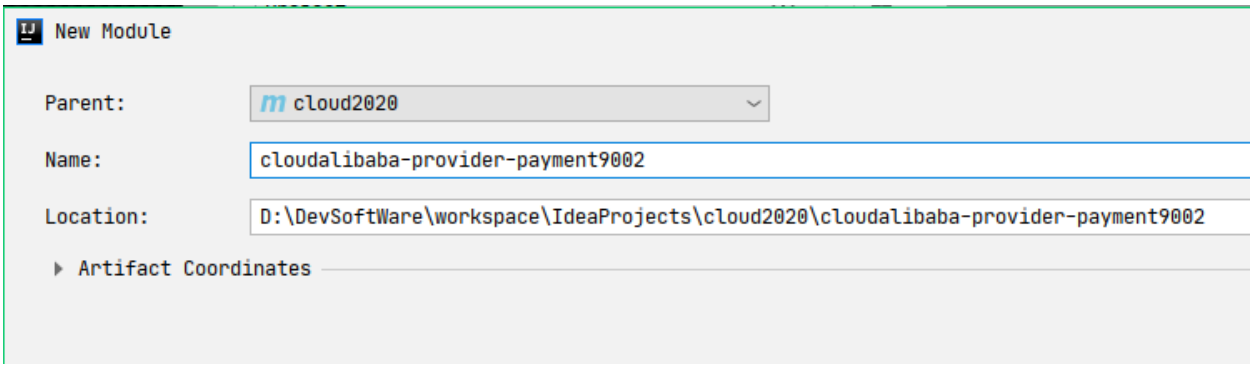
浏览器访问nacos管理台http://localhost:8848/nacos/，服务管理-> 服务列表

可以看到nacos服务注册中心+服务提供者9001都ok了

3.2.7. 为了下一章节演示nacos的负载均衡，参照9001新建9002

3.2.7.1. 新建Module

新建cloudalibaba-provider-payment9002



3.2.7.2. POM

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloudalibaba-provider-payment9002</artifactId>

    <dependencies>
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
```

```
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.62</version>
        </dependency>
    </dependencies>
</project>
```

3.2.7.3. YML

```
server:
  port: 9002
spring:
  application:
    name: nacos-payment-provider
  cloud:
    nacos:
      discovery:
        server-addr: localhost:8848 #配置Nacos地址

management:
  endpoints:
    web:
      exposure:
        include: '*'
```

3.2.7.4. 主启动

```
```

```java
package cn.sitedev.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@EnableDiscoveryClient
@SpringBootApplication
public class PaymentMain9002 {
    public static void main(String[] args) {
        SpringApplication.run(PaymentMain9002.class, args);
    }
}
```

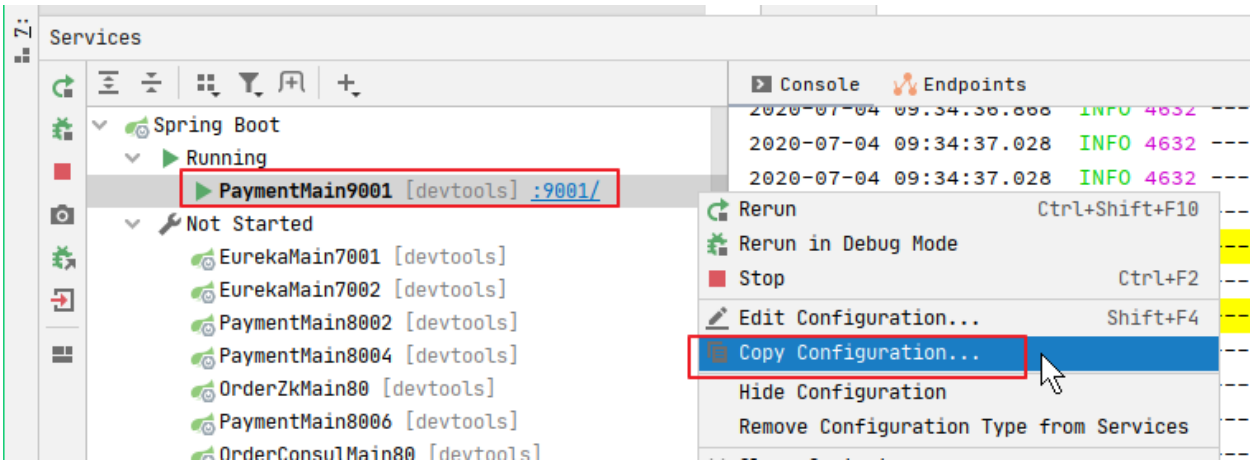
3.2.7.5. 业务类

3.2.7.5.1. Controller

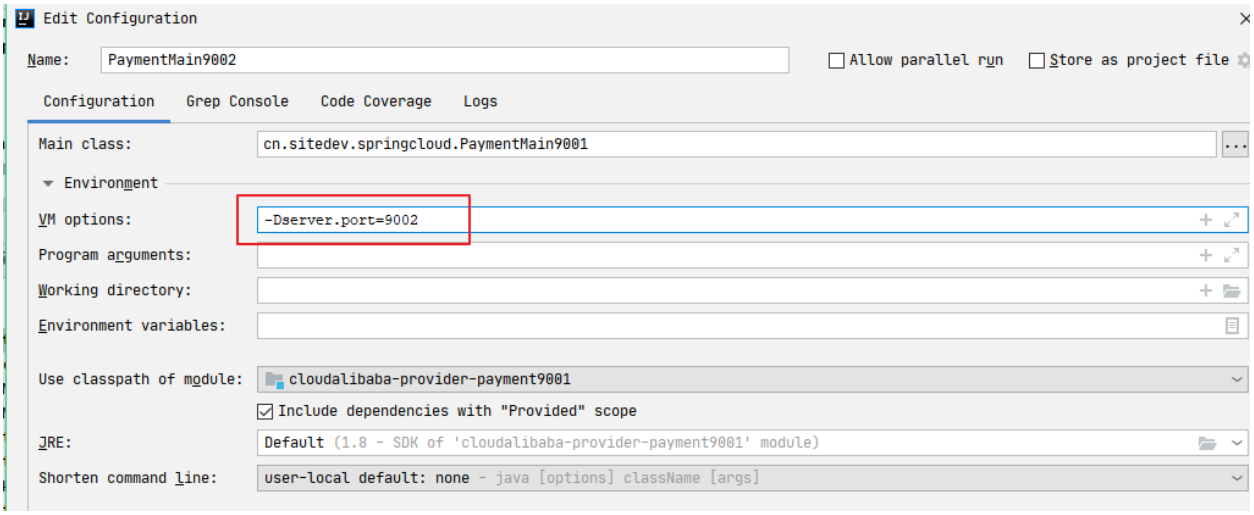
```java
package cn.sitedev.springcloud.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;


@RestController
public class PaymentController {
    @Value("${server.port}")
    private String serverPort;

    @GetMapping(value = "/payment/nacos/{id}")
    public String getPayment(@PathVariable("id") Integer id) {
        return "nacos registry, serverPort: " + serverPort + "\t id" + id;
    }
}
```

3.2.7.6. 如果取巧不想新建重复体力劳动，可以直接拷贝虚拟端口映射

在Services中找到状态为Running的PaymentMain9001, 鼠标右键, 选择Copy Configuration...



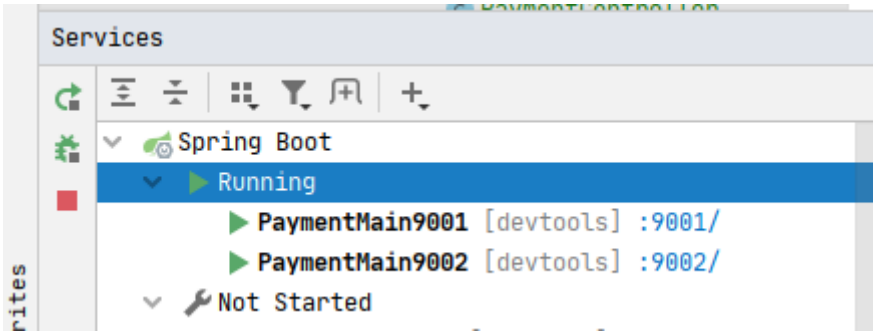
VM options填入：`-Dserver.port=9002`



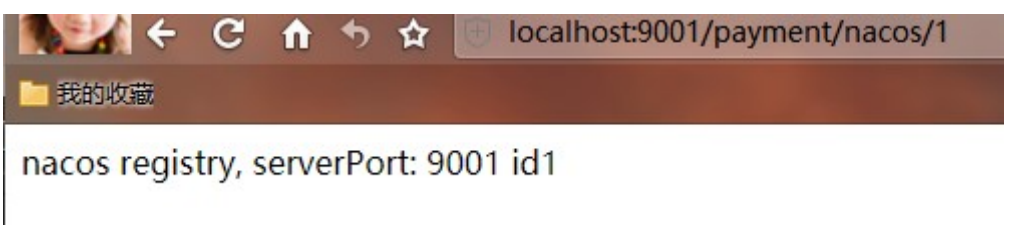
3.2.7.7. 测试

启动PaymentMain9001

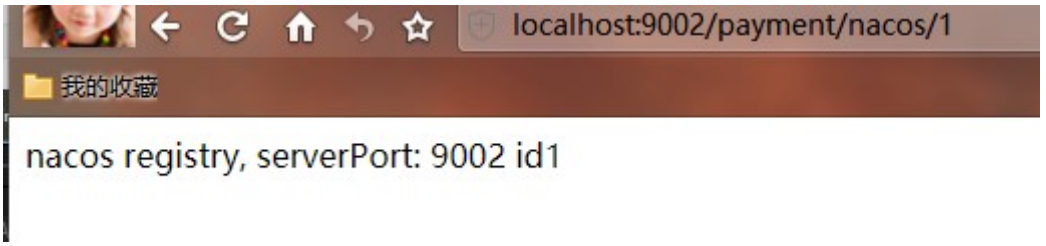
启动PaymentMain9002



浏览器访问http://localhost:9001/payment/nacos/1

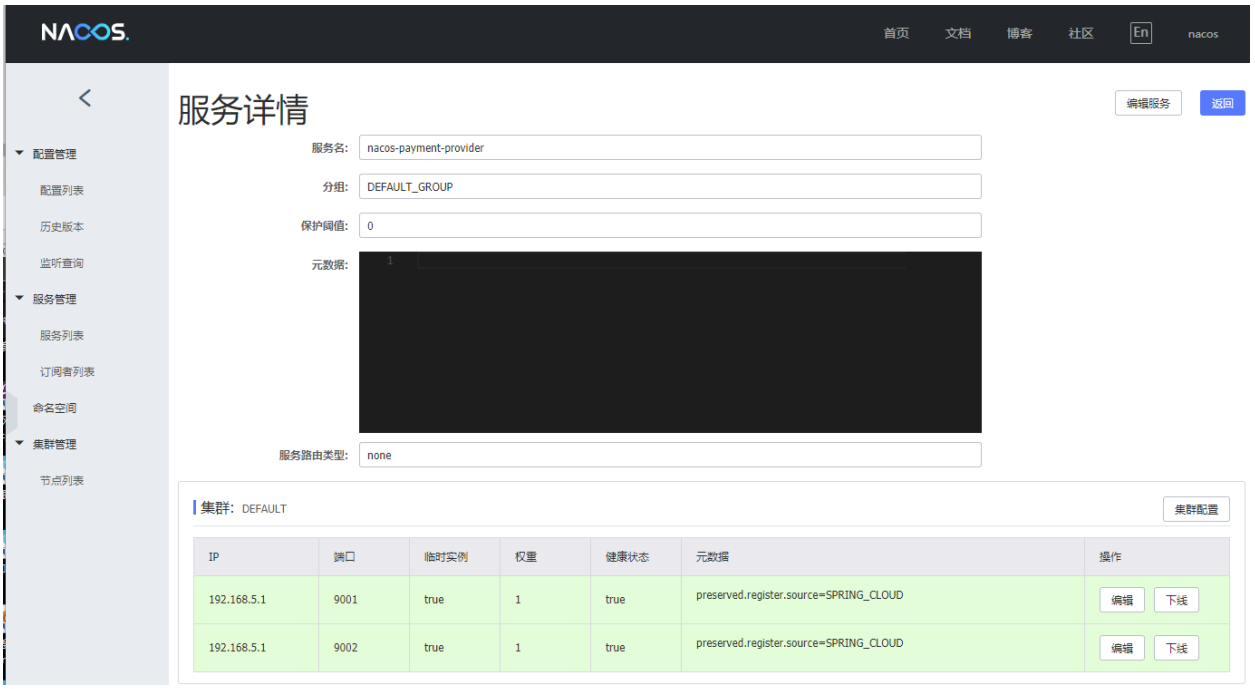


浏览器访问http://localhost:9002/payment/nacos/1



浏览器访问http://localhost:8848/nacos => 服务管理->服务列表

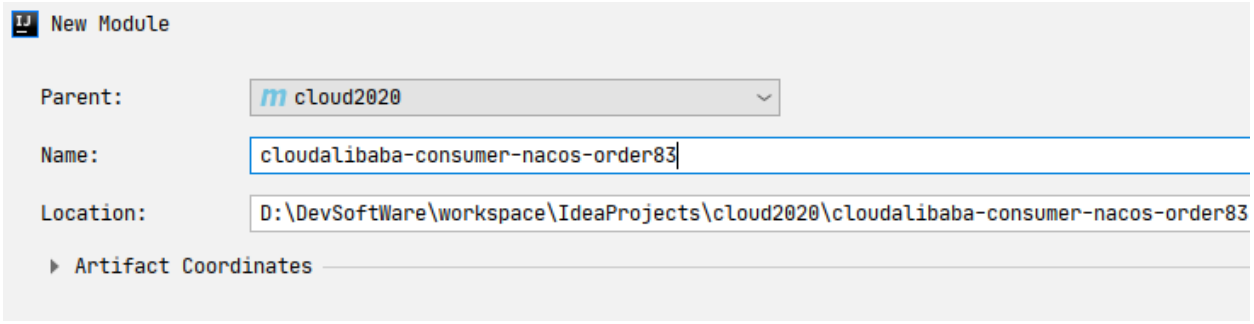
点击"详情"按钮



3.3. 基于Nacos的服务消费者

3.3.1. 新建Module

新建cloudalibaba-consumer-nacos-order83

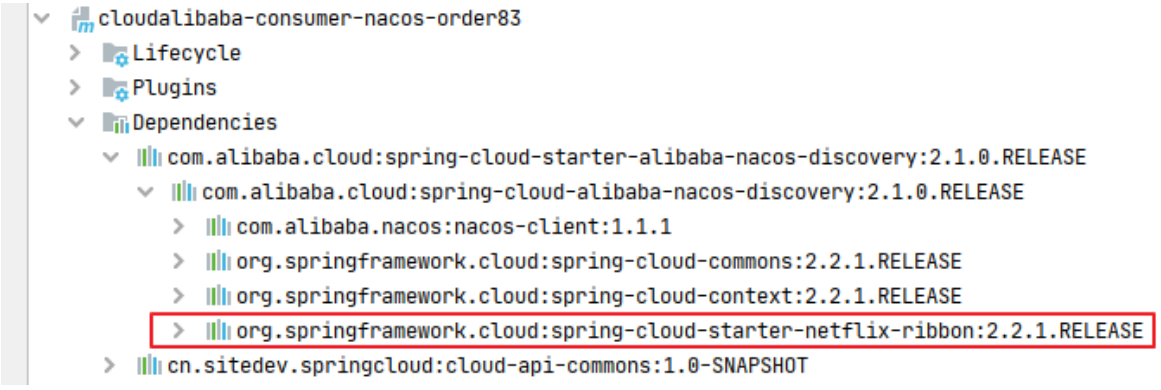


3.3.2. POM

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloudalibaba-consumer-nacos-order83</artifactId>

    <dependencies>
        <!--SpringCloud ailibaba nacos -->
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
        </dependency>
        <dependency>
            <groupId>cn.sitedev.springcloud</groupId>
            <artifactId>cloud-api-commons</artifactId>
            <version>${project.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
```

```xml
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

为什么nacos支持负载均衡



3.3.3. YML

```yaml
server:
  port: 83
spring:
  application:
    name: nacos-order-consumer
  cloud:
    nacos:
      discovery:
        server-addr: localhost:8848 # 配置nacos地址
```

```yaml

# 消费者将会去访问的微服务名称(注册成功进入nacos的微服务提供者)
service-url:
  nacos-user-service: http://nacos-payment-provider
```

3.3.4. 主启动

```java
package cn.sitedev.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;


@EnableDiscoveryClient
@SpringBootApplication
public class OrderNacosMain83 {
    public static void main(String[] args) {
        SpringApplication.run(OrderNacosMain83.class, args);
    }
}
```

3.3.5. 业务类

3.3.5.1. 配置类

```java
package cn.sitedev.springcloud.config;

import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;


@Configuration
public class ApplicationContextConfig {
    @Bean
    @LoadBalanced
    public RestTemplate getRestTemplate() {
```

```
        return new RestTemplate();
    }
}
```

3.3.5.2. Controller

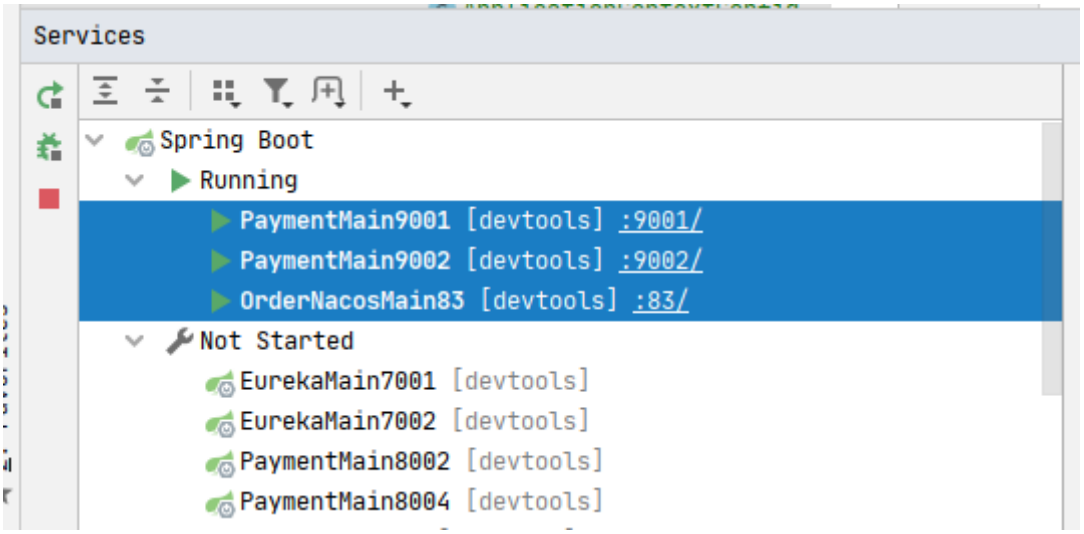
```
package cn.sitedev.springcloud.controller;

import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

import javax.annotation.Resource;


@RestController
@Slf4j
public class OrderNacosController {
    @Resource
    private RestTemplate restTemplate;

    @Value("${service-url.nacos-user-service}")
    private String serverURL;

    @GetMapping(value = "/consumer/payment/nacos/{id}")
    public String paymentInfo(@PathVariable("id") Long id) {
        return restTemplate.getForObject(serverURL + "/payment/nacos/" + id, String.
    }

}
```

3.3.6. 测试

启动PaymentMain9001

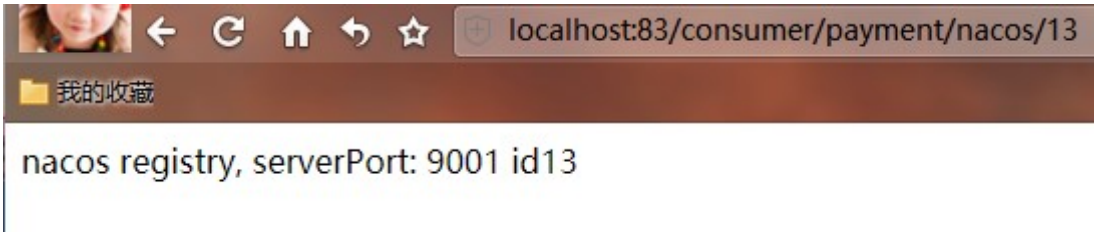
启动PaymentMain9002

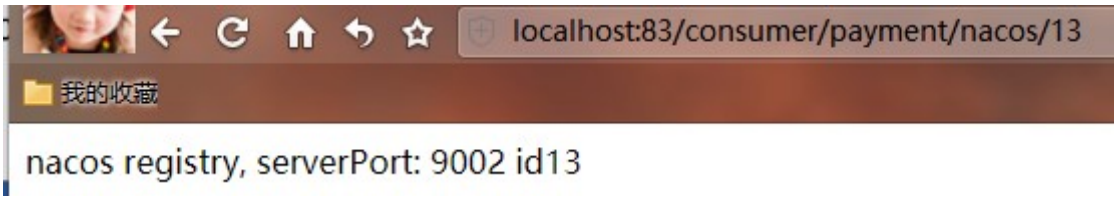
启动OrderNacosMain83



浏览器访问nacos控制台http://localhost:8848/nacos

浏览器访问http://localhost:83/consumer/payment/nacos/13

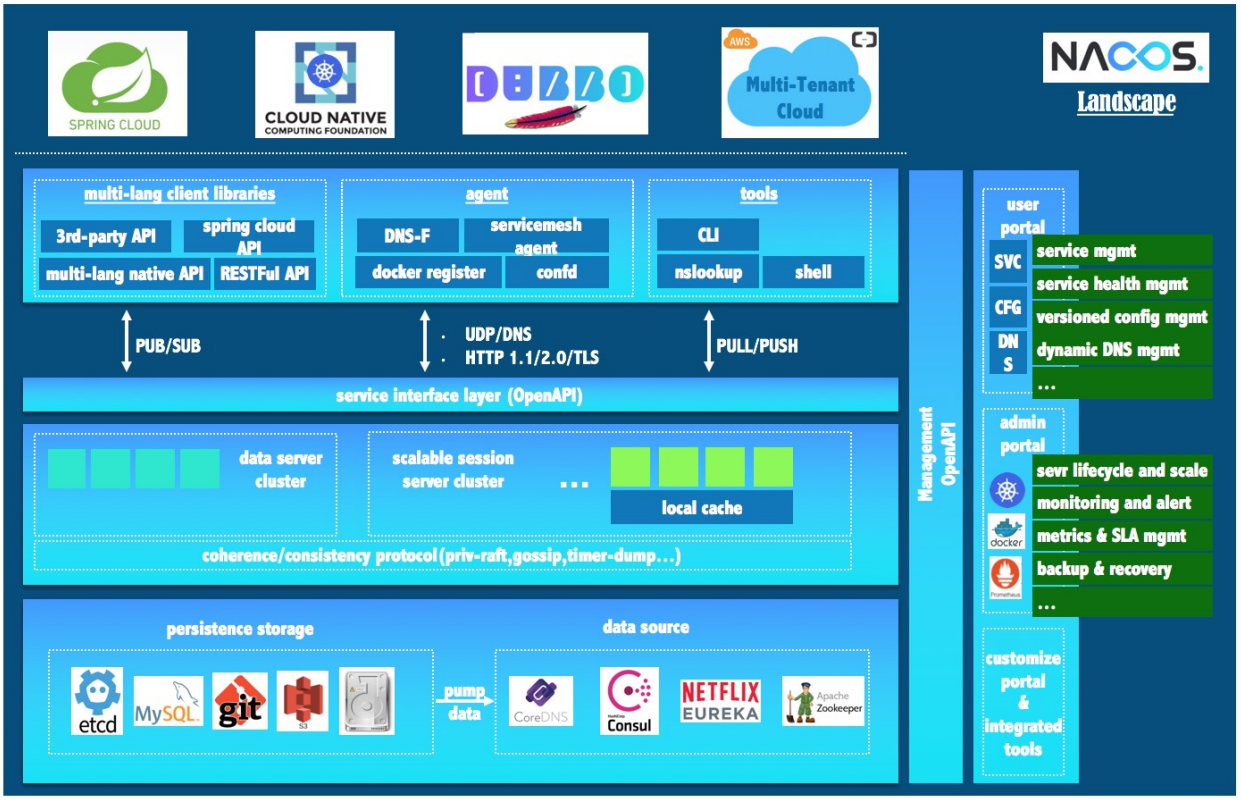




可以看到, 83访问9001/9002，轮询负载OK

3.4. 服务注册中心对比

3.4.1. Nacos全景图



3.4.2. Nacos和CAP

Nacos 与其他注册中心特性对比

	Nacos	Eureka	Consul	CoreDNS	ZooKeeper
一致性协议	CP+AP	AP	CP	/	CP
健康检查	TCP/HTTP/MySQL/Client Beat	Client Beat	TCP/HTTP/gRPC/Cmd	/	Client Beat
负载均衡	权重/DSL/metadata/CMDB	Ribbon	Fabio	RR	/
雪崩保护	支持	支持	不支持	不支持	不支持
自动注销实例	支持	支持	不支持	不支持	支持
访问协议	HTTP/DNS/UDP	HTTP	HTTP/DNS	DNS	TCP
监听支持	支持	支持	支持	不支持	支持
多数据中心	支持	支持	支持	不支持	不支持
跨注册中心	支持	不支持	支持	不支持	不支持
SpringCloud 集成	支持	支持	支持	不支持	不支持
Dubbo 集成	支持	不支持	不支持	不支持	支持
K8s 集成	支持	不支持	支持	支持	不支持

3.4.3. 模式切换

> Nacos支持AP和CP模式的切换

C是所有节点在同一时间看到的数据是一致的；而A的定义是所有的请求都会收到响应。

何时选择使用何种模式?

- 一般来说，如果不需要存储服务级别的信息且服务实例是通过nacos-client注册，并能够保持心跳上报，那么就可以选择AP模式。当前主流的服务如 Spring cloud和 Dubbo服务，都适用于AP模式，AP模式为了服务的可能性而减弱了一致性，因此AP模式下只支持注册临时实例。
- 如果需要在服务级别编辑或者存储配置信息，那么CP是必须，K8S服务和DNS服务则适用于CP模式。
- CP模式下则支持注册持久化实例，此时则是以Raft协议为集群运行模式，该模式下注册实例之前必须先注册服务，如果服务不存在，则会返回错误。

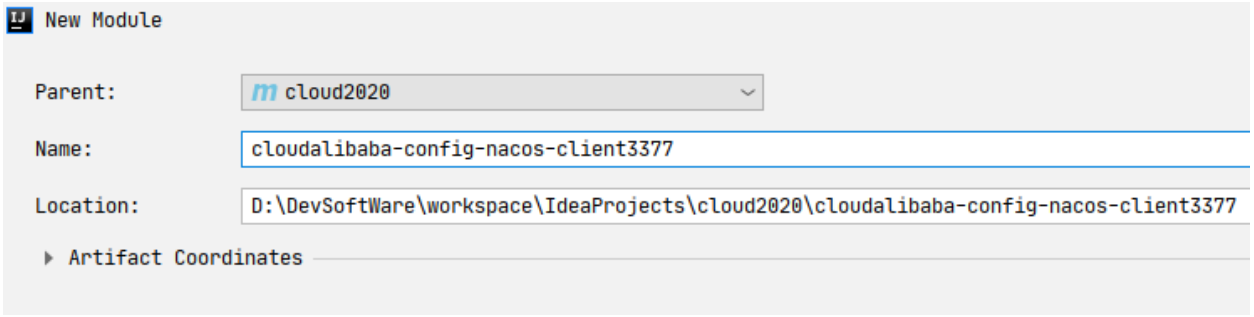
```
curl -X PUT '$NACOS_SERVER:8848/nacos/v1/ns/operator/switches?entry=serverMode&value
```

4. Nacos作为服务配置中心演示

4.1. 基础配置

4.1.1. 新建Module

新建cloudalibaba-config-nacos-client3377



4.1.2. POM

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloudalibaba-config-nacos-client3377</artifactId>

    <dependencies>
        <!--nacos-config-->
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
        </dependency>
        <!--nacos-discovery-->
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
        </dependency>
        <!--web + actuator-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
```

```
        </dependency>
        <!--一般基础配置-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

4.1.3. YML

4.1.3.1. bootstrap.yml

```yaml
server:
  port: 3377

spring:
  application:
    name: nacos-config-client
  cloud:
    nacos:
      discovery:
        server-addr: localhost:8848 #服务注册中心地址
      config:
        server-addr: localhost:8848 #配置中心地址
        file-extension: yaml #指定yaml格式的配置
```

4.1.3.2. application.yml

```yaml
spring:
  profiles:
    active: dev
```

4.1.3.3. 为什么配置两个yml文件

Nacos同 springcloud-config一样，在项目初始化时，要保证先从配置中心进行配置拉取拉取配置之后，才能保证项目的正常启动。

springboot中配置文件的加载是存在优先级顺序的，bootstrap优先级高于 application

4.1.4. 主启动

```java
package cn.sitedev.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@EnableDiscoveryClient
@SpringBootApplication
public class NacosConfigClientMain3377 {
    public static void main(String[] args) {
        SpringApplication.run(NacosConfigClientMain3377.class, args);
    }
}
```

4.1.5. 业务类

4.1.5.1. Controller

```java
package cn.sitedev.springcloud.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.cloud.context.config.annotation.RefreshScope;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

```

```
@RestController
@RefreshScope
public class ConfigClientController {
    @Value("${config.info}")
    private String configInfo;

    @GetMapping("/config/info")
    public String getConfigInfo() {
        return configInfo;
    }
}
```

4.1.5.2. @RefreshScope

通过 Spring Cloud原生注解 @RefreshScope实现配置自动更新

4.1.6. 在Nacos中添加配置信息

Nacos中的匹配规则

4.1.6.1. 理论

Nacos中的dataid的组成格式与SpringBoot配置文件中的匹配规则参见官网

> https://nacos.io/zh-cn/docs/quick-start-spring-cloud.html

说明：之所以需要配置 `spring.application.name` ，是因为它是构成 Nacos 配置管理 `dataId` 字段的一部分。

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

```
${prefix}-${spring.profile.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.profile.active` 即为当前环境对应的 profile，详情可以参考 Spring Boot文档。 **注意：当 `spring.profile.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-exetension` 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。

- 通过 Spring Cloud 原生注解 `@RefreshScope` 实现配置自动更新

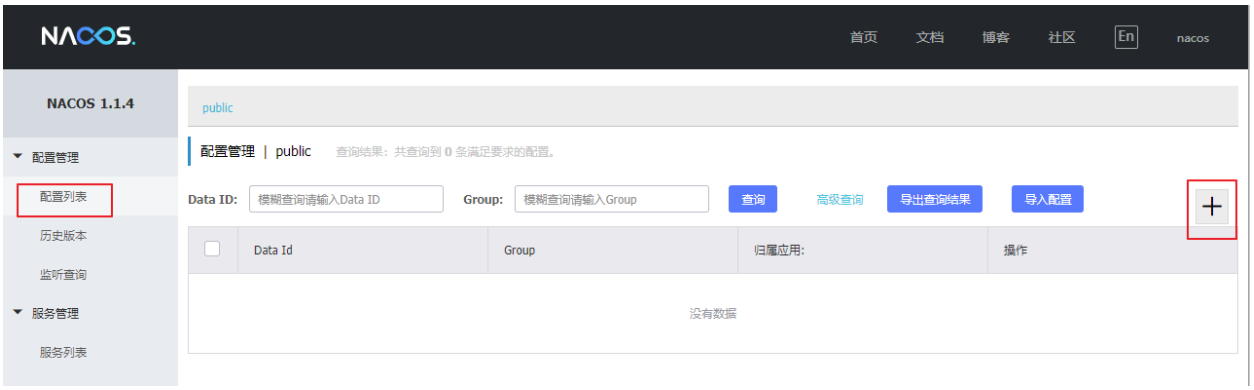
> 公式:
>
> `${spring.application.name}-${spring.profiles.active}.${spring.cloud.nacos.config.file-extension}`
>
> 示例
>
> `nacos-config-client-dev.yml`

4.1.6.2. 实操

4.1.6.2.1. 配置新增

1) 打开nacos管理界面http://localhost:8848/nacos

2) 配置管理 -> 配置列表 -> 点击右侧的"+"按钮



4.1.6.2.2. Nacos界面配置对应

1) Data ID 配置为: `nacos-config-client-dev.yaml`

> 注意不能配置为: `nacos-config-client-dev.yml`

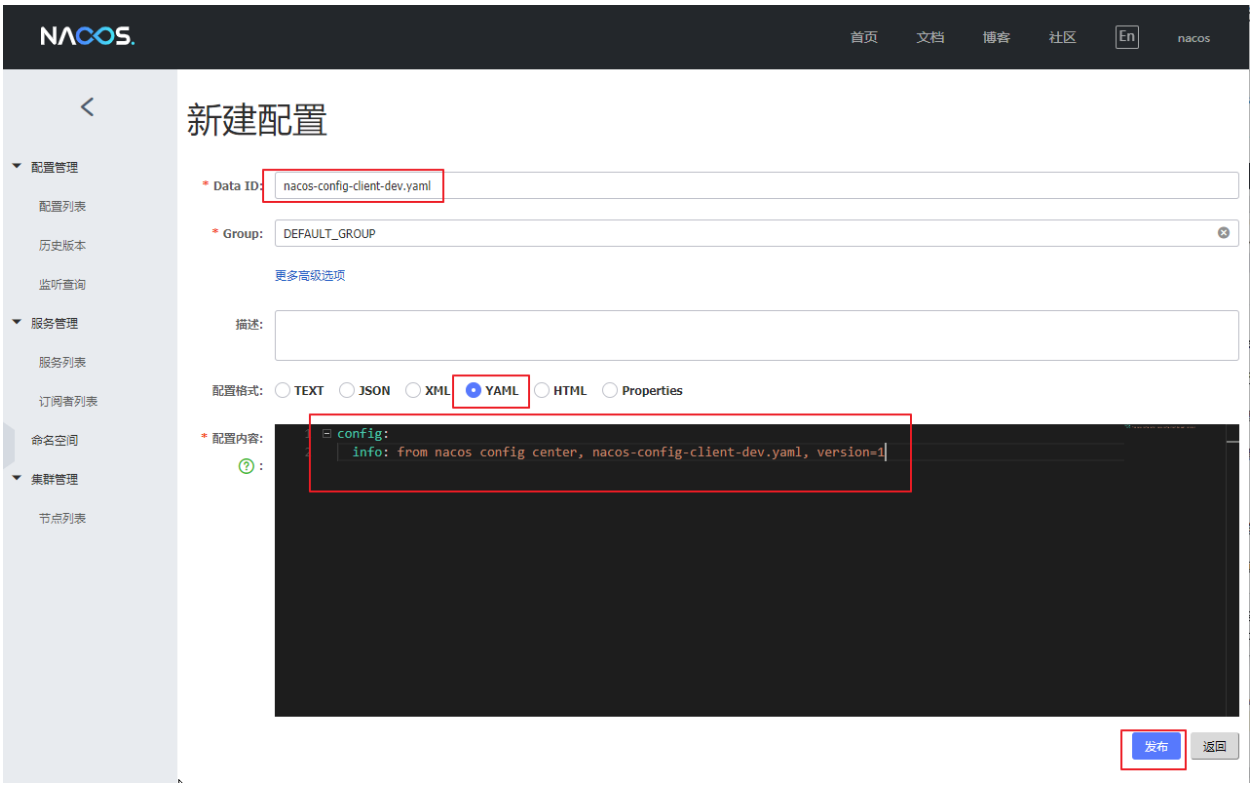
2) Group 为 默认值, 不做修改

3) 描述 不填

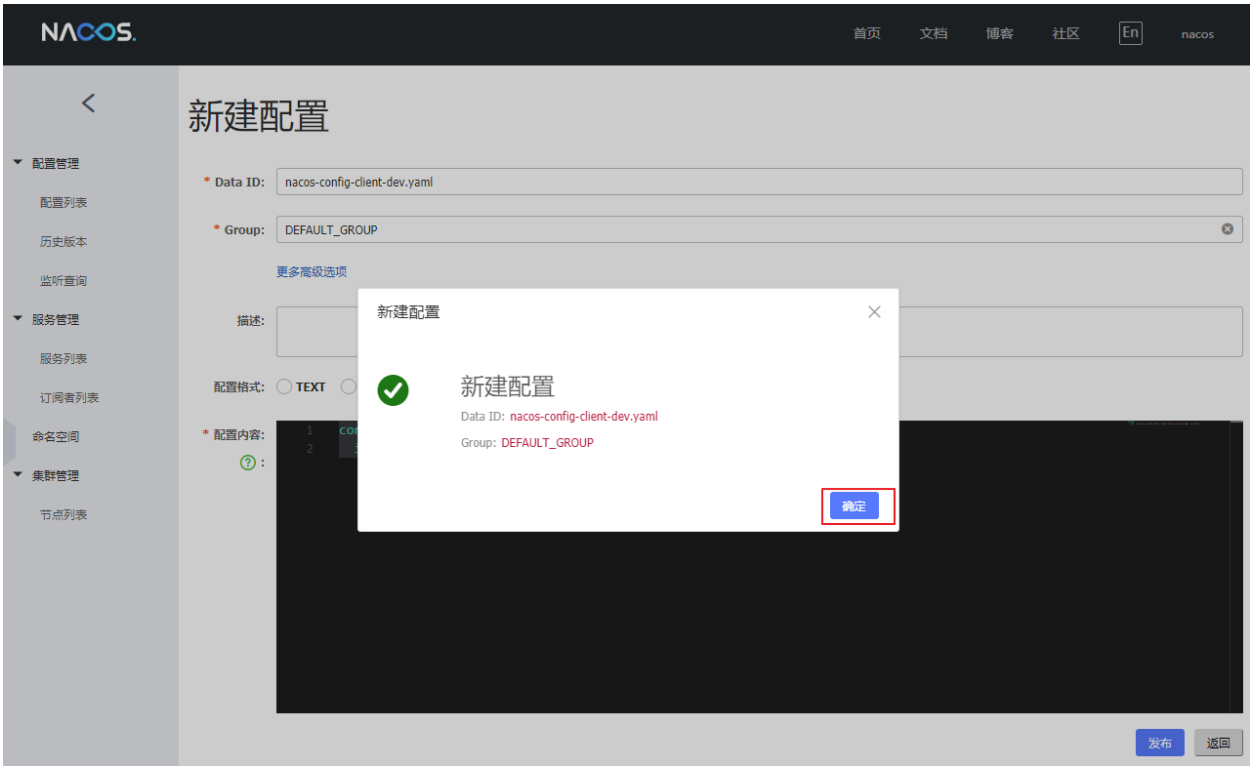
4) 配置格式 选择 `YAML`

5) 配置内容 填入:

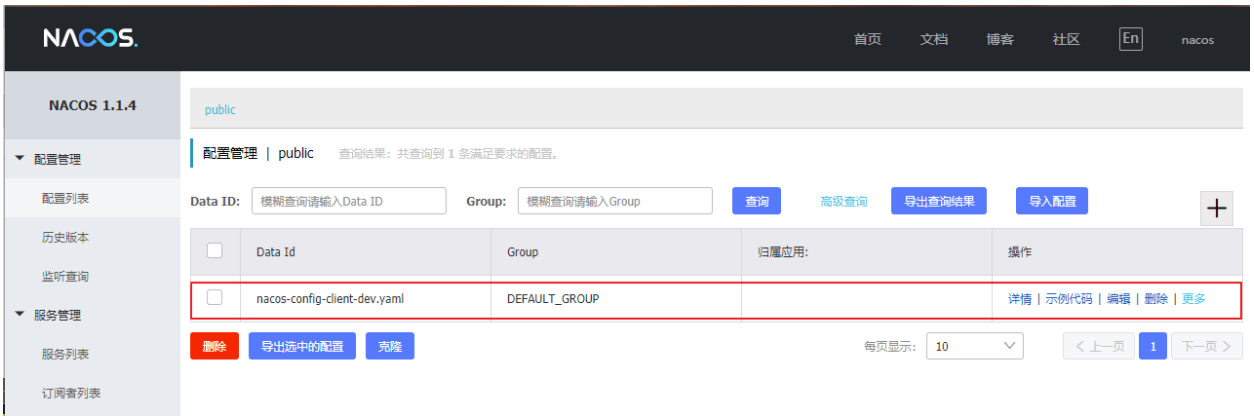
```yaml
config:
  info: from nacos config center, nacos-config-client-dev.yaml, version=1
```



6) 填写完毕后, 点击"发布"按钮



7) 配置管理 -> 配置列表 , 可以看到已经新增了一条配置

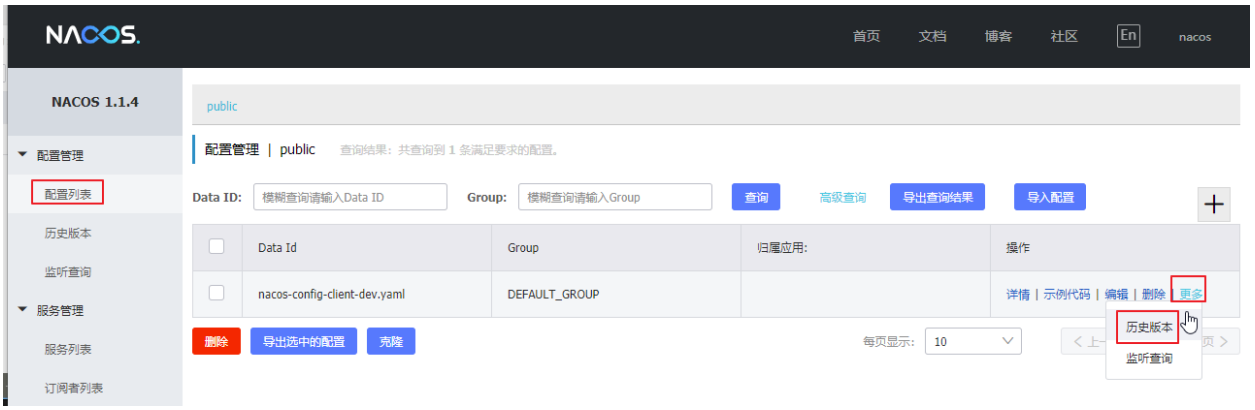


说明-设置DataId:

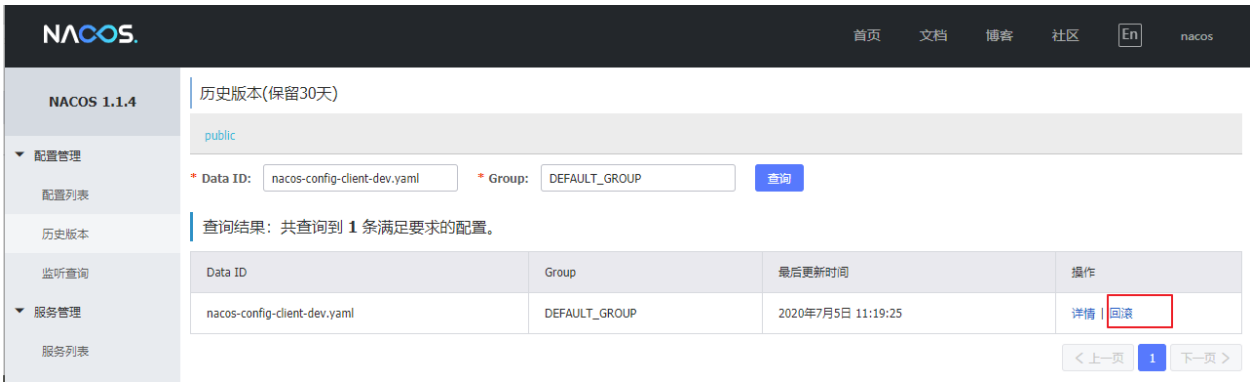
- 公式: `${spring.application.name}-${spring.profile.active}.${spring.cloud.nacos.config.file-extension}`
- `prefix` 默认为 `spring.application.name` 的值
- `spring.profiles.active` 既为当前环境对应的profile,可以通过配置项 `spring.profiles.active` 来配置
- `file-exetension` 为配置内容的数据格式，可以通过配置项 `spring.cloud.nacos.config.file-extension` 配置
- 小总结说明

4.1.6.2.3. 历史配置

Nacos会记录配置文件的历史版本, 默认保留30天, 此外还有一键回滚功能

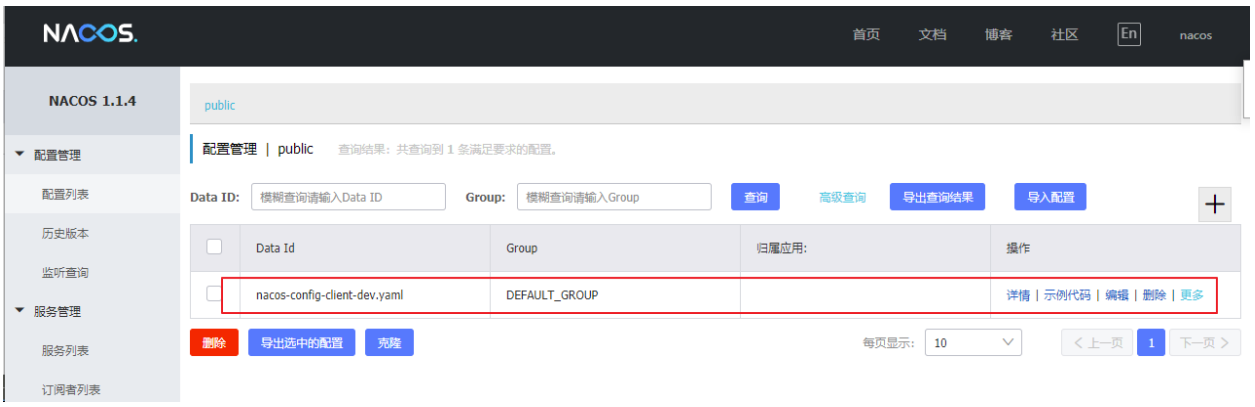


点击操作中的回滚, 可进行回滚操作

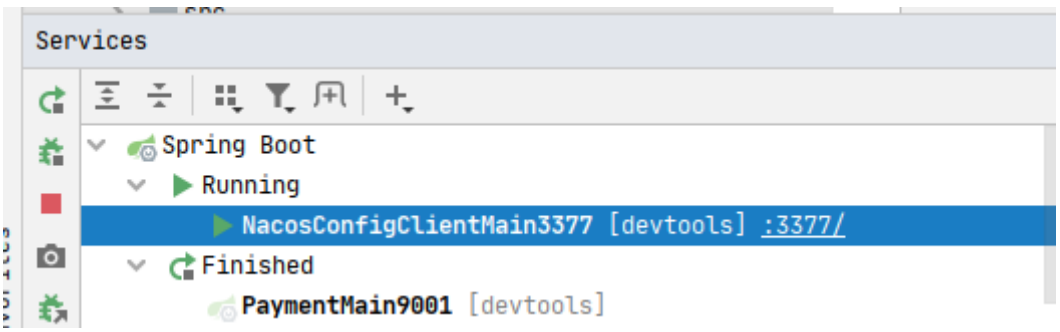


4.1.7. 测试

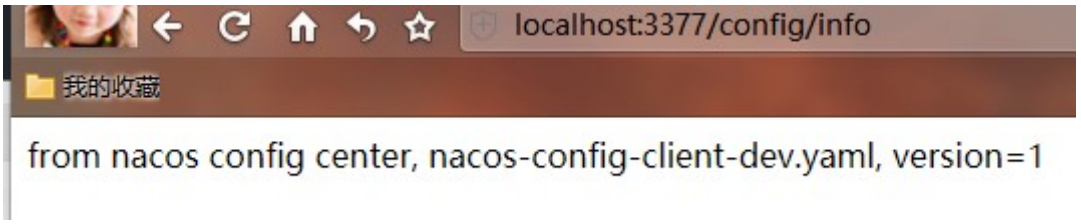
1) 启动前需要在nacos客户端-配置管理-配置管理栏目下有没有对应的yaml配置文件



2) 运行cloud-config-nacos-client3377的主启动类

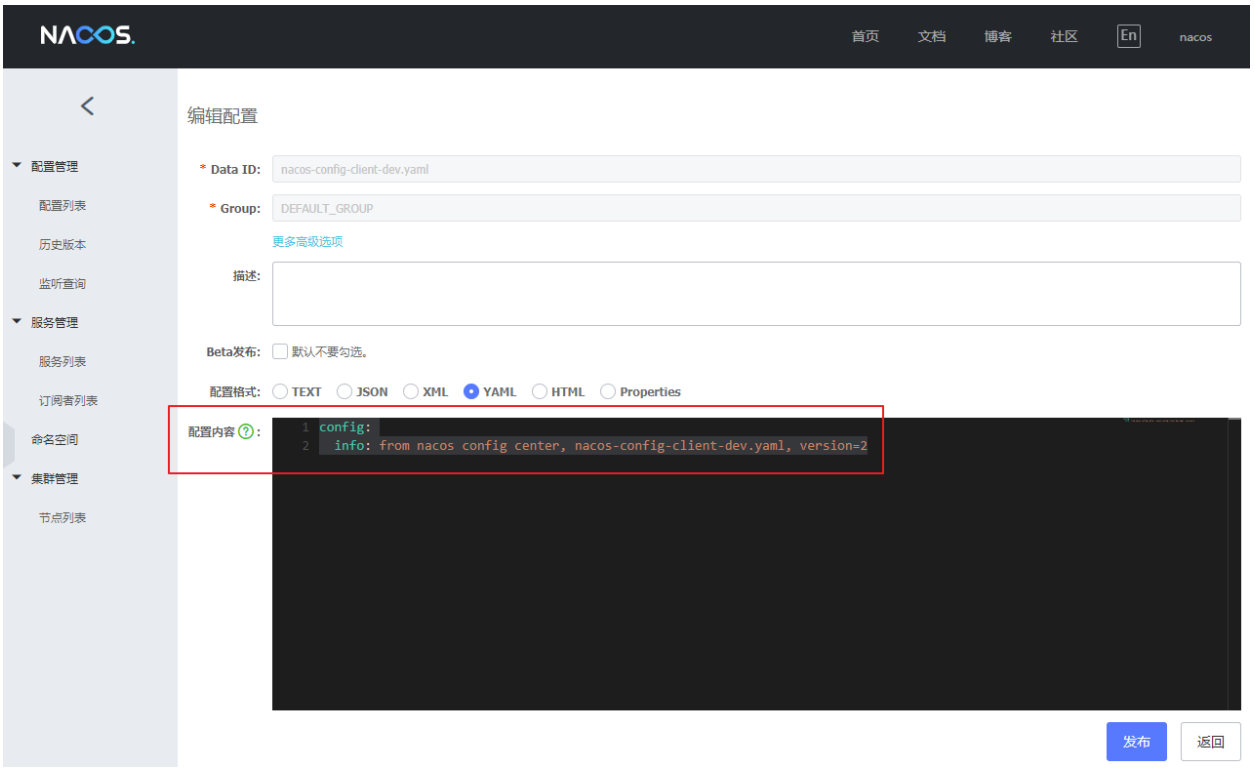


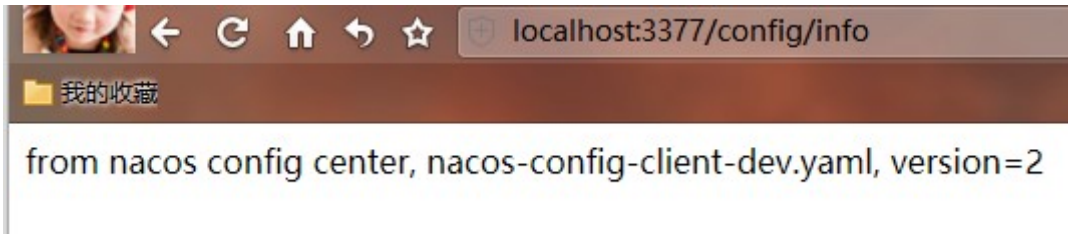
3) 调用接口查看配置信息http://localhost:3377/config/info



4) Nacos自带动态刷新: 修改下Nacos中的yaml配置文件，再次调用查看配置的接口，就会发现配置已经刷新

```yaml
config:
  info: from nacos config center, nacos-config-client-dev.yaml, version=2
```





4.2. 分类配置

4.2.1. 问题

问题1：

- 实际开发中，通常一个系统会准备
  - dev开发环境

- test测试环境
- prod生产环境

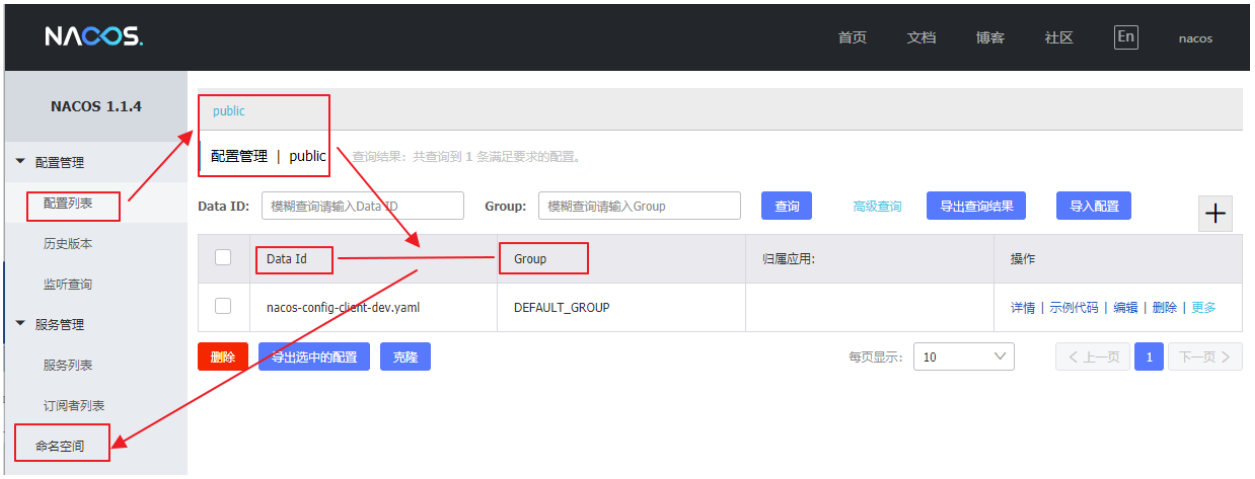
- 如何保证指定环境启动时服务能正确读取到Nacos上相应环境的配置文件呢?

问题2:

- 一个大型分布式微服务系统会有很多微服务子项目，每个微服务项目又都会有相应的开发环境、测试环境、预发环境、正式环境...
- 那怎么对这些微服务配置进行管理呢?

4.2.2. Nacos图形化管理界面

4.2.2.1. 配置管理



4.2.2.2. 命名空间

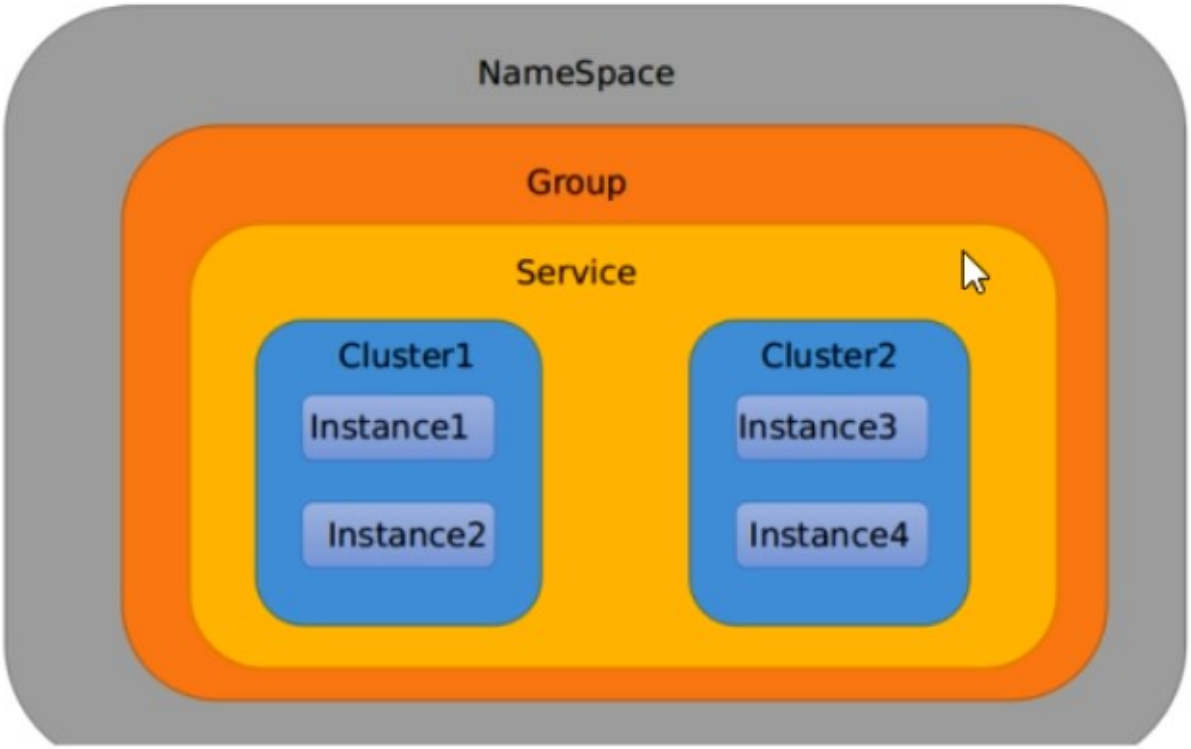
4.2.3. Namespace+Group+Data ID三者关系? 为什么这么设计?

4.2.3.1. 是什么

- 类似Java里面的 package名和类名

- 最外层的 namespace是可以用于区分部署环境的，Group和 DataID逻辑上区分两个目标对象。

4.2.3.2. 三者情况



4.2.3.3. 默认情况

- Namespace=public，Group=DEFAULT_GROUP默认 Cluster是 DEFAULT
- Nacos默认的命名空间是 public，Namespace主要用来实现隔离。
- 比方说我们现在有三个环境：开发、测试、生产环境，我们就可以创建三个 Namespace，不同的 Namespace之间是隔离的
- Group默认是 DEFAULT_GROUP，Group可以把不同的微服务划分到同个分组里面去
- Service就是微服务；一个 Service可以包含多个 Cluster（集群），Nacos默认 Cluster是 DEFAULT，Cluster是对指定微服务的一个虚拟划分。
- 比方说为了容灾，将 Service微服务分别部署在了杭州机房和广州机房，这时就可以给杭州机房的 Service微服务起一个集群名称（HZ），给广州机房的 Service微服务起个集群名称（GZ），还可以尽量让同一个机房的微服务互相调用，以提升性能。
- 最后是 Instance，就是微服务的实例

4.2.4. 案例

4.2.4.1. DataID方案

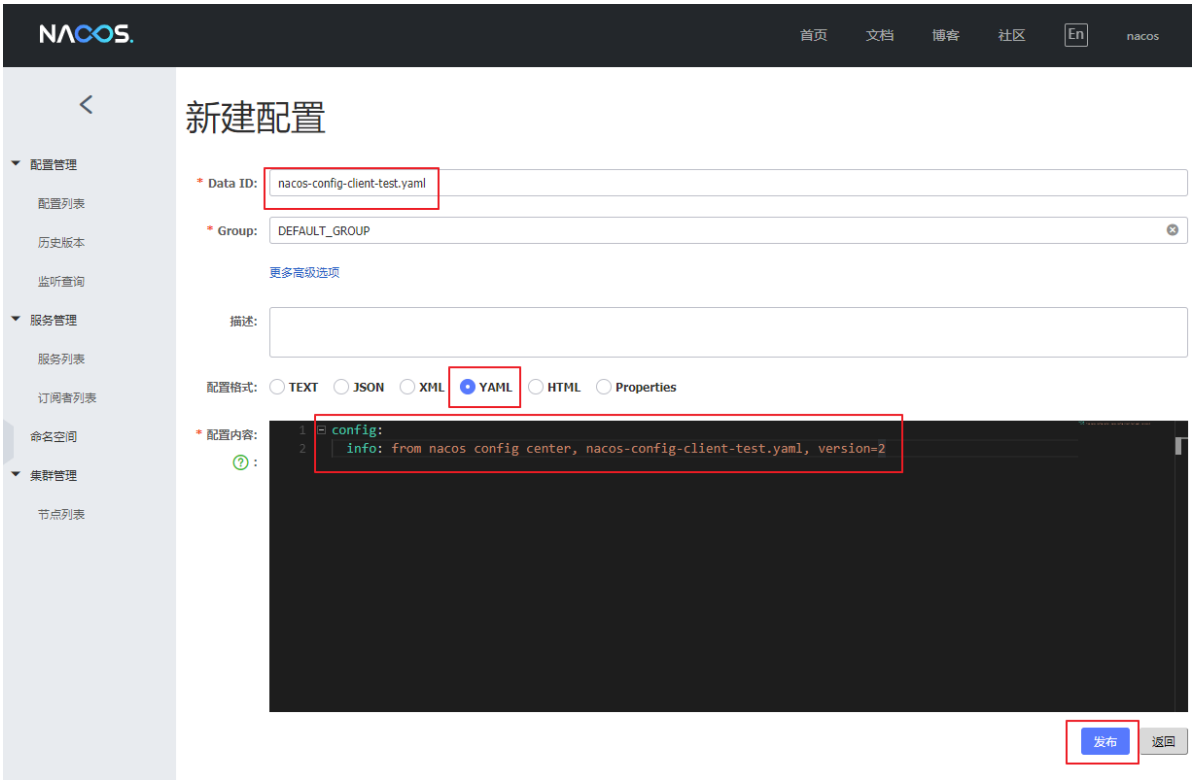
1) 指定 `spring.profiles.active` 和配置文件的DataID来使不同环境下读取不同的配置

2) 默认空间+默认分组+新建dev和test两个DataID

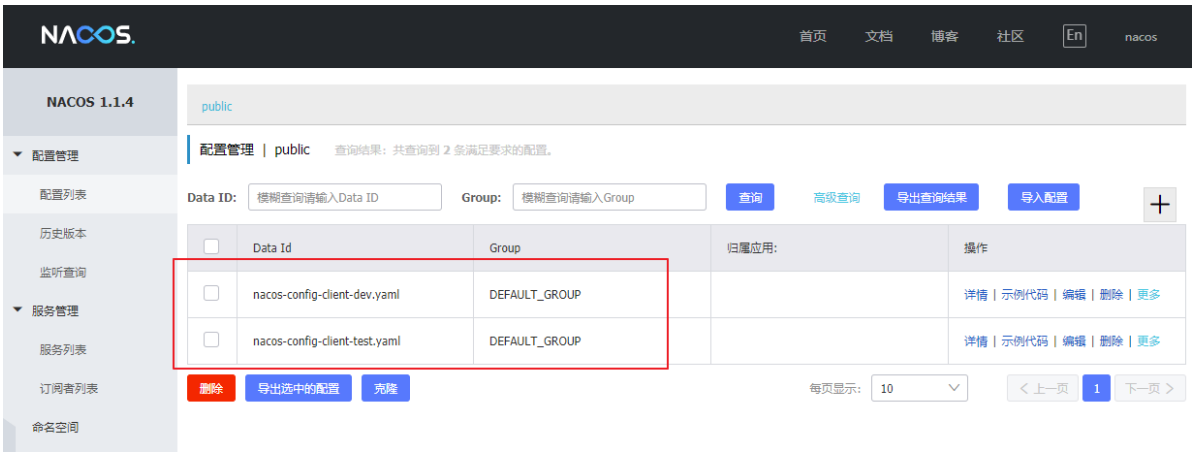
- 新建dev配置DataID
  - 之前已经创建过, 这里不再重新创建

- 新建test配置DataID
  - Data ID 配置为 `nacos-config-client-test.yaml`
  - Group 不进行配置
  - 配置格式选择为YAML
  - 配置内容为

```yaml
config:
    info: from nacos config center, nacos-config-client-dev.yaml, version=2
```



- 最终我们配置列表中有两条配置



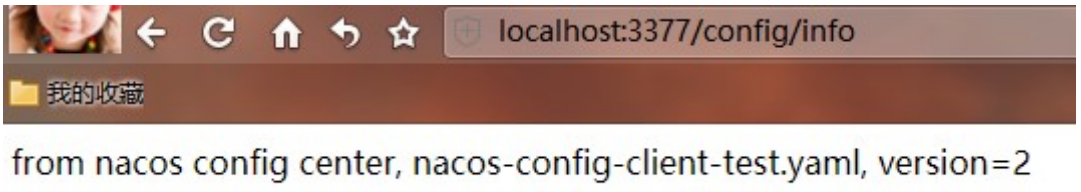
3) 通过 `spring.profiles.active` 属性就能进行多环境下配置文件的读取

- 修改cloudalibaba-config-nacos-client3377的application.yml

```yaml
spring:
  profiles:
    #    active: dev # 表示开发环境
    active: test # 表示测试环境
```

4) 测试

- 浏览器访问 http://localhost:3377/config/info



5) 测试结果: 配置是什么就加载什么

4.2.4.2. Group方案

1) 通过Group实现环境区分

- 新建DEV_GROUP
  - Data ID: nacos-config-client-info.yaml
  - Group: DEV_GROUP
  - 描述: DEV_GROUP
  - 配置格式: YAML
  - 配置内容:

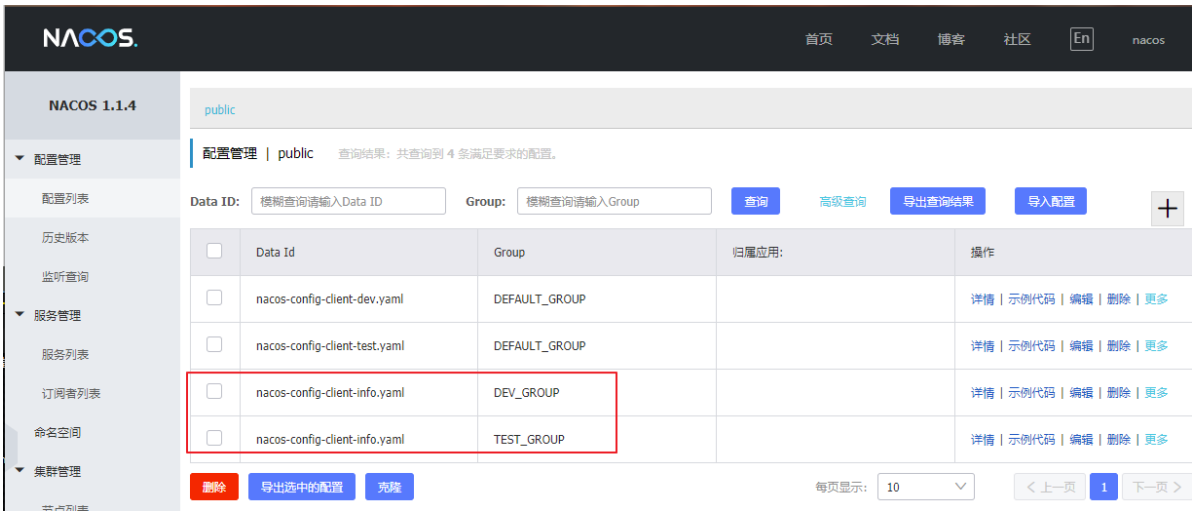
```yaml
config:
  info: nacos-config-client-info.yaml,DEV_GROUP,version=1
```

- 新建TEST_GROUP
  - Data ID: nacos-config-client-info.yaml
  - Group: TEST_GROUP
  - 描述: TEST_GROUP
  - 配置格式: YAML
  - 配置内容:

```yaml
config:
  info: nacos-config-client-info.yaml,TEST_GROUP,version=1
```

2) 在nacos图形界面控制台上新建配置文件DataID

- 新建配置完成后, 即可在配置列表中看到新增的两条配置



3) bootstrap.yml + application.yml

- 修改bootstrap.yml: 在config下增加一条group的配置即可。可配置为DEV_GROUP或TEST_GROUP
  - 修改内容:

```yaml
spring:
  cloud:
    nacos:
      config:
        group: TEST_GROUP
```

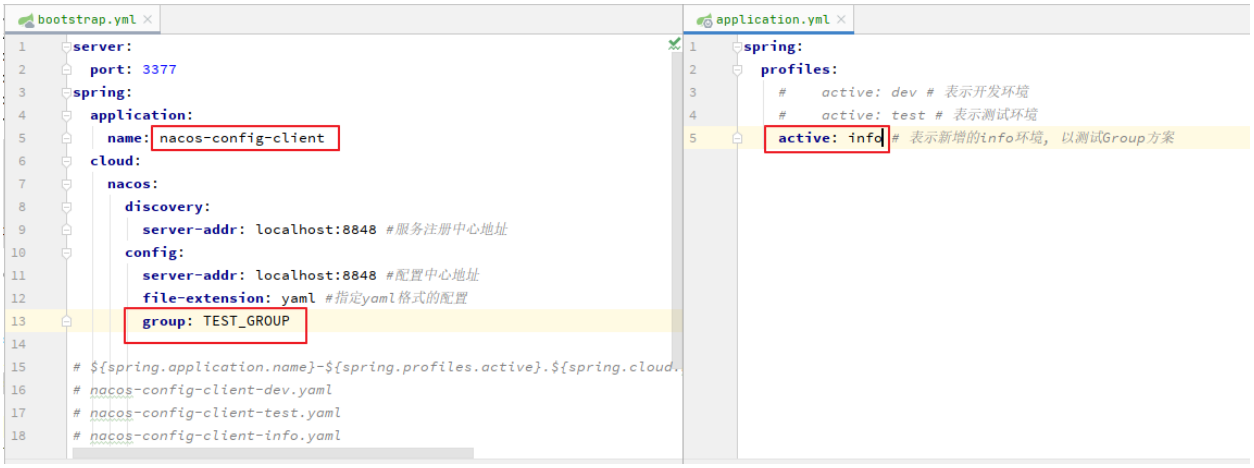
  - 完整内容:

```yaml
server:
  port: 3377
spring:
  application:
    name: nacos-config-client
  cloud:
    nacos:
      discovery:
        server-addr: localhost:8848 #服务注册中心地址
      config:
        server-addr: localhost:8848 #配置中心地址
        file-extension: yaml #指定yaml格式的配置
        group: TEST_GROUP

```

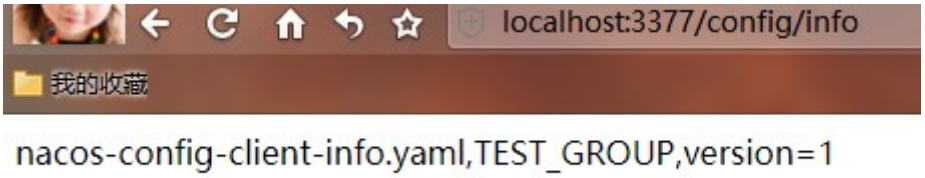
```
15
16 # ${spring.application.name}-${spring.profiles.active}.${spring.cloud.nacos
17
18
19 # nacos-config-client-dev.yaml
20
21
22 # nacos-config-client-test.yaml
23
24
25 # nacos-config-client-info.yaml
26
```

- 修改application.yml: 修改 `spring.profiles.active` 配置

```yaml
spring:
  profiles:
    #    active: dev # 表示开发环境
    #    active: test # 表示测试环境
    active: info # 表示新增的info环境, 以测试Group方案
```



4) 测试, 浏览器访问http://localhost:3377/config/info

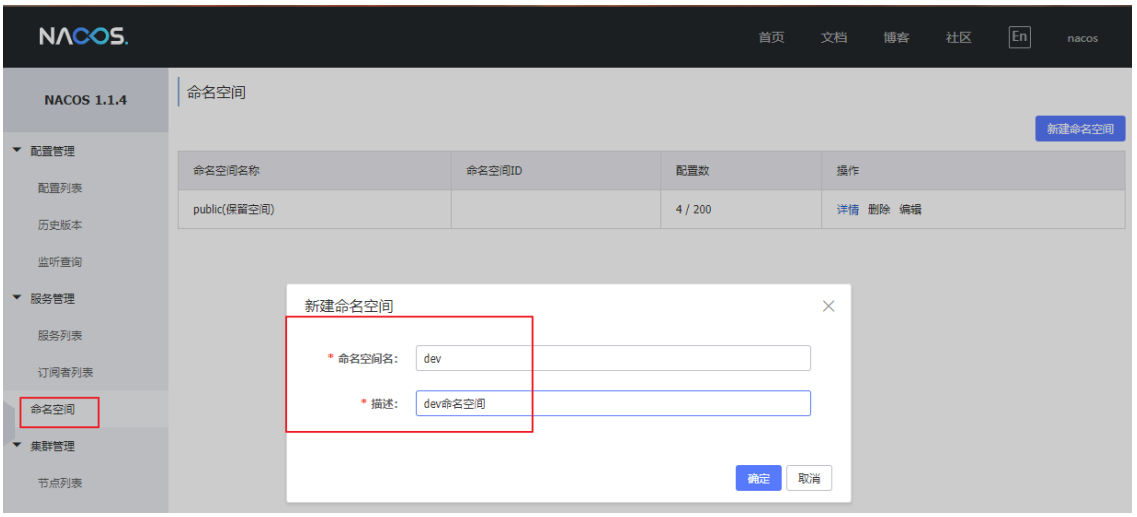


nacos-config-client-info.yaml,TEST_GROUP,version=1

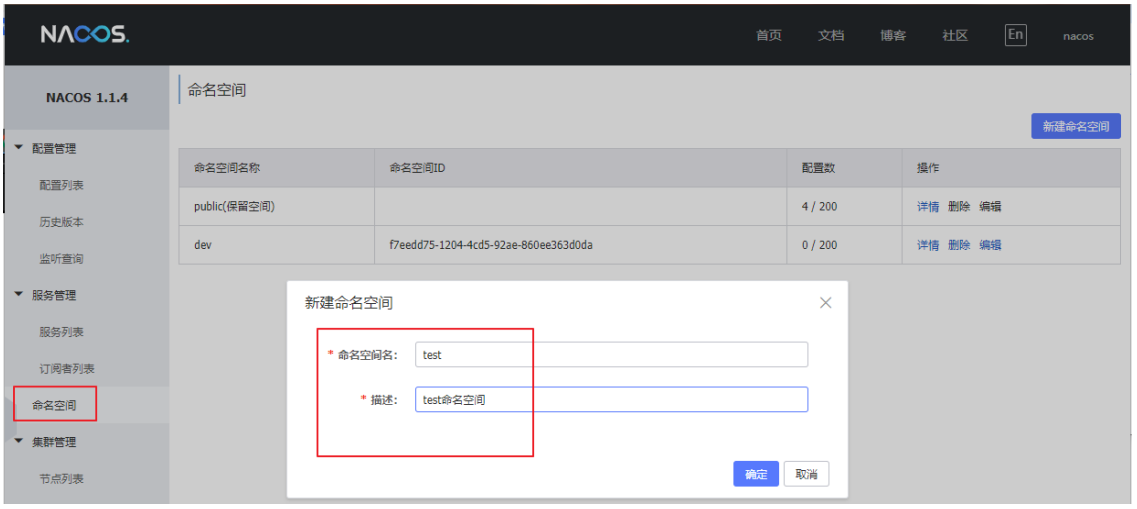
4.2.4.3. Namespace方案

1) 新建dev/test的Namespace

- 服务管理 -> 命名空间 -> 新建命名空间
- 新建dev命名空间
  - 命名空间名: dev
  - 描述: dev命名空间



- 新建test命名空间
  - 命名空间名: test
  - 描述: test命名空间



- 创建完毕后, 在命名空间中可以看到新增的2条命名空间记录

2) 回到服务管理->服务列表查看

- 可以看到, 除了public之外, 还多了两个命名空间, dev和test

3) 按照域名配置填写

- 配置管理 -> 配置列表 -> 切换命名空间至dev

- 新建配置, Group为DEFAULT_GROUP
  - Data ID: nacos-config-client-dev.yaml
  - Group: DEFAULT_GROUP
  - 描述: 空
  - 配置格式: YAML
  - 配置内容

```
config:
```

```
  info: f7eedd75-1204-4cd5-92ae-860ee363d0da,DEFAULT_GROUP,nacos-config-cli
```

- 新建配置, Group为DEV_GROUP
  - Data ID: nacos-config-client-dev.yaml
  - Group: DEV_GROUP
  - 描述: 空
  - 配置格式: YAML
  - 配置内容

```
config:
  info: f7eedd75-1204-4cd5-92ae-860ee363d0da,DEV_GROUP,nacos-config-client-
```

- 新建配置, Group为TEST_GROUP
  - Data ID: nacos-config-client-dev.yaml
  - Group: TEST_GROUP
  - 描述: 空
  - 配置格式: YAML
  - 配置内容

```
config:
  info: f7eedd75-1204-4cd5-92ae-860ee363d0da,TEST_GROUP,nacos-config-client
```

- 新建完毕, 可以在命名空间为dev的配置列表中, 查看到3条配置

4) YML

- bootstrap.yml
  - 修改内容:

```yaml
spring:
  cloud:
    nacos:
      config:
        namespace: f7eedd75-1204-4cd5-92ae-860ee363d0da
```

- 完整内容:

```yaml
server:
  port: 3377
spring:
  application:
    name: nacos-config-client
  cloud:
    nacos:
      discovery:
        server-addr: localhost:8848 #服务注册中心地址
      config:
        server-addr: localhost:8848 #配置中心地址
        file-extension: yaml #指定yaml格式的配置
        group: TEST_GROUP
        namespace: f7eedd75-1204-4cd5-92ae-860ee363d0da # 命名空间id


# ${spring.application.name}-${spring.profiles.active}.${spring.cloud.nacos


# nacos-config-client-dev.yaml


# nacos-config-client-test.yaml


# nacos-config-client-info.yaml

```

- application.yaml
  - 完整内容:

```yaml
spring:
  profiles:
```

```
    active: dev # 表示开发环境
    #    active: test # 表示测试环境

#    active: info # 表示新增的info环境，以测试Group方案
```

bootstrap.yml

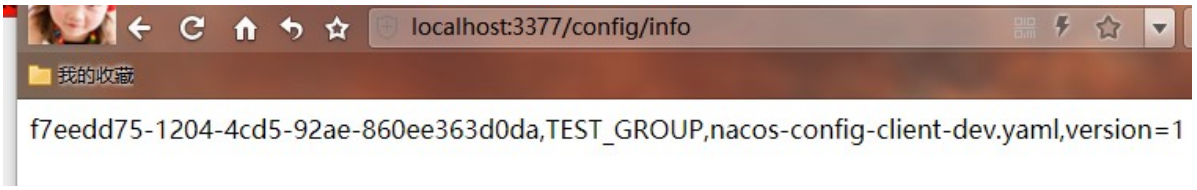
```
server:
  port: 3377
spring:
  application:
    name: nacos-config-client
  cloud:
    nacos:
      discovery:
        server-addr: localhost:8848 #服务注册中心地址
      config:
        server-addr: localhost:8848 #配置中心地址
        file-extension: yaml #指定yaml格式的配置
        group: TEST_GROUP
        namespace: f7eedd75-1204-4cd5-92ae-860ee363d0da # 命名空间id
```

application.yml

```
spring:
  profiles:
    active: dev # 表示开发环境
    #    active: test # 表示测试环境
#    active: info # 表示新增的info环境，以测试Group方案
```

5) 测试

- 浏览器访问http://localhost:3377/config/info



5. Nacos集群和持久化配置（重要）

5.1. 官网说明

5.1.1. 官网

官网: https://nacos.io/zh-cn/docs/cluster-mode-quick-start.html

5.1.2. 官网架构图

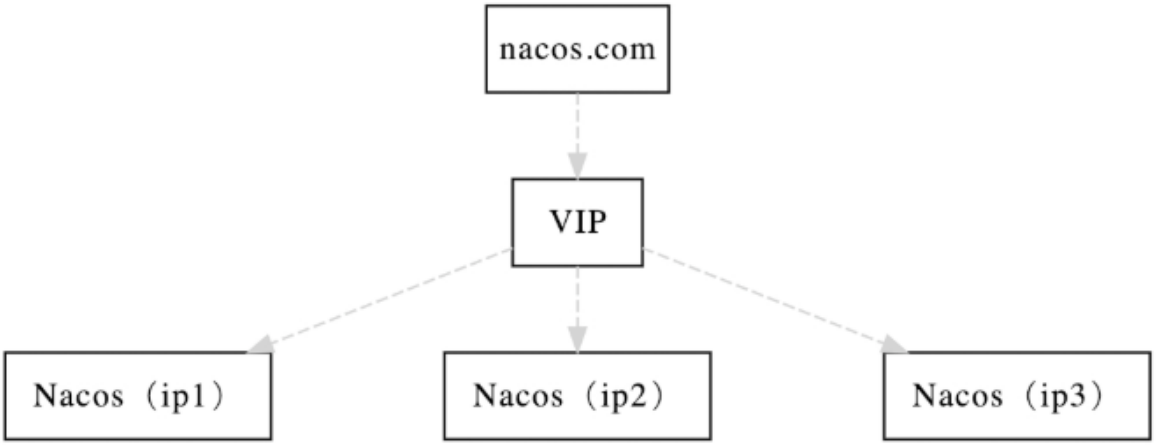
5.1.3. 集群部署架构图

推荐用户把所有服务列表放到一个vip下面，然后挂到一个域名下面

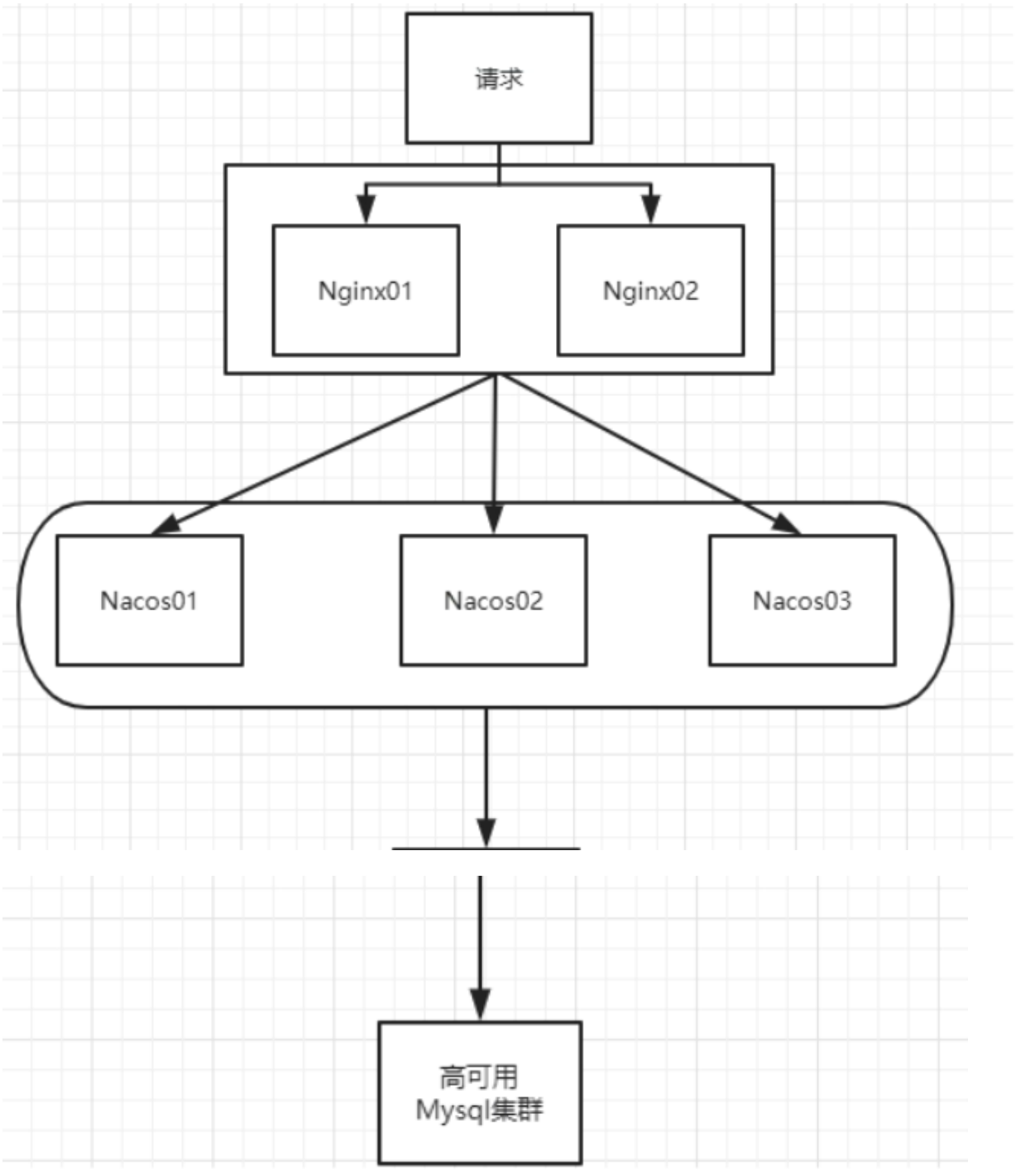
http://ip1:port/openAPI 直连ip模式，机器挂则需要修改ip才可以使用。

http://VIP:port/openAPI 挂载VIP模式，直连vip即可，下面挂server真实ip，可读性不好。

http://nacos.com:port/openAPI 域名 + VIP模式，可读性好，而且换ip方便，推荐模式


nacos.com
VIP
Nacos（ip1）
Nacos（ip2）
Nacos（ip3）

对上图的解释说明:


请求
Nginx01
Nginx02
Nacos01
Nacos02
Nacos03
高可用
Mysql集群

5.1.4. 说明

5.1.4.1. 官网说明

> https://nacos.io/zh-cn/docs/deployment.html

Nacos支持三种部署模式

- 单机模式 - 用于测试和单机试用。
- 集群模式 - 用于生产环境，确保高可用。
- 多集群模式 - 用于多数据中心场景。

单机模式下运行Nacos

- **Linux/Unix/Mac**
  - Standalone means it is non-cluster Mode. * sh startup.sh -m standalone
- **Windows**
  - cmd startup.cmd 或者双击 startup.cmd 文件

单机模式支持mysql

在0.7版本之前，在单机模式时nacos使用嵌入式数据库实现数据的存储，不方便观察数据存储的基本情况。0.7版本增加了支持mysql数据源能力，具体的操作步骤：

- 1.安装数据库，版本要求：5.6.5+
- 2.初始化mysql数据库，数据库初始化文件：nacos-mysql.sql
- 3.修改conf/application.properties文件，增加支持mysql数据源配置（目前只支持mysql），添加mysql数据源的url、用户名和密码。

```
spring.datasource.platform=mysql

db.num=1
db.url.0=jdbc:mysql://11.162.196.16:3306/nacos_devtest?characterEncoding=utf8&connec
db.user=nacos_devtest
db.password=youdontknow

```

再以单机模式启动nacos，nacos所有写嵌入式数据库的数据都写到了mysql

5.1.4.2. 对官网说明的补充

默认 Nacos使用嵌入式数据库实现数据的存储。所以，如果启动多个默认配置下的 Nacos节点，数据存储是存在一致性问题的。

为了解决这个问题，Nacos采用了集中式存储的方式来支持集群化部署，目前只支持 MySQL的存储。

按照上述，我们需要mysql数据库

> ## Nacos支持三种部署模式
>
> - 单机模式 - 用于测试和单机试用。
> - 集群模式 - 用于生产环境，确保高可用。
> - 多集群模式 - 用于多数据中心场景。
>
> ### Windows
>
> cmd startup.cmd 或者双击 startup.cmd 文件
>
> ### 单机模式支持mysql
>
> 在0.7版本之前，在单机模式时nacos使用嵌入式数据库实现数据的存储，不方便观察数据存储的基本情况。
> 0.7版本增加了支持mysql数据源能力，具体的操作步骤：
>
> - 1.安装数据库，版本要求：5.6.5+
> - 2.初始化mysql数据库，数据库初始化文件：nacos-mysql.sql
> - 3.修改conf/application.properties文件，增加支持mysql数据源配置（目前只支持mysql），添加mysql数据源的url、用户名和密码。
>
> ```
> spring.datasource.platform=mysql
>
> db.num=1
> db.url.0=jdbc:mysql://11.162.196.16:3306/nacos_devtest?characterEncoding=utf8&connectTimeout=1000
> db.user=nacos_devtest
> db.password=youdontknow
> ```
>
> 再以单机模式启动nacos，nacos所有写嵌入式数据库的数据都写到了mysql

5.2. Nacos持久化配置解释

5.2.1. Nacos默认数据库

Nacos默认自带的是嵌入式数据库derby

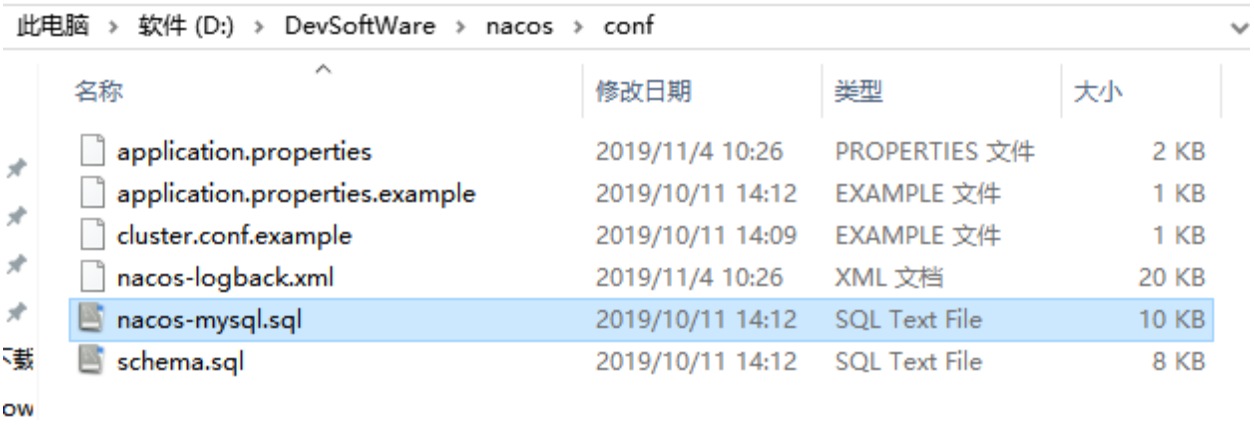
> https://github.com/alibaba/nacos/blob/develop/config/pom.xml

可以看到, nacos引入了derby的依赖

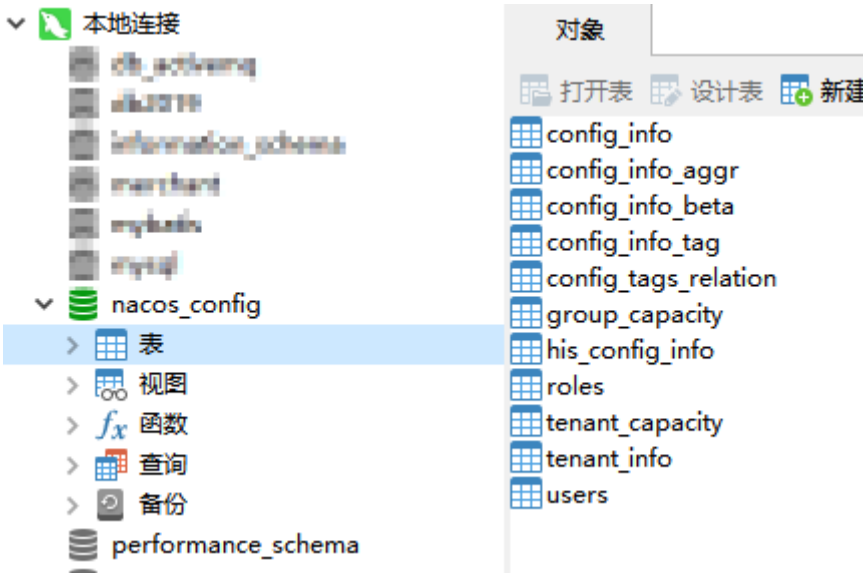
```
<dependency>
    <groupId>org.apache.derby</groupId>
    <artifactId>derby</artifactId>
</dependency>
```

5.2.2. derby到mysql切换配置步骤

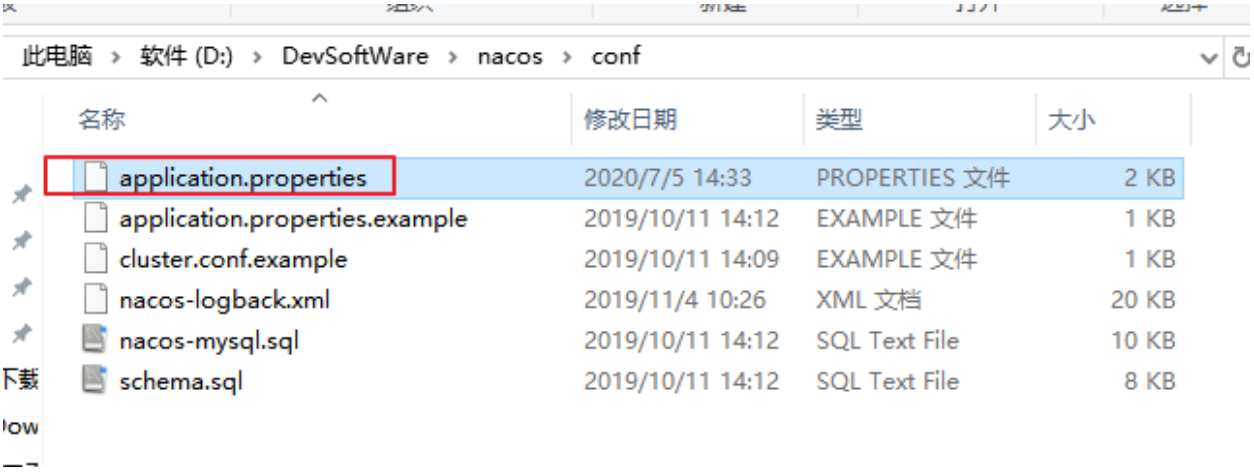
1) nacos-server-1.1.4\nacos\conf目录下找到名称为nacos-mysql.sql的sql脚本



2) 执行该脚本



3) nacos-server-1.1.4\nacos\conf目录下找到application.properties



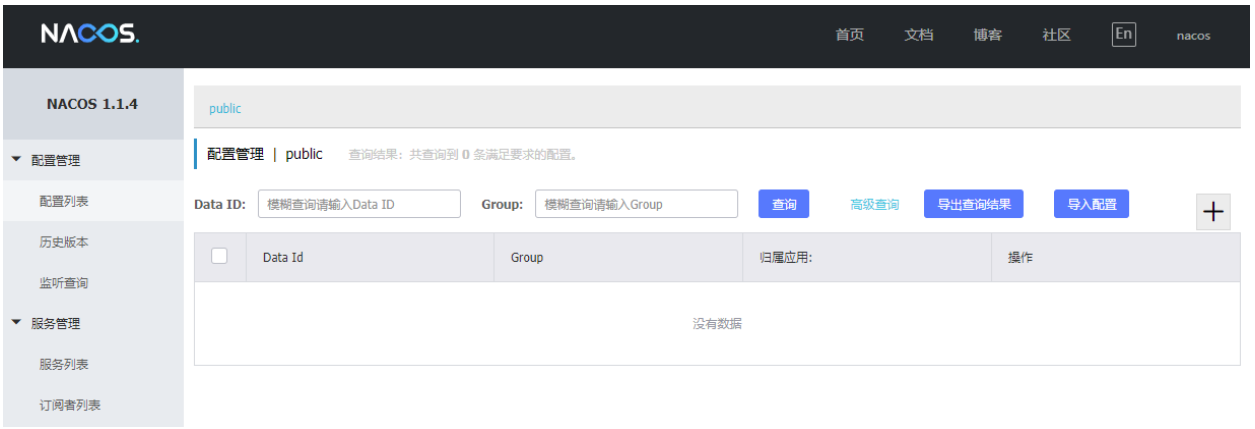
4) application.properties中添加如下配置

```
# 从derby切换至mysql
spring.datasource.platform=mysql

```

```
db.num=1
db.url.0=jdbc:mysql://localhost:3306/nacos_config?characterEncoding=utf8&connectTime
db.user=root
db.password=root
```

5.2.3. 测试

启动nacos，可以看到是个全新的空记录界面，以前是记录进derby



5.3. Linux版Nacos+MySQL生产环境配置

5.3.1. 环境要求

预计需要，1个nginx+3个nacos注册中心+1个mysql

5.3.2. Nacos下载linux版本

下载地址: https://github.com/alibaba/nacos/releases/tag/1.1.4

选择nacos-server-1.1.4.tar.gz进行下载

解压后安装

```
[root@sitedev soft]# ll
总用量 347984
-rw-r--r--. 1 root root  62666908 4月   9 22:16 apache-activemq-5.15.12-bin.tar.gz
-rw-r--r--. 1 root root 190890122 11月  9 2018 jdk-8u171-linux-x64.tar.gz
-rw-r--r--. 1 root root  52115157 7月   5 15:38 nacos-server-1.1.4.tar.gz
drwxr-xr-x. 9 1001 1001       186 2月  12 12:47 nginx-1.16.1
-rw-r--r--. 1 root root   1032630 2月  12 12:22 nginx-1.16.1.tar.gz
drwxrwxr-x. 6 root root      4096 5月  16 2019 redis-5.0.5
-rw-r--r--. 1 root root   1975750 5月  16 2019 redis-5.0.5.tar.gz
-rw-r--r--. 1 root root  12589252 3月   5 15:57 rocketmq-all-4.5.2-bin-release.zip
-rw-r--r--. 1 root root  35042811 8月  29 2019 zookeeper-3.4.10.tar.gz
[root@sitedev soft]# tar -zxvf nacos-server-1.1.4.tar.gz -C /opt/module
nacos/LICENSE
nacos/NOTICE
nacos/target/nacos-server.jar
nacos/conf/
nacos/conf/application.properties
nacos/conf/application.properties.example
nacos/conf/cluster.conf.example
nacos/conf/nacos-logback.xml
nacos/conf/nacos-mysql.sql
nacos/conf/schema.sql
nacos/bin/shutdown.cmd
nacos/bin/shutdown.sh
nacos/bin/startup.cmd
nacos/bin/startup.sh
[root@sitedev soft]#
```

5.3.2. 集群配置步骤

5.3.2.1. Linux服务器上mysql数据库配置

SQL脚本路径: /opt/module/nacos/conf/nacos-mysql.sql

```
[root@sitedev module]# cd nacos/conf
[root@sitedev conf]# pwd
/opt/module/nacos/conf
[root@sitedev conf]# ll
总用量 52
-rw-r--r--. 1 root root  1564 10月 24 2019 application.properties
-rw-r--r--. 1 root root   408 10月 24 2019 application.properties.example
-rw-r--r--. 1 root root    58 10月 24 2019 cluster.conf.example
-rw-r--r--. 1 root root 20210 10月 24 2019 nacos-logback.xml
-rw-r--r--. 1 root root  9788 10月 24 2019 nacos-mysql.sql
-rw-r--r--. 1 root root  7196 10月 24 2019 schema.sql
[root@sitedev conf]#
```

执行该SQL脚本后, 查看结果:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| nacos_config       |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> use nacos_config
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+------------------------+
| Tables_in_nacos_config |
+------------------------+
| config_info            |
| config_info_aggr       |
| config_info_beta       |
| config_info_tag        |
| config_tags_relation   |
| group_capacity         |
| his_config_info        |
| roles                  |
| tenant_capacity        |
| tenant_info            |
| users                  |
+------------------------+
11 rows in set (0.01 sec)

mysql>
```

5.3.2.2. nacos的application.yml配置

application.yml配置文件路径: /opt/module/nacos/conf/application.yml

```
[root@sitedev module]# cd nacos/conf
[root@sitedev conf]# pwd
/opt/module/nacos/conf
[root@sitedev conf]# ll
总用量 52
-rw-r--r--. 1 root root  1564 10月  24 2019 application.properties
-rw-r--r--. 1 root root   408 10月  24 2019 application.properties.example
-rw-r--r--. 1 root root    58 10月  24 2019 cluster.conf.example
-rw-r--r--. 1 root root 20210 10月  24 2019 nacos-logback.xml
-rw-r--r--. 1 root root  9788 10月  24 2019 nacos-mysql.sql
-rw-r--r--. 1 root root  7196 10月  24 2019 schema.sql
[root@sitedev conf]#
```

在该文件中添加如下配置

```
# 从derby切换至mysql
spring.datasource.platform=mysql

db.num=1
```

```
db.url.0=jdbc:mysql://localhost:3306/nacos_config?characterEncoding=utf8&connectTime
db.user=root
db.password=root
```

```
# 从derby切换至mysql
spring.datasource.platform=mysql

db.num=1
db.url.0=jdbc:mysql://localhost:3306/nacos_config?characterEncoding=utf8&connectTimeout=1000&socketTimeout=3000&autoReconnect=true
db.user=root
db.password=root
~
```

5.3.2.3. Linux服务器上nacos的集群配置cluster.conf

1) 梳理出3台nacos机器的不同服务端口号

2) 复制出cluster.conf

```
[root@sitedev conf]# cp cluster.conf.example  cluster.conf
```

```
[root@sitedev conf]# ll
总用量 52
-rw-r--r--. 1 root root  1795 7月    6 00:24 application.properties
-rw-r--r--. 1 root root   408 10月 24 2019 application.properties.example
-rw-r--r--. 1 root root    58 10月 24 2019 cluster.conf.example
-rw-r--r--. 1 root root 20210 10月 24 2019 nacos-logback.xml
-rw-r--r--. 1 root root  9788 10月 24 2019 nacos-mysql.sql
-rw-r--r--. 1 root root  7196 10月 24 2019 schema.sql
[root@sitedev conf]# cp cluster.conf.example  cluster.conf
[root@sitedev conf]# ll
总用量 56
-rw-r--r--. 1 root root  1795 7月    6 00:24 application.properties
-rw-r--r--. 1 root root   408 10月 24 2019 application.properties.example
-rw-r--r--. 1 root root    58 7月    6 00:28 cluster.conf
-rw-r--r--. 1 root root    58 10月 24 2019 cluster.conf.example
-rw-r--r--. 1 root root 20210 10月 24 2019 nacos-logback.xml
-rw-r--r--. 1 root root  9788 10月 24 2019 nacos-mysql.sql
-rw-r--r--. 1 root root  7196 10月 24 2019 schema.sql
[root@sitedev conf]#
```

3) 修改该配置文件内容:

> 这个IP不能写127.0.0.1,必须是Linux命令 `hostname -i` 能够识别的IP

```
192.168.5.10:3333
192.168.5.10:4444
192.168.5.10:5555
```

```
[root@sitedev conf]# hostname -i
fe80::5dfc:e0e8:62bb:dd28%ens32 192.168.5.10 192.168.122.1
[root@sitedev conf]# vim cluster.conf

#it is ip
#example
# 10.10.109.214
# 11.16.128.34
# 11.16.128.36
192.168.5.10:3333
192.168.5.10:4444
192.168.5.10:5555
```

5.3.2.4. 编辑Nacos的启动脚本startup.sh，使它能够接受不同的启动端

/mynacos/nacos/bin目录下有startup.sh

5.3.2.4.1. 思考

nacos/bin目录下有 startup.sh平时单机版的启动，都是/startup.sh即可。

但是集群启动，我们希望可以类似其它软件的shell命令，传递不同的端口号启动不同的 nacos实例。

命令： `/startup.sh -p 3333` 表示启动端口号为3333的 nacos服务器实例，和上一步的cluster.conf配置的一致。

5.3.2.4.2. 修改内容

1) 备份startup.sh

```
[root@sitedev bin]# cp startup.sh startup.sh.bk
```

2) 修改startup.sh

```
# 省略...
while getopts ":m:f:s:p:" opt
do
    case $opt in
        m)
            MODE=$OPTARG;;
        f)
            FUNCTION_MODE=$OPTARG;;
        s)
            SERVER=$OPTARG;;
        p)
            PORT=$OPTARG;;
        ?)
```

```
        echo "Unknown parameter"
        exit 1;;
    esac
done

# 省略...
nohup $JAVA -Dserver.port=${PORT} ${JAVA_OPT} nacos.nacos >> ${BASE_DIR}/logs/start.
```

```
[root@sitedev bin]# ll
总用量 20
-rwxr-xr-x. 1 root root  954 10月 24 2019 shutdown.cmd
-rwxr-xr-x. 1 root root  949 10月 24 2019 shutdown.sh
-rwxr-xr-x. 1 root root 2854 10月 24 2019 startup.cmd
-rwxr-xr-x. 1 root root 4894 10月 24 2019 startup.sh
[root@sitedev bin]# cp startup.sh startup.sh.bk
[root@sitedev bin]# vim startup.sh
[root@sitedev bin]#
```

3) 修改前后对比

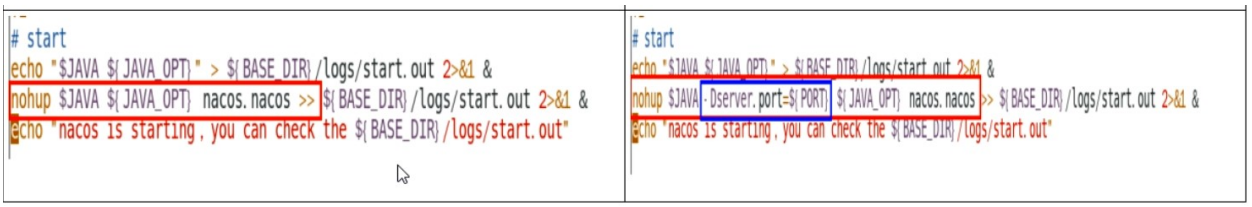
```
[root@zzyy bin]# pwd
/mynacos/nacos/bin
[root@zzyy bin]# ll
总用量 40
-rw-r--r--. 1 root root  705 1月   7 12:58 derby.log
drwxr-xr-x. 2 root root 4096 1月   8 17:11 logs
-rwxr-xr-x. 1 root root  954 1月   7 12:53 shutdown.cmd
-rwxr-xr-x. 1 root root  949 1月   7 12:53 shutdown.sh
-rwxr-xr-x. 1 root root 2854 1月   7 12:53 startup.cmd
-rwxr-xr-x. 1 root root 4856 1月   7 14:52 startup.sh
-rwxr-xr-x. 1 root root 4801 1月   7 13:28 startup.sh.bk
drwxr-xr-x. 3 root root 4096 1月   7 12:55 work
[root@zzyy bin]#
```

修改前

```
while getopts ":m:f:s:" opt
do
    case $opt in
        m)
            MODE=$OPTARG;;
        f)
            FUNCTION_MODE=$OPTARG;;
        s)
            SERVER=$OPTARG;;
        ?)
        echo "Unknown parameter"
        exit 1;;
    esac
done
```

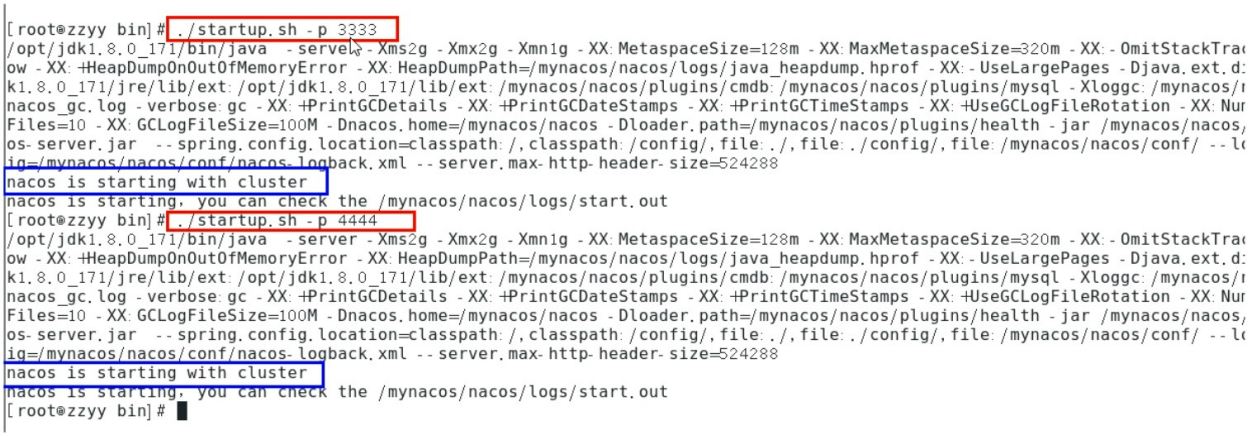
修改后

```
while getopts ":m:f:s:p:" opt
do
    case $opt in
        m)
            MODE=$OPTARG;;
        f)
            FUNCTION_MODE=$OPTARG;;
        s)
            SERVER=$OPTARG;;
        p)
            PORT=$OPTARG;;
        ?)
        echo "Unknown parameter"
        exit 1;;
    esac
done
```



5.3.2.4.3. 执行方式

```
[root@sitedev bin]# ./startup.sh -p 3333
```



5.3.2.5. Nginx的配置，由它作为负载均衡器

1) nginx配置文件路径为: /opt/module/nginx/conf/nginx.conf

2) 备份nginx配置文件

```
[root@sitedev conf]# cp nginx.conf nginx.conf.bk
[root@sitedev conf]# vim nginx.conf
```

```
[root@sitedev conf]# pwd
/opt/module/nginx/conf
[root@sitedev conf]# ll
总用量 68
drwxr-xr-x. 2 root root   63 2月  12 21:16 cert
-rw-r--r--. 1 root root 1077 2月  12 13:02 fastcgi.conf
-rw-r--r--. 1 root root 1077 2月  12 13:02 fastcgi.conf.default
-rw-r--r--. 1 root root 1007 2月  12 13:02 fastcgi_params
-rw-r--r--. 1 root root 1007 2月  12 13:02 fastcgi_params.default
-rw-r--r--. 1 root root 2837 2月  12 13:02 koi-utf
-rw-r--r--. 1 root root 2223 2月  12 13:02 koi-win
-rw-r--r--. 1 root root 5231 2月  12 13:02 mime.types
-rw-r--r--. 1 root root 5231 2月  12 13:02 mime.types.default
-rw-r--r--. 1 root root 2957 2月  12 21:24 nginx.conf
-rw-r--r--. 1 root root 2656 2月  12 13:02 nginx.conf.default
-rw-r--r--. 1 root root  636 2月  12 13:02 scgi_params
-rw-r--r--. 1 root root  636 2月  12 13:02 scgi_params.default
-rw-r--r--. 1 root root  664 2月  12 13:02 uwsgi_params
-rw-r--r--. 1 root root  664 2月  12 13:02 uwsgi_params.default
-rw-r--r--. 1 root root 3610 2月  12 13:02 win-utf
[root@sitedev conf]# cp nginx.conf nginx.conf.bk
[root@sitedev conf]# vim nginx.conf
```

3) 修改nginx的配置文件

```
    upstream cluster {

        server 127.0.0.1:3333;
        server 127.0.0.1:4444;
        server 127.0.0.1:5555;
    }

    server {
        listen       1111;
        server_name  localhost;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

        location / {
            # root   html;
            # index  index.html index.htm;
            proxy_pass http://cluster;
        }
```

```
upstream cluster {

    server 127.0.0.1:3333;
    server 127.0.0.1:4444;
    server 127.0.0.1:5555;
}

server {
    listen       1111;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        # root   html;
        # index  index.html index.htm;
        proxy_pass http://cluster;
    }
```

4) nginx按照指定配置文件启动

```
[root@sitedev nginx]# sbin/nginx -c /opt/module/nginx/conf/nginx.conf
```

```
[root@sitedev conf]# cd ..
[root@sitedev nginx]# sbin/nginx -c /opt/module/nginx/conf/nginx.conf
nginx: [warn] the "ssl" directive is deprecated, use the "listen ... ssl" directive instead in /opt/module/nginx/conf/nginx.conf:36
[root@sitedev nginx]# ps -aux | grep nginx
root      28297  0.0  0.0  47224  1208 ?        Ss   00:59   0:00 nginx: master process sbin/nginx -c /opt/module/nginx/conf/nginx.conf
nobody    28298  0.0  0.1  49716  2060 ?        S    00:59   0:00 nginx: worker process
root      28324  0.0  0.0 112728   972 pts/1    S+   00:59   0:00 grep --color=auto nginx
[root@sitedev nginx]#
```

5.3.2.6. 测试集群是否搭建成功

1) 截止到此处，1个Nginx+3个nacos注册中心+1个mysql

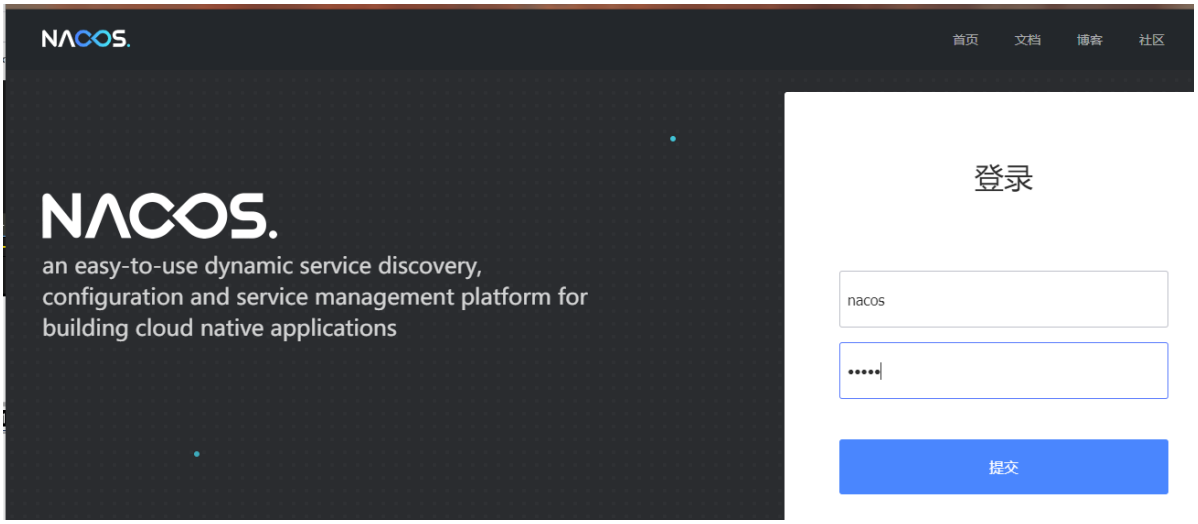
2) 启动nacos3333/nacos4444/nacos5555

```
[root@sitedev nacos]# bin/startup.sh -p 3333

[root@sitedev nacos]# bin/startup.sh -p 4444

[root@sitedev nacos]# bin/startup.sh -p 5555
```

```
[root@sitedev nacos]# pwd
/opt/module/nacos
[root@sitedev nacos]# bin/startup.sh -p 3333
/opt/module/jdk1.8.0_171/bin/java  -server -Xms2g -Xmx2g -Xmn1g -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m -XX:-OmitStackTraceInFastThrow -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/opt/module/nacos/logs/java_heapdump.hprof -XX:-UseLargePages -Djava.ext.dirs=/opt/module/jdk1.8.0_171/jre/lib/ext:/opt/module/jdk1.8.0_171/lib/ext:/opt/module/nacos/plugins/cmdb:/opt/module/nacos/plugins/mysql -Xloggc:/opt/module/nacos/logs/nacos_gc.log -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=100M -Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8000 -Dnacos.home=/opt/module/nacos -Dloader.path=/opt/module/nacos/plugins/health -jar /opt/module/nacos/target/nacos-server.jar  --spring.config.location=classpath:/,classpath:/config/,file:./,file:./config/,file:/opt/module/nacos/conf/ --logging.config=/opt/module/nacos/conf/nacos-logback.xml --server.max-http-header-size=524288
nacos is starting with cluster
nacos is starting, you can check the /opt/module/nacos/logs/start.out
[root@sitedev nacos]# bin/startup.sh -p 4444
/opt/module/jdk1.8.0_171/bin/java  -server -Xms2g -Xmx2g -Xmn1g -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m -XX:-OmitStackTraceInFastThrow -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/opt/module/nacos/logs/java_heapdump.hprof -XX:-UseLargePages -Djava.ext.dirs=/opt/module/jdk1.8.0_171/jre/lib/ext:/opt/module/jdk1.8.0_171/lib/ext:/opt/module/nacos/plugins/cmdb:/opt/module/nacos/plugins/mysql -Xloggc:/opt/module/nacos/logs/nacos_gc.log -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=100M -Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8000 -Dnacos.home=/opt/module/nacos -Dloader.path=/opt/module/nacos/plugins/health -jar /opt/module/nacos/target/nacos-server.jar  --spring.config.location=classpath:/,classpath:/config/,file:./,file:./config/,file:/opt/module/nacos/conf/ --logging.config=/opt/module/nacos/conf/nacos-logback.xml --server.max-http-header-size=524288
nacos is starting with cluster
nacos is starting, you can check the /opt/module/nacos/logs/start.out
[root@sitedev nacos]# bin/startup.sh -p 5555
/opt/module/jdk1.8.0_171/bin/java  -server -Xms2g -Xmx2g -Xmn1g -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m -XX:-OmitStackTraceInFastThrow -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/opt/module/nacos/logs/java_heapdump.hprof -XX:-UseLargePages -Djava.ext.dirs=/opt/module/jdk1.8.0_171/jre/lib/ext:/opt/module/jdk1.8.0_171/lib/ext:/opt/module/nacos/plugins/cmdb:/opt/module/nacos/plugins/mysql -Xloggc:/opt/module/nacos/logs/nacos_gc.log -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=100M -Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8000 -Dnacos.home=/opt/module/nacos -Dloader.path=/opt/module/nacos/plugins/health -jar /opt/module/nacos/target/nacos-server.jar  --spring.config.location=classpath:/,classpath:/config/,file:./,file:./config/,file:/opt/module/nacos/conf/ --logging.config=/opt/module/nacos/conf/nacos-logback.xml --server.max-http-header-size=524288
nacos is starting with cluster
nacos is starting, you can check the /opt/module/nacos/logs/start.out
[root@sitedev nacos]# ps -aux | grep nacos
root      28597  254 55.0 5873880 1117372 pts/1 Sl   01:02   1:23 /opt/module/jdk1.8.0_171/bin/java -Dserver.port=3333 -server -Xms2g -Xmx2g -Xmn1g -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m -XX:-OmitStackTraceInFastThrow -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/opt/module/nacos/logs/java_heapdump.hprof -XX:-UseLargePages -Djava.ext.dirs=/opt/module/jdk1.8.0_171/jre/lib/ext:/opt/module/jdk1.8.0_171/lib/ext:/opt/module/nacos/plugins/cmdb:/opt/module/nacos/plugins/mysql -Xloggc:/opt/module/nacos/logs/nacos_gc.log -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=100M -Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8000 -Dnacos.home=/opt/module/nacos -Dloader.path=/opt/module/nacos/plugins/health -jar /opt/module/nacos/target/nacos-server.jar --spring.config.location=classpath:/,classpath:/config/,file:./,file:./config/,file:/opt/module/nacos/conf/ --logging.config=/opt/module/nacos/conf/nacos-logback.xml --server.max-http-header-size=524288 nacos.nacos
root      28826  9.0  0.0 112728   920 pts/1    S+   01:03   0:00 grep --color=auto nacos
[root@sitedev nacos]#
```

3) 测试通过nginx访问nacos, 浏览器访问http://192.168.5.10:1111/nacos/#/login

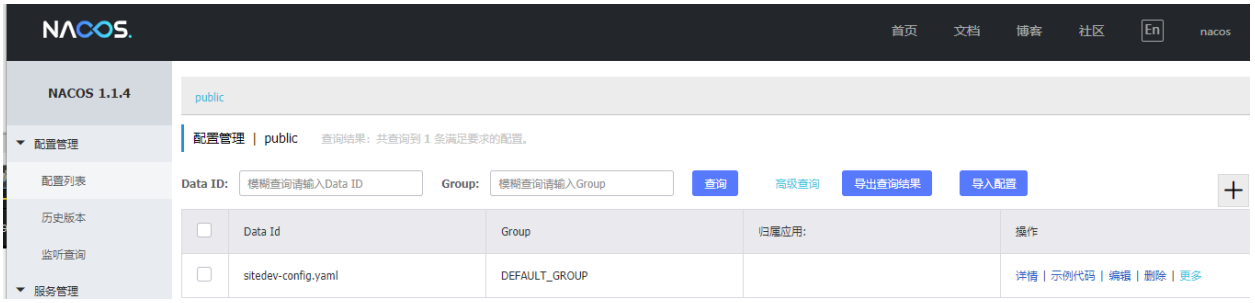
> 注意:访问地址中的IP为虚拟机的IP



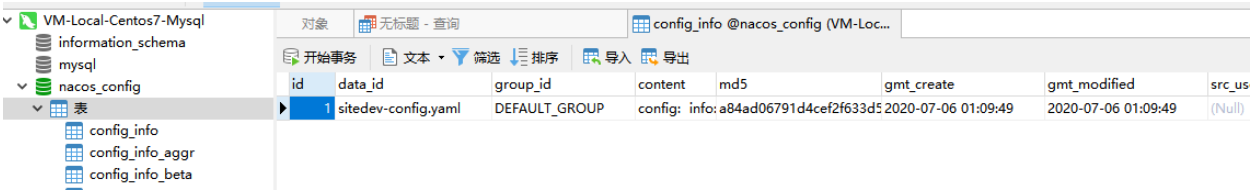
4) 新建一个配置测试

- 配置管理 -> 配置列表 -> 新建配置
- Data ID: sitedev-config.yaml
- Group: DEFAULT_GROUP
- 描述: 默认为空
- 配置格式: YAML
- 配置内容:

```yaml
config:
  info: sitedev-config.yaml,version=1
```



5) 查看linux服务器的mysql数据库, 可以看到 `config_info` 插入了一条记录



5.3.3. 测试

1) 微服务cloudalibaba-provider-payment9002启动注册进nacos集群

2) 修改YML

- 修改内容:

```yaml
spring:
  cloud:
    nacos:
      discovery:
        #        server-addr: localhost:8848 #配置Nacos地址
        server-addr: 192.168.5.10:1111 # 换成Linux虚拟机中的nginx的1111端口，做集群
```

- 完整内容:

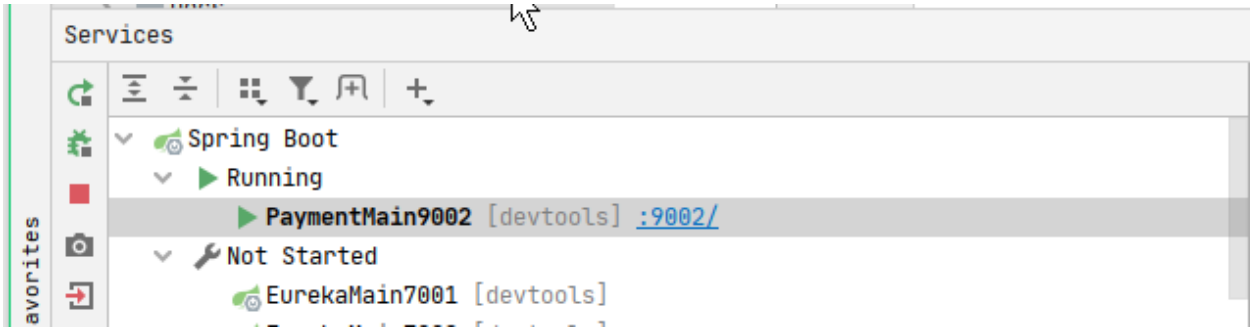
```yaml
server:
  port: 9002
spring:
  application:
    name: nacos-payment-provider
  cloud:
    nacos:
      discovery:
        #        server-addr: localhost:8848 #配置Nacos地址
        server-addr: 192.168.5.10:1111 # 换成Linux虚拟机中的nginx的1111端口，做集群
management:
```

```yaml
endpoints:
  web:
    exposure:
      include: '*'
```

3) 查看nacos服务列表(服务启动前)

- 服务管理 -> 服务列表

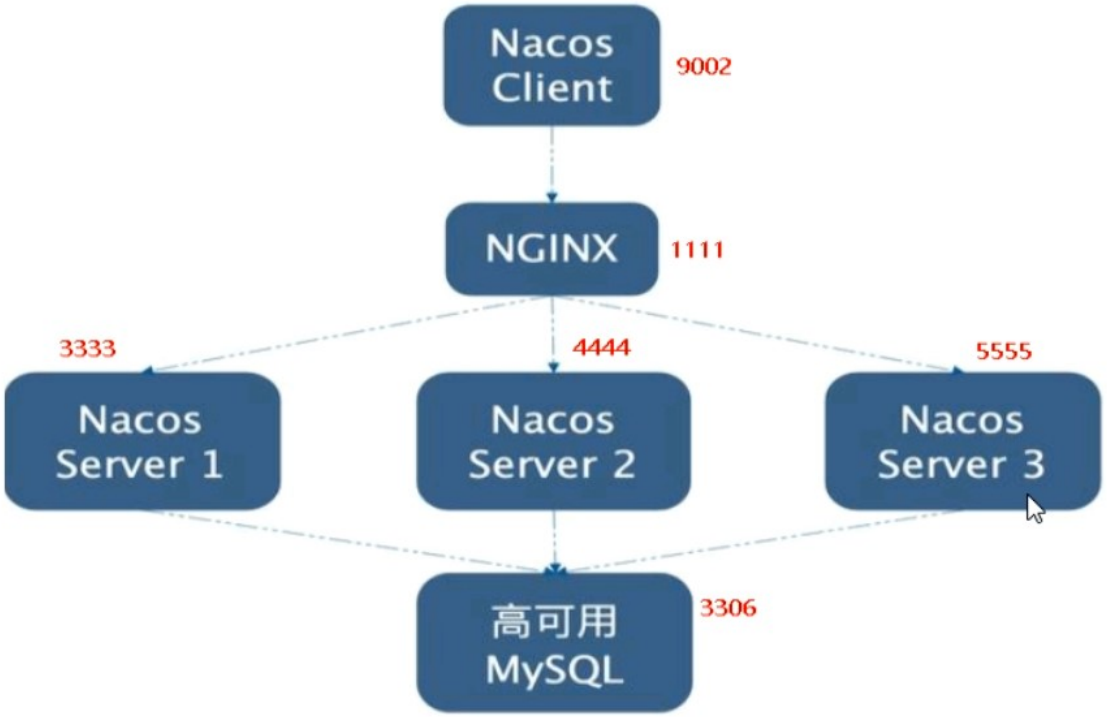
4) 启动cloudalibaba-provider-payment9002



5) 查看nacos服务列表(服务启动后)

- 服务管理 -> 服务列表

5.3.4. 高可用小结


Nacos Client
9002
NGINX
1111
3333
4444
5555
Nacos Server 1
Nacos Server 2
Nacos Server 3
高可用 MySQL
3306