【**Nacos** 注册中心】

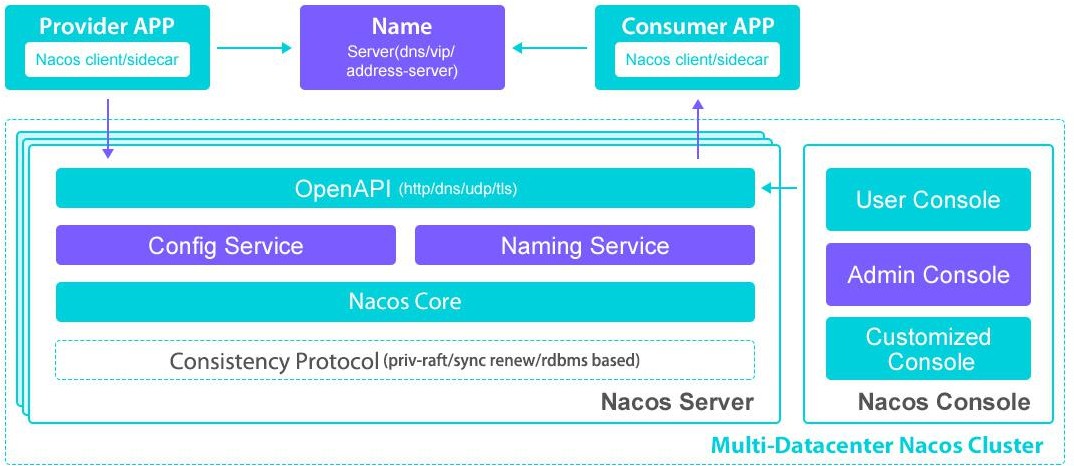
1. **Nacos** 简介

官网： <https://nacos.io/zh-cn/>

Nacos 致力于发现、配置和管理微服务。Nacos 提供了一组简单易用的特性集，帮助您快速实现动态服务发现、服务配置、服务元数据及流量管理。

Nacos 帮助您更敏捷和容易地构建、交付和管理微服务平台。 Nacos 是构建以“服务”为中心的现代应用架构 (例如微服务范式、云原生范式) 的服务基础设施。

1. **Nacos** 的核心概念



* 1. 服务 **(Service)**

服务是指一个或一组软件功能（例如特定信息的检索或一组操作的执行），其目的是不同的客户端可以为不同的目的重用（例如通过跨进程的网络调用）。Nacos 支持主流的服务生态，

如 Kubernetes Service 、 gRPC|Dubbo RPC Service 或者 Spring Cloud RESTful Service.

* 1. 服务注册中心 **(Service Registry)**

服务注册中心，它是服务实例及元数据的数据库。服务实例在启动时注册到服务注册表，并在关闭时注销。服务和路由器的客户端查询服务注册表以查找服务的可用实例。服务注册中心可能会调用服务实例的健康检查 API 来验证它是否能够处理请求。

* 1. 服务元数据 **(Service Metadata)**

服务元数据是指包括服务端点(endpoints)、服务标签、服务版本号、服务实例权重、路由规则、安全策略等描述服务的数据

* 1. 服务提供方 **(Service Provider)**

是指提供可复用和可调用服务的应用方

* 1. 服务消费方 **(Service Consumer)**

是指会发起对某个服务调用的应用方

* 1. 配置 **(Configuration)---**配置文件中心

在系统开发过程中通常会将一些需要变更的参数、变量等从代码中分离出来独立管理，以独立的配置文件的形式存在。目的是让静态的系统工件或者交付物（如 WAR，JAR 包等）更好地和实际的物理运行环境进行适配。配置管理一般包含在系统部署的过程中，由系统管理员或者运维人员完成这个步骤。配置变更是调整系统运行时的行为的有效手段之一。

# 配置管理 (Configuration Management)

在数据中心中，系统中所有配置的编辑、存储、分发、变更管理、历史版本管理、变更审计等所有与配置相关的活动统称为配置管理。

* 1. 名字服务 **(Naming Service)**

提供分布式系统中所有对象(Object)、实体(Entity)的“名字”到关联的元数据之间的映射管理服务， 例如 ServiceName -> Endpoints Info, Distributed Lock Name -> Lock Owner/Status Info, DNS Domain Name -> IP List, 服务发现和 DNS 就是名字服务的 2 大场景。

# 配置服务 (Configuration Service)

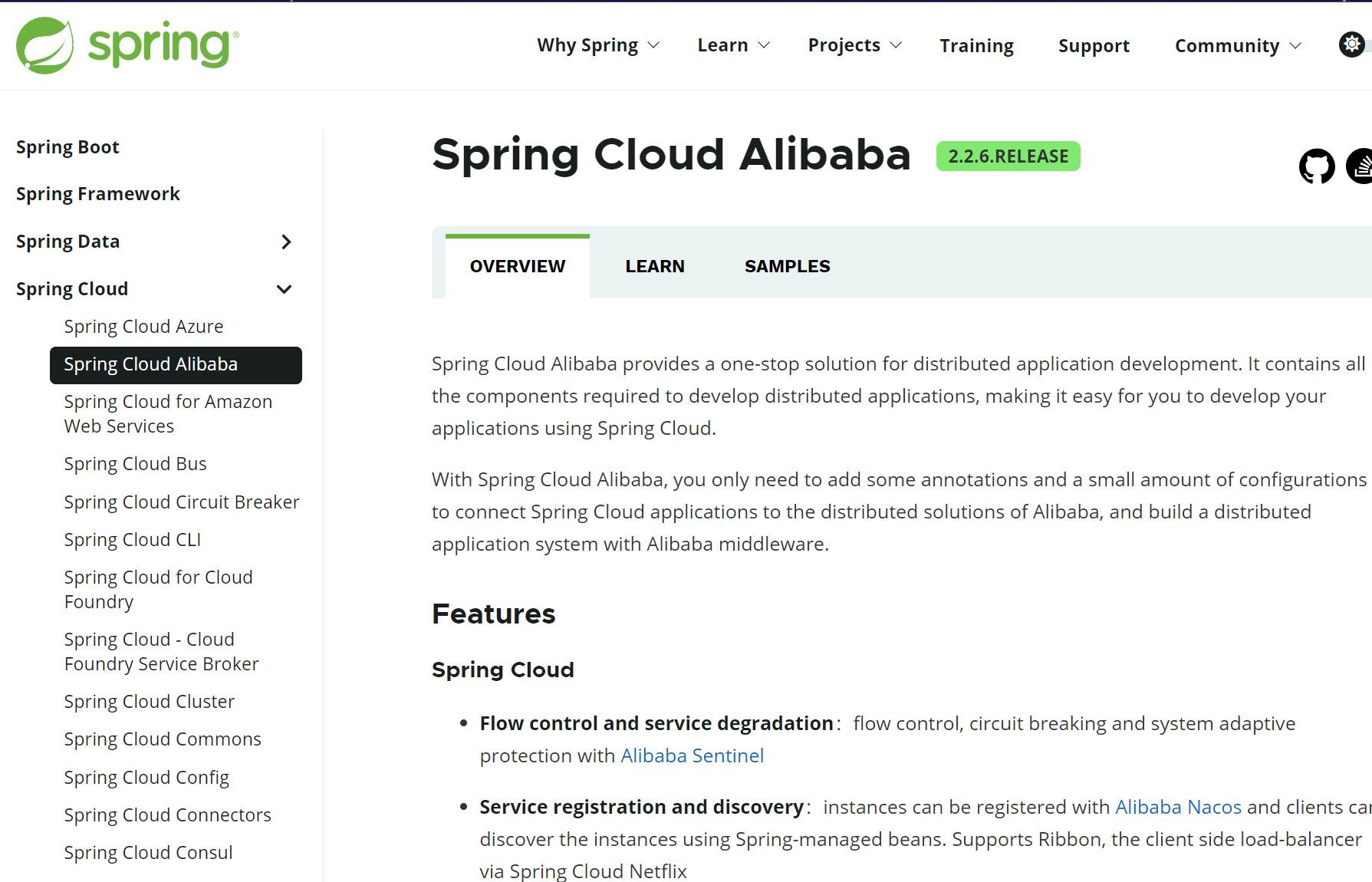
在服务或者应用运行过程中，提供动态配置或者元数据以及配置管理的服务提供者。

1. **NacosServer** 的安装和启动

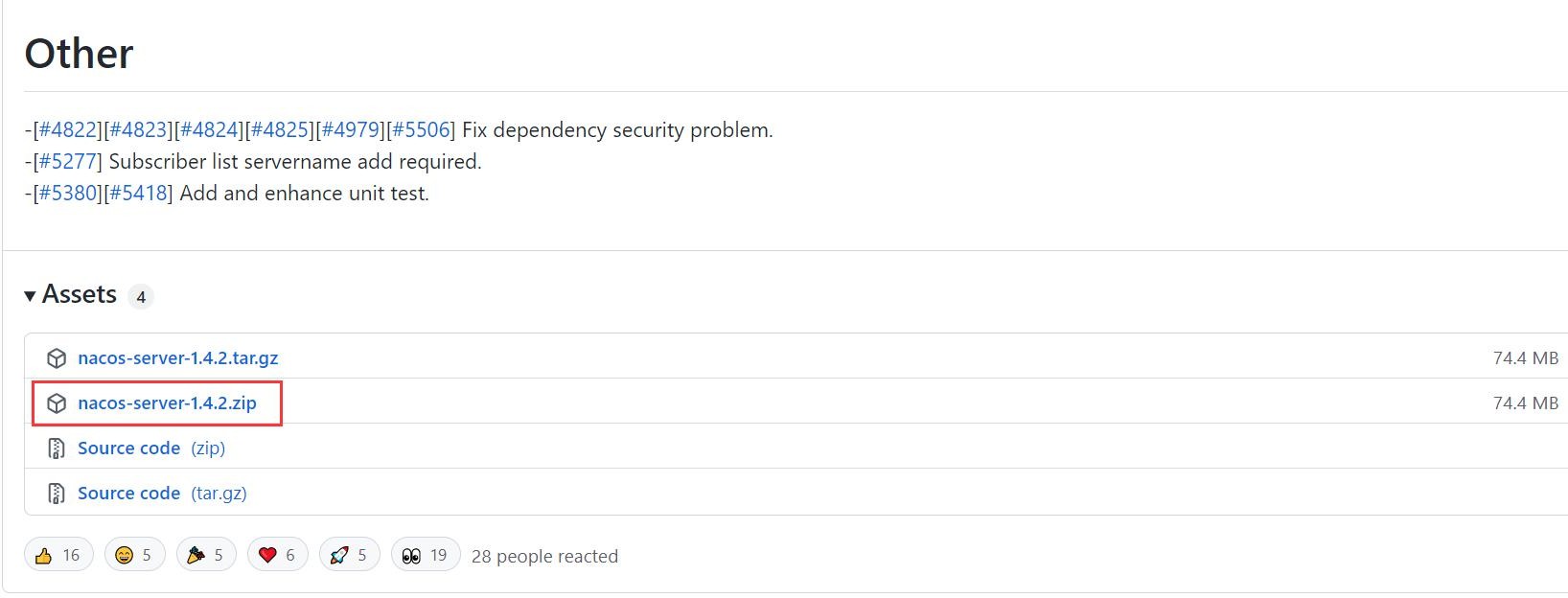
NacosServer 相当于 EurekaServer，只不过 eurekaServer 使我们自己搭建的一个项目，而 NacosServer 别人已经提供好了

# NacosServer 的下载

我们要对应版本，目前 alibaba 稳定版是 2.2.6.RELEASE



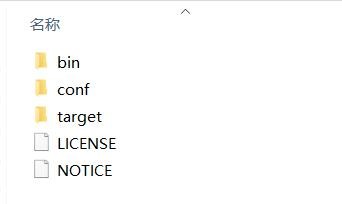
在上一节中了解了版本对应关系，**alibaba** 的 **2.2.6.RELEASE** 对应的 **nacos** 版本为 **1.4.2**

地址：https://github.com/alibaba/nacos/releases/tag/2.0.3

本项目用的是2.0.3

下载起来比较慢，我这边已经下载好了

## 解压以及目录说明



**bin**：可执行文件夹目录，包含：启动、停止命令等等 **conf**：配置文件目录

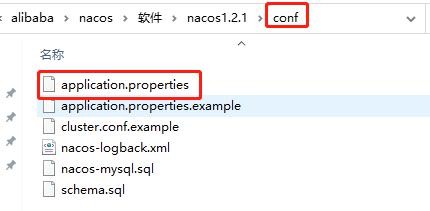
target：存放 naocs-server.jar

LICENSE：授权信息，Nacos 使用 Apache License Version 2.0 授权

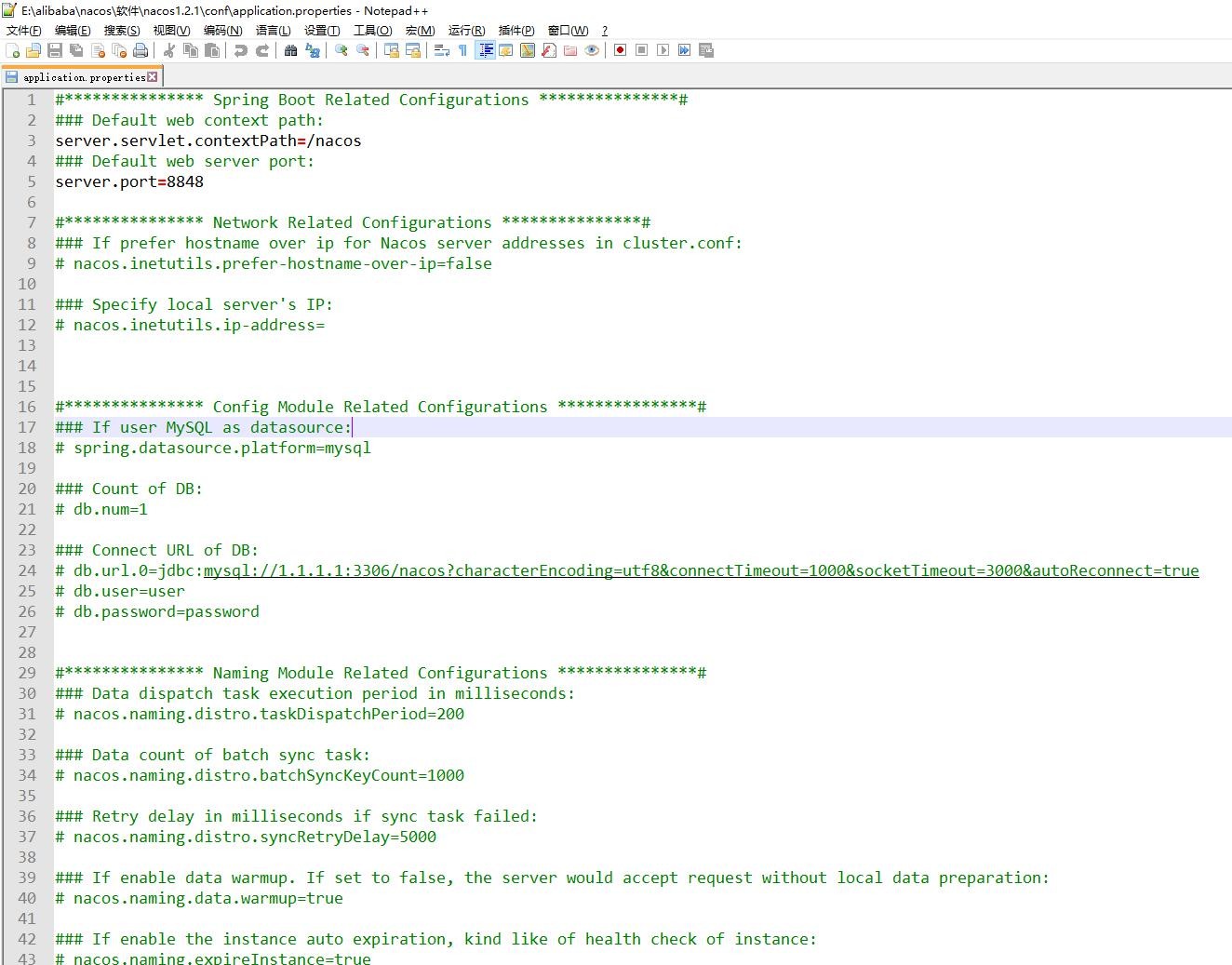
NOTICE：公告信息

解压后放这里了C:\Program Files\Java\nacos

## 修改配置文件【重点】

进入${Nacos}/conf 目录里面，使用文件编辑器打开 application.properties 文件，这里面我使用的是 Nodepad++：

打开后如图：



Nacos 默认使用嵌入式数据库实现数据的存储，并不方便观察数据存储的基本情况，这里面我们修改为使用 Mysql 数据库做数据的存储，方便我们观察数据的结构。

（把33到41行的注释打开就是了

在配置文件末尾添加如下配置：

spring.datasource.platform=mysql db.num=1

db.url.0=jdbc:mysql://**localhost:3306/nacos**?characterEncoding=utf8&connectTimeout=1 000&socketTimeout=3000&autoReconnect=true

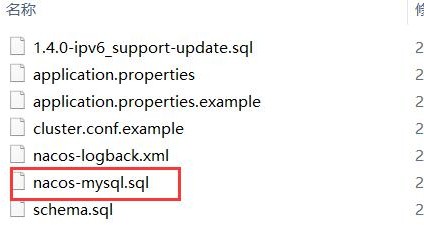
db.user=**root** db.password=**123456**

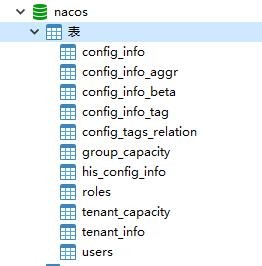
注意：上面的 url 地址是我的服务器地址，你们的就填写自己的地址。

* 1. **Mysql** 表的导入

在 config 目录下找到对应的 sql 脚本，提示：Nacos 建议使用 5.7 的 Mysql 数据库，版本较低或者较高可能存储兼容性问题

我使用的 Mysql 数据库版本为 5.7



创建一个数据库nacos，运行 sql 脚本。拖进去就可以了

* 1. **NacosServer** 的启动、

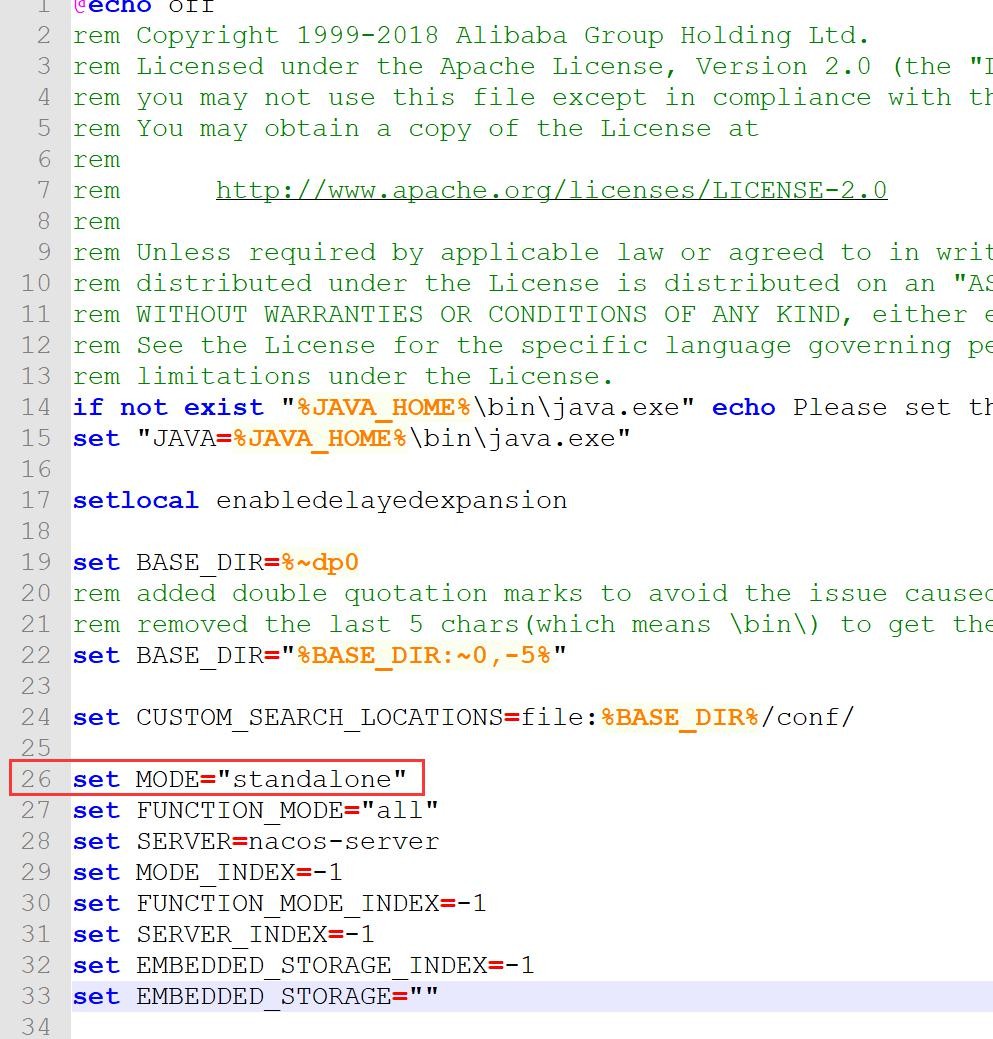
可以直接 startup.cmd -m standalone 启动单击版本

或者修改startup.cmd

上面工作都完成后，现在我们来启动一个单机版的 Nacos 服务器。进入到${Nacos}/bin 目录里面：

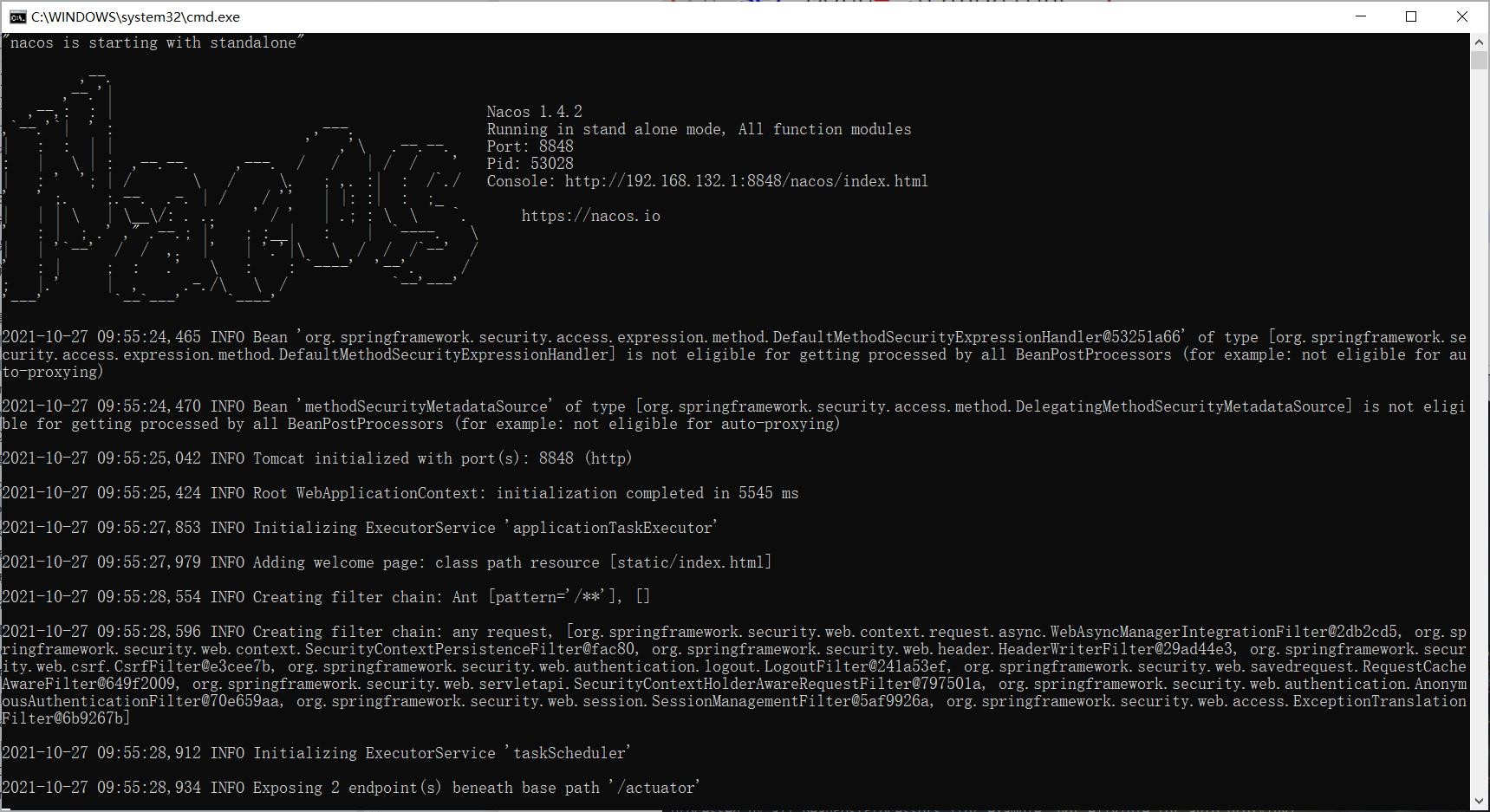
使用 **nodepad++**打开 **startup.cmd** 修改默认参数

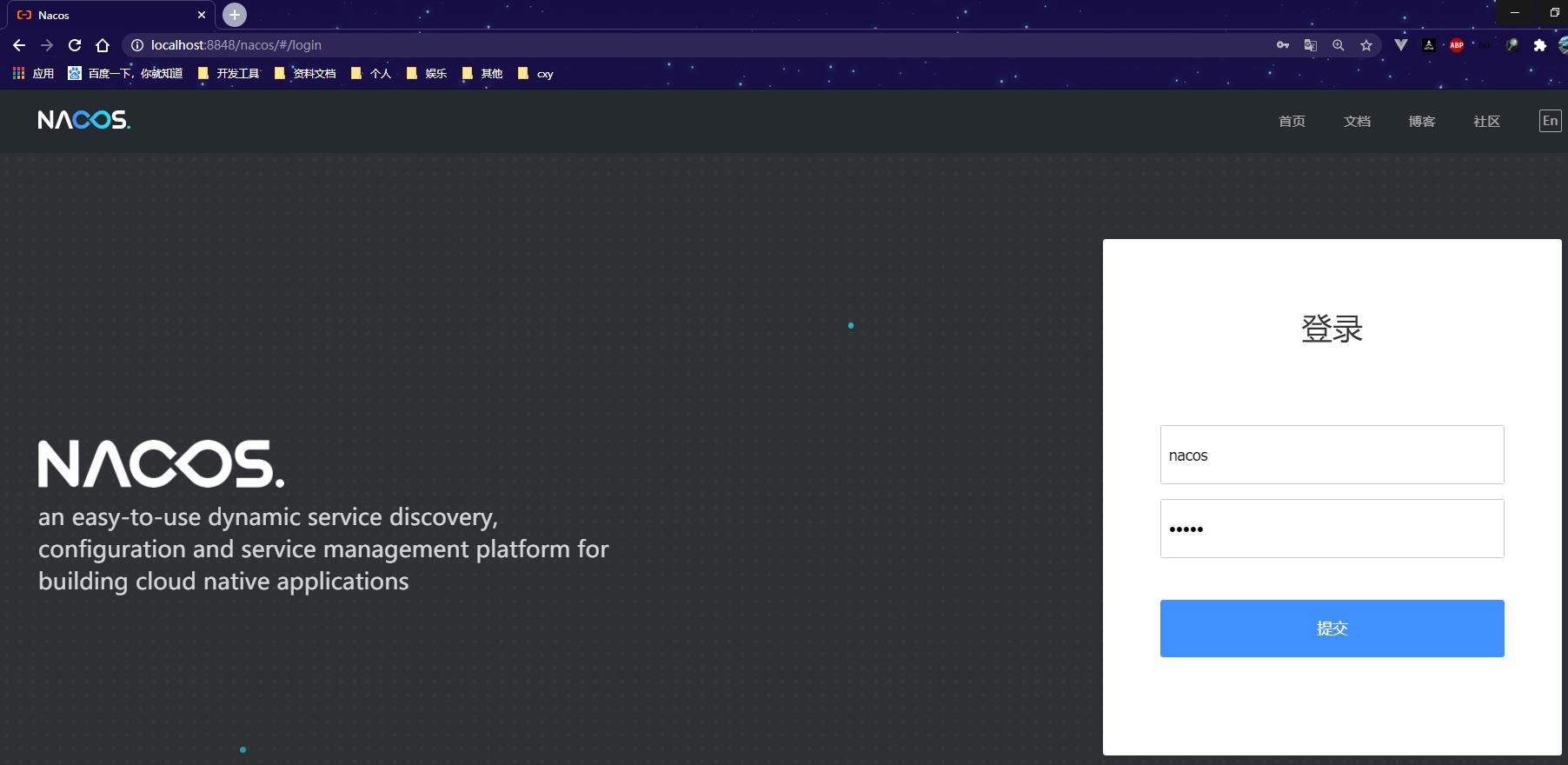
将 **set MODE=”cluster”**修改为 **standalone**



双击 startup.cmd 文件，完成 nacosServer 的启动。

（由于放在了c盘，所以要用右键管理员身份运行



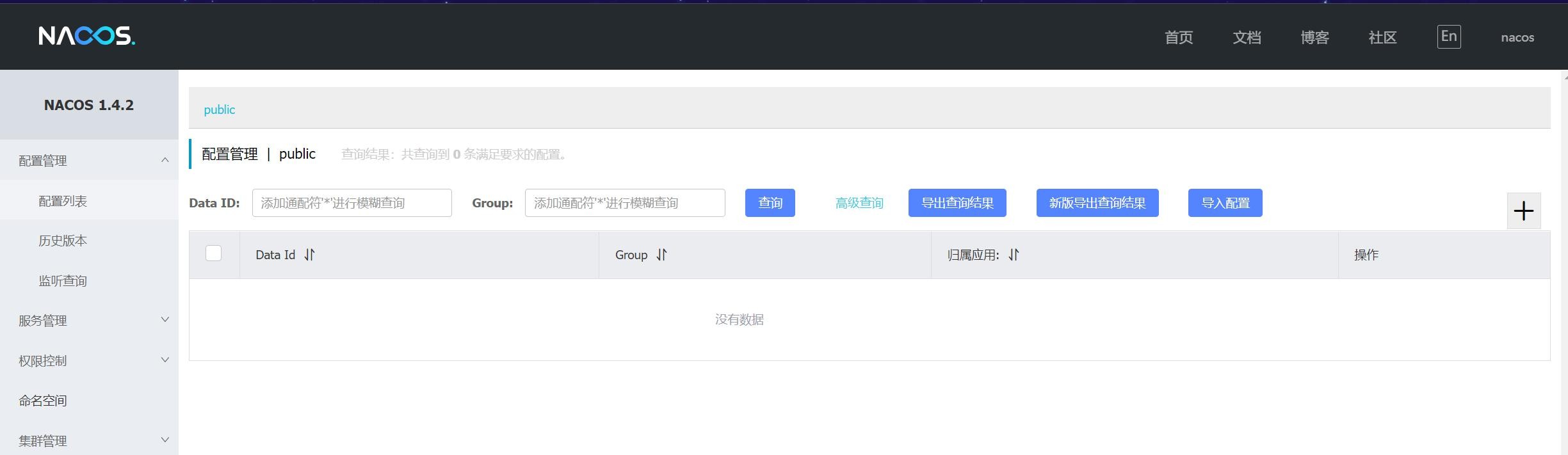
http://localhost:8848/nacos ，即可访问启动 Nacos 实例。

**Nacos** 默认用户名和密码都是 **nacos**。

如果想修改密码，可以直接修改数据库的 **user** 表，密码可以使用 **BcryptPasswordEncoder**

加密

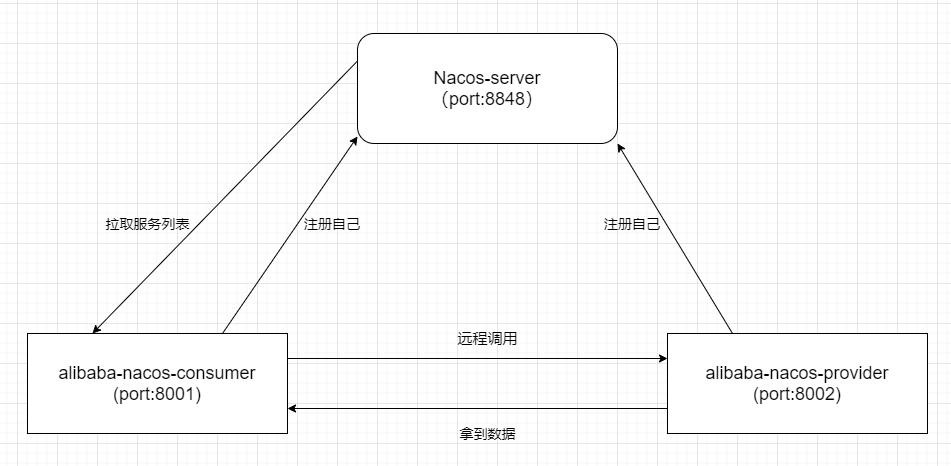
输入正确的用户名和密码提交后，出现 Nacos 的控制台界面。

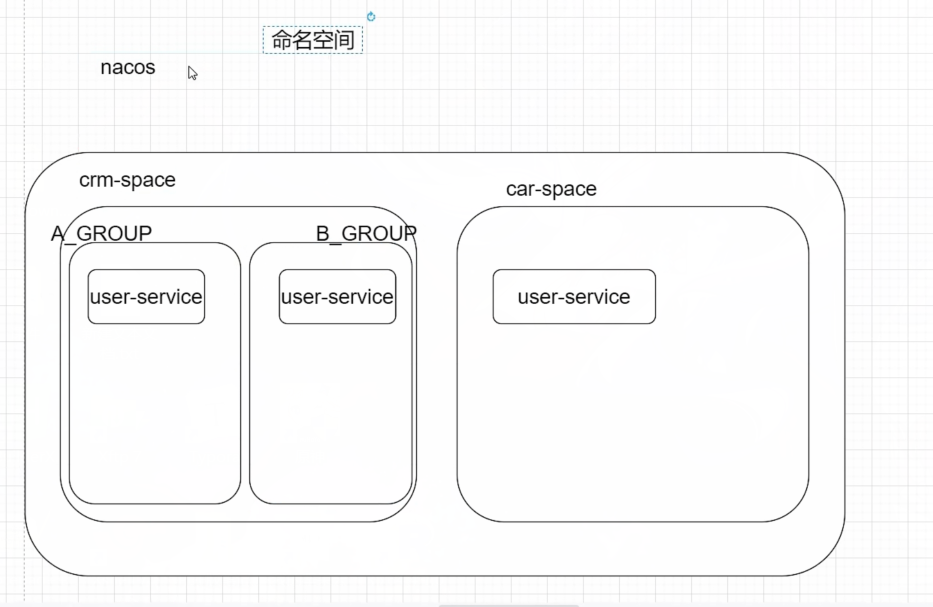


至此，Nacos Server 已经安装成功。

1. 使用 **Nacos** 做注册中心

与eureka的一些区别：nacos一开始需要登陆验证，nacos有命名空间，可以将不同域相同的服务名的服务区分。而eureka还要自己处理如加项目前缀





* 1. 搭建两个 **nacos** 的客户端

我们搭建 alibaba-nacos-consumer 和 alibaba-nacos-provider，就是一个消费者一个提供者

## 版本依赖【重点**-**再贴图一次】



* 1. 创建两个项目，选择依赖

这两个项目的依赖都是一样的，因为 **springboot** 更新原因，导致创建项目时选择不了低版本，所以直接贴出 **pom** 文件参考

*<?*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> https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

**<parent>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-parent</artifactId>**

#### <version>2.3.2.RELEASE</version>

**<relativePath/> *<!-- lookup parent from repository -->***

#### </parent>

<groupId>com.powernode</groupId>

<artifactId>01-alibaba-nacos-consumer</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>alibaba-nacos-consumer</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

***<!-- spring-cloud-alibaba*** 的当前稳定发行版本***2.2.6 -->***

#### <spring-cloud-alibaba.version>2.2.6.RELEASE</spring-cloud-alibaba.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

***<!--*** 添加***nacos*** 的依赖 ***-->***

**<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-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

***<!--*** 添加***alibaba*** 的依赖管理 ***-->***

**<dependencyManagement>**

**<dependencies>**

**<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>**

**</dependencies>**

**</dependencyManagement>**

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

# application.yml

两个配置文件基本一致，注意端口和应用名称

server:

port: 8001 spring:

application:

name: alibaba-nacos-consumer cloud:

nacos: *#* 客户端注册的地址 server-addr: localhost:8848 username: nacos

password: nacos

*# discovery: #* 命名空间可以做项目隔离 *# namespace: car-namespace*

*# group: dev #* 在命名空间下的组别，可以用来做细粒度的隔离

* 1. 修改两个启动类

@SpringBootApplication

**@EnableDiscoveryClient //**开启服务发现客户端 也就是 **nacosServer** 的客户端

public class AlibabaNacosConsumerApplication {

public static void main(String[] args) { SpringApplication.*run*(AlibabaNacosConsumerApplication.class, args);

}

}

# 给 alibaba-nacos-provider 添加一个 controller

package com.bjpowernode.controller;

import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.RestController;

/\*\*

* @Author 武汉动力节点

\*/ @RestController

public class ProviderController {

/\*\*

* + 打招呼的接口

\*

* + @param name
  + @return

\*/ @GetMapping("hello")

public String hello(String name) { return "hello:"+name;

}

}

* 1. 启动 **alibaba-nacos-provider** 测试

启动后去看 nacosServer 的控制台，已经有实例注册上去了



访问测试： http://localhost:8002/hello?name=cxs

* 1. 给 **alibaba-nacos-consumer** 添加一个 **controller** 做服务发现

package com.bjpowernode.controller;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.cloud.client.ServiceInstance;

import org.springframework.cloud.client.discovery.DiscoveryClient; import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

/\*\*

* @Author 武汉动力节点

\*/ @RestController

public class ConsumerController {

/\*\*

* + 注入服务发现组件 在 eureka 中也用过

\*/ @Autowired

private DiscoveryClient discoveryClient;

/\*\*

* + 服务发现的接口

\*

* + @param serviceId
  + @return

\*/ @GetMapping("discovery")

public String discoveryService(String serviceId) {

// 根据实例名称拿到实例集合

List<ServiceInstance> instances = discoveryClient.getInstances(serviceId);

// 从实例集合列表中获取一个实例对象

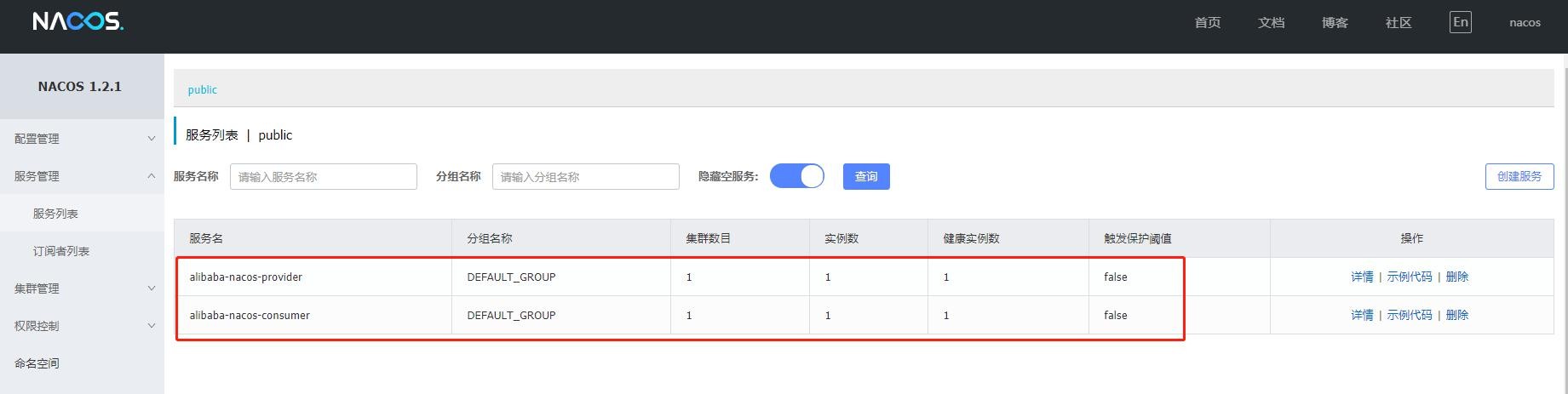
ServiceInstance serviceInstance = instances.get(0); System.*out*.println(serviceInstance.getHost() + ":" + serviceInstance.getPort());

return serviceInstance.getHost() + ":" + serviceInstance.getPort();

}

}

# 启动 alibaba-nacos-consumer 测试

先查看 NacosServer 是否注册上线

访问测试 http://localhost:8001/discovery?serviceId=**alibaba-nacos-provider**



至此，服务注册和服务发现已经完成了，基本和 eureka 一样

（目前只能同命名空间和同组才能发现。其他的没讲到，要的话自己去学。。

<http://localhost:8080/test>

加了第三个项目的网关后，

直接localhost:nacos-client-a/test

网关就配这点就行了

server:  
 port: 80  
spring:  
 application:  
 name: gateway  
 cloud:  
 nacos:  
 server-addr: localhost:8848  
 username: nacos  
 password: nacos  
 discovery:  
 namespace: 3242d90a-9222-4285-93b9-efad95db278f  
 group: A\_GROUP  
 gateway:  
 discovery:  
 locator:  
 enabled: true *# 开启动态路由* lower-case-service-id: true

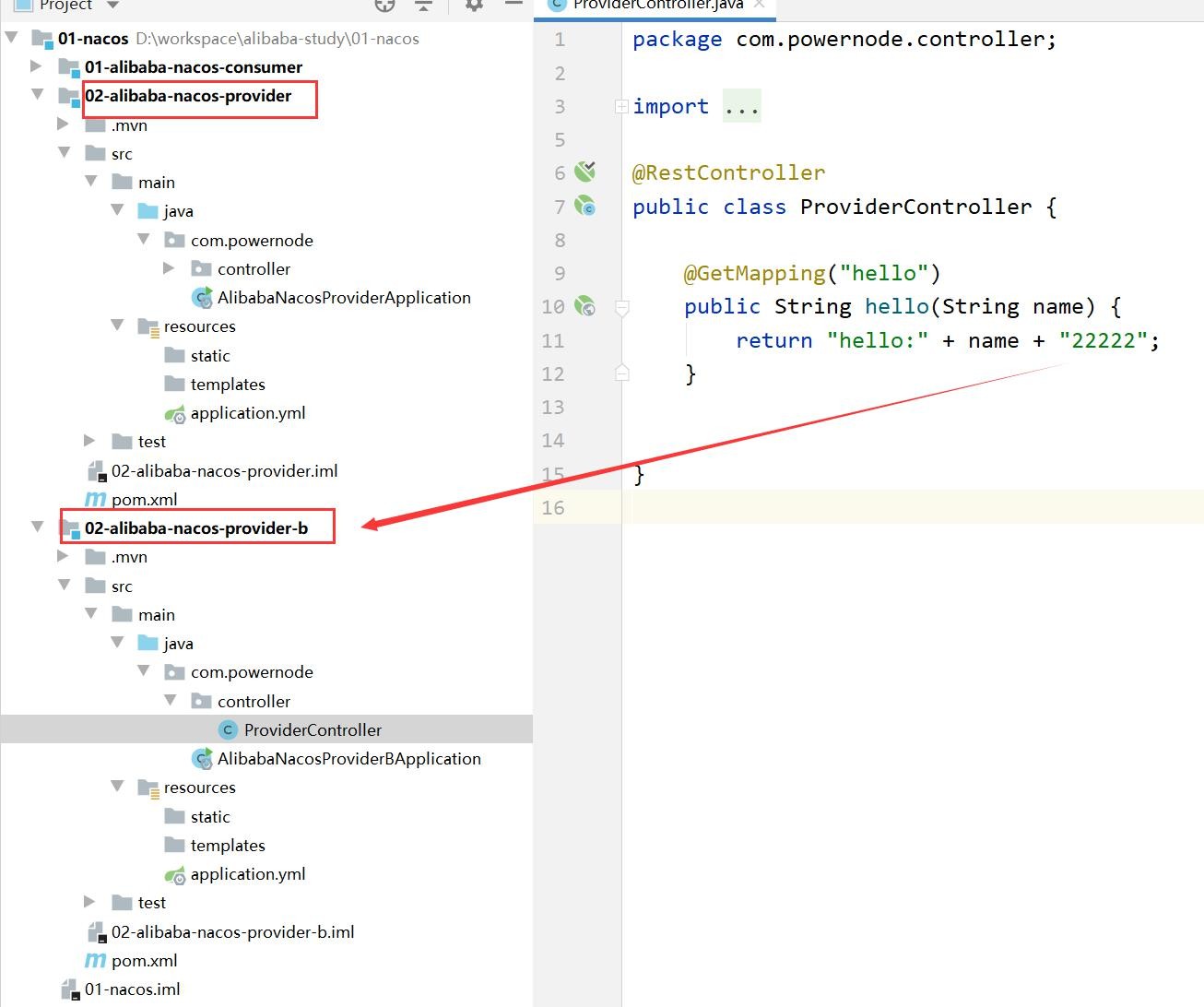
1. 集成 **openfeign** 做远程调用和负载均衡

如果没有学 feign 的同学，可以使用 restTemplate 来做

a去调用b。nacos比eureka快

# 启动多台 alibaba-nacos-provider

我们可以再建一个项目，来达到多台 **provider** 的效果，我们也可以通过修改配置文件的方式多启动多台，我们这里选择在建一个项目，只有端口和 **controller** 输出改变，其他都不改变



# 修改 alibaba-nacos-consumer

* + 1. 添加 **openfeign** 的依赖，注意还需要 **cloud** 的依赖管理

*<?*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> https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.3.2.RELEASE</version>

<relativePath/> *<!-- lookup parent from repository -->*

</parent>

<groupId>com.powernode</groupId>

<artifactId>01-alibaba-nacos-consumer</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>alibaba-nacos-consumer</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

#### <spring-cloud.version>Hoxton.SR9</spring-cloud.version>

*<!-- spring-cloud-alibaba* 的当前稳定发行版本*2.2.6 -->*

<spring-cloud-alibaba.version>2.2.6.RELEASE</spring-cloud-alibaba.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-starter-openfeign</artifactId>**

**</dependency>**

*<!--* 添加*nacos* 的依赖 *-->*

<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-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

*<!--* 添加*alibaba* 的依赖管理 *-->*

<dependencyManagement>

<dependencies>

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-dependencies</artifactId>**

**<version>${spring-cloud.version}</version>**

**<type>pom</type>**

**<scope>import</scope>**

**</dependency>**

<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>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

### 修改启动类，添加注解

@SpringBootApplication

@EnableDiscoveryClient //开启服务发现客户端 也就是 nacosServer 的客户端

@EnableFeignClients //开启 feign 的客户端

public class AlibabaNacosConsumerApplication {

public static void main(String[] args) { SpringApplication.*run*(AlibabaNacosConsumerApplication.class, args);

}

}

* + 1. 添加一个 **feign** 的接口，注意和提供者一致

package com.bjpowernode.feign;

import org.springframework.cloud.openfeign.FeignClient; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.RequestParam;

/\*\*

* @Author 武汉动力节点

\*/

@FeignClient(value = "alibaba-nacos-provider") public interface ProviderFeign {

/\*\*

* + 远程调用打招呼的接口

\*

* + @param name
  + @return

\*/ @GetMapping("hello")

String hello(@RequestParam("name") String name);

}

* + 1. 添加一个 **controller**

@Autowired

private ProviderFeign providerFeign;

/\*\*

* 测试远程调用
* @return

\*/ @GetMapping("rpc")

public String testRpc() {

String bjpowernode = providerFeign.hello("bjpowernode"); System.*out*.println(bjpowernode);

return bjpowernode;

}

### 启动测试

访问：http://localhost:8001/rpc

至此，nacos 做注册中心，服务发现，以及远程调用都完成了

1. **Nacos Discovery** 对外暴露 **Endpoint**

Nacos Discovery 内 部 提 供 了 一 个 Endpoint, 对 应 的 endpoint id 为

nacos-discovery。我们通过该 Endpoint，能获取到：

* 当前服务有哪些服务订阅者 ；
* 当前应用 Nacos 的基础配置信息 ；

## 给任意项目添加依赖

假设我们想看消费者的一些信息，我们给消费者添加

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

## 修改配置文件

Endpoint 本身对外界隐藏显示，我们需要在配置里面开启对 Endponit 的显示支持。修改 application.yml 配置文件，在里面添加如下的配置：

management:

endpoints: web:

exposure:

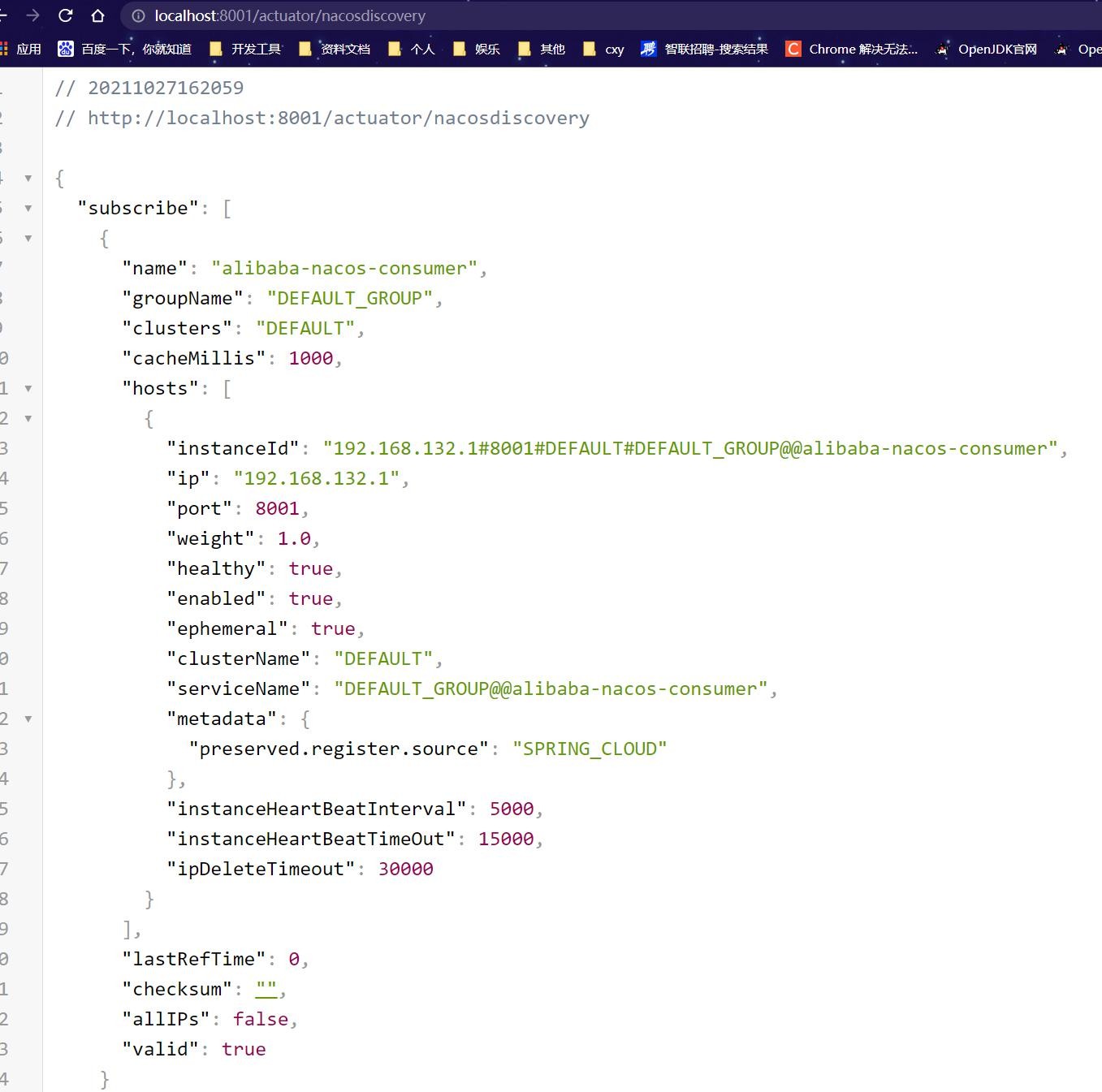
include: '\*'

说明：

* exposure.include：对外界保留那些 Endpoint，若是所有则使用\* ；

## 启动项目访问查看效果

http://localhost:8001/actuator http://localhost:8001/actuator/nacosdiscovery



1. **Nacos Discovery Starter** 更多的配置项

|  |  |  |  |
| --- | --- | --- | --- |
| 配置项 **Key** 默认值 说明 | | | |
| 服务端地址 | spring.cloud.nacos.discovery.s erver-addr | 无 | Nacos Server 启动监听的ip 地址和端口 |
| 服务名 | spring.cloud.nacos.discovery.s ervice | ${spring.applic ation.name} | 给当前的服务命名 |
| 服务分组 | spring.cloud.nacos.discovery.g roup | DEFAULT\_GROUP | 设置服务所处的分组 |
| 权重 | spring.cloud.nacos.discovery.w eight | 1 | 取值范围 1 到 100，数值越大，权重越大 |
| 网卡名 | spring.cloud.nacos.discovery.n etwork-interface | 无 | 当 IP 未配置时，注册的IP 为此网卡所对应的 IP 地址，如果此项也未配置，则默认取第一块网卡的地址 |
| 注册的IP  地址 | spring.cloud.nacos.discovery.i p | 无 | 优先级最高 |
| 注册的端口 | spring.cloud.nacos.discovery.p ort | -1 | 默认情况下不用配置，会自动探测 |
| 命名空间 | spring.cloud.nacos.discovery.n amespace | 无 | 常用场景之一是不同环境的注册的区分隔离，例如开发测试环境和生产环境的资源（如配置、服务）隔离等。 |
| AccessKey | spring.cloud.nacos.discovery.a ccess-key | 无 | 当要上阿里云时，阿里云上面的一个云账号名 |
| SecretKey | spring.cloud.nacos.discovery.s ecret-key | 无 | 当要上阿里云时，阿里云上面的一个云账号密码 |
| Metadata | spring.cloud.nacos.discovery.m etadata | 无 | 使用Map 格式配置，用户可以根据自己的需要自定义一些和服务相关的元数据信息 |
| 日志文件名 | spring.cloud.nacos.discovery.l og-name | 无 |  |
| 集群 | spring.cloud.nacos.discovery.c luster-name | DEFAULT | 配置成 Nacos 集群名称 |
| 接入点 | spring.cloud.nacos.discovery.e npoint | UTF-8 | 地域的某个服务的入口域名，通过此域名可以动态地拿到服务端地址 |
| 是否集成  Ribbon | ribbon.nacos.enabled | true | 一般都设置成 true 即可 |
| 是否开启 Nacos Watch | spring.cloud.nacos.discovery.w atch.enabled | true | 可以设置成false 来关闭 watch |

后面还有个配置nacos集群的。就是改一些配置啥的，要的话自己看后一个视频