

## 【Nacos 注册中心】

### 1. Nacos 简介

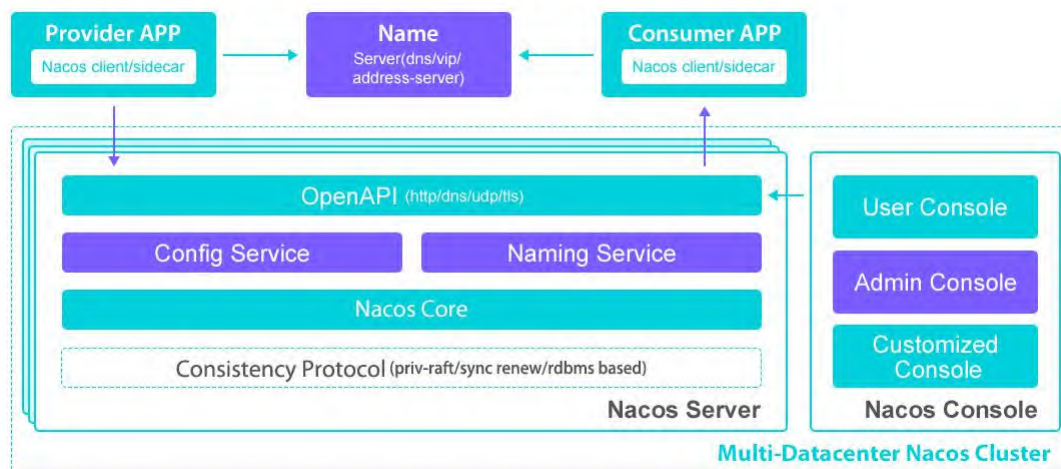
官网: <https://nacos.io/zh-cn/>



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。

## 2. Nacos 的核心概念



### 2.1 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。

### 2.2 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。

### 2.3 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

### 2.4 服务提供方 (Service Provider)

是指提供可复用和可调用服务的应用方

## 2.5 服务消费方 (Service Consumer)

是指会发起对某个服务调用的应用方

## 2.6 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。

## 2.7 配置管理 (Configuration Management)

在数据中心的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。

## 2.8 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。

## 2.9 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。

# 3. 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

## 3.1 NacosServer 的下载

我们要对应版本，目前 alibaba 稳定版是 2.2.6.RELEASE



- Spring Boot
- Spring Framework
- Spring Data
- Spring Cloud
  - Spring Cloud Azure
  - Spring Cloud Alibaba**
  - Spring Cloud for Amazon Web Services
  - Spring Cloud Bus
  - Spring Cloud Circuit Breaker
  - Spring Cloud CLI
  - Spring Cloud for Cloud Foundry
  - Spring Cloud - Cloud Foundry Service Broker
  - Spring Cloud Cluster
  - Spring Cloud Commons
  - Spring Cloud Config
  - Spring Cloud Connectors
  - Spring Cloud Consul

## Spring Cloud Alibaba 2.2.6.RELEASE

OVERVIEW LEARN SAMPLES

Spring Cloud Alibaba provides a one-stop solution for distributed application development. It contains all the components required to develop distributed applications, making it easy for you to develop your applications using Spring Cloud.

With Spring Cloud Alibaba, you only need to add some annotations and a small amount of configurations to connect Spring Cloud applications to the distributed solutions of Alibaba, and build a distributed application system with Alibaba middleware.

### Features

#### Spring Cloud

- **Flow control and service degradation:** flow control, circuit breaking and system adaptive protection with [Alibaba Sentinel](#)
- **Service registration and discovery:** instances can be registered with [Alibaba Nacos](#) and clients can discover the instances using Spring-managed beans. Supports Ribbon, the client side load-balancer via [Spring Cloud Netflix](#)

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/1.4.2>

### Other

-[#4822][#4823][#4824][#4825][#4979][#5506] Fix dependency security problem.  
-[#5277] Subscriber list servername add required.  
-[#5380][#5418] Add and enhance unit test.

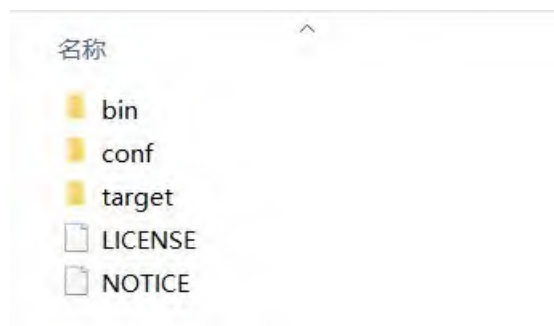
#### Assets

- [nacos-server-1.4.2.tar.gz](#) 74.4 MB
- [nacos-server-1.4.2.zip](#) 74.4 MB
- [Source code \(zip\)](#)
- [Source code \(tar.gz\)](#)

16 5 5 5 5 19 28 people reacted

下载起来比较慢，我这边已经下载好了

## 3.2 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

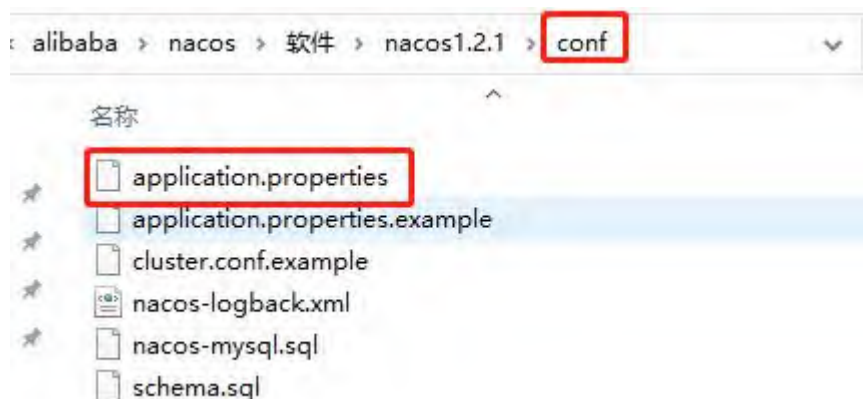
**target:** 存放 naocs-server.jar

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

**NOTICE:** 公告信息

## 3.3 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, 这里我使用的是 Nodepad++:



打开后如图:



```
E:\alibaba\nacos\软件\nacos1.2.1\conf\application.properties - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(O) 工具(T) 宏(M) 运行(R) 插件(P) 窗口(W) 2
application.properties
1  ##### Spring Boot Related Configurations #####
2  ### Default web context path:
3  server.servlet.contextPath=/nacos
4  ### Default web server port:
5  server.port=8848
6
7  ##### Network Related Configurations #####
8  ### If prefer hostname over ip for Nacos server addresses in cluster.conf:
9  # nacos.inetutils.prefer-hostname-over-ip=false
10
11 ### Specify local server's IP:
12 # nacos.inetutils.ip-address=
13
14
15
16 ##### Config Module Related Configurations #####
17 ### If user MySQL as datasource:
18 # spring.datasource.platform=mysql
19
20 ### Count of DB:
21 # db.num=1
22
23 ### Connect URL of DB:
24 # db.url.0=jdbc:mysql://1.1.1.1:3306/nacos?characterEncoding=utf8&connectTimeout=1000&socketTimeout=3000&autoReconnect=true
25 # db.user=user
26 # db.password=password
27
28
29 ##### Naming Module Related Configurations #####
30 ### Data dispatch task execution period in milliseconds:
31 # nacos.naming.distro.taskDispatchPeriod=200
32
33 ### Data count of batch sync task:
34 # nacos.naming.distro.batchSyncKeyCount=1000
35
36 ### Retry delay in milliseconds if sync task failed:
37 # nacos.naming.distro.syncRetryDelay=5000
38
39 ### If enable data warmup. If set to false, the server would accept request without local data preparation:
40 # nacos.naming.data.warmup=true
41
42 ### If enable the instance auto expiration, kind like of health check of instance:
43 # nacos.naming.expireInstance=true
```

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。

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=1000&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。

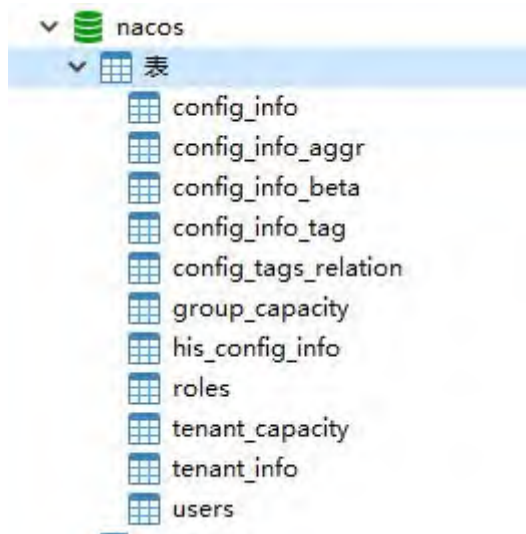
### 3.4 Mysql 表的导入

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

名称	修
1.4.0-ipv6_support-update.sql	2
application.properties	2
application.properties.example	2
cluster.conf.example	2
nacos-logback.xml	2
nacos-mysql.sql	2
schema.sql	2

创建数据库，运行 sql 脚本



### 3.5 NacosServer 的启动、

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

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 notepad++ 打开 startup.cmd 修改默认参数

将 `set MODE="cluster"` 修改为 `standalone`

```
1  #!/bin/sh
2  rem Copyright 1999-2018 Alibaba Group Holding Ltd.
3  rem Licensed under the Apache License, Version 2.0 (the "License");
4  rem you may not use this file except in compliance with the License.
5  rem You may obtain a copy of the License at
6  rem
7  rem     http://www.apache.org/licenses/LICENSE-2.0
8  rem
9  rem Unless required by applicable law or agreed to in writing, software
10 rem distributed under the License is distributed on an "AS IS" BASIS,
11 rem WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 rem See the License for the specific language governing permissions and
13 rem limitations under the License.
14 if not exist "%JAVA_HOME%\bin\java.exe" echo Please set the JAVA_HOME variable in your environment, for instance:
15 set "JAVA_HOME=%JAVA_HOME%\bin\java.exe"
16
17 setlocal enabledelayedexpansion
18
19 set BASE_DIR=%~dp0
20 rem added double quotation marks to avoid the issue caused by spaces
21 rem removed the last 5 chars(which means \bin\) to get the base dir
22 set BASE_DIR="%BASE_DIR:~0,-5%"
23
24 set CUSTOM_SEARCH_LOCATIONS=file:%BASE_DIR%/conf/
25
26 set MODE="standalone"
27 set FUNCTION_MODE="all"
28 set SERVER=nacos-server
29 set MODE_INDEX=-1
30 set FUNCTION_MODE_INDEX=-1
31 set SERVER_INDEX=-1
32 set EMBEDDED_STORAGE_INDEX=-1
33 set EMBEDDED_STORAGE=""
34
```

双击 startup.cmd 文件，完成 nacosServer 的启动。

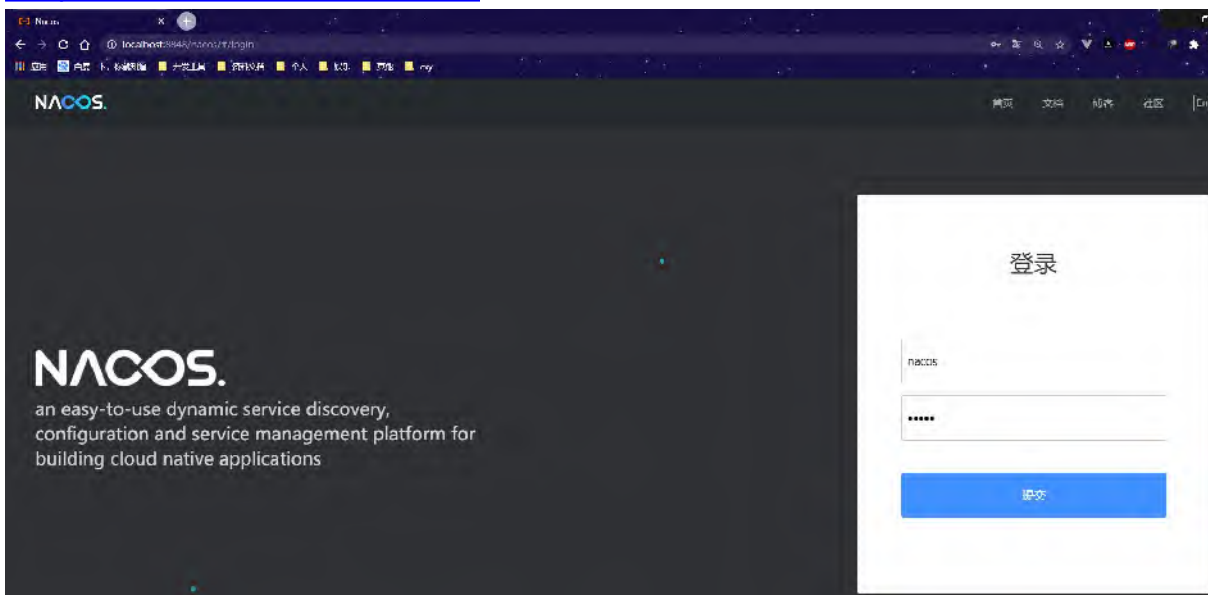


```
C:\WINDOWS\system32\cmd.exe
nacos is starting with standalone

Nacos 1.1.2
Running in stand alone mode. All function modules
Port: 8848
Pid: 53228
Console: http://192.168.132.1:8848/nacos/index.html
https://nacos.io

2021-10-27 09:55:24.465 INFO Bean 'org.springframework.security.access.expression.method.DefaultMethodSecurityExpressionHandler$MethodSecurityMetadataSource' of type [org.springframework.security.access.expression.method.DefaultMethodSecurityExpressionHandler] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto proxying)
2021-10-27 09:55:24.470 INFO Bean 'methodSecurityMetadataSource' of type [org.springframework.security.access.expression.method.DefaultMethodSecurityExpressionHandler] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto proxying)
2021-10-27 09:55:25.042 INFO Tomcat initialized with port(s): 8848 (http)
2021-10-27 09:55:25.424 INFO Root WebApplicationContext: initialization completed in 5545 ms
2021-10-27 09:55:27.853 INFO Initializing ExecutorService 'applicationTaskExecutor'
2021-10-27 09:55:27.970 INFO Adding welcome page: class path resource [static/index.html]
2021-10-27 09:55:28.554 INFO Creating filter chain: Ant [pattern='/*']
2021-10-27 09:55:28.596 INFO Creating filter chain: any request, [org.springframework.security.web.context.request.async.WebAsyncManagerIntegrationFilter@2b2cd5, org.springframework.security.web.context.SecurityContextHolder@2a4143, org.springframework.security.web.header.HeaderWriterFilter@22ad4143, org.springframework.security.web.csrf.CsrfFilter@2c3cc7b, org.springframework.security.web.authentication.logout.LogoutFilter@241a33cf, org.springframework.security.web.savedrequest.RequestCacheAwareFilter@6648f2009, org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@8797901a, org.springframework.security.web.authentication.AnonymousAuthenticationFilter@70c6b9aa, org.springframework.security.web.session.SessionManagementFilter@3a19923a, org.springframework.security.web.access.ExceptionTranslationFilter@6659267b]
2021-10-27 09:55:28.912 INFO Initializing ExecutorService 'taskScheduler'
2021-10-27 09:55:28.934 INFO Exposing 2 endpoint(s) beneath base path '/actuator'
```

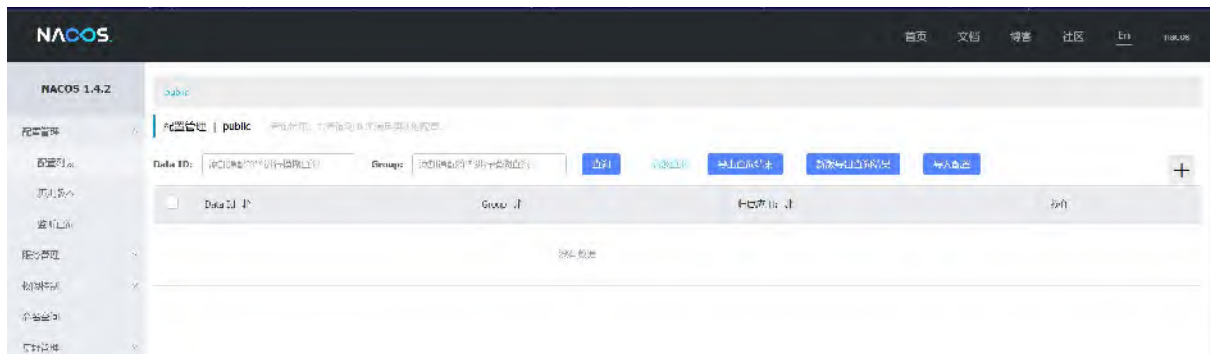
<http://localhost:8848/nacos> , 即可访问启动 Nacos 实例。



**Nacos 默认用户名和密码都是 nacos。**

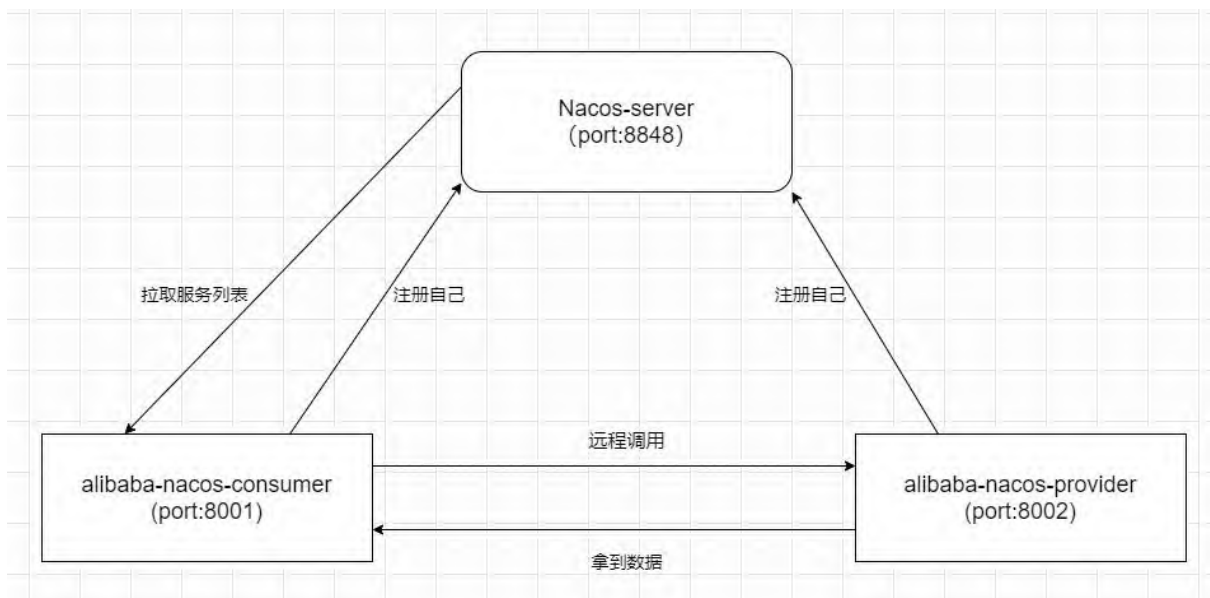
**如果想修改密码，可以直接修改数据库的 user 表，密码可以使用 BcryptPasswordEncoder 加密**

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。

## 4. 使用 Nacos 做注册中心



### 4.1 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

## 4.2 版本依赖【重点-再贴图一次】

### 毕业版本依赖关系(推荐使用)

Spring Cloud Version	Spring Cloud Alibaba Version	Spring Boot Version
Spring Cloud 2020.0.1	2021.1	2.4.2
Spring Cloud Hoxton.SR9	2.2.6.RELEASE	2.3.2.RELEASE
Spring Cloud Greenwich.SR6	2.1.4.RELEASE	2.1.13.RELEASE
Spring Cloud Hoxton.SR3	2.2.1.RELEASE	2.2.5.RELEASE
Spring Cloud Hoxton.RELEASE	2.2.0.RELEASE	2.2.X.RELEASE
Spring Cloud Greenwich	2.1.2.RELEASE	2.1.X.RELEASE
Spring Cloud Finchley	2.0.4.RELEASE(停止维护, 建议升级)	2.0.X.RELEASE
Spring Cloud Edgware	1.5.1.RELEASE(停止维护, 建议升级)	1.5.X.RELEASE

## 4.3 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>
```

## 4.4 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
```

## 4.5 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);
    }

}
```

## 4.6 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;
    }

}
```

## 4.7 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>



hello:cxs

## 4.8 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();
}
}
```

## 4.9 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

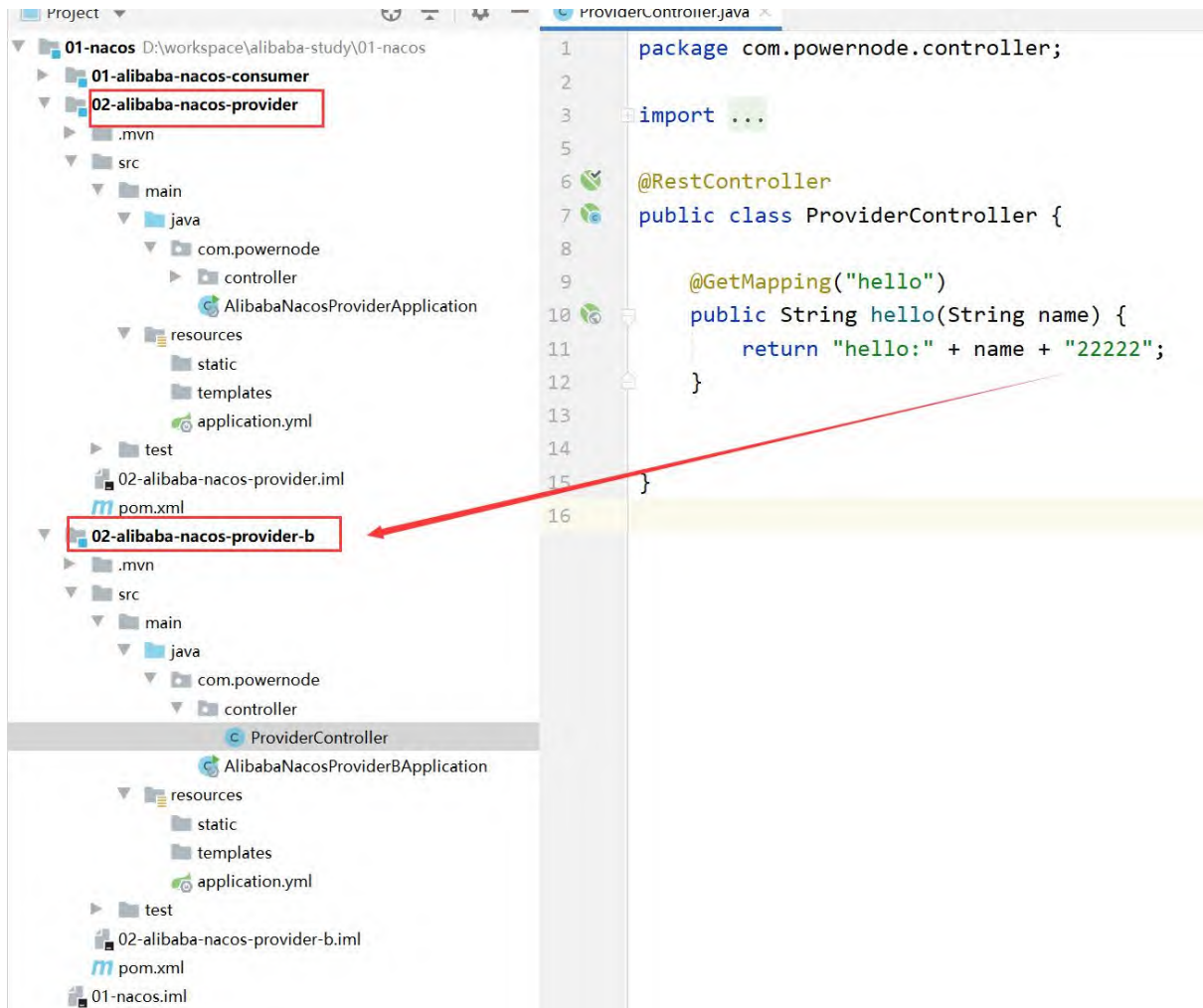
## 5. 集成 openfeign 做远程调用和负载均衡

如果没有学 feign 的同学，可以使用 restTemplate 来做

### 5.1 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，我们这里选择在建立一个项目，只有端口和 controller 输出改变，其他都不改变





## 5.2 修改 alibaba-nacos-consumer

### 5.2.1 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>
```

```
</plugins>

</build>

</project>
```

### 5.2.2 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);
    }
}
```

### 5.2.3 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);
}
```

### 5.2.4 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;
}
```

### 5.2.5 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

## 6. 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 ;

### 6.1 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>
```

### 6.2 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: '*'
```

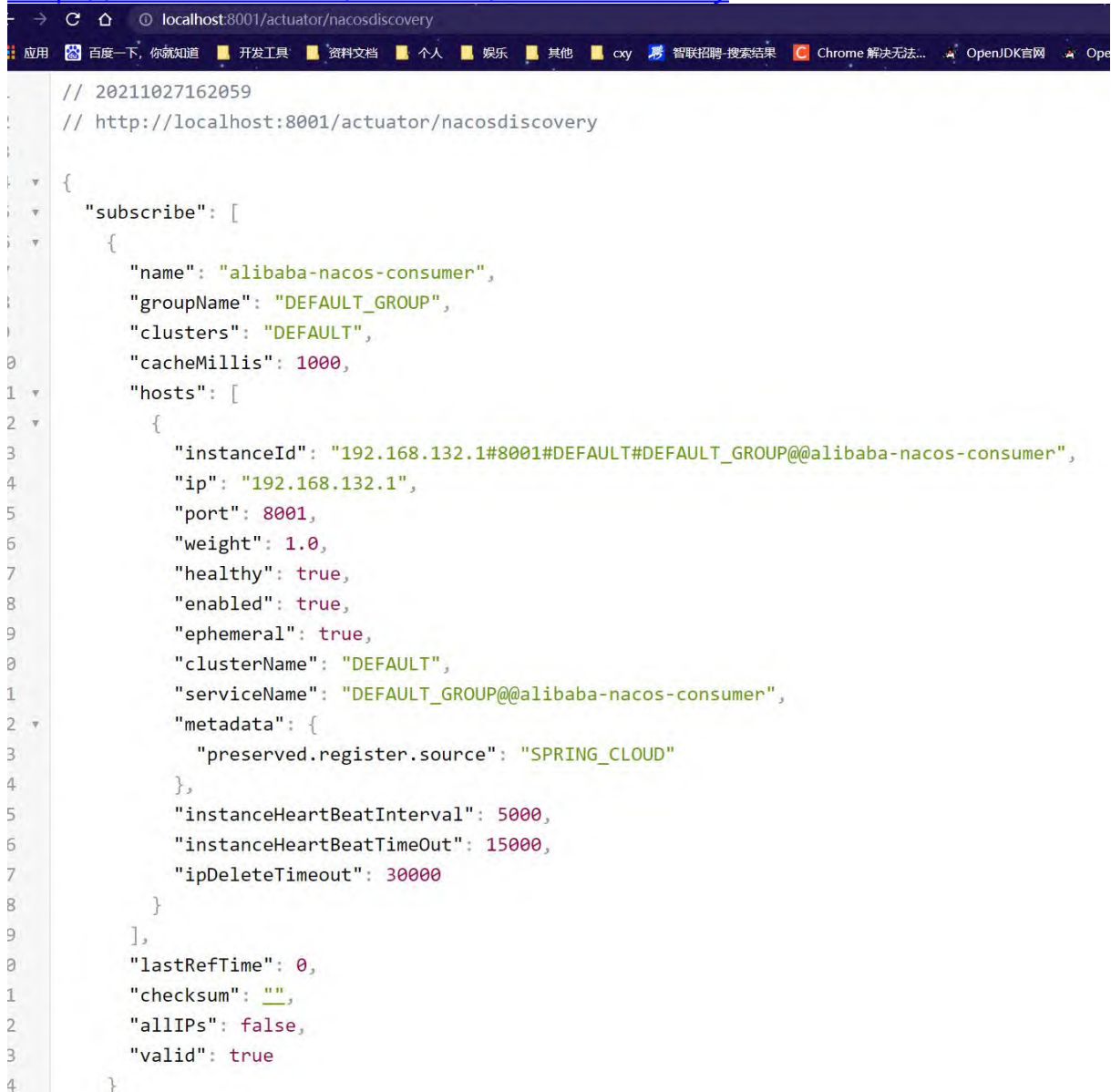
说明:

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

## 6.3 启动项目访问查看效果

<http://localhost:8001/actuator>

<http://localhost:8001/actuator/nacosdiscovery>



```
// 20211027162059
// http://localhost:8001/actuator/nacosdiscovery

{
  "subscribe": [
    {
      "name": "alibaba-nacos-consumer",
      "groupName": "DEFAULT_GROUP",
      "clusters": "DEFAULT",
      "cacheMillis": 1000,
      "hosts": [
        {
          "instanceId": "192.168.132.1#8001#DEFAULT#DEFAULT_GROUP@@alibaba-nacos-consumer",
          "ip": "192.168.132.1",
          "port": 8001,
          "weight": 1.0,
          "healthy": true,
          "enabled": true,
          "ephemeral": true,
          "clusterName": "DEFAULT",
          "serviceName": "DEFAULT_GROUP@@alibaba-nacos-consumer",
          "metadata": {
            "preserved.register.source": "SPRING_CLOUD"
          },
          "instanceHeartBeatInterval": 5000,
          "instanceHeartBeatTimeOut": 15000,
          "ipDeleteTimeout": 30000
        }
      ],
      "lastRefTime": 0,
      "checksum": "",
      "allIPs": false,
      "valid": true
    }
  ]
}
```

## 7.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.server-addr	无	Nacos Server 启动监听的 ip 地址和端口
服务名	spring.cloud.nacos.discovery.service	\${spring.application.name}	给当前的服务命名
服务分组	spring.cloud.nacos.discovery.group	DEFAULT_GROUP	设置服务所处的分组
权重	spring.cloud.nacos.discovery.weight	1	取值范围 1 到 100, 数值越大, 权重越大
网卡名	spring.cloud.nacos.discovery.network-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.ip	无	优先级最高
注册的端口	spring.cloud.nacos.discovery.port	-1	默认情况下不用配置, 会自动探测
命名空间	spring.cloud.nacos.discovery.namespace	无	常用场景之一是不同的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.access-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.secret-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.metadata	无	使用 Map 格式配置, 用户可以根据自己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.log-name	无	
集群	spring.cloud.nacos.discovery.cluster-name	DEFAULT	配置成 Nacos 集群名称
接入点	spring.cloud.nacos.discovery.endpoint	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.watch.enabled	true	可以设置成 false 来关闭 watch

