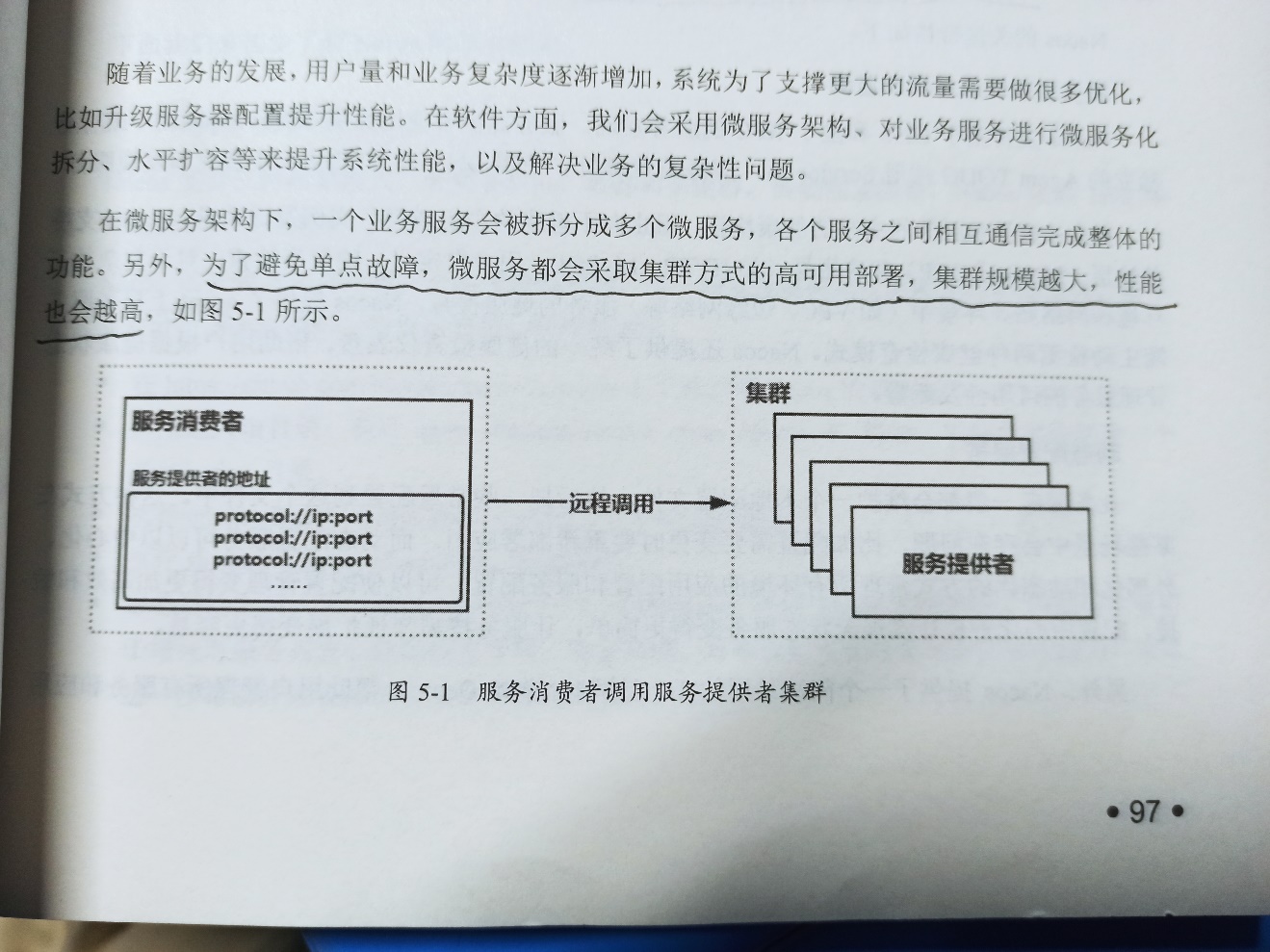
# 微服务与注册中心

随着业务的发展，用户量和业务复杂度逐渐增加，系统为了支撑更大的流量需要做很多优化，比如在硬件方面，升级服务器配置提升性能；在软件方面，采用微服务架构、对业务服务进行微服务化拆分、水平扩容等来提升系统性能，以及解决业务的复杂性问题。

在微服务架构下，一个业务服务会被拆分成多个微服务，各个服务之间相互通信完成整体的功能。另外，为了避免单点故障，微服务会采用集群方式的高可用部署，集群规模越大，性能也越高。如下图：



服务消费者要去调用多个服务提供者组成的集群。首先，服务消费者需要在本地配置文件中维护服务提供者集群的每个节点的请求地址。其次，服务提供者集群中如果某个节点下线或者宕机，服务消费者的本地配置中需要同步删除这个节点的请求地址，防止请求发送到已宕机的节点上造成请求失败。为了解决这类问题，就需要引入服务注册中心，它主要有以下功能：

·服务地址的管理

·服务注册

·服务动态感知

能够实现这类功能的组件很多，比如ZooKeeper、Eureka、Consul、Etcd、Nacos等。

# 认识Alibaba Nacos

## 什么是Alibaba Nacos

Nacos致力于解决微服务中的统一配置、服务注册与发现等问题。它提供了一组简单易用的特性集，帮助开发者快速实现动态服务发现、服务配置、服务元数据及流量管理等。

## Nacos官网

Nacos官网为：

<https://nacos.io/zh-cn/docs/what-is-nacos.html>

并且官网是中文版的，值得参考学习。

## Nacos关键特性

Nacos的关键特性如下：

1. **服务发现和服务健康监测**

Nacos支持基于DNS和基于RPC的服务发现。服务提供者使用原生SDK、OpenAPI或一个独立的Agent TODO注册Service后，服务消费者可以使用DNS或HTTP&API查看和发现服务。

Nacos提供对外服务的实时的健康检查，阻止向不健康的主机或服务实例发送请求。Nacos支持传输层（PING或TCP）和应用层（如HTTP、MySQL、用户自定义）的健康检查。对于复杂的云环境和网络拓扑环境中（如VPC、边缘网络）服务的健康检查，Nacos提供了agent上报和服务端主动检测两种健康检查模式。Nacos还提供了统一的健康检查仪表盘，帮助用户根据健康状态管理服务的可用性及流量。

1. **动态配置服务**

业务服务一般都会维护一个本地配置文件，然后把一些常量配置到这个文件中。这种方法在某些场景中会存在问题，比如配置需要变更时要重新部署应用。而动态配置服务可以以中心化、外部化和动态化的方式管理所有环境的应用配置和服务配置，可以使配置管理变得更加高效和敏捷。配置中心化管理让实现无状态服务变得更简单，让服务按需弹性扩展变得更容易。另外，Nacos提供了一个简洁易用的UI帮助用户管理所有服务和应用的配置。Nacos还提供了包括配置版本跟踪、金丝雀发布、一键回滚配置及客户端配置更新状态跟踪在内的一系列开箱即用的配置管理特性，帮助用户更安全地在生产环境中管理配置变更，降低配置变更带来的风险。

1. **动态DNS服务**

动态DNS服务支持权重路由，让开发者更容易地实现中间层负载均衡、更灵活的路由策略、流量控制，以及数据中心内网的简单DNS解析服务。

**4、服务及其元数据管理**

Nacos可以使开发者从微服务平台建设的视角管理数据中心的所有服务及元数据，包括管理服务的描述、生命周期、服务的静态依赖分析、服务的健康状态、服务的流量管理、路由及安全策略、服务的SLA及最重要的metrics统计数据。

本书主要围绕Nacos中注册中心的特性以及动态配置服务的特性进行讲解。

# Nacos的基本使用

## 安装Nacos

Nacos支持三种部署模式，分别是单机、集群和多集群。需要注意，Nacos依赖Java环境，并且要求使用JDK1.8以上版本。Nacos的安装方式有两种，一种是源码安装，另一种是直接使用已经编译好的安装包。由于后续要分析Nacos源码，所以选择第一种安装方式。

·在<https://github.com/alibaba/nacos/release>上下载Nacos，书上使用的1.1.4版本

·传送到Linux中，进行解压后进入根目录，执行mvn -Prelease-nacos clean install

-U命令构建，构建完成之后会创建一个distribution目录。

·进入distrubution/target/nacos-server-$version/nacos/bin目录中

·执行startup.sh -m standalone命令启动服务

·服务启动后，可以通过<http://127.0.0.1:8848/nacos>访问Nacos的控制台。控制台主要

用增强对服务列表、健康状态管理、服务治理、分布式配置管理等方面的管控能力，可

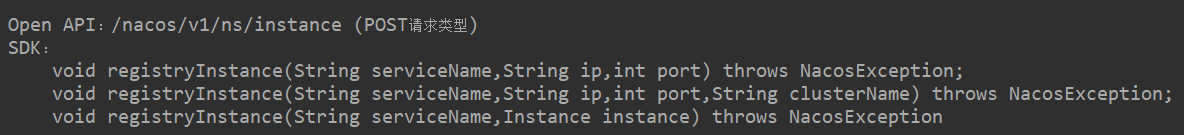
以进一步帮助开发者降低管理微服务应用架构的成本（默认用户名密码为nacos）。

## Nacos服务注册发现相关的API说明

Nacos提供了SDK及Open API的方式来完成服务的注册与发现等操作，由于服务端只提供了REST接口，所以SDK本质上是对HTTP请求的封装。下面简单列一下服务注册相关的核心接口。

1. **注册实例**

将服务地址信息注册到Nacos Server：



参数说明：

·serviceName：参数名称

·ip：服务实例

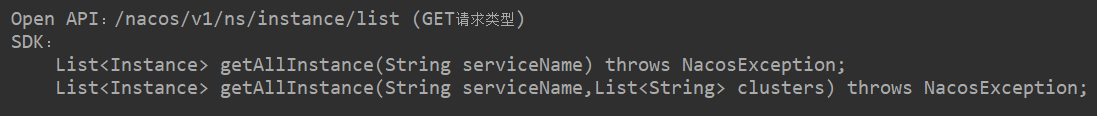
·port：服务实例端口

·clusterName：集群名称，表示该服务实例属于哪个集群

·instance：实例属性，实际上就是把上面这些参数封装成一个对象。

1. **获取全部实例**

根据服务名称从Nacos Server上获取所有服务实例



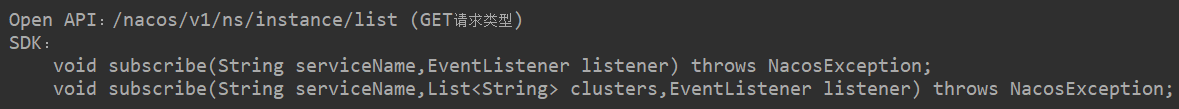
参数说明如下：

·serviceName：服务名称

·cluster：集群列表，可以传递多个值

1. **监听服务**

监听服务是指监听指定服务下的实例变化。客户端从Nacos Server上获取的实例必须是简况的，否则会造成客户端请求失败。监听机制可以让客户端及时感知服务提供者实例的变化。



参数说明如下：

·EventListener：当服务提供者实例发生上、下线时，会触发一个事件回调。

需要注意的是，监听服务的Open API也访问/nacos/v1/ns/instance/list，具体原理会在源码分析中讲解。服务监听有两种方式：

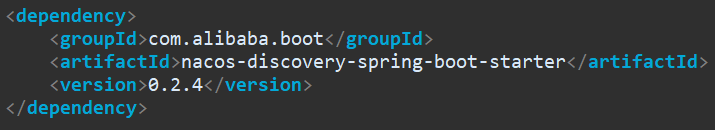
·第一种是客户端调用/nacos/v1/ns/instance/list定时轮询

·第二种是基于DatagramSocket的UDP协议，实现服务端的主动推送

以上只列举了一小部分API，更多API请参考Nacos官网。

## Nacos集成Spring Boot实现服务注册与发现

1. 创建一个名为spring-boot-nacos-discovery的Spring Boot工程
2. 添加Maven依赖



1. 创建DiscoveryController类，通过@NacosInjected注入Nacos的NamingService，并提供discovery方法，可以根据服务名称获得注册到Nacos上的服务地址。



1. 在application.properties中添加Nacos服务地址的配置。



1. 启动Spring Boot项目，调用

curl http://127.0.0.1:8080/discovery?serviceName=null

去Nacos服务器上查看example的服务实例，结果会返回一个空的JSON数组。因为Nacos服务器上没有名为的example的服务。

1. 接着，通过Nacos提供的Open API，向Nacos Server注册一个名字为example的服务

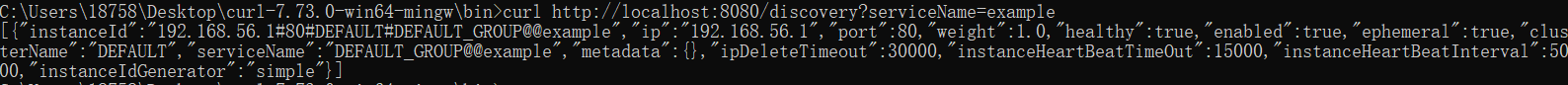
curl -X PUT 'http://127.0.0.1:8848/nacos/v1/ns/instance?serviceName=example&ip=192.168.56.1&port=80'

这个命令是在Linux服务上执行的。

1. 再次访问调用

curl http://127.0.0.1:8080/discovery?serviceName=example

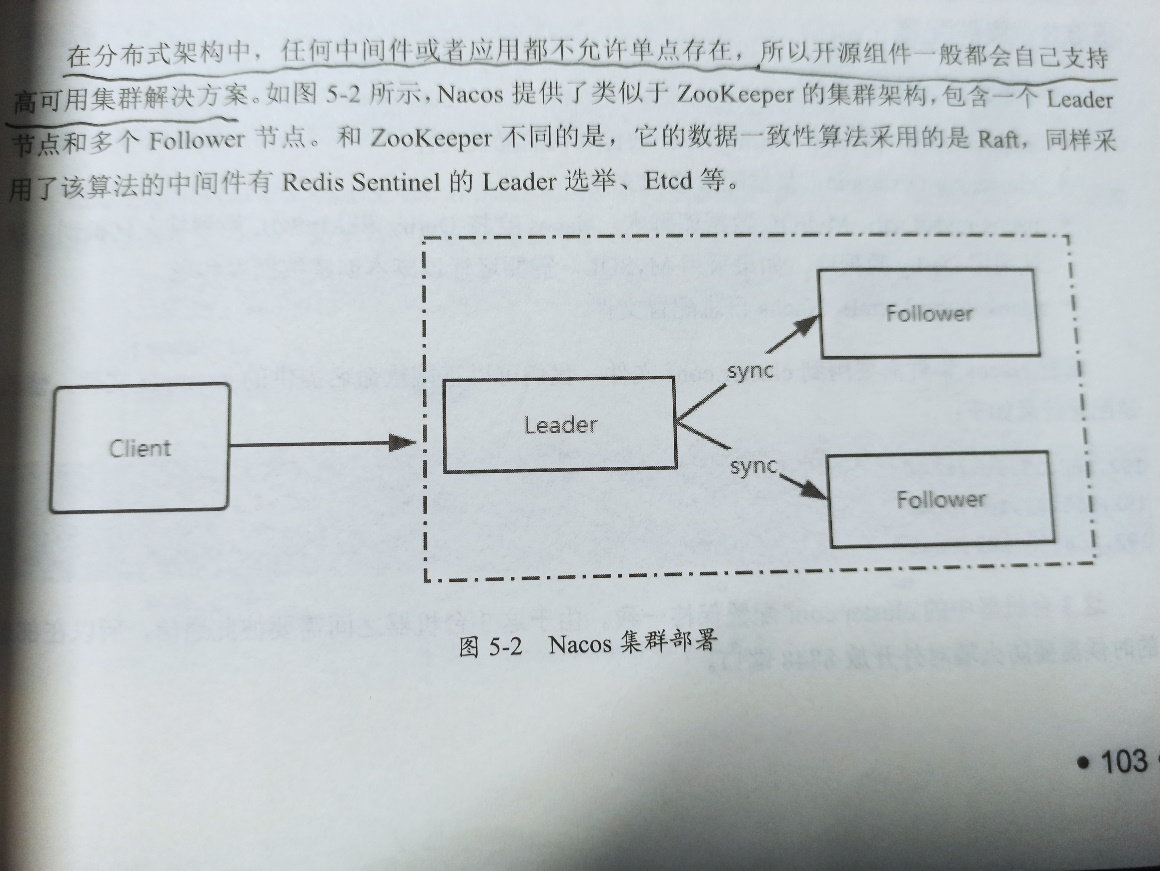
将返回以下信息：



# Nacos的高可用部署

## Nacos集群结构

在分布式架构中，任何中间件或者应用都不允许单点存在，所以开源组件一般都会自己支持高可用集群解决方案。Nacos提供了类似于ZooKeeper的集群结构，如下所示：



它包含一个Leader节点和多个Follower节点。和ZooKeeper不同的是，它的数据一致性算法采用的是Raft，同样采用了该算法的中间件有Redis Sentinel的Leader选举、Etcd等。

## 安装环境要求

请确保在环境中安装使用：

·64 bit OS Linux/UNIX/Mac，推荐使用Linux

·64 bit JDK1.8及以上，下载并配置

·Maven 3.2.x及以上，下载并配置

·MySQL数据库

需要注意：3个或3个以上的Nacos节点才能构成集群。

## 安装包及环境准备

因为3个或3个以上的Nacos节点才能构成集群，因此，准备3台服务器，作者采用的是CentOS7.x系统。

1. 下载nacox安装包，分别进行解压：tar -zxvf nacos-server-1.1.4.tar.gz
2. 解压后会得到5个文件夹：bin（服务启动/停止脚本）、conf（配置文件）、logs（日志）、data（derby数据库存储）、target（编译打包后的文件）。

## 集群配置

在conf目录下包含以下文件：

·application.properties：Spring Boot项目默认的配置文件。

·cluster.conf.example：集群配置样例文件

·nacos-mysql.sql：MySQL数据库脚本。Nacos支持Derby和MySQL两种持久化机

制，默认采用Derby数据库。如果采用MySQL，需要运行该脚本创建数据库和表。

·nacos-logback.xml：Nacos日志配置文件

配置Nacos集群需要用到cluster.conf文件，我们可以直接重命名提供的

cluster.conf.example文件，修改该配置信息如下：



这3台机器中的cluster.conf配置保持一致。由于这3台机器之间需要彼此通信，所以在部署的时候需要防火墙对外开放8848端口。

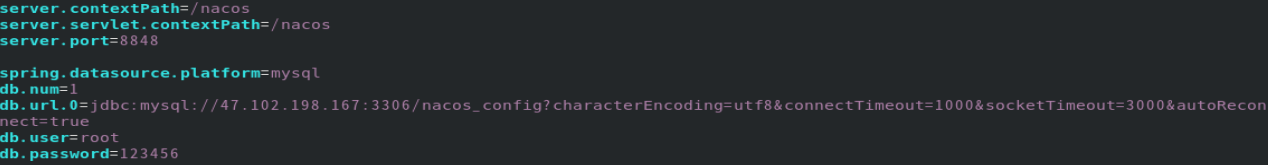
## 配置MySQL数据库

Derby数据库是一种文件类型的数据库，在使用时会存在一定的局限性，比如它无法支持多用户同时操作，在数据量大、连接较多的情况下会产生大量连接的积压。所以在生产环境中，可以用MySQL替换。

·执行${NACOS\_HOME}/conf下的nacos-mysql.sql文件

·分别修改3台机器中的${NACOS\_HOME}/conf下的application.properties文件，增加

MySQL的配置。



需要注意：Nacos默认采用5.1版本的数据库驱动包，如果你指定的MySQL版本为8.0及以上的，则启动Nacos服务会失败。有两种解决方案：

1. 使用5.x版本的数据库
2. 下载Nacos的源码包，将源码中的MySQL驱动包的版本改为8.x版本。

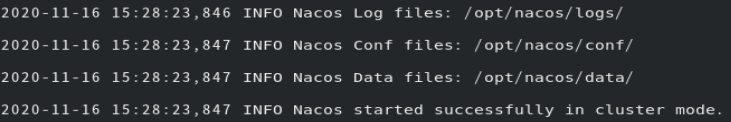
答案来源于：

<https://blog.csdn.net/hitler698/article/details/88017610>

## 启动Nacos服务

分别进入3台机器的bin目录，执行sh .startup.sh命令启动服务，服务器启动之后，在

${NACOS\_HOME}\logs\start.out文件中可以获得如下日志，则表示服务启动成功。



## 查看集群节点

通过http:${NACOS\_CLUSTER\_IP}:8848/nacos访问Nacos控制台，在“节点列表”下可以看到如下的内容：



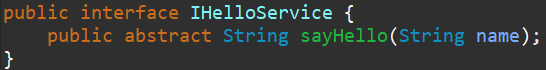
表示当前集群由哪些节点组成及节点的状态。

# Dubbo使用Nacos实现注册中心

## 服务提供者

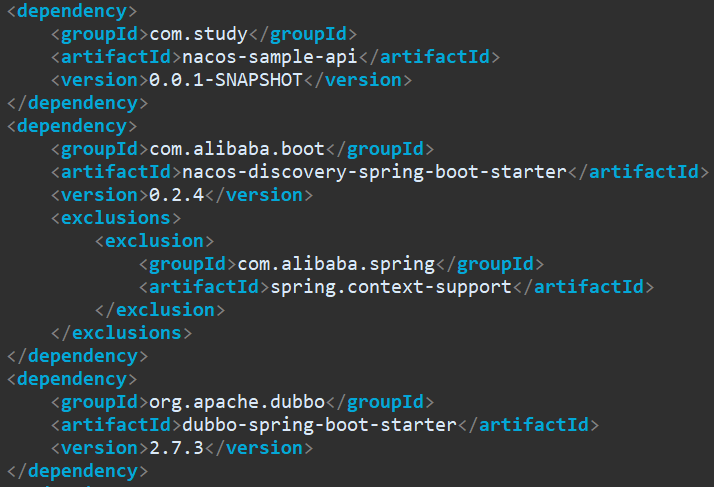
Dubbo可以支持多种注册中心，如ZooKeeper、以及Consul、Nacos等，接下来讲解如何使用Nacos作为Dubbo服务的注册中心，为Dubbo提供服务注册与发现的功能。

1. 创建一个Maven项目，spring-boot-dubbo-nacos-sample，添加两个模块：nacos-sample-api和nacos-sample-provider。其中，nacos-sample-provider是一个Spring Boot工程。将Maven项目执行mvn install打包到仓库，供nacos-sample-provider依赖。
2. 在nacos-sample-api模块中定义接口



将该模块打包到仓库，供其他模块依赖。

1. 在nacos-sample-provider模块中添加依赖

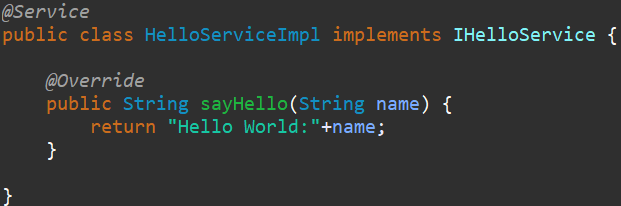


nacos-sample-api：nacos-sample-api接口依赖。

nacos-discovery-spring-boot-starter：Nacos的Starter组件

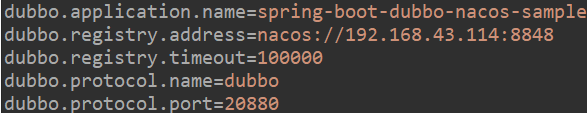
dubbo-spring-boot-starter：Dubbo的Starter组件

1. 定义接口实现



注意：@Service是Dubbo提供的注解。

1. 编写application.properties文件



dubbo.application.name：用来描述服务提供方的应用名称

dubbo.registry.address：注册中心的地址，如果不需要注册中心，可以设置为N/A。 Dubbo支持多种注册中心，比如ZooKeeper、Nacos等。nacos

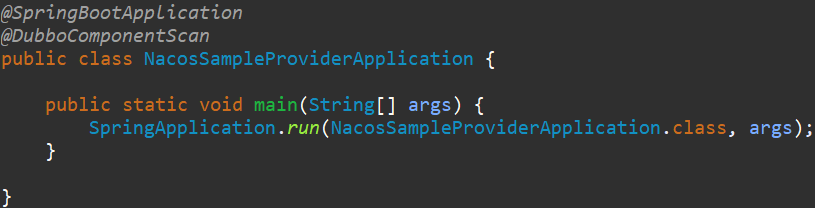
开头表示使用Nacos作为注册中心。

dubbo.registry.timeout： 超时时间

dubbo.protocol.name：使用Dubbo协议来发布服务。

dubbo.protocol.port：指定端口。

1. 在项目启动类上标注@DubboComponentScan注解



7）启动服务，访问Nacos控制台，在服务列表中可以看到



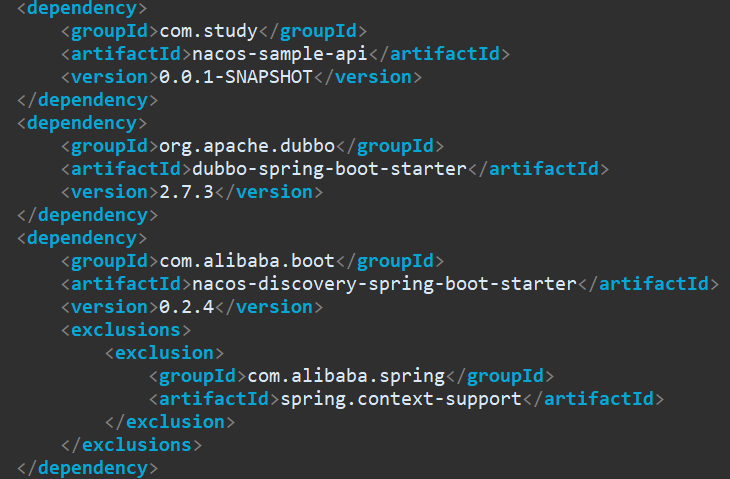
在“操作”列单击“详情”，可以看到IHelloService下所有服务提供者的实例元数

句。如下：



## 服务消费者

1. 创建一个名为nacos-test的Spring Boot项目，并添加以下依赖

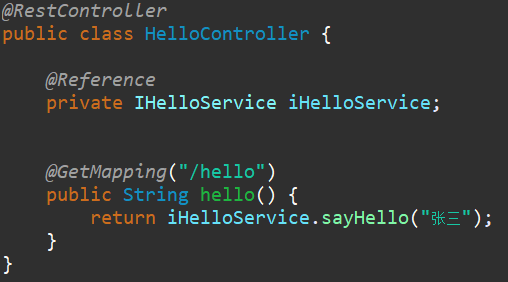


nacos-sample-api：服务提供者提供的API接口

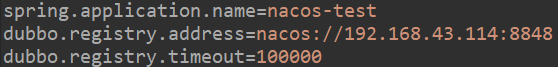
dubbo-spring-boot-starter：Dubbo的starter组件

nacos-discovery-spring-boot-starter：Nacos的starter组件

1. 调用服务提供者提供的服务



1. 在application.properties中进行相关配置



spring.application.name：应用名称

dubbo.registry.address：注册中心的地址

dubbo.registry.timeout：超时时间

1. 启动服务，测试



# Spring Cloud Nacos Discovery

Nacos作为Spring Cloud Alibaba中服务注册与发现的核心组件，可以很好地开发者将服务注册到Nacos服务器，并且能够动态感知和刷新某个服务实例的服务列表。使用Spring Cloud Alibaba Nacos Discovery可以基于Spring Cloud规范快速接入Nacos，实现服务注册与发现。

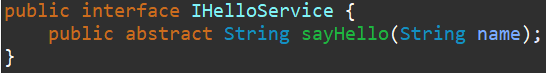
## 服务端开发

1. 创建一个普通的Maven项目名为spring-cloud-nacos-sample，基于该项目创建两个模块：

·spring-cloud-nacos-sample-api：暴露服务接口

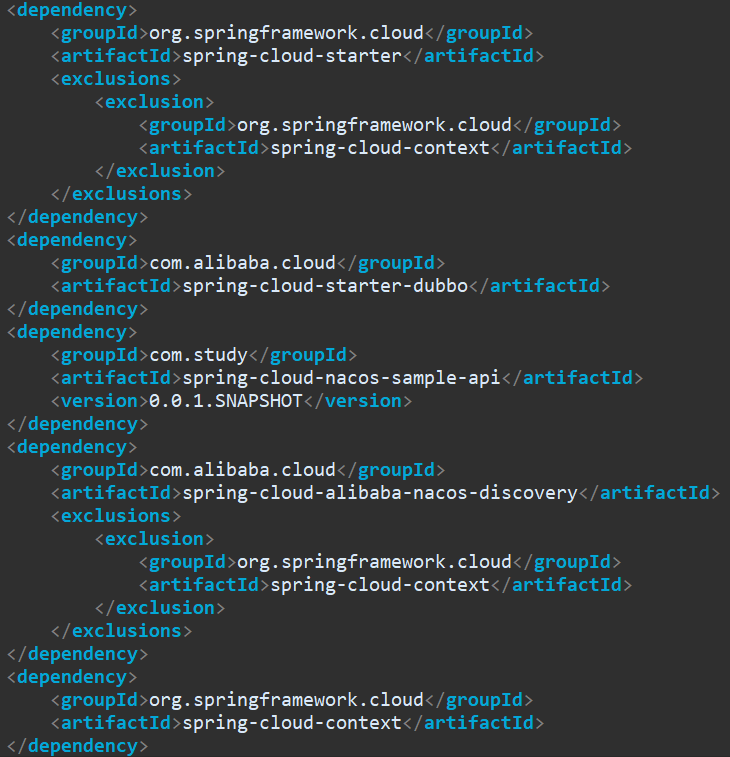
·spring-cloud-nacos-sample-provider：项目类型为Spring Boot，它是接口的实现

1. 在spring-cloud-nacos-sample-api模块中定义一个接口



打包到仓库供其他模块或项目依赖。

1. 在spring-cloud-nacos-sample-provider模块的pom.xml文件中添加相关依赖



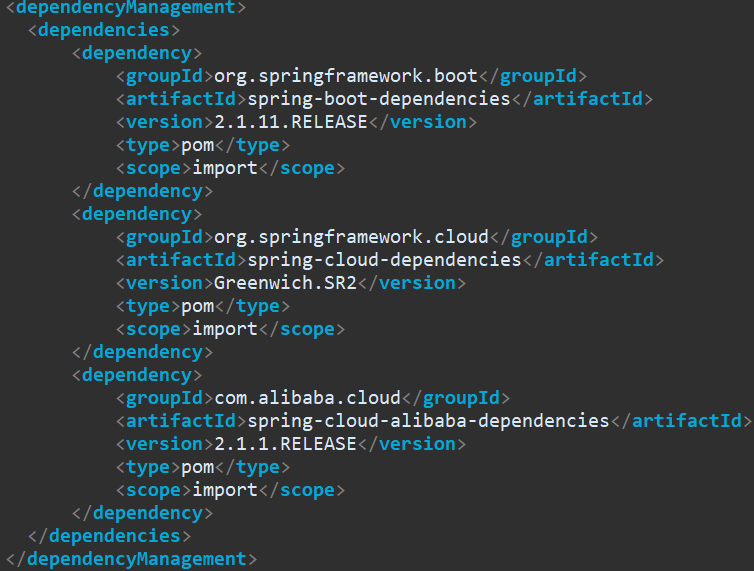
spring-cloud-starter：Spring Cloud核心包

spring-cloud-nacos-sample-api：接口依赖

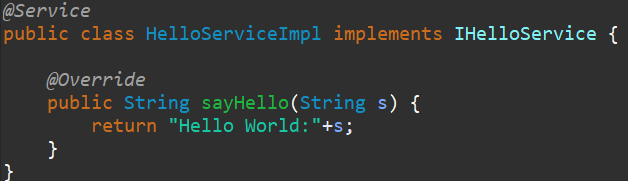
spring-cloud-alibaba-nacos-discovery：基于Nacos的服务注册与发现

spring-cloud-context：spring-cloud-alibaba-nacos-discovery传递的依赖

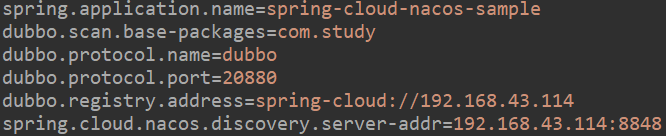
需要注意，笔者使用的版本中，spring-cloud-starter传递依赖的spring-cloud-context为2.2.1.RELEASE，但这个版本的包存在兼容问题，会出现java.lang.ClassNotFoundException异常，所以在spring-cloud-alibaba-nacos-discovery中排除了，并引入了其他版本的来解决。上述依赖都没有指定版本，所以需要在父pom.xml中显式地声明版本。



1. 在spring-cloud-nacos-sample-provider中创建接口的实现类HelloServiceImpl，其中@Service是Dubbo服务的注解，表示当前服务会发布成一个远程服务。



1. 在application.properties中提供Dubbo以及Nacos的配置，用于声明Dubbo服务暴露的网络端口和协议，以及服务注册的地址信息。



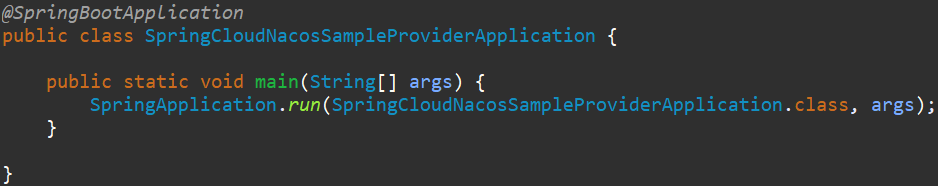
dubbo.scan.base-package：功能等同于@DubboComponentScan，指定Dubbo服务实

现类的扫描包路径。

dubbo.registry.address：Dubbo服务注册中心配置地址，它的值spring-cloud://192.168.43.114表示加载到Spring Cloud注册中心，不配置的话会提示没有配置注册中心的错误。

Spring.cloud.nacos.discovery.server-addr：Nacos服务注册中心的地址

1. 启动服务

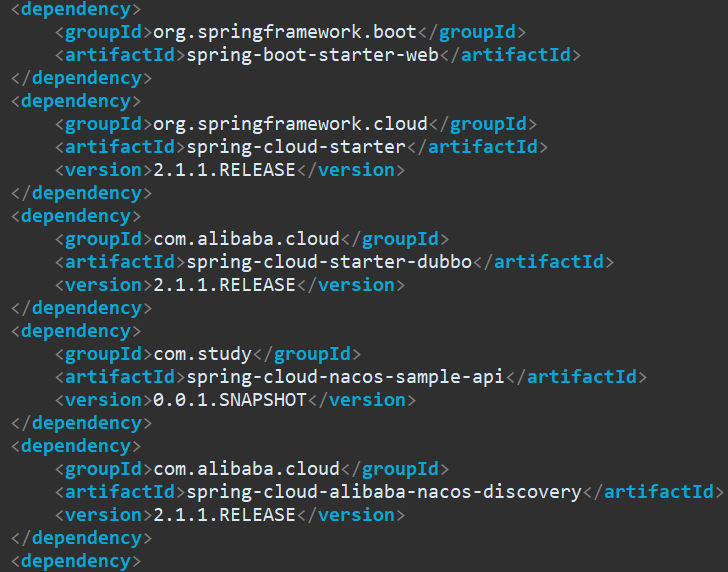


启动完成后，进入Nacos控制台的“服务管理”-->“服务列表”，如果看到服务了，则说明服务已经发布成功了。如下：

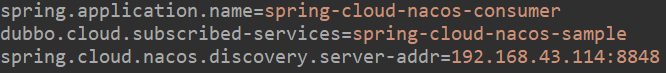


## 消费端开发

1. 创建一个Spring Boot项目spring-cloud-nacos-consumer，并添加相关依赖



1. 在application.properties中添加配置信息

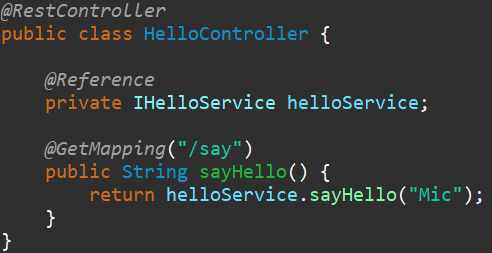


dubbo.cloud.subscribed-services：表示服务调用者订阅的服务提供方的应用名称列表，如果有多个名称，可以通过“，”分隔开，默认为“\*”，不推荐使用默认值。使用默认值，控制台日志会输入一段警告信息。

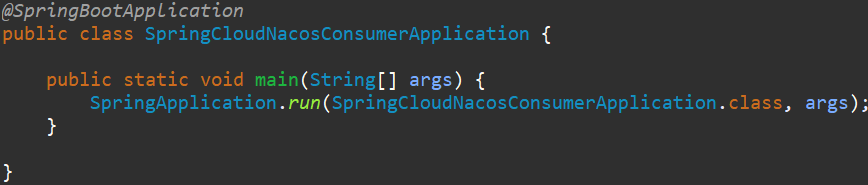
dubbo.scan.base-package：功能等同于@DubboComponentScan，指定Dubbo服务实现类的扫描路包路径。

spring-cloud.nacos.discovery.server-addr：Nacos服务注册中心的地址。

1. 定义HelloController，使用提供的服务



1. 启动服务



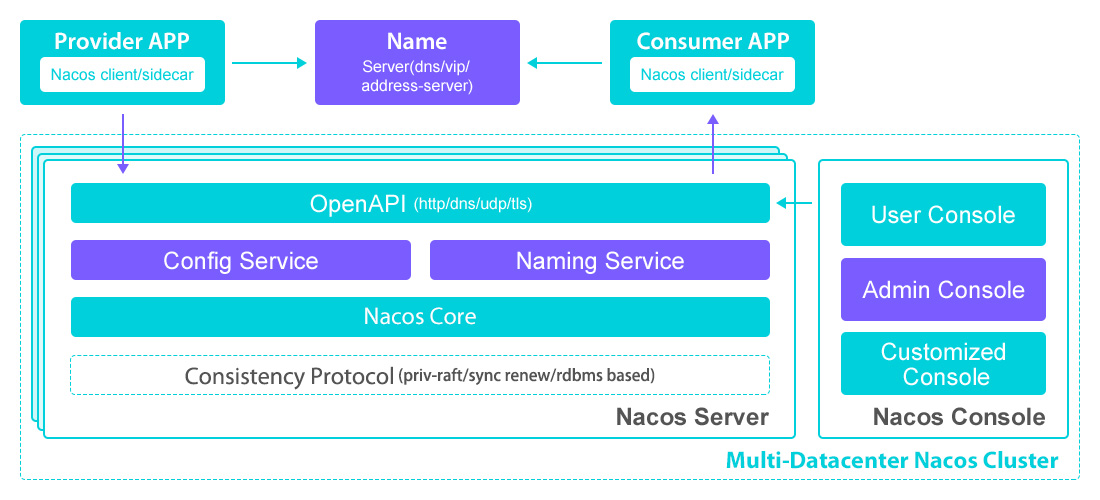
通过curl命令执行HTTP GET方法，查看响应结果：



# Nacos实现原理分析

## Nacos结构图

下图是Nacos官方提供的架构图：



·Provider APP：服务提供者

·Consumer APP：服务消费者

·Name Server：通过VIP（Vritual IP）或者DNS方式实现Nacos高可用集群的服务

路由

·Nacos Server：Nacos服务提供者，里面包含的Open API是功能访问入口，Config

Services是Nacos提供的配置服务模块、Naming Service是Nacos提供的名字服务

模块。Consistency Protocol是一致性协议，用来实现Nacos集群节点的数据同步，

这是使用的Raft算法（使用类似算法的中间件还有Etcd、Redis哨兵选举）

·Nacos Console：Nacos控制台

整体来说，服务提供者通过VIP（Virtual IP）访问Nacos Server高可用集群，基于Open API完成服务的注册和服务的查询。Nacos Server本身可以支持主备模式，所以底层会采用数据一致性算法来完成从节点的数据同步。服务消费者也是如，基于Open API从Nacos Server中查询服务列表。

## 服务注册中心的原理

服务注册的功能主要体现在：

·服务实例在启动时注册到服务注册表，并在关闭时注销。

·服务消费者查询服务注册表，获得可用实例。

·服务注册中心需要调用服务实例的健康检查API来验证它是否能够处理请求。

Nacos服务注册与发现的实现原理如下图：

