## 一站式微服务解决方案Spring Cloud Alibaba

主讲：Cat 老师

北京动力节点教育科技有限公司  
2020 • 北京

动力节点•版权所有•禁止传播

## What is Spring Cloud Alibaba？

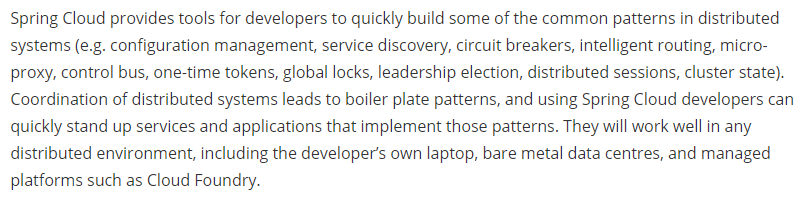
首先我们需要了解一下Spring Cloud，然后再来了解Spring Cloud Alibaba；

### Spring Cloud

官网：<https://spring.io/projects/spring-cloud>

当前最新版本：Hoxton SR6

源自官方描述:



Spring Cloud为开发人员提供了一些工具用来快速构建分布式系统中的一些常见模式和解决一些常见问题（例如配置管理、服务发现、断路器、智能路由、微代理、控制总线、一次性令牌、全局锁、领导选举、分布式会话、群集状态）。分布式系统的协调导致了很多样板式的代码（很多固定套路的代码），使用Spring Cloud开发人员可以快速建立实现这些模式的服务和应用程序。它们在任何分布式环境中都能很好地运行，包括开发人员自己的笔记本电脑、裸机数据中心和云计算等托管平台；

**主要特性：**

Spring Cloud为分布式系统开发的典型应用场景提供良好的开箱即用的功能:

分布式/版本化配置

服务注册和发现

路由

服务到服务的调用

负载均衡

断路器

全局锁

领导选举和集群状态

分布式消息传递

Spring Cloud采用了一种声明性的方式，通常只需更改类路径和/或注解即可实现许多功能特性；

Spring Cloud包含的主要工程项目如下：

Spring Cloud Azure

Spring Cloud Alibaba

Spring Cloud for Amazon Web Services

Spring Cloud Bus

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 Contract

Spring Cloud Function

Spring Cloud Gateway

Spring Cloud GCP

Spring Cloud Netflix

Spring Cloud Open Service Broker

Spring Cloud Pipelines

Spring Cloud Schema Registry

Spring Cloud Security

Spring Cloud Skipper

Spring Cloud Sleuth

Spring Cloud Stream

Spring Cloud Stream Applications

Spring Cloud Stream App Starters

Spring Cloud Task

Spring Cloud Task App Starters

Spring Cloud Vault

Spring Cloud Zookeeper

Spring Cloud App Broker

Spring Cloud Circuit Breaker

Spring Cloud Kubernetes

Spring Cloud OpenFeign

### Spring Cloud Alibaba

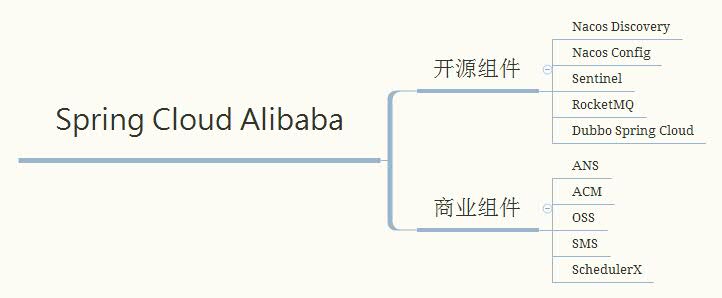
Spring Cloud Alibaba是Spring Cloud下的一个子项目，Spring Cloud Alibaba为分布式应用程序开发提供了一站式解决方案，它包含开发分布式应用程序所需的所有组件，使您可以轻松地使用Spring Cloud开发应用程序，使用Spring Cloud Alibaba，您只需要添加一些注解和少量配置即可将Spring Cloud应用程序连接到Alibaba的分布式解决方案，并使用Alibaba中间件构建分布式应用程序系统；

Spring Cloud Alibaba 是阿里巴巴开源中间件跟 Spring Cloud 体系的融合:



主要特性：

1. 流量控制和服务降级：使用阿里巴巴Sentinel进行流量控制，断路和系统自适应保护；
2. 服务注册和发现：实例可以在Alibaba Nacos上注册，客户可以使用Spring管理的bean发现实例，通过Spring Cloud Netflix支持Ribbon客户端负载均衡器；
3. 分布式配置：使用阿里巴巴Nacos作为数据存储；
4. 事件驱动：构建与Spring Cloud Stream RocketMQ Binder连接的高度可扩展的事件驱动微服务；
5. 消息总线：使用Spring Cloud Bus RocketMQ链接分布式系统的节点；
6. 分布式事务：支持高性能且易于使用的Seata分布式事务解决方案；
7. Dubbo RPC：通过Apache Dubbo RPC扩展Spring Cloud服务到服务调用的通信协议；



### Spring Cloud alibaba版本对应

目前最新版本：Spring Cloud Alibaba 2.2.1

Spring Cloud Alibaba 2.1.0 RELEASE对应Spring Cloud Greenwich版本

Spring Cloud Alibaba 2.2.0 RELEASE对应Spring Cloud Hoxton.RELEASE版本

Spring Cloud Alibaba 2.2.1 RELEASE对应Spring Cloud Hoxton.SR3版本

## What is Nacos？



Nacos是阿里巴巴2018年7月推出来的一个开源项目，是一个更易于构建云原生应用的动态服务注册与发现、配置管理和服务管理平台；（Nacos：纳科斯）

Nacos致力于快速实现动态服务注册与发现、服务配置、服务元数据及流量管理；

Nacos 属于Spring cloud alibaba下的一个组件；

Nacos 约等于 spring cloud eureka（注册中心）+ spring cloud config（配置中心）

Nacos官网：<https://nacos.io/>

## Nacos运行环境部署

1、下载nacos最新的二进制压缩包；

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

2、解压下载下来的nacos最新的二进制压缩包；

tar -zxvf nacos-server-1.3.1.tar.gz  
cd nacos/bin

3、启动nacos server

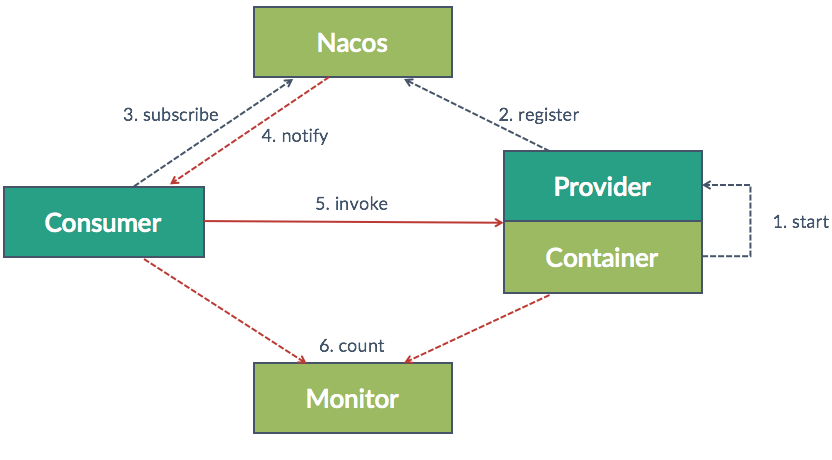
./startup.sh -m standalone

注：单机环境必须带-m standalone参数启动，否则无法启动，不带参数启动的是集群环境；

4、启动日志：/usr/local/nacos/logs/start.out

5、访问：<http://192.168.172.128:8848/nacos> 默认用户名密码：nacos/nacos

## Spring Cloud Alibaba Nacos Discovery服务注册发现



和Dubbo类似，微服务开发是controller调用controller，调用者是服务消费者，被调用者是服务提供者，服务消费者和服务提供者是相对概念，服务消费者也可以被另一个服务调用，那么此时的服务消费者也是一个服务提供者；

在实际开发中，我们会把所有服务都注册到nacos注册中心上，由nacos去维护和管理我们的所有服务；

通过添加一个starter依赖：spring-cloud-starter-alibaba-nacos-discovery它通过自动配置、注解以及Spring Boot 编程模型与Nacos无缝集成，实现服务注册与发现，nacos经过双十一考验，可以在生产环境中作为大规模分布式系统的服务注册中心；

### 实现服务提供者

1、首先修改 pom.xml 文件，引入Nacos Discovery Starter；

<dependency>

<groupId>com.alibaba.cloud</groupId>

<artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>

</dependency>

2、在应用的/src/main/resources/application.properties(也可以是application.yaml) 配置文件中配置 Nacos Server 地址；

spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848

如果不想使用Nacos作为您的服务注册与发现，可以将 spring.cloud.nacos.discovery.enabled 设置为 false；

3、使用 @EnableDiscoveryClient 注解开启服务注册与发现功能；

@SpringBootApplication

@EnableDiscoveryClient

public class ProviderApplication {

public static void main(String[] args) {

SpringApplication.run(ProviderApplication.class, args);

}

}

此时就可以在Nacos的web管控台上看到注册上来的服务信息了；

具体细节可以参考视频和代码；

### 实现服务消费者

消费者应用比提供者应用要稍微复杂一点，因为在消费端需要去调用提供者提供的REST 服务，此时需要用到RestTemplate、Feign及负载均衡Ribbon，这几个组件我们第一期课程中学习过，如果不了解可以回顾一下第一期微服务专题SpringCloud课程；

服务消费者的具体细节可以参考视频和代码；

### 服务EndPoint端点

每一个采用spring-cloud-starter-alibaba-nacos-discovery实现的服务都提供了一个EndPoint，EndPoint的访问地址为

http://ip:port/actuator/nacos-discovery

EndPoint主要提供了两类信息:

1、subscribe: 显示了当前有哪些服务订阅者；

2、NacosDiscoveryProperties: 显示当前服务实例关于Nacos的基础配置；

## Spring Cloud Alibaba Nacos Config配置中心

Nacos提供用于存储配置和其他元数据功能，为分布式系统中的外部化配置提供服务器端和客户端支持，使用Spring Cloud Alibaba Nacos Config就可以在Nacos Server集中管理Spring Cloud应用的外部属性配置；

Spring Cloud Alibaba Nacos config是在启动的bootstrap阶段，将配置加载到Spring环境中；

Spring Cloud Alibaba Nacos Config使用DataId和GROUP确定一个配置；

启动好Nacos之后，在Nacos添加如下的配置：

Data ID: nacos-config.properties

Group : DEFAULT\_GROUP

配置格式: Properties

配置内容：

user.name=zhangsan

user.age=18

DataId是以.properties为扩展名（默认的文件扩展名方式就是.properties）；

在运行时必须使用 bootstrap.properties配置文件配置nacos server地址，同时，spring.application.name值必须与Nacos中配置的Data Id匹配（除了.properties或者.yaml后缀）；

DataId默认使用`spring.application.name`配置跟文件扩展名结合(配置格式默认使用.properties)，GROUP不配置默认使用DEFAULT\_GROUP；

注意当你使用域名的方式来访问Nacos时，

spring.cloud.nacos.config.server-addr 配置的方式为 域名:port，例如 Nacos的域名为nacos.power.com，监听的端口为80，

则 spring.cloud.nacos.config.server-addr=nacos.power.com:80,注意80 端口不能省略；

如果要在项目中使用Nacos来实现应用的外部化配置，需要添加如下依赖：

<dependency>

<groupId>com.alibaba.cloud</groupId>

<artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>

</dependency>

具体细节可以参考视频和代码；

## Nacos Config配置中心动态刷新

Nacos Config Starter 默认为所有获取数据成功的 Nacos 的配置项添加了监听功能，在监听到服务端配置发生变化时会实时触发 org.springframework.cloud.context.refresh.ContextRefresher的refresh 方法；

可以通过配置 spring.cloud.nacos.config.refresh.enabled=false 来关闭动态刷新；

具体细节可以参考视频和代码；

## 基于DataId为yam扩展名配置方式

Nacos Config 除了支持.properties格式以外，也支持yaml格式，这个时候只需要完成以下两步：

1、在应用的bootstrap.properties配置文件中显式地声明DataId文件扩展名；bootstrap.properties文件配置如下：

spring.cloud.nacos.config.file-extension=yaml

2、在Nacos的web管控台新增一个DataId为yaml扩展名的配置，如下所示：

Data ID: nacos-config.yaml

Group : DEFAULT\_GROUP

配置格式: YAML

配置内容: user.name: zhangcuishan

user.age: 35

这两步完成后，重启程序测试；

具体细节可以参考视频和代码；

## 基于profile粒度的多环境配置

spring-cloud-starter-alibaba-nacos-config 在加载配置的时候，不仅仅加载了以 dataid 为 ${spring.application.name}.${file-extension:properties} 为的基础配置，还加载了dataid为

${spring.application.name}-${profile}.${file-extension:properties} 的基础配置；

在日常开发中如果遇到多套环境下的不同配置，可以通过Spring提供的 ${spring.profiles.active} 配置项来激活使用某个配置文件；

spring.profiles.active=develop

${spring.profiles.active}当通过配置文件来指定时必须放在 bootstrap.properties文件中；

比如在Nacos上新增一个dataid为：nacos-config-dev.yaml的基础配置，如下所示：

Data ID: nacos-config-deve.yaml

Group : DEFAULT\_GROUP

配置格式: YAML

配置内容: current.env: dev-env

启动 Spring Boot 应用测试；

如果需要切换到生产环境，只需要更改 ${spring.profiles.active} 参数配置即可，如下所示：

spring.profiles.active=product

具体细节可以参考视频和代码；