# Spring Cloud学习指南

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 作者 | 备注 |
| V1 | 2019-08-29 | 谭祖好 | 1. 核心概念 2. 注册中心Eureka 3. 客户端负载均衡Ribbon 4. 声明式rest调用Feign 5. Eureka高可用 6. 相关源码：spring-cloud-V1 7. 备注：版本说明.txt |
| V2 | 待续 | 谭祖好 | 1. 熔断处理Hystrix 2. 网关Zuul 3. 配置管理Config 4. 服务跟踪Sleuth 5. 安全认证Security |
| V3 | 待续 | 谭祖好 | 1. spring cloud 集成redis 2. spring cloud 集成rabbitmq |
| V4 | 待续 | 谭祖好 | 1. docker安装和使用 2. 在docker上部署spring cloud项目 |

## Spring Cloud简介

SpringCloud是基于SpringBoot之上的用来快速构建微服务系统的工具集，拥有功能完善的轻量级微服务组件。它为基于JVM的应用开发中涉及的配置管理、服务发现、熔断器、智能路由、微代理、控制总线、分布式会话和集群状态管理等操作提供了一种简单的开发方式。

## Spring Cloud核心子项目

### 2.1、Spring Cloud Netflix：核心组件，可以对多个Netflix OSS开源套件进行整合，包括以下几个组件：

#### 2.1.1、Eureka：服务治理组件，包含服务注册与发现。

#### 2.1.2、Ribbon：客户端负载均衡的服务调用组件。

#### 2.1.3、Feign：基于Ribbon和Hystrix的声明式服务调用组件。

#### 2.1.4、Hystrix：容错管理组件，实现了熔断器。

#### 2.1.5、Zuul：网关组件，提供智能路由、访问过滤等功能。

#### 2.1.6、Archaius：外部化配置组件。

### 2.2、Spring Cloud Config：配置管理工具，实现应用配置的外部化存储，支持客户端配置信息刷新、加密/解密配置内容等。

### 2.3、Spring Cloud Bus：事件、消息总线，用于传播集群中的状态变化或事件，以及触发后续的处理。

### 2.4、Spring Cloud Security：基于spring security的安全工具包，为我们的应用程序添加安全控制。

### 2.5、Spring Cloud Consul : 封装了Consul操作，Consul是一个服务发现与配置工具（与Eureka作用类似），与Docker容器可以无缝集成。

## 3、优缺点

### 3.1、优点

### 3.1.1、每个服务可以独立运行在自己的进程里

### 3.1.2、一系列独立运行的微服务(goods,order,pay,user…)共同构建了整个系统。

### 3.1.3、每个服务为独立的业务开发，一个微服务只关注某个特定的功能，例如用户管理，商品管理微服务。

### 3.1.4、微服务之间通过一些轻量级的通信机制进行通讯，例如通过Restful API进行调用。

### 3.1.5、技术栈不受限:可以使用不同的开发语言和数据存储技术。

### 3.1.6、全自动的部署机制。

### 3.1.7、按需伸缩:根据需求和应用场景，实现细粒度的水平扩展。

### 3.2、缺点

### 3.2.1、运维要求较高

### 3.2.2、分布式的复杂性

### 3.2.3、接口调整成本较高

## 5、设计原则

### 5.1、单一职责原则

### 5.2、服务自治原则

### 5.3、轻量级通讯机制

### 5.4、微服务粒度

## 6、微服务开发框架

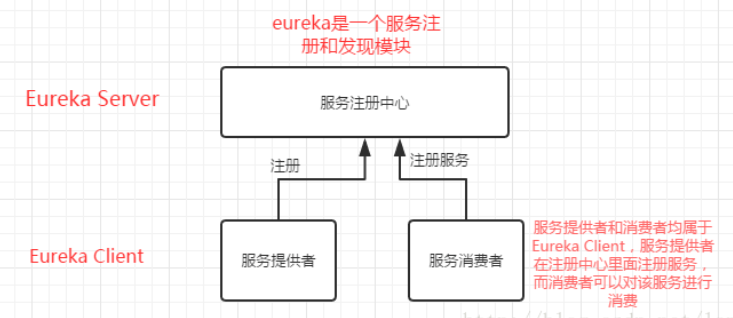
### 6.1、SpringCloud:众多组件构造完善的分布式系统

### 6.2、Dubbo/Dubbox:关注服务治理

### 6.3、Dropwizard:关注单个微服务开发

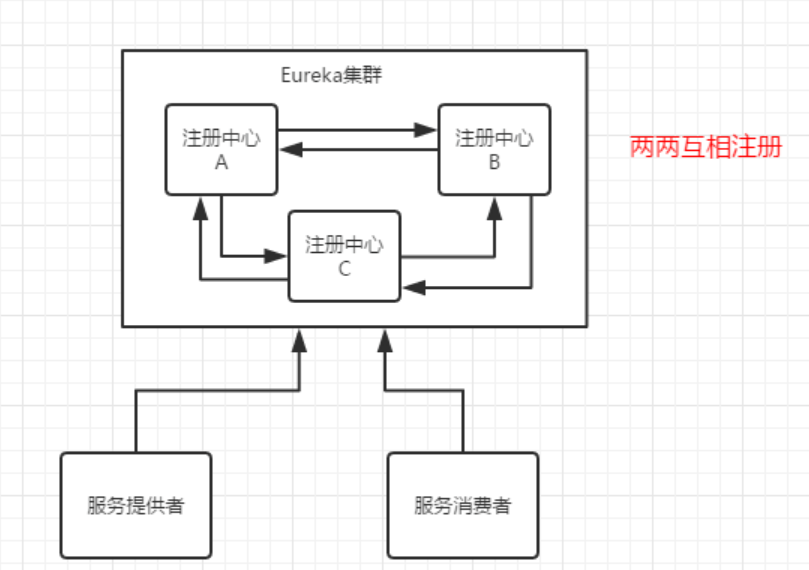
## 7、工作原理图

### 7.1、服务发现——Netflix Eureka



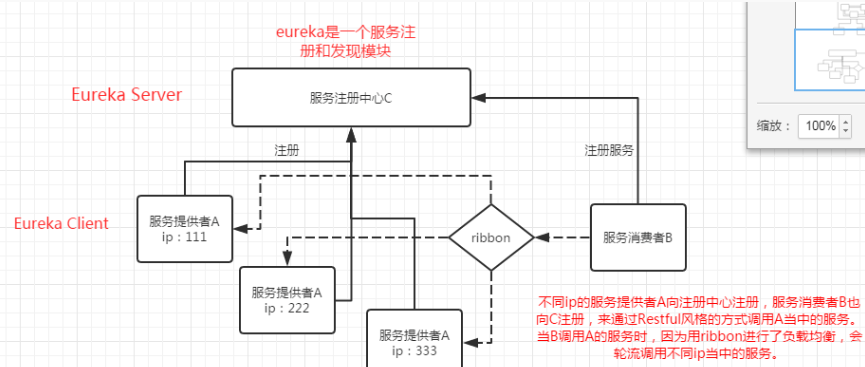
### 7.2、集群Eureka（注册中心）

当有服务注册时，三个Eureka-Server是对等的，它们都存有相同的信息，这就是通过服务器的冗余来增加可靠性，当有一台服务器宕机了，服务并不会终止，因为另两台服务存有相同的数据。



### 7.3、客户端负载均衡——Netflix Ribbon

Feign默认集成了ribbon。



## 8、消费服务方式

### 8.1、RestTemplate+Ribbon

### 8.2、声明式rest调用Feign

#### 8.2.1、Feign 采用的是基于接口的注解

#### 8**.2.2、Feign 整合了Ribbon**

#### 8.2.4.3、使用Feign，只需要创建一个接口并注解