

# Spring Cloud Consul: 服务治理与配置中心

Spring Cloud Consul 为 SpringBoot 应用提供了 Consul的支持, Consul既可以作为注册中心使用,也可以作为配置中心使用, 本文将对其用法进行详细介绍。

## Consul 简介

Consul是HashiCorp公司推出的开源软件, 提供了微服务系统中的服务治理、配置中心、控制总线等功能。这些功能中的每一个都可以根据需要单独使用, 也可以一起使用以构建全方位的服务网格, 总之Consul提供了一种完整的服务网格解决方案。



Spring Cloud Consul 具有如下特性:




- 支持服务治理: Consul作为注册中心时, 微服务中的应用可以向Consul注册自己, 并且可以从Consul获取其他应用信息;
- 支持客户端负责均衡: 包括Ribbon和Spring Cloud LoadBalancer;
- 支持Zuul: 当Zuul作为网关时, 可以从Consul中注册和发现应用;
- 支持分布式配置管理: Consul作为配置中心时, 使用键值对来存储配置信息;
- 支持控制总线: 可以在整个微服务系统中通过 Control Bus 分发事件消息。

## 使用Consul作为注册中心

### 安装并运行Consul

- 首先我们从官网下载Consul, 地址: <https://www.consul.io/downloads.html>

 Learn how Consul fits into the  HashiCorp Suite >

 Use Cases ▾ Intro Guides Docs API Community Enterprise  Download  GitHub

[Download Consul](#)  
[Download Consul Tools](#)

### Download Consul

Below are the available downloads for the latest version of Consul (1.6.1). Please download the proper package for your operating system and architecture.

You can find the [SHA256 checksums for Consul 1.6.1](#) online and you can [verify the checksums signature file](#) which has been signed using [HashiCorp's GPG key](#). You can also [download older versions of Consul](#) from the releases service.

Check out the [v1.6.1 CHANGELOG](#) for information on the latest release.

- 下载完成后只有一个exe文件, 双击运行;
- 在命令行中输入以下命令可以查看版本号:

```
1 consul --versionCopy to clipboardErrorCopied
```

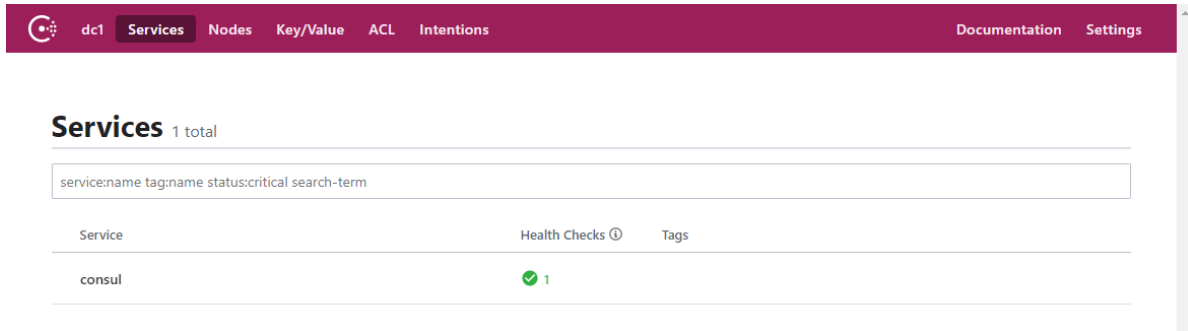
- 查看版本号信息如下:

```
1 Consul v1.6.1
2 Protocol 2 spoken by default, understands 2 to 3 (agent will automatically use
  protocol >2 when speaking to compatible agents)Copy to clipboardErrorCopied
```

- 使用开发模式启动：

```
1 consul agent -dev Copy to clipboardErrorCopied
```

- 通过以下地址可以访问Consul的首页：<http://localhost:8500>



## 创建应用注册到Consul

我们通过改造user-service和ribbon-service来演示下服务注册与发现的功能，主要是将应用原来的Eureka注册中心支持改为Consul注册中心支持。

- 创建consul-user-service模块和consul-ribbon-service模块；
- 修改相关依赖，把原来的Eureka注册发现的依赖改为Consul的，并添加SpringBoot Actuator的依赖：

```
1 <dependency>
2   <groupId>org.springframework.cloud</groupId>
3   <artifactId>spring-cloud-starter-consul-discovery</artifactId>
4 </dependency>
5 <dependency>
6   <groupId>org.springframework.boot</groupId>
7   <artifactId>spring-boot-starter-actuator</artifactId>
8 </dependency>Copy to clipboardErrorCopied
```

- 修改配置文件application.yml，将Eureka的注册发现配置改为Consul的：

```
1 server:
2   port: 8206
3 spring:
4   application:
5     name: consul-user-service
6   cloud:
7     consul: #Consul服务注册发现配置
8     host: localhost
9     port: 8500
10    discovery:
11      service-name: ${spring.application.name}Copy to clipboardErrorCopied
```

- 运行两个consul-user-service和一个consul-ribbon-service，在Consul页面上可以看到如下信息：

dc1 Services Nodes Key/Value ACL Intentions Documentation Settings			
Services 3 total			
service:name tag:name status:critical search-term			
Service	Health Checks ⓘ	Tags	
consul	✓ 1		
consul-ribbon-service	✓ 2	secure=false	
consul-user-service	✓ 4	secure=false	

## 负载均衡功能

由于我们运行了两个consul-user-service，而consul-ribbon-service默认会去调用它的接口，我们调用consul-ribbon-service的接口来演示下负载均衡功能。

多次调用接口：<http://localhost:8308/user/1>，可以发现两个consul-user-service的控制台交替打印如下信息。

```
1 2019-10-20 10:39:32.580 INFO 12428 --- [io-8206-exec-10]
   c.macro.cloud.controller.UserController : 根据id获取用户信息，用户名称为: macroCopy to
   clipboardErrorCopied
```

## 使用Consul作为配置中心

我们通过创建consul-config-client模块，并在Consul中添加配置信息来演示下配置管理的功能。

### 创建consul-config-client模块

- 在pom.xml中添加相关依赖：

```
1 <dependency>
2   <groupId>org.springframework.cloud</groupId>
3   <artifactId>spring-cloud-starter-consul-config</artifactId>
4 </dependency>
5 <dependency>
6   <groupId>org.springframework.cloud</groupId>
7   <artifactId>spring-cloud-starter-consul-discovery</artifactId>
8 </dependency>Copy to clipboardErrorCopied
```

- 添加配置文件application.yml，启用的是dev环境的配置：

```
1 spring:
2   profiles:
3     active: devCopy to clipboardErrorCopied
```

- 添加配置文件bootstrap.yml，主要是对Consul的配置功能进行配置：

```
1 server:
2   port: 9101
3 spring:
4   application:
```

```

5     name: consul-config-client
6     cloud:
7       consul:
8         host: localhost
9         port: 8500
10        discovery:
11          serviceName: consul-config-client
12        config:
13          enabled: true #是否启用配置中心功能
14          format: yaml #设置配置值的格式
15          prefix: config #设置配置所在目录
16          profile-separator: ':' #设置配置的分隔符
17          data-key: data #配置key的名字，由于Consul是K/V存储，配置存储在对应K的V中Copy to
clipboardErrorCopied

```

- 创建ConfigClientController，从Consul配置中心中获取配置信息：

```

1  /**
2   * Created by macro on 2019/9/11.
3   */
4  @RestController
5  @RefreshScope
6  public class ConfigClientController {
7
8      @Value("${config.info}")
9      private String configInfo;
10
11      @GetMapping("/configInfo")
12      public String getConfigInfo() {
13          return configInfo;
14      }
15  }Copy to clipboardErrorCopied

```

## 在Consul中添加配置

- 在consul中添加配置存储的key为:

```

1  config/consul-config-client:dev/dataCopy to clipboardErrorCopied

```

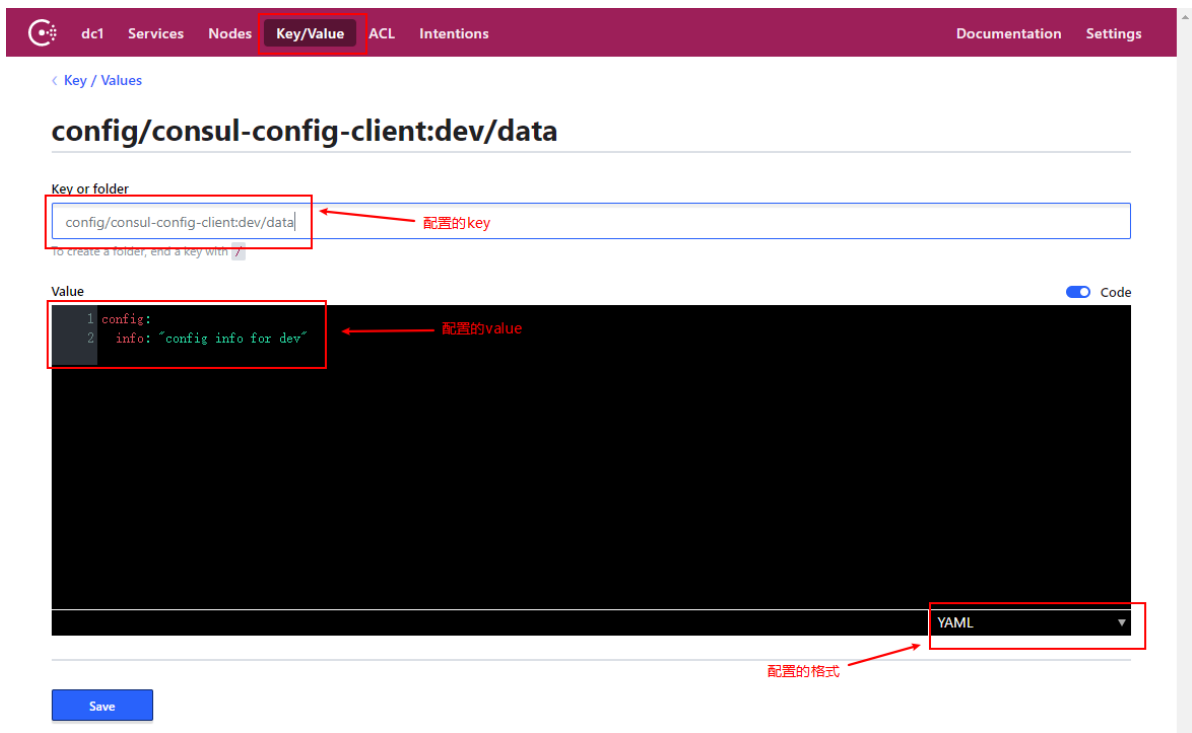
- 在consul中添加配置存储的value为:

```

1  config:
2  info: "config info for dev"Copy to clipboardErrorCopied

```

- 存储信息截图如下:



- 启动consul-config-client，调用接口查看配置信息：<http://localhost:9101/configInfo>

```
1 config info for devCopy to clipboardErrorCopied
```

## Consul的动态刷新配置

我们只要修改下Consul中的配置信息，再次调用查看配置的接口，就会发现配置已经刷新。回想下在使用Spring Cloud Config的时候，我们需要调用接口，通过Spring Cloud Bus才能刷新配置。Consul使用其自带的Control Bus 实现了一种事件传递机制，从而实现了动态刷新功能。

## 使用到的模块

```
1 springcloud-learning
2 |—— consul-config-client -- 用于演示consul作为配置中心的consul客户端
3 |—— consul-user-service -- 注册到consul的提供User对象CRUD接口的服务
4 |—— consul-service -- 注册到consul的ribbon服务调用测试服务
```