# Springcloud ——Consul

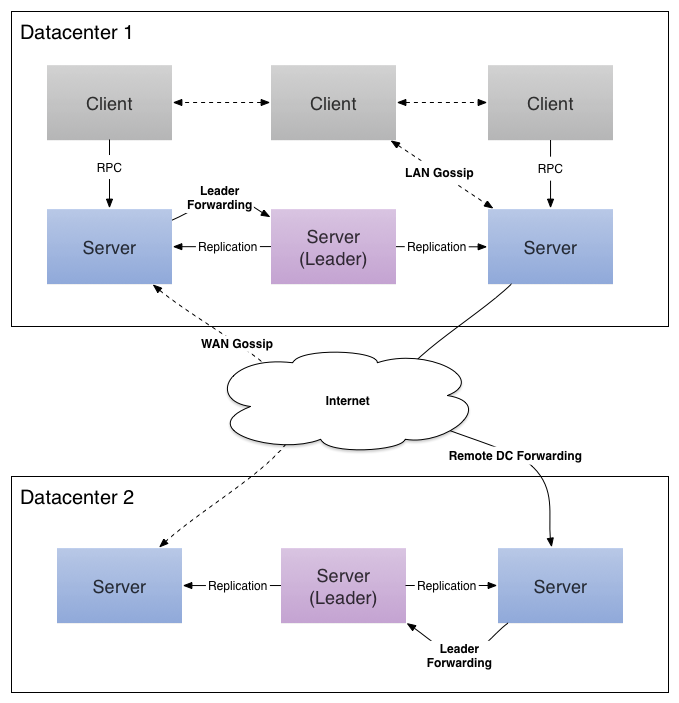
**Consul 介绍：**

Consul 是用于实现分布式系统的服务发现与配置。与其它分布式服务注册与发现的方案，Consul 的方案更“一站式”，内置了服务注册与发现框 架、分布一致性协议实现、健康检查、Key/Value 存储、多数据中心方案，不再需要依赖其它工具（比如 ZooKeeper 等）。使用起来也较 为简单。Consul 使用 Go 语言编写，因此具有天然可移植性(支持Linux、windows和Mac OS X)；安装包仅包含一个可执行文件，方便部署，与 Docker 等轻量级容器可无缝配合。

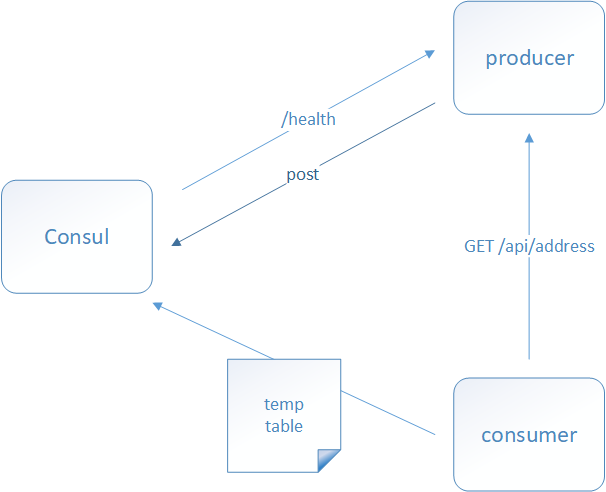
**Consul 角色：**

* client: 客户端, 无状态, 将 HTTP 和 DNS 接口请求转发给局域网内的服务端集群。
* server: 服务端, 保存配置信息, 高可用集群, 在局域网内与本地客户端通讯, 通过广域网与其它数据中心通讯。 每个数据中心的 server 数量推荐为 3 个或是 5 个。

Consul 客户端、服务端还支持夸中心的使用，更加提高了它的高可用性。



**Consul 工作原理：**



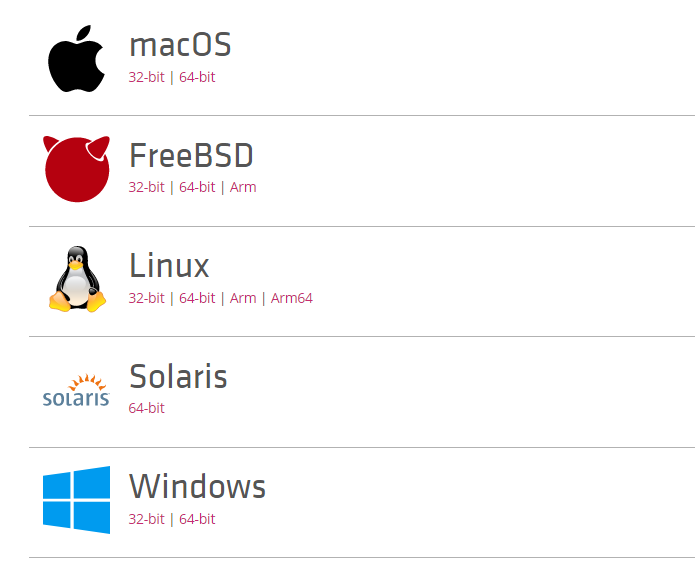
* 1、当 Producer 启动的时候，会向 Consul 发送一个 post 请求，告诉 Consul 自己的 IP 和 Port
* 2、Consul 接收到 Producer 的注册后，每隔10s（默认）会向 Producer 发送一个健康检查的请求，检验Producer是否健康
* 3、当 Consumer 发送 GET 方式请求 /api/address 到 Producer 时，会先从 Consul 中拿到一个存储服务 IP 和 Port 的临时表，从表中拿到 Producer 的 IP 和 Port 后再发送 GET 方式请求 /api/address
* 4、该临时表每隔10s会更新，只包含有通过了健康检查的 Producer

Spring Cloud Consul 项目是针对 Consul 的服务治理实现。Consul 是一个分布式高可用的系统，它包含多个组件，但是作为一个整体，在微服务架构中为我们的基础设施提供服务发现和服务配置的工具。

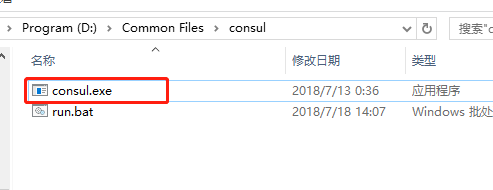
**Consul 安装：**

Consul 不同于 Eureka 需要单独安装，访问[Consul 官网](https://www.consul.io/downloads.html)下载 Consul 的最新版本，我这里是 consul\_1.2.1。

根据不同的系统类型选择不同的安装包，从下图也可以看出 Consul 支持所有主流系统。



这里以 Windows 为例，下载下来是一个 consul\_1.2.1\_windows\_amd64.zip 的压缩包，解压是是一个 consul.exe 的执行文件。



cd 到对应的目录下，使用 cmd 启动 Consul

cd D:\Common Files\consul

#cmd启动：

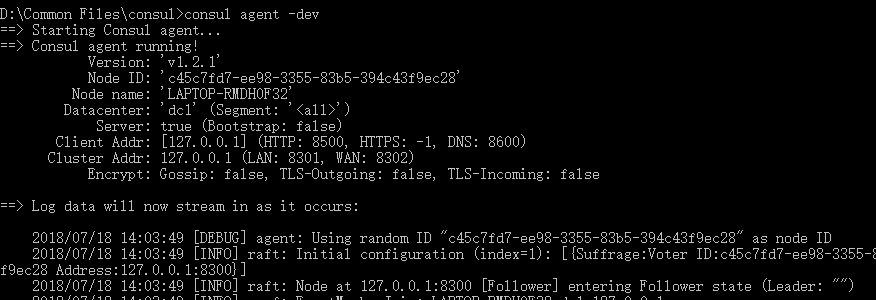
consul agent -dev # -dev表示开发模式运行，另外还有-server表示服务模式运行

为了方便期间，可以在同级目录下创建一个 run.bat 脚本来启动，脚本内容如下：

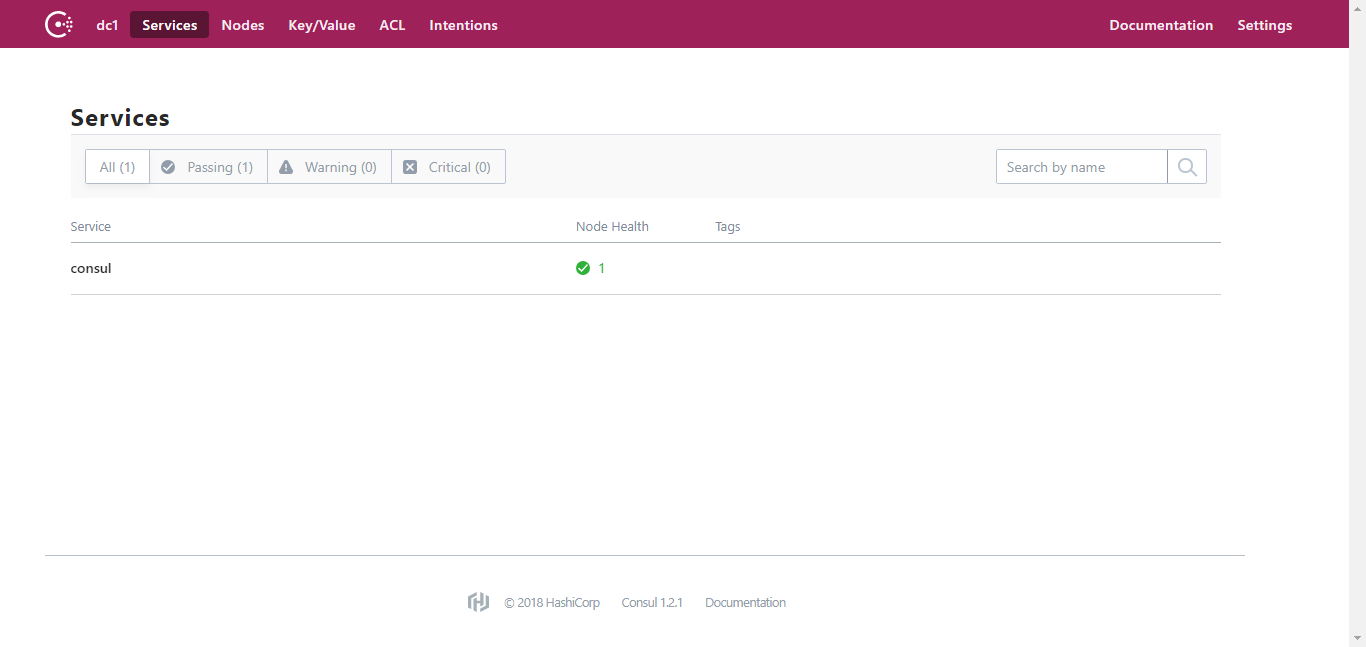
consul agent -dev

pause

启动结果如下：



启动成功之后访问：http://localhost:8500，可以看到 Consul 的管理界面：



这样就意味着Consul 服务启动成功了。

**Consul 服务端：**

接下来开发 Consul 的服务端，首先创建一个 spring-cloud-consul-producer 项目

### 添加依赖包

依赖包如下：

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-consul-discovery</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

* spring-boot-starter-actuator 健康检查依赖于此包。
* spring-cloud-starter-consul-discovery Spring Cloud Consul 的支持。

Spring Boot 版本使用的是 2.0.3.RELEASE，Spring Cloud 最新版本是 Finchley.RELEASE 依赖于 Spring Boot 2.x.

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.3.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<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>

</dependencies>

</dependencyManagement>

### 配置文件

配置文件内容如下：

spring.application.name=spring-cloud-consul-producer

server.port=8501

spring.cloud.consul.host=localhost

spring.cloud.consul.port=8500

#注册到consul的服务名称

spring.cloud.consul.discovery.serviceName=service-producer

Consul 的地址和端口号默认是 localhost:8500 ，如果不是这个地址可以自行配置。 spring.cloud.consul.discovery.serviceName 是指注册到 Consul 的服务名称，后期客户端会根据这个名称来进行服务调用。

**启动类：**

**@SpringBootApplication**

**@EnableDiscoveryClient**

**public** **class** **ConsulProducerApplication** **{**

**public** **static** **void** **main(**String**[]** args**)** **{**

SpringApplication**.**run**(**ConsulProducerApplication**.**class**,** args**);**

**}**

**}**

添加了 @EnableDiscoveryClient 注解表示支持服务发现。

### 提供服务：

我们在创建一个 Controller，推文提供 hello 的服务。

**@RestController**

**public** **class** **HelloController** **{**

**@RequestMapping(**"/hello"**)**

**public** String **hello()** **{**

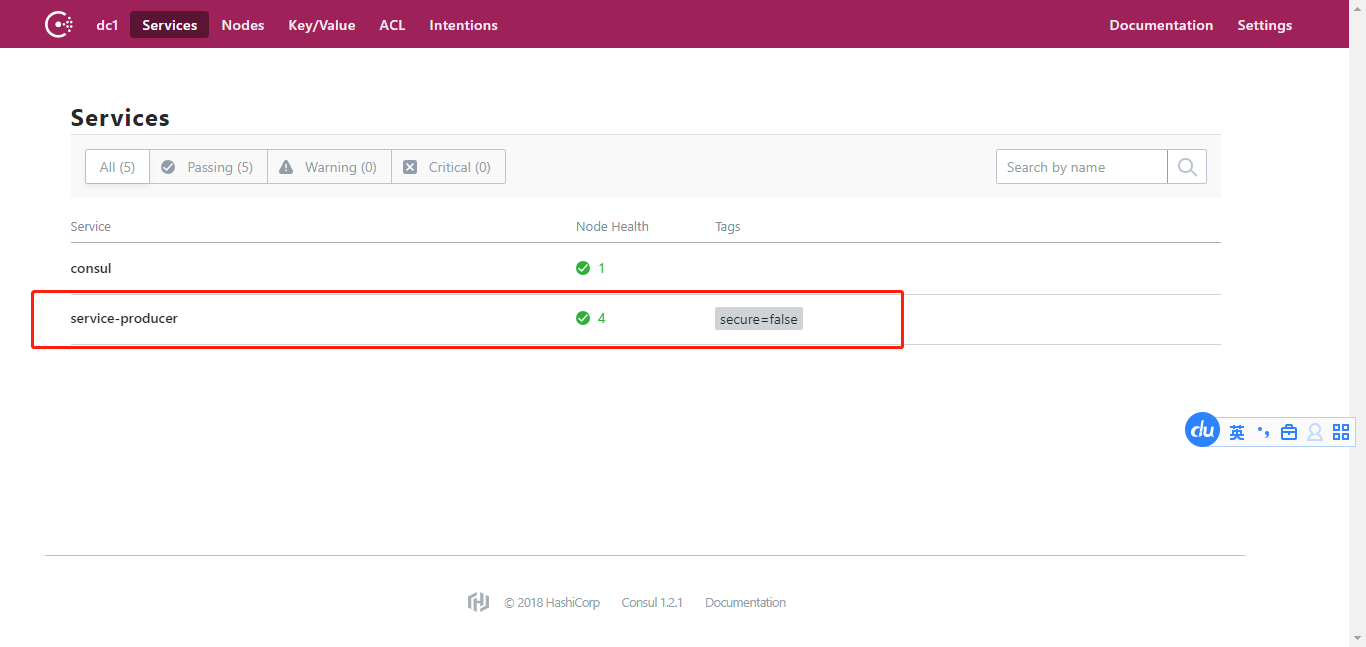
**return** "hello consul"**;**

**}**

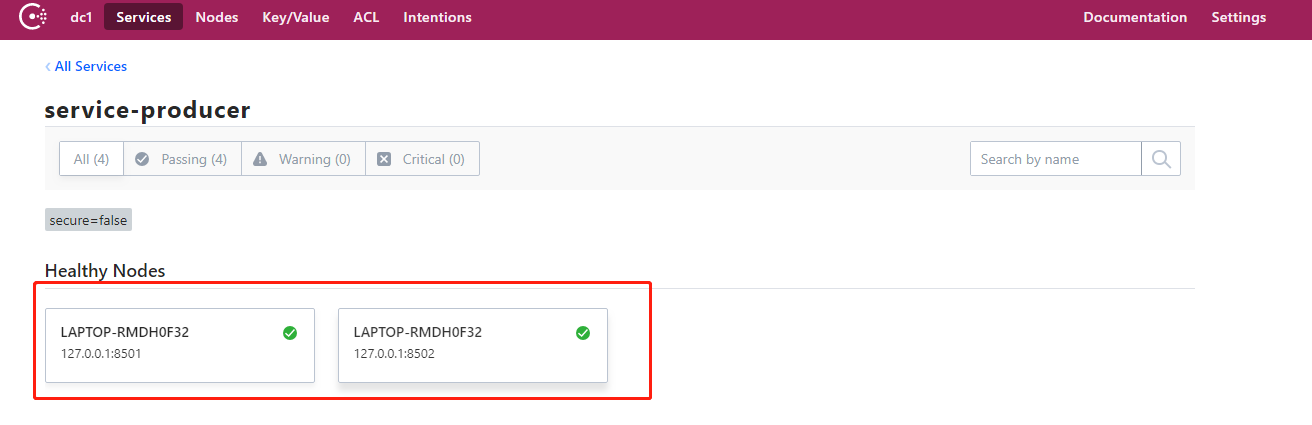
**}**

为了模拟注册均衡负载复制一份上面的项目重命名为 spring-cloud-consul-producer-2 ,修改对应的端口为 8502，修改 hello 方法的返回值为：”hello consul two”，修改完成后依次启动两个项目。

这时候我们再次在浏览器访问地址：http://localhost:8500，显示如下：



我们发现页面多了 service-producer 服务，点击进去后页面显示有两个服务提供者：

这样服务提供者就准备好了。

**Consul 消费端：**

我们创建一个 spring-cloud-consul-consumer 项目，pom 文件和上面示例保持一致。

### 配置文件

配置文件内容如下

spring.application.name=spring-cloud-consul-consumer

server.port=8503

spring.cloud.consul.host=127.0.0.1

spring.cloud.consul.port=8500

#设置不需要注册到 consul 中

spring.cloud.consul.discovery.register=false

客户端可以设置注册到 Consul 中，也可以不注册到 Consul 注册中心中，根据我们的业务来选择，只需要在使用服务时通过 Consul 对外提供的接口获取服务信息即可。

**启动类：**

**@SpringBootApplication**

**public** **class** **ConsulConsumerApplication** **{**

**public** **static** **void** **main(**String**[]** args**)** **{**

SpringApplication**.**run**(**ConsulConsumerApplication**.**class**,** args**);**

**}**

**}**

### 进行测试：

我们先来创建一个 ServiceController ,试试如果去获取 Consul 中的服务

**@RestController**

**public** **class** **ServiceController** **{**

**@Autowired**

**private** LoadBalancerClient loadBalancer**;**

**@Autowired**

**private** DiscoveryClient discoveryClient**;**

*/\*\**

*\* 获取所有服务*

*\*/*

**@RequestMapping(**"/services"**)**

**public** Object **services()** **{**

**return** discoveryClient**.**getInstances**(**"service-producer"**);**

**}**

*/\*\**

*\* 从所有服务中选择一个服务（轮询）*

*\*/*

**@RequestMapping(**"/discover"**)**

**public** Object **discover()** **{**

**return** loadBalancer**.**choose**(**"service-producer"**).**getUri**().**toString**();**

**}**

**}**

Controller 中有俩个方法，一个是获取所有服务名为service-producer的服务信息并返回到页面，一个是随机从服务名为service-producer的服务中获取一个并返回到页面。

添加完 ServiceController 之后我们启动项目，访问地址：http://localhost:8503/services，返回：

[{"serviceId":"service-producer","host":"windows10.microdone.cn","port":8501,"secure":false,"metadata":{"secure":"false"},"uri":"http://windows10.microdone.cn:8501","scheme":null},{"serviceId":"service-producer","host":"windows10.microdone.cn","port":8502,"secure":false,"metadata":{"secure":"false"},"uri":"http://windows10.microdone.cn:8502","scheme":null}]

发现我们刚才创建的端口为 8501 和 8502 的两个服务端都存在。

多次访问地址：http://localhost:8503/discover，页面会交替返回下面信息：

http://windows10.microdone.cn:8501

http://windows10.microdone.cn:8502

...

说明 8501 和 8502 的两个服务会交替出现，从而实现了获取服务端地址的均衡负载。

大多数情况下我们希望使用均衡负载的形式去获取服务端提供的服务，因此使用第二种方法来模拟调用服务端提供的 hello 方法。

创建 CallHelloController ：

**@RestController**

**public** **class** **CallHelloController** **{**

**@Autowired**

**private** LoadBalancerClient loadBalancer**;**

**@RequestMapping(**"/call"**)**

**public** String **call()** **{**

ServiceInstance serviceInstance **=** loadBalancer**.**choose**(**"service-producer"**);**

System**.**out**.**println**(**"服务地址：" **+** serviceInstance**.**getUri**());**

System**.**out**.**println**(**"服务名称：" **+** serviceInstance**.**getServiceId**());**

String callServiceResult **=** **new** RestTemplate**().**getForObject**(**serviceInstance**.**getUri**().**toString**()** **+** "/hello"**,** String**.**class**);**

System**.**out**.**println**(**callServiceResult**);**

**return** callServiceResult**;**

**}**

**}**

使用 RestTemplate 进行远程调用。添加完之后重启 spring-cloud-consul-consumer 项目。在浏览器中访问地址：http://localhost:8503/call，依次返回结果如下：

hello consul

hello consul two

...

说明已经成功的调用了 Consul 服务端提供的服务，并且实现了服务端的均衡负载功能。通过以上我们发现 Consul 提供的服务发现易用、强大。