7. Consul

在 Spring Cloud 中，大部分组件都有备选方案，例如注册中心，除了常见 Eureka 之外，像 zookeeper 我们也可以直接使用在 Spring Cloud 中，还有另外一个比较重要的方案，就是 Consul。

Consul 是 HashiCorp 公司推出来的开源产品。主要提供了：服务发现、服务隔离、服务配置等功能。

相比于 Eureka 和 zookeeper，Consul 配置更加一站式，因为它内置了很多微服务常见的需求：服务发现与注册、分布式一致性协议实现、健康检查、键值对存储、多数据中心等，我们不再需要借助第三方组件来实现这些功能。

7.1 安装

不同于 Eureka，Consul 使用 Go 语言开发，所以，使用 Consul，我们需要先安装软件。

在 Linux 中，首先执行如下命令下载 Consul：

```
wget https://releases.hashicorp.com/consul/1.6.2/consul_1.6.2_linux_amd64.zip
```

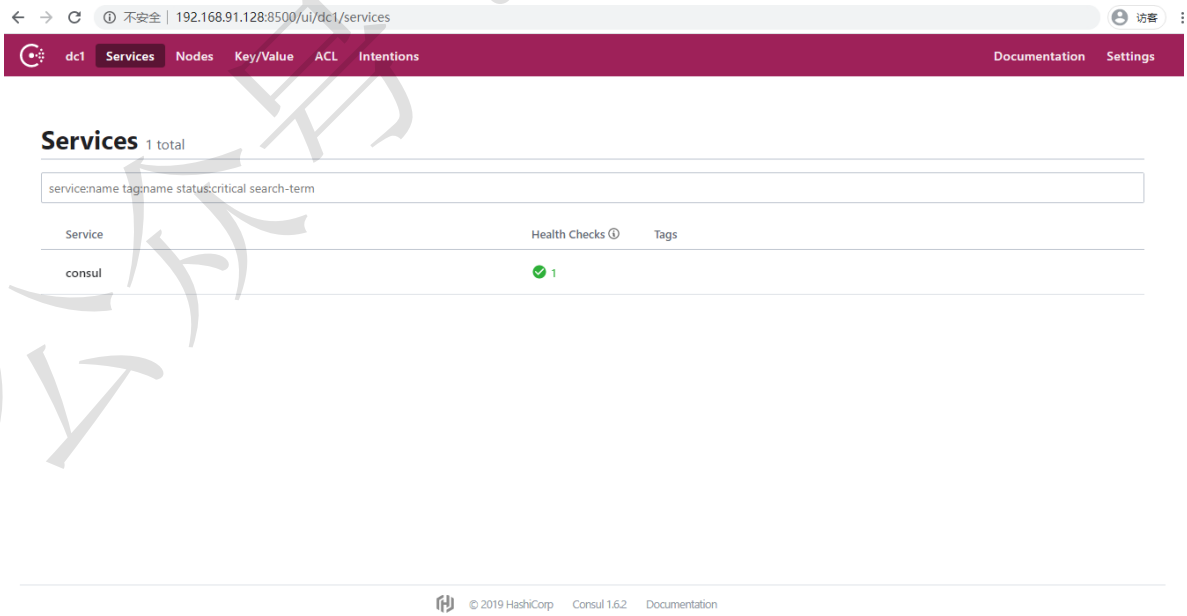
然后解压下载文件：

```
unzip consul_1.6.2_linux_amd64.zip
```

解压完成后，我们在当前目录下就可以看到 consul 文件，然后执行如下命令，启动 Consul：

```
./consul agent -dev -ui -node=consul-dev -client=192.168.91.128
```

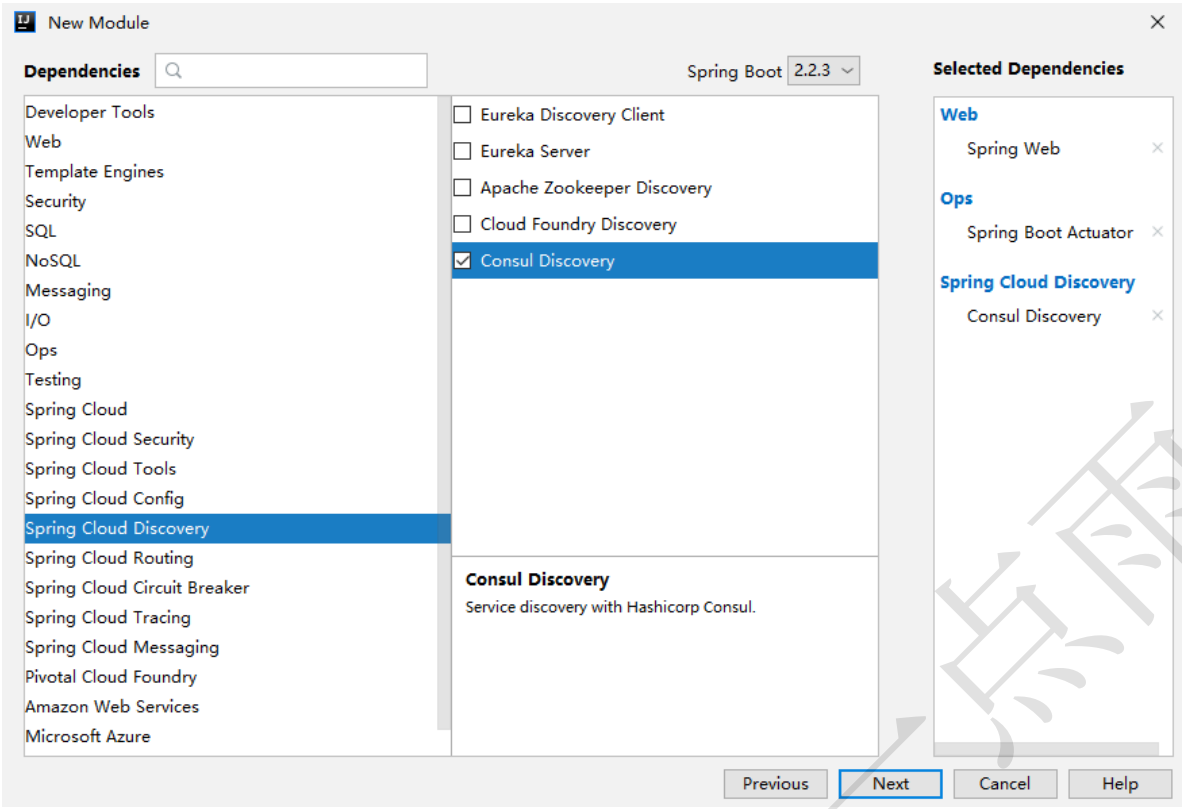
启动成功后，在物理机中，我们可以直接访问 Consul 的后台管理页面（注意，这个访问要确保 8500 端口可用，或者直接关闭防火墙）：



7.2 Consul 使用

简单看一个注册消费的案例。

首先我们来创建一个服务提供者。就是一个普通的 Spring Boot 项目，添加如下依赖：



项目创建成功后，添加如下配置：

```
spring.application.name=consul-provider
server.port=2000
# Consul 相关配置
spring.cloud.consul.host=192.168.91.128
spring.cloud.consul.port=8500
spring.cloud.consul.discovery.service-name=consul-provider
```

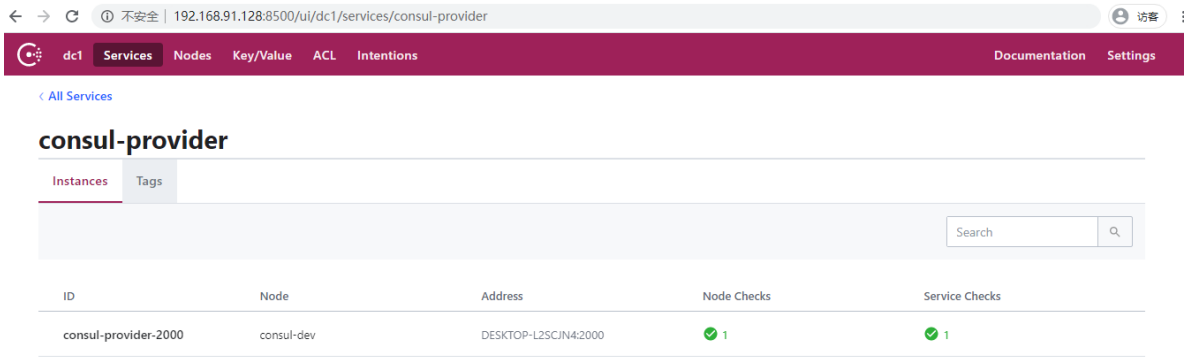
在项目启动类上开启服务发现的功能：

```
@SpringBootApplication
@EnableDiscoveryClient
public class ConsulProviderApplication {

    public static void main(String[] args) {
        SpringApplication.run(ConsulProviderApplication.class, args);
    }

}
```

最后添加一个测试接口：

```
@RestController
public class HelloController {
    @GetMapping("/hello")
    public String hello() {
        return "hello";
    }
}
```

接下来就是启动项目，项目启动成功后，访问 consul 后台管理页面，看到如下信息，表示 consul 已经注册成功了。



7.3 Consul 集群注册

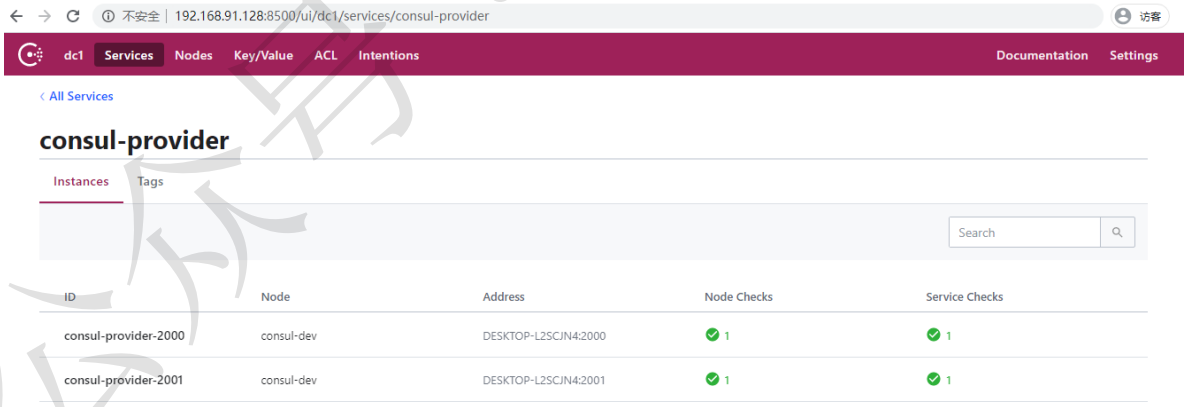
为了区分集群中的哪一个 provider 提供的服务，我们修改一下 consul 中的接口：

```
@RestController
public class HelloController {
    @Value("${server.port}")
    Integer port;

    @GetMapping("/hello")
    public String hello() {
        return "hello>>" + port;
    }
}
```

修改完成后，对项目进行打包。打包成功后，命令行执行如下两行命令，启动两个 provider 实例：

```
java -jar consul-provider-0.0.1-SNAPSHOT.jar --server.port=2000
java -jar consul-provider-0.0.1-SNAPSHOT.jar --server.port=2001
```

启动成功后，再去 consul 后台管理页面，就可以看到有两个实例了：



7.4 消费

首先创建一个消费实例，创建方式和 provider 一致。

创建成功后，添加如下配置：

```
spring.application.name=consul-consumer
server.port=2002
spring.cloud.consul.host=192.168.91.128
spring.cloud.consul.port=8500
spring.cloud.consul.discovery.service-name=consul-consumer
```

开启服务发现，并添加 RestTemplate：

```
@SpringBootApplication
@EnableDiscoveryClient
public class ConsulConsumerApplication {

    public static void main(String[] args) {
        SpringApplication.run(ConsulConsumerApplication.class, args);
    }

    @Bean
    RestTemplate restTemplate() {
        return new RestTemplate();
    }
}
```

最后，提供一个服务调用的方法：

```
@RestController
public class HelloController {
    @Autowired
    LoadBalancerClient loadBalancerClient;
    @Autowired
    RestTemplate restTemplate;

    @GetMapping("/hello")
    public void hello() {
        ServiceInstance choose = loadBalancerClient.choose("consul-provider");
        System.out.println("服务地址: " + choose.getUri());
        System.out.println("服务名称:" + choose.getServiceId());
        String s = restTemplate.getForObject(choose.getUri() + "/hello",
String.class);
        System.out.println(s);
    }
}
```

这里，我们通过 loadBalancerClient 实例，可以获取要调用的 ServiceInstance。获取到调用地址之后，再用 RestTemplate 去调用。

然后，启动项目，浏览器输入 http://localhost:2002/hello ，查看请求结果，这个请求自带负载均衡功能。