第 1 章

Spring Cloud Consul 初识

Spring Cloud Consul 在 Spring Cloud 生态环境中承担注册中心的角色，本章将对其进行简要说明。

1.1 构建 Spring Cloud Consul 源码环境

本节将对 Spring Cloud Consul 源码环境搭建相关内容进行说明，从零开始搭建 Spring Cloud Consul 源码阅读环境，搭建阅读环境需要 git、JDK8 及以上（Java 开发环境）、IDEA 编辑器和 Maven（包管理工具）。

在 GitHub 上找到 Spring Cloud Consul 服务器端项目仓库并将代码克隆到本地，具体操作命令如下：

```
git clone git@github.com:spring-cloud/spring-cloud-consul.git
```

当执行上述命令时有可能会出现如下异常信息（下文异常信息只截取部分）：

```
error: unable to create file (文件路径): F:\ilename too long
```

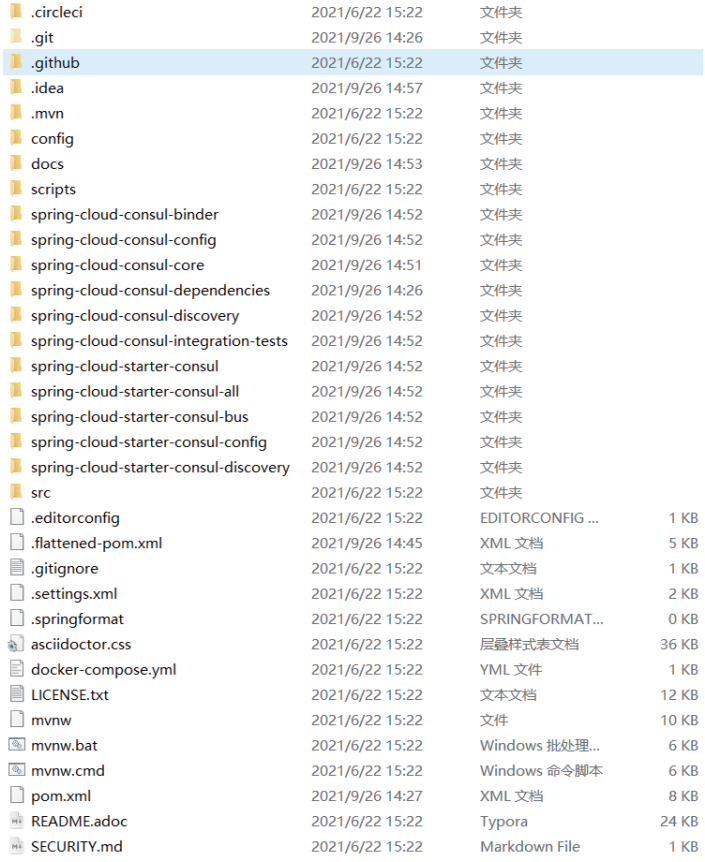
当出现上述异常信息时表示文件名称太长，这个现象在 Windows 系统中比较常见，原因是 git 调用的是 Windows 系统提供的旧 API，长度限制是 260，解决该问题的处理方案只需要执行如下一行代码：

```
git config --global core.longpaths true
```

当执行完成上述命令后就可以重新执行克隆语句，将 GitHub 上的 Spring Cloud Consul 服务器端项目工程拉取到本地，执行完成后会拥有 Spring Cloud Consul 本地仓库，具体内容如图 1.1 所示。

接下来需要在该文件夹下打开 git bash 命令行工具，在命令行工具中输入如下命令：

```
git branch sh-v3.0.3 v3.0.3
```



.circleci	2021/6/22 15:22	文件夹	
.git	2021/9/26 14:26	文件夹	
.github	2021/6/22 15:22	文件夹	
.idea	2021/9/26 14:57	文件夹	
.mvn	2021/6/22 15:22	文件夹	
config	2021/6/22 15:22	文件夹	
docs	2021/9/26 14:53	文件夹	
scripts	2021/6/22 15:22	文件夹	
spring-cloud-consul-binder	2021/9/26 14:52	文件夹	
spring-cloud-consul-config	2021/9/26 14:52	文件夹	
spring-cloud-consul-core	2021/9/26 14:51	文件夹	
spring-cloud-consul-dependencies	2021/9/26 14:26	文件夹	
spring-cloud-consul-discovery	2021/9/26 14:52	文件夹	
spring-cloud-consul-integration-tests	2021/9/26 14:52	文件夹	
spring-cloud-starter-consul	2021/9/26 14:52	文件夹	
spring-cloud-starter-consul-all	2021/9/26 14:52	文件夹	
spring-cloud-starter-consul-bus	2021/9/26 14:52	文件夹	
spring-cloud-starter-consul-config	2021/9/26 14:52	文件夹	
spring-cloud-starter-consul-discovery	2021/9/26 14:52	文件夹	
src	2021/6/22 15:22	文件夹	
.editorconfig	2021/6/22 15:22	EDITORCONFIG ...	1 KB
.flattened-pom.xml	2021/9/26 14:45	XML 文档	5 KB
.gitignore	2021/6/22 15:22	文本文档	1 KB
.settings.xml	2021/6/22 15:22	XML 文档	2 KB
.springformat	2021/6/22 15:22	SPRINGFORMAT...	0 KB
asciidoctor.css	2021/6/22 15:22	层叠样式表文档	36 KB
docker-compose.yml	2021/6/22 15:22	YML 文件	1 KB
LICENSE.txt	2021/6/22 15:22	文本文档	12 KB
mvnw	2021/6/22 15:22	文件	10 KB
mvnw.bat	2021/6/22 15:22	Windows 批处理...	6 KB
mvnw.cmd	2021/6/22 15:22	Windows 命令脚本	6 KB
pom.xml	2021/9/26 14:27	XML 文档	8 KB
README.adoc	2021/6/22 15:22	Typora	24 KB
SECURITY.md	2021/6/22 15:22	Markdown File	1 KB

图 1.1　Spring Cloud Consul 本地仓库

上述命令表示创建一个分支，该分支的源头是 tag 为 v3.0.3 的提交节点。执行上述命令后需要执行 git branch 命令来确定是否创建成功。执行 git branch 命令，命令行会输出如下内容：

```
$ git branch
* main
sh-v3.0.3
```

sh-v3.0.3 表示分支已经创建成功，接下来需要切换到该分支，具体切换命令如下：

```
git checkout sh-v3.0.3
```

此时，再次输入 git branch 执行命令，在控制台输出如下内容：

```
$ git branch
  main
* sh-v3.0.3
```

表示已经完成分支切换。下面介绍如何将 Spring Cloud Consul 源码导入 IDEA 中，导入过程十分简单，只需要用 IDEA 将 Spring Cloud Consul 源码文件夹打开即可。打开的 IDEA 项目视图界面如图 1.2 所示。

在打开 Spring Cloud Consul 源码文件夹后，IDEA 会自动从 Maven 中央仓库拉取相关依赖，此时需要等待 10~20 分钟完成依赖下载。

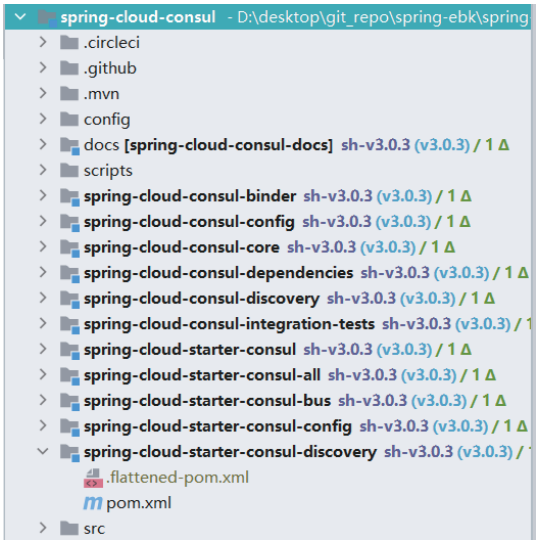


图 1.2　Spring Cloud Consul IDEA 项目视图界面

1.2　验证 Spring Cloud Consul 源码环境

本节将在 Spring Cloud Consul 源码基础上进行源码环境的测试，主要测试目标是进行 Spring Cloud Consul 的使用。在开始进行测试之前需要先获取 Consul 项目，Consul 的官方网站是 https：//www.consul.io/。在该网站中下载符合当前操作系统的版本，本例中下载的是 Windows 版本的 Consul，下载地址是 https：//www.consul.io/downloads，在下载页面中选择 Windows 操作系统，再单击 64-bit 文字即可进行下载。

下载后得到的是一个压缩包，本例中压缩包的文件名为 consul_1.10.2_windows_amd64.zip，将其解压后得到 consul.exe 文件。在解压目录中启动控制台，向控制台中输入如下代码启动 Consul：

```
consul agent -dev
```

执行命令，会看到如下代码：

```
D:\desktop\git_repo\spring-ebk\spring-cloud\consule>consul agent -dev
==> Starting Consul agent...
           Version: '1.10.2'
           Node ID: '2d7a2bbf-032e-4bfe-2906-40dee9ace162'
         Node name: 'LAPTOP-D7DF36F6'
        Datacenter: 'dc1' (Segment: '<all>')
            Server: true (Bootstrap: false)
       Client Addr: [127.0.0.1] (HTTP: 8500, HTTPS: -1, gRPC: 8502, DNS: 8600)
      Cluster Addr: 127.0.0.1 (LAN: 8301, WAN: 8302)
           Encrypt: Gossip: false, TLS-Outgoing: false, TLS-Incoming: false,
Auto-Encrypt-TLS: false

==> Log data will now stream in as it occurs:

2021-09-26T15:08:30.177+0800 [INFO]  agent.server.raft: initial configuration:
index=1 servers="[{Suffrage:Voter ID:2d7a2bbf-032e-4bfe-2906-40dee9ace162 Address:
```

```
127.0.0.1: 8300}]"
    2021-09-26T15: 08: 30.191+0800 [INFO]  agent.server.raft: entering follower state: follower="Node at 127.0.0.1: 8300 [Follower]" leader=
    2021-09-26T15: 08: 30.192+0800 [INFO]  agent.server.serf.wan: serf: EventMemberJoin: LAPTOP-D7DF36F6.dc1 127.0.0.1
    2021-09-26T15: 08: 30.192+0800 [INFO]  agent.server.serf.lan: serf: EventMemberJoin: LAPTOP-D7DF36F6 127.0.0.1
    2021-09-26T15: 08: 30.192+0800 [INFO]  agent.router: Initializing LAN area manager
    2021-09-26T15: 08: 30.192+0800 [INFO]  agent.server: Adding LAN server: server="LAPTOP-D7DF36F6 (Addr: tcp/127.0.0.1: 8300) (DC: dc1)"
    2021-09-26T15: 08: 30.193+0800 [WARN]  agent: grpc: addrConn.createTransport failed to connect to {dc1-127.0.0.1: 8300 0 LAPTOP-D7DF36F6 <nil>}. Err : connection error: desc = "transport: Error while dialing dial tcp 127.0.0.1: 0->127.0.0.1: 8300: operation was canceled". Reconnecting...
    2021-09-26T15: 08: 30.193+0800 [WARN]  agent: grpc: addrConn.createTransport failed to connect to {dc1-127.0.0.1: 8300 0 LAPTOP-D7DF36F6 <nil>}. Err : connection error: desc = "transport: Error while dialing dial tcp 127.0.0.1: 0->127.0.0.1: 8300: operation was canceled". Reconnecting...
    2021-09-26T15: 08: 30.193+0800 [INFO]  agent.server: Handled event for server in area: event=member-join server=LAPTOP-D7DF36F6.dc1 area=wan
    2021-09-26T15: 08: 30.193+0800 [INFO]  agent: Started DNS server: address=127.0.0.1: 8600 network=udp
    2021-09-26T15: 08: 30.194+0800 [INFO]  agent: Started DNS server: address=127.0.0.1: 8600 network=tcp
    2021-09-26T15: 08: 30.194+0800 [INFO]  agent: Starting server: address=127.0.0.1: 8500 network=tcp protocol=http
    2021-09-26T15: 08: 30.195+0800 [INFO]  agent: Started gRPC server: address=127.0.0.1: 8502 network=tcp
    2021-09-26T15: 08: 30.195+0800 [WARN]  agent: DEPRECATED Backwards compatibility with pre-1.9 metrics enabled. These metrics will be removed in a future version of Consul. Set 'telemetry { disable_compat_1.9 = true }' to disable them
    2021-09-26T15: 08: 30.196+0800 [INFO]  agent: started state syncer
    2021-09-26T15: 08: 30.196+0800 [INFO]  agent: Consul agent running
    2021-09-26T15: 08: 30.261+0800 [WARN]  agent.server.raft: heartbeat timeout reached, starting election: last-leader=
    2021-09-26T15: 08: 30.261+0800 [INFO]  agent.server.raft: entering candidate state: node="Node at 127.0.0.1: 8300 [Candidate]" term=2
    2021-09-26T15: 08: 30.263+0800 [DEBUG] agent.server.raft: votes: needed=1
    2021-09-26T15: 08: 30.263+0800 [DEBUG] agent.server.raft: vote granted: from=2d7a2bbf-032e-4bfe-2906-40dee9ace162 term=2 tally=1
    2021-09-26T15: 08: 30.264+0800 [INFO]  agent.server.raft: election won: tally=1
    2021-09-26T15: 08: 30.264+0800 [INFO]  agent.server.raft: entering leader state: leader="Node at 127.0.0.1: 8300 [Leader]"
    2021-09-26T15: 08: 30.264+0800 [INFO]  agent.server: cluster leadership acquired
    2021-09-26T15: 08: 30.264+0800 [DEBUG] agent.server: Cannot upgrade to new ACLs: leaderMode=0 mode=0 found=true leader=127.0.0.1: 8300
    2021-09-26T15: 08: 30.264+0800 [INFO]  agent.server: New leader elected: payload=LAPTOP-D7DF36F6
    2021-09-26T15: 08: 30.266+0800 [DEBUG] agent.server.autopilot: autopilot is now running
    2021-09-26T15: 08: 30.266+0800 [INFO]  agent.leader: started routine: routine="federation state anti-entropy"
    2021-09-26T15: 08: 30.266+0800 [DEBUG] agent.server.autopilot: state update routine is now running
    2021-09-26T15: 08: 30.267+0800 [INFO]  agent.leader: started routine: routine="federation state pruning"
    2021-09-26T15: 08: 30.267+0800 [DEBUG] connect.ca.consul: consul CA provider configured: id=07: 80: c8: de: f6: 41: 86: 29: 8f: 9c: b8: 17: d6: 48: c2: d5: c5: 5c: 7f: 0c: 03: f7: cf: 97: 5a: a7: c1: 68: aa: 23: ae: 81 is_primary=true
    2021-09-26T15: 08: 30.276+0800 [INFO]  agent.server.connect: initialized primary
```

```
datacenter CA with provider: provider=consul
    2021-09-26T15: 08: 30.278+0800 [INFO]  agent.leader: started routine: routine="intermediate cert renew watch"
    2021-09-26T15: 08: 30.279+0800 [INFO]  agent.leader: started routine: routine="CA root pruning"
    2021-09-26T15: 08: 30.279+0800 [DEBUG] agent.server: successfully established leadership: duration=15.0117ms
    2021-09-26T15: 08: 30.279+0800 [INFO]  agent.server: member joined, marking health alive: member=LAPTOP-D7DF36F6
    2021-09-26T15: 08: 30.510+0800 [DEBUG] agent: Skipping remote check since it is managed automatically: check=serfHealth
    2021-09-26T15: 08: 30.513+0800 [INFO]  agent: Synced node info
    2021-09-26T15: 08: 30.515+0800 [DEBUG] agent: Node info in sync
    2021-09-26T15: 08: 30.573+0800 [INFO]  agent.server: federation state anti-entropy synced
    2021-09-26T15: 08: 30.762+0800 [DEBUG] agent: Skipping remote check since it is managed automatically: check=serfHealth
    2021-09-26T15: 08: 30.762+0800 [DEBUG] agent: Node info in sync
```

在看到上述内容后就可以访问 http://localhost:8500 地址，该地址是 Consul 的管理端界面，具体信息如图 1.3 所示。

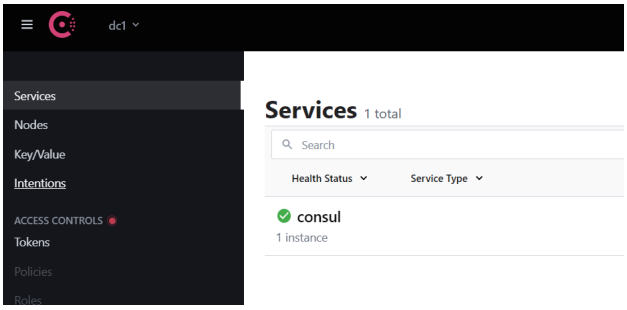


图 1.3　Consul 的管理端界面

通过前面的操作已经完成 Consul 的环境搭建，接下来将进入 Java 代码的编写，在 Spring Cloud Consul 源码工程中创建如下三个工程：

（1）consul-example 工程表示 Consul 实例工程；

（2）client-example 工程表示 Consul 客户端实例工程；

（3）server-example 工程表示 Consul 服务器端实例工程。

工程创建完成后 Spring Cloud Consul 测试工程结构如图 1.4 所示。

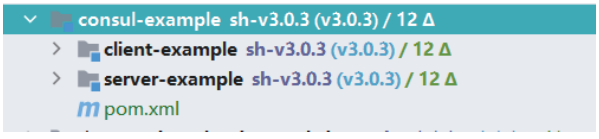


图 1.4　Spring Cloud Consul 测试工程结构

下面进行服务器端代码的编写。

（1）引入如下依赖：

```
<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>
```

（2）编写配置文件，配置文件名称为 application.properties，具体代码如下：

```
spring.application.name=spring-cloud-consul-producer
server.port=8501
spring.cloud.consul.host=localhost
spring.cloud.consul.port=8500
spring.cloud.consul.discovery.serviceName=consul-server-example
```

（3）编写服务器端启动类，类名为 ConsulProducerApplication，具体代码如下：

```
package com.github.huifer.cloud.consul;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
@EnableDiscoveryClient
public class ConsulProducerApplication {

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

（4）启动 ConsulProducerApplication，直接启动会出现如下内容：

```
    15: 39: 41.732 [main] ERROR org.springframework.boot.SpringApplication-Application
run failed
    java.lang.IllegalStateException: No subdirectories found for mandatory directory
location 'file: ./config/*/'.
        at org.springframework.util.Assert.state(Assert.java: 76)
        at org.springframework.boot.context.config.StandardConfigDataLocationResolver.
resolvePatternEmptyDirectories(StandardConfigDataLocationResolver.java: 299)
        at org.springframework.boot.context.config.StandardConfigDataLocationResolver.
resolveEmptyDirectories(StandardConfigDataLocationResolver.java: 288)
```

出现上述问题的原因是没有在 IDEA 编辑器中设置 Working directory，在 IDEA 中打开 Run/debug Configurations 选项卡，将数据信息设置为 application.properties 文件所在的路径。Spring Cloud Consul 启动配置如图 1.5 所示。

在设置完成后即可正常启动 ConsulProducerApplication，启动后可以在 Consul 控制台查看相关内容。Consul 管理端注册信息如图 1.6 所示。

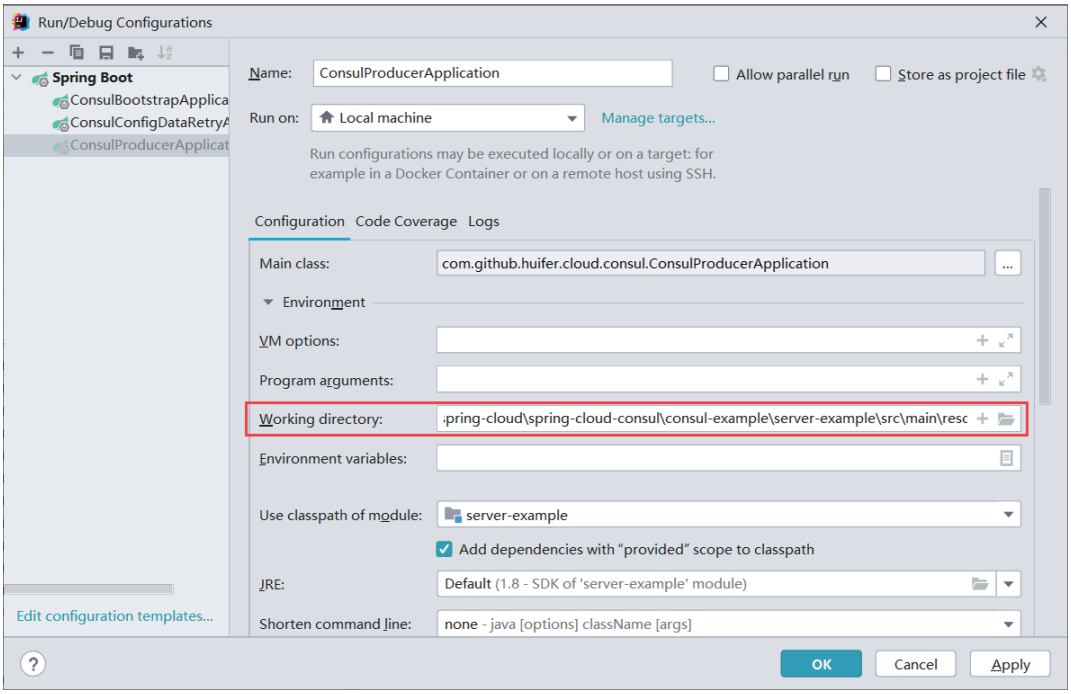


图 1.5 Spring Cloud Consul 启动配置

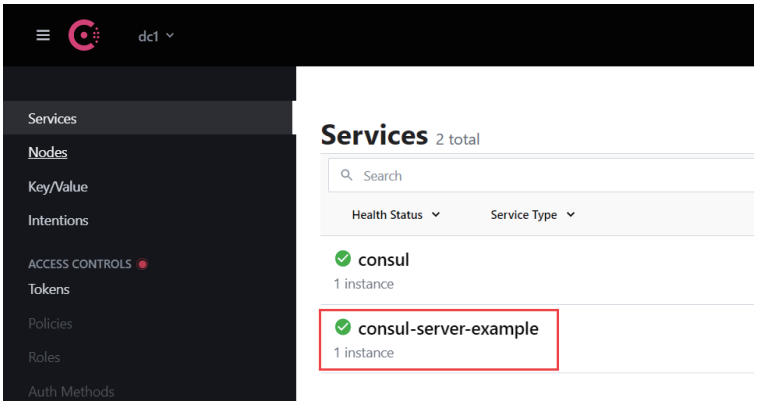


图 1.6 Consul 管理端注册信息

在 Consul 控制台中可以发现服务已经注册成功。下面进入客户端程序的编写。

（1）引入依赖，这里需要的依赖和服务器端的依赖一样，直接复制即可。

（2）编写应用配置，配置文件名称为 application.properties，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
spring.cloud.consul.discovery.register=true
```

（3）编写启动类，启动类类名为 ConsulConsumerApplication，详细代码如下：

```
package com.github.huifer.cloud.consul;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ConsulConsumerApplication {

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

在 ConsulConsumerApplication 的启动过程中会遇到与 ConsulProducerApplication 启动一样的问题，解决方式也是一样的。

（4）编写一个 Controller 对象，该对象的目标是用于获取服务实例，具体处理代码如下：

```
package com.github.huifer.cloud.consul;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.discovery.DiscoveryClient;
import org.springframework.cloud.client.loadbalancer.LoadBalancerClient;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ServiceController {

    @Autowired
    private LoadBalancerClient loadBalancer;
    @Autowired
    private DiscoveryClient discoveryClient;

    /**
     * 获取服务
     */
    @RequestMapping("/services")
    public Object services() {
        return discoveryClient.getInstances("consul-server-example");
    }

}
```

（5）启动 ConsulConsumerApplication，在 Consul 控制台查看数据，ConsulConsumerApplication 注册信息如图 1.7 所示。

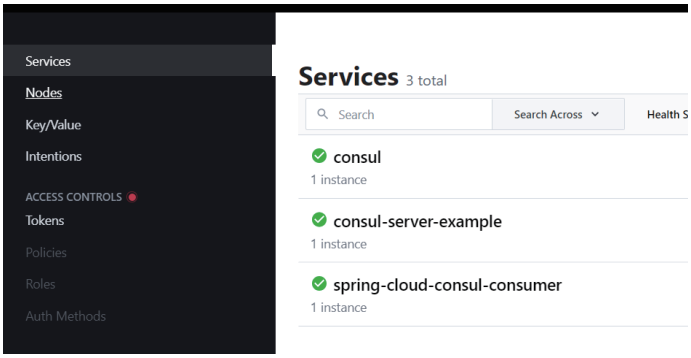


图 1.7　ConsulConsumerApplication 注册信息

可以发现 spring-cloud-consul-consumer 项目也成功在 Consul 上注册。最后进行接口测试，访问 http：//localhost：8503/services 接口查看数据信息，至此 Spring Cloud Consul 的源码阅读环境全部搭建完成。

本章小结

本章对 Spring Cloud Consul 项目的源码环境搭建进行了相关说明，从 GitHub 上的搜索开始到本地将源码导入 IDEA 中。此外，还进行了 Spring Cloud Consul 的测试，在测试阶段对 Consul 的下载、服务器端和客户端的使用进行了简单说明。