Spring Cloud Config是Spring Cloud团队创建的一个全新项目，用来为分布式系统中的基础设施和微服务应用提供集中化的外部配置支持，它分为服务端与客户端两个部分。其中服务端也称为分布式配置中心，它是一个独立的微服务应用，用来连接配置仓库并为客户端提供获取配置信息、加密/解密信息等访问接口；而客户端则是微服务架构中的各个微服务应用或基础设施，它们通过指定的配置中心来管理应用资源与业务相关的配置内容，并在启动的时候从配置中心获取和加载配置信息。Spring Cloud Config实现了对服务端和客户端中环境变量和属性配置的抽象映射，所以它除了适用于Spring构建的应用程序之外，也可以在任何其他语言运行的应用程序中使用。由于Spring Cloud Config实现的配置中心默认采用Git来存储配置信息，所以使用Spring Cloud Config构建的配置服务器，天然就支持对微服务应用配置信息的版本管理，并且可以通过Git客户端工具来方便的管理和访问配置内容。当然它也提供了对其他存储方式的支持，比如：SVN仓库、本地化文件系统。

在本文中，我们将学习如何构建一个基于Git存储的分布式配置中心，并对客户端进行改造，并让其能够从配置中心获取配置信息并绑定到代码中的整个过程。

准备配置仓库

- 准备一个git仓库，可以在码云或Github上创建都可以。比如本文准备的仓库示例：https://gitee.com/cuilijian/SpringCloudConfig
- 假设我们读取配置中心的应用名为config-client，那么我们可以在git仓库中该项目的默认配置文件config-client.yml：

```
info:
  profile: default
```

- 为了演示加载不同环境的配置，我们可以在git仓库中再创建一个针对dev环境的配置文件config-client-dev.yml：

```
info:
  profile: dev
```

构建配置中心

通过Spring Cloud Config来构建一个分布式配置中心非常简单，只需要三步：

- 创建一个基础的Spring Boot工程，命名为：config-server，并在pom.xml中引入下面的依赖（省略了parent和dependencyManagement部分）：

```
<dependencies>
        <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-config-server</artifactId>
        </dependency>
</dependencies>
```

- 创建Spring Boot的程序主类，并添加@EnableConfigServer注解，开启Spring Cloud Config的服务端功能。

```
@EnableConfigServer
@SpringBootApplication
public class ConfigServerGitApplication {
    public static void main(String[] args) {
        new SpringApplicationBuilder(ConfigServerGitApplication.class).web(true).run(args);
    }
}
```

- 在application.properties中添加配置服务的基本信息以及Git仓库的相关信息：

```
server.port=2201
spring.application.name=config-server
spring.cloud.config.server.git.uri=https://gitee.com/cuilijian/SpringCloudConfig
spring.cloud.config.server.git.searchPaths=respo
spring.cloud.config.label=master
spring.cloud.config.server.git.username=
spring.cloud.config.server.git.password=
```

- spring.cloud.config.server.git.uri：配置git仓库地址
- spring.cloud.config.server.git.searchPaths：配置仓库路径

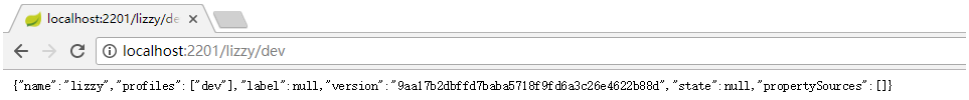
- spring.cloud.config.label：配置仓库的分支
- spring.cloud.config.server.git.username：访问git仓库的用户名
- spring.cloud.config.server.git.password：访问git仓库的用户密码

如果Git仓库为公开仓库，可以不填写用户名和密码，如果是私有仓库需要填写，本例子是公开仓库，放心使用。远程仓库https://gitee.com/cuilijian/SpringCloudConfig中有个文件config-client-dev.properties文件中有一个属性：

```
lizzy = This is a SpringCloud config center
spring.cloud.config.address= This is a config center
```

启动程序：访问http://localhost:2201/lizzy/dev 返回如下信息：

{"name":"lizzy","profiles":
["dev"],"label":null,"version":"9aa17b2dbffd7baba5718f9fd6a3c26e4622b88d","state":null,"propertySources":
[]}



证明配置服务中心可以从远程程序获取配置信息。

http请求地址和资源文件映射如下：

- /{application}/{profile}[/{label}]
- /{application}-{profile}.yml
- /{label}/{application}-{profile}.yml
- /{application}-{profile}.properties
- /{label}/{application}-{profile}.properties

构建客户端

在完成了上述验证之后，确定配置服务中心已经正常运作，下面我们尝试如何在微服务应用中获取上述的配置信息。

- 创建一个Spring Boot应用，命名为config-client，并在pom.xml中引入下述依赖：

```
<dependencies>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-starter-config</artifactId>
        </dependency>
</dependencies>
```

- 创建Spring Boot的应用主类，程序的入口类，写一个API接口“/hi”，返回从配置中心读取变量的值，代码如下：

```
@SpringBootApplication
@RestController
public class ConfigClientApplication {
    public static void main(String[] args) {
        new SpringApplicationBuilder(ConfigClientApplication.class).web(true).run(args);
    }
    @Value("${lizzy}")
    String lizzy;

    @Value("${spring.cloud.config.address}")
    String address;

    @RequestMapping(value = "/hi")
    public String hi(){
        return lizzy + "_" + address;
```

```
    }
}
```

- 创建**bootstrap.properties**配置文件：

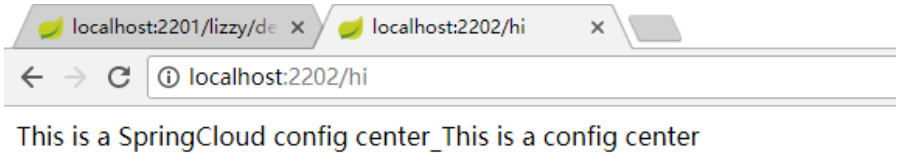
```
spring.application.name=config-client
spring.cloud.config.label=master
spring.cloud.config.profile=dev
spring.cloud.config.uri= http://localhost:2201/
server.port=2202
```

上述配置参数与Git中存储的配置文件中各个部分的对应关系如下：

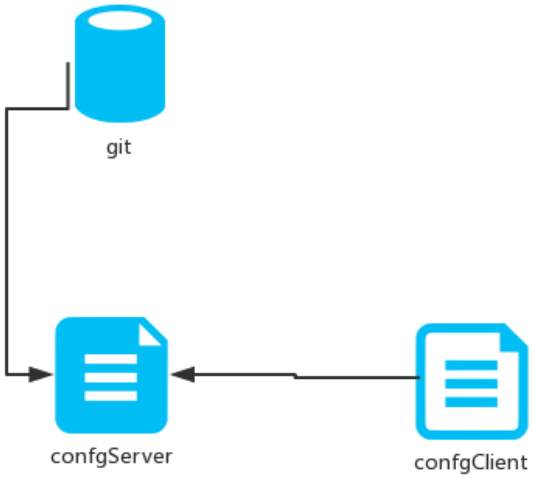
- spring.application.name：对应配置文件规则中的{application}部分
- spring.cloud.config.profile：对应配置文件规则中的{profile}部分
- spring.cloud.config.label：对应配置文件规则中的{label}部分
- spring.cloud.config.uri：配置中心config-server的地址

这里需要格外注意：上面这些属性必须配置在bootstrap.properties中，这样config-server中的配置信息才能被正确加载。

在完成了上面代码编写之后，将config-server、config-client都启动起来，然后访问http://localhost:2202/hi，我们可以看到该端点将会返回从git仓库中获取的配置信息：



这就说明，config-client从config-server获取了lizzy的属性，而config-server是从git仓库读取的,如图：



具体工程说明如下：

- 基于Git仓库的配置中心：config-server
- 使用配置中心的客户端：config-client