

SpringCloud之七

分布式配置中心SpringCloudConfig

一、Spring Cloud Config 介绍

微服务常用的招式已经有了：

- 注册中心 (eureka, consul, zk, etcd)
- 配置中心 (Spring Cloud Config, disconf)
- API网关 (Spring Cloud zuul, kong)
- 熔断器 (hystrix)

那么还需要什么呢？当然就是一个行走江湖，说走就走的理由了，俗话说的号，师出有名，接下来要介绍的就是，SpringCloud的配置中心解决方案——Spring Cloud Config。

在官网上截图如下：



它支持配置服务放在配置服务的内存中（即本地），也支持放在远程Git，SVN等仓库。

在Spring Cloud Config组件中，分两个角色，一是**config server**，二是**config client**

Spring Cloud Config，主要有以下**特性**

- 配置集中管理
- 依赖于git支持版本回退
- 可以通过git 仓库的WebHook和消息队列来实现自动更新

- 配置加密解密
- 可以通过调用接口手动更新配置

其他配置中心技术还有就是国内的disconf, disconf 可以为各种业务平台提供统一的**配置管理服务**。

- 支持配置（配置项+配置文件）的分布式化管理
- 配置发布统一化
- 极简的使用方式（注解式编程 或 XML代码无代码侵入模式）
- 低侵入性 or 无侵入性、强兼容性
- 需要Spring编程环境

虽然有很多选择，如果说是Spring Cloud来构建微服务，那么Spring Cloud Config肯定是比较好的一个选择，集成非常方便。

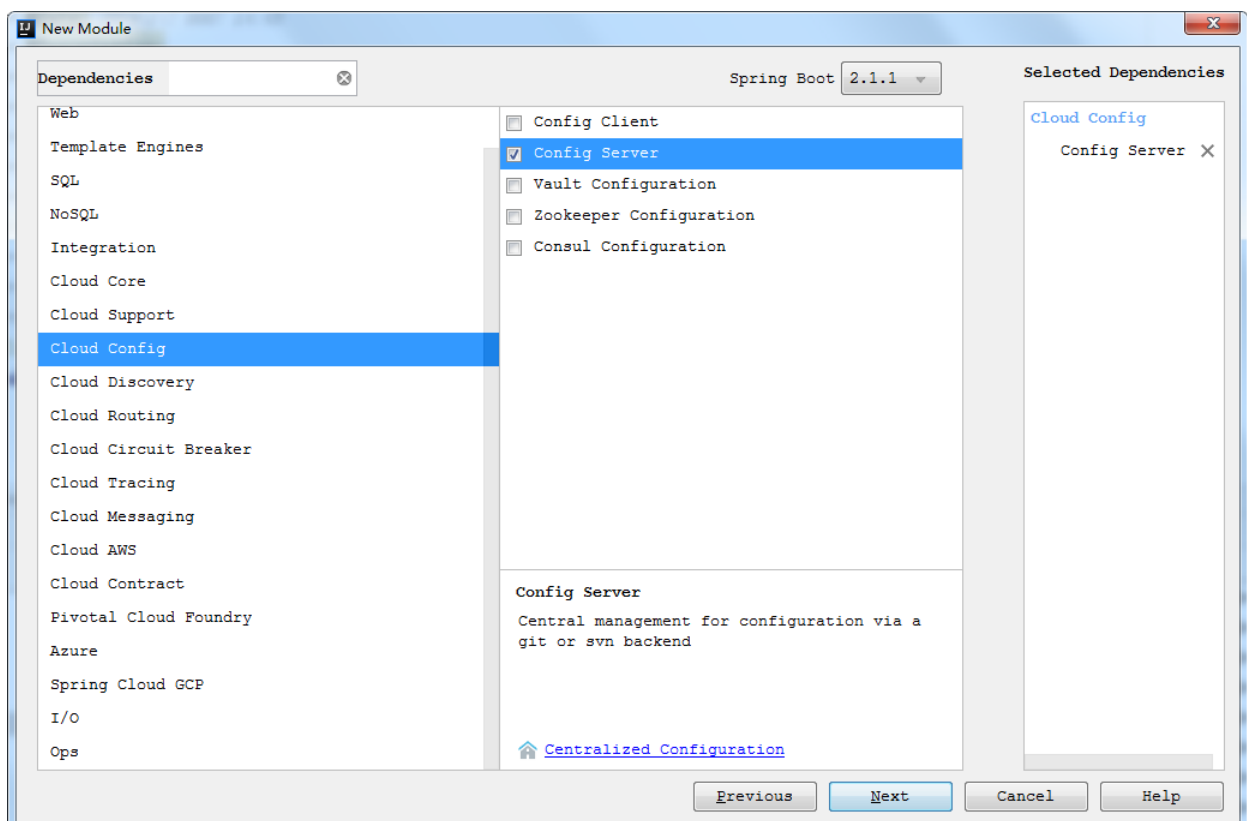
而且disconf虽然也是一个很好的作品，但是也是个人开源的，也不维护了，也许功能够用了。有时间得话，觉得可以个人再重新写一个，那样会更好。

二、构建Config Server

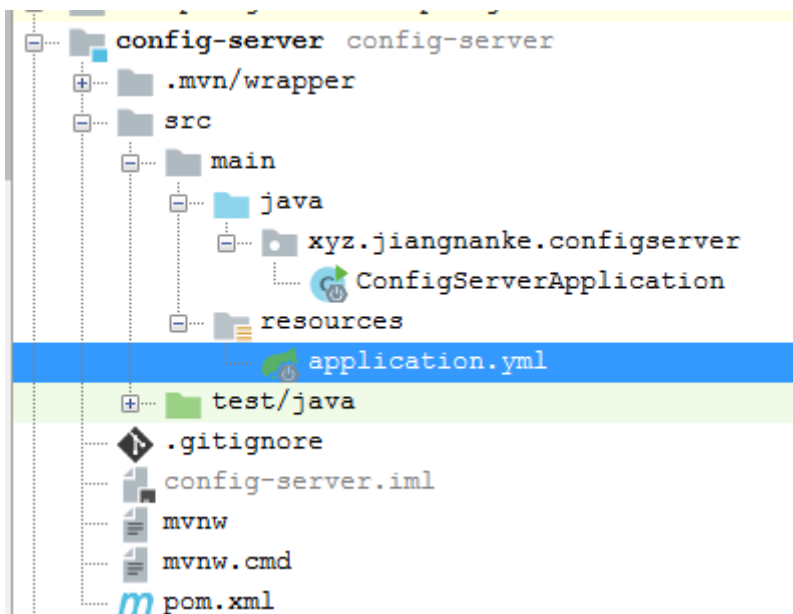
2.1、创建项目

创建一个Config server的spring-boot项目，我将之命名为 config-server。

如图：



创建好后，如图结构：



2.2、配置环境

配置pom.xml文件，当然也要修改main项目的pom.xml文件，添加上子模块（这里不多介绍），如下是config-server的pom.xml文件内容：

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.
  w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apach
  e.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <groupId>xyz.jiangnanke</groupId>
6   <artifactId>config-server</artifactId>
7   <version>0.0.1-SNAPSHOT</version>
8   <name>config-server</name>
9   <description>Demo project for Spring Boot</description>
10
11   <parent>
12     <groupId>xyz.jiangnanke</groupId>
13     <artifactId>main</artifactId>
14     <version>0.0.1-SNAPSHOT</version>
15   </parent>
16
17   <dependencies>
18     <dependency>
19       <groupId>org.springframework.cloud</groupId>
20       <artifactId>spring-cloud-config-server</artifactId>
```

```
21 </dependency>
22 </dependencies>
23
24 </project>
```

然后再application启动类添加上@EnableConfigServer注解来启动config服务。
代码如下

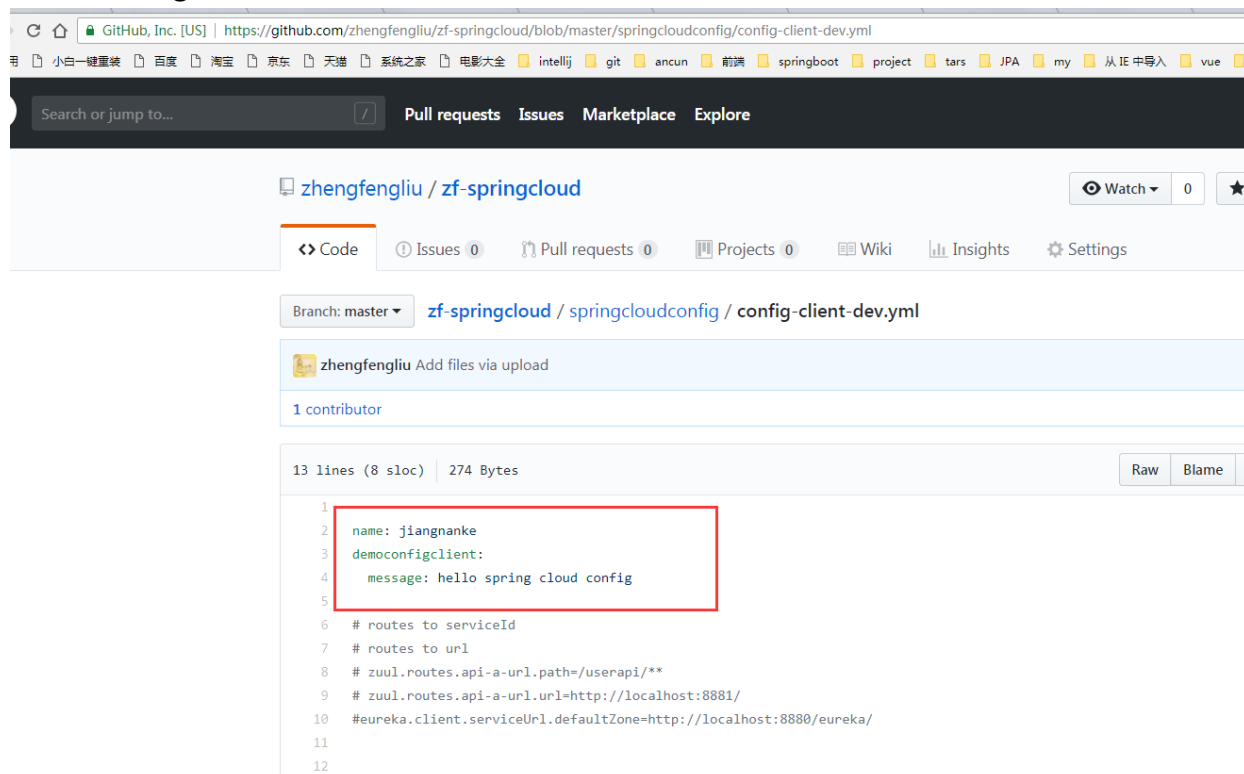
```
1 package xyz.jiangnanke.configserver;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.config.server.EnableConfigServer;
6
7 @SpringBootApplication
8 @EnableConfigServer
9 public class ConfigServerApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(ConfigServerApplication.class, args);
13     }
14
15 }
16
```

再在application.yml配置文件添加如下内容：

```
1 spring:
2   application:
3     name: config-server
4   cloud:
5     config:
6       server:
7         git:
8           # 配置git仓库地址
9           uri: https://github.com/zhengfengliu/zf-springcloud/
10          # 配置仓库路径
11          search-paths: springcloudconfig
12          # 访问git仓库的用户名,密码
13          username: zhengfeng
14          password: zhengfeng
15          # 配置仓库的分支
16          label: master
```

```
17 # 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。
18
19 server:
20   port: 10089
21
```

然后再配置git里面添加上：

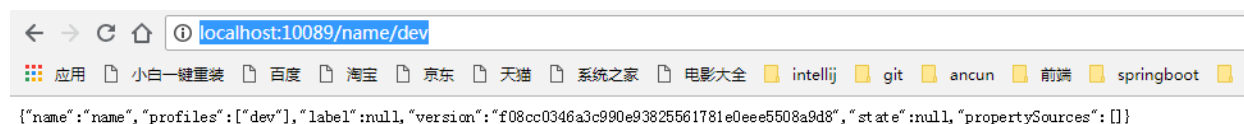


2.3、启动运行

启动项目，由于远程仓库git上config-client-dev.yml上有内容，接下来访问：

<http://localhost:10089/name/dev>

得到结果如下图：



证明配置服务中心可以从远程程序获取配置信息。

当然**http请求地址和资源文件映射**如下：

`/ {application} / {profile} [/ {label}]`

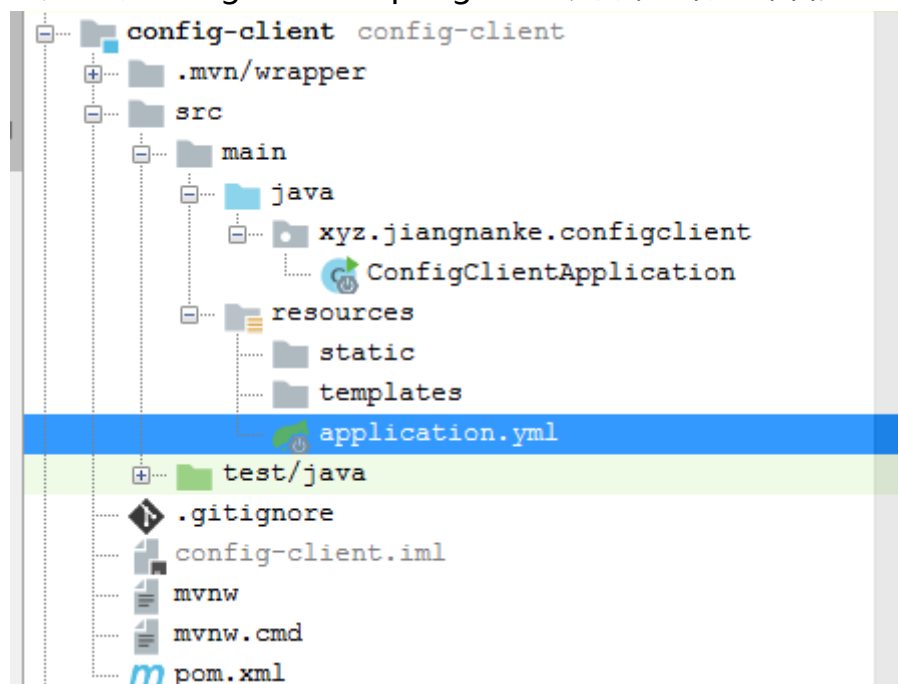
`/ {application} - {profile} . yml`

```
/{label}/{application}-{profile}.yml  
/{application}-{profile}.properties  
/{label}/{application}-{profile}.properties
```

三、构建Config Client

3.1、创建ConfigClient项目

创建一个Config Client的Spring-boot项目，我将之命名为config-client。如下：



其pom.xml配置文件如下：

```
1 <?xml version="1.0" encoding="UTF-8"?>  
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.  
w3.org/2001/XMLSchema-instance"  
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apach  
e.org/xsd/maven-4.0.0.xsd">  
4   <modelVersion>4.0.0</modelVersion>  
5   <groupId>xyz.jiangnanke</groupId>  
6   <artifactId>config-client</artifactId>  
7   <version>0.0.1-SNAPSHOT</version>  
8   <name>config-client</name>  
9   <description>Demo project for Spring Boot</description>  
10  
11   <parent>  
12     <groupId>xyz.jiangnanke</groupId>  
13     <artifactId>main</artifactId>
```

```
14 <version>0.0.1-SNAPSHOT</version>
15 </parent>
16
17 <dependencies>
18 <dependency>
19 <groupId>org.springframework.boot</groupId>
20 <artifactId>spring-boot-starter-web</artifactId>
21 </dependency>
22 <dependency>
23 <groupId>org.springframework.cloud</groupId>
24 <artifactId>spring-cloud-starter-config</artifactId>
25 </dependency>
26
27 </dependencies>
28
29 </project>
```

3.2、配置环境

首先配置application.yml文件，如下：

```
1 spring:
2   application:
3     name: config-client
4   cloud:
5     config:
6       # 指明远程仓库的分支
7       label: master
8       # spring.cloud.config.profile
9       # dev开发环境配置文件
10      # test测试环境
11      # pro正式环境
12      profile: dev
13      # 指明配置服务中心的网址。
14      uri: http://localhost:10089/
15    server:
16      port: 10090
```

接下来创建一个程序入口类LoginController.java。具体代码如下：

```
1 package xyz.jiangnanke.configclient.controller;
2
```

```

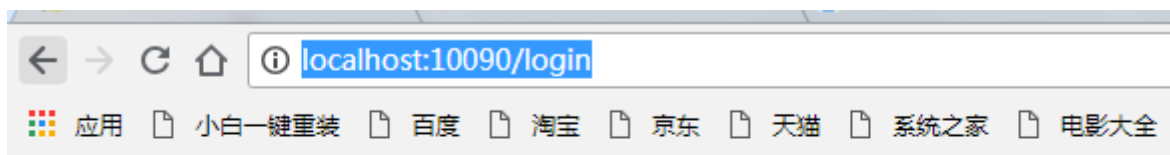
3 import org.springframework.beans.factory.annotation.Value;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RestController;
6
7 /**
8  * @Auther: zhengfeng
9  * @Date: 2019\1\7 0007 17:32
10  * @Description: 返回从配置中心读取的name变量的值
11  */
12 @RestController
13 public class LoginController {
14
15     @Value("${name}")
16     String name;
17
18     @RequestMapping(value = "/login")
19     public String login(){
20         return name;
21     }
22 }

```

3.3、启动运行

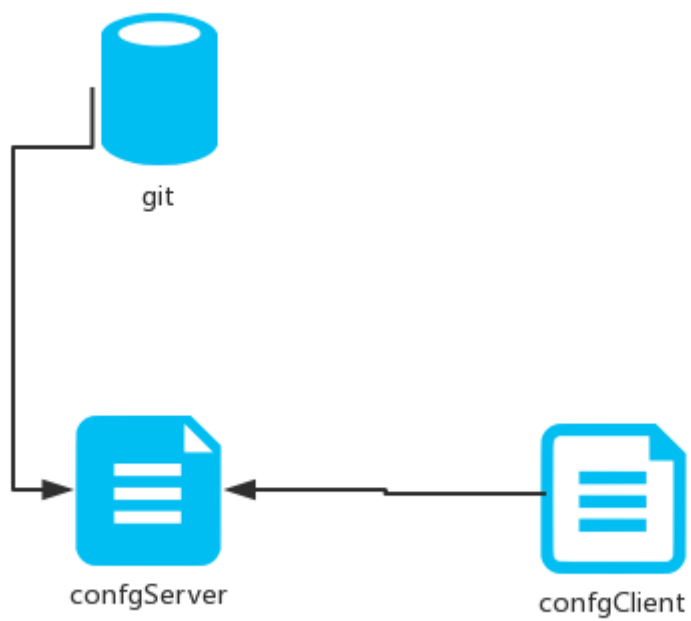
启动运行之后，访问网址：<http://localhost:10090/login>

得到结果，如图：

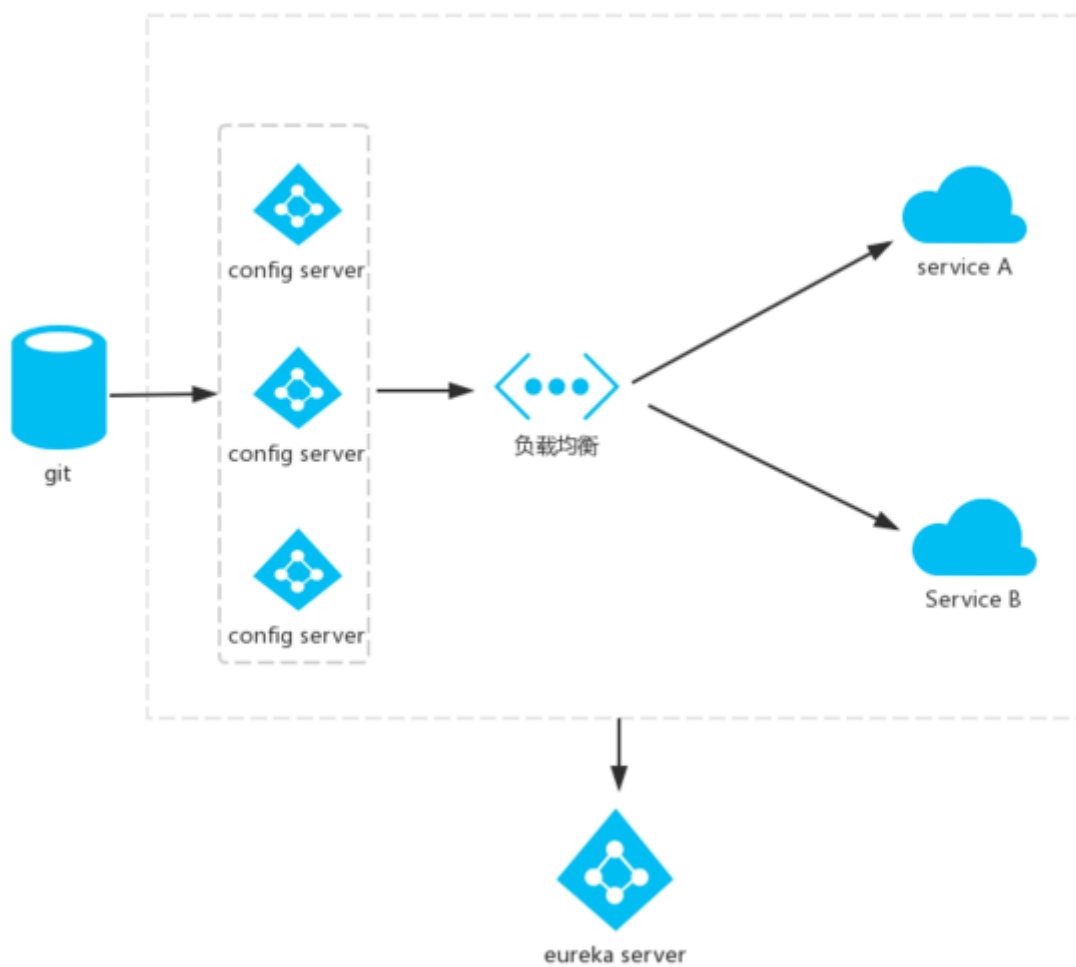


jiangnanke

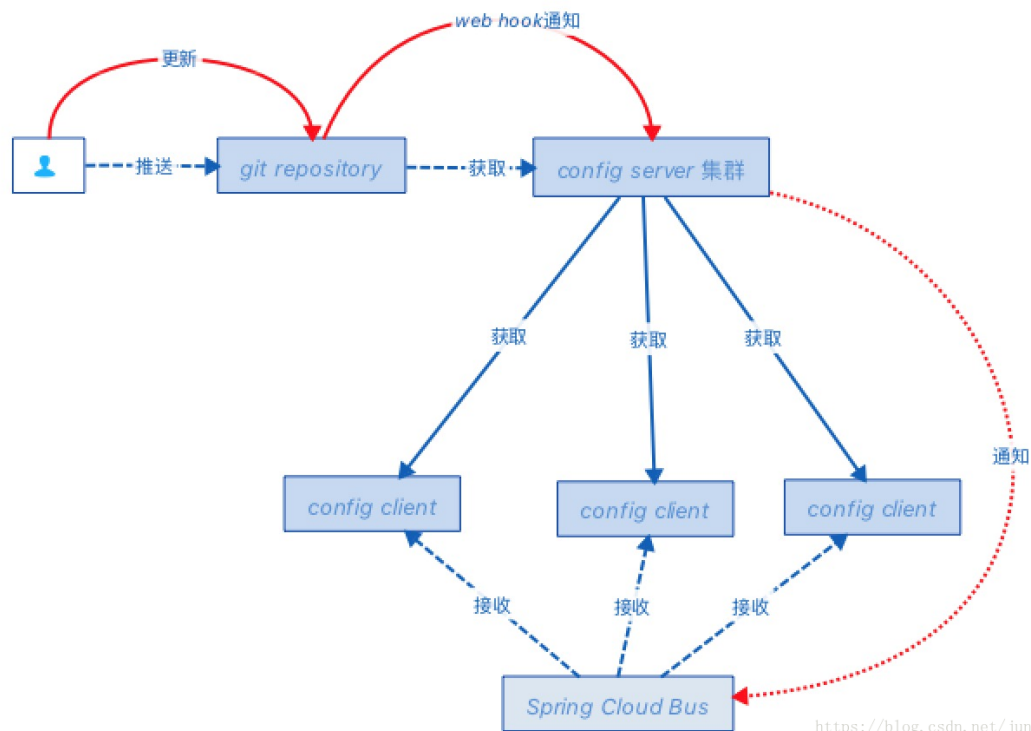
这就说明，config-client从config-server获取了 name 的属性，而config-server是从git仓库读取的,如图：



config-server在微服务中的定位：



config-client在微服务中的定位：

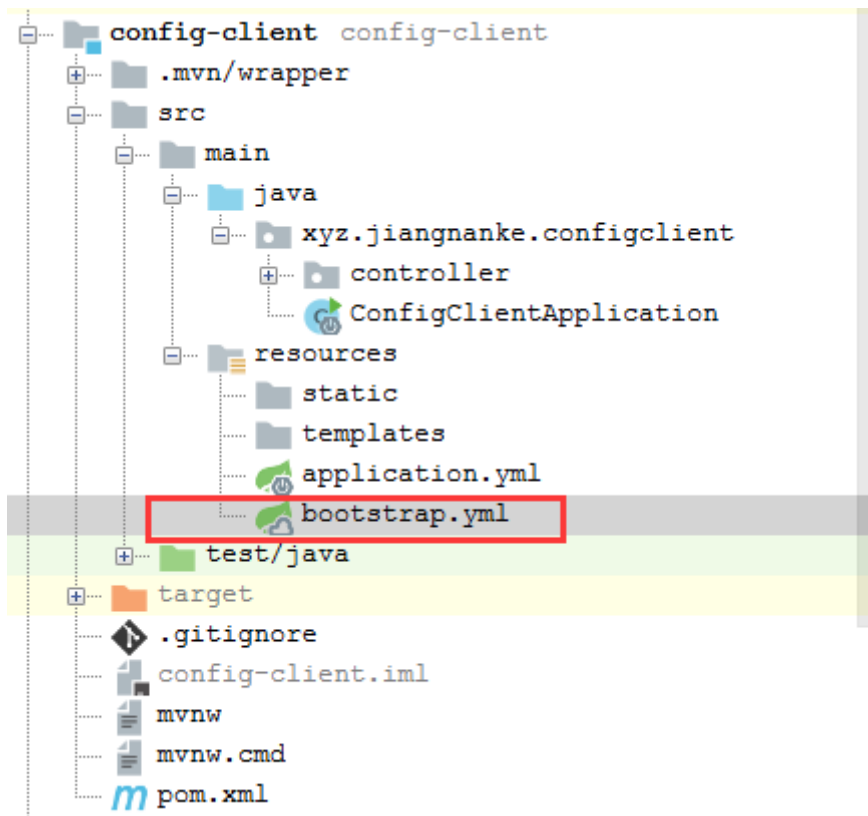


主要实现需要以下几个步骤：

1. 创建git仓库。用于存放配置文件。
2. 安装kafka，zookeeper。用于分布式项目统一刷新。
3. 创建ConfigServer服务，提供项目统一获取配置的中心服务。可配多个。
4. 创建ConfigClient服务，作为客户端。
5. 利用gitlab的webhook功能，每次push操作，发送post请求，通知刷新配置。

附录

一定要注意，config-client 的配置信息是放在，bootstrap.yml 文件中的 或者 bootstrap.properties文件中。不然启动会报错。修改之后的项目结构如图：



四、改造spring cloud config 与eureka 结合

4.1、改造config-server项目

4.1.1、添加依赖

```
1 <dependency>
2 <groupId>org.springframework.cloud</groupId>
3 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
4 </dependency>
```

4.1.2、添加环境配置

```
1 eureka:
2   client:
3     serviceUrl:
4       defaultZone: http://localhost:8761/eureka/
```

4.1.3、添加启动配置

```
1 package xyz.jiangnanke.configserver;
2
3 import org.springframework.boot.SpringApplication;
```

```

4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.config.server.EnableConfigServer;
6 import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
7
8 @SpringBootApplication
9 @EnableConfigServer
10 @EnableEurekaClient
11 public class ConfigServerApplication {
12
13     public static void main(String[] args) {
14         SpringApplication.run(ConfigServerApplication.class, args);
15     }
16
17 }
18

```

启动之后如图：

The screenshot shows the Spring Eureka web interface running on localhost:8761. The page has a dark header with the 'spring Eureka' logo and navigation links for 'HOME' and 'LAST 1000 SINCE STARTED'. Below the header, the 'System Status' section displays a table with system information:

System Status	
Environment	test
Data center	default
Current time	2019-01-10 04:01:10
Uptime	04:01:10
Lease expiration enabled	false
Renews threshold	11
Renews (last min)	10

Below the status table, a red warning message is displayed: "EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING DELETED. PLEASE RENEW THEM MANUALLY OR CHECK THE LOGS FOR MORE DETAILS."

The 'DS Replicas' section shows the instances currently registered with Eureka:

Application	AMIs	Availability Zones	Status
CONFIG-SERVER	n/a (1)	(1)	UP (1) - zhengfeng.mshome.net:config-server:10089
FEIGN-SERVICE	n/a (1)	(1)	UP (1) - zhengfeng.mshome.net:feign-service:8765
LOGIN-SERVICE	n/a (2)	(2)	UP (2) - zhengfeng.mshome.net:login-service:8762, zhengfeng.mshome.net:login-service:8763
RIBBON-SERVICE	n/a (1)	(1)	UP (1) - zhengfeng.mshome.net:ribbon-service:8764
ZUUL-SERVICE	n/a (1)	(1)	UP (1) - zhengfeng.mshome.net:zuul-service:8769

4.2、改造config-client项目

基本上如config-server一样修改

启动后如图：

← → ↻ 🏠 localhost:8761

应用 小白一键重装 百度 淘宝 京东 天涯 系统之家 电影大全 intelliJ git ancun 前阵 springboot project tars JPA my 从IE中导入 vue JWT jenkins bigdat

springEureka

HOME LAST 1000 SINCE STARTUP

System Status

Environment	test	Current time	2019-01-07T18:21:57
Data center	default	Uptime	04:04
		Lease expiration enabled	false
		Renews threshold	13
		Renews (last min)	12

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES SAFE.

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CONFIG-CLIENT	n/a (1)	(1)	UP (1) - zhengfeng.mshome.net:config-client:10090
CONFIG-SERVER	n/a (1)	(1)	UP (1) - zhengfeng.mshome.net:config-server:10089
FEIGN-SERVICE	n/a (1)	(1)	UP (1) - zhengfeng.mshome.net:feign-service:8765
LOGIN-SERVICE	n/a (2)	(2)	UP (2) - zhengfeng.mshome.net:login-service:8762 , zhengfeng.mshome.net:login-service:8763
RIBBON-SERVICE	n/a (1)	(1)	UP (1) - zhengfeng.mshome.net:ribbon-service:8764
ZUUL-SERVICE	n/a (1)	(1)	UP (1) - zhengfeng.mshome.net:zuul-service:8769

General Info

当然如果可以的话，其实一开始创建项目就直接加上这些配置，就不用现在继续改造这个过程了。

参考资料

https://blog.csdn.net/liqi_q/article/details/81158002
<https://springcloud.cc/spring-cloud-config.html>
<https://springcloud.cc/>
<https://blog.csdn.net/forezp/article/details/81041028>
<https://github.com/zhengfengl原因/zf-springcloud>
<http://cxytiandi.com/blog/detail/12466>
<https://blog.csdn.net/forezp/article/details/81041045>