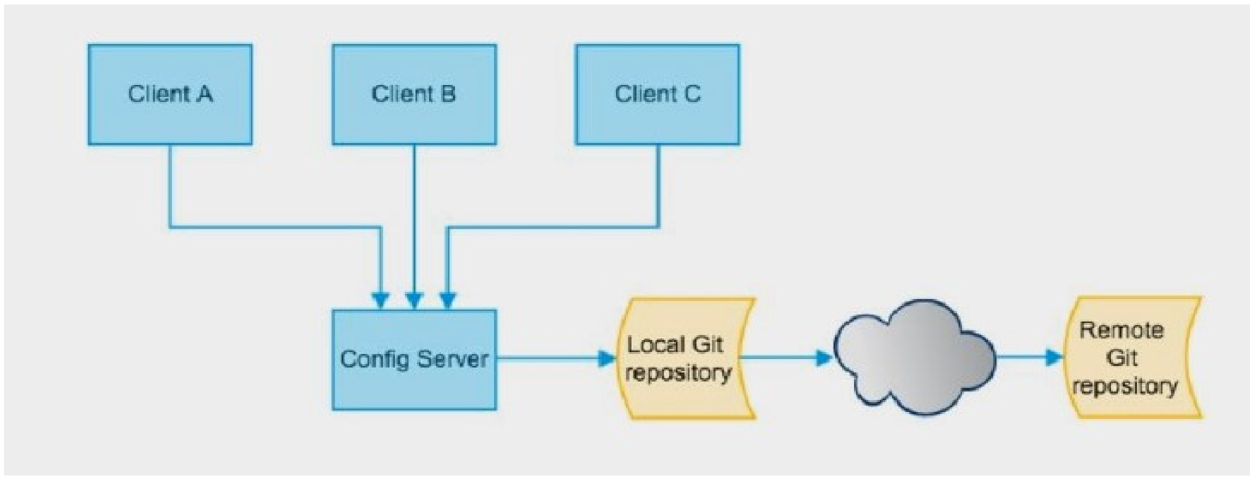
1. 概述

1.1. 分布式系统面临的配置问题

微服务意味着要将单体应用中的业务拆分成一个个子服务，每个服务的粒度相对较小，因此系统中会出现大量的服务。由于每个服务都需要必要的配置信息才能运行，所以一套集中式的、动态的配置管理设施是必不可少的。

Spring Cloud提供了 ConfigServer来解决这个问题，否则我们每个微服务自己带着一个application.yml，上百个配置文件的管理...

1.2. 是什么



是什么

- Spring Cloud Config为微服务架构中的微服务提供集中化的外部配置支持，配置服务器为各个不同微服务应用的所有环境提供了一个中心化的外部配置

怎么玩

- Spring Cloud Config分为服务端和客户端两部分。
- 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，用来连接配置服务器并为客户端提供获取配置信息，加密/解密信息等访问接口
- 客户端则是通过指定的配置中心来管理应用资源，以及与业务相关的配置内容，并在启动的时候从配置中心取和加载配置信息.配置服务器默认采用Git来存储配置信息，这样就有助于对环境配置进行版本管理，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

1.3. 能干嘛

集中管理配置文件

不同环境不同配置，动态化的配置更新，分环境部署比如dev/test/prod/beta/release

运行期间动态调整配置，不再需要在每个服务部署的机器上编写配置文件，服务会向配置中心统一拉取配置自己的信息

当配置发生变动时，服务不需要重启即可感知到配置的变化并应用新的配置

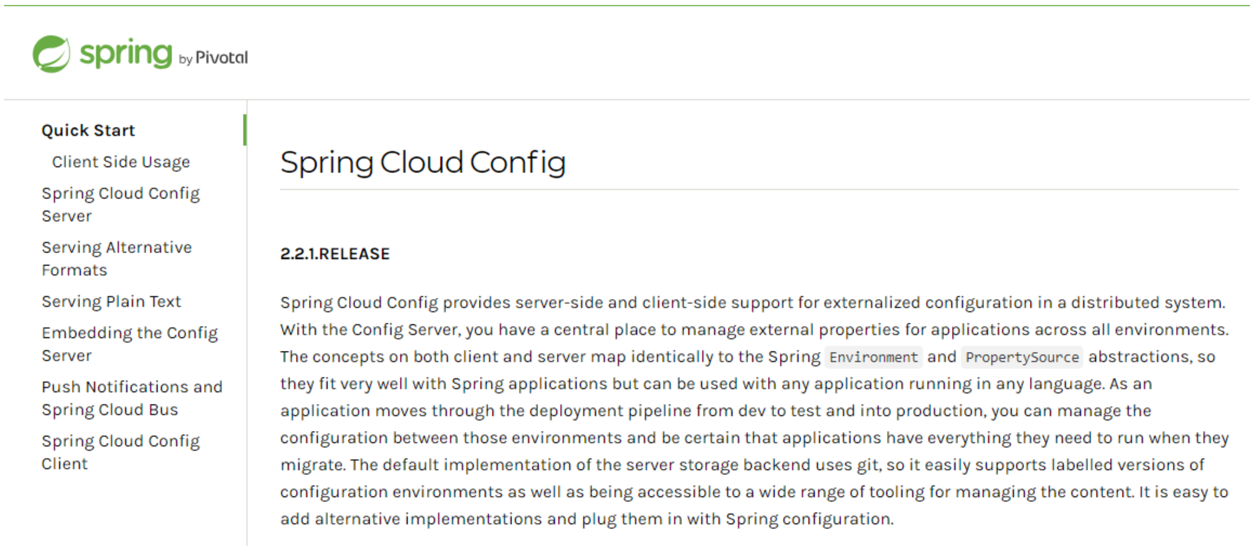
将配置信息以REST接口的形式暴露:post、curl访问刷新均可....

1.4. 与Github整合配置

由于SpringCloud Config默认使用Git来存储配置文件（也有其它方式，比如支持svn和本地文件，但最推荐的还是Git，而且使用的是http/https访问的形式）

1.5. 官网

> https://cloud.spring.io/spring-cloud-static/spring-cloud-config/2.2.1.RELEASE/reference/html/


spring by Pivotal
Quick Start
Client Side Usage
Spring Cloud Config Server
Serving Alternative Formats
Serving Plain Text
Embedding the Config Server
Push Notifications and Spring Cloud Bus
Spring Cloud Config Client

Spring Cloud Config

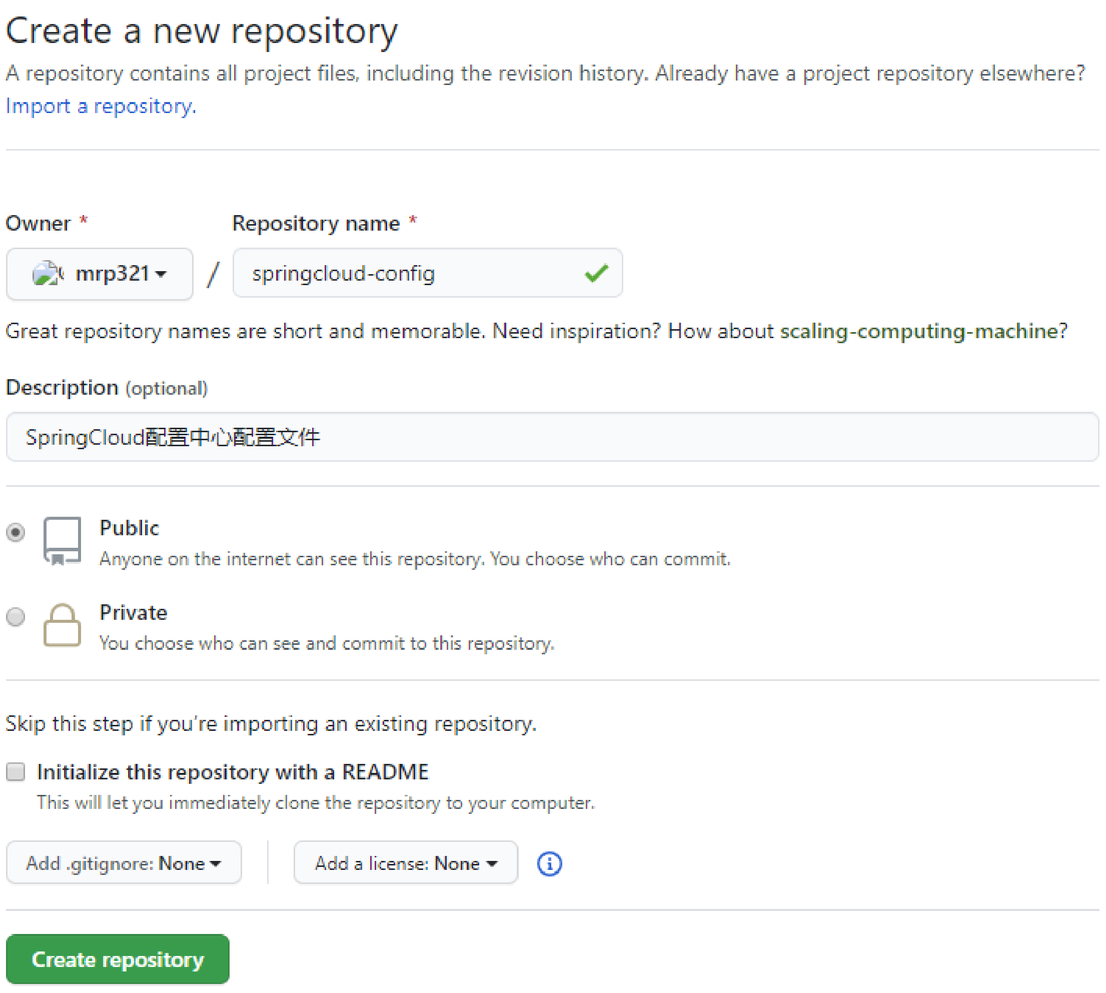
2.2.1.RELEASE

Spring Cloud Config provides server-side and client-side support for externalized configuration in a distributed system. With the Config Server, you have a central place to manage external properties for applications across all environments. The concepts on both client and server map identically to the Spring `Environment` and `PropertySource` abstractions, so they fit very well with Spring applications but can be used with any application running in any language. As an application moves through the deployment pipeline from dev to test and into production, you can manage the configuration between those environments and be certain that applications have everything they need to run when they migrate. The default implementation of the server storage backend uses git, so it easily supports labelled versions of configuration environments as well as being accessible to a wide range of tooling for managing the content. It is easy to add alternative implementations and plug them in with Spring configuration.

2. Config服务端配置与测试

2.1. Github远程仓库准备

2.1.1. 在Github上新建一个名为springcloud-config的新Repository



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * mrp321 / Repository name * springcloud-config

Great repository names are short and memorable. Need inspiration? How about scaling-computing-machine?

Description (optional): SpringCloud配置中心配置文件

(●) Public — Anyone on the internet can see this repository. You choose who can commit.

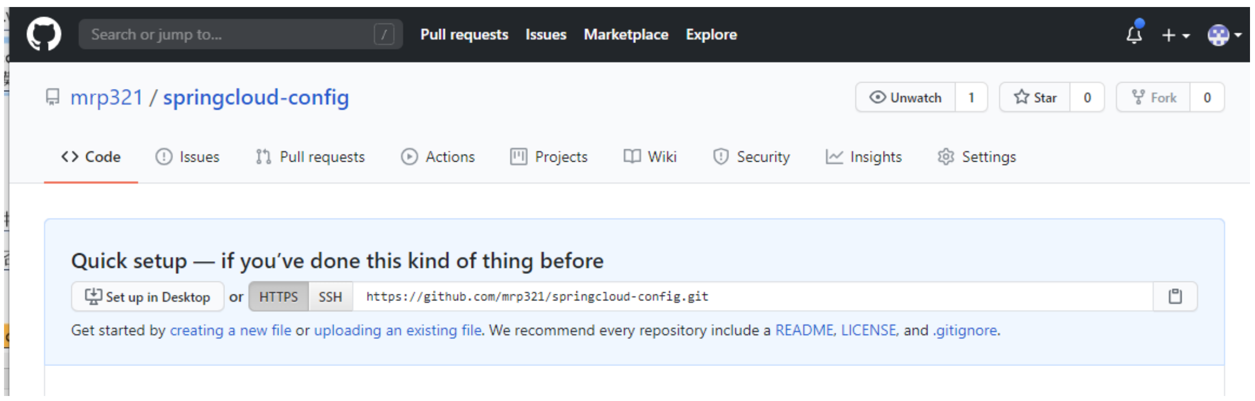
( ) Private — You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

[ ] Initialize this repository with a README — This will let you immediately clone the repository to your computer.

Add .gitignore: None | Add a license: None

Create repository



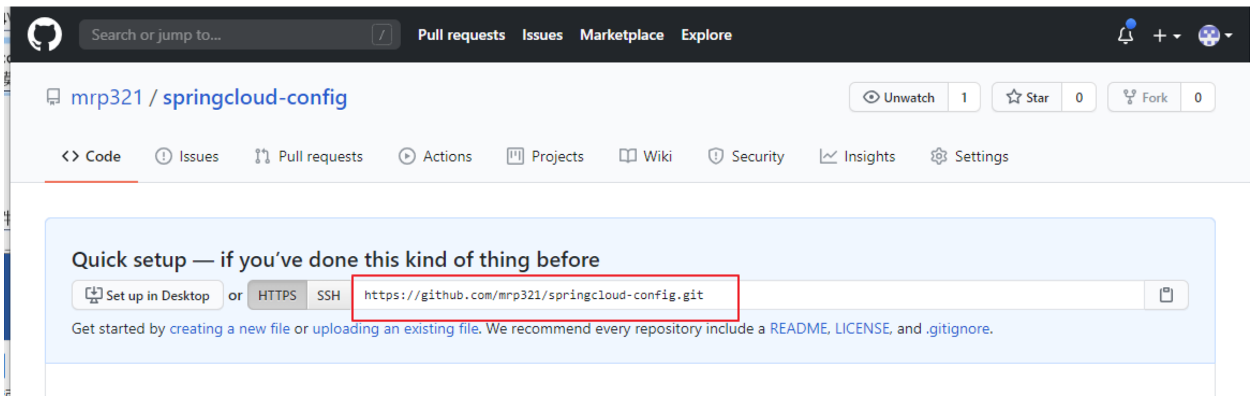
mrp321 / springcloud-config

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/mrp321/springcloud-config.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

2.1.2. 由上一步获得刚新建的git地址



mrp321 / springcloud-config

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/mrp321/springcloud-config.git`

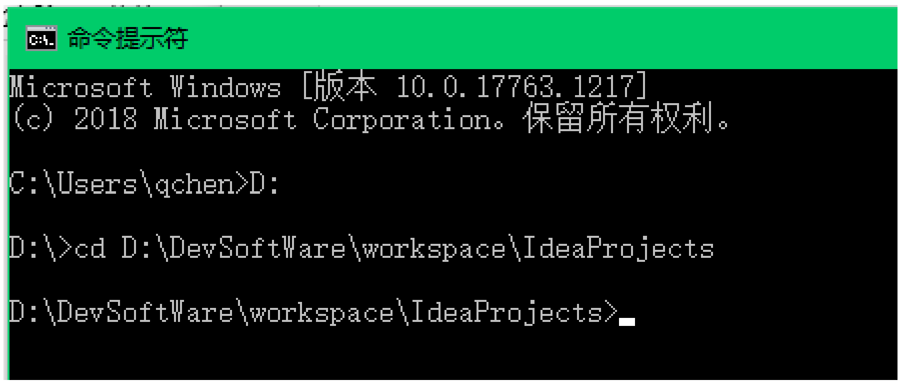
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

https://github.com/mrp321/springcloud-config.git

2.1.3. 本地硬盘上新建git仓库并clone

打开命令行, cd到路径D:\DevSoftWare\workspace\IdeaProjects下

```
C:\Users\qchen>D:

D:\>cd D:\DevSoftWare\workspace\IdeaProjects

D:\DevSoftWare\workspace\IdeaProjects>
```

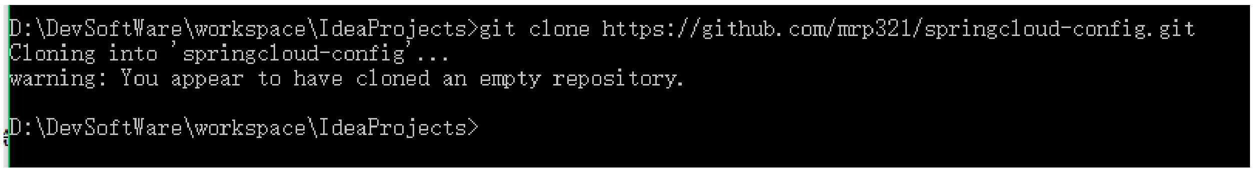


命令提示符

```
Microsoft Windows [版本 10.0.17763.1217]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\qchen>D:

D:\>cd D:\DevSoftWare\workspace\IdeaProjects

D:\DevSoftWare\workspace\IdeaProjects>
```

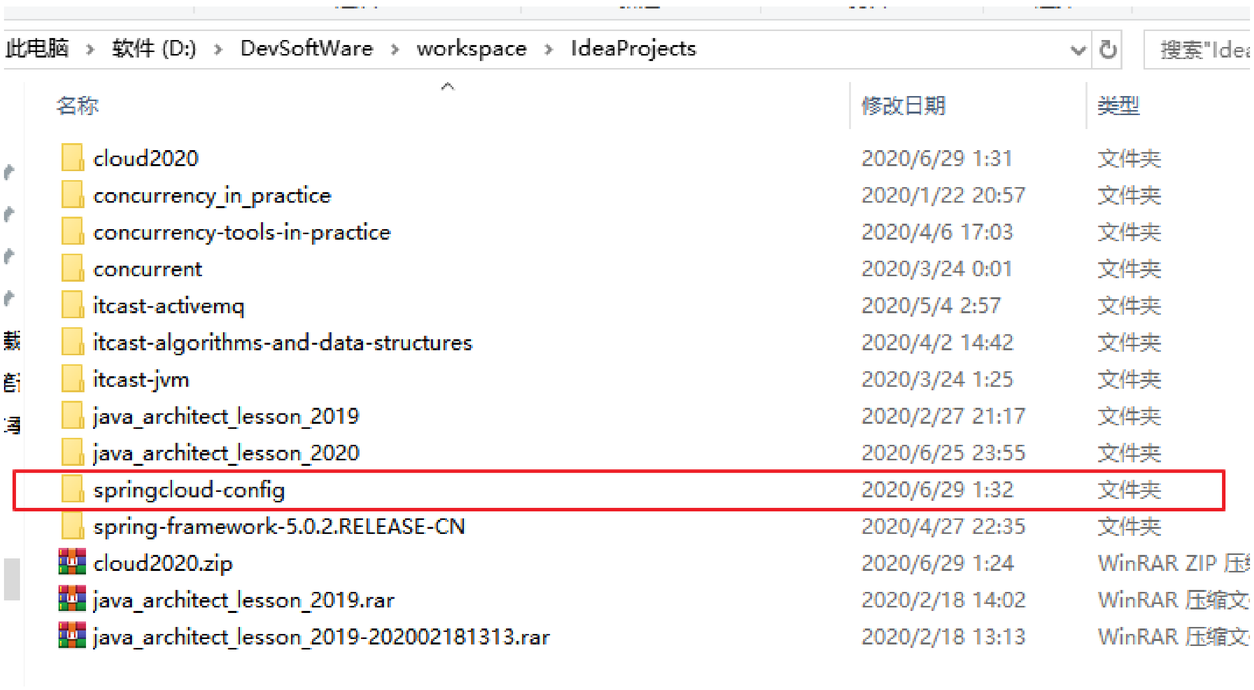
执行clone命令

```
D:\DevSoftWare\workspace\IdeaProjects>git clone https://github.com/mrp321/springclou
```



```
D:\DevSoftWare\workspace\IdeaProjects>git clone https://github.com/mrp321/springcloud-config.git
Cloning into 'springcloud-config'...
warning: You appear to have cloned an empty repository.

D:\DevSoftWare\workspace\IdeaProjects>
```

可以看到远程仓库已经clone到本地

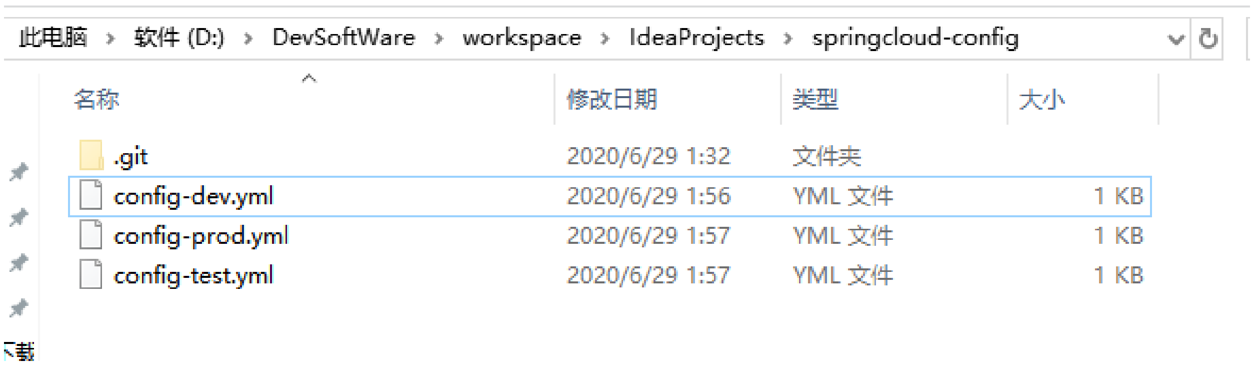


此电脑 › 软件 (D:) › DevSoftWare › workspace › IdeaProjects

名称	修改日期	类型
cloud2020	2020/6/29 1:31	文件夹
concurrency_in_practice	2020/1/22 20:57	文件夹
concurrency-tools-in-practice	2020/4/6 17:03	文件夹
concurrent	2020/3/24 0:01	文件夹
itcast-activemq	2020/5/4 2:57	文件夹
itcast-algorithms-and-data-structures	2020/4/2 14:42	文件夹
itcast-jvm	2020/3/24 1:25	文件夹
java_architect_lesson_2019	2020/2/27 21:17	文件夹
java_architect_lesson_2020	2020/6/25 23:55	文件夹
springcloud-config	2020/6/29 1:32	文件夹
spring-framework-5.0.2.RELEASE-CN	2020/4/27 22:35	文件夹
cloud2020.zip	2020/6/29 1:24	WinRAR ZIP 压
java_architect_lesson_2019.rar	2020/2/18 14:02	WinRAR 压缩文
java_architect_lesson_2019-202002181313.rar	2020/2/18 13:13	WinRAR 压缩文

2.1.4. 提交配置文件

在路径D:\DevSoftWare\workspace\IdeaProjects\springcloud-config下分别创建三个配置文件



此电脑 › 软件 (D:) › DevSoftWare › workspace › IdeaProjects › springcloud-config

名称	修改日期	类型	大小
.git	2020/6/29 1:32	文件夹	
config-dev.yml	2020/6/29 1:56	YML 文件	1 KB
config-prod.yml	2020/6/29 1:57	YML 文件	1 KB
config-test.yml	2020/6/29 1:57	YML 文件	1 KB

config-dev.yml

```yaml
config:
  info: "master branch,springcloud-config/config-dev.yml version=1"
```

config-test.yml

```yaml
config:
  info: "master branch,springcloud-config/config-test.yml version=1"
```

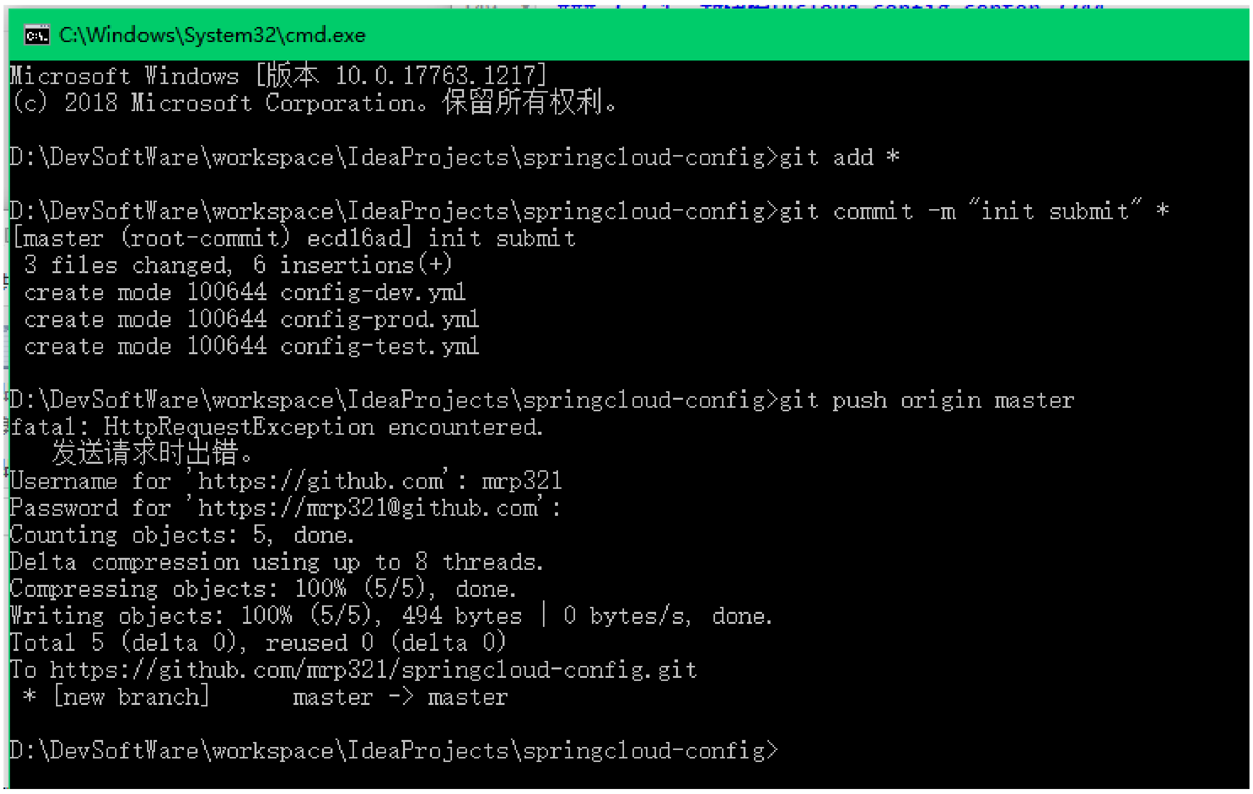
config-prod.yml

```yaml
config:
```

```
info: "master branch,springcloud-config/config-prod.yml version=1"
```

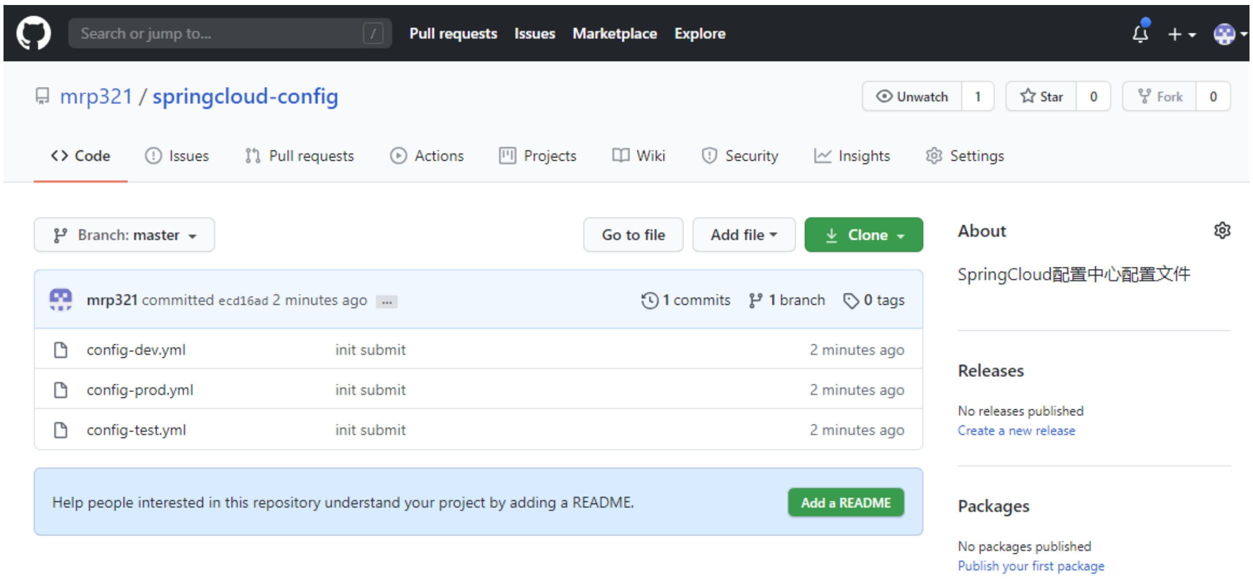
提交这三个配置文件

```
D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>git add *

D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>git commit -m "init submit'

D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>git push origin master
```



```
Microsoft Windows [版本 10.0.17763.1217]
(c) 2018 Microsoft Corporation。保留所有权利。

D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>git add *

D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>git commit -m "init submit" *
[master (root-commit) ecd16ad] init submit
 3 files changed, 6 insertions(+)
 create mode 100644 config-dev.yml
 create mode 100644 config-prod.yml
 create mode 100644 config-test.yml

D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>git push origin master
fatal: HttpRequestException encountered.
   发送请求时出错。
Username for 'https://github.com': mrp321
Password for 'https://mrp321@github.com':
Counting objects: 5, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 494 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/mrp321/springcloud-config.git
 * [new branch]      master -> master

D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>
```

访问远程仓库地址 https://github.com/mrp321/springcloud-config



mrp321 / springcloud-config

mrp321 committed ecd16ad 2 minutes ago — 1 commits · 1 branch · 0 tags

config-dev.yml	init submit	2 minutes ago
config-prod.yml	init submit	2 minutes ago
config-test.yml	init submit	2 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

About

SpringCloud配置中心配置文件

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

可以看到配置文件已经提交成功了

2.1.5. 注意事项

该仓库下主要包含的是多个环境的配置文件

其中文件的保存格式必须是UTF-8

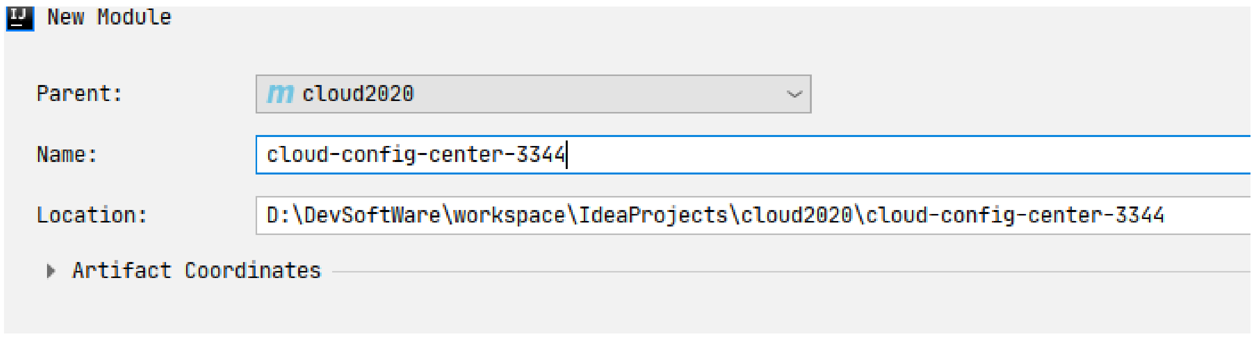
如果需要修改, 此处需要模拟运维人员来操作git和github

```
git add

git commit -m "init yml"

git push origin master
```

2.2. 新建配置中心模块

2.2.1. 新建模块cloud-config-center-3344

新建Module模块cloud-config-center-3344,它既为Cloud的配置中心模块cloudConfig Center



Parent: cloud2020

Name: cloud-config-center-3344

Location: D:\DevSoftWare\workspace\IdeaProjects\cloud2020\cloud-config-center-3344

2.2.2. POM

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloud-config-center-3344</artifactId>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-config-server</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>
        <dependency>
            <groupId>cn.sitedev.springcloud</groupId>
            <artifactId>cloud-api-commons</artifactId>
            <version>${project.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

```xml

</project>
```

2.2.3. YML

```yaml
server:
  port: 3344
spring:
  application:
    name: cloud-config-center
  cloud:
    config:
      server:
        git:
          uri: https://github.com/mrp321/springcloud-config.git # 填写你自己的githul
          search-paths:
            - springcloud-config
      label: master
eureka:
  client:
    service-url:
      defaultZone:  http://localhost:7001/eureka
```

2.2.4. 主启动类

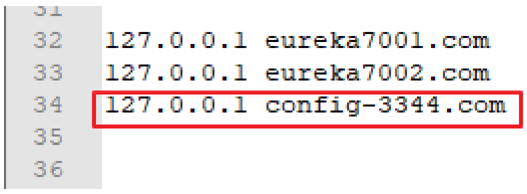
主启动类需要添加@EnableConfigServer注解

```java
package cn.sitedev.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.config.server.EnableConfigServer;

@SpringBootApplication
@EnableConfigServer
public class ConfigCenterMain3344 {
    public static void main(String[] args) {
        SpringApplication.run(ConfigCenterMain3344.class, args);
    }
}
```

2.2.5. windows下修改hosts文件,添加映射

hosts文件路径: C:\Windows\System32\drivers\etc\hosts

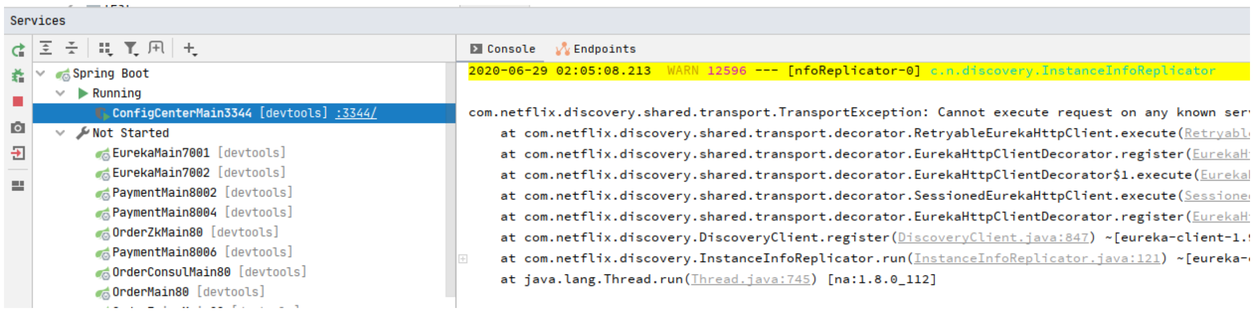
```
127.0.0.1 config-3344.com
```



```
127.0.0.1 eureka7001.com
127.0.0.1 eureka7002.com
127.0.0.1 config-3344.com
```

2.2.6. 测试

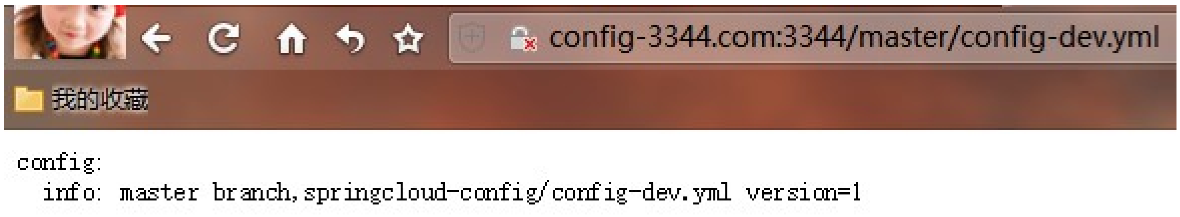
启动微服务3344(因为我们没有启动eureka7001, 所以会看到控制台报错)



```
Services
Spring Boot
  Running
    ConfigCenterMain3344 [devtools] :3344/
  Not Started
    EurekaMain7001 [devtools]
    EurekaMain7002 [devtools]
    PaymentMain8002 [devtools]
    PaymentMain8004 [devtools]
    OrderZkMain80 [devtools]
    PaymentMain8006 [devtools]
    OrderConsulMain80 [devtools]
    OrderMain80 [devtools]
```

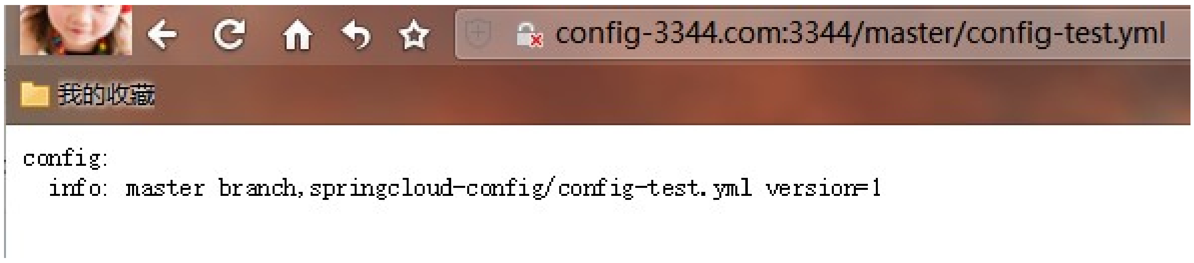
```
2020-06-29 02:05:08.213  WARN 12596 --- [nfoReplicator-0] c.n.discovery.InstanceInfoReplicator

com.netflix.discovery.shared.transport.TransportException: Cannot execute request on any known ser
    at com.netflix.discovery.shared.transport.decorator.RetryableEurekaHttpClient.execute(Retryabl
    at com.netflix.discovery.shared.transport.decorator.EurekaHttpClientDecorator.register(EurekaH
    at com.netflix.discovery.shared.transport.decorator.EurekaHttpClientDecorator$1.execute(Eureka
    at com.netflix.discovery.shared.transport.decorator.SessionedEurekaHttpClient.execute(Sessione
    at com.netflix.discovery.shared.transport.decorator.EurekaHttpClientDecorator.register(EurekaH
    at com.netflix.discovery.DiscoveryClient.register(DiscoveryClient.java:847) ~[eureka-client-1.
    at com.netflix.discovery.InstanceInfoReplicator.run(InstanceInfoReplicator.java:121) ~[eureka-
    at java.lang.Thread.run(Thread.java:745) [na:1.8.0_112]
```

浏览器访问 http://config-3344.com:3344/master/config-dev.yml



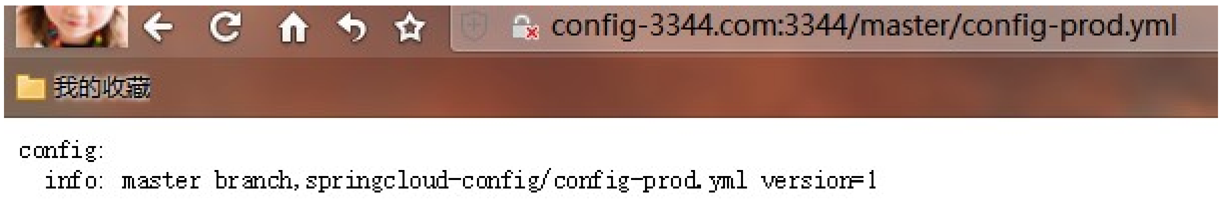
```
config:
  info: master branch,springcloud-config/config-dev.yml version=1
```

浏览器访问 http://config-3344.com:3344/master/config-test.yml



```
config:
  info: master branch,springcloud-config/config-test.yml version=1
```

浏览器访问 http://config-3344.com:3344/master/config-prod.yml



```
config:
  info: master branch,springcloud-config/config-prod.yml version=1
```

2.2.7. 结论

成功实现了用SpringCloud Config 通过GitHub获取配置信息

2.3. 配置读取规则

2.3.1. 官网

> https://cloud.spring.io/spring-cloud-static/spring-cloud-config/2.2.1.RELEASE/reference/html/#_quick_start

The default strategy for locating property sources is to clone a git repository (at `spring.cloud.config.server.git.uri`) and use it to initialize a mini `SpringApplication` . The mini-application’ s `Environment` is used to enumerate property sources and publish them at a JSON endpoint.

The HTTP service has resources in the following form:

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

where `application` is injected as the `spring.config.name` in the `SpringApplication` (what is normally `application` in a regular Spring Boot app), `profile` is an active profile (or comma-separated list of properties), and `label` is an optional git label (defaults to `master` .)

Spring Cloud Config Server pulls configuration for remote clients from various sources. The following example gets configuration from a git repository (which must be provided), as shown in the following example:

```
spring:
  cloud:
    config:
      server:
        git:
          uri: https://github.com/spring-cloud-samples/config-repo
```

Other sources are any JDBC compatible database, Subversion, Hashicorp Vault, Credhub and local filesystems.

2.3.2. /{label}/{application}-{profile}.yml（最推荐使用这种方式）

2.3.2.1. master分支

http://config-3344.com:3344/master/config-dev.yml

http://config-3344.com:3344/master/config-test.yml

http://config-3344.com:3344/master/config-prod.yml

2.3.2.2. dev分支

http://config-3344.com:3344/dev/config-dev.yml

http://config-3344.com:3344/dev/config-test.yml

http://config-3344.com:3344/dev/config-prod.yml

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

http://config-3344.com:3344/config-dev.yml

http://config-3344.com:3344/config-test.yml

http://config-3344.com:3344/config-prod.yml

http://config-3344.com:3344/config-xxxx.yml(不存在的配置)

2.3.4. /{application}-{profile}[/{label}]

http://config-3344.com:3344/config/dev/master

http://config-3344.com:3344/config/test/master

http://config-3344.com:3344/config/prod/master

2.3.5. 重要配置细节总结

针对形如如下名称的配置文件:

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

/{label}/{name}-{profile}.yml
```

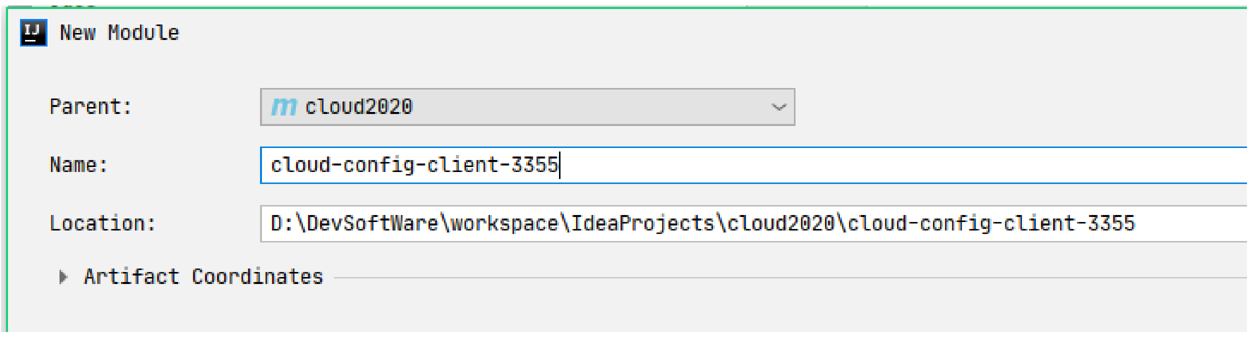
有:

```
label: 分支(branch)

name: 服务名

profiles: 环境(dev/test/prod)
```

3. Config客户端配置与测试

3.1. 新建配置客户端模块

3.1.1. 新建模块cloud-config-client-3355


New Module

Parent: cloud2020

Name: cloud-config-client-3355

Location: D:\DevSoftWare\workspace\IdeaProjects\cloud2020\cloud-config-client-3355

Artifact Coordinates

3.1.2. POM

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloud-config-client-3355</artifactId>
    <dependencies>

        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-config</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>
        <dependency>
            <groupId>cn.sitedev.springcloud</groupId>
            <artifactId>cloud-api-commons</artifactId>
            <version>${project.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

3.1.3. bootstrap.xml

3.1.3.1. 是什么

applicaiton.yml是用户级的资源配置项

bootstrap.yml是系统级的，优先级更加高

Spring Cloud会创建一个"Bootstrap Context"，作为 Spring应用的 Application Context的父上下文。初始化的时候，Bootstrap Context负责从外部源加载配置属性并解析配置。这两个上下文共享一个从外部获取的 Environment

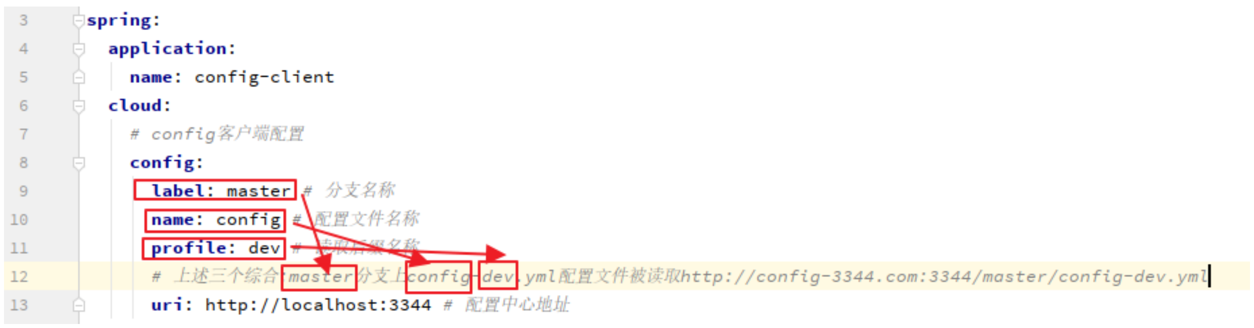
Bootstrap属性有高优先级，默认情况下，它们不会被本地配置覆盖。Bootstrap Cntext和 Application Context有着不同的约定，所以新增了一个 bootstrap.yml文件，保证 Bootstrap Context和 Application Context配置的分离。

要将 Client 模块下的 application.yml文件改为 bootstrap.yml，这是很关键的，因为 bootstrap.yml是比 application.yml先加载的。bootstrap.yml优先级高于 application.yml

3.1.3.2. 内容

```yaml
server:
  port: 3355
spring:
  application:
    name: config-client
  cloud:
    # config客户端配置
    config:
      label: master # 分支名称
      name: config # 配置文件名称
      profile: dev # 读取后缀名称
      # 上述三个综合:master分支上config-dev.yml配置文件被读取http://config-3344.com:
      uri: http://localhost:3344 # 配置中心地址
eureka:
  client:
    service-url:
      defaultZone: http://eureka7001.com:7001/eureka
```

3.1.3.3. 说明



```yaml
spring:
  application:
    name: config-client
  cloud:
    # config客户端配置
    config:
      label: master # 分支名称
      name: config # 配置文件名称
      profile: dev # 读取后缀名称
      # 上述三个综合：master分支上config-dev.yml配置文件被读取http://config-3344.com:3344/master/config-dev.yml
      uri: http://localhost:3344 # 配置中心地址
```

3.1.4. 修改config-dev.yml配置并提交至Github(该步可忽略不做)

修改config-dev.yml配置并提交到GitHub中，比如加个变量age或者版本号version

3.1.5. 主启动

```java
package cn.sitedev.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

@SpringBootApplication
@EnableEurekaClient
public class ConfigClientMain3355 {
    public static void main(String[] args) {
        SpringApplication.run(ConfigClientMain3355.class, args);
    }
}
```

3.1.6. 业务类

controller:

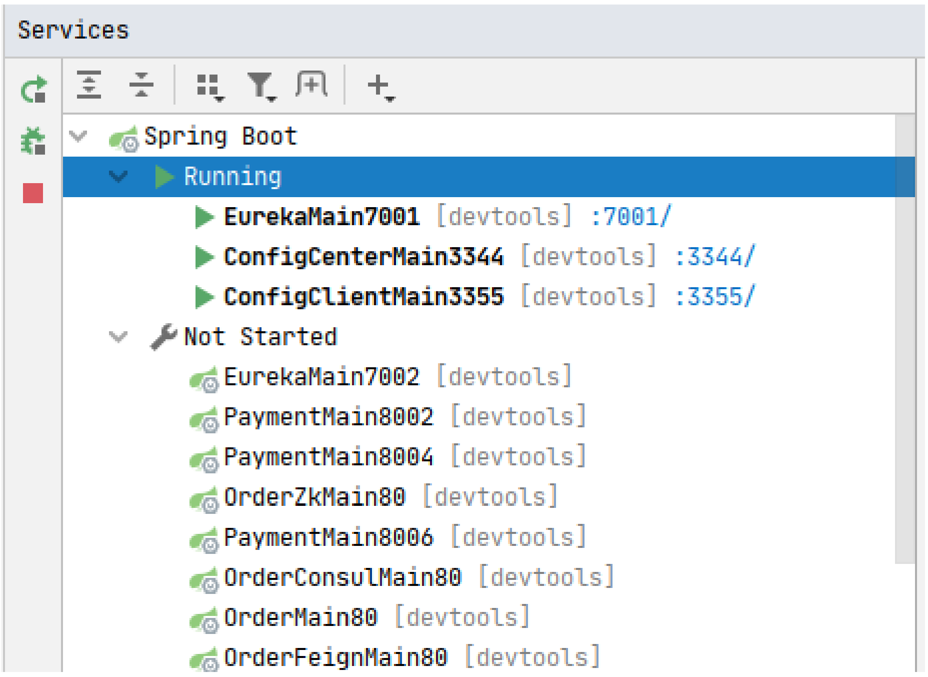
```java
package cn.sitedev.springcloud.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ConfigClientController {
    @Value("${config.info}")
    private String configInfo;

    @GetMapping("/configInfo")
    public String getConfigInfo() {
        return configInfo;
    }
}
```

3.1.7. 测试

启动EurekaMain7001

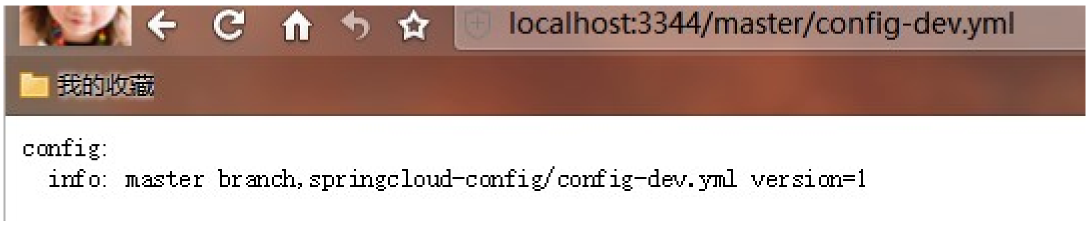
启动ConfigCenterMain3344

启动ConfigClientMain3355



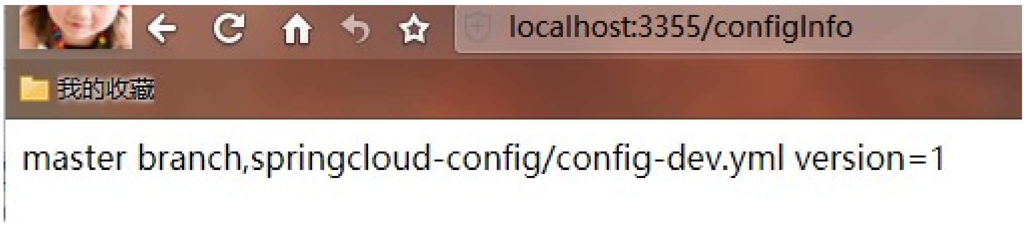
```
Services
Spring Boot
  Running
    EurekaMain7001 [devtools] :7001/
    ConfigCenterMain3344 [devtools] :3344/
    ConfigClientMain3355 [devtools] :3355/
  Not Started
    EurekaMain7002 [devtools]
    PaymentMain8002 [devtools]
    PaymentMain8004 [devtools]
    OrderZkMain80 [devtools]
    PaymentMain8006 [devtools]
    OrderConsulMain80 [devtools]
    OrderMain80 [devtools]
    OrderFeignMain80 [devtools]
```

浏览器访问 http://localhost:3344/master/config-dev.yml



```
config:
  info: master branch,springcloud-config/config-dev.yml version=1
```

浏览器访问 http://localhost:3355/configInfo



```
master branch,springcloud-config/config-dev.yml version=1
```

3.1.8. 结论

成功实现了客户端3355访问SpringCloud Config3344通过GitHub获取配置信息

3.2. 问题随之而言，分布式配置的动态刷新

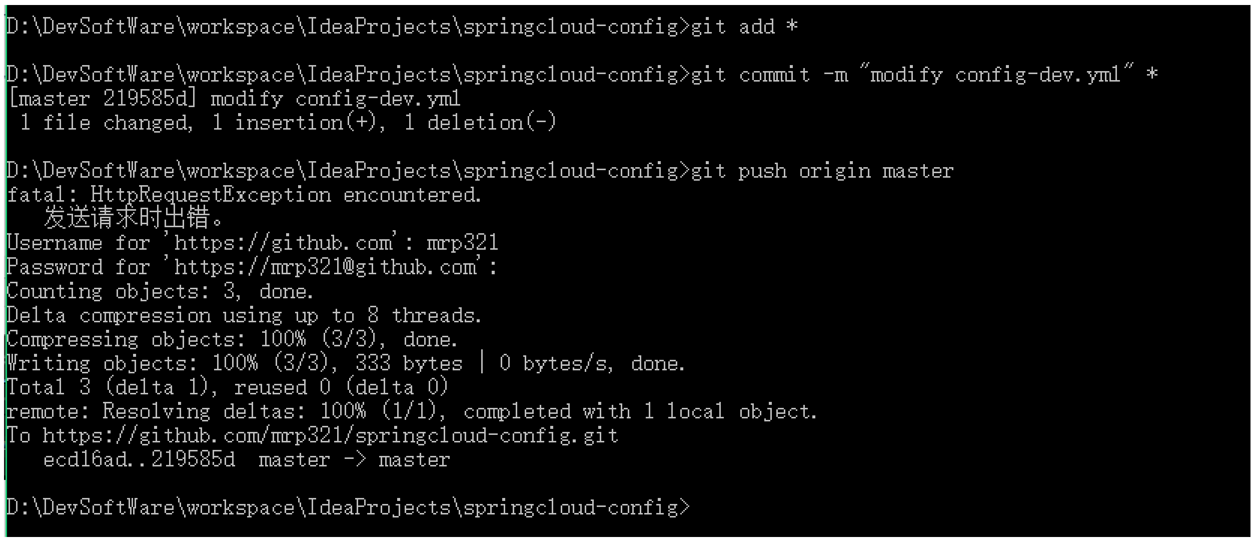
- Linux运维修改GitHub上的配置文件内容做调整

> 修改 config-dev.yml

```yaml
config:
  info: "master branch,springcloud-config/config-dev.yml version=2"
```

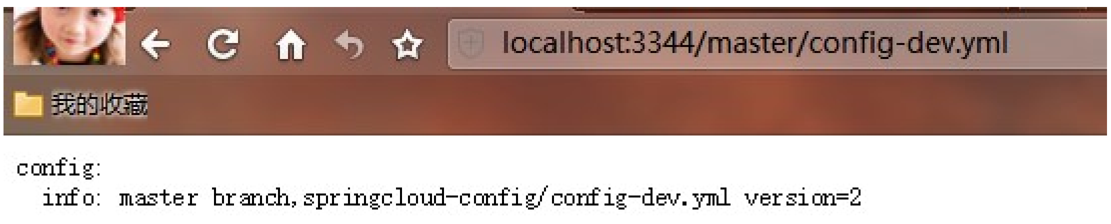
> 提交修改后的配置文件至Github

```
D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>git add *
D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>git commit -m "modify conf
D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>git push origin master
```



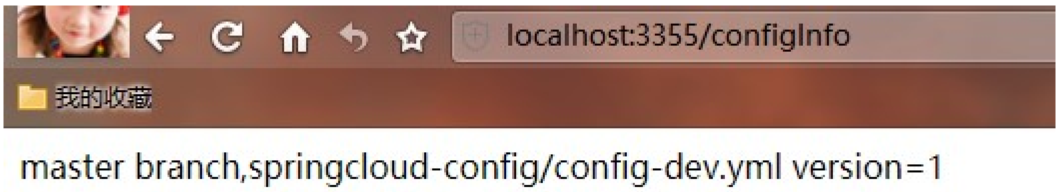
```
D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>git add *

D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>git commit -m "modify config-dev.yml" *
[master 219585d] modify config-dev.yml
 1 file changed, 1 insertion(+), 1 deletion(-)

D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>git push origin master
fatal: HttpRequestException encountered.
   发送请求时出错。
Username for 'https://github.com': mrp321
Password for 'https://mrp321@github.com':
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 333 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/mrp321/springcloud-config.git
   ecd16ad..219585d  master -> master

D:\DevSoftWare\workspace\IdeaProjects\springcloud-config>
```

- 刷新3344，发现ConfigServer配置中心立刻响应

> 浏览器访问配置中心3344地址: http://localhost:3344/master/config-dev.yml
>
> 
>
>
> ```
> config:
>   info: master branch,springcloud-config/config-dev.yml version=2
> ```

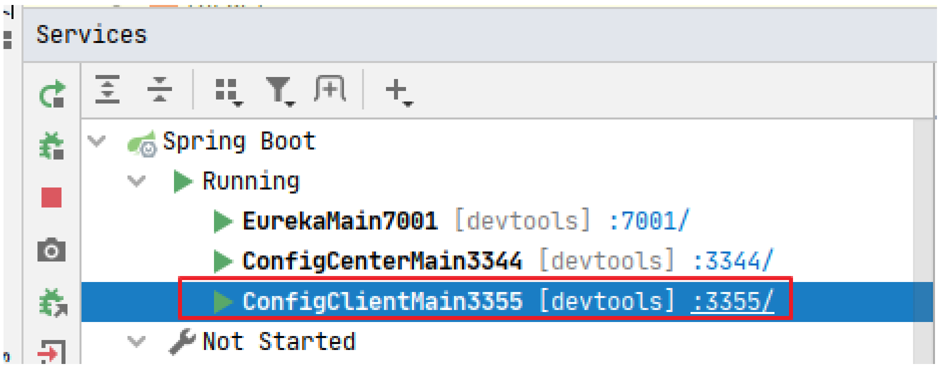
- 刷新3355，发现ConfigServer客户端没有任何响应

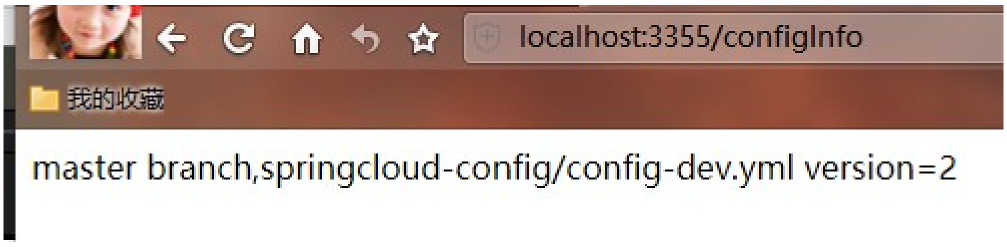
> 浏览器访问配置客户端3355地址: http://localhost:3355/configInfo



master branch,springcloud-config/config-dev.yml version=1

- 3355没有变化除非自己重启或者重新加载

> 重启配置客户端3355服务
>
> 
>

Services
> Spring Boot
> Running
> EurekaMain7001 [devtools] :7001/
> ConfigCenterMain3344 [devtools] :3344/
> ConfigClientMain3355 [devtools] :3355/
> Not Started
>
> 浏览器再次访问配置客户端3355地址: http://localhost:3355/configInfo
>
> 
>
> master branch,springcloud-config/config-dev.yml version=2

- 难道每次运维修改配置文件，客户端都需要重启？？ 噩梦

4. Config客户端之动态刷新

4.1. 目标

避免每次更新配置都要重启客户端微服务3355

4.2. 动态刷新

4.2.1. 修改3355模块

4.2.1.1. POM

修改pom.xml, 引入actuator监控

修改内容:

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

```xml
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
```

完整内容:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.o
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloud-config-client-3355</artifactId>
    <dependencies>

        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-config</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>
        <dependency>
            <groupId>cn.sitedev.springcloud</groupId>
            <artifactId>cloud-api-commons</artifactId>
            <version>${project.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

4.2.1.2. YML

修改bootstrap.xml, 暴露监控端口

修改内容:

```yaml
# 暴露监控端口
management:
  endpoints:
    web:
      exposure:
        include: "*"
```

完整内容:

```yaml
server:
  port: 3355
spring:
  application:
    name: config-client
  cloud:
    # config客户端配置
    config:
      label: master # 分支名称
      name: config # 配置文件名称
      profile: dev # 读取后缀名称
      # 上述三个综合:master分支上config-dev.yml配置文件被读取http://config-3344.com:
      uri: http://localhost:3344 # 配置中心地址
eureka:
  client:
    service-url:
      defaultZone: http://eureka7001.com:7001/eureka

# 暴露监控端口
management:
  endpoints:
    web:
      exposure:
        include: "*"
```

4.2.1.3. 业务类

controller添加@RefreshScope注解

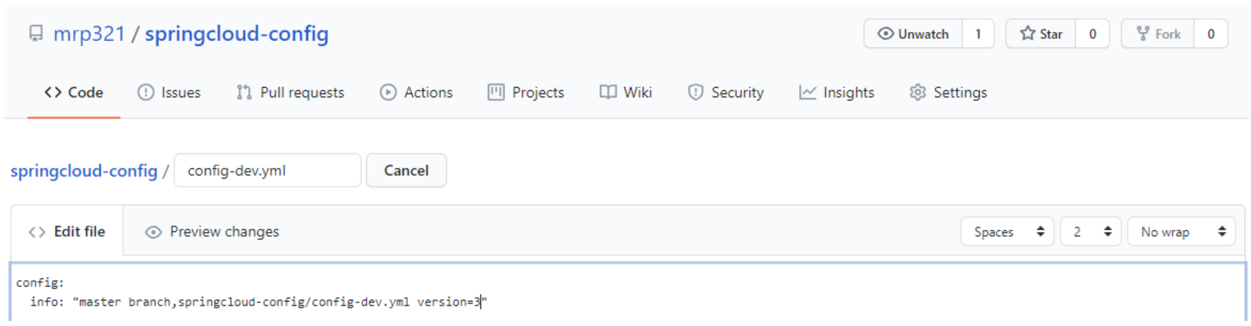
```java
package cn.sitedev.springcloud.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.cloud.context.config.annotation.RefreshScope;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RefreshScope
public class ConfigClientController {
    @Value("${config.info}")
    private String configInfo;

    @GetMapping("/configInfo")
    public String getConfigInfo() {
        return configInfo;
    }
}
```

4.2.2. 测试

修改配置文件config-dev.yml, 提交至Github

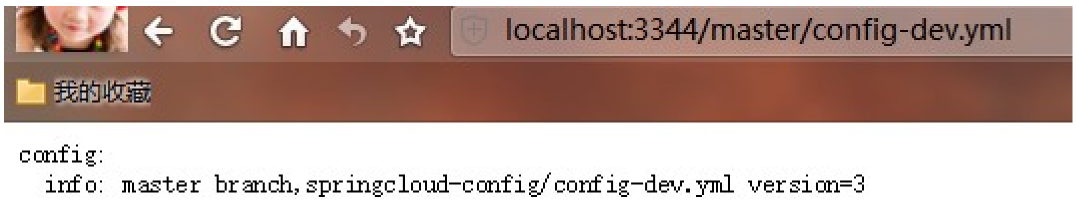
> 为了方便起见, 我们直接在Github上操作该代码

```
config:
  info: "master branch,springcloud-config/config-dev.yml version=3"
```



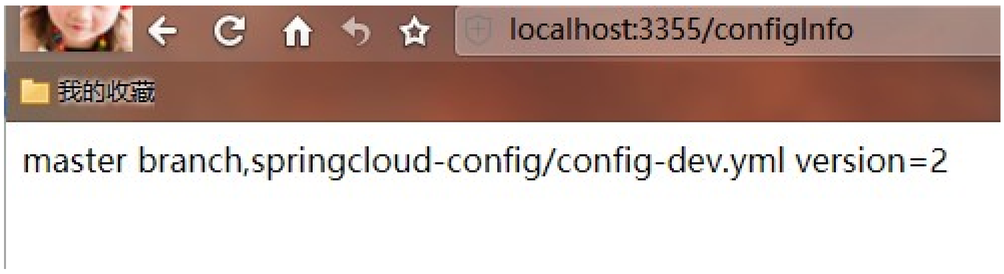
```
config:
  info: "master branch,springcloud-config/config-dev.yml version=3"
```

浏览器访问配置中心3344地址: http://localhost:3344/master/config-dev.yml



```
config:
  info: master branch,springcloud-config/config-dev.yml version=3
```

浏览器访问配置客户端3355地址: http://localhost:3355/configInfo



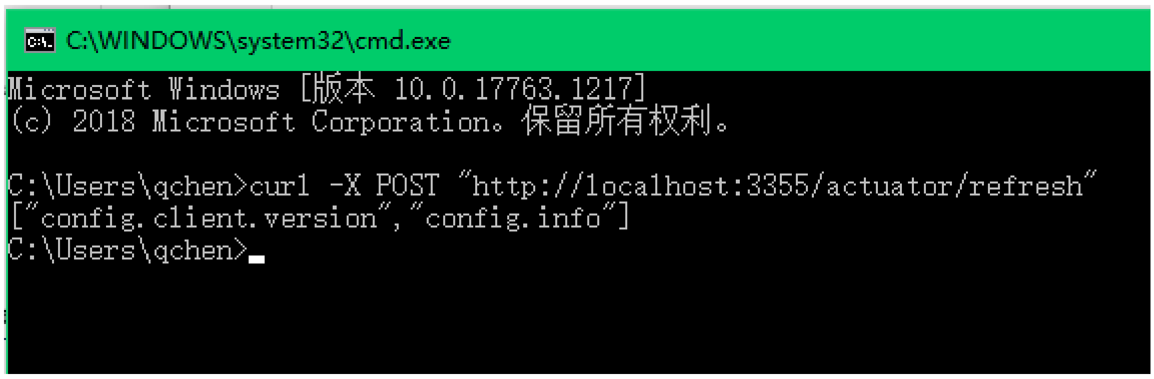
master branch,springcloud-config/config-dev.yml version=2

可以看到配置客户端3355获取到的配置并没有发生改变, 那应该怎样做呢

4.2.3. 怎样做

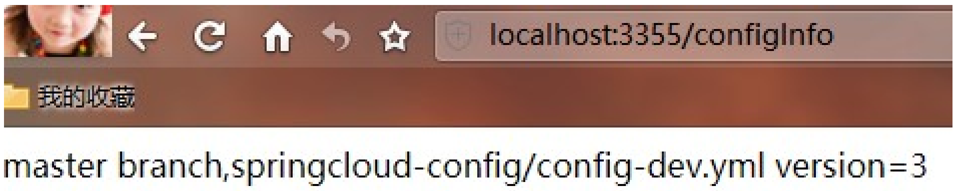
需要运维人员发送Post请求刷新3355(必须是POST请求)

```
curl -X POST "http://localhost:3355/actuator/refresh"
```



```
Microsoft Windows [版本 10.0.17763.1217]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\qchen>curl -X POST "http://localhost:3355/actuator/refresh"
["config.client.version","config.info"]
C:\Users\qchen>
```

浏览器访问配置客户端3355地址: http://localhost:3355/configInfo



```
master branch,springcloud-config/config-dev.yml version=3
```

4.2.4. 结论

成功实现了客户端3355刷新到最新配置内容, 避免了服务的重启

4.3. 想想还有什么问题

假如有多个微服务客户端3355/3366/3377。。。。

每个微服务都要执行一次post请求，手动刷新?

可否广播，一次通知，处处生效?

我们想大范围的自动刷新，求方法