简介

在分布式系统中，由于服务数量巨多，为了方便服务配置文件统一管理，实时更新，所以需要分布式配置中心组件。在Spring Cloud中，有分布式配置中心组件spring cloud config ，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仓库中。在spring cloud config 组件中，分两个角色，一是config server，二是config client。

构建Config Server

1.创建一个父maven工程，pom文件如下：

```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.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.gewdata</groupId>
 <artifactId>springCloudConfigDemo</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 <packaging>pom</packaging>

 <name>springCloudConfigDemo</name>
 <description>Demo project for Spring Boot</description>

 <parent>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-parent</artifactId>
 <version>2.0.3.RELEASE</version>
 <relativePath/>
 </parent>

 <properties>
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
 <java.version>1.8</java.version>
 <spring-cloud.version>Finchley.RELEASE</spring-cloud.version>
 </properties>
```

```

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

<dependencyManagement>
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-dependencies</artifactId>
<version>${spring-cloud.version}</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>

<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>

</project>
```

2.创建一个spring boot子项目，取名为config-server，pom文件如下：

```
<?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.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.gewdata</groupId>
 <artifactId>config-server</artifactId>
```

```xml
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

 <name>config-server</name>
 <description>Demo project for Spring Boot</description>

 <parent>
 <groupId>com.gewdata</groupId>
 <artifactId>springCloudConfigDemo</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 </parent>

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

 <build>
 <plugins>
 <plugin>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-maven-plugin</artifactId>
 </plugin>
 </plugins>
 </build>


</project>
```

3.在程序入口处加上@EnableConfigServer注解开启配置服务器的功能，代码如下：

```java
package com.gewdata;

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

```

```java
@SpringBootApplication
@EnableConfigServer // 开启配置服务器功能
public class ConfigServerApplication {

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

4.需要在程序的配置文件application.properties文件配置以下：

```properties
spring.application.name=config-server
server.port=8888

# 配置git仓库地址
spring.cloud.config.server.git.uri=http://139.159.143.78:10086/wangjunyao
pringCloudDemo.git
# 配置仓库路径
spring.cloud.config.server.git.searchPaths=config
# 配置仓库分支
spring.cloud.config.label=master
# 访问git仓库用户名
spring.cloud.config.server.git.username=wangjunyao
# 访问git仓库用户密码
spring.cloud.config.server.git.password=12345678
```

5.git中文件config下有个文件wjy-client.properties，其中有个属性为

```properties
msg=hahahahahahahaha
```

6.启动程序，访问：访问http://localhost:8888/ wjy-client/dev：

```
{"name":"wjy-client","profiles":["dev"],"label":null,"version":"fafa6a08d
52653f51c6259162f7526b88b552dee","state":null,"propertySources":[{"name":"h
ttp://139.159.143.78:10086/wangjunyao/SpringCloudDemo.git/config/wjy-clien
t.properties","source":{"msg":"hahahahahahahaha"}}]}
```

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

构建一个Config Client

1.重新创建一个spring boot项目，取名为config-client，pom文件如下：

```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.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.gewdata</groupId>
 <artifactId>config-client</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 <packaging>jar</packaging>

 <name>config-client</name>
 <description>Demo project for Spring Boot</description>

 <parent>
 <groupId>com.gewdata</groupId>
 <artifactId>springCloudConfigDemo</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 </parent>

 <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>

 <build>
 <plugins>
 <plugin>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-maven-plugin</artifactId>
 </plugin>
```

```
  </plugins>
  </build>
</project>
```

2.配置文件bootstrap.properties：

```
# 这里的配置是和git上文件名相对应的
spring.application.name=wjy-client
# 指明分支
spring.cloud.config.label=master
spring.cloud.config.profile=dev
# 指明配置服务中心网址
spring.cloud.config.uri= http://localhost:8888/
server.port=8881
```

这里要注意的是：spring.application.name这个属性值一定要和git上配置文件名相对应，否则运行客户端时会报找不到值的错误！！！

3.程序入口，写一个接口，返回从配置中心读取的变量的值：

```
package com.gewdata;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
 @RestController
 public class ConfigClientApplication {

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

  @Value("${msg}")
  String msg;

  @RequestMapping(value = "/hi")
  public String hi(){
  return msg;
  }
 }
```

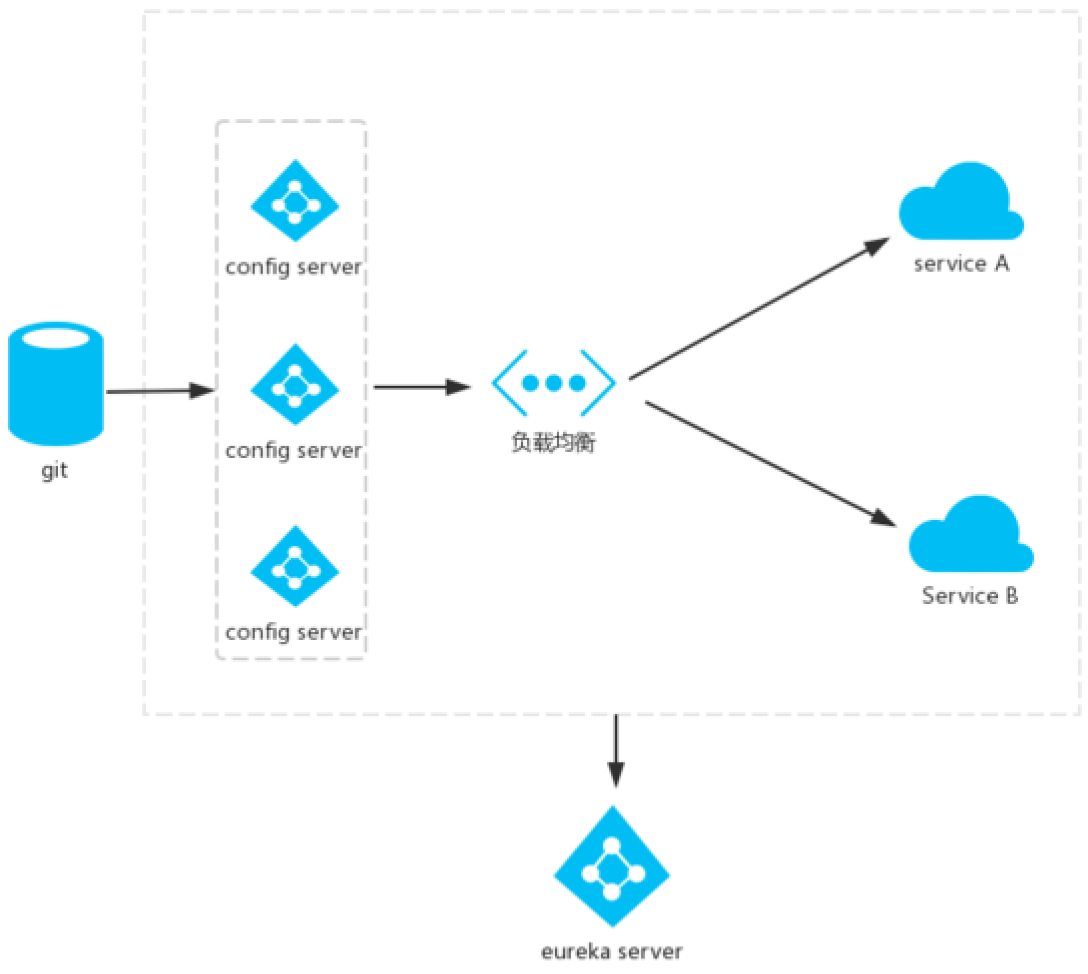
3.访问网址http://localhost:8881/hi，网页显示：

```
hahahahahahahaha
```

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

高可用的分布式配置中心

当服务实例很多时，都从配置中心读取文件，这时可以考虑将配置中心做成一个微服务，将其集群化，从而达到高可用：



准备工作

1.创建一个eureka-server工程，用作服务注册中心，pom文件如下：

```
<?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.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
```

```xml

<groupId>com.gewdata</groupId>
<artifactId>eureka-server</artifactId>
<version>0.0.1-SNAPSHOT</version>

 <parent>
 <groupId>com.gewdata</groupId>
 <artifactId>springCloudConfigDemo</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 </parent>

 <name>eureka-server</name>
 <description>Demo project for Spring Boot</description>

 <dependencies>
 <dependency>
 <groupId>org.springframework.cloud</groupId>
 <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
 </dependency>

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

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

 <build>
 <plugins>
 <plugin>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-maven-plugin</artifactId>
 </plugin>
 </plugins>
 </build>

</project>
```

2.在配置文件application.yml上，指定服务端口8889，加上作为服务注册中心的基本配置：

```
server:
 port: 8889

eureka:
 instance:
 hostname: localhost
 client:
 registerWithEureka: false
 fetchRegistry: false
  serviceUrl:
  defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
```

3.入口类：

```java
package com.gewdata;

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

@EnableEurekaServer
@SpringBootApplication
public class EurekaServerApplication {

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

 }

```

改造config-server

1.在pom文件加上EurekaClient的依赖：

```xml
<dependency>
 <groupId>org.springframework.cloud</groupId>
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

2.配置文件application.yml，指定服务注册地址为http://localhost:8889/eureka/，其他配置同上一篇文章，完整的配置如下：

```
spring.application.name=config-server
server.port=8888

# 配置git仓库地址
spring.cloud.config.server.git.uri=http://139.159.143.78:10086/wangjunyao/SpringCloudDemo.git
# 配置仓库路径
spring.cloud.config.server.git.searchPaths=config
# 配置仓库分支
spring.cloud.config.label=master
 # 访问git仓库用户名
 spring.cloud.config.server.git.username=wangjunyao
 # 访问git仓库用户密码
 spring.cloud.config.server.git.password=12345678
 # 服务注册地址
 eureka.client.serviceUrl.defaultZone=http://localhost:8889/eureka/
```

3.在程序启动类上加上@EnableEurekaClient注解

改造config-client

1.将其注微到服务注册中心，作为Eureka客户端，需要pom文件加上起步依赖spring-cloud-starter-netflix-eureka-client，代码如下：

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

2.配置文件bootstrap.properties，注意是bootstrap。加上服务注册地址http://localhost:8889/eureka/：

```
# 这里的配置是和git上文件名相对应的
spring.application.name=wjy-client
# 指明分支
spring.cloud.config.label=master
spring.cloud.config.profile=dev
# 指明配置服务中心网址
#spring.cloud.config.uri= http://localhost:8888/
server.port=8881

```

```
# 指定服务注册地址
eureka.client.serviceUrl.defaultZone=http://localhost:8889/eureka/
# 是从配置中心读取文件
spring.cloud.config.discovery.enabled=true
# 配置中心的servield，即是服务名
spring.cloud.config.discovery.serviceId=config-server
```

这时发现，在读取配置文件不再写ip地址，而是服务名，这时如果配置服务部署多份，通过负载均衡，从而高可用。

3.依次启动eureka-servr,config-server,config-client，访问http://localhost:8881/hi，浏览器显示：

```
hahahahahahahaha
```