1. 概述

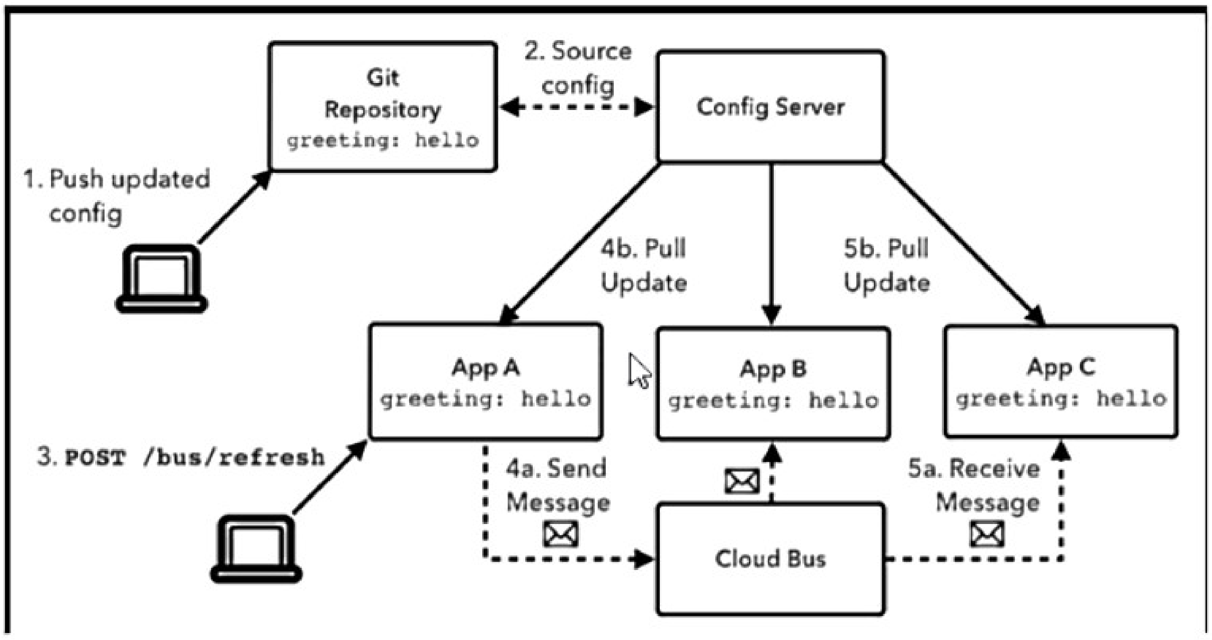
1.1. 上一讲的加深和扩展, 一言以蔽之

分布式自动刷新配置功能

Spring Cloud Bus配合Spring Cloud Config使用可以实现配置的动态刷新

1.2. 是什么

Spring Cloud Bus配合 Spring Cloud Config使用可以实现配置的动态刷新



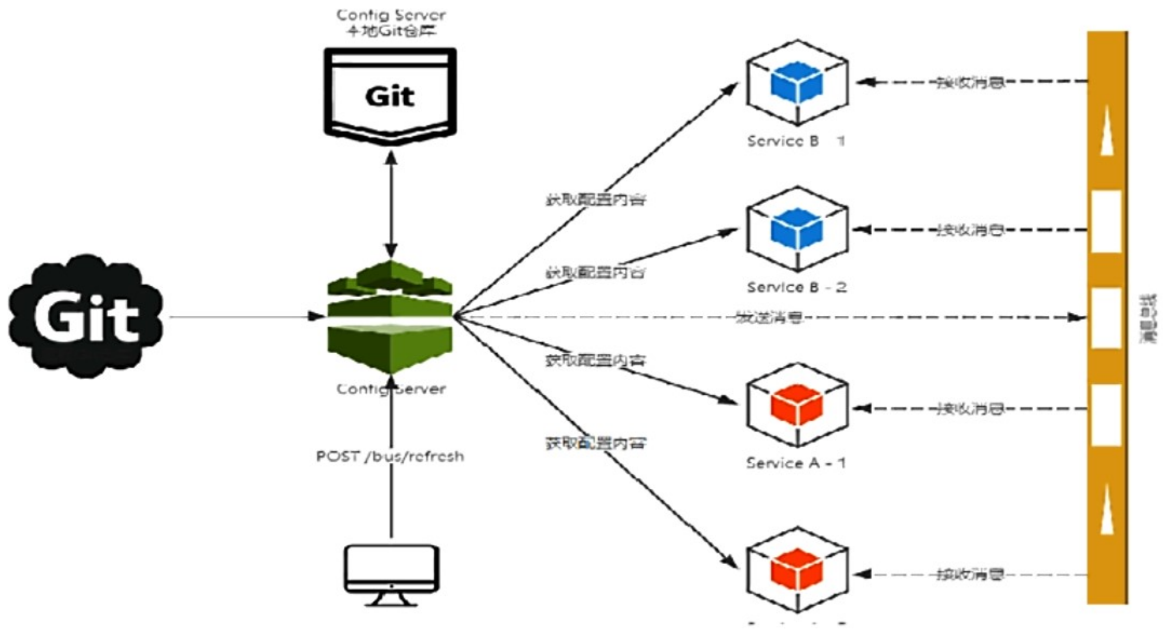
Spring Cloud Bus是用来将分布式系统的节点与轻量级消息系统链接起来的框架

它整合了Java的事件处理机制和消息中间件的功能

Spring Cloud Bus目前支持 RabbitMQ和 Kafka两种消息代理

1.3. 能干嘛

Spring Cloud Bus能管理和传播分布式系统间的消息，就像一个分布式执行器，可用于广播状态更改、事件推送等，也可以当作微服务间的通信通道



1.4. 为何被称为总线

什么是总线

- 在微服务架构的系统中，通常会使用轻量级的消息代理来构建一个共用的消息主题，并让系统中所有微服务实例都连接上来。由于该主题中产生的消息会被所有实例监听和消费，所以

称它为消息总线。在总线上的各个实例，都可以方便地广播一些需要让其他连接在该主题上的实例都知道的消息。

基本原理

- ConfigClient实例都监听MQ中同个 topic（默认是 SpringCloudBus）。当个服务刷新数据的时候，它会把这个信息放入到 Topic中，这样其它监听同一 Topic的服务就能得到通知，然后去更新自身的配置。

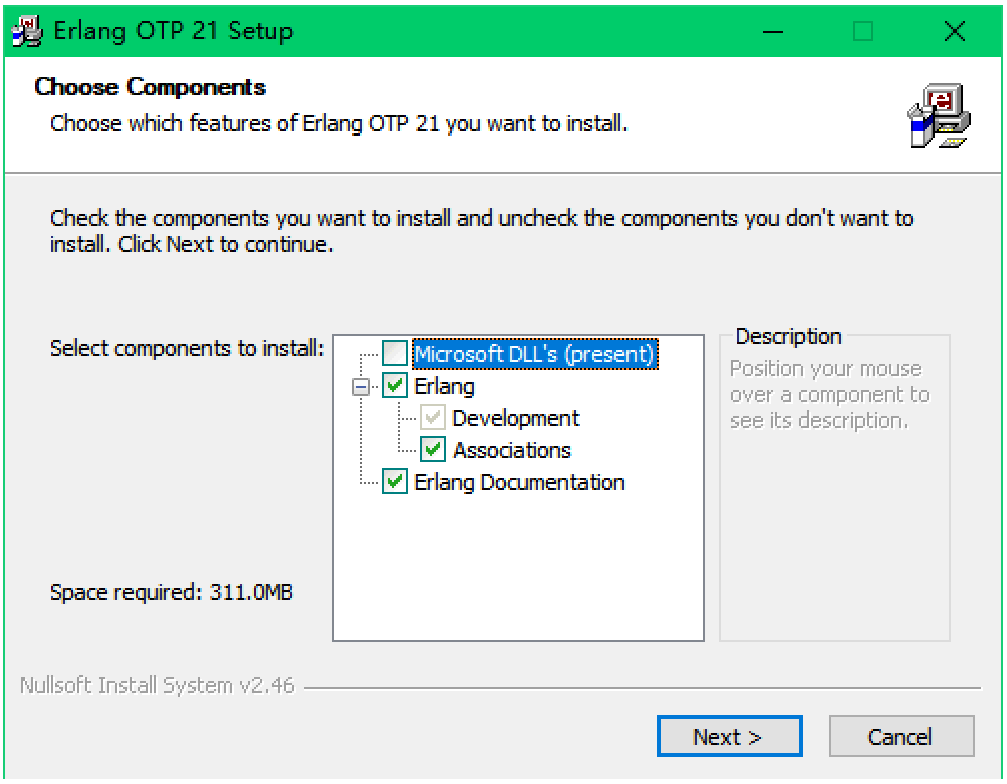
2. RabbitMQ环境配置

安装时, 需要注意Erlang与RabbitMQ的版本兼容性

> https://www.rabbitmq.com/which-erlang.html#compatibility-matrix

2.1. 安装Erlang

> 里使用的版本和课程中演示的版本一致, 均为 otp_win64_21.3.exe
>
> http://erlang.org/download/otp_win64_21.3.exe


Erlang OTP 21 Setup

Choose Components

Choose which features of Erlang OTP 21 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [ ] Microsoft DLL's (present)
- [✓] Erlang
  - [✓] Development
  - [✓] Associations
- [✓] Erlang Documentation

Description

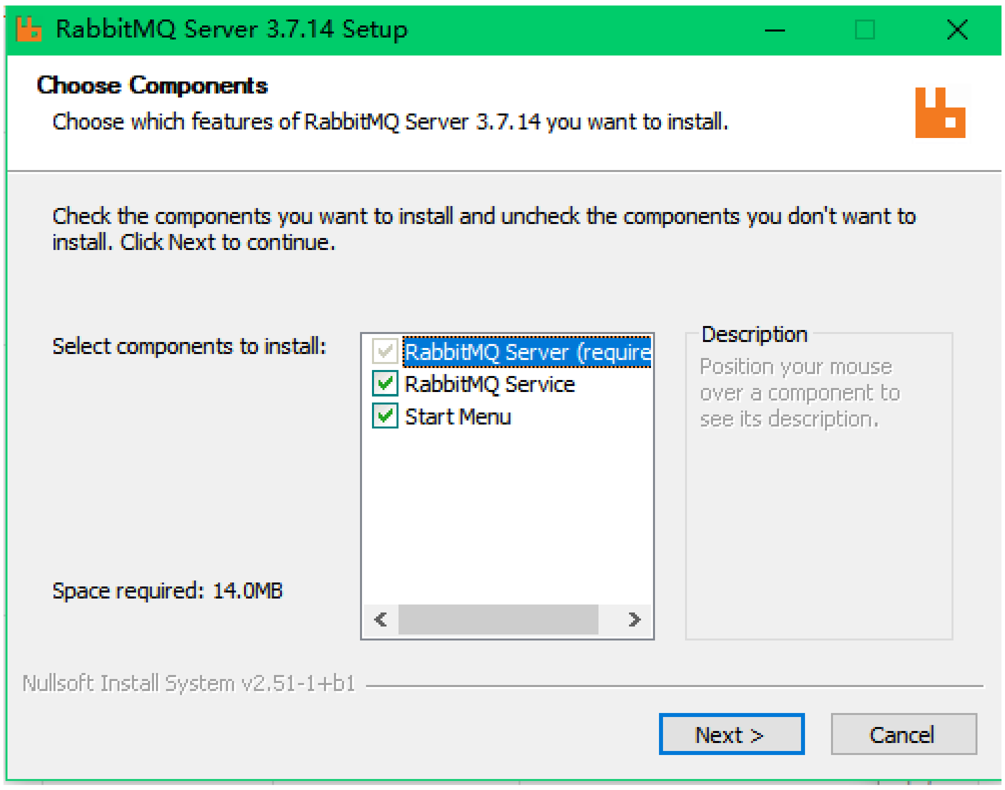
Position your mouse over a component to see its description.

Space required: 311.0MB

Nullsoft Install System v2.46

2.2. 安装RabbitMQ

> 这里使用的版本与课程中演示的版本一致, 均为rabbitmq-server-3.7.14.exe
>
> https://dl.bintray.com/rabbitmq/all/rabbitmq-server/3.7.14/rabbitmq-server-3.7.14.exe


RabbitMQ Server 3.7.14 Setup

Choose Components

Choose which features of RabbitMQ Server 3.7.14 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:
- [✓] RabbitMQ Server (require
- [✓] RabbitMQ Service
- [✓] Start Menu

Description
Position your mouse over a component to see its description.

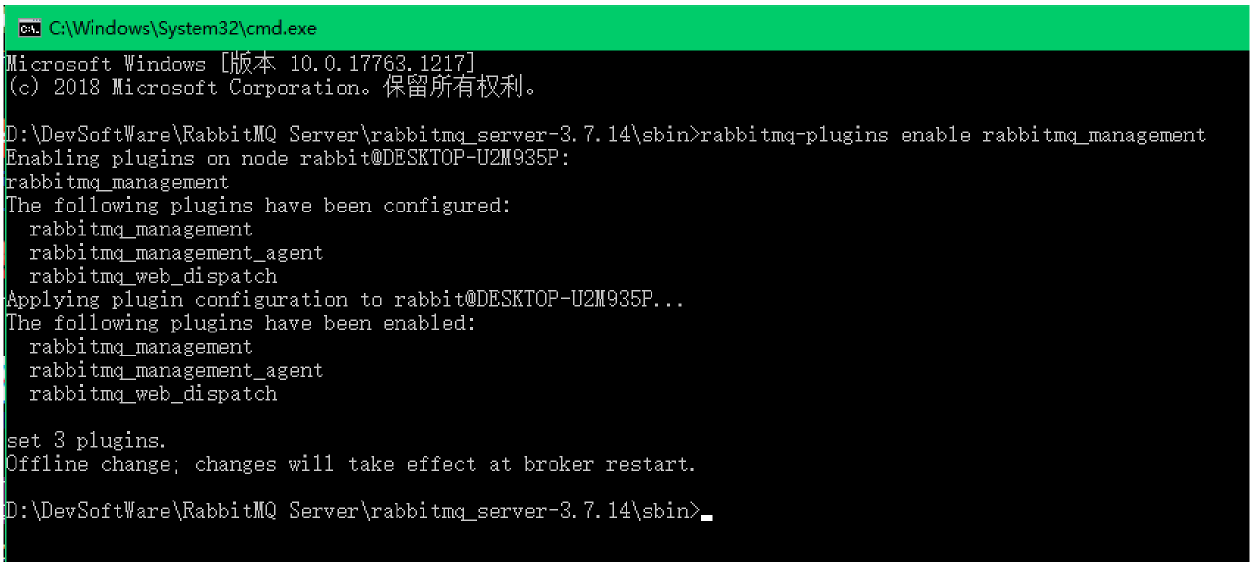
Space required: 14.0MB

Nullsoft Install System v2.51-1+b1

2.3. 启动RabbitMQ管理功能

进入RabbitMQ安装目录下的sbin目录, 命令行执行

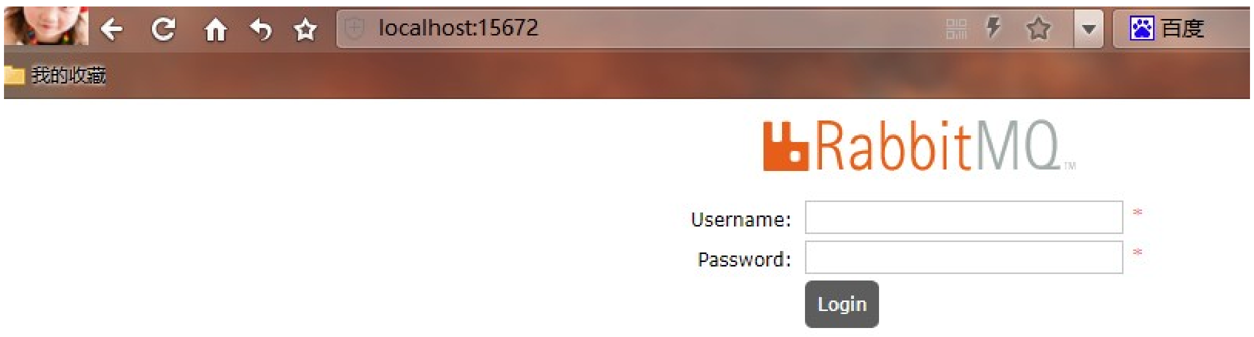
```
D:\DevSoftWare\RabbitMQ Server\rabbitmq_server-3.7.14\sbin>rabbitmq-plugins enable r
```


C:\Windows\System32\cmd.exe

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

D:\DevSoftWare\RabbitMQ Server\rabbitmq_server-3.7.14\sbin>rabbitmq-plugins enable rabbitmq_management
Enabling plugins on node rabbit@DESKTOP-U2M935P:
rabbitmq_management
The following plugins have been configured:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch
Applying plugin configuration to rabbit@DESKTOP-U2M935P...
The following plugins have been enabled:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch

set 3 plugins.
Offline change; changes will take effect at broker restart.

D:\DevSoftWare\RabbitMQ Server\rabbitmq_server-3.7.14\sbin>
```

2.4. 验证是否安装成功

访问地址http://localhost:15672/ 查看是否安装成功

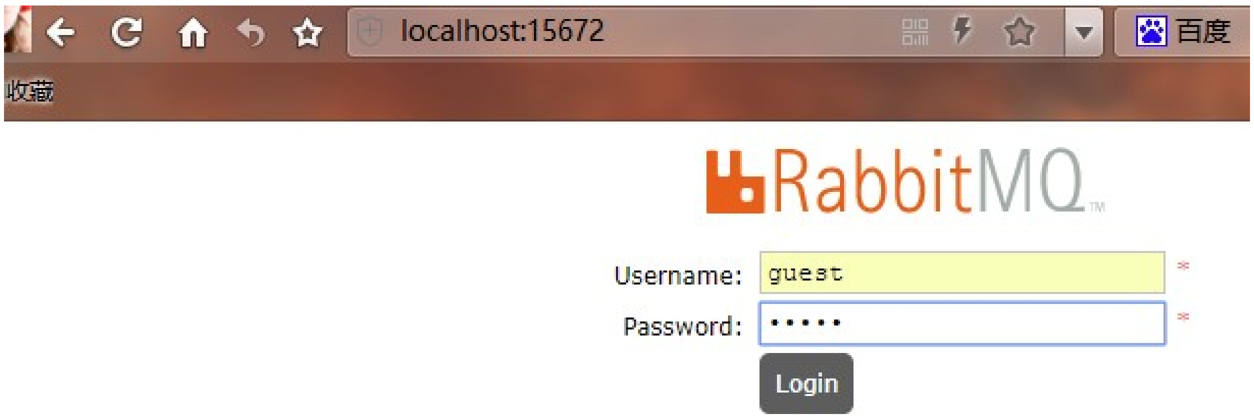


Username:

Password:

Login

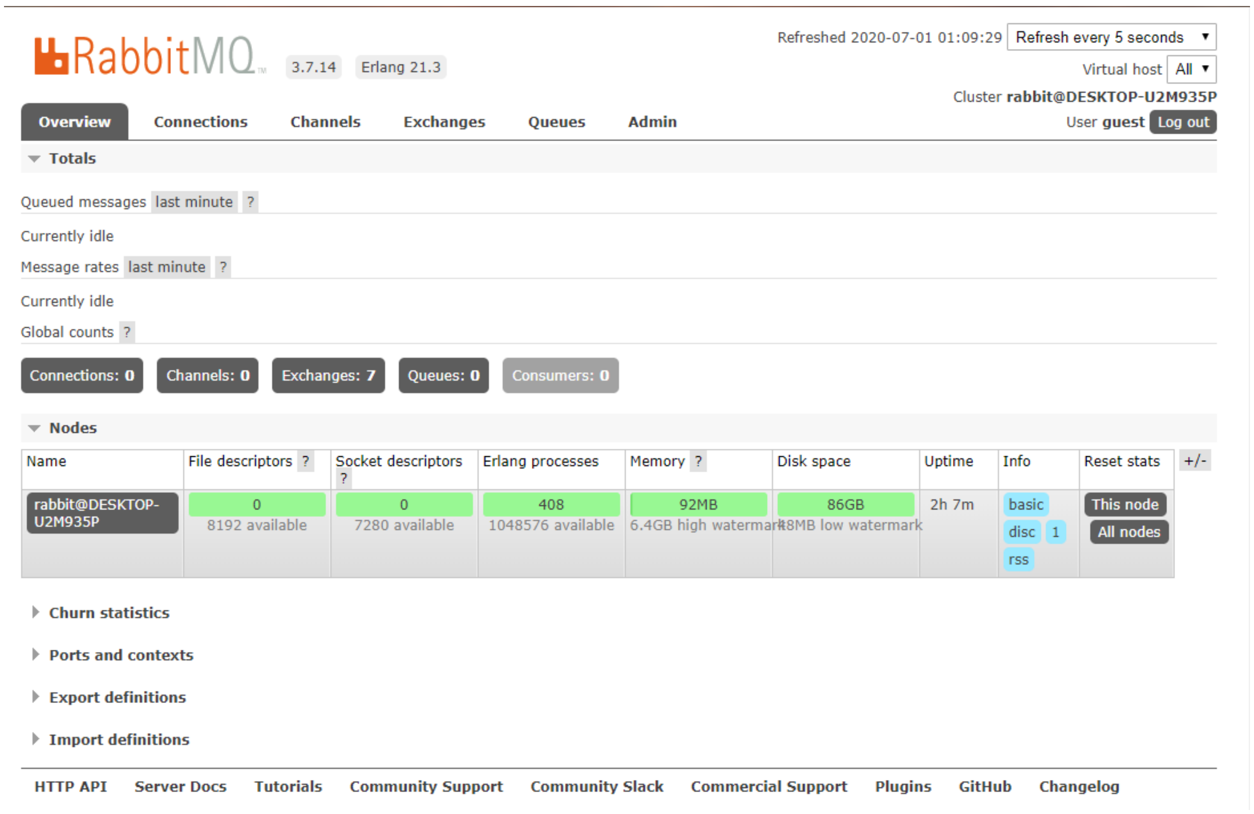
输入用户名guest, 密码guest, 点击"Login"



Username: guest

Password: •••••

Login



3.7.14 Erlang 21.3

Refreshed 2020-07-01 01:09:29 Refresh every 5 seconds

Virtual host All

Cluster rabbit@DESKTOP-U2M935P

User guest Log out

Overview Connections Channels Exchanges Queues Admin

Totals

Queued messages last minute ?

Currently idle

Message rates last minute ?

Currently idle

Global counts ?

Connections: 0 Channels: 0 Exchanges: 7 Queues: 0 Consumers: 0

Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats	+/-
rabbit@DESKTOP-U2M935P	0 8192 available	0 7280 available	408 1048576 available	92MB 6.4GB high watermark	86GB 48MB low watermark	2h 7m	basic disc 1 rss	This node All nodes	

Churn statistics

Ports and contexts

Export definitions

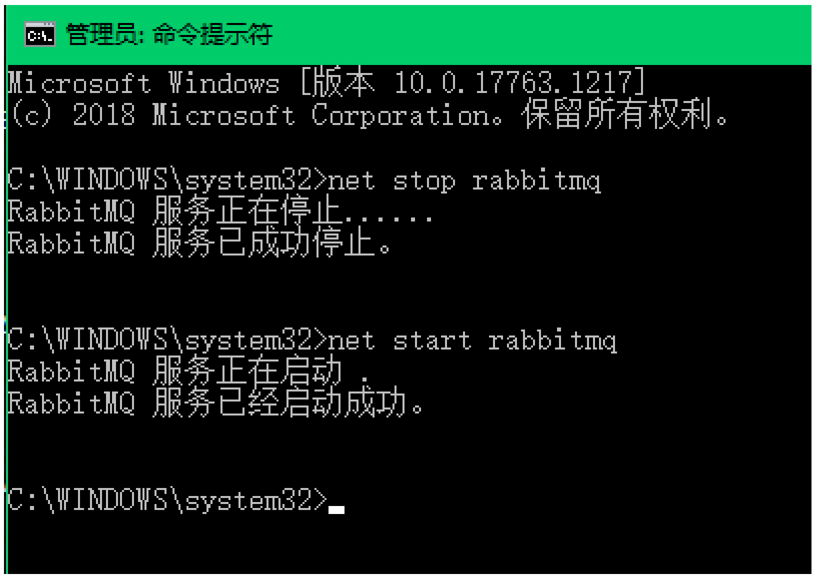
Import definitions

HTTP API Server Docs Tutorials Community Support Community Slack Commercial Support Plugins GitHub Changelog

2.5. 注意事项

1) 注意事项: 如果在安装插件之前已经启动了rabbitmq.需要重启rabbitmq以使上一步安装的插件生效

```
# 需要在管理员权限下运行
C:\WINDOWS\system32>net stop rabbitmq

C:\WINDOWS\system32>net start rabbitmq
```

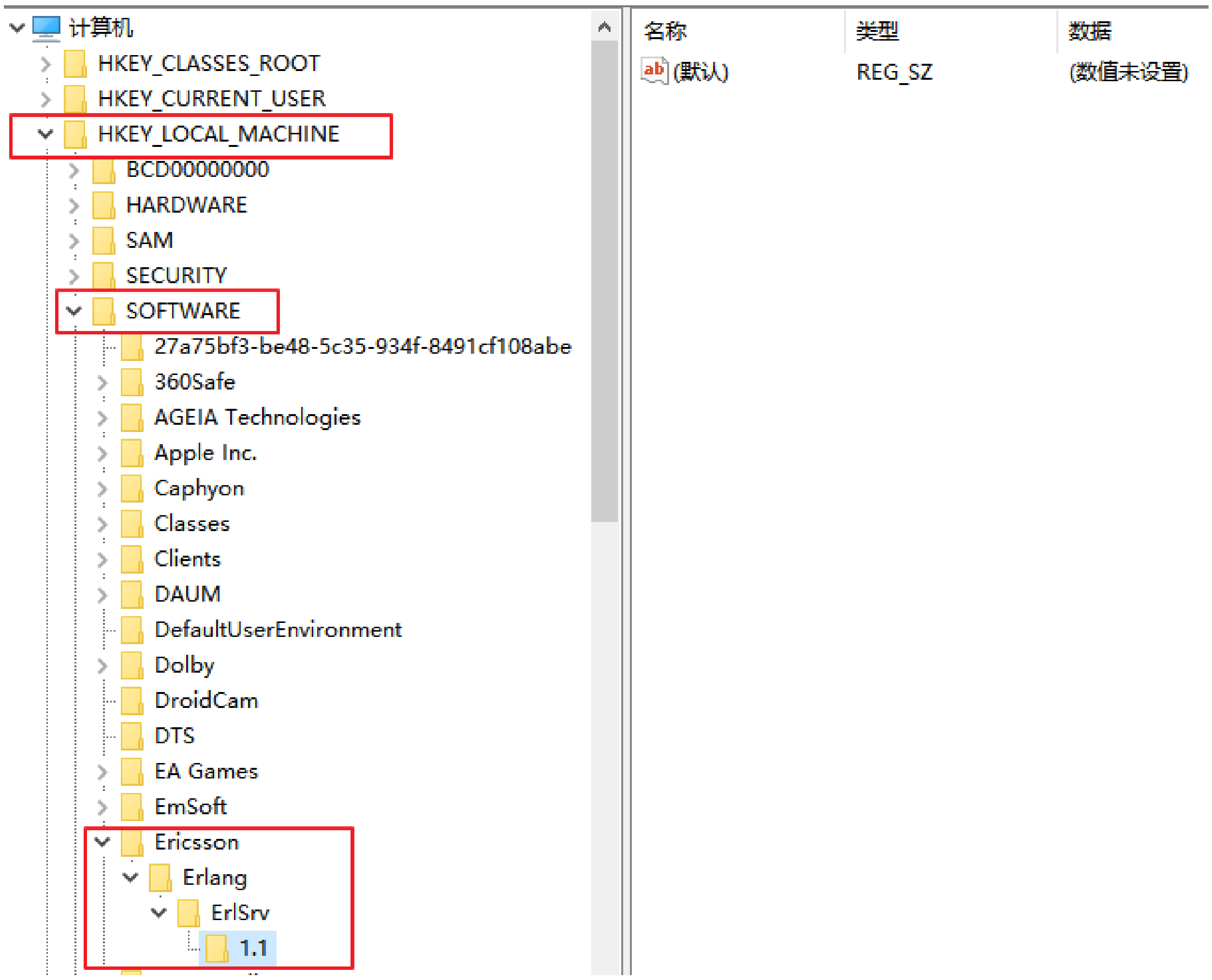


管理员: 命令提示符

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

C:\WINDOWS\system32>net stop rabbitmq
RabbitMQ 服务正在停止......
RabbitMQ 服务已成功停止。


C:\WINDOWS\system32>net start rabbitmq
RabbitMQ 服务正在启动 .
RabbitMQ 服务已经启动成功。


C:\WINDOWS\system32>
```

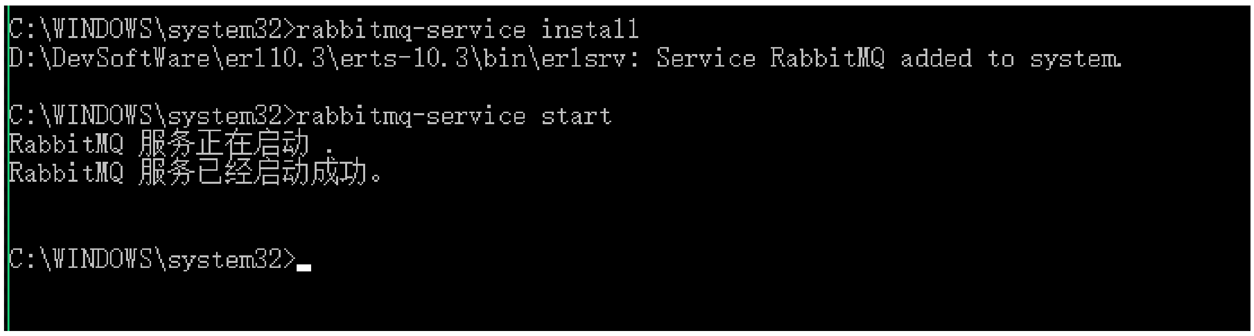
2) 如果遇到服务无法成功启动的情况下, 可以尝试如下操作:

打开注册表编辑器(搜索框内输入 `regedit`), 清除 `HKEY_LOCAL_MACHINE\SOFTWARE\Ericsson\Erlang\ErlSrv\1.1` 下所有内容



打开命令行, 执行

```
rabbitmq-service install

rabbitmq-service start
```

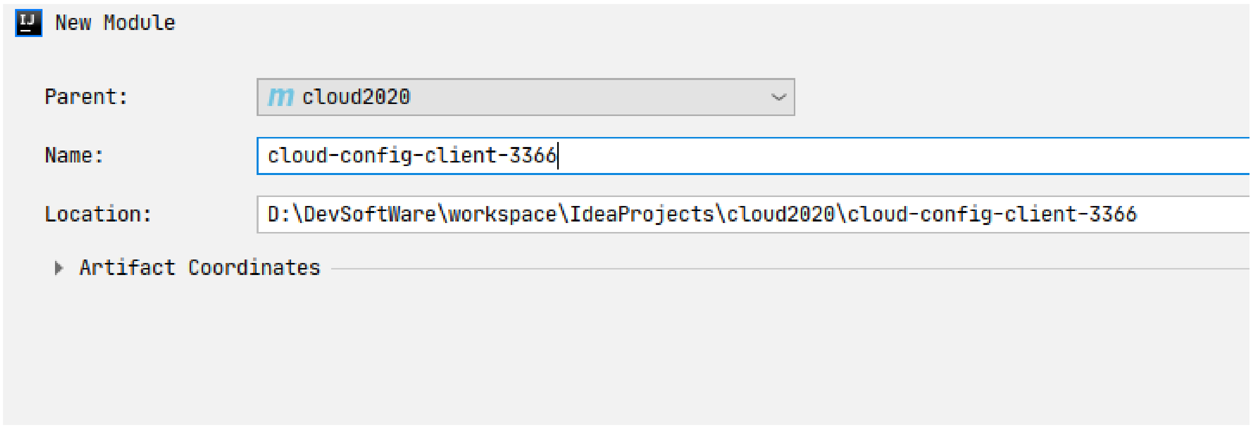


```
C:\WINDOWS\system32>rabbitmq-service install
D:\DevSoftWare\erl10.3\erts-10.3\bin\erlsrv: Service RabbitMQ added to system.

C:\WINDOWS\system32>rabbitmq-service start
RabbitMQ 服务正在启动 .
RabbitMQ 服务已经启动成功。


C:\WINDOWS\system32>_
```

3. SpringCloud Bus动态刷新全局广播

3.1. 演示广播效果，增加复杂度，再以3355为模板再制作一个3366

3.1.1. 新建cloud-config-client-3366模块


New Module

Parent: cloud2020

Name: cloud-config-client-3366

Location: D:\DevSoftWare\workspace\IdeaProjects\cloud2020\cloud-config-client-3366

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-3366</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. YML

bootstrap.yml

```yaml
server:
  port: 3366
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
      uri: http://localhost:3344 # 配置中心地址
# 服务注册到eureka地址
eureka:
  client:
    service-url:
      defaultZone: http://eureka7001.com:7001/eureka
management: # 暴露监控端点
  endpoints:
    web:
      exposure:
        include: "*"

```

3.1.4. 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;

@EnableEurekaClient
@SpringBootApplication
public class ConfigClientMain3366 {
    public static void main(String[] args) {
        SpringApplication.run(ConfigClientMain3366.class, args);
    }
}

```

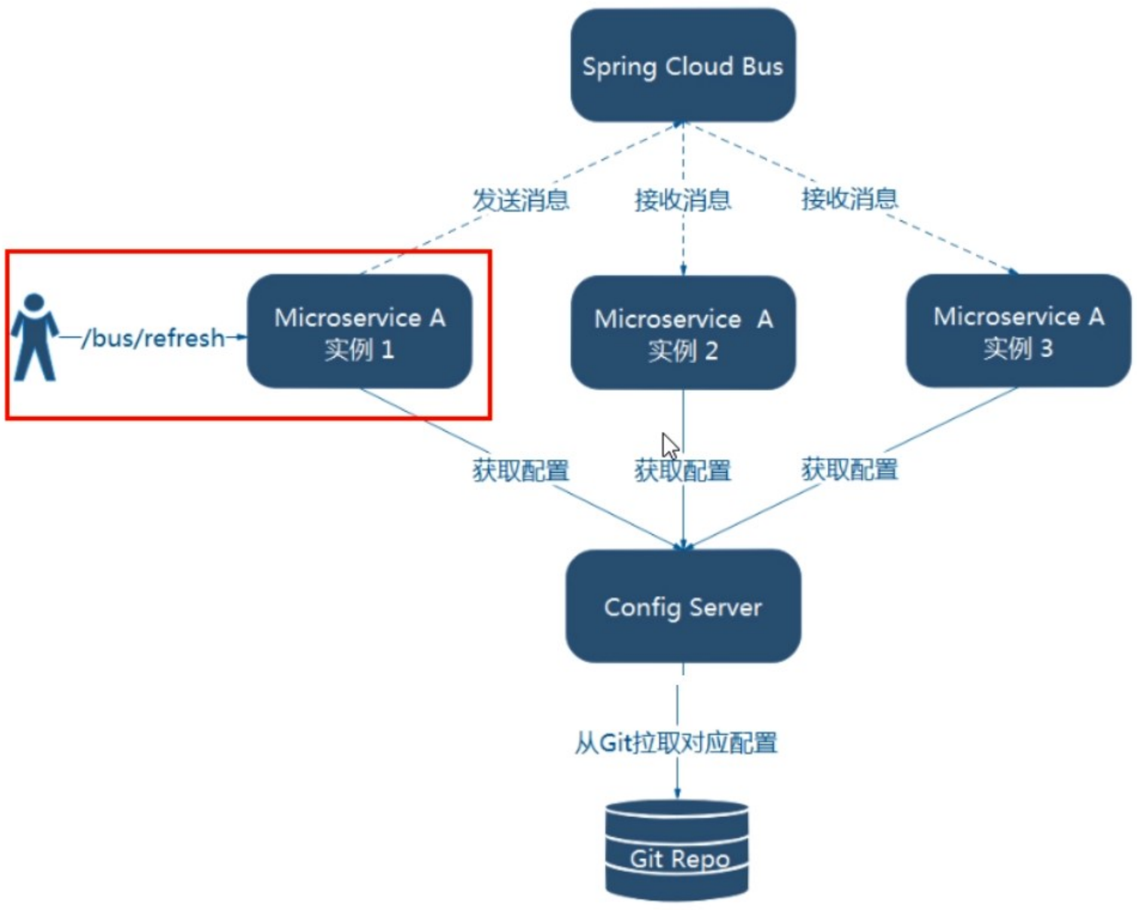
3.1.5. 业务类

controller:

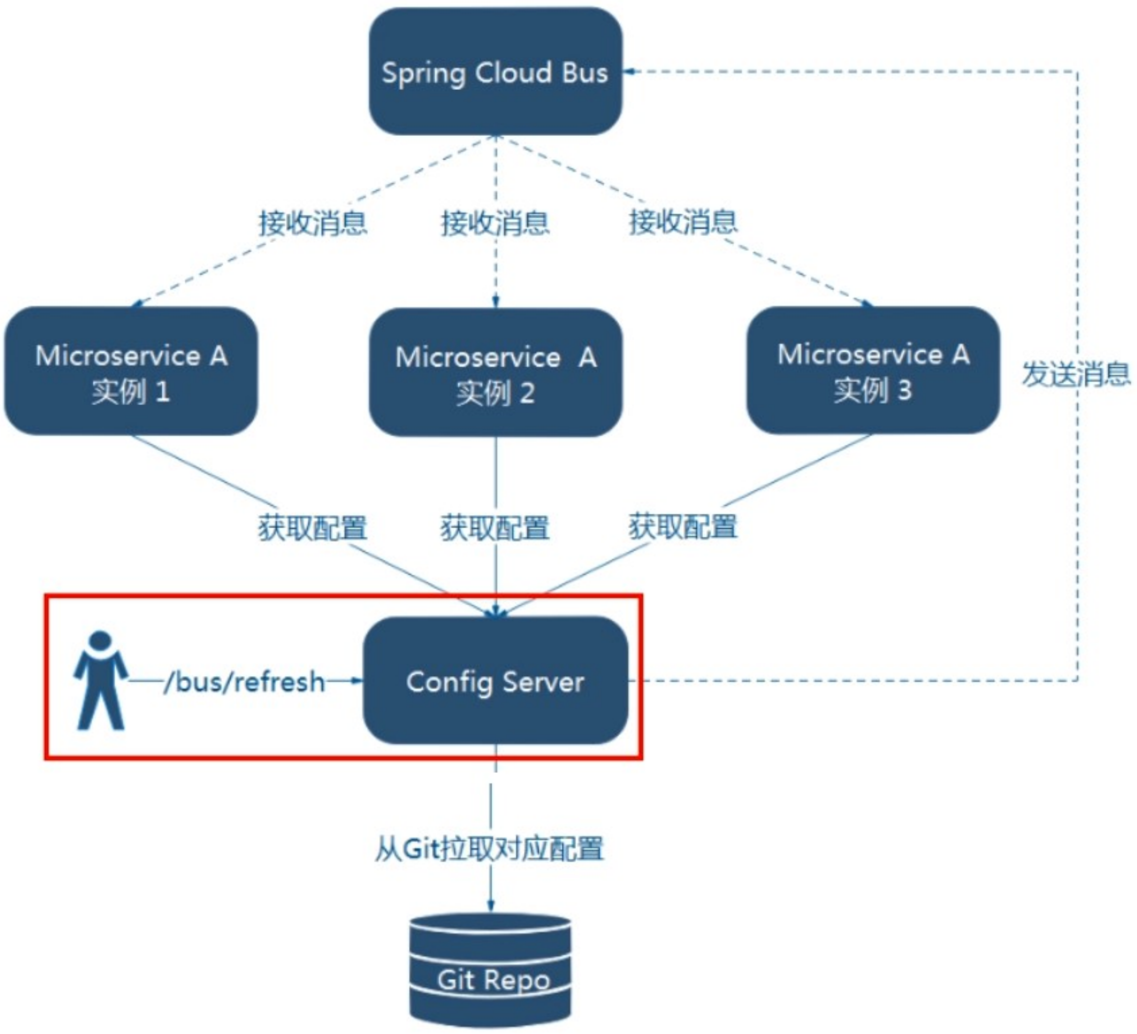
```java

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("${server.port}")
    private String serverPort;

    @Value("${config.info}")
    private String configInfo;


    @GetMapping("/configInfo")
    public String getConfigInfo() {
        return "serverPort:" + serverPort + "\t\n\n configInfo: " + configInfo;
    }
}
```

3.2. 设计思想

1) 利用消息总线触发一个客户端/bus/refresh,而刷新所有客户端的配置



2) 利用消息总线触发一个服务端ConfigServer的/bus/refresh端点,而刷新所有客户端的配置（更加推荐）



第二种的架构显然更加合适，第一种不适合的原因如下

- 打破了微服务的职责单一性，因为微服务本身是业务模块，它本不应该承担配置刷新职责
- 破坏了微服务各节点的对等性
- 有一定的局限性。例如，微服务在迁移时，它的网络地址常常会发生变化，此时如果想要做到自动刷新，那就会增加更多的修改

3.3. 给cloud-config-center-3344配置中心服务端添加消息总线支持

3.3.1. POM

修改内容:

```
        <!--添加消息总线RabbitMQ支持-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-bus-amqp</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.
    <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>
        <!--添加消息总线RabbitMQ支持-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-bus-amqp</artifactId>
        </dependency>
        <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>

</project>
```

3.3.2. YML

application.yaml

修改内容:

```yaml
spring:
  # RabbitMQ相关配置
  rabbitmq:
    host: localhost
    port: 5672
    username: guest
    password: guest
# 暴露bus刷新配置的端口
management:
  endpoints:
    web:
      exposure:
        include: 'bus-refresh'
```

完整内容:

```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githu
          search-paths:
            - springcloud-config
      label: master
  # RabbitMQ相关配置
  rabbitmq:
    host: localhost
    port: 5672
    username: guest
    password: guest
eureka:
  client:
    service-url:
      defaultZone:  http://localhost:7001/eureka

# 暴露bus刷新配置的端口
management:
  endpoints:
    web:
      exposure:
        include: 'bus-refresh'
```

3.4. 给cloud-config-center-3355客户端添加消息总线支持

3.4.1. POM

修改内容:

```xml
        <!--添加消息总线RabbitMQ支持-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-bus-amqp</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.c
    <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>
        <!--添加消息总线RabbitMQ支持-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-bus-amqp</artifactId>
        </dependency>
        <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>
```

3.4.2. YML

bootstrap.yml

修改内容:

```yaml
spring:
  # RabbitMQ相关配置，15672是web管理界面的端口;5672是MQ的端口
  rabbitmq:
    host: localhost
    port: 5672
    username: guest
    password: guest
```

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
  # RabbitMQ相关配置, 15672是web管理界面的端口;5672是MQ的端口
  rabbitmq:
    host: localhost
    port: 5672
    username: guest
    password: guest
eureka:
  client:
    service-url:
      defaultZone: http://eureka7001.com:7001/eureka

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

3.5. 给cloud-config-center-3366客户端添加消息总线支持

3.5.1. POM

修改内容:

```xml
        <!--添加消息总线RabbitMQ的支持-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-bus-amqp</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.
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloud-config-client-3366</artifactId>

    <dependencies>
        <!--添加消息总线RabbitMQ的支持-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-bus-amqp</artifactId>
        </dependency>
        <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>
```

3.5.2. YML

bootstrap.yml

修改内容:

```yaml
spring:
  # RabbitMQ配置
  rabbitmq:
    host: localhost
    port: 5672
    username: guest
    password: guest
```

全部内容:

```yaml
server:
  port: 3366
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
      uri: http://localhost:3344 # 配置中心地址

  # RabbitMQ配置
  rabbitmq:
    host: localhost
    port: 5672
    username: guest
    password: guest
# 服务注册到eureka地址
eureka:
  client:
    service-url:
      defaultZone: http://eureka7001.com:7001/eureka
management: # 暴露监控端点
  endpoints:
    web:
      exposure:
        include: "*"
```

3.6. 测试

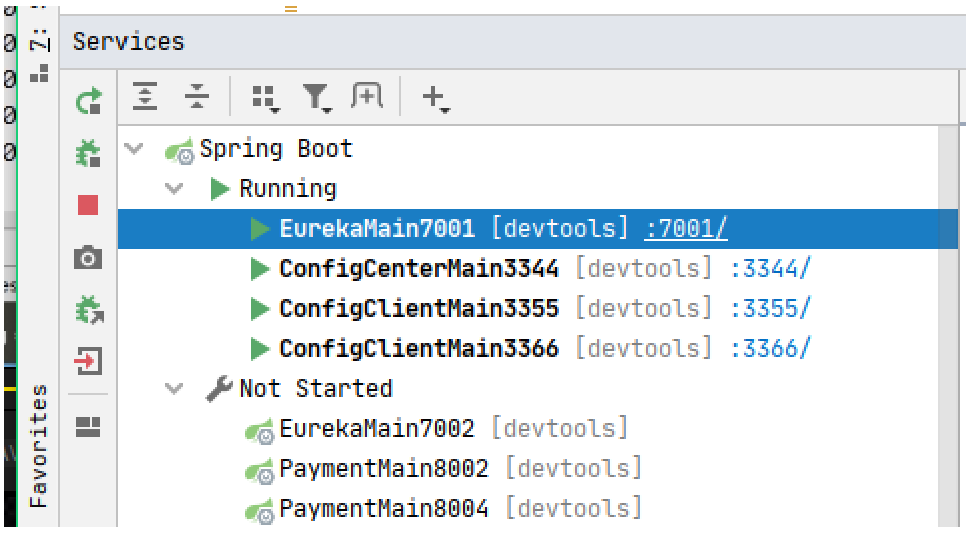
3.6.0. 启动所需服务

启动EurekaMain7001

启动ConfigCenterMain3344

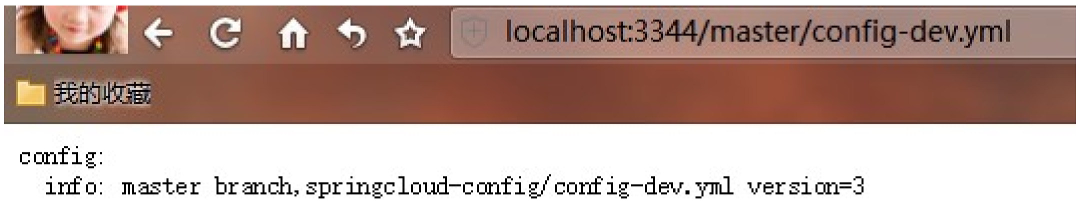
启动ConfigClientMain3355

启动ConfigClientMain3366



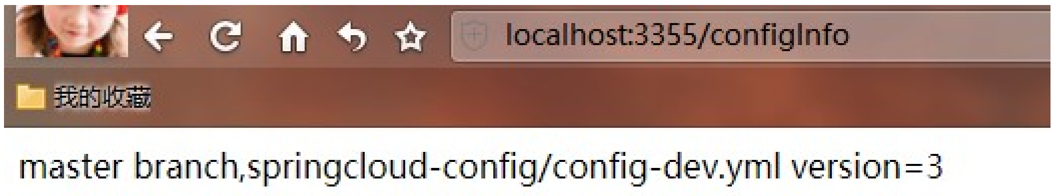
```
Spring Boot
  Running
    EurekaMain7001 [devtools] :7001/
    ConfigCenterMain3344 [devtools] :3344/
    ConfigClientMain3355 [devtools] :3355/
    ConfigClientMain3366 [devtools] :3366/
  Not Started
    EurekaMain7002 [devtools]
    PaymentMain8002 [devtools]
    PaymentMain8004 [devtools]
```

先通过配置中心3344查看当前Github上config-dev.yml的配置信息
http://localhost:3344/master/config-dev.yml



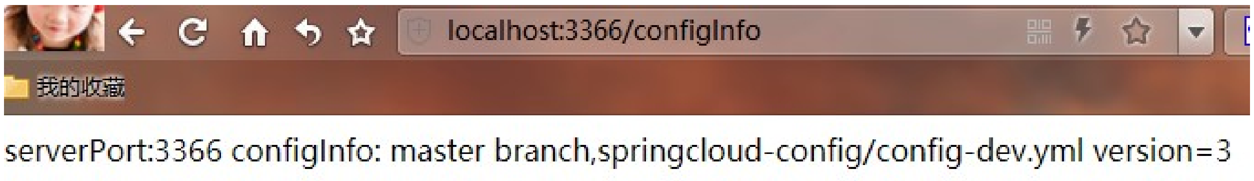
```
config:
  info: master branch,springcloud-config/config-dev.yml version=3
```

再通过配置客户端3355查看查看config-dev.yml的配置信息http://localhost:3355/configInfo



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

然后通过配置客户端3366查看查看config-dev.yml的配置信息
http://localhost:3366/configInfo



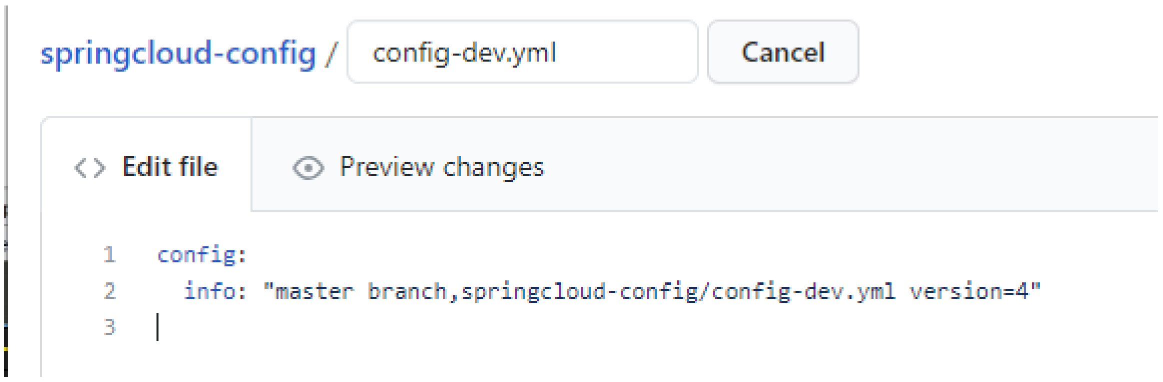
```
serverPort:3366 configInfo: master branch,springcloud-config/config-dev.yml version=3
```

可以看到, 三个服务读取到的配置信息一致

3.6.1. 运维工程师

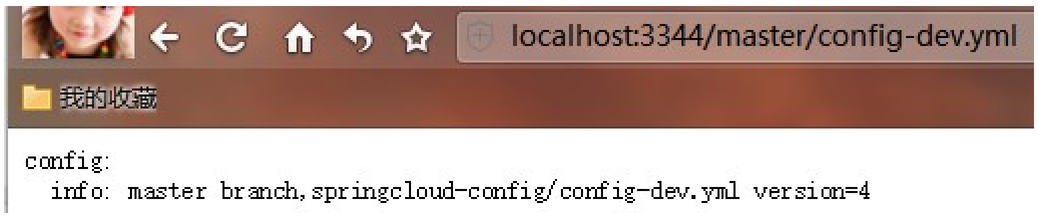
修改Github上的配置文件config-dev.yml(为了方便起见, 直接在Github网站上对远程仓库上的该配置文件进行修改)

```yaml
config:
  info: "master branch,springcloud-config/config-dev.yml version=4"
```



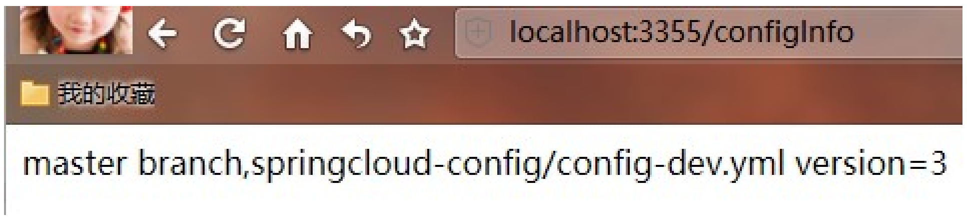
```
config:
  info: "master branch,springcloud-config/config-dev.yml version=4"
```

先通过配置中心3344查看当前Github上config-dev.yml的配置信息
http://localhost:3344/master/config-dev.yml



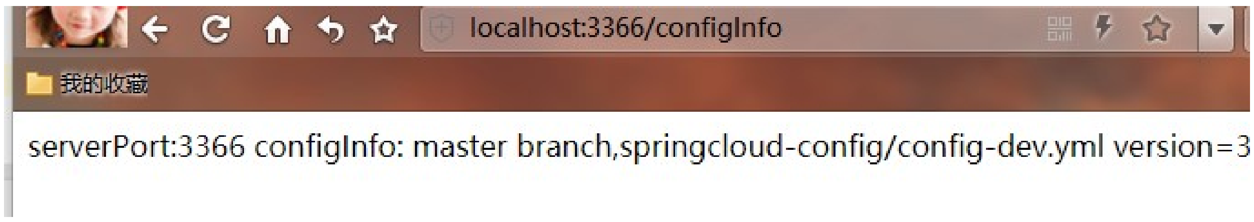
```
config:
  info: master branch,springcloud-config/config-dev.yml version=4
```

再通过配置客户端3355查看查看config-dev.yml的配置信息http://localhost:3355/configInfo



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

然后通过配置客户端3366查看查看config-dev.yml的配置信息
http://localhost:3366/configInfo

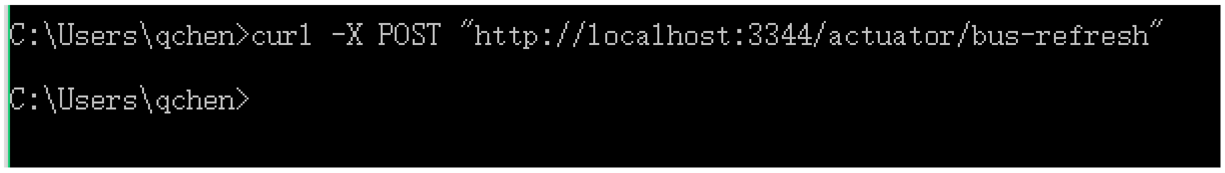


```
serverPort:3366 configInfo: master branch,springcloud-config/config-dev.yml version=3
```

可以看到, 只有3344的读取到的配置信息发生了变化, 3355和3366仍旧保持不变

命令行执行

```
curl -X POST "http://localhost:3344/actuator/bus-refresh"
```

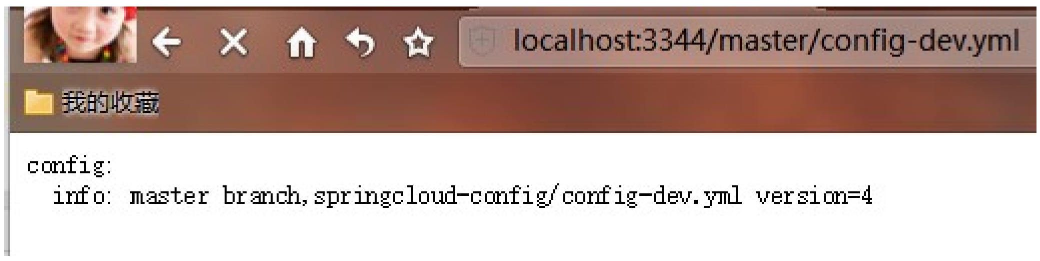


```
C:\Users\qchen>curl -X POST "http://localhost:3344/actuator/bus-refresh"

C:\Users\qchen>
```

一次发送，处处生效

3.6.2. 配置中心

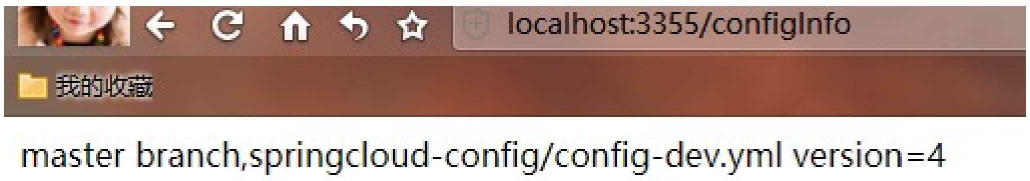
浏览器访问: http://config-3344.com/config-dev.yml



```
config:
  info: master branch,springcloud-config/config-dev.yml version=4
```

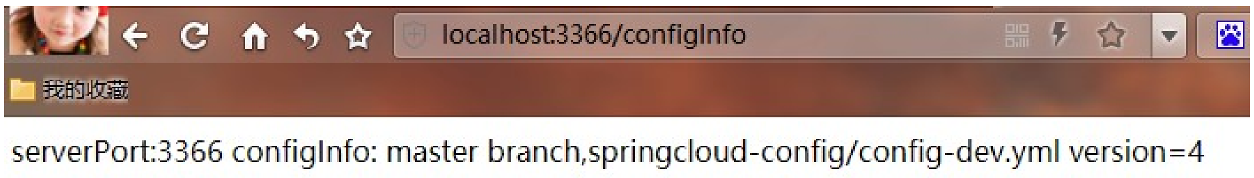
3.6.3. 客户端

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



```
master branch,springcloud-config/config-dev.yml version=4
```

浏览器访问: http://localhost:3366/configInfo



```
serverPort:3366 configInfo: master branch,springcloud-config/config-dev.yml version=4
```

获取配置信息，发现都已经刷新了

3.7. 结论

一次修改，广播通知，处处生效

4. SpringCloud Bus动态刷新定点通知

4.1. 目标

不想全部通知，只想定点通知

4.2. 简单一句话

指定具体某一个实例生效而不是全部

公式：http://localhost:配置中心的端口号/actuator/bus-refresh/{destination}

/bus/refresh请求不再发送到具体的服务实例上，而是发给config server并通过destination参数类指定需要更新配置的服务或实例

4.3. 案例

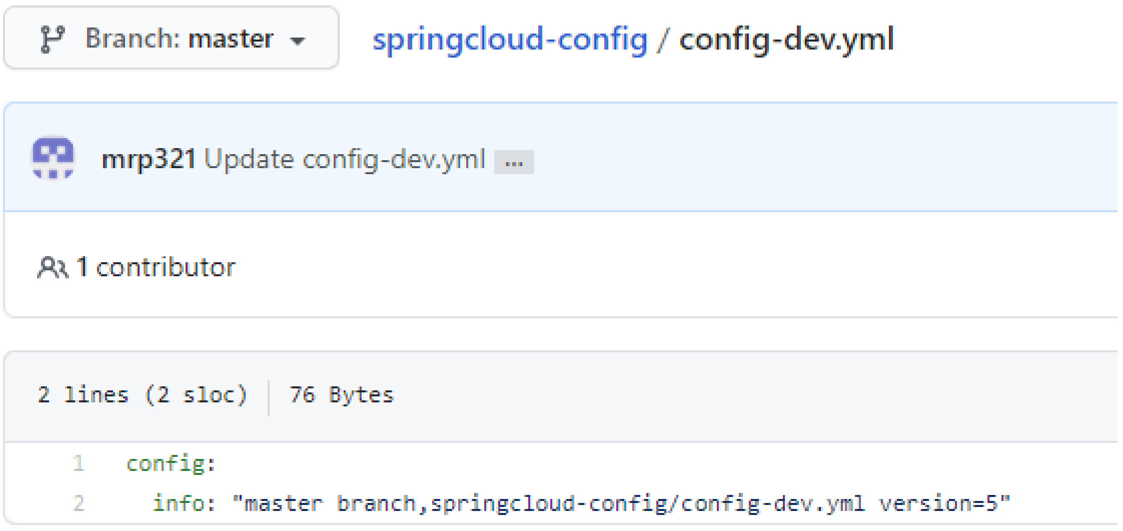
我们这里以刷新运行在3355端口上的config-client为例: 只通知3355, 不通知3366

```
curl -X POST "http://localhost:3344/actuator/bus-refresh/config-client:3355"
```

4.3.1. 运维工程师

修改Github上远程仓库中config-dev.yml的配置信息

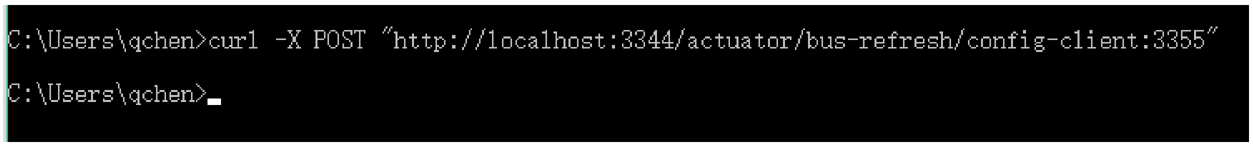
```yaml
config:
  info: "master branch,springcloud-config/config-dev.yml version=5"
```


Branch: master
springcloud-config / config-dev.yml
mrp321 Update config-dev.yml
1 contributor
2 lines (2 sloc) | 76 Bytes

```yaml
config:
  info: "master branch,springcloud-config/config-dev.yml version=5"
```

命令行执行以下命令

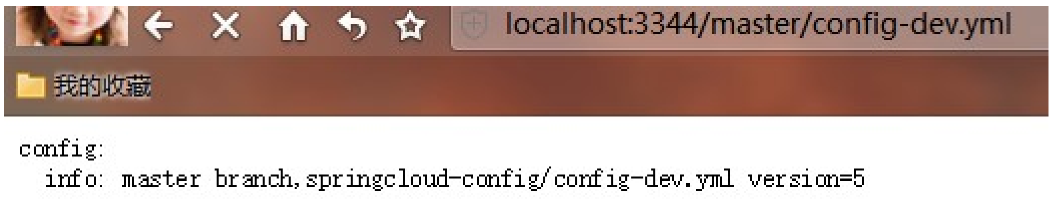
```
curl -X POST "http://localhost:3344/actuator/bus-refresh/config-client:3355"
```



```
C:\Users\qchen>curl -X POST "http://localhost:3344/actuator/bus-refresh/config-client:3355"

C:\Users\qchen>_
```

4.3.2. 配置中心

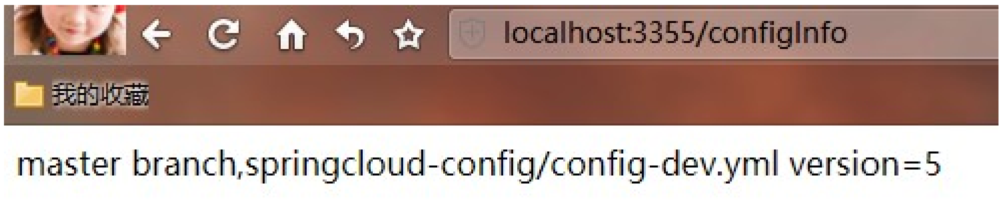
浏览器访问: http://config-3344.com/config-dev.yml



```
config:
  info: master branch,springcloud-config/config-dev.yml version=5
```

4.3.3. 客户端

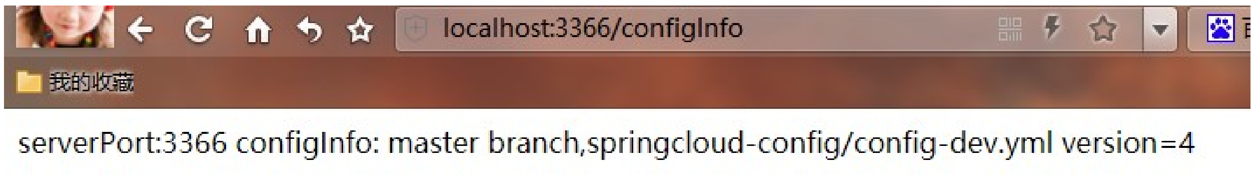
浏览器访问: http://localhost:3355/configInfo


localhost:3355/configInfo

我的收藏

master branch,springcloud-config/config-dev.yml version=5

浏览器访问: http://localhost:3366/configInfo

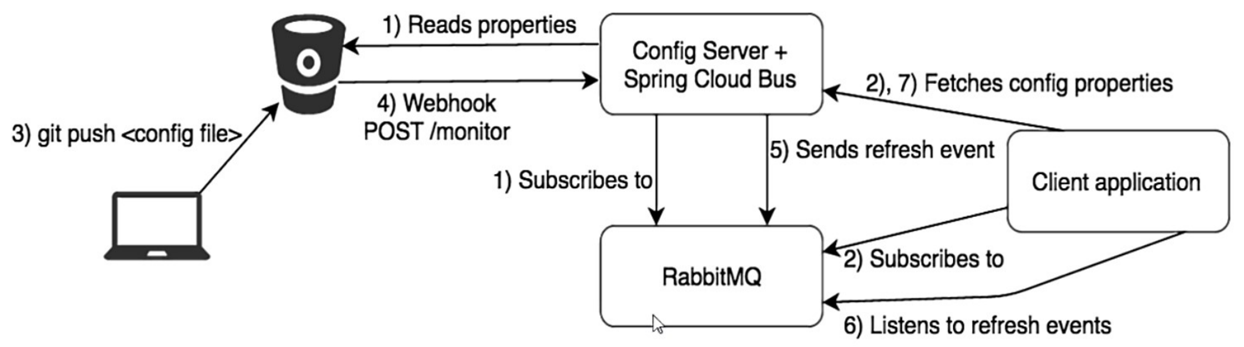

localhost:3366/configInfo

我的收藏

serverPort:3366 configInfo: master branch,springcloud-config/config-dev.yml version=4

可以看到, 3355读取到的配置信息已经发生了变化, 3366还和之前保持一致

4.4. 通知总结


1) Reads properties
Config Server +
Spring Cloud Bus
4) Webhook
POST /monitor
3) git push <config file>
2), 7) Fetches config properties
5) Sends refresh event
1) Subscribes to
Client application
RabbitMQ
2) Subscribes to
6) Listens to refresh events