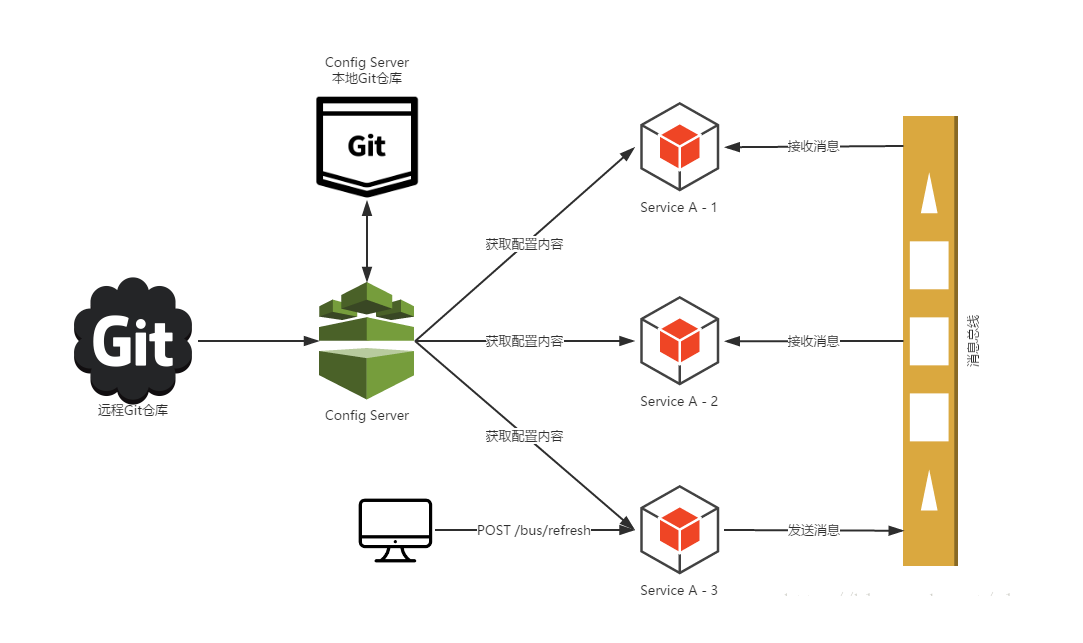
# SpringCloud 组件之 配置中心-中篇

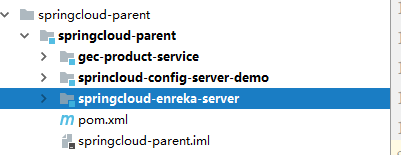
# 1.为何需要配置中心

# 2.分布式配置中心原理



# 3.搭建架构

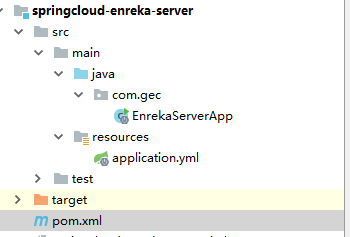
## 3.1.搭建聚合工程



父工程导入依赖

|  |
| --- |
| *<?***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.gec</**groupId**>  <**artifactId**>springcloud-parent</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  <**modules**>  <**module**>springcloud-enreka-server</**module**>  <**module**>gec-product-service</**module**>  <**module**>sprincloud-config-server-demo</**module**>  </**modules**>  <**packaging**>pom</**packaging**>   <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.2.6.RELEASE</**version**>  </**parent**>   <**properties**>  <**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>  <**project.reporting.outputEncoding**>UTF-8</**project.reporting.outputEncoding**>  <**java.version**>1.8</**java.version**>  <**springcloud.version**>Hoxton.SR3</**springcloud.version**>  <**spring-cloud-alibaba.version**>2.1.0.RELEASE</**spring-cloud-alibaba.version**>  </**properties**>   <**dependencyManagement**>  <**dependencies**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-dependencies</**artifactId**>  <**version**>${springcloud.version}</**version**>  <**type**>pom</**type**>  <**scope**>import</**scope**>  </**dependency**>  <**dependency**>  <**groupId**>com.alibaba.cloud</**groupId**>  <**artifactId**>spring-cloud-alibaba-dependencies</**artifactId**>  <**version**>${spring-cloud-alibaba.version}</**version**>  <**type**>pom</**type**>  <**scope**>import</**scope**>  </**dependency**>  </**dependencies**>  </**dependencyManagement**>  <**repositories**>  <**repository**>  <**id**>central</**id**>  <**url**>http://maven.aliyun.com/nexus/content/groups/public/</**url**>  </**repository**>  </**repositories**>  <**pluginRepositories**>  <**pluginRepository**>  <**id**>spring-snapshots</**id**>  <**name**>Spring Snapshots</**name**>  <**url**>http://repo.spring.io/libs-snapshot-local</**url**>  <**snapshots**>  <**enabled**>true</**enabled**>  </**snapshots**>  </**pluginRepository**>  <**pluginRepository**>  <**id**>spring-milestones</**id**>  <**name**>Spring Milestones</**name**>  <**url**>http://repo.spring.io/libs-milestone-local</**url**>  <**snapshots**>  <**enabled**>false</**enabled**>  </**snapshots**>  </**pluginRepository**>  </**pluginRepositories**>   <**build**>  <**plugins**>  <**plugin**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-maven-plugin</**artifactId**>  </**plugin**>  </**plugins**>  </**build**>    </**project**> |

## 3.2.搭建enreka-server



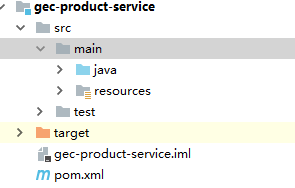
编写启动类

|  |
| --- |
| @SpringBootApplication @EnableEurekaServer **public class** EnrekaServerApp {  **public static void** main( String[] args )  {  SpringApplication.*run*(EnrekaServerApp.**class**,args);  } } |

编写配置文件

|  |
| --- |
| **spring:  application:  name:** eureka-server **server:  port:** 8761 *#端口 #配置eureka server* **eureka:  client:  register-with-eureka: false** *#是否将自己注册到注册中心* **fetch-registry: false** *#是否从eureka中获取注册信息* **service-url:** *#配置暴露给Eureka Client的请求地址* **defaultZone:** http://127.0.0.1:8761/eureka/ |

## 3.3.搭建商品服务



编写配置文件



|  |
| --- |
| **server:  port:** 9000 **eureka:  client:  service-url:  defaultZone:** http://localhost:8761/eureka **spring:  application:  name: 'product-service' name:** df-dev |

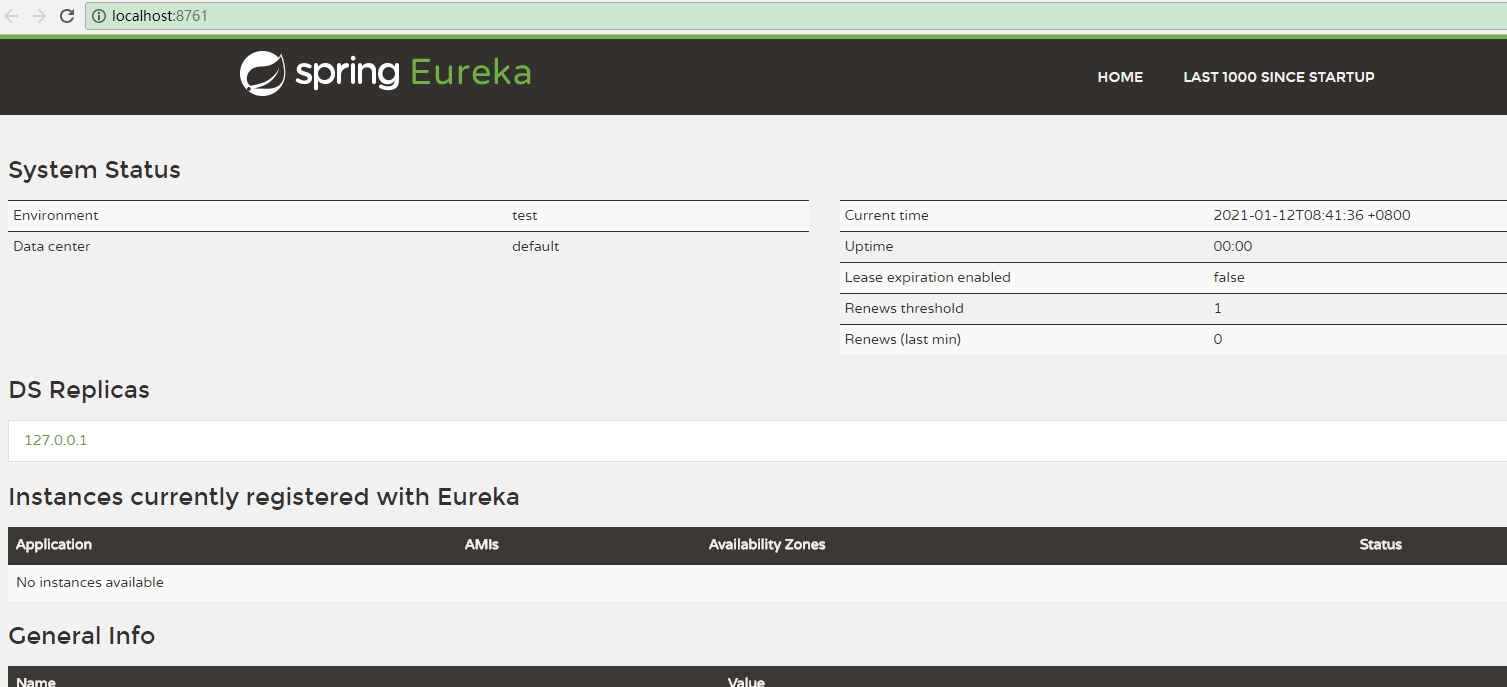
编写控制器

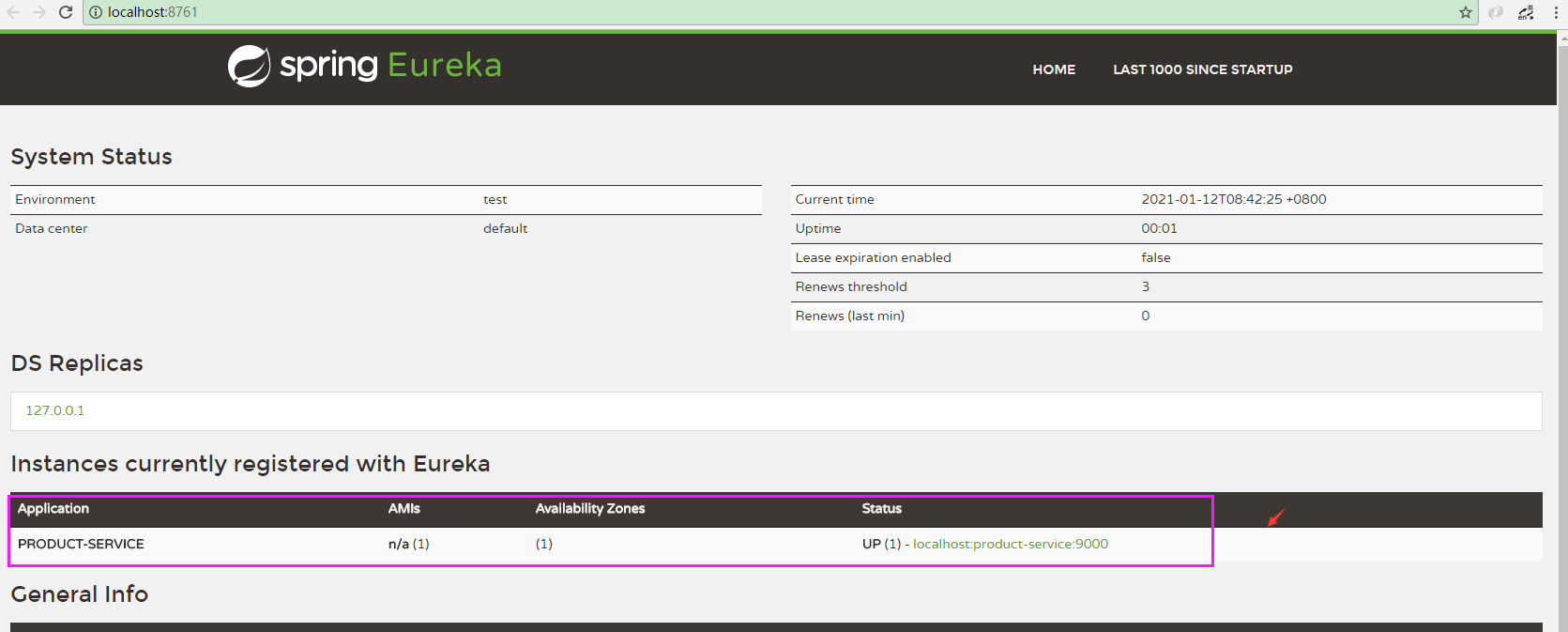
|  |
| --- |
| **package** com.gec.controller;  **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.beans.factory.annotation.Value; **import** org.springframework.cloud.context.config.annotation.RefreshScope; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RestController;  **import** java.util.HashMap; **import** java.util.Map; @RestController**public class** ProductController {    @Value(**"${name}"**)  **private** String **myname**;   @RequestMapping(value = **"/getuserbyid"**)  **public** Map<String,Object> getUserById()  {  Map<String,Object> map = **new** HashMap<String, Object>();  map.put(**"msg"**,**"获取用户信息成功"**);  map.put(**"status"**,200);  **return** map;  }  @RequestMapping(value = **"/getname"**)  **public** String getname()  {  **return myname**;  }    } |

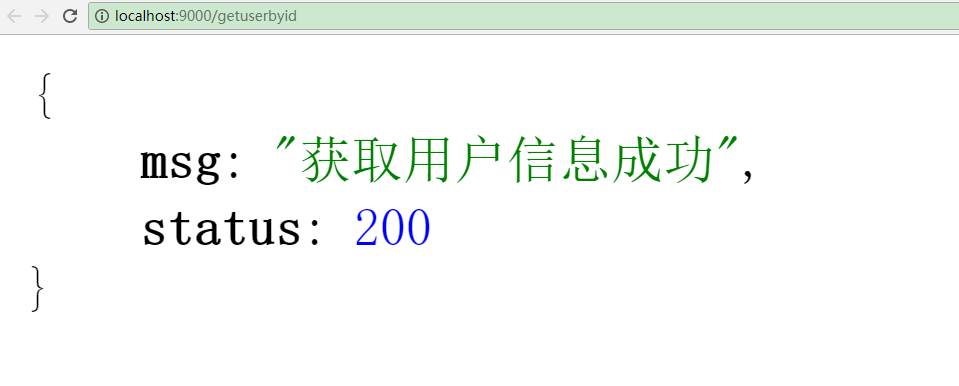
编写启动类

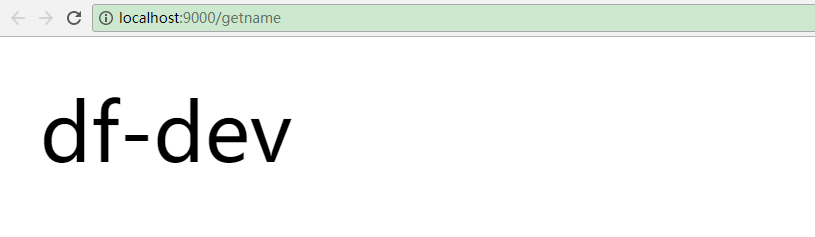
|  |
| --- |
| **package** com.gec;  **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication;   @SpringBootApplication **public class** ProductApp {  **public static void** main( String[] args )  {  SpringApplication.*run*(ProductApp.**class**,args);  } } |

测试访问

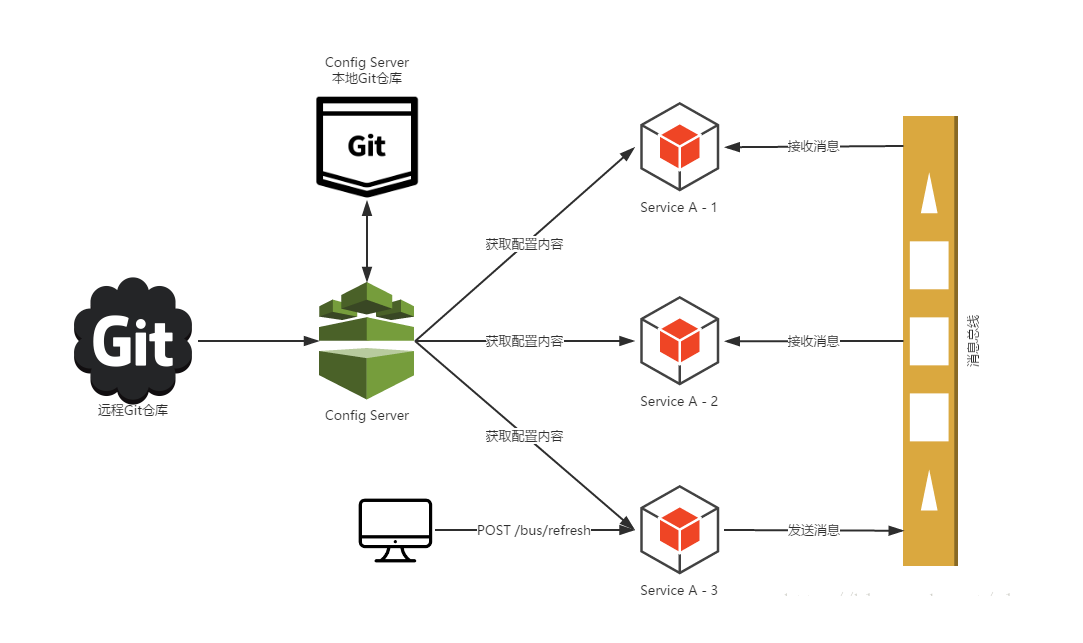




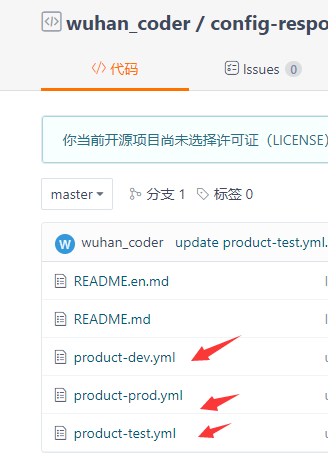




# 4.搭建配置中心



## 4.1.编写三个文件分配拷贝到 gitee



product-dev.yml

|  |
| --- |
| server:  port: 9000  eureka:  client:  service-url:  defaultZone: http://localhost:8761/eureka  spring:  application:  name: 'product-service'  name: df-dev |

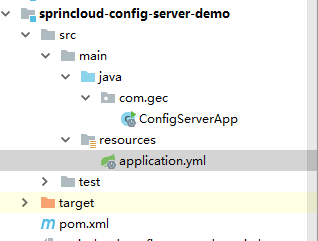
product-test.yml

|  |
| --- |
| server:  port: 9003  eureka:  client:  service-url:  defaultZone: http://localhost:8761/eureka  spring:  application:  name: 'product-service'  name: df-test |

product-prod.yml

|  |
| --- |
| server:  port: 9002  eureka:  client:  service-url:  defaultZone: http://localhost:8761/eureka  spring:  application:  name: 'product-service'  name: df-prod |

## 4.2.搭建配置中心结构



导入依赖

|  |
| --- |
| <**dependencies**>   <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-config-server</**artifactId**>  </**dependency**>   <**dependency**>  <**groupId**>junit</**groupId**>  <**artifactId**>junit</**artifactId**>  <**version**>4.11</**version**>  <**scope**>test</**scope**>  </**dependency**> </**dependencies**> |

编写核心配置文件

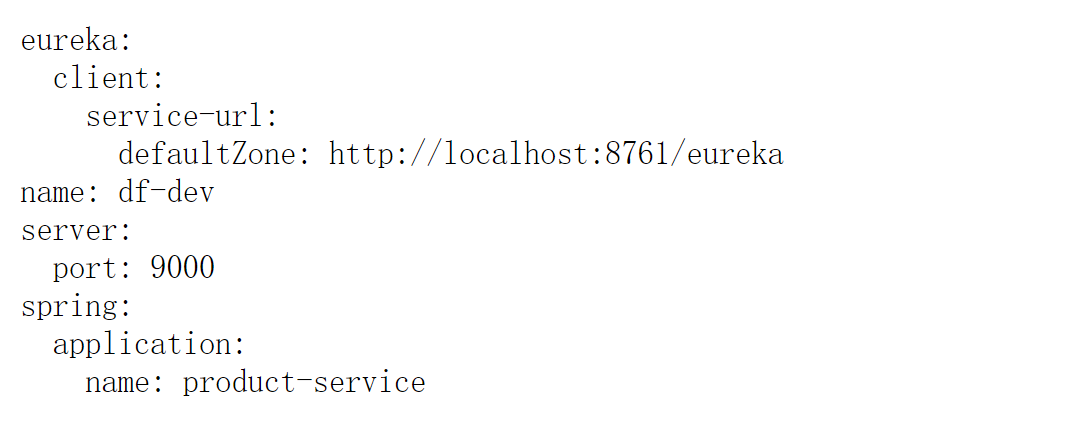
|  |
| --- |
| **server:  port:** 9900 *#服务端口* **spring:  application:  name:** config-server *#指定服务名* **cloud:  config:  server:  git:  uri:** https://gitee.com/wuhancoder\_ios/config-respostry-01.git |

编写启动类

|  |
| --- |
| @SpringBootApplication @EnableConfigServer **public class** ConfigServerApp {  **public static void** main( String[] args )  {  SpringApplication.*run*(ConfigServerApp.**class**,args);  } } |

## 4.3.测试访问

<http://localhost:9900/product-dev.yml>



…….

# 5.改造商品服务模块

导入依赖

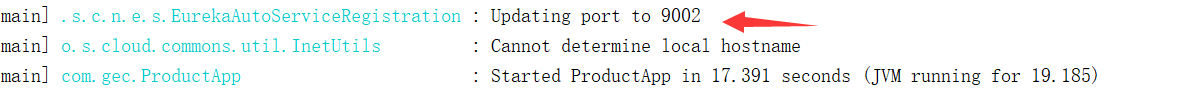
|  |
| --- |
| <**dependencies**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-web</**artifactId**>  </**dependency**>   *<!--引入EurekaClient-->* <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-netflix-eureka-client</**artifactId**>  </**dependency**>    <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-config</**artifactId**>  </**dependency**>  </**dependencies**> |

编写配置文件



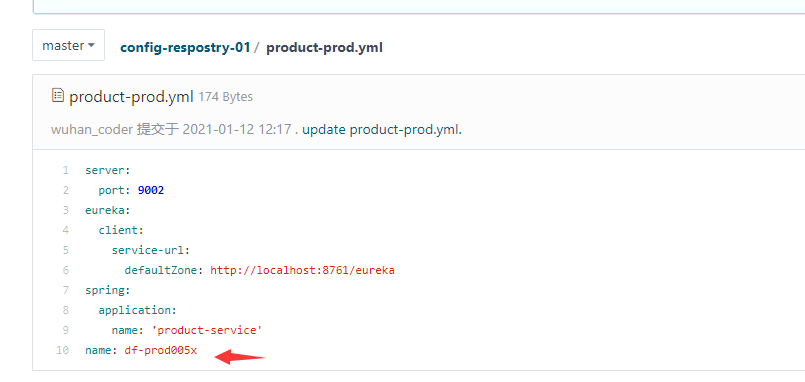
|  |
| --- |
| **spring:  cloud:  config:  name:** product *#应用名称,需要对应git中配置文件名称的前半部分* **profile:** prod *#开发环境* **label:** master *#git中的分支* **uri:** http://localhost:9900 *#config-server的请求地址* |

访问正常通过



# 6.手动刷新

先在gitee中修改文件，然后刷新



<http://localhost:9900/product-prod.yml>

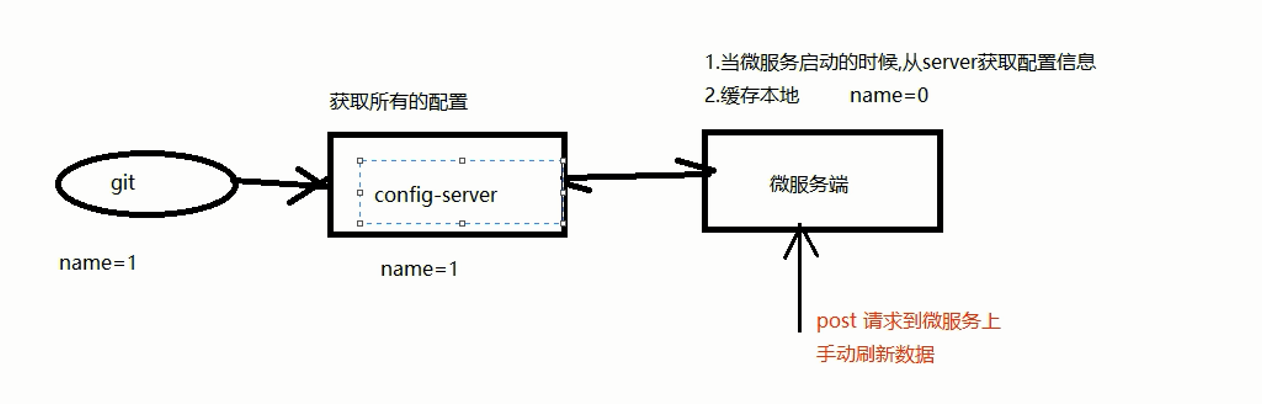


测试访问 9002 的 商品服务 发现并没有刷新

<http://localhost:9002/getname>



为何呢?



gec-product-service

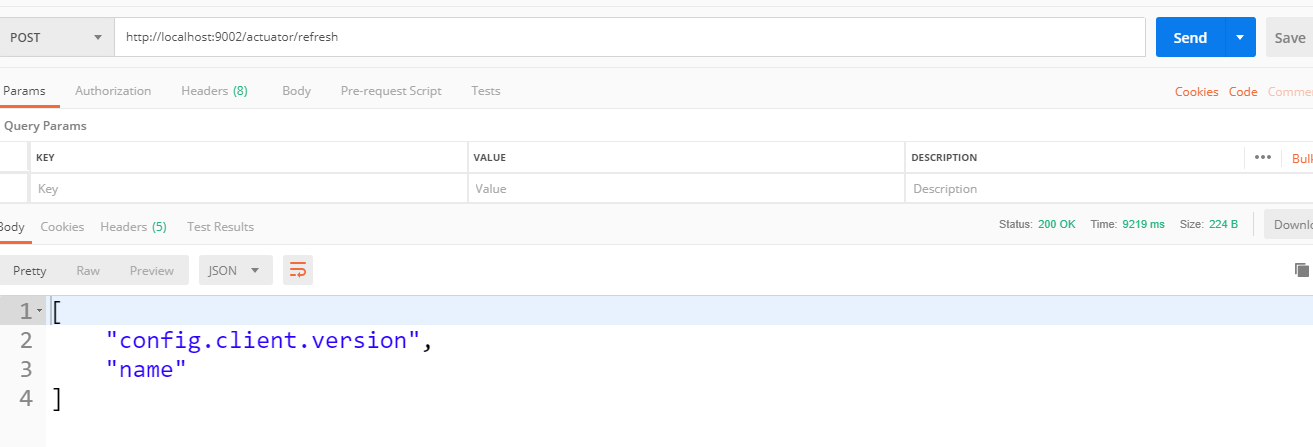
|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-actuator</**artifactId**> </**dependency**> |

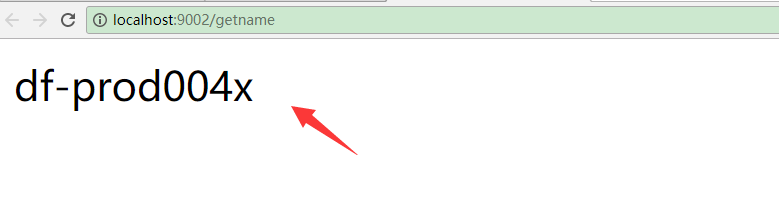
|  |
| --- |
| **spring:  cloud:  config:  name:** product *#应用名称,需要对应git中配置文件名称的前半部分* **profile:** prod *#开发环境* **label:** master *#git中的分支* **uri:** http://localhost:9900 *#config-server的请求地址* **management:  endpoints:  web:  exposure:  include:** refresh |

|  |
| --- |
| @RestController @RefreshScope *// 开* **public class** ProductController {    @Value(**"${name}"**)  **private** String **myname**;   @RequestMapping(value = **"/getuserbyid"**)  **public** Map<String,Object> getUserById()  {  Map<String,Object> map = **new** HashMap<String, Object>();  map.put(**"msg"**,**"获取用户信息成功"**);  map.put(**"status"**,200);  **return** map;  }  @RequestMapping(value = **"/getname"**)  **public** String getname()  {  **return myname**;  } |

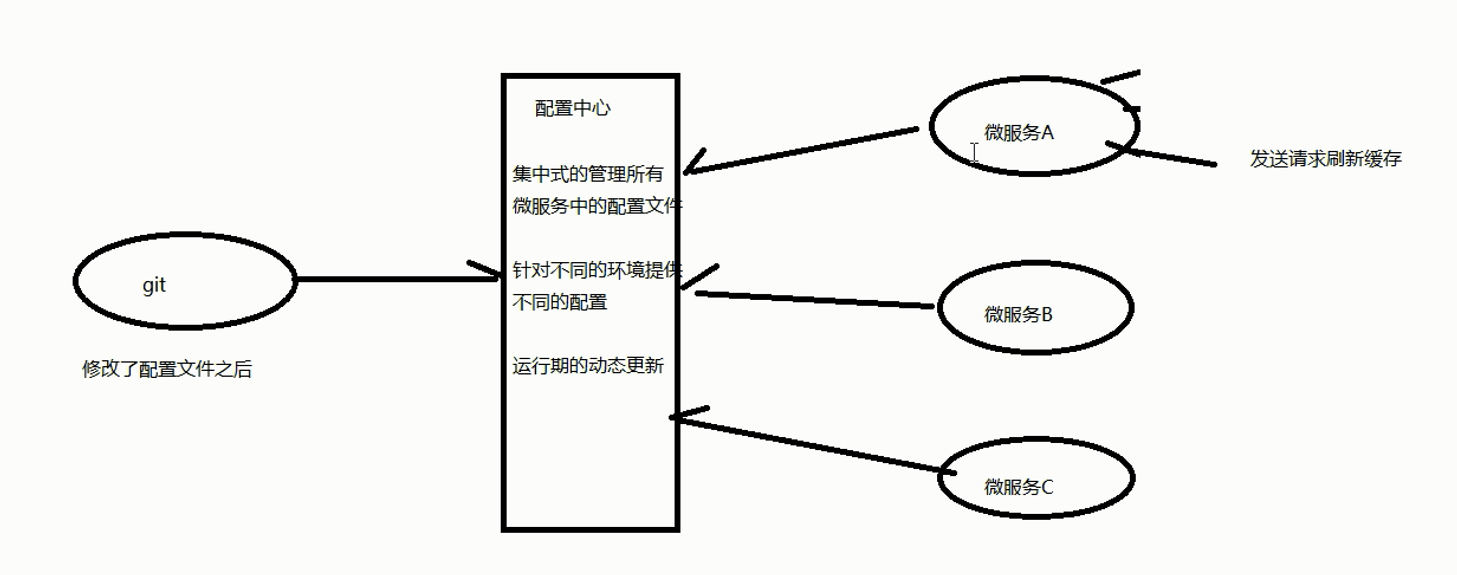
使用postMan测试手动触发一下 refresh 方法

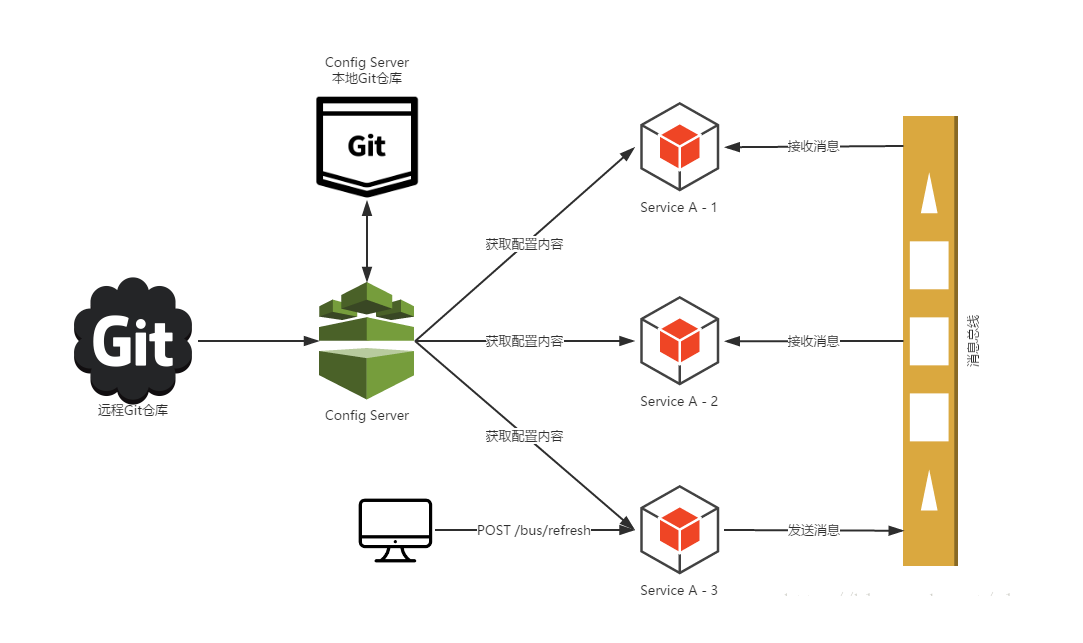
<http://localhost:9002/actuator/refresh>





# 7.springcloud config-bus





在微服务架构中，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来连接各个微服务实例，它广播的消息会被所有在注册中心的微服务实例监听和消费，也称消息总线。

SpringCloud中也有对应的解决方案，SpringCloud Bus 将分布式的节点用轻量的消息代理连接起来，可以很容易搭建消息总线，配合SpringCloud config 实现微服务应用配置信息的动态更新。

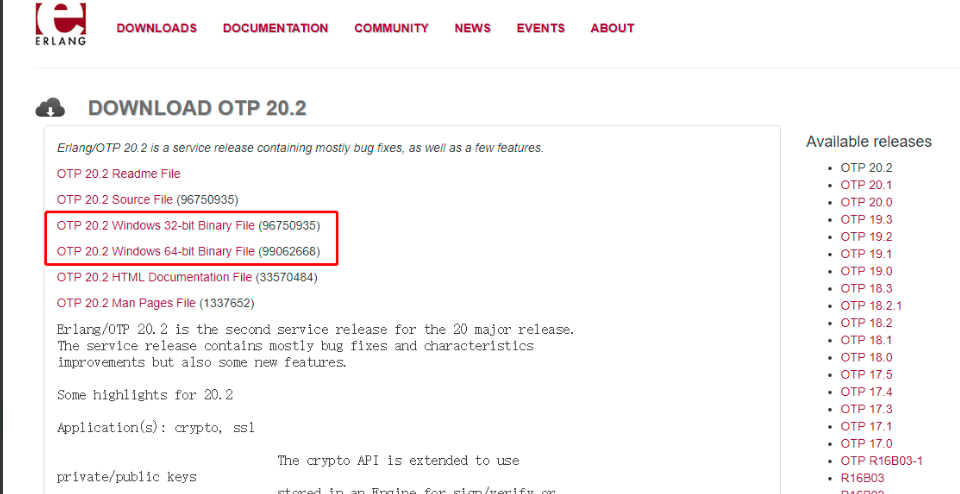
我们可以看出利用Spring Cloud Bus做配置更新的步骤:

* 提交代码触发post请求给bus/refresh
* server端接收到请求并发送给Spring Cloud Bus
* Spring Cloud bus接到消息并通知给其它客户端
* 其它客户端接收到通知，请求Server端获取最新配置
* 全部客户端均获取到最新的配置

## 7.0.安装rabbitmq

### 第一步：下载并安装erlang

* 原因：RabbitMQ服务端代码是使用并发式语言Erlang编写的，安装Rabbit MQ的前提是安装Erlang。
* 下载地址：<http://www.erlang.org/downloads>

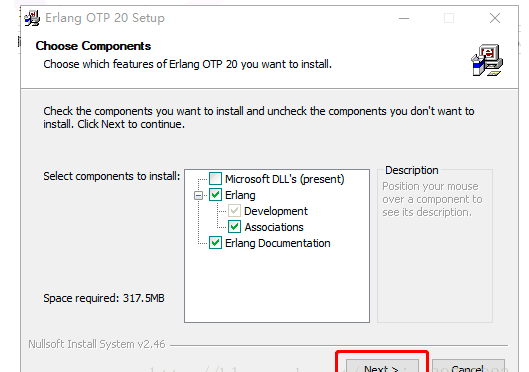
[](https://img2020.cnblogs.com/blog/1182199/202008/1182199-20200804160401265-1461179220.png)

根据本机位数选择erlang下载版本。

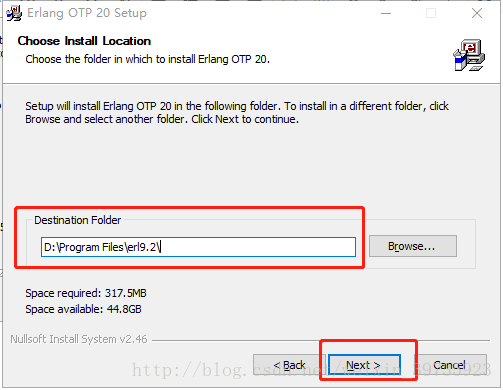
* 下载完是这么个东西：

[](https://img2020.cnblogs.com/blog/1182199/202008/1182199-20200804160422706-305607870.png)

* 双击，点next就可以。

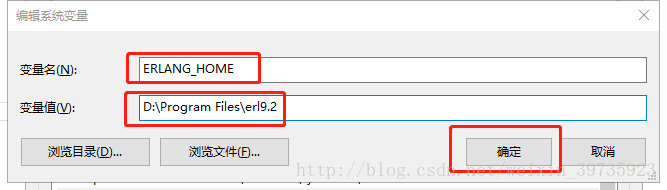
[](https://img2020.cnblogs.com/blog/1182199/202008/1182199-20200804160501158-1882154276.png)

* 选择一个自己想保存的地方，然后next、finish就可以。

[](https://img-blog.csdn.net/20180208133016752?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvd2VpeGluXzM5NzM1OTIz/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70)

* 安装完事儿后要记得配置一下系统的环境变量。

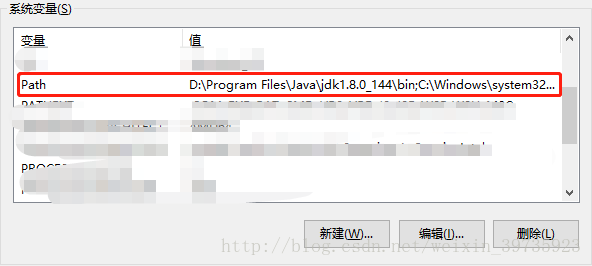
此电脑-->鼠标右键“属性”-->高级系统设置-->环境变量-->“新建”系统环境变量

[](https://img-blog.csdn.net/20180208133656531?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvd2VpeGluXzM5NzM1OTIz/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70)

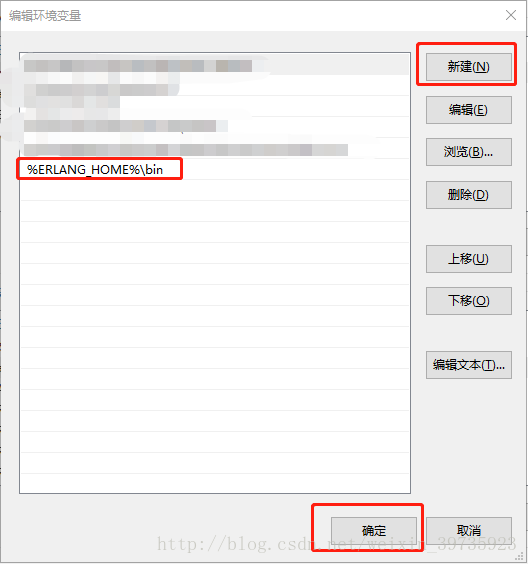
变量名：ERLANG\_HOME

变量值就是刚才erlang的安装地址，点击确定。

然后双击系统变量path

[](https://img-blog.csdn.net/20180208133941480?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvd2VpeGluXzM5NzM1OTIz/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70)

点击“新建”，将%ERLANG\_HOME%\bin加入到path中。

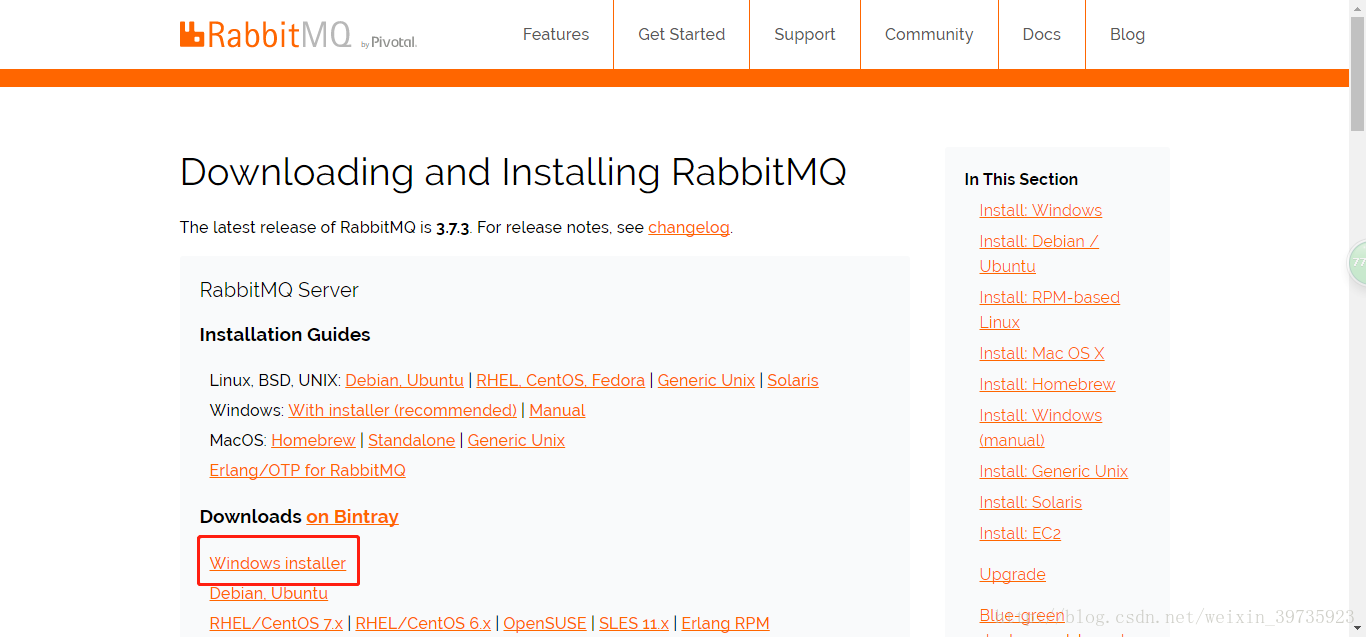
[](https://img-blog.csdn.net/20180208134038141?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvd2VpeGluXzM5NzM1OTIz/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70)

* 最后windows键+R键，输入cmd，再输入erl，看到版本号就说明erlang安装成功了。

[](https://img-blog.csdn.net/20180208134728511?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvd2VpeGluXzM5NzM1OTIz/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70)

### 第二步：下载并安装RabbitMQ

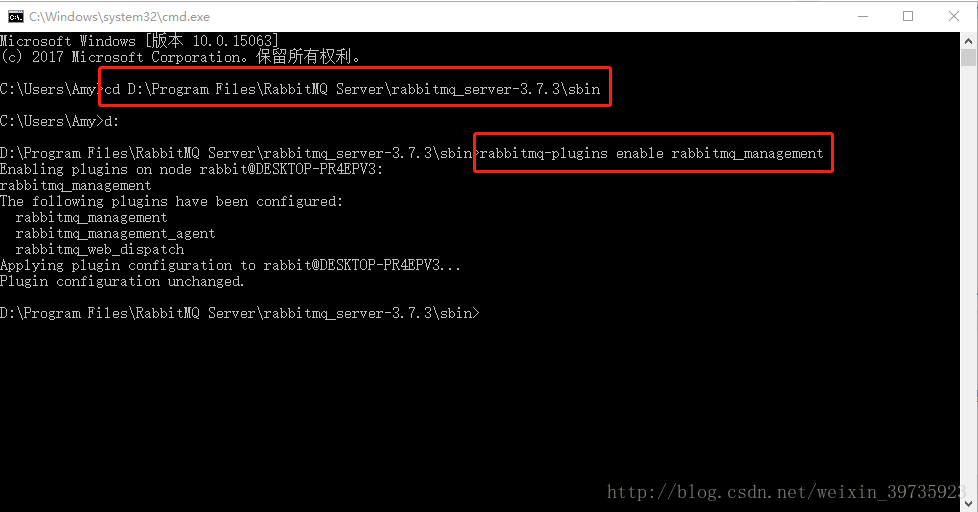
* 下载地址：<http://www.rabbitmq.com/download.html>

[](https://img-blog.csdn.net/20180208140346384?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvd2VpeGluXzM5NzM1OTIz/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70)

* 双击下载后的.exe文件，安装过程与erlang的安装过程相同。
* RabbitMQ安装好后接下来安装RabbitMQ-Plugins。打开命令行cd，输入RabbitMQ的sbin目录。

我的目录是：D:\Program Files\RabbitMQ Server\rabbitmq\_server-3.7.3\sbin

然后在后面输入rabbitmq-plugins enable rabbitmq\_management命令进行安装

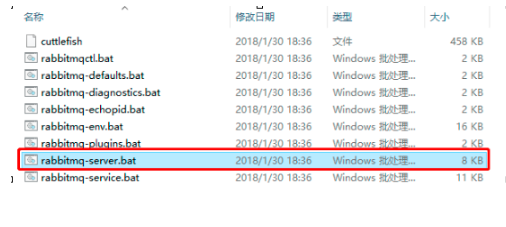
[](https://img-blog.csdn.net/20180208141846581?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvd2VpeGluXzM5NzM1OTIz/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70)

打开命令行命令行,进入RabbitMQ的安装目录: sbin

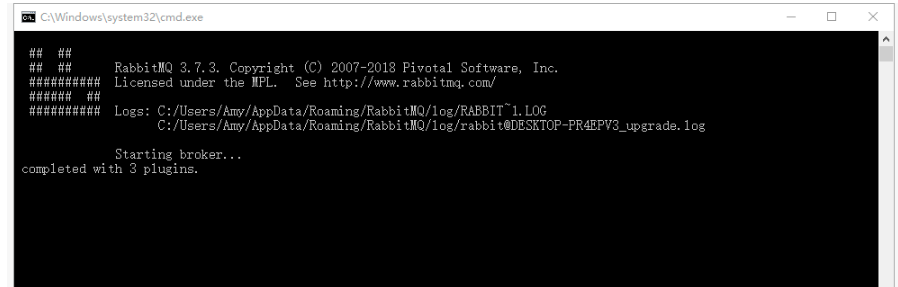
,输入 rabbitmqctl status , 如果出现以下的图，说明安装是成功的，并且说明现在RabbitMQ Server已经启动了,运行正常。

[](https://imgsa.baidu.com/exp/w=500/sign=f02f000de2dde711e7d243f697efcef4/b03533fa828ba61e8d9e78274634970a304e5941.jpg)

打开sbin目录，双击rabbitmq-server.bat

[](https://img2020.cnblogs.com/blog/1182199/202008/1182199-20200804160709298-542008519.png)

等几秒钟看到这个界面后，访问http://localhost:15672

[](https://img2020.cnblogs.com/blog/1182199/202008/1182199-20200804160733421-1654901382.png)

然后可以看到如下界面

[](https://img2020.cnblogs.com/blog/1182199/202008/1182199-20200804160234264-1848768815.png)

默认用户名和密码都是guest

登陆即可。

常见问题:

最近在Windows server 2012 R2的百度云服务器上安装rabbitmq时，在执行rabbitmqctl status命令时，一直报错，开始时出现的如下错误：

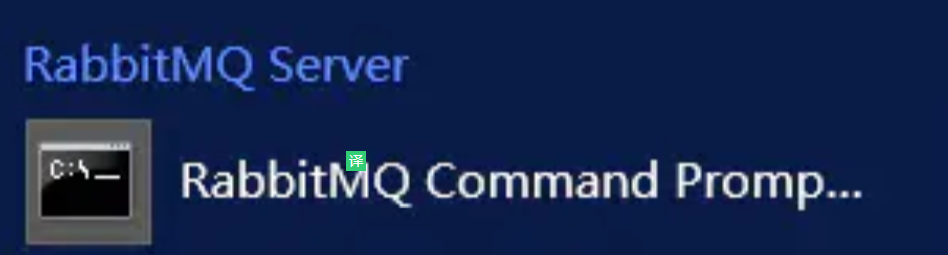
Error: unable to connect to node rabbit@xxx: nodedown

### erlang的cookie问题排解

Google、百度了一通后，大多数都说的是cookie的问题，需要保持C:\Windows下的.erlang.cookie和C:\Users\Administrator下的.erlang.cookie一致，然后我搜索了一下，总共在三个地方找到了这个文件，如下图：

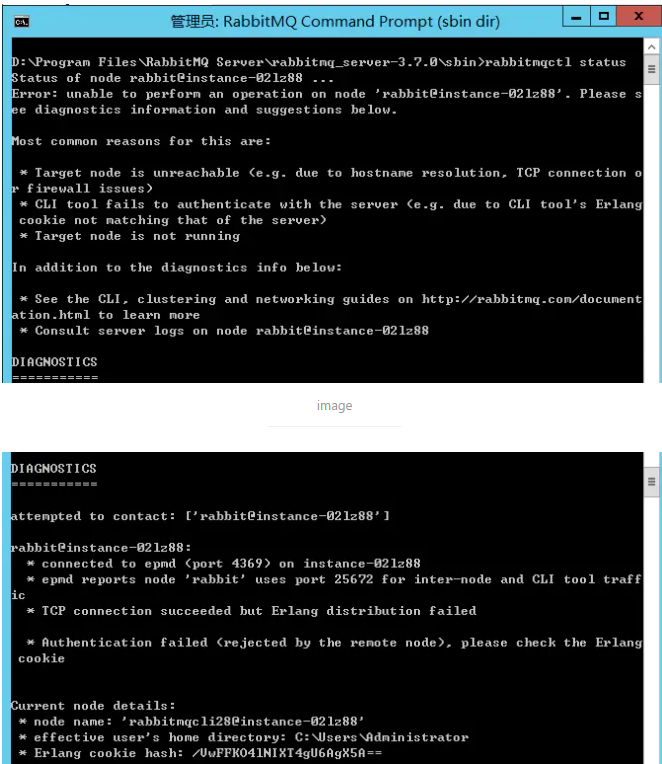


然后，我用其中的一个文件中的内容完全替换掉其他两个，这样使三个文件保持一致，然后打开下面这个窗口：



### 新的错误： unable to perform an operation on node

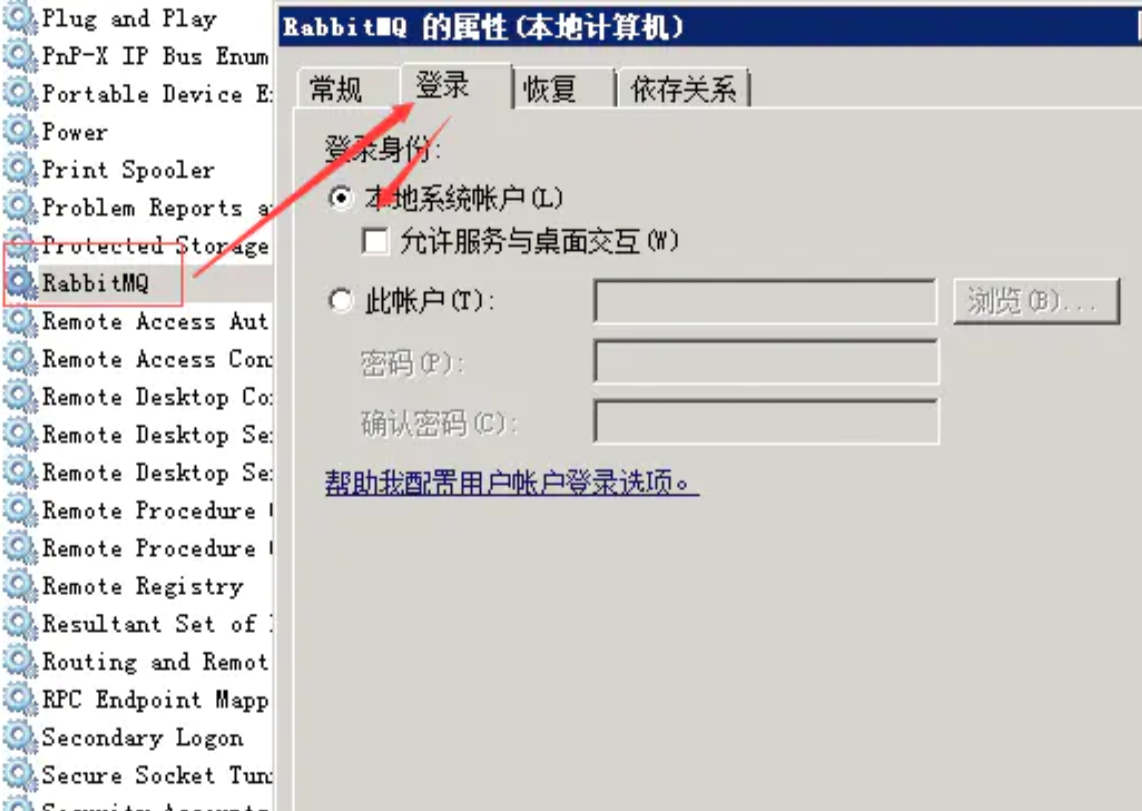
再次执行rabbitmqctl status命令，此时报了另一个错误：



不解，cookie也设置相同了，环境变量也添加正确了，为什么还是报错，继续寻找问题。

### 登录账号权限问题

后来看到一个网友说可能是登录账号的问题，我是用administrator账号登录服务器并安装rabbitmq的，rabbitmq也自动在C：\user\administrator下创建了一个.erlang.cookie文件，但是RabbitMQ服务启动的时候，执行者是local system account，如下图所示：



此时，我想到将RabbitMQ服务的启动账号改成administrator，果然，更改后重启服务，再次执行rabbitmqctl status，成功！具体操作如下图所示：



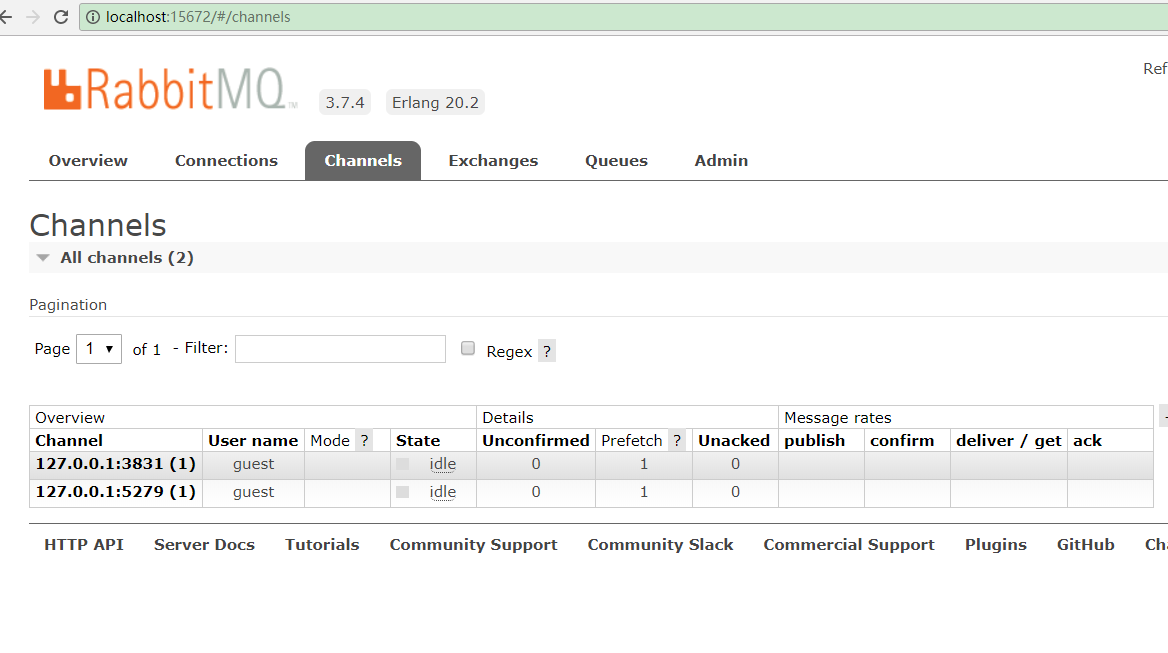


测试访问:

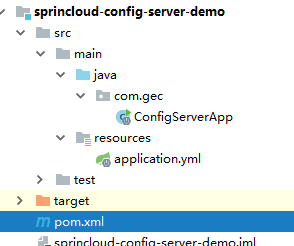
<http://localhost:15672>

默认用户名和密码都是guest

登陆即可。



## 7.1.修改configserver

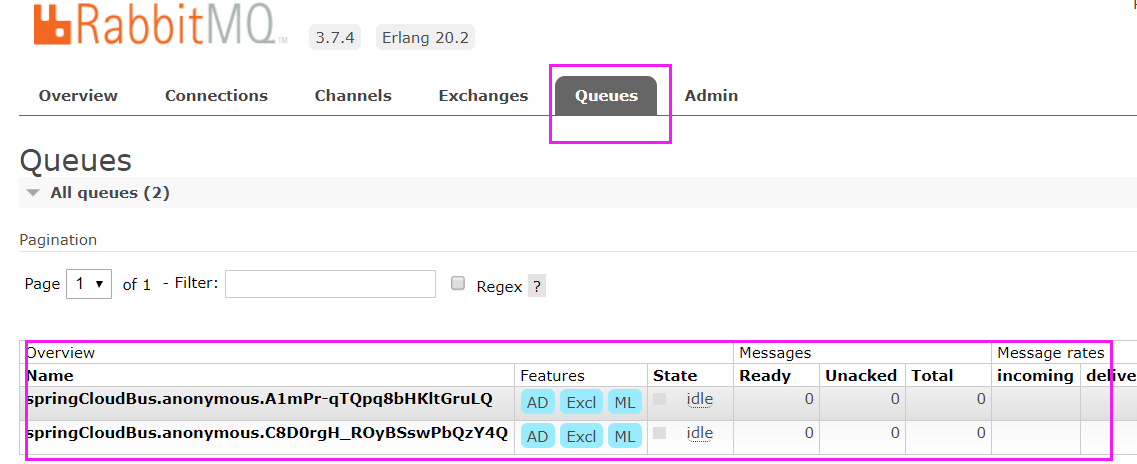


导入依赖

|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-config-server</**artifactId**> </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-config</**artifactId**> </**dependency**> <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-actuator</**artifactId**> </**dependency**> <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-bus</**artifactId**> </**dependency**> <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-stream-binder-rabbit</**artifactId**> </**dependency**> |

## 7.2.配置文件处理

|  |
| --- |
| **server:  port:** 9900 *#服务端口* **spring:  application:  name:** config-server *#指定服务名* **cloud:  config:  server:  git:  uri:** https://gitee.com/wuhancoder\_ios/config-respostry-01.git  **rabbitmq:  host:** 127.0.0.1  **port:** 5672  **username:** guest  **password:** guest **management:  endpoints:  web:  exposure:  include:** bus-refresh |



## 7.3.修改客户端 gec-product-service

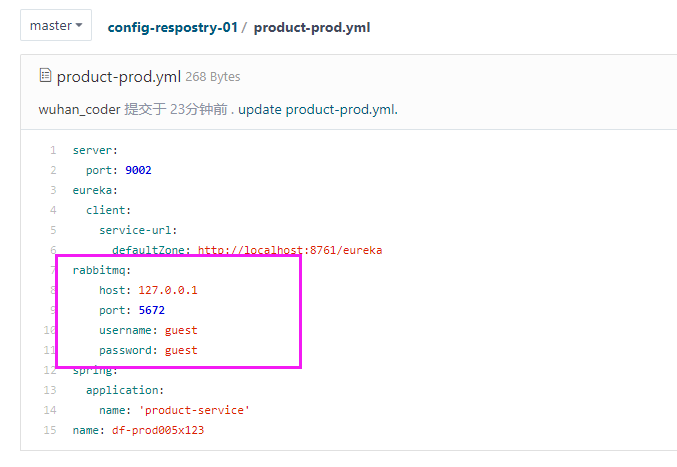
导入依赖

|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-config</**artifactId**> </**dependency**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-actuator</**artifactId**> </**dependency**>   <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-bus</**artifactId**> </**dependency**> <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-stream-binder-rabbit</**artifactId**> </**dependency**> |

修改配置文件

|  |
| --- |
| **spring:  cloud:  config:  name:** product *#应用名称,需要对应git中配置文件名称的前半部分* **profile:** prod *#开发环境* **label:** master *#git中的分支* **uri:** http://localhost:9900 *#config-server的请求地址* |

将rabbitmq的配置信息 添加到gitee中几个配置文件



…..

7.4.使用postMan给 bus 提交一下

<http://localhost:9900/actuator/bus-refresh>

再去访问

<http://localhost:9002/getname> 试试看看结果