springCloud精简使用版

目录

[1 实例演示 1](#_Toc15037644)

[2 实例源码和部署地址 2](#_Toc15037645)

[3 预期目标： 2](#_Toc15037646)

[4 开发注册发现服务 2](#_Toc15037647)

[4.1 新建工程 2](#_Toc15037648)

[4.2 编写 pom 3](#_Toc15037649)

[4.3 编写properties文件 3](#_Toc15037650)

[4.4 部署项目 3](#_Toc15037651)

[4.4.1 创建容器 3](#_Toc15037652)

[4.4.2 生成jar 4](#_Toc15037653)

[4.4.3 发布 4](#_Toc15037654)

[5 开发统一配置中心服务 4](#_Toc15037655)

[5.1 搭建gitlab服务 4](#_Toc15037656)

[5.1.1 下载docker镜像 4](#_Toc15037657)

[5.1.2 建立目录 4](#_Toc15037658)

[5.1.3 建立容器 4](#_Toc15037659)

[5.1.4 访问 4](#_Toc15037660)

[5.2 开发步骤同 “开发注册发现服务” 5](#_Toc15037661)

[6 开发生产者和消费者服务 5](#_Toc15037662)

[6.1 开发生产者（producers）服务 5](#_Toc15037663)

[6.1.1 pom 5](#_Toc15037664)

[6.1.2 properties文件说明 6](#_Toc15037665)

[6.1.3 注解 6](#_Toc15037666)

[6.1.4 写一个api接口实例 6](#_Toc15037667)

[6.2 开发消费者（consumers）服务 7](#_Toc15037668)

[6.2.1 和生产者有区别的地方如下： 7](#_Toc15037669)

[6.3 关于生产者和消费者两种角色 8](#_Toc15037670)

[7 将api接口用nginx做反向代理 8](#_Toc15037671)

[8 测试 8](#_Toc15037672)

[9 综述：实现的架构结果 9](#_Toc15037673)

# 实例演示

在外网通过nginx反向代理，访问内网的某个api业务系统服务，此服务调用另外一个业务系统服务返回结果。

<http://172.16.31.43/wsbq-api/user/demo?id=111&name=%E5%BC%A0%E4%B8%89&currentpage=1&pagesize=1>

# 实例源码和部署地址

<http://172.16.14.101/>

账号 ： liuixng

密码 ： 6789@jkl

部署地址： 172.16.14.241 docker

# 预期目标：

1.所有业务系统在eureka统一管控

2.所有业务系统的配置在gitlab上管理维护

3.所有业务系统之间的互相调用精简成如同调用本系统一个方法

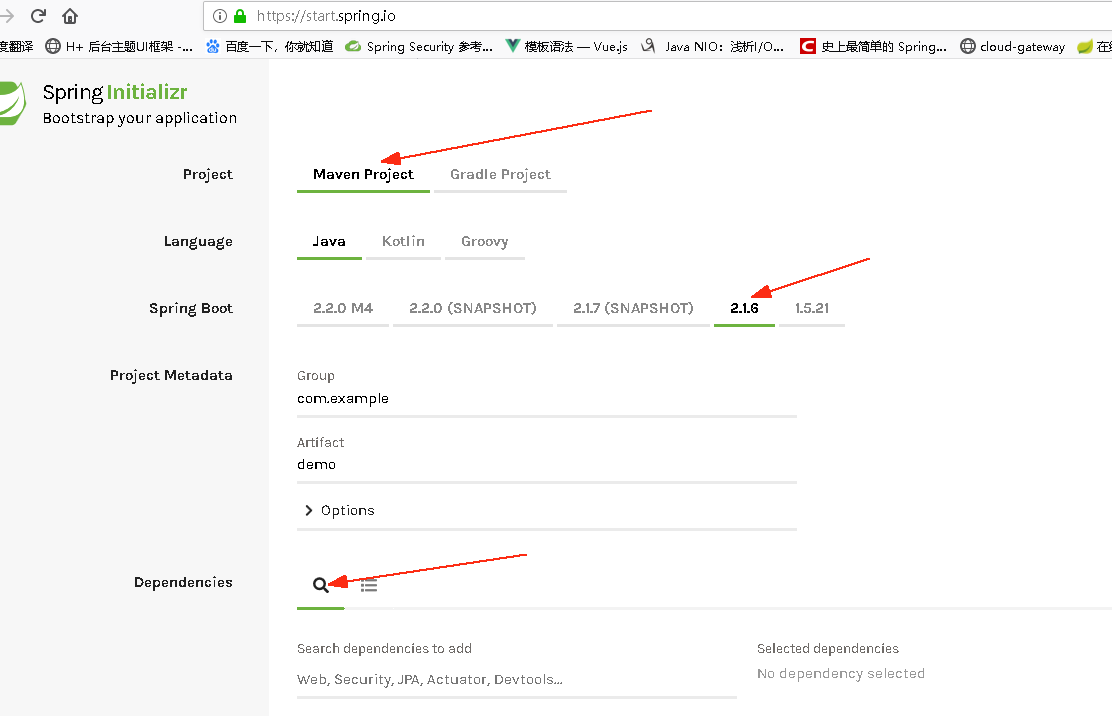
4.业务系统弹性部署

# 开发注册发现服务

实例：<http://172.16.14.101/liuxing/sc-eureka-server>

## 新建工程

新建一个普通的springboot工程，<https://start.spring.io/>



## 编写 pom

参考实例demo

## 编写properties文件

参考实例demo

## 部署项目

### 创建容器

通常一个项目一个container，有了后就不必再新建了。

docker run --name wsbq-api -dit --net=mcv --privileged=true --ip=172.16.14.244 -v /opt/wsbqapi:/usr/liuxing:ro -v /var/log/applog:/var/log/applog:rw baoquanjavabase

只需要修改 --name 你起的项目名

--ip 你指定个ip地址

-v 你把jar包放到的目录

第二个 –v 将容器中的log日志映射到宿主机上，以便查看

### 生成jar

dos进入项目目录下，执行: mvn clean package –DskipTests

将生成的jar包放到服务器里边，重命名为 app.jar

### 发布

docker stop sc-eureka-server

docker start sc-eureka-server

# 开发统一配置中心服务

实例：http://172.16.14.101/liuxing/sc-configserver

## 搭建gitlab服务

### 下载docker镜像

docker pull beginor/gitlab-ce:11.0.1-ce.0

### 建立目录

通常会将 GitLab 的配置 (etc) 、 日志 (log) 、数据 (data) 放到容器之外， 便于日后升级， 因此请先准备这三个目录

mkdir -p /mnt/gitlab/etc

mkdir -p /mnt/gitlab/log

mkdir -p /mnt/gitlab/data

### 建立容器

docker run --name gitlab -dit --net=mcv --privileged=true --ip=172.16.14.101 -v /mnt/gitlab/etc:/etc/gitlab -v /mnt/gitlab/log:/var/log/gitlab -v /mnt/gitlab/data:/var/opt/gitlab beginor/gitlab-ce:11.0.1-ce.0

### 访问

<http://172.16.14.101/>

## 开发步骤同 “开发注册发现服务”

# 开发生产者和消费者服务

## 开发生产者（producers）服务

实例：http://172.16.14.101/liuxing/wsbq-common

项目搭建参考 “开发注册发现服务”，有特点的地方描述如下：

### pom



第一个依赖：注册发现服务的客户端包

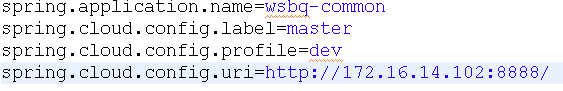
第二个依赖：web服务支持包

第三个依赖：调用链监控包

第四个依赖：测试包

第五个依赖：统一配置中心服务的客户端包

### properties文件说明



第一个：项目名称，其它服务就是通过这个名字来调用此服务的。

第二个：分支名称，master表示可用稳定版

develop 表示持续开发的分支

release 表示发布过程中的分支

第三个：表示项目环境， dev 表示开发环境

test 表示测试环境

prod表示生产环境

第四个：统一配置服务的地址。本例中是ip地址，可以是域名，然后通过nginx代理两个ip做负载均衡来保证其高可用。

### 注解

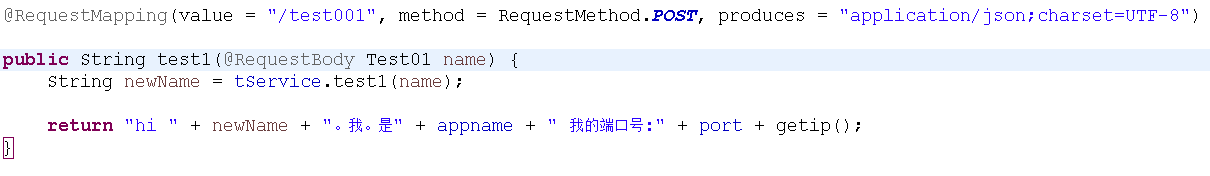
启动类上追加注解 @EnableEurekaClient，这样就实现了将此服务注册到eureka。其它的项目就可以调用这个服务了。

### 写一个api接口实例

我们姑且把这个接口实例命名业务名为：“推送业务员信息”

就是普普通通一个从 controller –service-dao的一套代码

注意的地方是：此例中请求参数可以为对象方式。也就是说生产者和消费者之间传递数据可以直接写成对象的方式。就不用json和反json 解析了。而且这个对象内属性内容可以是不同的。

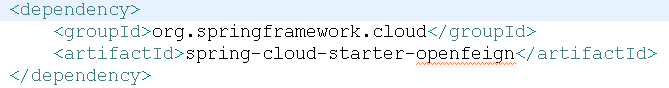


## 开发消费者（consumers）服务

实例：<http://172.16.14.101/liuxing/wsbq-api>

### 和生产者有区别的地方如下：

#### pom



#### properties

feign.hystrix.enabled=true 开启断路器

feign负载均衡策略及参数配置，参考实例的配置文件

#### 调用生产者

调用生产者的 “推送业务员信息” 接口服务

##### 写interface

实例：com.thunisoft.wsbq.outeriface.CommonApi



第一个箭头：要调用的项目的properties中的项目名称spring.application.name

第二个箭头：熔断器异常处理声明

第三个箭头：生产者中接口服务的地址

第四个箭头：调用过程中可以使用javabean传值，就像是调用自己工程的方法一样。

##### 写熔断器异常捕获类

实例：com.thunisoft.wsbq.outeriface.schedual.SchedualServiceHiHystric

## 关于生产者和消费者两种角色

通常一个project中会同时扮演生产者和消费者两种角色。本文强化这两种角色概念是为了给大家一个调用关系的感知。

# 将api接口用nginx做反向代理

实例：172.16.31.43 nginx配置

# 测试

<http://172.16.31.43/wsbq-api/user/demo?id=111&name=%E5%BC%A0%E4%B8%89&currentpage=1&pagesize=1>

# 综述：实现的架构结果

