# 微服务入门

搭建eureka过程：

创建一个maven父项目，pom内容如下

<?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>org.example</groupId>  
 <artifactId>eureka-demo</artifactId>  
 <packaging>pom</packaging>  
 <version>1.0-SNAPSHOT</version>  
 <modules>  
 <module>eureka-server</module>  
 </modules>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.2.4.RELEASE</version>  
 </parent>  
  
 <properties>  
 <spring.cloud.version>Hoxton.SR1</spring.cloud.version>  
 </properties>  
<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>  
</project>

创建一个子项目，pom内容如下：

<?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">  
 <parent>  
 <artifactId>eureka-demo</artifactId>  
 <groupId>org.example</groupId>  
 <version>1.0-SNAPSHOT</version>  
 </parent>  
 <modelVersion>4.0.0</modelVersion>  
 <artifactId>eureka-server</artifactId>  
 <properties>  
 </properties>  
 <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>  
 </dependencies>  
</project>

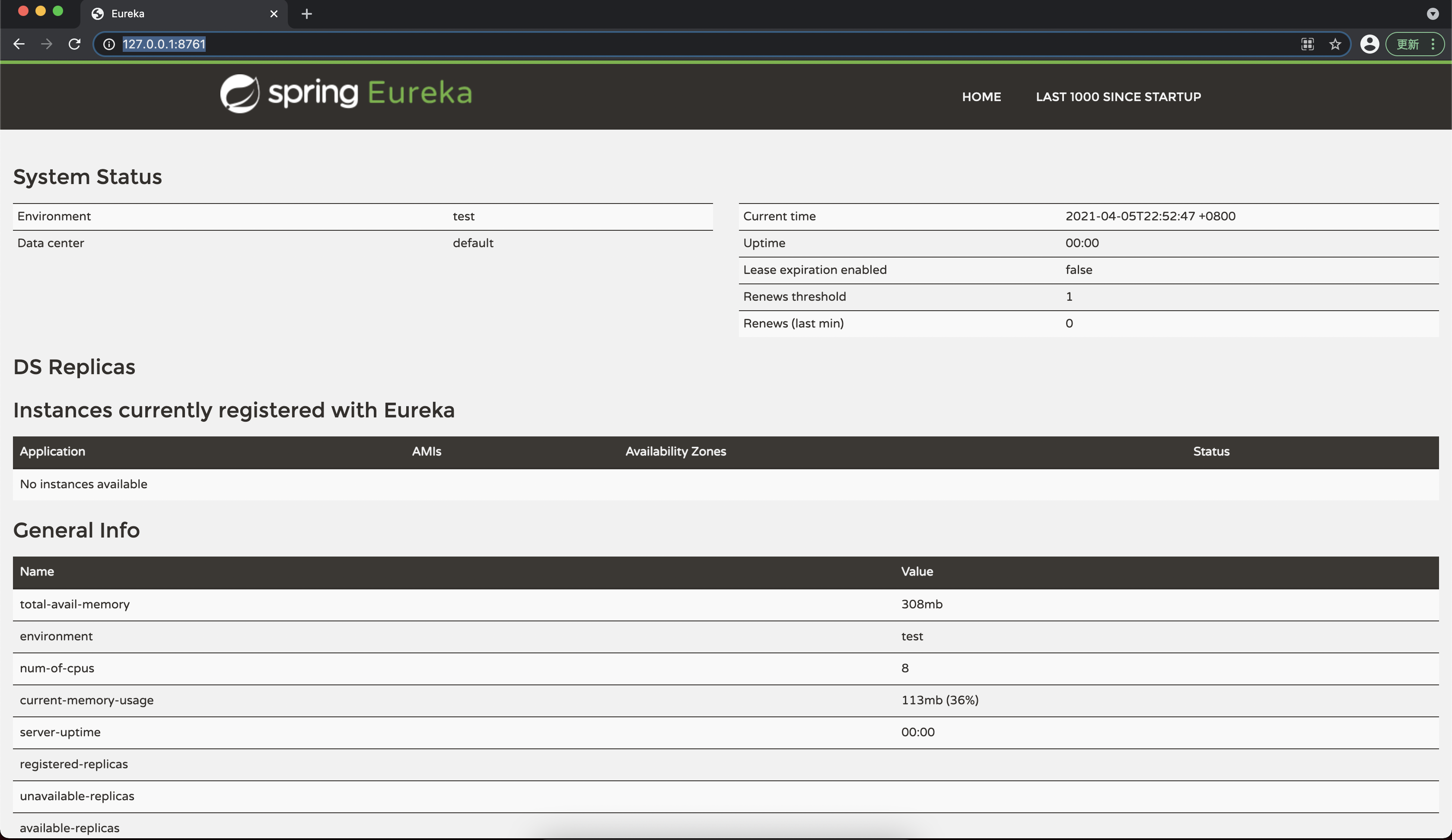
子项目配置文件application.yml内容如下：

server:  
 port: 8761  
spring:  
 application:  
 name: eureka-server  
eureka:  
 instance:  
 hostname: localhost *#主机名* client:  
 register-with-eureka: false *#是否将自己注册到注册中心* fetch-registry: false *#是否在注册中心获取服务信息* service-url: *#注册中心对外暴露的服务地址* defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/

springboot启动类文件如下：

package com;  
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);  
 }  
}

启动后访问 <http://127.0.0.1:8761/> 内容如下：



搭建eureka集群过程：

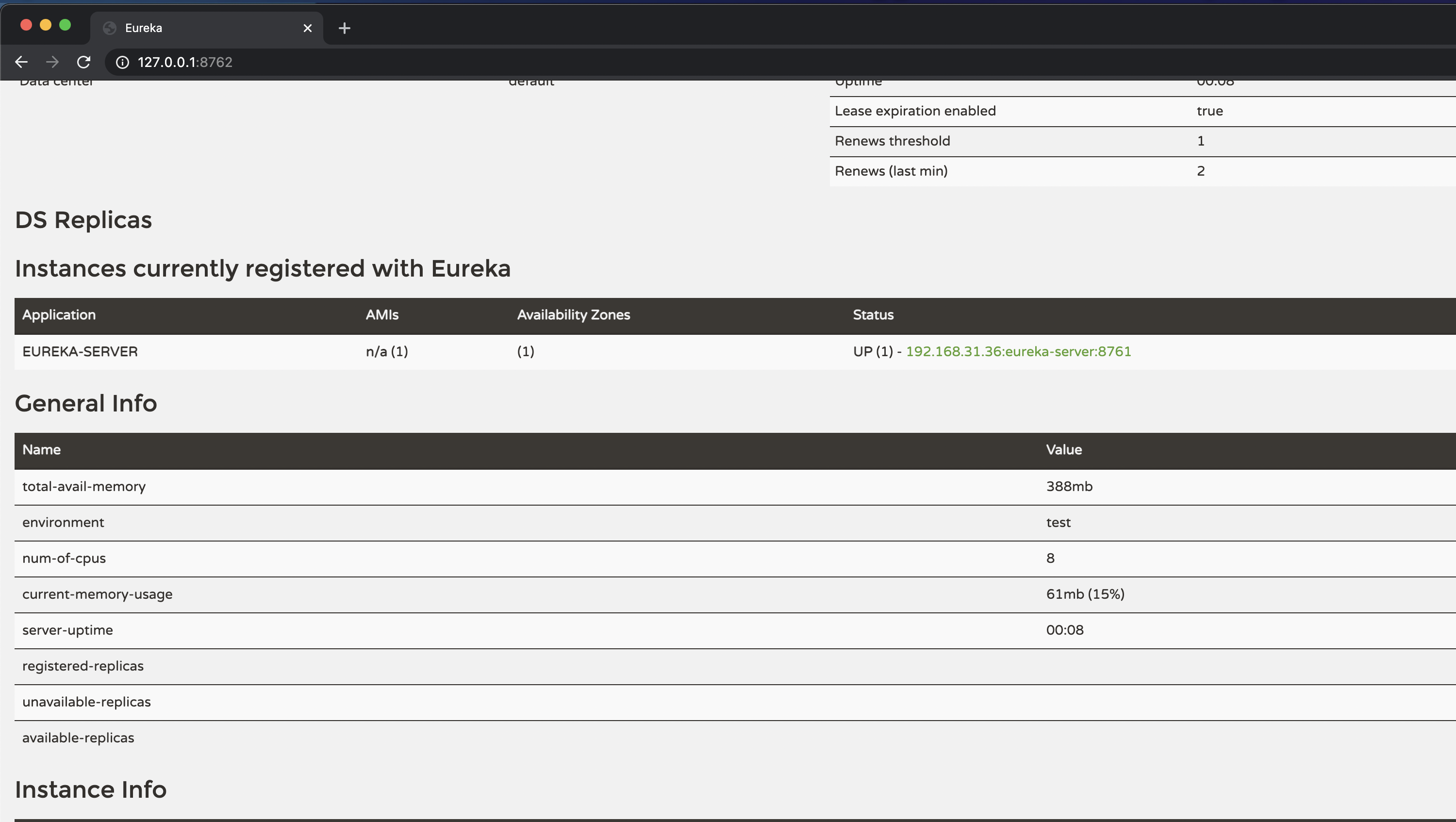
过程同单机搭建过程，修改application.yml配置文件

两个服务的内容如下：去掉了对自身注册的配置，服务地址改为另个一服务的ip+port

server:  
 port: 8762  
spring:  
 application:  
 name: eureka-server  
eureka:  
 instance:  
 hostname: localhost *#主机名* client:  
 service-url: *#注册中心对外暴露的服务地址* defaultZone: http://${eureka.instance.hostname}:8761/eureka/

server:  
 port: 8761  
spring:  
 application:  
 name: eureka-server  
eureka:  
 instance:  
 hostname: localhost *#主机名* client:  
 service-url: *#注册中心对外暴露的服务地址* defaultZone: http://${eureka.instance.hostname}:8762/eureka/

两个服务启动完成后，8761和8762端口都可以访问



管理页中，服务名可以通过服务启动配置文件的配置设置：

eureka:

instance:

hostname: localhost #主机名

prefer-ip-address: true

instance-id: ${spring.cloud.client.ip-address}:${server.port}

eureka集群搭好后，新建一个provider生产者：

普通的springboot项目，pom内容如下：

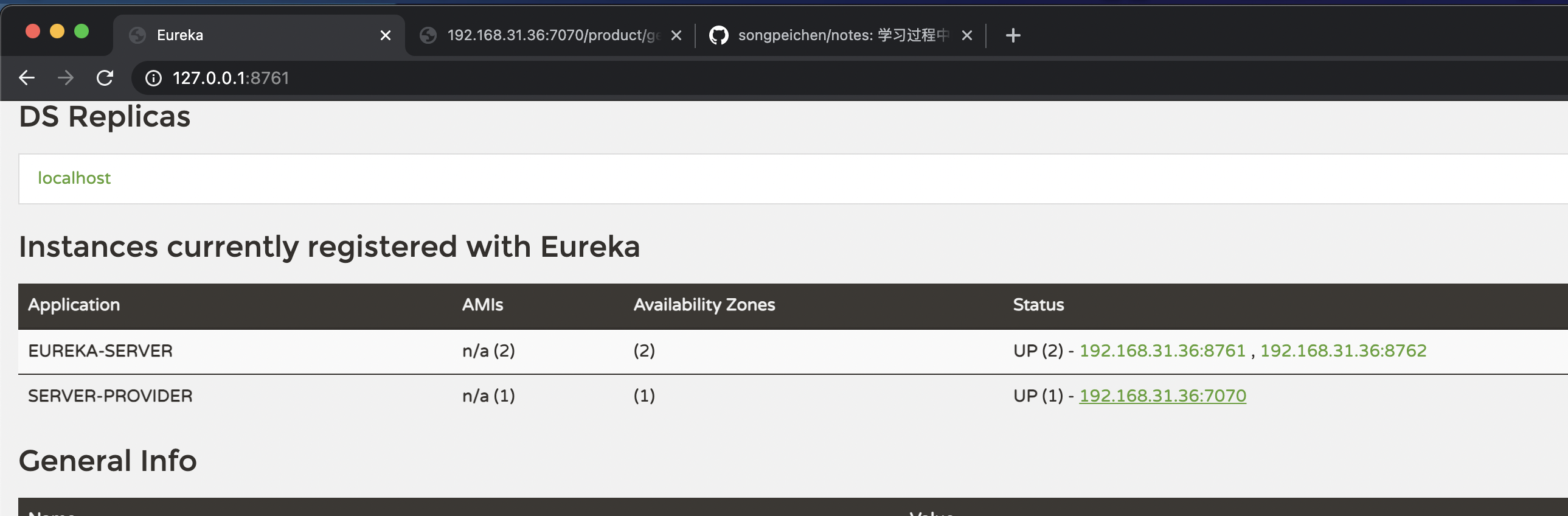
<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <parent>  
 <artifactId>eureka-demo</artifactId>  
 <groupId>org.example</groupId>  
 <version>1.0-SNAPSHOT</version>  
 </parent>  
 <modelVersion>4.0.0</modelVersion>  
  
 <artifactId>server-provider</artifactId>  
  
 <properties>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 </dependency>  
 </dependencies>  
</project>

配置文件application.yml内容:

server:  
 port: 7070  
spring:  
 application:  
 name: server-provider  
eureka:  
 instance:  
 hostname: localhost *#主机名* prefer-ip-address: true *#是否使用IP地址注册* instance-id: ${spring.cloud.client.ip-address}:${server.port}  
 client:  
 service-url: *#注册中心对外暴露的服务地址* defaultZone: http://${eureka.instance.hostname}:8761/eureka/,http://${eureka.instance.hostname}:8762/eureka/

创建controller、service文件和注解、pom文件中写了客户端并且配置文件写了注册中心，springboot会认为这是一个客户端服务，启动类上的@EnableEurekaClient注解可以省略

启动后在eureka管理页面可看到provider服务，访问URL也可以得到provider服务的返回值



搭建consumer服务：

内容基本和provider相同，区别在于application.yml配置文件没有注册到eureka,直接在eureka里面拉取相应的服务，服务调用也改成远程调用，代码如下：

server:  
 port: 9090  
spring:  
 application:  
 name: server-consumer  
eureka:  
 client:  
 service-url: *#注册中心对外暴露的服务地址* defaultZone: http://localhost:8761/eureka/,http://localhost:8762/eureka/  
 register-with-eureka: false *#是否注册到eureka* registry-fetch-interval-seconds: 10 *#隔多久去eureka里拉取*

package com.service.impl;  
import com.pojo.Order;  
import com.pojo.Product;  
import com.service.OrderService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.cloud.client.ServiceInstance;  
import org.springframework.cloud.client.discovery.DiscoveryClient;  
import org.springframework.core.ParameterizedTypeReference;  
import org.springframework.http.HttpMethod;  
import org.springframework.http.ResponseEntity;  
import org.springframework.stereotype.Service;  
import org.springframework.web.client.RestTemplate;  
import java.util.List;  
@Service  
public class OrderServiceImpl implements OrderService {  
 @Autowired  
 private RestTemplate restTemplate;  
 @Autowired  
 private DiscoveryClient discoveryClient;  
 @Override  
 public Order selectOrderById(Long id) {  
 List<Product> productList = selectProductListByDiscoveryClient();  
 Order order = new Order(id, "orderNo111", "订单地址", 204.3, productList);  
 return order;  
 }  
 private List<Product> selectProductListByDiscoveryClient() {  
 List<String> services = discoveryClient.getServices();  
 List<ServiceInstance> instances = discoveryClient.getInstances("server-provider");  
 ServiceInstance s1 = instances.get(0);  
 ResponseEntity<List<Product>> response = restTemplate.exchange("http://" + s1.getHost() + ":" + s1.getPort() + "/product/getProductList", HttpMethod.*GET*, null, new ParameterizedTypeReference<List<Product>>() {  
 });  
 return response.getBody();  
 }  
}

优雅停服：

pom文件中添加依赖：

<dependency>

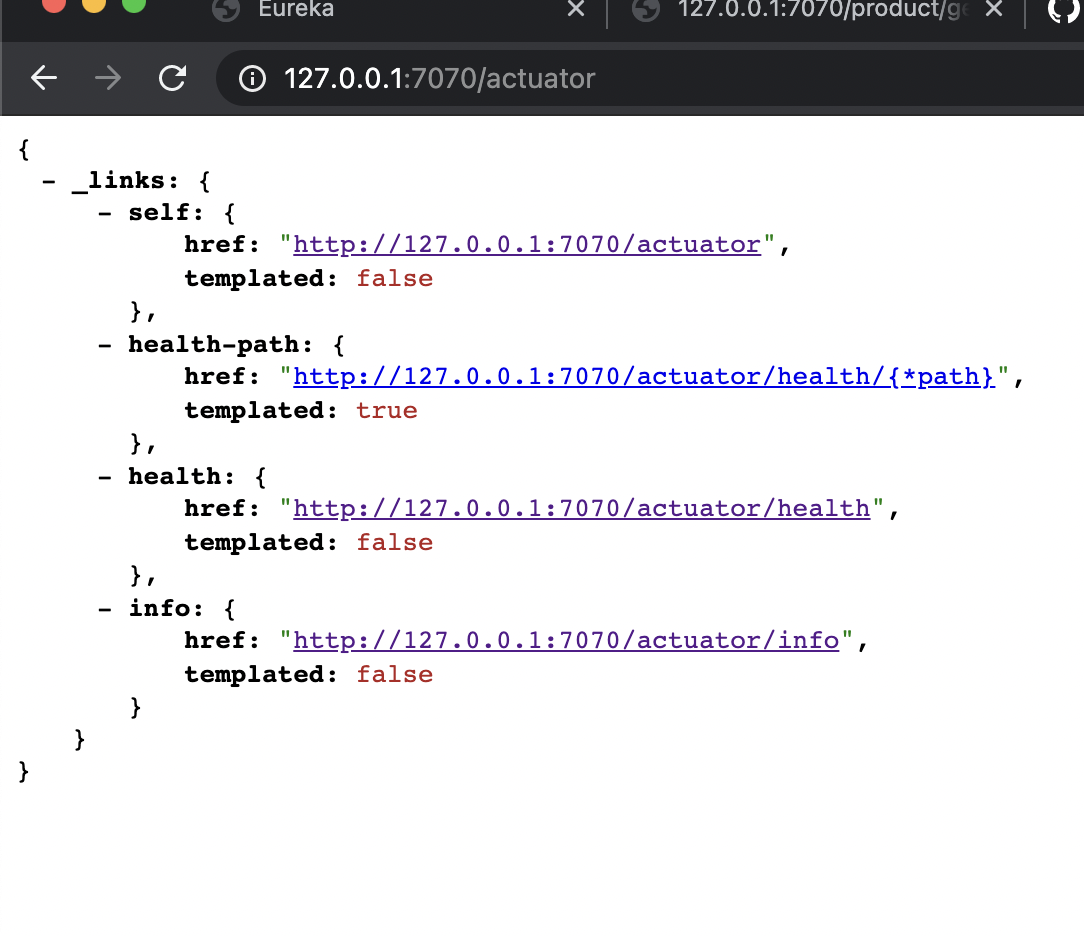
<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

在浏览器中访问

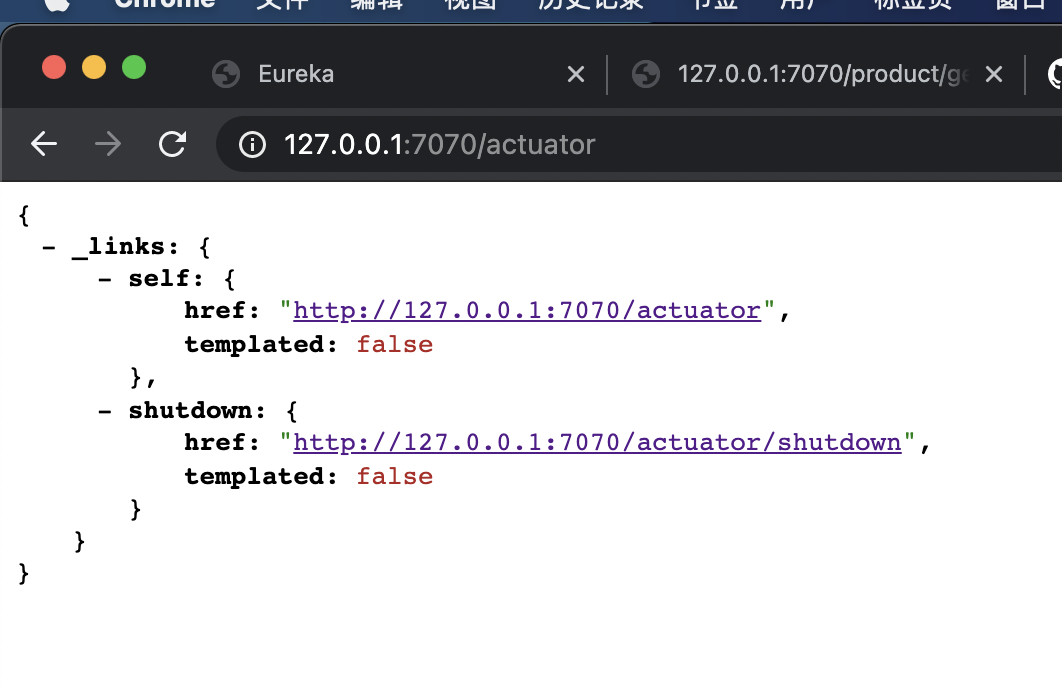
<http://127.0.0.1:7070/actuator> 即可看到服务状态 默认都没有开启,服务列表如下

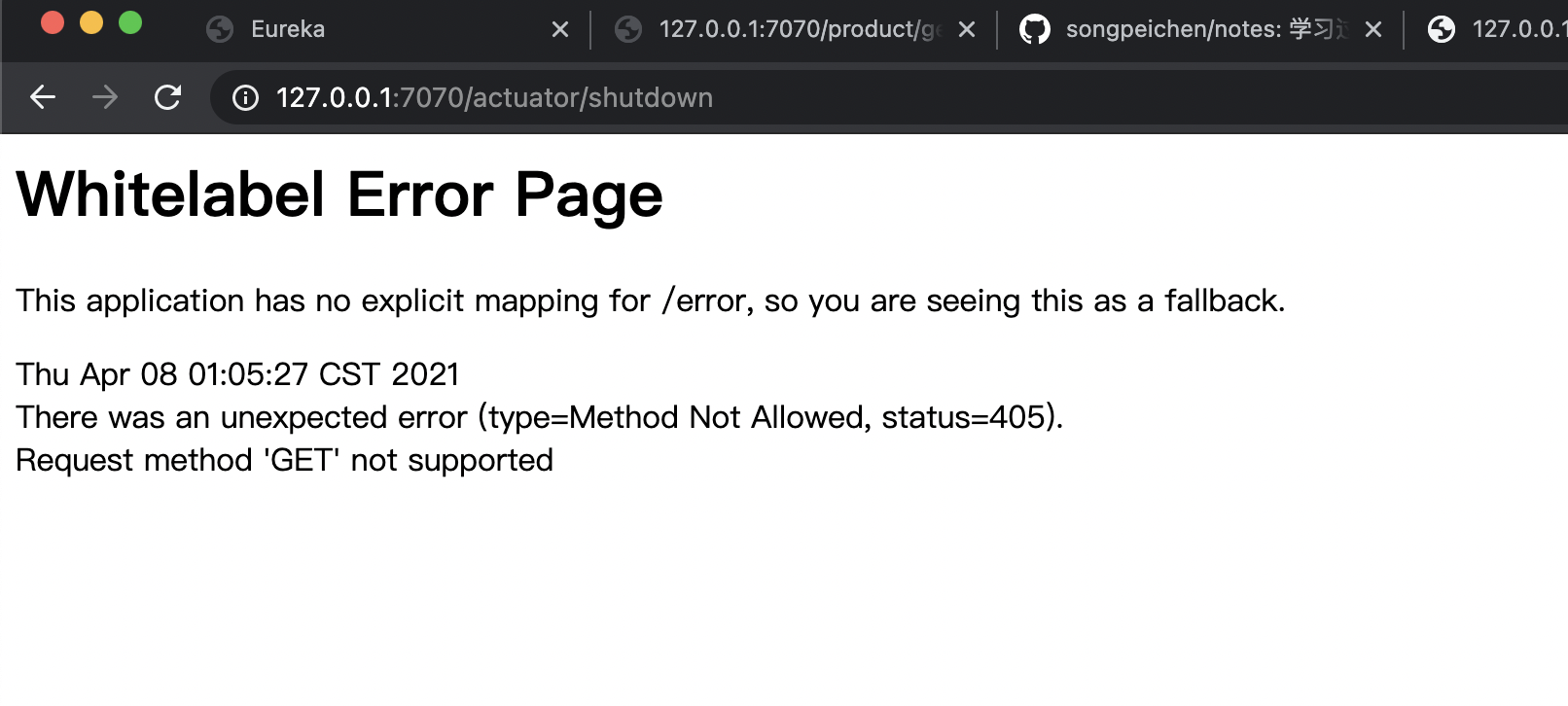


在配置文件中可指定开启端点的服务：

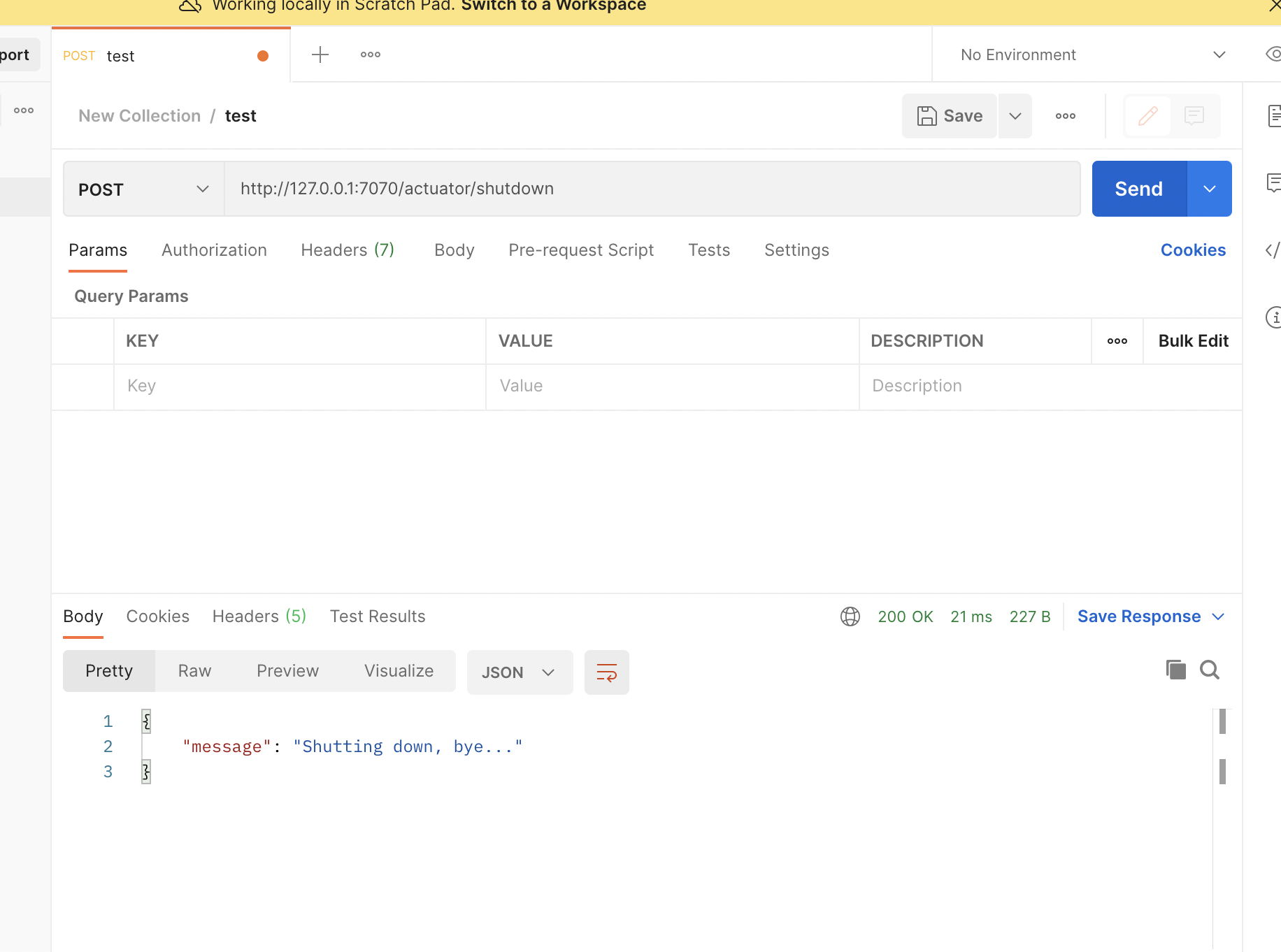
management:  
 endpoints:  
 web:  
 exposure:  
 include: shutdown *#开启的端点， 也可以写星号 "\*"* endpoint:  
 shutdown:  
 enabled: true

开启停服服务，在<http://127.0.0.1:7070/actuator> 可以看到





用postman即可发送post请求停止服务



eureka安全认证：

安全认证用的security

pom文件添加依赖：

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

配置文件添加配置：

spring:   
 security:  
 user:  
 name: root  
 password: 123456

配置文件修改注册中心地址：

defaultZone: http://root:123456@${eureka.instance.hostname}:${server.port}/eureka/

修改配置之后，启动两个eureka节点，启动之后查看注册信息需要用户名密码验证，验证后依然没有节点信息，是因为eureka开启了CSRF防御机制，认为post、put、delete请求都是不安全的，需要携带token，如果没有token 会被拦截并返回403

首先注册中心配置一个@EnableWebSecurity配置类，继承 org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter然后重写configure方法

方法1：忽略 /eureka/\*\*

@Override  
protected void configure(HttpSecurity http) throws Exception {  
 super.configure(http);  
 http.csrf().ignoringAntMatchers("/eureka/\*\*");  
}

方法2：关掉防疫机制

@Override  
protected void configure(HttpSecurity http) throws Exception {  
 http.csrf().disable().authorizeRequests().anyRequest().authenticated().and().httpBasic();  
}