**系统应用服务平滑升级**

**—技术解决方案V1.1**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **更新内容** | **操作人** | **更新时间** |
| V1.0 | 创建 | 林建峰 | 2021.04.07 |
| V1.1 | 增加现状与实施内容 | 林建峰 | 2021.04.13 |

架构设计部：林建峰

2021年4月13日

目录

[1. 需求描述 3](#_Toc69224351)

[2. 目标要求 3](#_Toc69224352)

[3. 项目现状 3](#_Toc69224353)

[3.1 运维环境 3](#_Toc69224354)

[3.2 项目结构 4](#_Toc69224355)

[3.3 操作流程 7](#_Toc69224356)

[3.3.1 检查升级 7](#_Toc69224357)

[3.3.2 执行升级 8](#_Toc69224358)

[4. 方案 9](#_Toc69224359)

[4.1 Nacos部分 9](#_Toc69224360)

[4.1.1 流程逻辑 9](#_Toc69224361)

[4.1.2 相关接口 11](#_Toc69224362)

[4.2 Gateway部分 13](#_Toc69224363)

[4.2.1 流程逻辑 14](#_Toc69224364)

[4.2.2 策略模式 15](#_Toc69224365)

[5. 实施 15](#_Toc69224366)

[5.1 Nacos部分 15](#_Toc69224367)

[5.1.1 修改位置 15](#_Toc69224368)

[5.1.2 新增逻辑 16](#_Toc69224369)

[5.2 Gateway部分 18](#_Toc69224370)

[5.2.1 全局过滤器 18](#_Toc69224371)

[5.2.2 筛选策略 19](#_Toc69224372)

[6. 验证 21](#_Toc69224373)

[7. 结果 24](#_Toc69224374)

# 1. 需求描述

1. 云枢平台内应用的部署存在多实例，将应用的全部实例升到指定版本，升级期间保证服务可用。
2. 应用在升级期间会存在不同版本的实例同时存在并运行的状态，要保证应用的可靠性。
3. 应用的实例采用k8s进行部署和运维，且部署的环境资源有限。

# 2. 目标要求

* 升级过程中保证服务提供正常的使用，即可用性
* 升级过程中保证业务调用服务实例的版本稳定，不能同一操作调用的服务版本有差异，即可靠性
* 升级过程中保证业务流程中涉及的实例所参与事务的一致性

# 3. 项目现状

## 3.1 运维环境

1. 平台内的应用通过k8s部署对应的服务
2. 服务对外的网关采用SpringCloud Gateway
3. 服务的注册中心使用Nacos
4. 服务间的负载均衡使用Ribbon

## 3.2 项目结构



* 用户通过应用设计器创建应用，并根据版本打包出不同版本的部署包
* 部署包经PaaS平台，对应用进行部署、升级等管理
* PaaS负责云枢内所有应用的运行态管理



PaaS内部的模块依赖关系

* start启动模块
* app-paas-controller控制器模块
* app-paas-app应用模块
* app-paas-infrastructure基础设施模块
* app-paas-domain领域模块
* app-paas-client二方库

其中应用升级相关功能主要涉及两个模块：

* app-paas-controller对外暴露供升级用的api接口
* app-paas-app实现具体的升级逻辑

## 3.3 操作流程

### 3.3.1 检查升级



升级前检查操作流程

### 3.3.2 执行升级



升级操作流程

# 4. 方案

## 4.1 Nacos部分

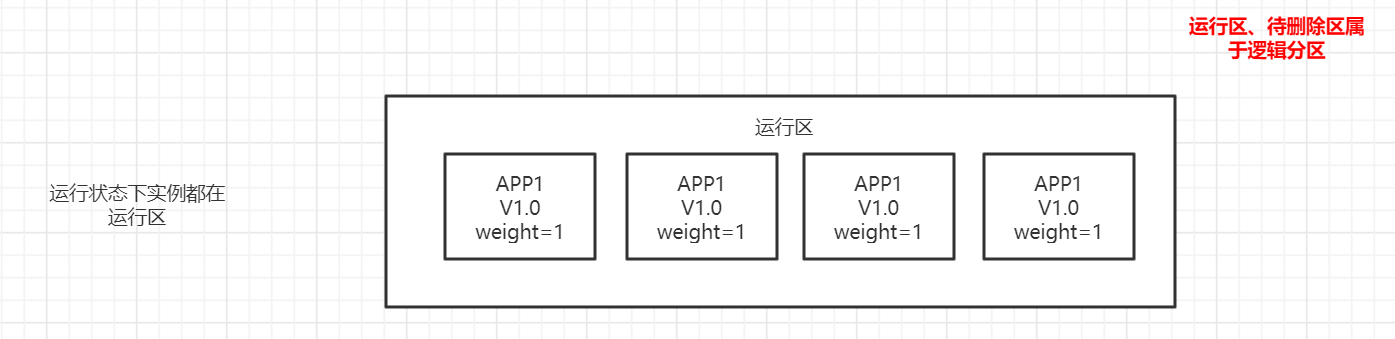
1. 在旧版本实例列表中，筛选出一个实例
2. 将实例的权重设为0，并存入待删除区
3. 启动新版本实例，修改实例权重与其他实例权重相同
4. 运行5min后，重复步骤1、2
5. 在待删除区，按先进先出队列，筛选出一个实例进行下线删除操作
6. 重复此步骤，直至旧版本实例全部下线删除

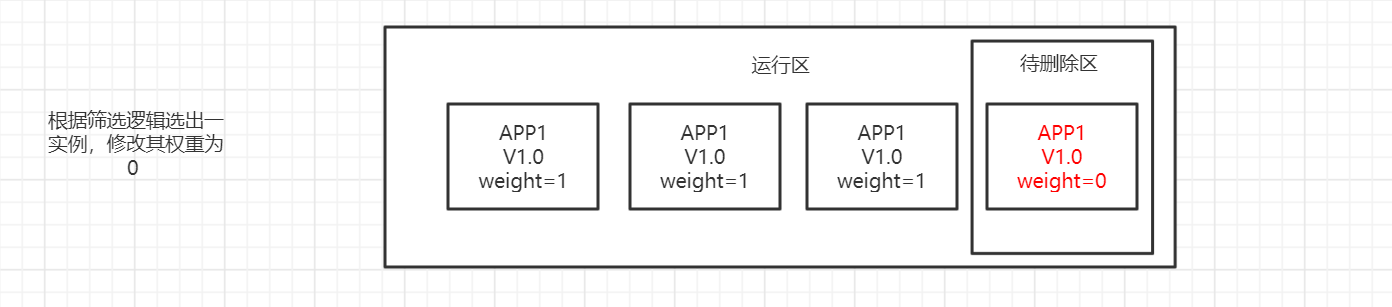
备注1：nacos要在实例元数据声明版本

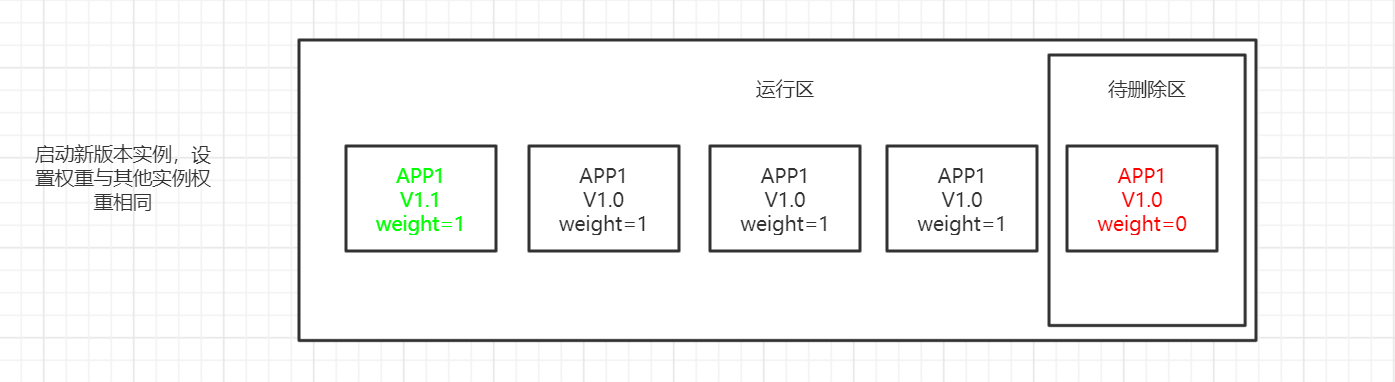
备注2：筛选逻辑做可插拔式策略更改（如：随机选取、分布式事务协调器选取、根据活跃状态选取、待删除区存活时常判定选取等）

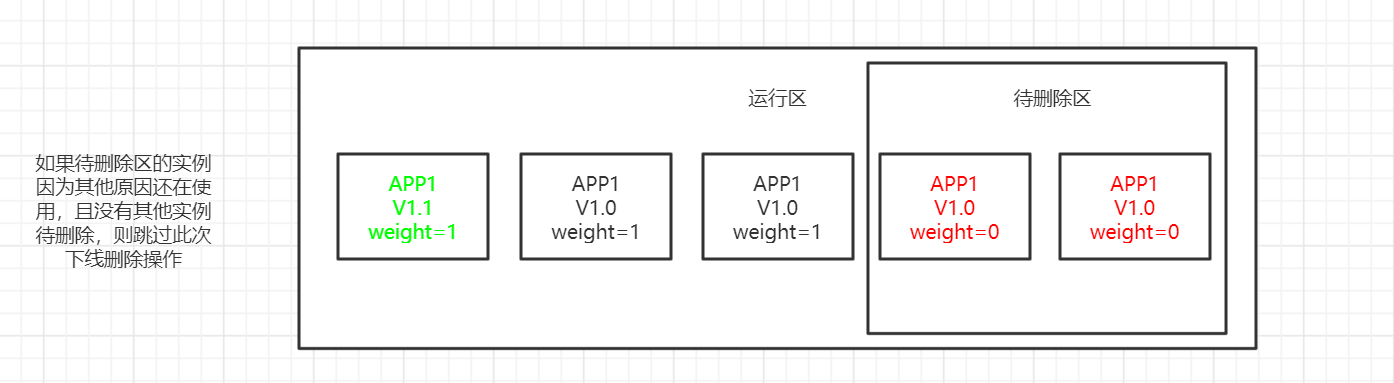
备注3：Ribbon缓存会影响实例的调用关系，可通过配置心跳时间间隔降低影响，这也是第4步等待的原因

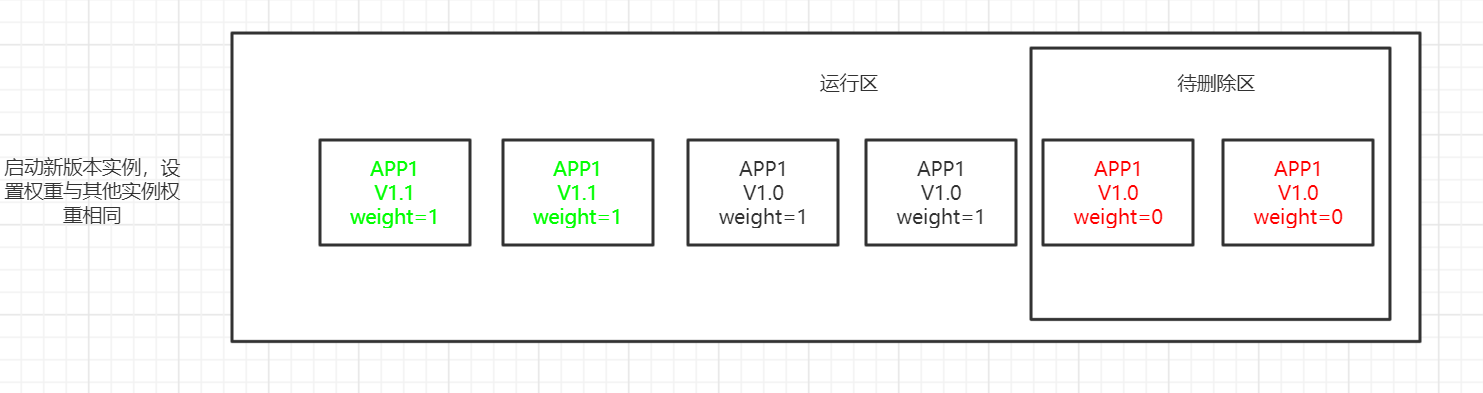
### 4.1.1 流程逻辑

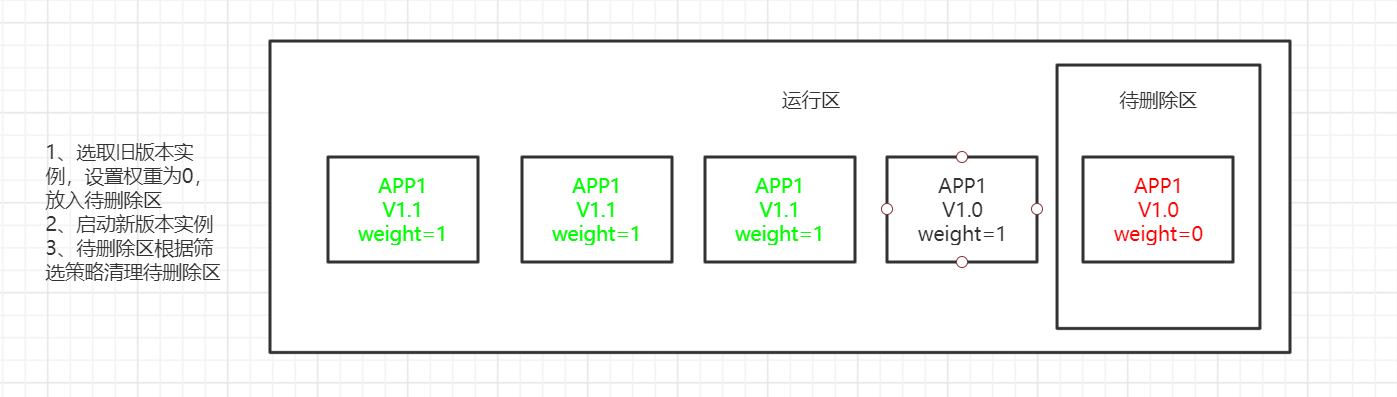


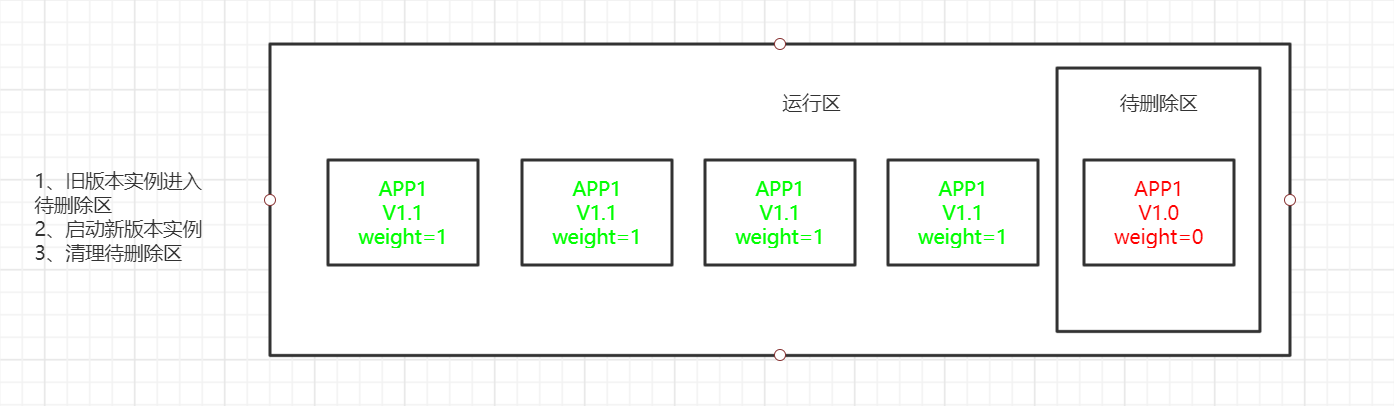


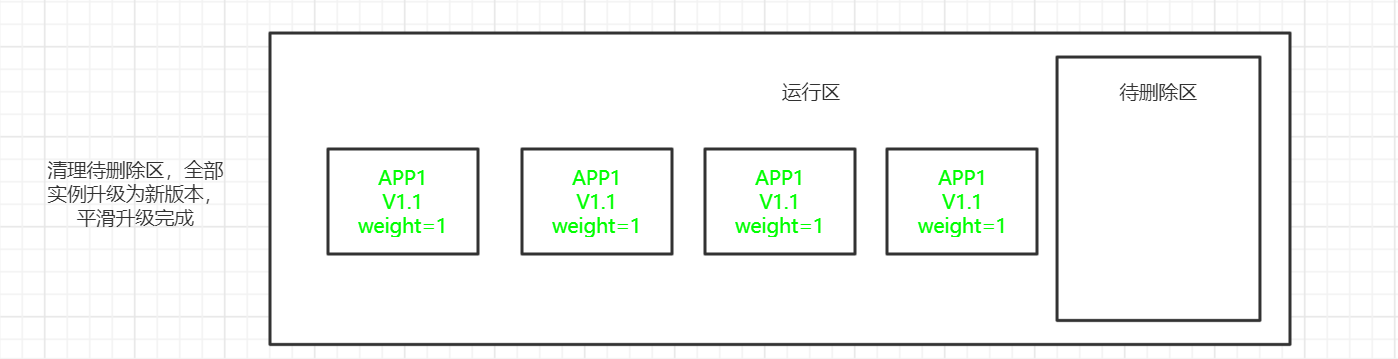












### 4.1.2 相关接口

[**查询实例列表**](https://nacos.io/zh-cn/docs/open-api.html#2.4)

| **请求类型** | GET | **请求路径** | /nacos/v1/ns/instance/list |
| --- | --- | --- | --- |
| **请求参数** | | | |
| **名称** | **类型** | **是否必选** | **描述** |
| serviceName | 字符串 | 是 | 服务名 |
| groupName | 字符串 | 否 | 分组名 |
| namespaceId | 字符串 | 否 | 命名空间ID |
| clusters | 字符串，多个集群用逗号分隔 | 否 | 集群名称 |
| healthyOnly | boolean | 否，默认为false | 是否只返回健康实例 |

**修改实例**

| **请求类型** | PUT | **请求路径** | /nacos/v1/ns/instance |
| --- | --- | --- | --- |
| **请求参数** | | | |
| **名称** | **类型** | **是否必选** | **描述** |
| serviceName | 字符串 | 是 | 服务名 |
| groupName | 字符串 | 否 | 分组名 |
| ip | 字符串 | 是 | 服务实例IP |
| port | int | 是 | 服务实例port |
| clusterName | 字符串 | 否 | 集群名称 |
| namespaceId | 字符串 | 否 | 命名空间ID |
| weight | double | 否 | 权重 |
| metadata | JSON | 否 | 扩展信息 |
| enabled | boolean | 否 | 是否打开流量 |
| ephemeral | boolean | 否 | 是否临时实例 |

**注销实例**

| **请求类型** | DELETE | **请求路径** | /nacos/v1/ns/instance |
| --- | --- | --- | --- |
| **请求参数** | | | |
| **名称** | **类型** | **是否必选** | **描述** |
| serviceName | 字符串 | 是 | 服务名 |
| groupName | 字符串 | 否 | 分组名 |
| ip | 字符串 | 是 | 服务实例IP |
| port | int | 是 | 服务实例port |
| clusterName | 字符串 | 否 | 集群名称 |
| namespaceId | 字符串 | 否 | 命名空间ID |
| ephemeral | boolean | 否 | 是否临时实例 |

**详见：https://nacos.io/zh-cn/docs/open-api.html**

## 4.2 Gateway部分

1. 自定义全局过滤器，拦截请求重定向所访问服务的实例
2. 添加平滑负载策略，使旧版本接收的请求降低或阻绝
3. 通过策略模式+动态配置，实现负载策略的动态变更

备注1：权限为0服务可用，但不可见

备注2：负载策略使用策略设计模式

备注3：配置可通过nacos动态变换

### 4.2.1 流程逻辑



### 4.2.2 策略模式

* 权重策略：根据实例的权重进行引流
* Scheme策略：根据请求的scheme进行引流
* 特殊请求头策略：根据请求信息进行引流

# 5. 实施

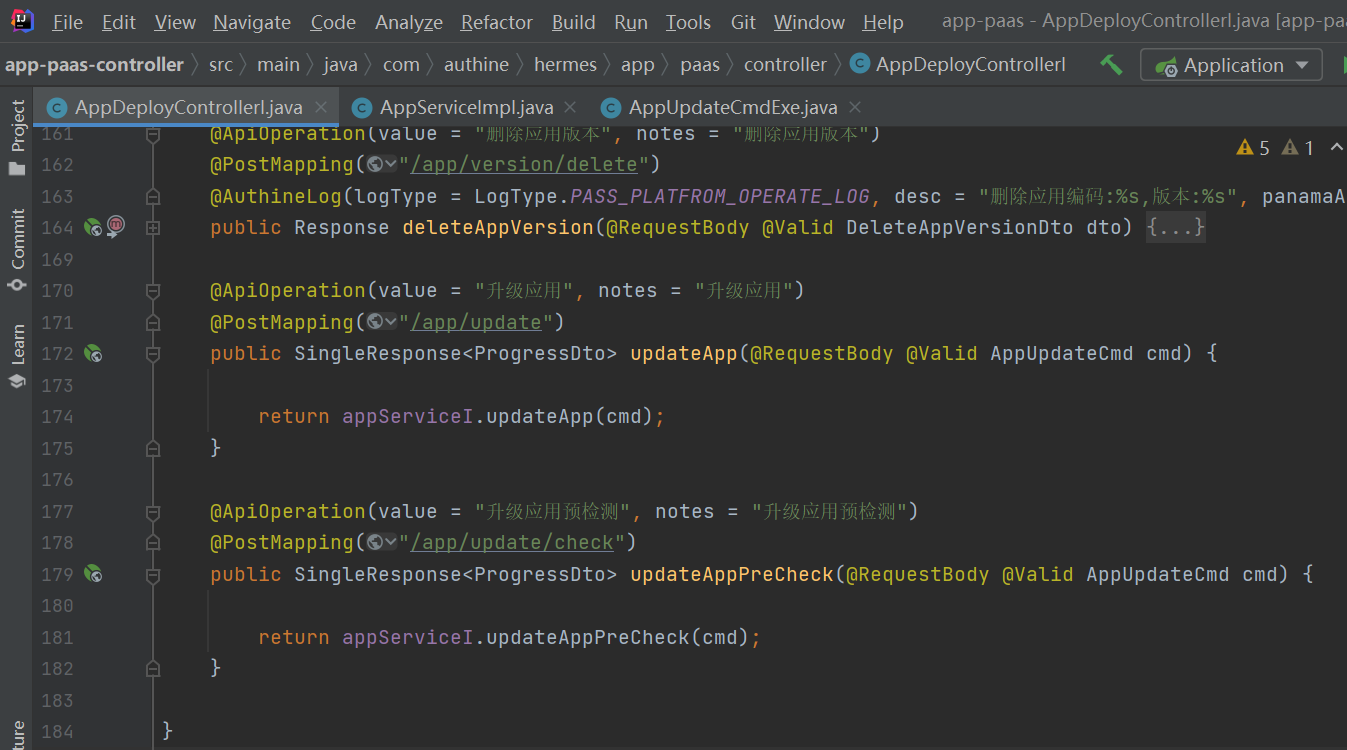
## 5.1 Nacos部分

### 5.1.1 修改位置

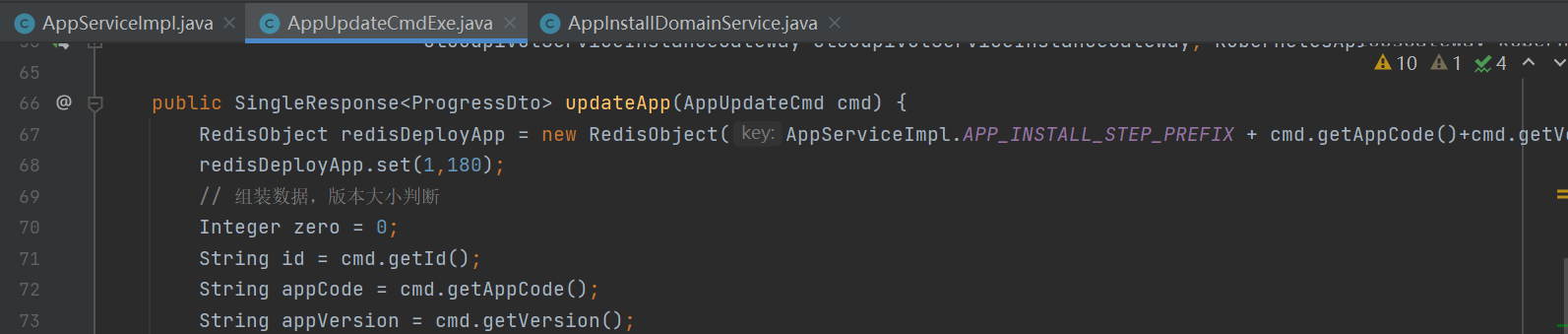
项目地址：git@gitlab.h3yun.com:hermes/paas/app-paas.git

分支：dev

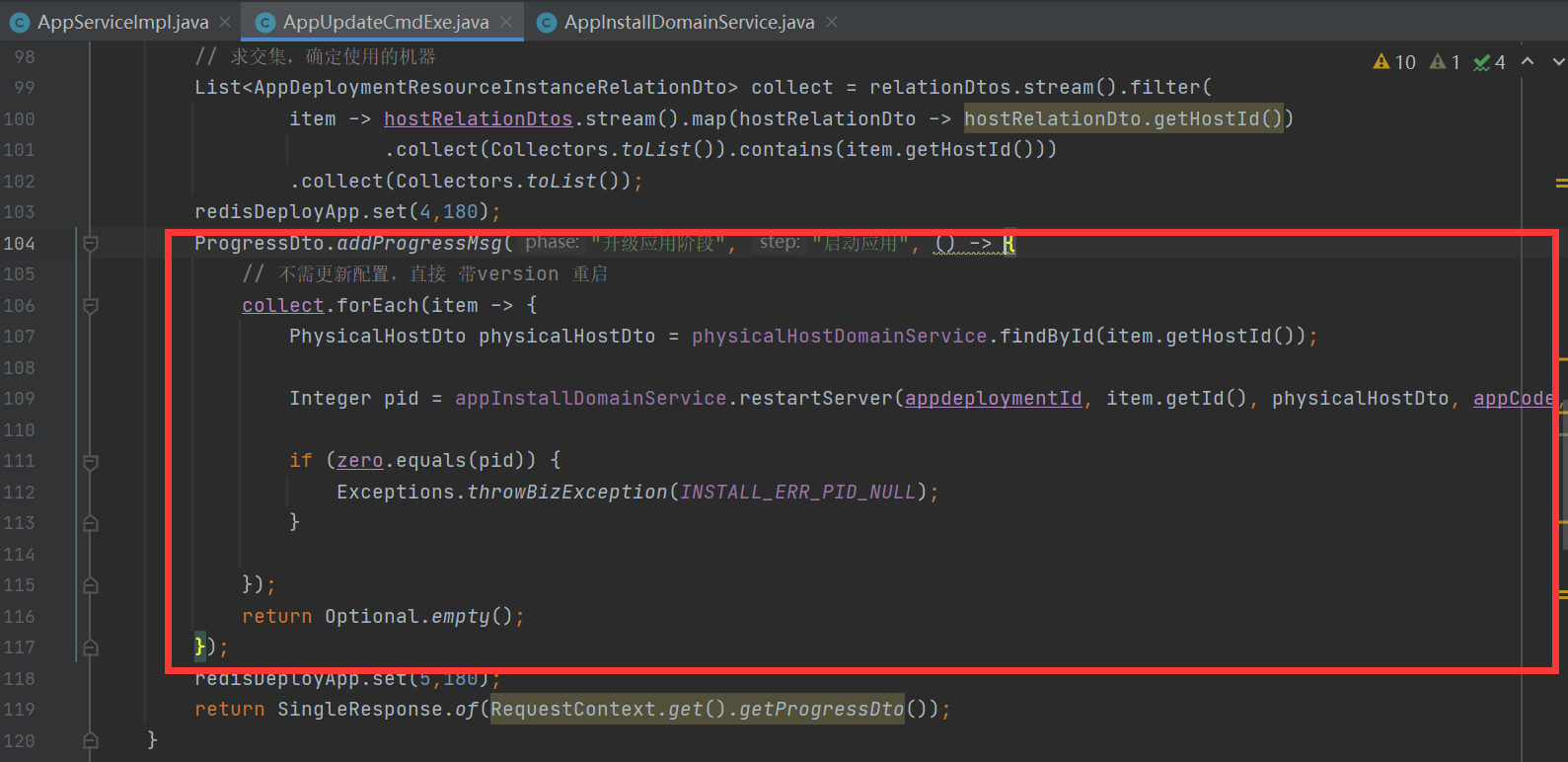
Module：app-paas-controller、app-paas-app



Api入口



升级逻辑方法



### 5.1.2 新增逻辑

伪代码如下：

class SmoothUpdateThread implements Runnable{

Integer aimCount;

public SmoothUpdateThread(Integer aimCount) {

this.aimCount = aimCount;

}

@Override

public void run() {

smoothUpdate();

}

private void smoothUpdate() {

//获取Nacos对应实例列表

List<Instance> instanceList = getInstancesListFromNacos();

//根据实例的metaData，选取其中一个实例

Instance readyToDie = choseInferiorInstance(instanceList);

//将选中实例不可见

invisiableInstance(readyToDie);

ThreadLocal<List<Instance>> threadLocal = new ThreadLocal();

List<Instance> waitToDieInstancesList = threadLocal.get("DELETE\_LIST");

//检查待删除区，将待删除区内的合适实例注销

Integer count = checkWaitToDieInstancesList(waitToDieInstancesList);

waitToDieInstancesList.add(readyToDie);

//启动新实例

Instance newInstances = turnNewInstance(count, aimCount);

//设置新实例权重

resetNewInstance(newInstance);

//再次调度

if (aimCount != getInstancesListFromNacos().size()) {

Thread.sleep(5 \* 60 \* 1000);

smoothUpdate();

}

}

}

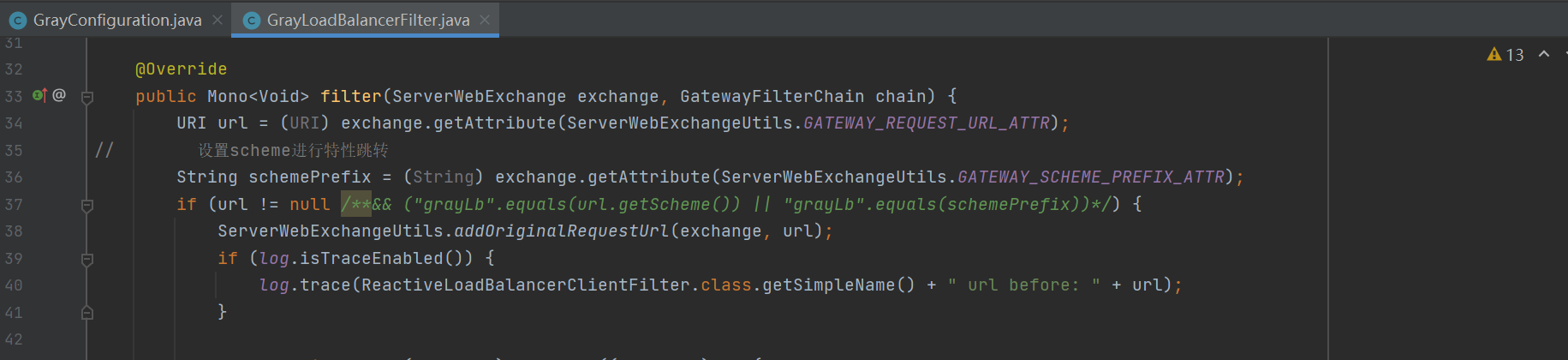
## 5.2 Gateway部分

### 5.2.1 全局过滤器

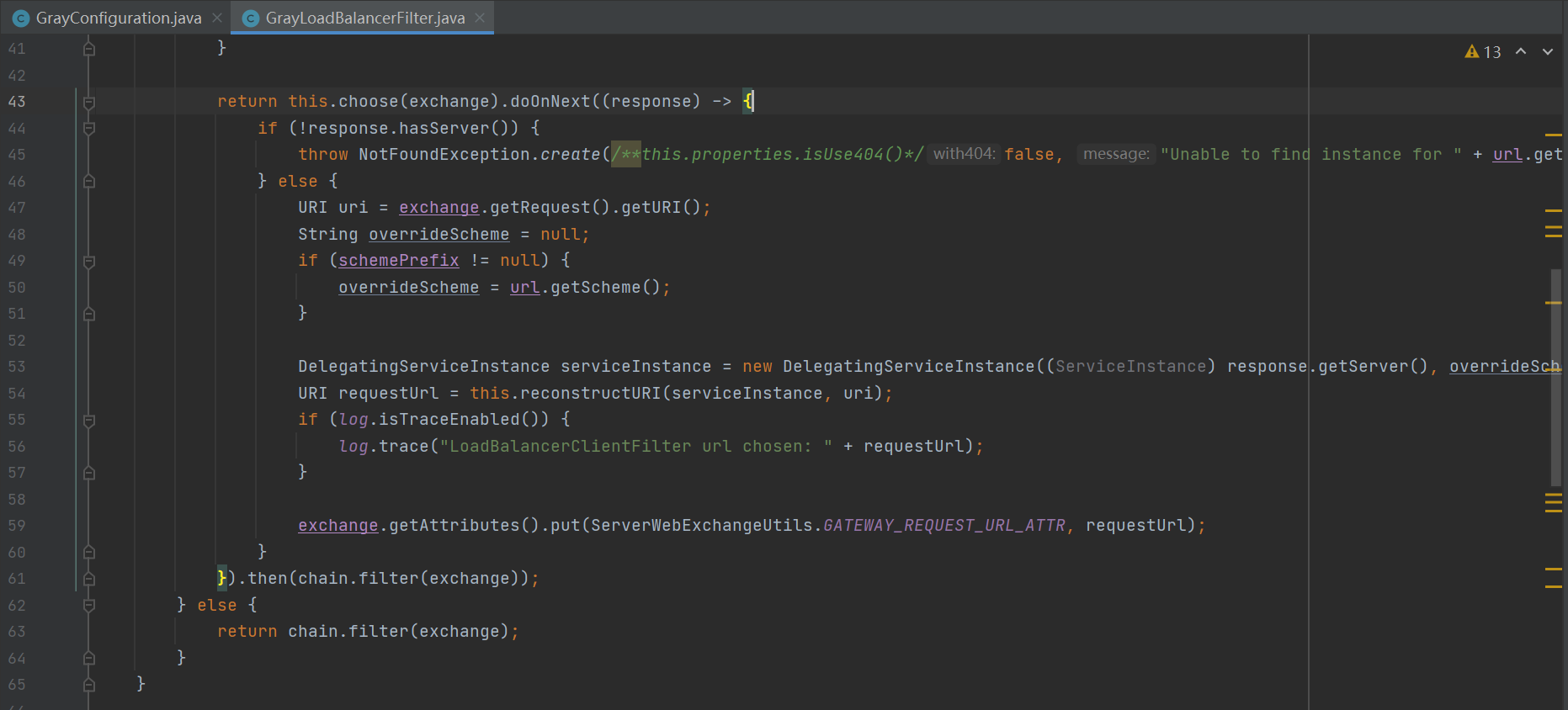
**1、注册**



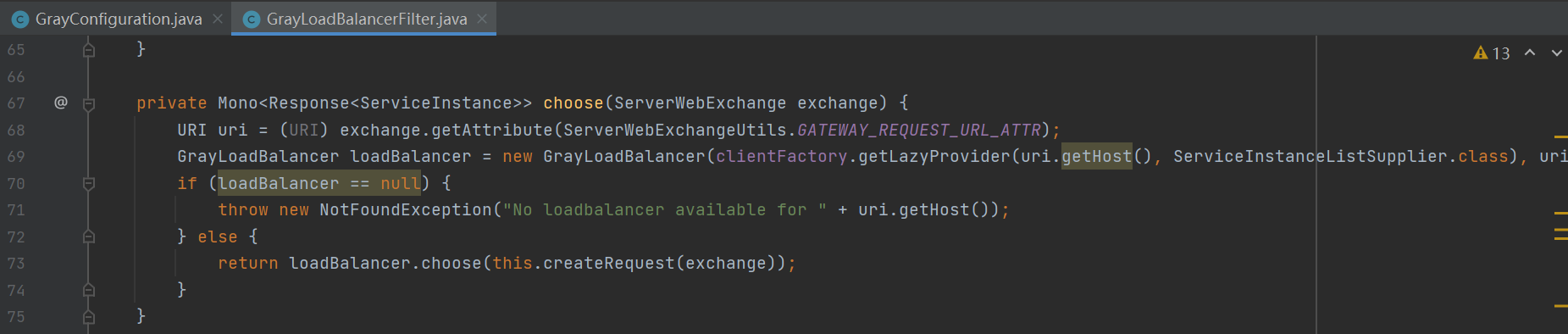
**2、过滤逻辑**



通过url、scheme判断，后期可扩展



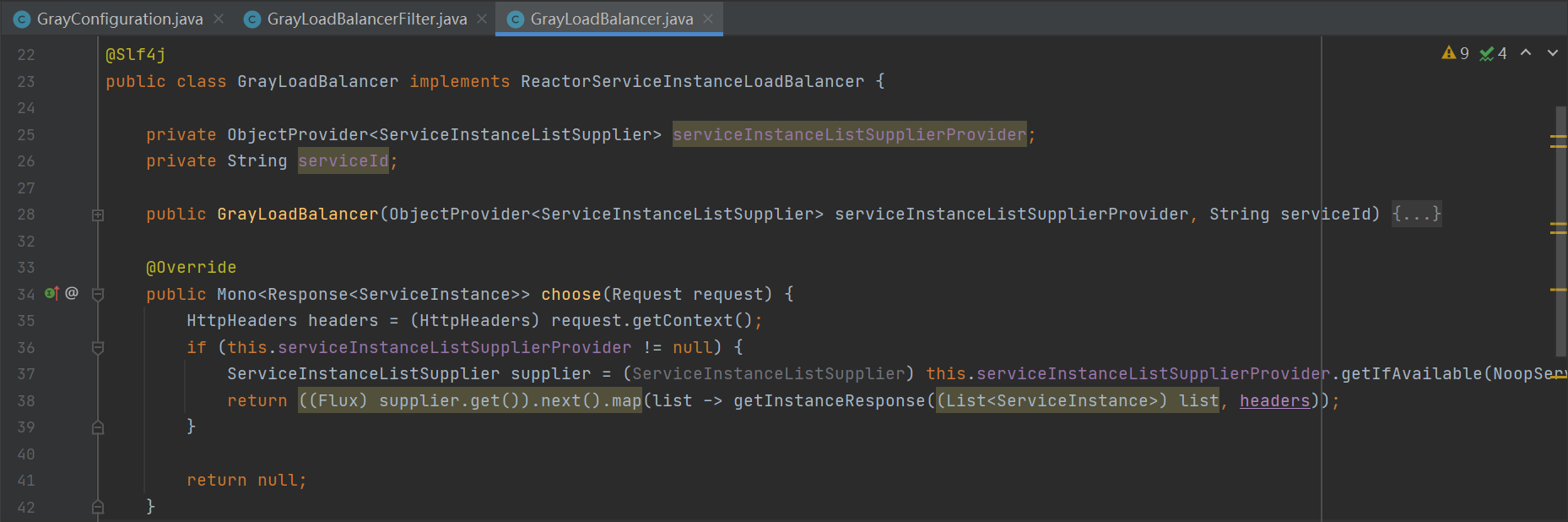
对response做处理



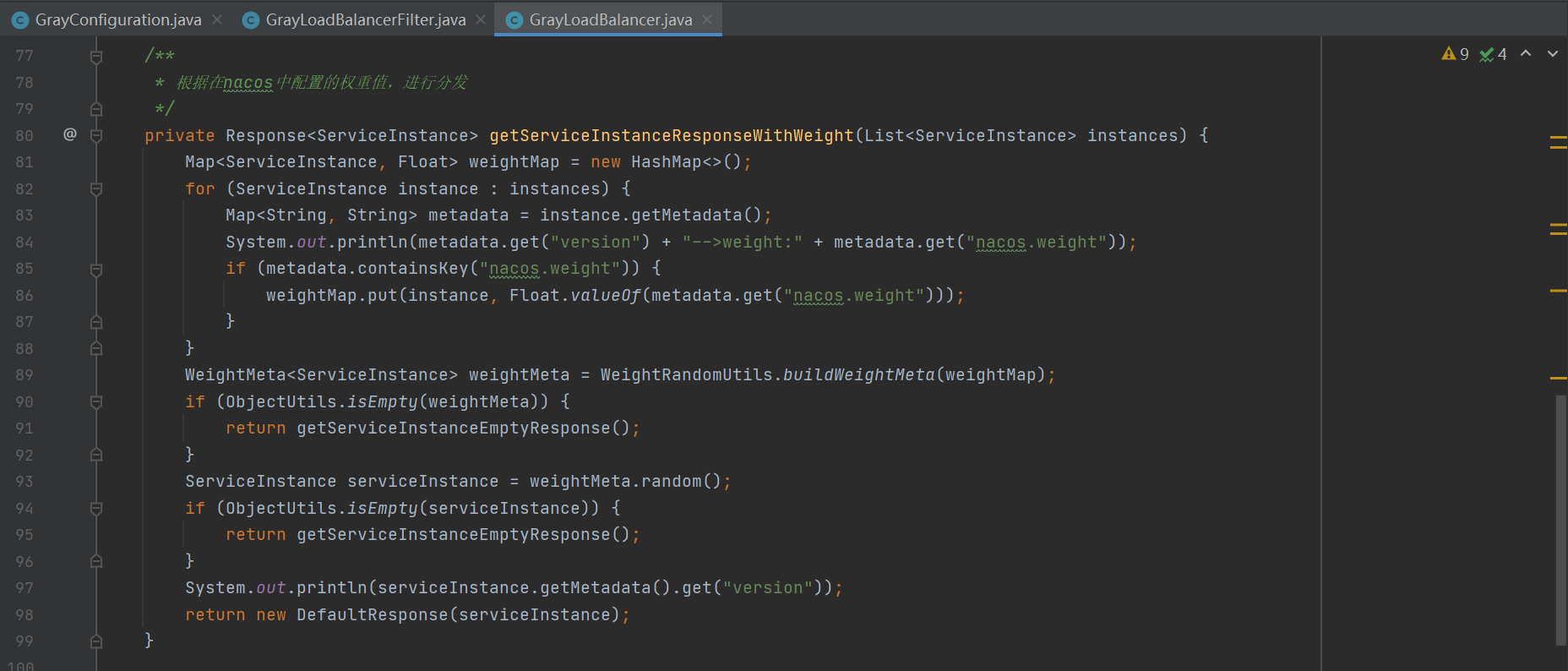
进入封装的负载均衡处理器

### 5.2.2 筛选策略

**1、筛选实例策略**

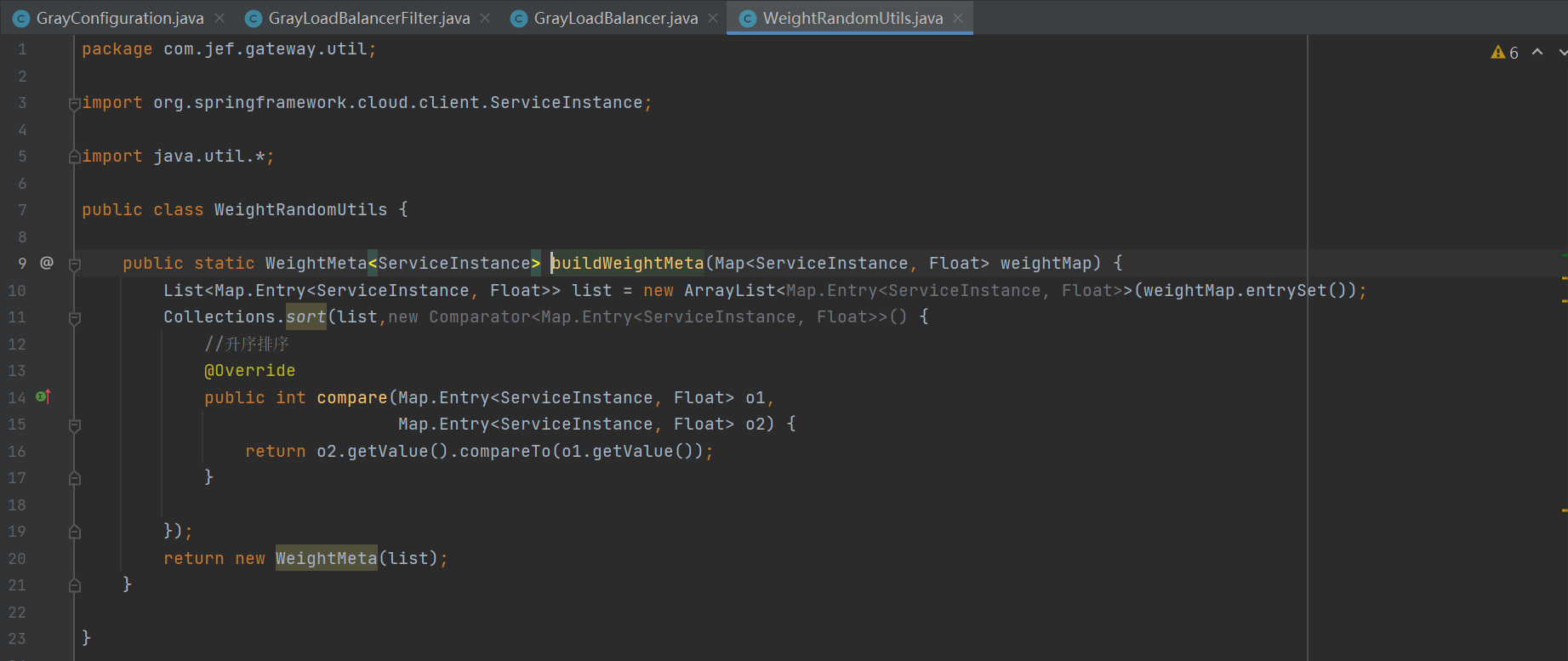


进入策略模式（后期编码实现此设计模式）

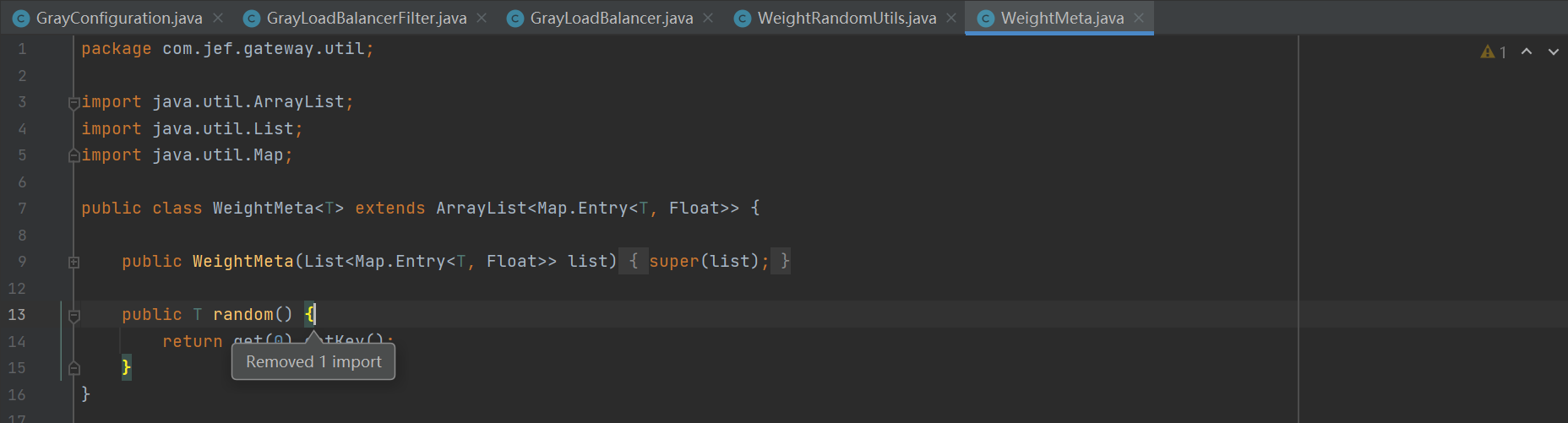


以权重为依据进行实例选择

**2、排序规则**



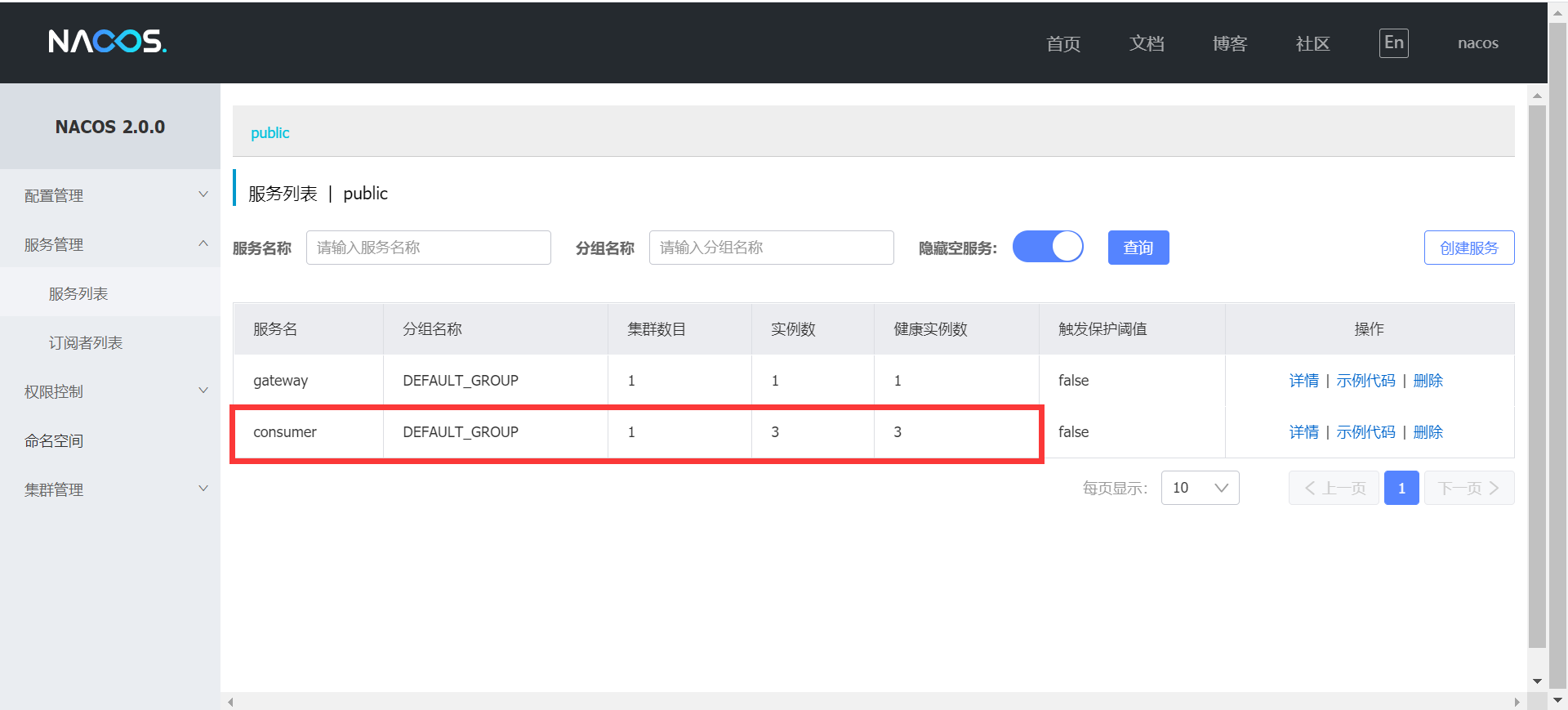
通过权重对实例排序



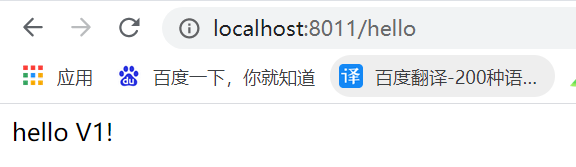
封装工具类，为后期扩展用

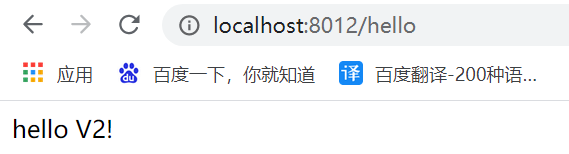
# 6. 验证

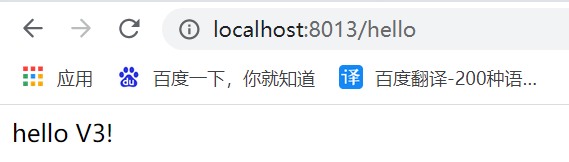
启动三个实例



单独验证实例







通过Nacos控制台修改权重

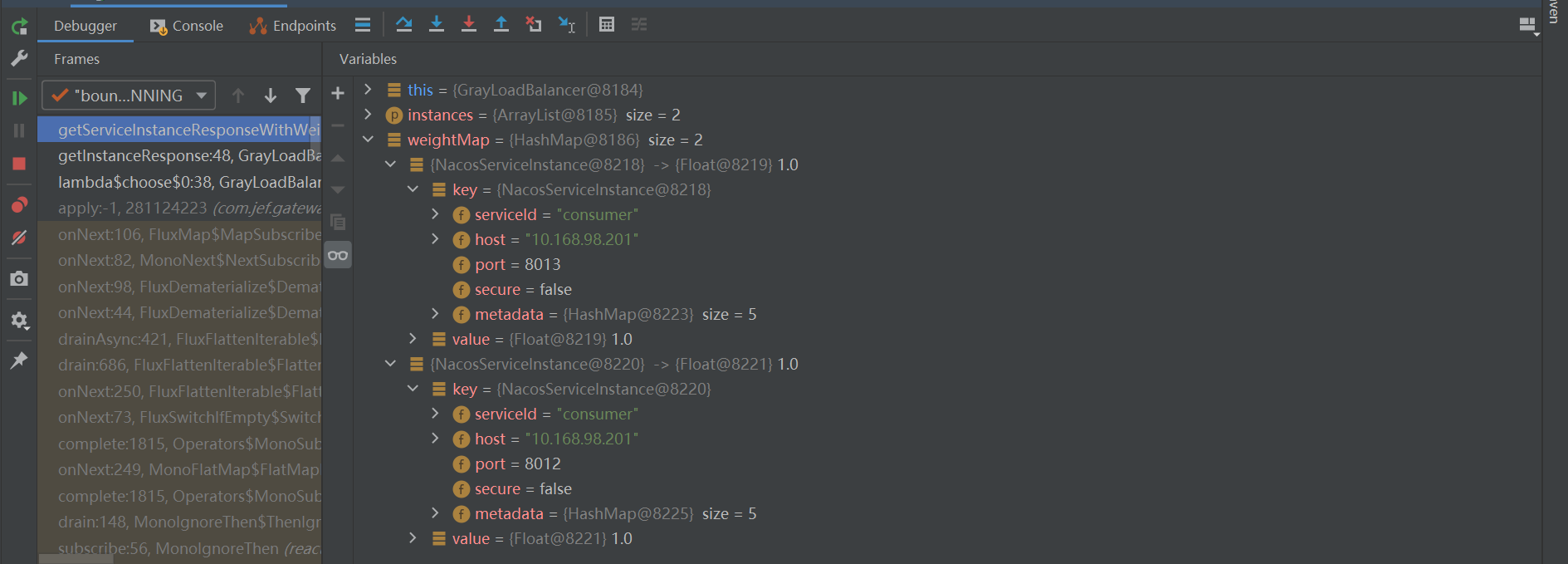




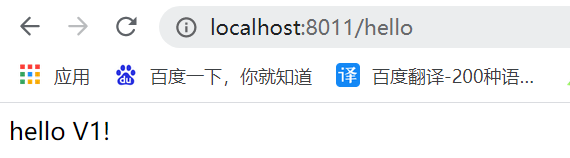
通过gateway访问服务



现在的负载策略是选取实例列表的第一项，所以结果返回正确，已将流量迁移到新版本实例。



通过对实例选取策略debug可以看到，旧版本实例的服务已不可见，但实例还在正常运行



证明请求已经引流到新版本实例

# 7. 结果

通过对gateway的改造已经达到，自定义策略的引流；

通过对Nacos的开发，能够实现对服务实例平稳的下线；

整体方案能够实现服务的平滑升级