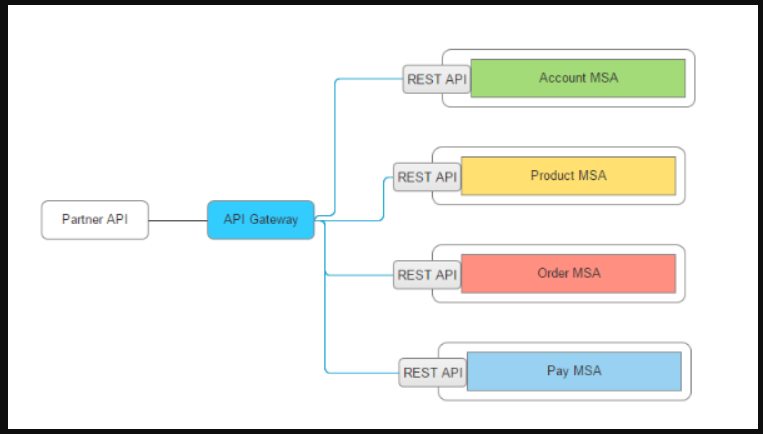
# 第四章 网关服务的搭建

# API网关介绍

API网关是一个服务器，是系统的唯一入口。从面向对象设计的角度看，它与外观模式类似。API网关封装了系统内部架构，为每个客户端提供一个定制的API。它可能还具有其它职责，如身份验证、监控、负载均衡、缓存、请求分片与管理、静态响应处理。API网关方式的核心要点是，所有的客户端和消费端都通过统一的网关接入微服务，在网关层处理所有的非业务功能。通常，网关也是提供REST/HTTP的访问API。



网关应当具备以下功能：

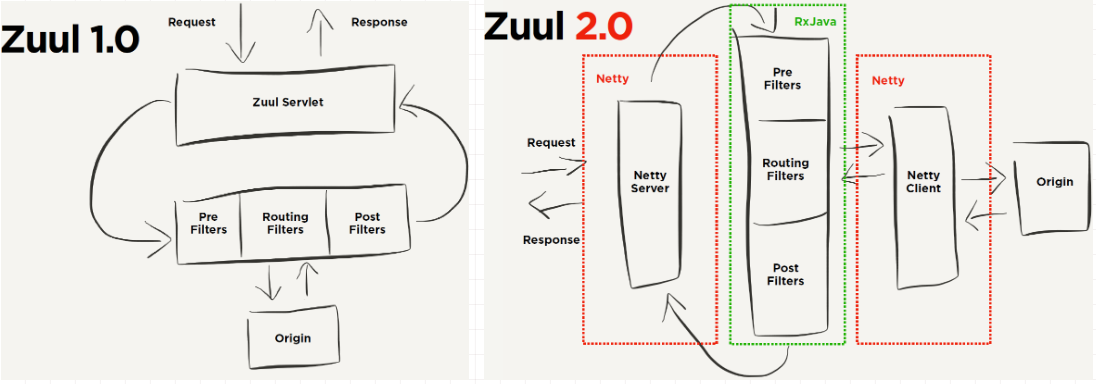
* 性能：API高可用，负载均衡，容错机制。
* 安全：权限身份认证、脱敏，流量清洗，后端签名（保证全链路可信调用）,黑名单（非法调用的限制）。
* 日志：日志记录（spainid,traceid）一旦涉及分布式，全链路跟踪必不可少。
* 缓存：数据缓存。
* 监控：记录请求响应数据，api耗时分析，性能监控。
* 限流：流量控制，错峰流控，可以定义多种限流规则。
* 灰度：线上灰度部署，可以减小风险。
* 路由：动态路由规则。

目前，比较流行的网关有：Nginx 、 Kong 、Orange等等，还有微服务网关Zuul 、Spring Cloud Gateway等等。

# Spring Cloud Gateway介绍

既然有Nginx这样成熟的网关，那为什么还需要Gateway呢？

* 像Nginx这类网关，性能肯定是没得说，它适合做那种门户网关，是作为整个全局的网关，是对外的，处于最外层的；而Gateway这种，更像是业务网关，主要用来对应不同的客户端提供服务的，用于聚合业务的。各个微服务独立部署，职责单一，对外提供服务的时候需要有一个东西把业务聚合起来。
* 像Nginx这类网关，都是用不同的语言编写的，不易于扩展；而Gateway就不同，它是用Java写的，易于扩展和维护
* Gateway这类网关可以实现熔断、重试等功能，这是Nginx不具备的
* Spring Cloud已经放弃Netflix Zuul了。现在Spring Cloud中引用的还是Zuul 1.x版本，而这个版本是基于过滤器的，是阻塞IO，不支持长连接。Zuul 2.x版本跟1.x的架构大一样，性能也有所提升。既然Spring Cloud已经不再集成Zuul 2.x了，那么是时候开始使用Spring Cloud Gateway了。



# 使用Spring Cloud Gateway

## 3.1 在gateway-server里面添加依赖

|  |
| --- |
| *<*dependencies*>  <*dependency*>  <*groupId*>*org.springframework.cloud*</*groupId*>  <*artifactId*>*spring-cloud-starter-gateway*</*artifactId*>  </*dependency*>  <*dependency*>  <*groupId*>*com.alibaba.cloud*</*groupId*>  <*artifactId*>*spring-cloud-starter-alibaba-nacos-discovery*</*artifactId*>  </*dependency*> </*dependencies*>* |

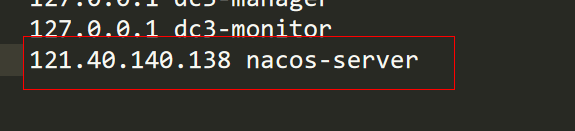
## 3.2 在gateway-server里面添加启动类

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

## 3.3 在gateway-server里面添加配置文件

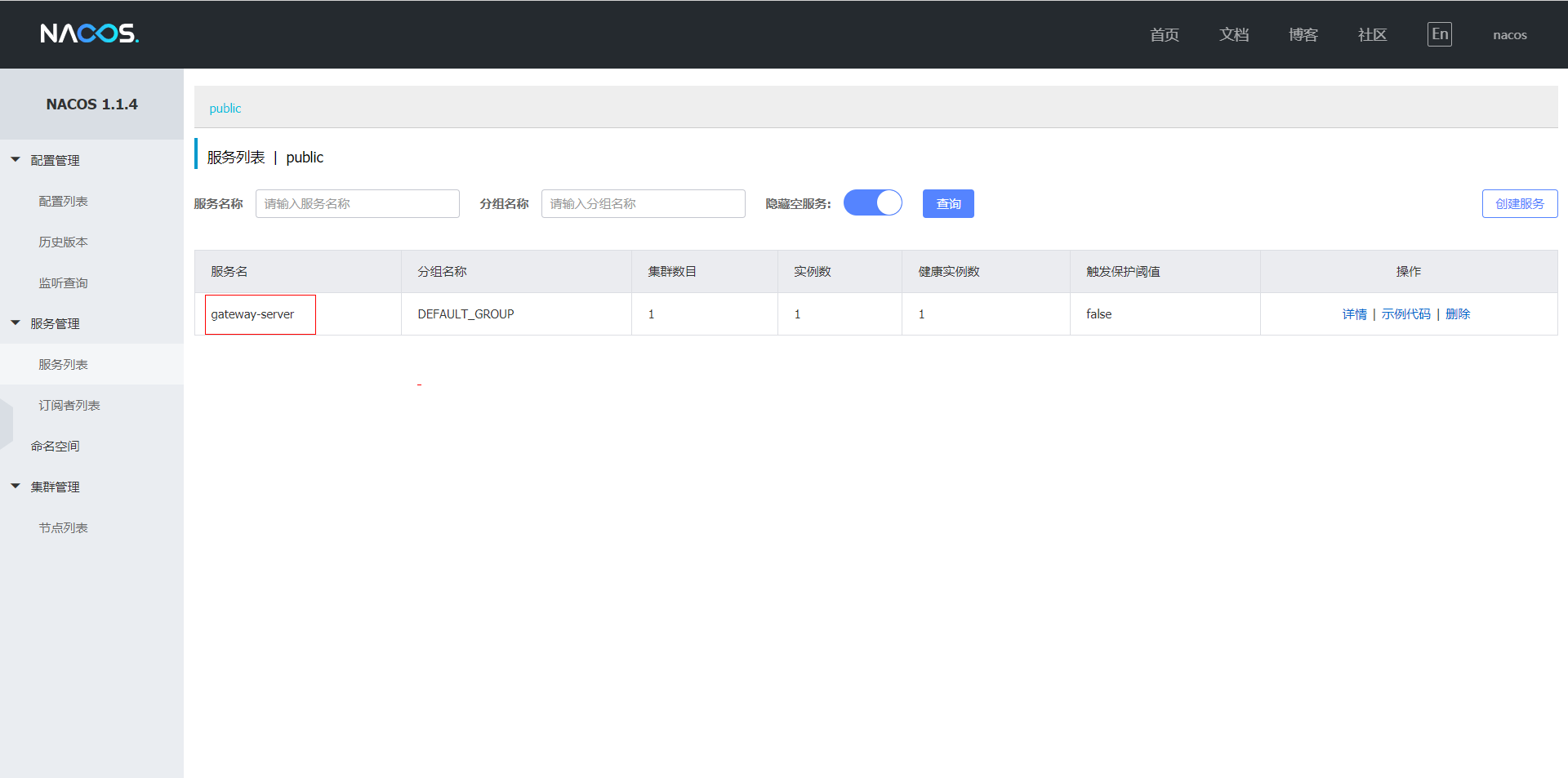
|  |
| --- |
| server:  port: 80 spring:  cloud:  gateway:  discovery:  locator:  enabled: true *# 启用gateway 服务发现功能* lower-case-service-id: true *# 启用小驼峰的服务名称访问服务* routes: *# 自定义路由机制* -id: admin-service-router *# /admin/\*\*--->admin-service* uri: lb://admin-service  predicates:  -Path = admin/\*\*  -id: member-service-router *# /member/\*\*--->member-service* uri: lb://member-service  predicates:  -Path = member/\*\*  - id: finance-service-router *# /finance/\*\*--->finance-service* uri: lb://finance-service  predicates:  - Path = finance/\*\*  - id: chan-service-router *# /chan/\*\*--->chan-service* uri: lb://chan-service  predicates:  - Path = chan/\*\*  - id: statistics-service-router *# /statistics/\*\*--->statistics-service* uri: lb://statistics-service  predicates:  - Path=statistics/\*\*  - id: match-service-router *# /match/\*\*--->match-service* uri: lb://match-service  predicates:  - Path=match/\*\*  - id: exchange-service-router *# /exchange/\*\*--->exchange-service* uri: lb://exchange-service  predicates:  - Path=match/\*\*  nacos:  server-addr: nacos-server:8848 |

## 3.4 修改本机的hosts文件



## 3.5 启动访问测试

启动gateway-server，观察Nacos注册中心：



发现网关服务已经上线。

测试路由功能：

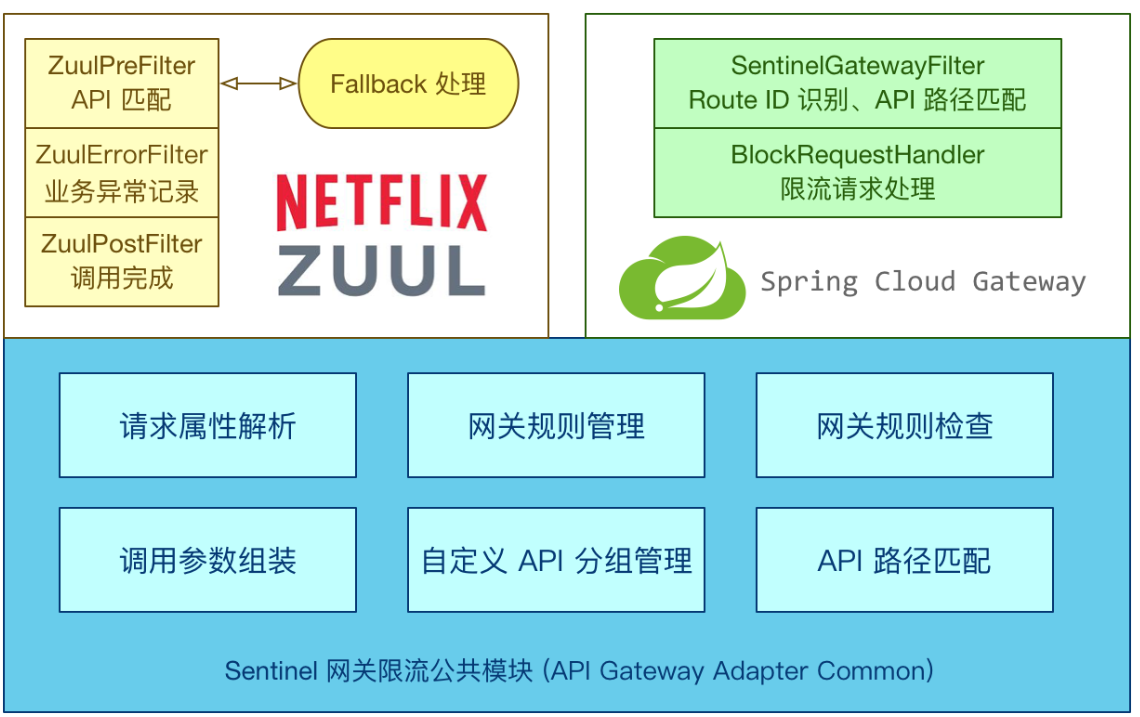


## 3.6 修改配置文件

|  |
| --- |
| server:  port: 80 spring:  application:  name: gateway-server  cloud:  nacos:  discovery:  server-addr: nacos-server:8848 *# 修改本机的host 文件* gateway:  discovery:  locator:  enabled: true  lower-case-service-id: true *# admin-service ADMIN-SERVICE /admin-service/\*\* -> 微服务（ADMIN-SERVICE）* routes:  - id: amdin-service\_router  uri: lb://admin-service *# 转发到那个目的地* predicates:  - Path=/admin/\*\*  **- id: test\_router  uri: http://www.aliyun.com  predicates:  - Path=/product** |

# Spring Cloud Gateway 集成Sentinel

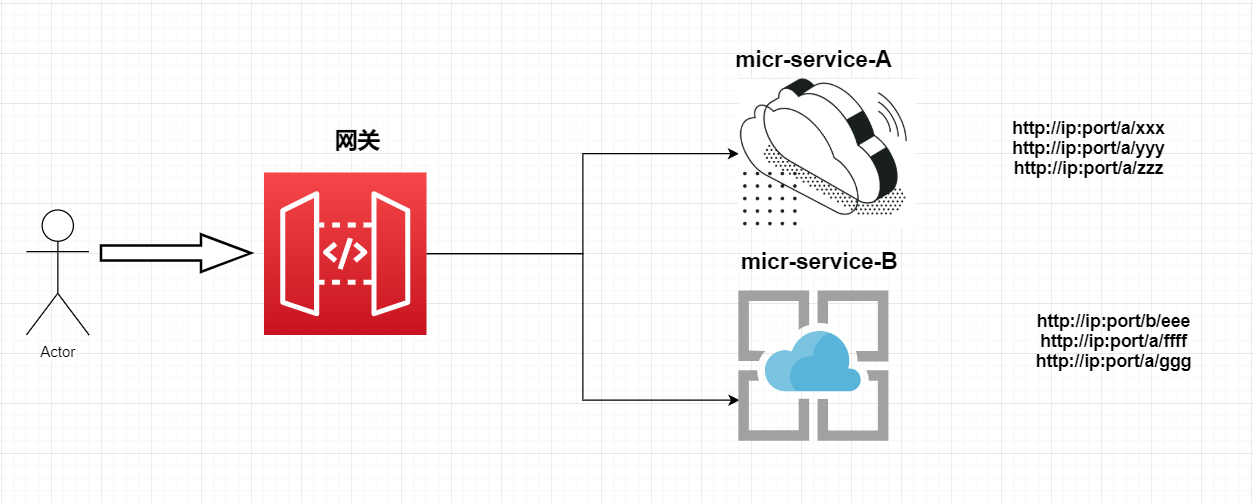
Sentinel 支持对 Spring Cloud Gateway、Zuul 等主流的 API Gateway 进行限流。



参考文档：

|  |
| --- |
| https://github.com/alibaba/Sentinel/wiki/%E7%BD%91%E5%85%B3%E9%99%90%E6%B5%81 |

## 4.1 限流规则的维度



### 4.1.1 网关维度

代表网关访问该服务时的限流规则

粒度粗--全局的配置

### 4.1.2 Api分组维度

代表网关访问该接口时的限流规则

粒度细--局部配置

## 4.2 添加依赖

|  |
| --- |
| *<*dependency*>  <*groupId*>*com.alibaba.cloud*</*groupId*>  <*artifactId*>*spring-cloud-starter-alibaba-sentinel*</*artifactId*> </*dependency*> <*dependency*>  <*groupId*>*com.alibaba.cloud*</*groupId*>  <*artifactId*>*spring-cloud-alibaba-sentinel-gateway*</*artifactId*> </*dependency*>* |

## 4.3 修改配置文件

|  |
| --- |
| server:  port: 80 spring:  application:  name: gateway-server  cloud:  nacos:  discovery:  server-addr: nacos-server:8848 *# 修改本机的host 文件* gateway:  discovery:  locator:  enabled: true  lower-case-service-id: true *# admin-service ADMIN-SERVICE /admin-service/\*\* -> 微服务（ADMIN-SERVICE）* routes:  - id: admin-service\_router  uri: lb://admin-service *# 转发到那个目的地* predicates:  - Path=/admin/\*\*  - id: test\_router  uri: http://www.aliyun.com  predicates:  - Path=/product  *# http://www.baidu.com/test  # http://www.aliyun.com/product* sentinel:  filter:  enabled: true  datasource:  ds1.file:  file: classpath:router-flow.json  ruleType: gw-flow  ds2.file:  file: classpath:api-flow.json *#api-flow.json 接口的分组* ruleType: gw\_api\_group |

## 4.4 添加限流规则

在resource 目录下新建

gw-flow.json

|  |
| --- |
| [  {  "resource": "admin-service\_router",  "resourceMode": 0 ,  "count": 2,  "intervalSec": 60  },  {  "resource": "admin-service-api",  "resourceMode":"1",  "count": 1,  "intervalSec": 60  } ] |

api-group.json

|  |
| --- |
| [  {  "apiName": "admin-service-api",  "predicateItems": [  {  "pattern": "/admin/login"  }  ]  } ] |

规则配置具体参考：

|  |
| --- |
| GatewayFlowRule：网关限流规则，针对 API Gateway 的场景定制的限流规则，可以针对不同 route 或自定义的 API 分组进行限流，支持针对请求中的参数、Header、来源 IP 等进行定制化的限流。  ApiDefinition：用户自定义的 API 定义分组，可以看做是一些 URL 匹配的组合。比如我们可以定义一个 API 叫 my\_api，请求 path 模式为 /foo/\*\* 和 /baz/\*\* 的都归到 my\_api 这个 API 分组下面。限流的时候可以针对这个自定义的 API 分组维度进行限流。  其中网关限流规则 GatewayFlowRule 的字段解释如下：  resource：资源名称，可以是网关中的 route 名称或者用户自定义的 API 分组名称。  resourceMode：规则是针对 API Gateway 的 route（RESOURCE\_MODE\_ROUTE\_ID）还是用户在 Sentinel 中定义的 API 分组（RESOURCE\_MODE\_CUSTOM\_API\_NAME），默认是 route。  grade：限流指标维度，同限流规则的 grade 字段。  count：限流阈值  intervalSec：统计时间窗口，单位是秒，默认是 1 秒。  controlBehavior：流量整形的控制效果，同限流规则的 controlBehavior 字段，目前支持快速失败和匀速排队两种模式，默认是快速失败。  burst：应对突发请求时额外允许的请求数目。  maxQueueingTimeoutMs：匀速排队模式下的最长排队时间，单位是毫秒，仅在匀速排队模式下生效。  paramItem：参数限流配置。若不提供，则代表不针对参数进行限流，该网关规则将会被转换成普通流控规则；否则会转换成热点规则。其中的字段：  parseStrategy：从请求中提取参数的策略，目前支持提取来源 IP（PARAM\_PARSE\_STRATEGY\_CLIENT\_IP）、Host（PARAM\_PARSE\_STRATEGY\_HOST）、任意 Header（PARAM\_PARSE\_STRATEGY\_HEADER）和任意 URL 参数（PARAM\_PARSE\_STRATEGY\_URL\_PARAM）四种模式。  fieldName：若提取策略选择 Header 模式或 URL 参数模式，则需要指定对应的 header 名称或 URL 参数名称。  pattern：参数值的匹配模式，只有匹配该模式的请求属性值会纳入统计和流控；若为空则统计该请求属性的所有值。（1.6.2 版本开始支持）  matchStrategy：参数值的匹配策略，目前支持精确匹配（PARAM\_MATCH\_STRATEGY\_EXACT）、子串匹配（PARAM\_MATCH\_STRATEGY\_CONTAINS）和正则匹配（PARAM\_MATCH\_STRATEGY\_REGEX）。（1.6.2 版本开始支持） |

## 4.6 添加Controller获取限流规则

|  |
| --- |
| @RestController public class FlowRulesController *{* @GetMapping*(*"/gateway"*)* public Set*<*GatewayFlowRule*>* getGatewayFlowRules*(){* return GatewayRuleManager.*getRules()* ;  *}* @GetMapping*(*"/api"*)* public Set*<*ApiDefinition*>* getApiGroupRules*(){* return GatewayApiDefinitionManager.*getApiDefinitions()*;  *} }* |

## 4.7 使用的动态规则的数据源

现在需要解决的问题：

* 规则的持久化问题；
* 规则动态的修改；
* 规则是json 文件，写起来不爽；



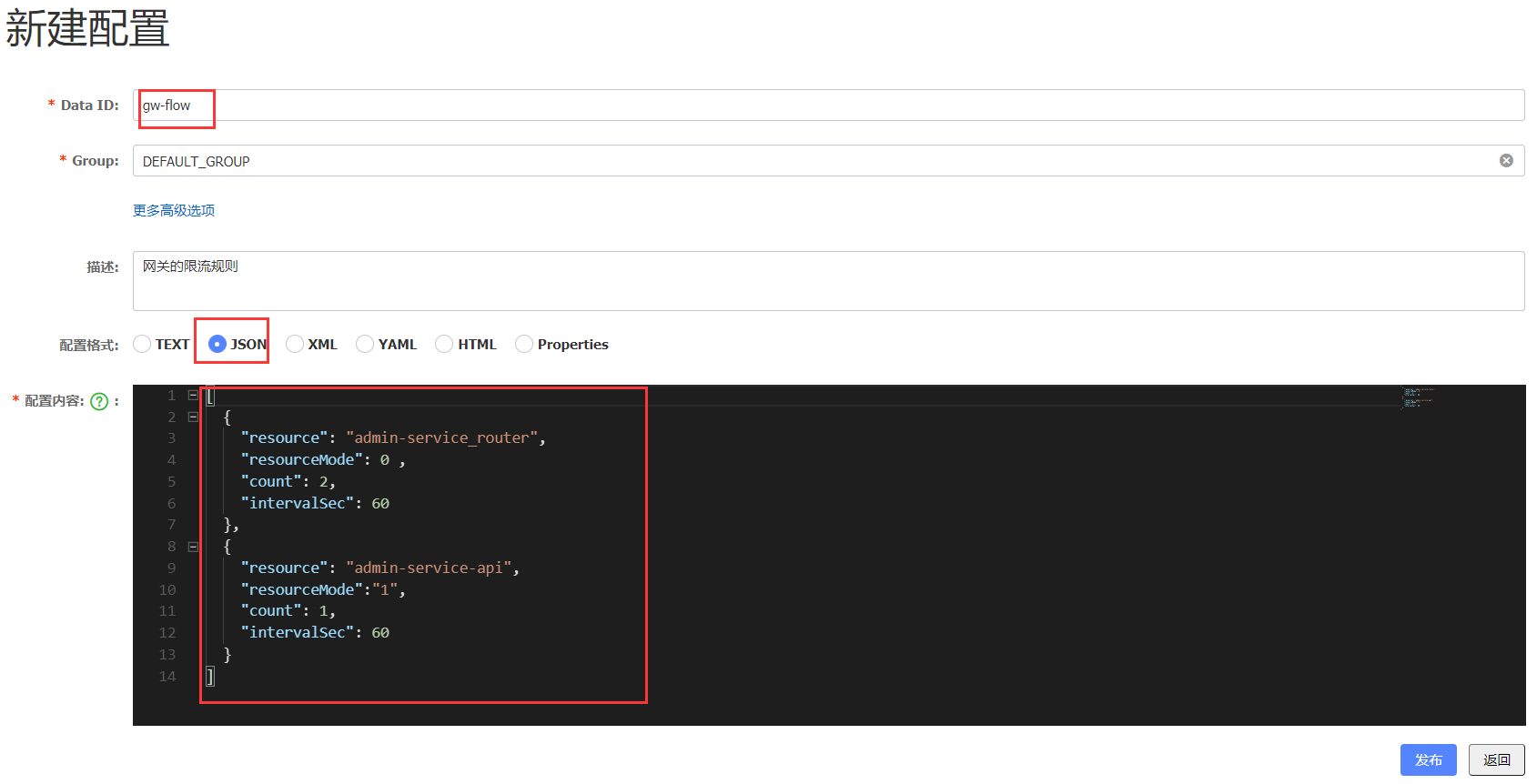
### 4.7.1 使用Nacos持久化规则

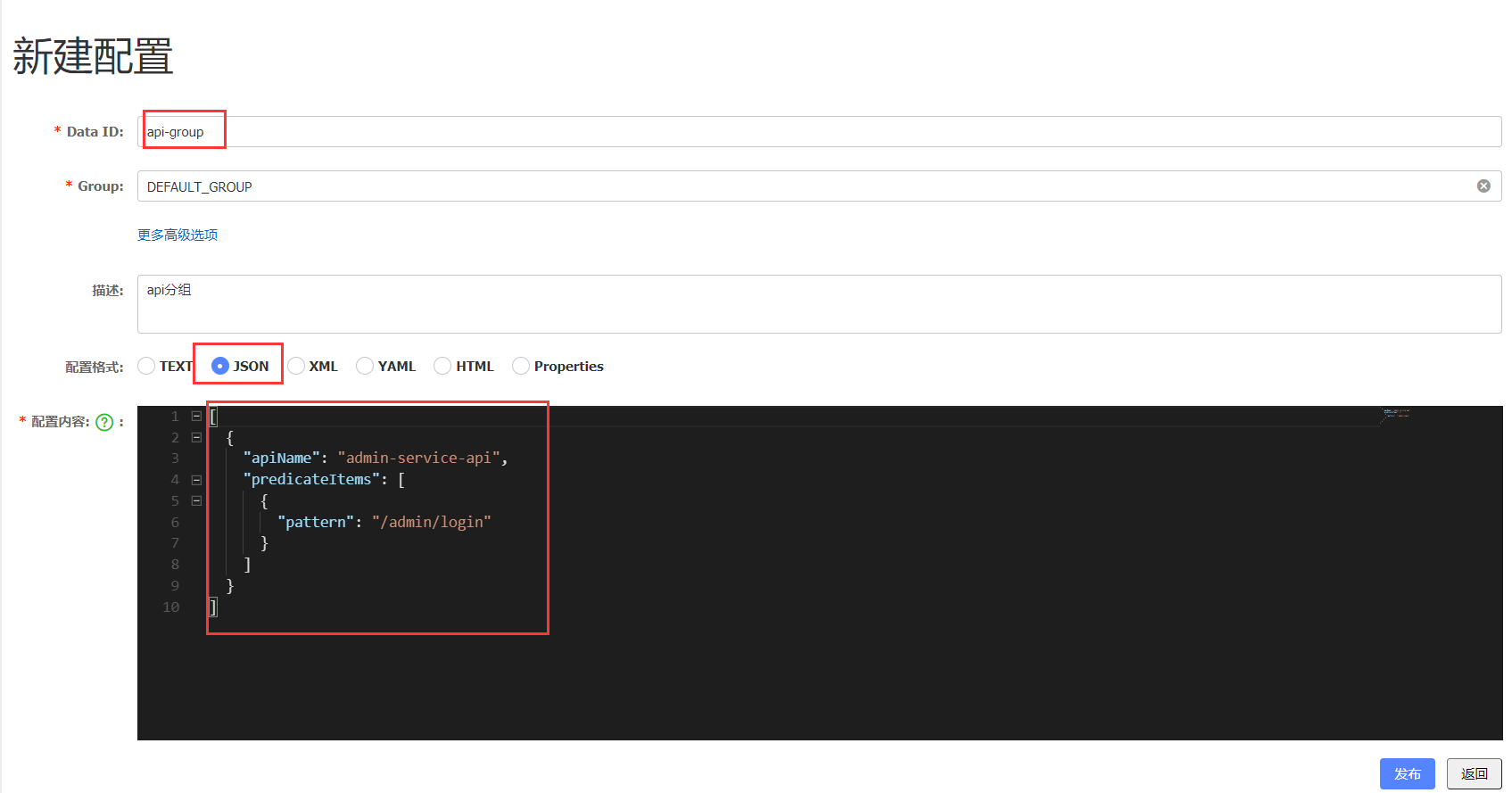
修改配置文件：

|  |
| --- |
| server:  port: 80 spring:  application:  name: gateway-server  cloud:  nacos:  discovery:  server-addr: nacos-server:8848 *# 修改本机的host 文件* gateway:  discovery:  locator:  enabled: true  lower-case-service-id: true *# admin-service ADMIN-SERVICE /admin-service/\*\* -> 微服务（ADMIN-SERVICE）* routes:  - id: admin-service\_router  uri: lb://admin-service *# 转发到那个目的地* predicates:  - Path=/admin/\*\*  - id: test\_router  uri: http://www.aliyun.com  predicates:  - Path=/product  *# /test->http://www.baidu.com 网关自己内部会构建请求去访问我们要访问的地址 GET:http://www.baidu.com/test  # 通过网关来完成对2 种维度的限流： 网关维度 + API 分组* sentinel:  transport:  dashboard: sentinel-server:8858 *# sentinel-dashboard 放在ecs 里面* datasource: *# ds1.file: # file: classpath:gw-flow.json # 网关 + API分组的限流 # ruleType: gw\_flow* ds1.nacos: *#com.alibaba.csp.sentinel.datasource.nacos.NacosDataSource 使用Nacos持久化我的sentinel 数据时，需要添加nacos-datasource的依赖* serverAddr: nacos-server:8848  dataId: gw-flow  ruleType: gw\_flow *# # ds2.file: # file: classpath:api-group.json # ruleType: gw\_api\_group* ds2.nacos:  serverAddr: nacos-server:8848  dataId: api-group  ruleType: gw\_api\_group |

在Nacos里面新增DataID:

gw-flow:



api-group:  


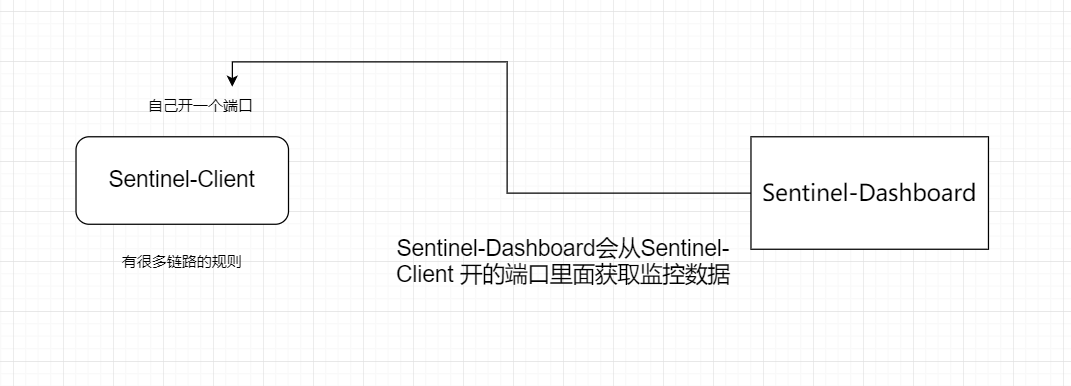
添加依赖：

|  |
| --- |
| <dependency>  <groupId>com.alibaba.csp</groupId>  <artifactId>sentinel-datasource-nacos</artifactId> </dependency> |

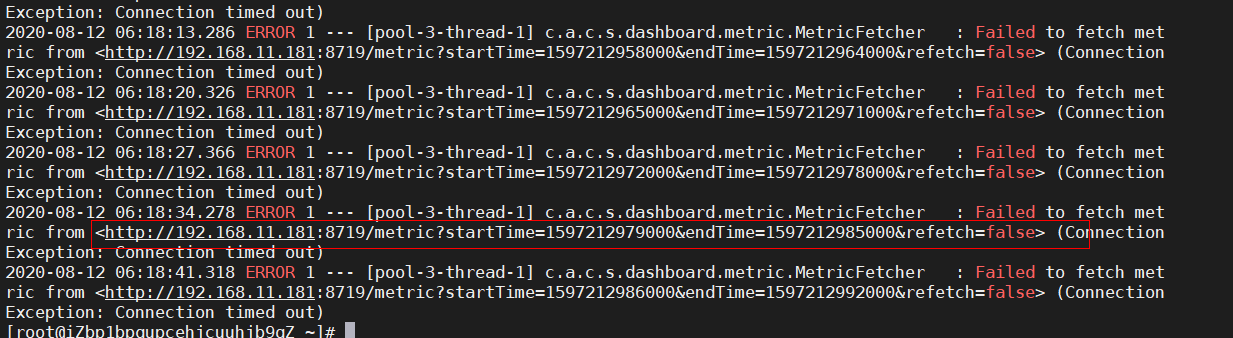
### 4.7.2 关于Sentinel-Dashboard

我们在Sentinel-Dashboard里面能看见gateway-server，但是看不见里面具体的链路。

原理解析：



原因在于，我们的Sentinel-Client没有一个公网的地址，只有一个局域网的地址，Sentinel-Dashboard会使用我们的局域网地址去拉取数据，结构可想而知。



### 4.7.3 使用Sentinel生成规则



