kong

konga: kong dashboard

kong knowledge

####################################

### kong

\* official-site: https://getkong.org/

\* github: https://github.com/Mashape/kong

\* install: https://konghq.com/install/?itm\_source=website&itm\_medium=nav

\* kong-admin-api: <https://docs.konghq.com/2.1.x/admin-api/>

#### kong 特性：

开源的API Gateway、 基于OpenResty+Lua，继承了动态路由

熔断、健康检查、日志(HTTP tcp udp 请求响应)

鉴权、黑白名单、SSL、监控、认证、限流、Rest API 进行配置管理、分布式

高性能,后端nginx

Kong 是基于 OpenResty，Kong 的相当于是 Nginx 的plus。 Kong = OpenResty + Nginx + Lua

###konga 特性:

多用户管理

管理多个Kong节点: kong 集群，指向同一数据库

电子邮件异常信息通知

管理所有Kong Admin API

使用快照备份，还原和迁移Kong节点 Snapshots Snapshot and export json

恢复：add connection -----> Snapshots restore

使用运行状况检查监控节点和API状态

轻松的数据库集成（MySQL，postgresSQL，MongoDB）

kong arch

####################################

database ----- kong ----- web(konga)

####数据库

database: 数据中心用于存储Kong集群节点信息、API、消费者、插件等信息，目前提供了PostgreSQL和Cassandra支持，如果需要高可用建议使用Cassandra

K8s yml详见：search deploy by k8s

####端口

client port: http 8000 https 8443

admin api: http 8001 https 8444

updateconfig； api(信息获取使用: GET 创建：POST 更新：PATCH 删除：DELETE)

\* admin: restful api<-->kong server<-(cache)-->db

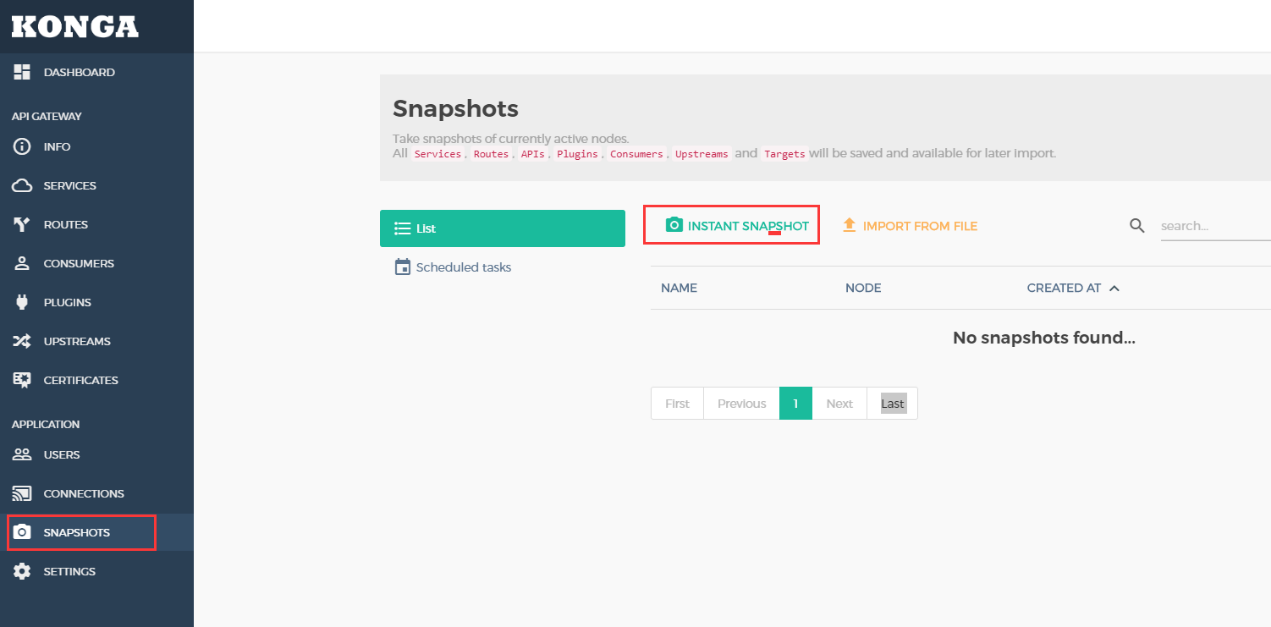
\* client:port--->[kong server<--(cache)-->db]-->api

client---------->auth monitor loggin ssl acl cache rate-limit serverless----------> api

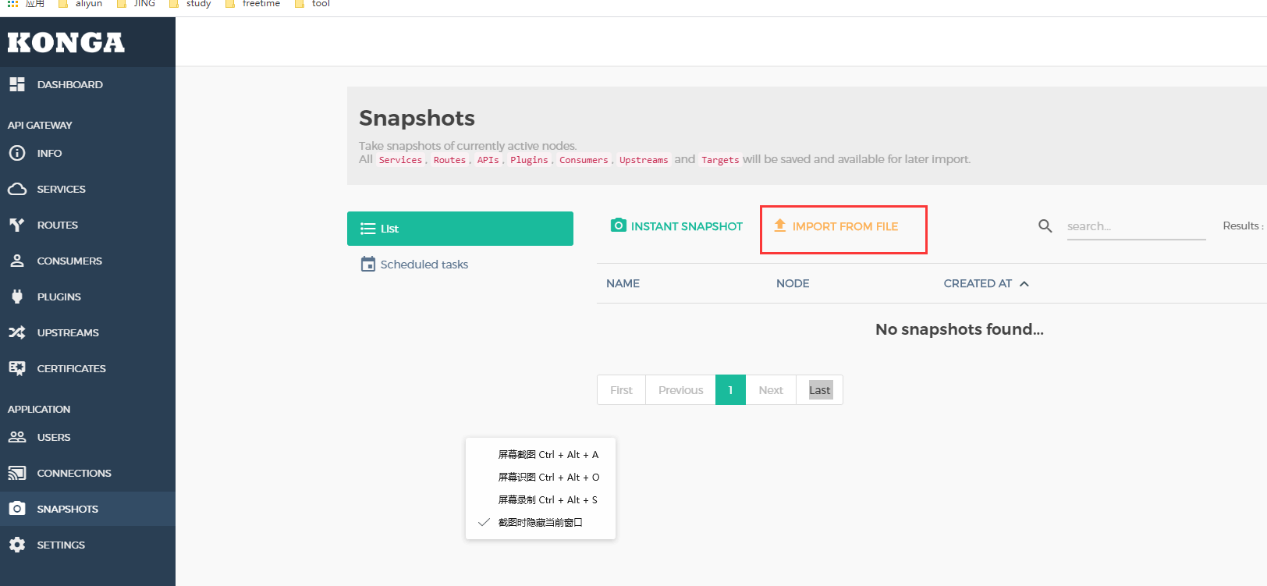
####配置文件备份 与 恢复

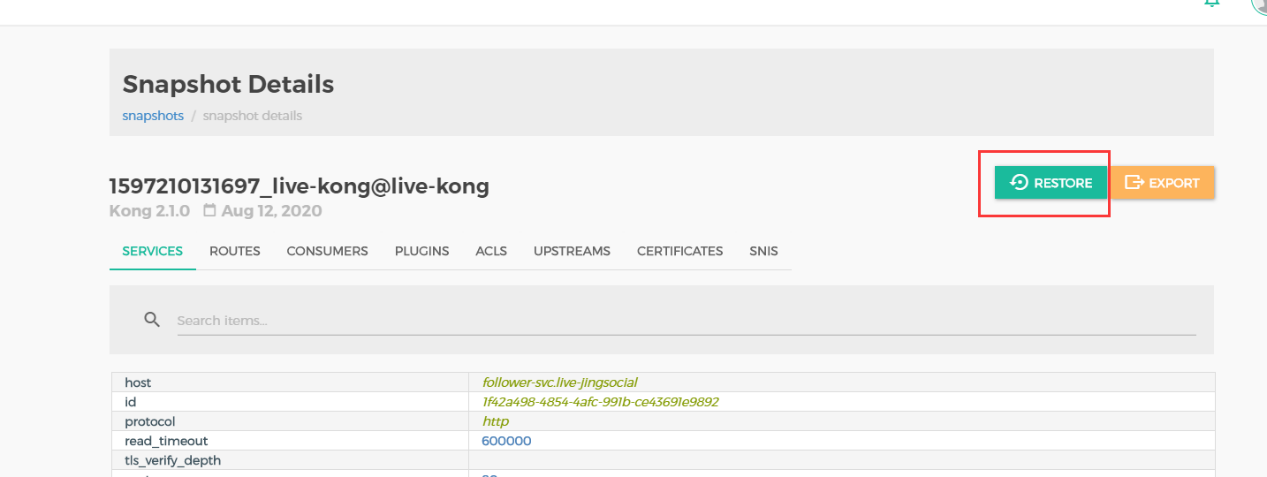
Kong dashboard snapshot: konga Snapshot

备份操作：



恢复操作：





kong deploy by k8s

####################################

search deploy by k8s

kong-init.yml

kong.yml

# 直接使用 k8s 部署 的 pgsql 不太好，资源没有真正隔离，要是研究的话也可以后面研究；这里直接部署的 pgsql 或使用 云 数据库

#### 集群配置：

多个 kong 使用到 同一个数据库，在 k8s 里 部署 多个 副本

（ replicas: 3）；对外服务发现使用 service selector 来绑定到 多个 kong 节点

kong config

####################################

web\_url:

<https://Konga_ip:konga_port>

<https://konga_ip:1337>

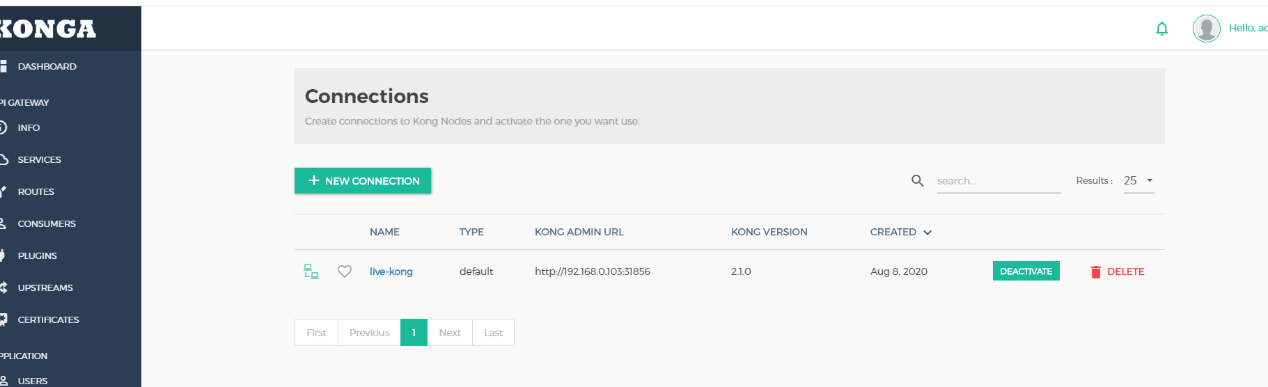
#### new connection and enable

kong\_api\_url:

<http://Kong_ip:kong_port>

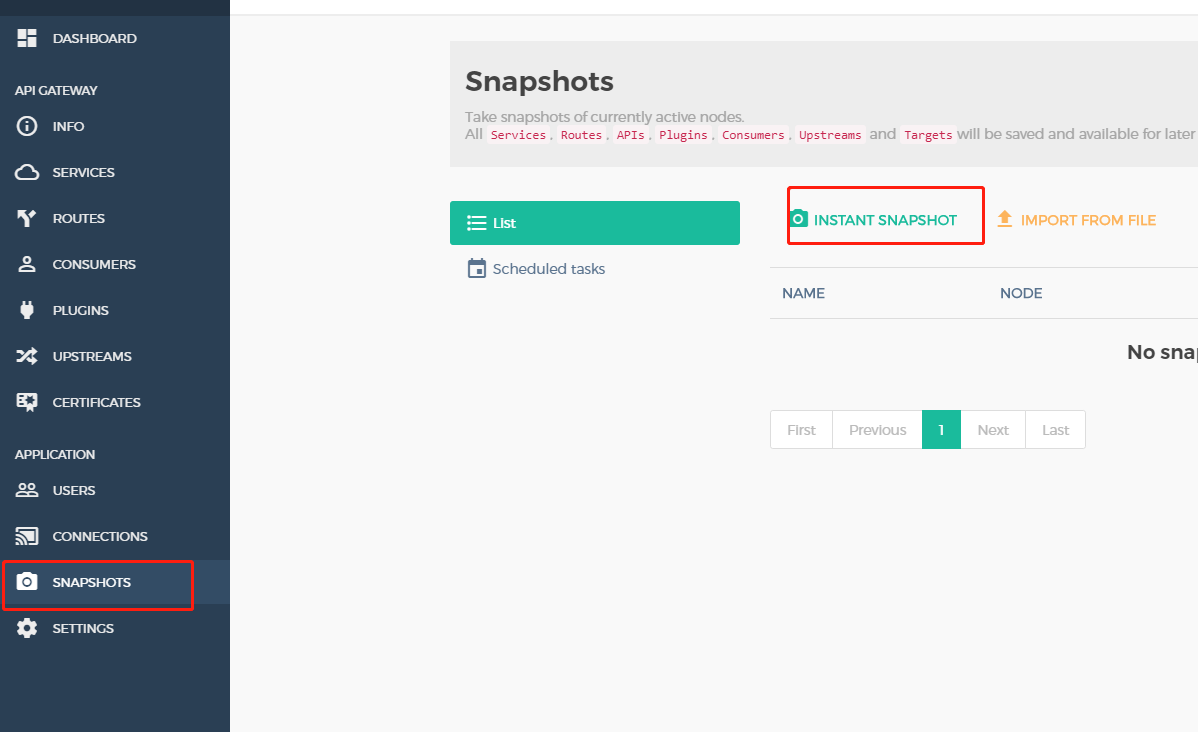
<http://kong_ip:8001>

<http://live-kong:8001> ##这里的 live-kong 是 k8s 里的 svc name

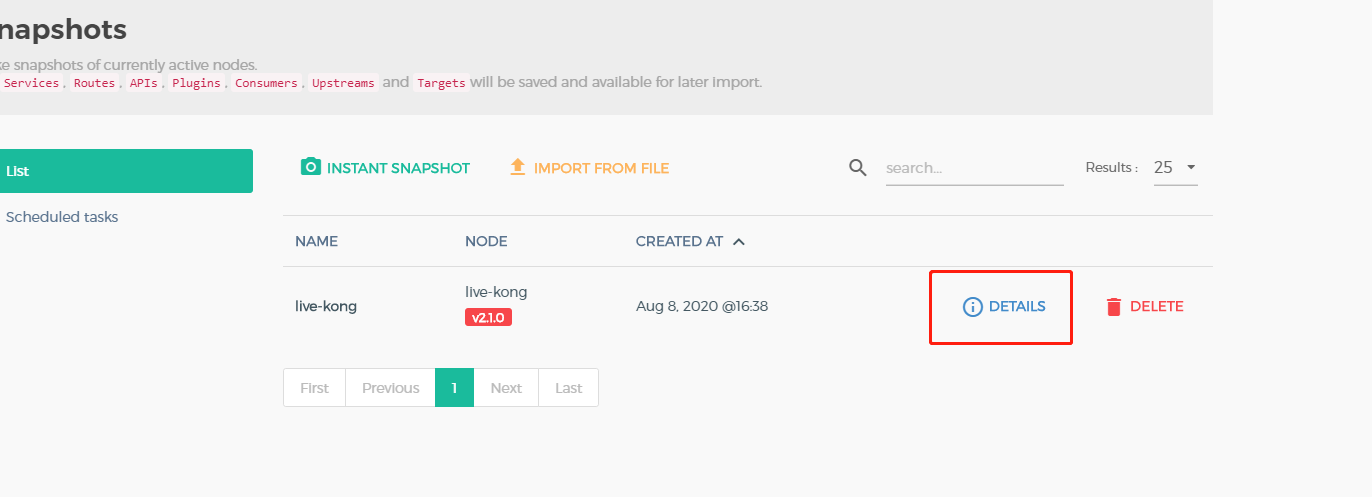


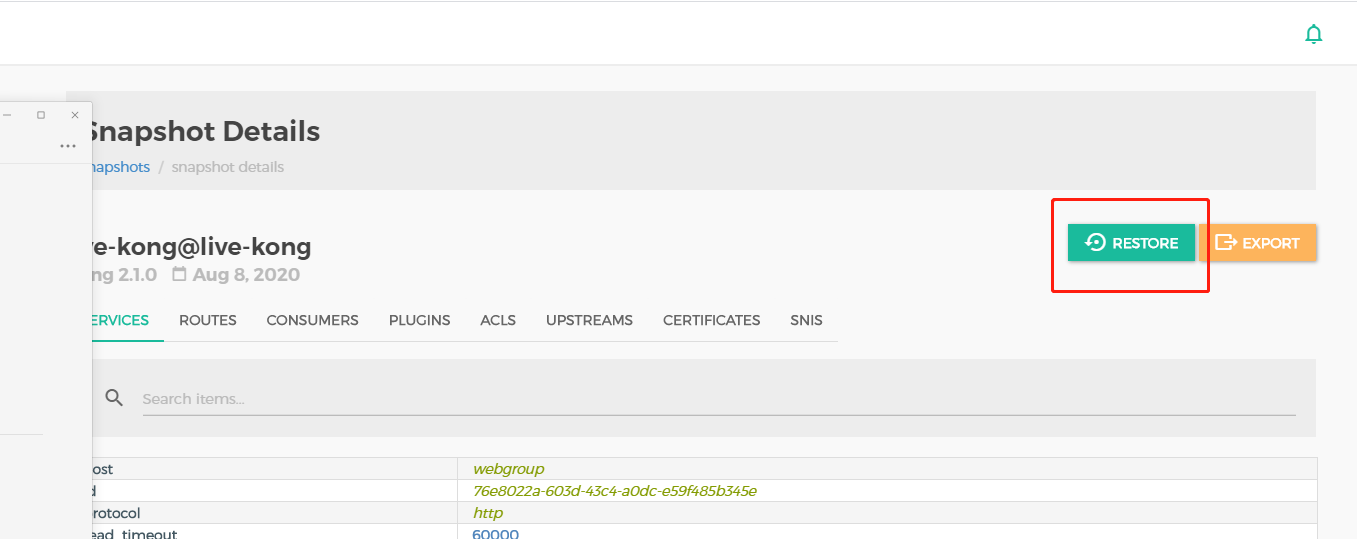
Kong的 配置 是 基于 kong admin api connection 的配置，数据会保存在 kong 对应的数据库；Kong 的 网关路由等配置信息会保存在 kong 对应的数据库，也可在 konga 界面通过 snapshops 导出为 json

#### Snapshop 的备份 和 恢复 如下：



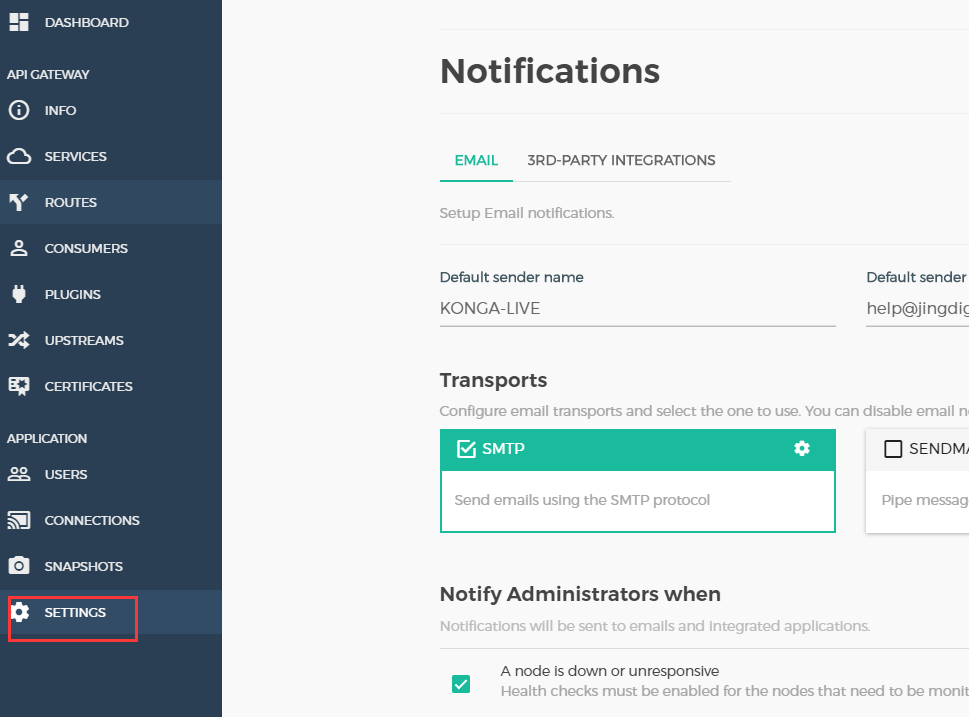
恢复时：先建立 对应 的 connection





####konga 的 用户配置 等数据 会保存在 konga 对应的数据库里

#### konga email 报警配置



#### 配置路由示例

路由顺序：

route ----> service ----> upstream ----> target

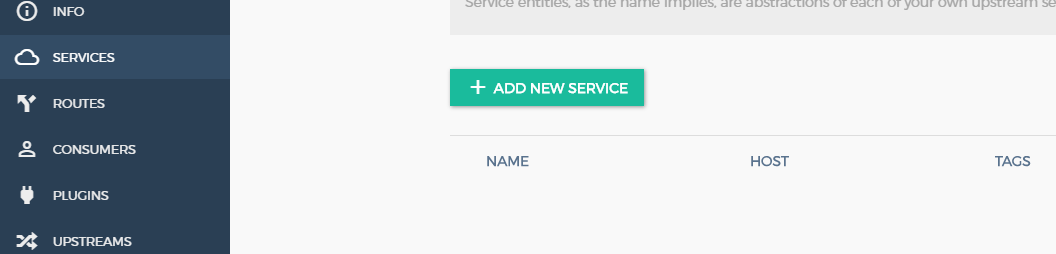
api访问示例：

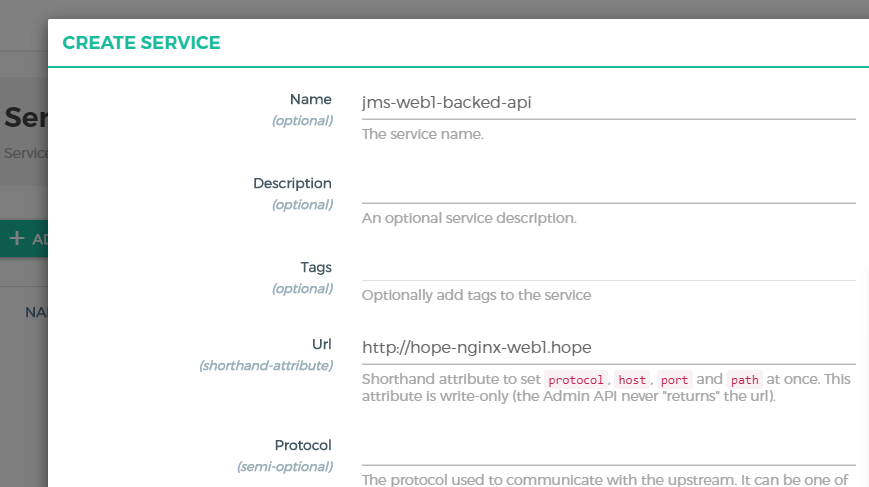
<https://kongip:kong-proxy-port/prefix>

<https://kongip:8443/omni>

上面的 omni 会匹配到不同 的 service , 最后到 upstream 和 target

## #### 配置路由示例 #1 添加 service





Name: service name

<Url:http://hope-nginx-web1.hope>

\\ 这里是用 k8s 里面的 路由名称，hope 是 k8s 里 的 namespace ;hope-nginx-web1 是svc name

其他字段默认

## #### 配置路由示例 #2 添加 service 对应 的 route:

\\ 添加时 设置 好的 变量 要 点击 回车

Name : service 下 对应 的 route name

Hosts: 定义外部访问时 的 路由 host 部分，这里的 host 是 kong 在 K8s 里的 的 svc ip,然后将 proxy 端口 8000 做了 代理后，使用 gwtapi.xxx.com 访问到，这个后面会有 nginx 的 反向代理配置示例

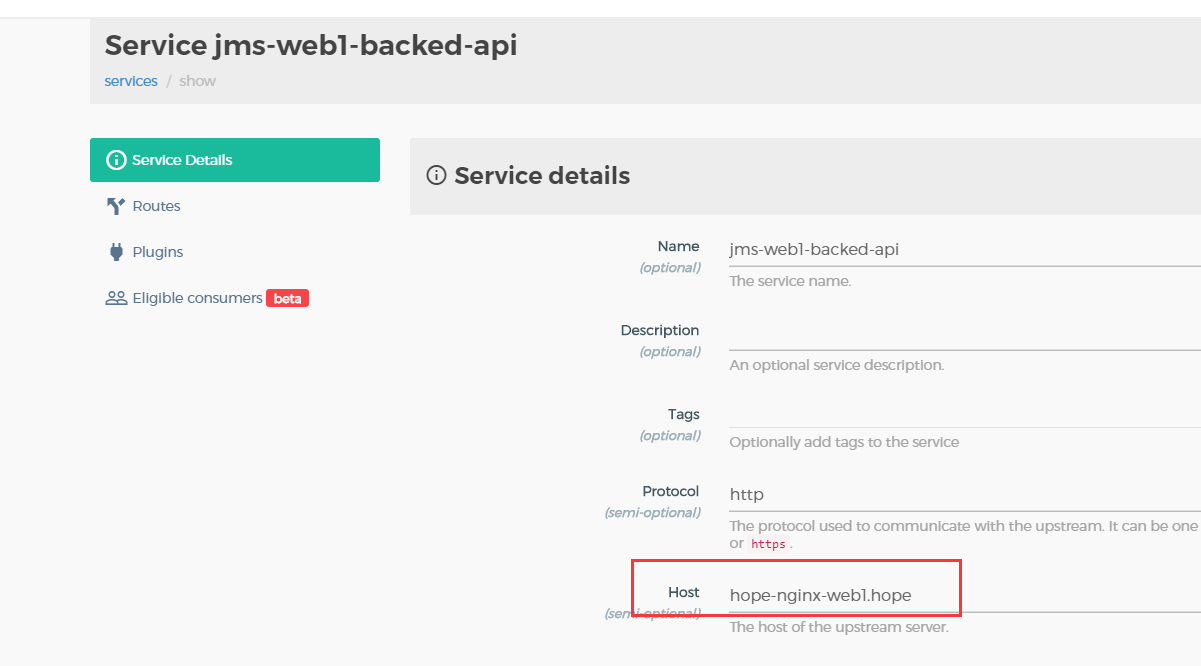
Paths: 路由匹配的prefix

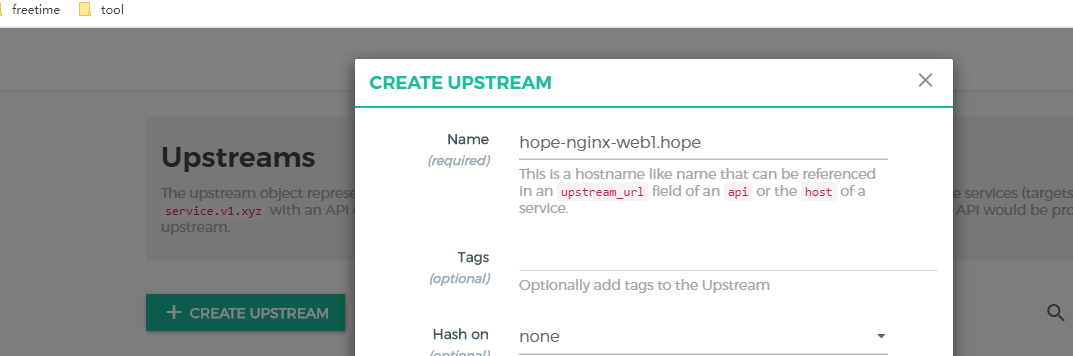
注意上述字段填好要敲回车，其他字段默认

## #### 配置路由示例 #3 配置 upstream 和 对应 的 target

# 先查看 service 的 host 信息， service 配置upstream时， upstream name 要与 service 的 host 值一致

如下，查看到的 service 的 host 值为： hope-nginx-web1.hope; 所以，该 service 对应 的 upstream name 为 hope-nginx-web1.hope





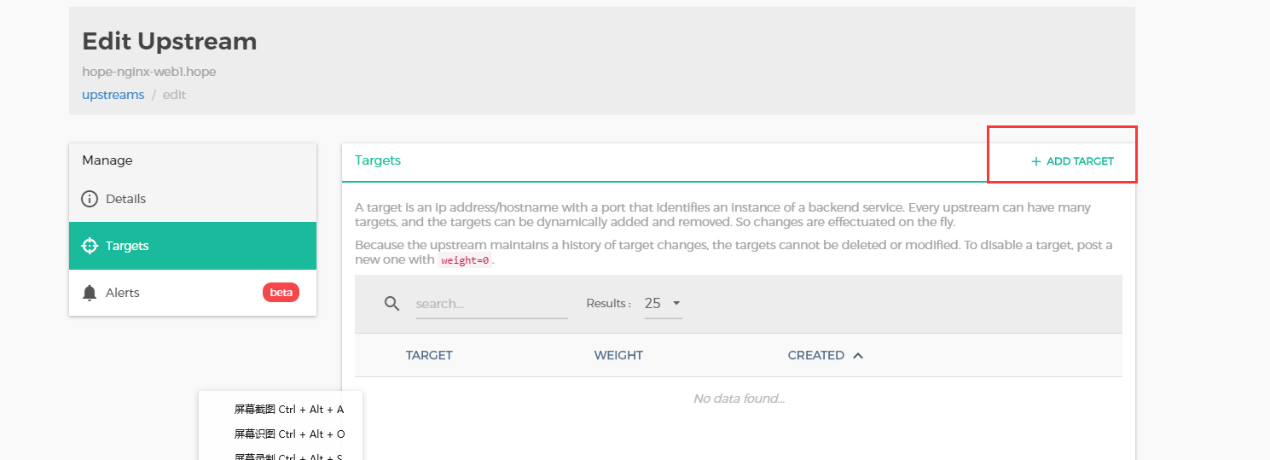
Name: hope-nginx-web1.hope

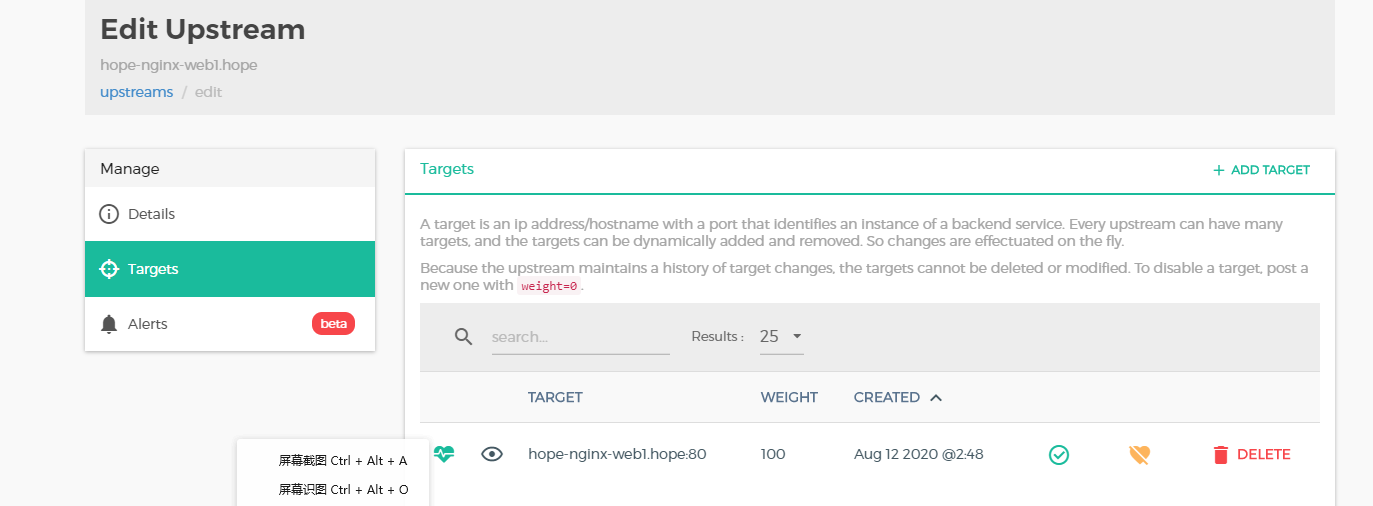
\\ 与 service 的 host 一致；

\\ 其他字段默认

# 再 对 upstream 配置对应的 target

这里 的 target 是 实际要访问的服务，同样，下面的服务名也是k8s 里 的 svc





## #### 配置路由示例 #4 nginx 反向代理配置 ：

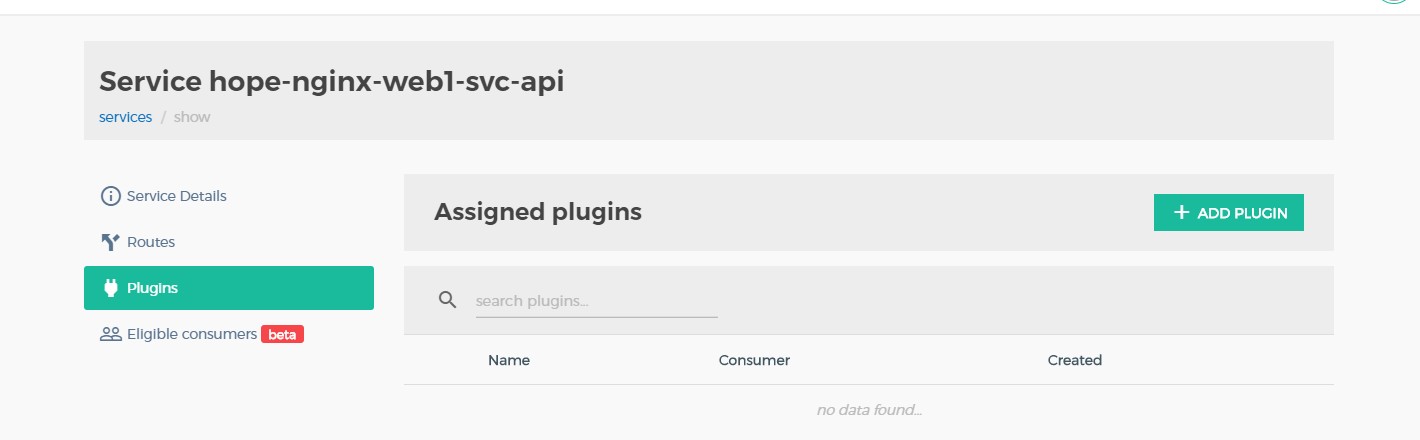
如果 已经 实现 把访问 gwtapi.jingsocial.com 代理到 kong-api server，下面的配置可以不配置

## #### 配置路由示例 #4访问测试：

Curl http://gwtapi.xxx.com/api/web1

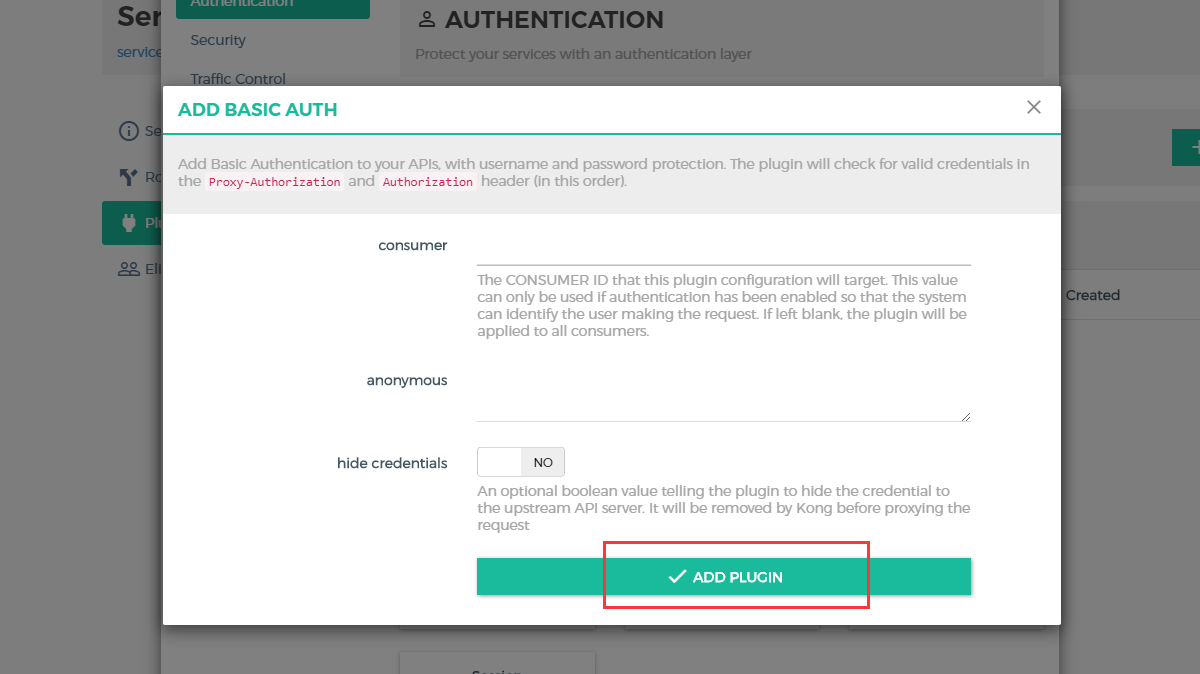
## #### 添加 service 访问认证

## #### 添加 basic auth service 访问认证 #1 添加 service 访问认证

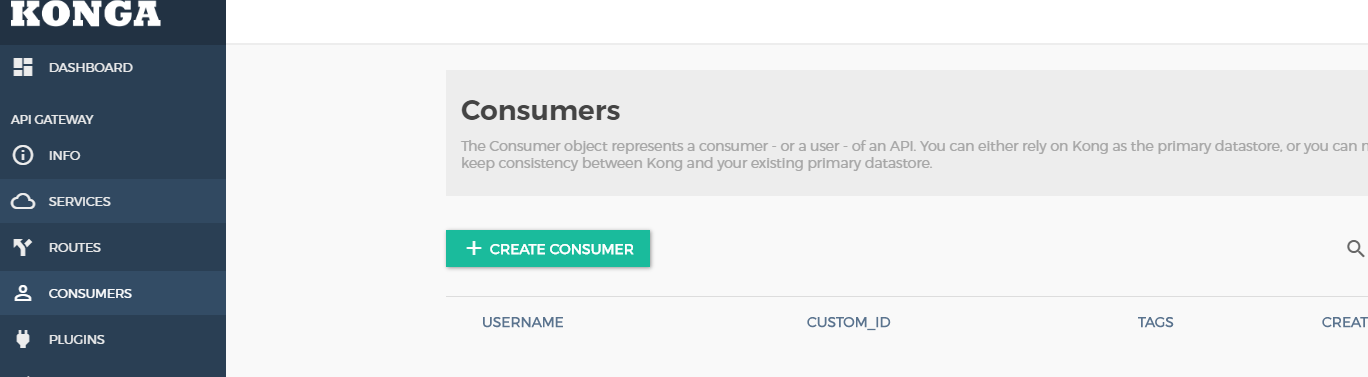


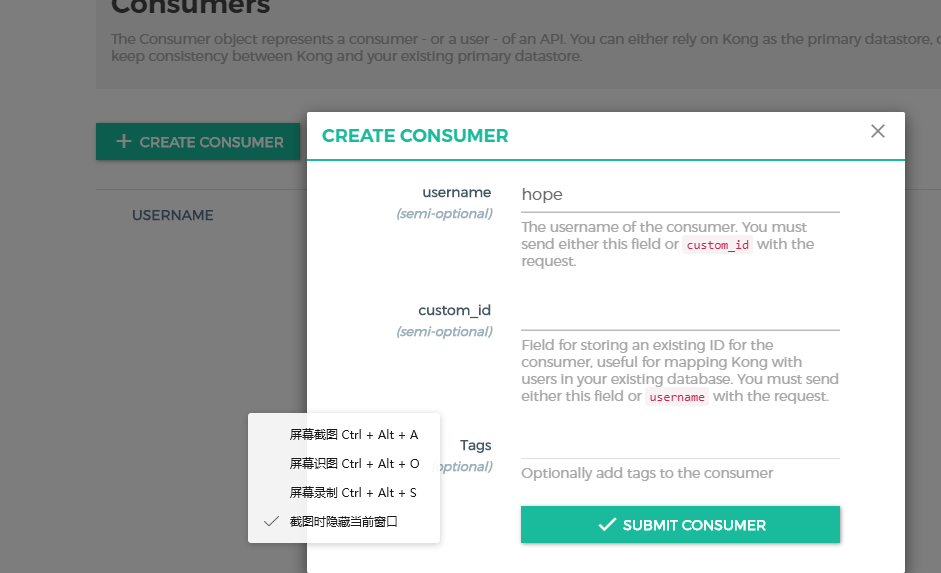
## #### 添加 basic auth service 访问认证 #2 add basic auth:

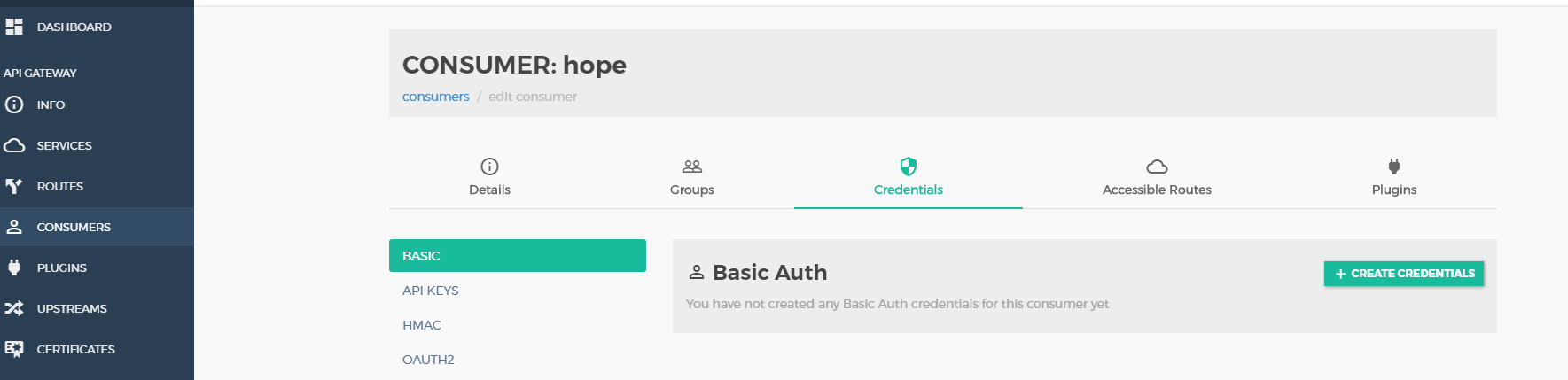
直接 点击 ADD PLUGIN：consumer 字段留空



## #### 添加 basic auth service 访问认证 #3 add consumer and set key:







## #### 添加 basic auth service 访问认证 #4 访问测试

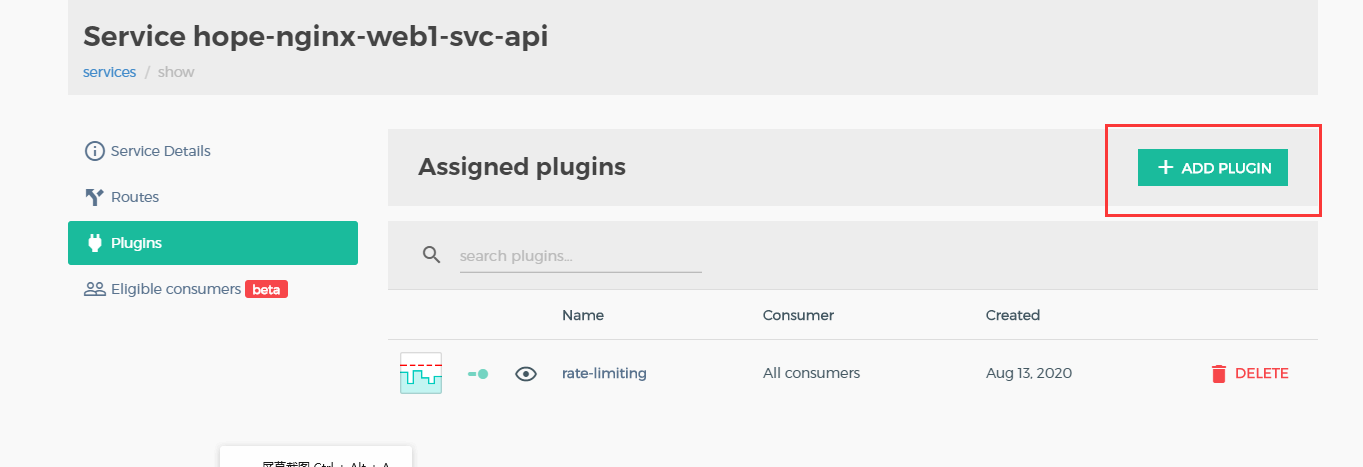
Curl <http://gwtapi.xxx.com/api/web1> -u hope

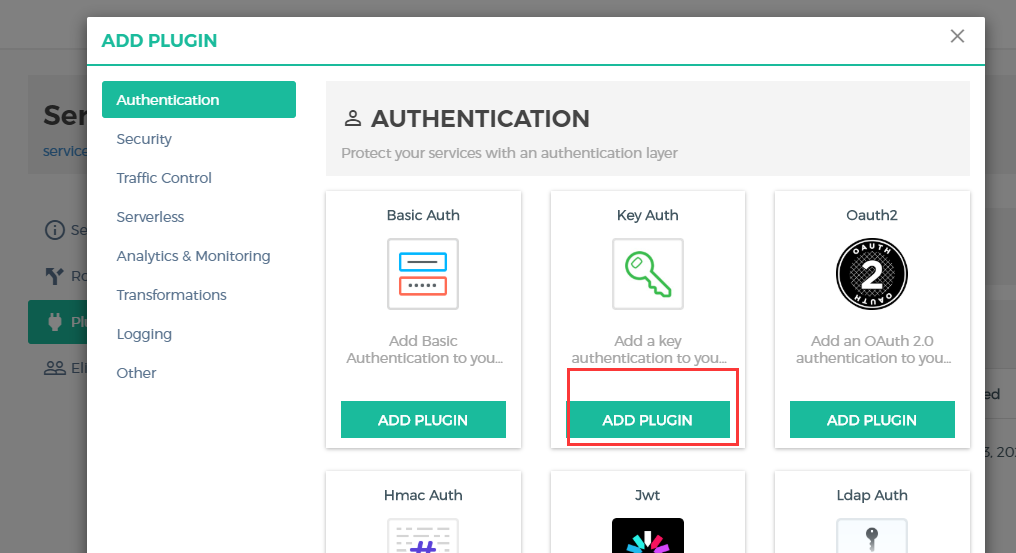
## #### 添加 key auth service 访问认证

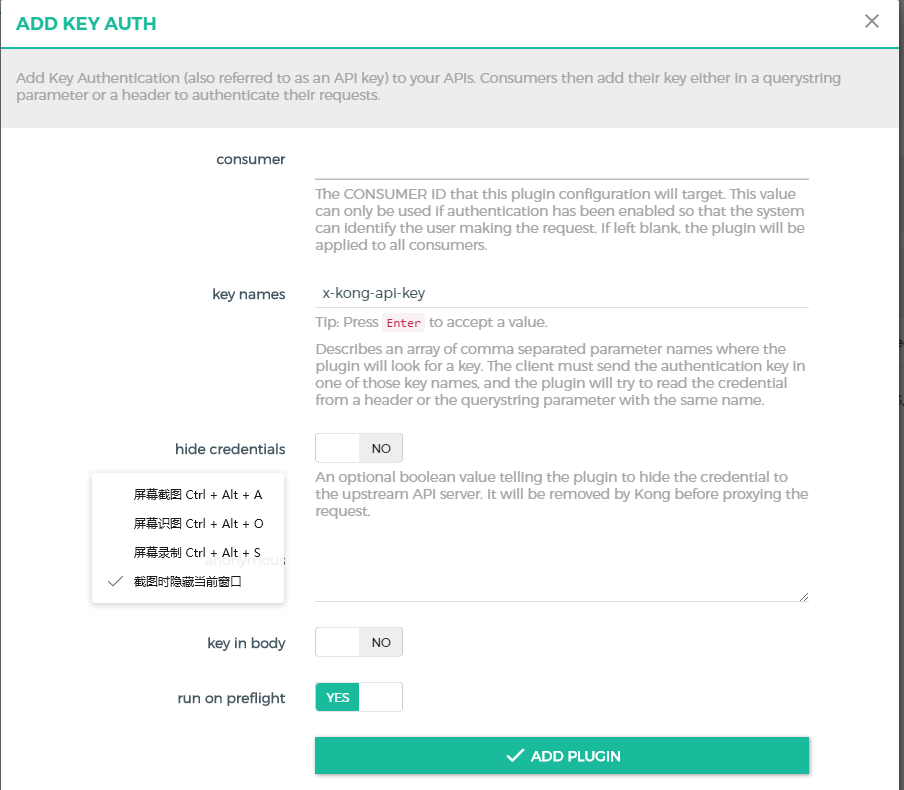
## #### 添加 key auth service 访问认证 #1 给对应 service 添加 key-auth

同样，在给 service 添加 key-auth 时，consumer 对应的字段也留空;

定义对应的key-names: eg : x-kong-api-key







## #### 添加 key auth service 访问认证 #2 创建 consumer 和 key-auth

# create consumer user

curl -i -X POST --url http://{kong\_host}:8001/consumers/ --data "username=apiuser"

curl -i -X POST --url http://192.168.10.242:30847/consumers/ --data "username=hope"

# create consumer key-auth

curl -X POST http://{kong\_host}:8001/consumers/apiuser/key-auth -d ''

curl -X POST http://192.168.10.242:30847/consumers/hope/key-auth -d ''

\\ 记住返回的 key

## #### 添加 key auth service 访问认证 #3 访问测试

curl -i --url http://api-gateway.xxx.com/api/user/xxx --header 'key-name: {consumer-auth-key}'

curl -i --url http://gwtapi.xxx.com/api/web1 --header 'x-kong-api-key: "0NtXPh3tQK2mZRjrs7JAAP15Fl4k0heM'

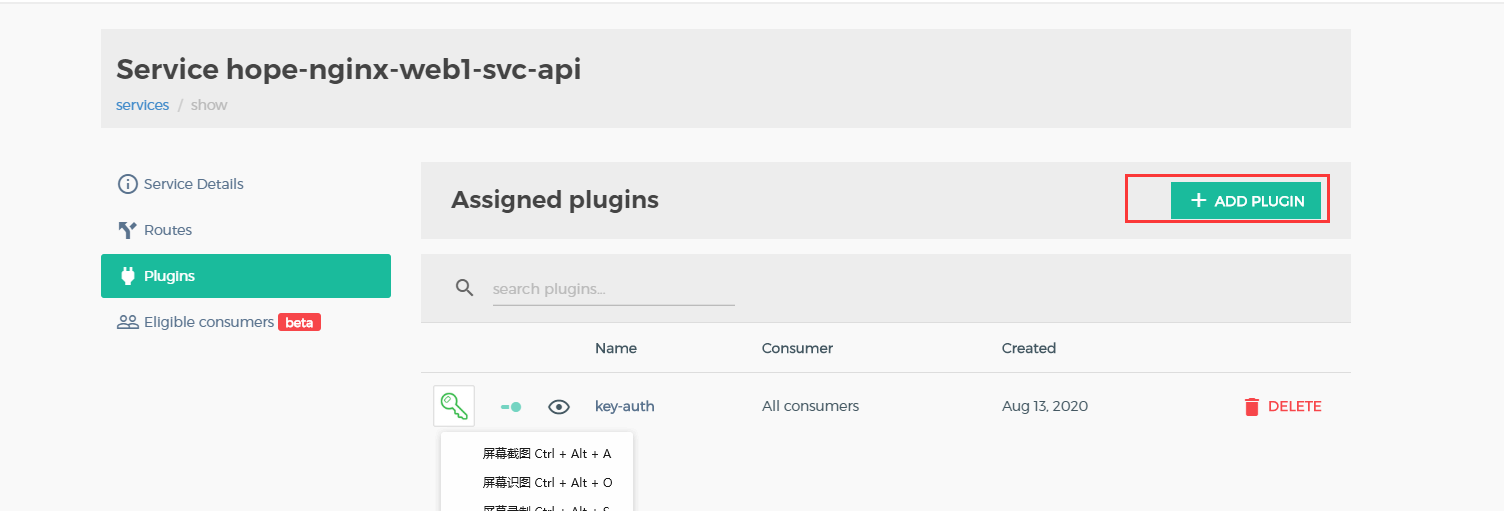
## #### 配置限流示例

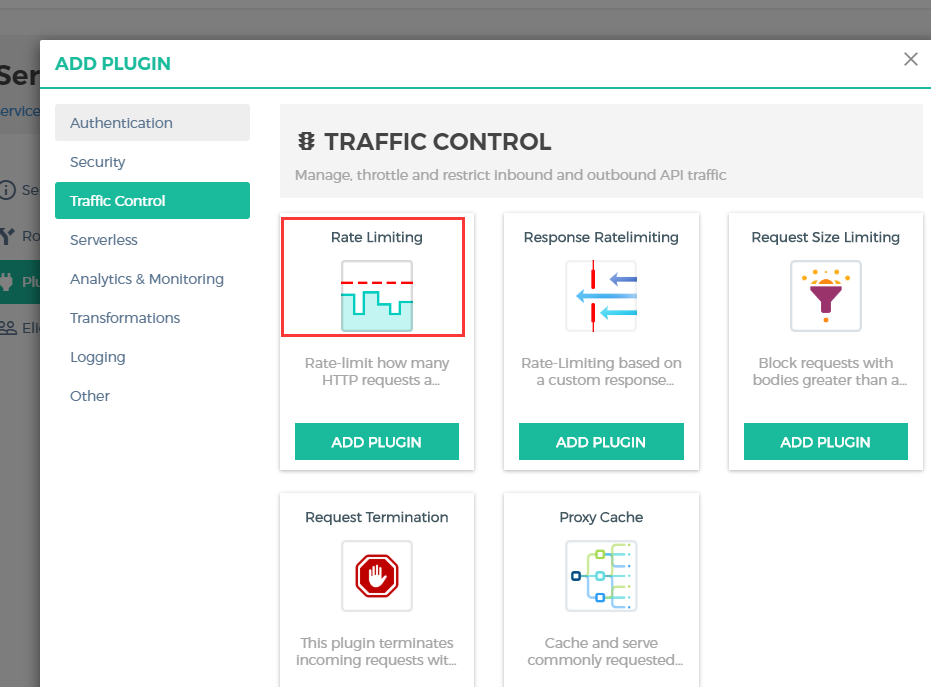
Kong的限流从技术上来讲支持三种，分别是本地限流（local）、数据库限流（cluster）和Redis限流，这三种限流方式采用的限流算法都是计数器法。支持按照秒/分/小时/日/月/年等不同时间单位限流，并且可以组合;

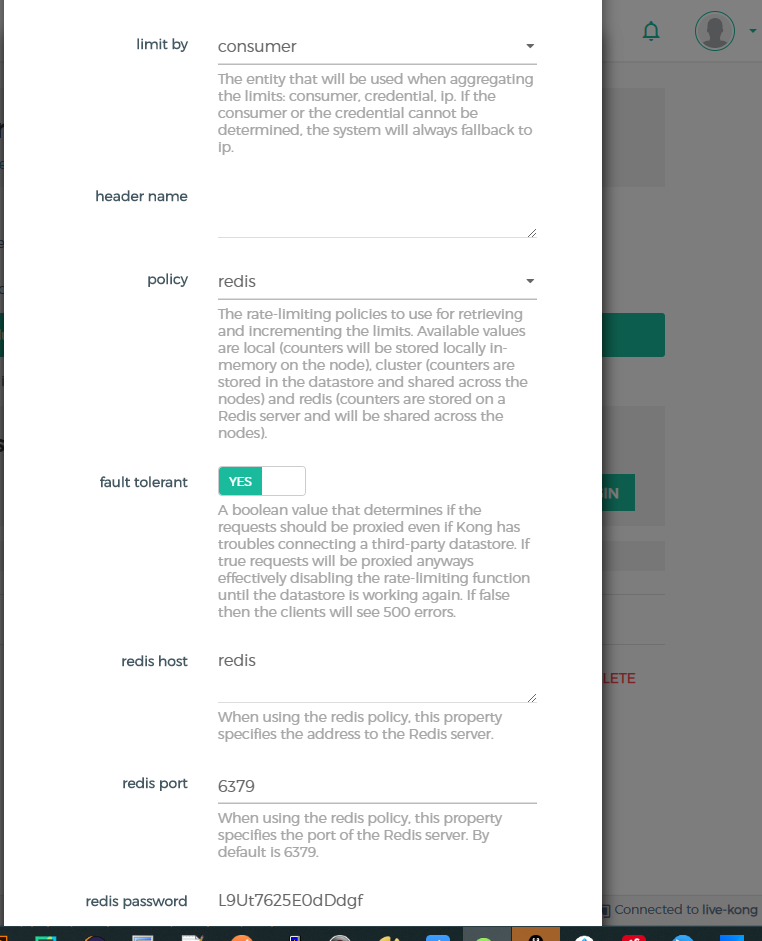
这里是用的redis 来做的限流

## #### 配置限流示例 #1 准备 redis

## #### 配置限流示例 #2 在 对应 service 上配置 限流







上面的限流参数：

Limit by : 分 credential、ip、header、service、consumer;

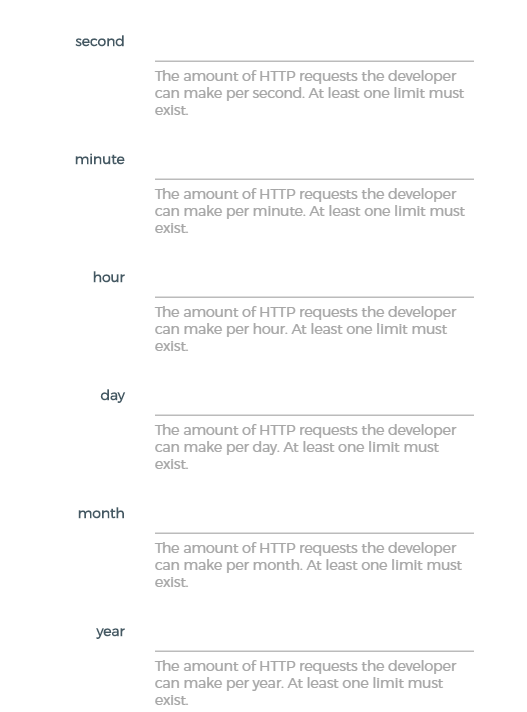
这里 用 consumer

Police : 分 local、cluster、redis

这里用的 Redis

## #### 配置限流示例 #3 限流维度

支持不同时间维度的限制



## #### 配置限流示例 #4 限流测试 5次/min

