



链滴

# 在 K8S 上使用 kong 做服务网关

作者: [Gakkiyomi2019](#)

原文链接: <https://ld246.com/article/1617939431355>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)



## 在K8S上使用kong做服务网关

Kong是一个开源API网关和微服务管理层。Kong基于Nginx和lua-nginx-module（特别是OpenResty），可插拔的体系结构使其灵活而强大。

网关的最主要的作用就是路由，各个服务的端口多且杂管理起来苦不堪言，所以有了网关来管理这些任务的转发，最出名的就是nginx，而今天要介绍的这个网关也是基于nginx的，性能非常强大，而且界面简单好用。下面我会来介绍在k8s上安装kong并且将一个带端口服务路由成一个不需要端口的url路径。

本文主要关注的是环境的搭建，如果想要学习kong的api请移步官方文档 [mile](#)

### 准备镜像

```
[root@paas-51 kong]# docker images | grep kong
hub.sky-cloud.net/k8s/konga      0.14.7      e3251e5022b      17 months ago      398MB
hub.sky-cloud.net/k8s/kong       1.0.3      bebc5704407b     20 months ago      404MB
[root@paas-51 kong]#
```

### kong

使用官方镜像即可

`docker pull kong`

### konga

`docker pull pantsel/konga`

### 准备yaml文件

konga是kong的dashboard, 如果觉得使用kong的api麻烦的话, 可以使用konga来管理方便快捷。

## kong

### cm.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kong-config
  namespace: kong
  labels:
    addonmanager.kubernetes.io/mode: Reconcile
data:
  nginx_kong.lua: |
    return [[
      charset UTF-8;

      > if anonymous_reports then
        ${{SYSLOG_REPORTS}}
      > end

      error_log ${{PROXY_ERROR_LOG}} ${{LOG_LEVEL}};

      > if nginx_optimizations then
      >-- send_timeout 60s;      # default value
      >-- keepalive_timeout 75s; # default value
      >-- client_body_timeout 60s; # default value
      >-- client_header_timeout 60s; # default value
      >-- tcp_nopush on;        # disabled until benchmarked
      >-- proxy_buffer_size 128k; # disabled until benchmarked
      >-- proxy_buffers 4 256k; # disabled until benchmarked
      >-- proxy_busy_buffers_size 256k; # disabled until benchmarked
      >-- reset_timedout_connection on; # disabled until benchmarked
      > end

      client_max_body_size ${{CLIENT_MAX_BODY_SIZE}};
      proxy_ssl_server_name on;
      underscores_in_headers on;

      lua_package_path '{{LUA_PACKAGE_PATH}};';
      lua_package_cpath '{{LUA_PACKAGE_CPATH}};';
      lua_socket_pool_size ${{LUA_SOCKET_POOL_SIZE}};
      lua_max_running_timers 4096;
      lua_max_pending_timers 16384;
      lua_shared_dict kong 5m;
      lua_shared_dict kong_db_cache ${{MEM_CACHE_SIZE}};
      lua_shared_dict kong_db_cache_miss 12m;
      lua_shared_dict kong_locks 8m;
      lua_shared_dict kong_process_events 5m;
      lua_shared_dict kong_cluster_events 5m;
      lua_shared_dict kong_healthchecks 5m;
      lua_shared_dict kong_rate_limiting_counters 12m;
```

```

> if database == "cassandra" then
lua_shared_dict kong_cassandra 5m;
> end
lua_socket_log_errors off;
> if lua_ssl_trusted_certificate then
lua_ssl_trusted_certificate '${LUA_SSL_TRUSTED_CERTIFICATE}';
> end
lua_ssl_verify_depth ${LUA_SSL_VERIFY_DEPTH};

# injected nginx_http_* directives
> for _, el in ipairs(nginx_http_directives) do
$(el.name) $(el.value);
> end

init_by_lua_block {
    Kong = require 'kong'
    Kong.init()
}

init_worker_by_lua_block {
    Kong.init_worker()
}

> if #proxy_listeners > 0 then
upstream kong_upstream {
    server 0.0.0.1;
    balancer_by_lua_block {
        Kong.balancer()
    }
}
> if upstream_keepalive > 0 then
keepalive ${UPSTREAM_KEEPALIVE};
> end
}

server {
    server_name kong;
    > for i = 1, #proxy_listeners do
listen $(proxy_listeners[i].listener);
    > end
    error_page 400 404 408 411 412 413 414 417 494 /kong_error_handler;
    error_page 500 502 503 504 /kong_error_handler;

    access_log ${PROXY_ACCESS_LOG};
    error_log ${PROXY_ERROR_LOG} ${LOG_LEVEL};

    client_body_buffer_size ${CLIENT_BODY_BUFFER_SIZE};

    > if proxy_ssl_enabled then
ssl_certificate ${SSL_CERT};
ssl_certificate_key ${SSL_CERT_KEY};
ssl_protocols TLSv1.1 TLSv1.2 TLSv1.3;
ssl_certificate_by_lua_block {
    Kong.ssl_certificate()
}
}
}

```

```

}

ssl_session_cache shared:SSL:10m;
ssl_session_timeout 10m;
ssl_prefer_server_ciphers on;
ssl_ciphers ${{SSL_CIPHERS}};
> end

> if client_ssl then
    proxy_ssl_certificate ${{CLIENT_SSL_CERT}};
    proxy_ssl_certificate_key ${{CLIENT_SSL_CERT_KEY}};
> end

    real_ip_header    ${{REAL_IP_HEADER}};
    real_ip_recursive ${{REAL_IP_RECURSIVE}};
> for i = 1, #trusted_ips do
    set_real_ip_from $(trusted_ips[i]);
> end

    # injected nginx_proxy_* directives
> for _, el in ipairs(nginx_proxy_directives) do
    $(el.name) $(el.value);
> end

location / {
    default_type            "";

    set $ctx_ref            "";
    set $upstream_host     "${upstream_host}";
    set $upstream_upgrade  "${upstream_upgrade}";
    set $upstream_connection "${upstream_connection}";
    set $upstream_scheme    "${upstream_scheme}";
    set $upstream_uri       "${upstream_uri}";
    set $upstream_x_forwarded_for "${upstream_x_forwarded_for}";
    set $upstream_x_forwarded_proto "${upstream_x_forwarded_proto}";
    set $upstream_x_forwarded_host "${upstream_x_forwarded_host}";
    set $upstream_x_forwarded_port "${upstream_x_forwarded_port}";

    rewrite_by_lua_block {
        Kong.rewrite()
    }

    access_by_lua_block {
        Kong.access()
    }

    proxy_http_version 1.1;
    proxy_set_header    Host            $upstream_host;
    proxy_set_header    Upgrade         $upstream_upgrade;
    proxy_set_header    Connection      $upstream_connection;
    proxy_set_header    X-Forwarded-For $upstream_x_forwarded_for;
    proxy_set_header    X-Forwarded-Proto $upstream_x_forwarded_proto;
    proxy_set_header    X-Forwarded-Host $upstream_x_forwarded_host;
    proxy_set_header    X-Forwarded-Port $upstream_x_forwarded_port;

```

```

proxy_set_header X-Real-IP      $remote_addr;
proxy_pass_header Server;
proxy_pass_header Date;
proxy_ssl_name    $upstream_host;
proxy_pass        $upstream_scheme://kong_upstream$upstream_uri;

header_filter_by_lua_block {
    Kong.header_filter()
}

body_filter_by_lua_block {
    Kong.body_filter()
}

log_by_lua_block {
    Kong.log()
}
}

location = /kong_error_handler {
    internal;
    uninitialized_variable_warn off;

    content_by_lua_block {
        Kong.handle_error()
    }

    header_filter_by_lua_block {
        Kong.header_filter()
    }

    body_filter_by_lua_block {
        Kong.body_filter()
    }

    log_by_lua_block {
        Kong.log()
    }
}
}
> end

> if #admin_listeners > 0 then
server {
    server_name kong_admin;
> for i = 1, #admin_listeners do
    listen $(admin_listeners[i].listener);
> end

    access_log ${{ADMIN_ACCESS_LOG}};
    error_log ${{ADMIN_ERROR_LOG}} ${{LOG_LEVEL}};

    client_max_body_size 10m;
    client_body_buffer_size 10m;

```

```

> if admin_ssl_enabled then
  ssl_certificate ${ADMIN_SSL_CERT}};
  ssl_certificate_key ${ADMIN_SSL_CERT_KEY}};
  ssl_protocols TLSv1.1 TLSv1.2 TLSv1.3;

  ssl_session_cache shared:SSL:10m;
  ssl_session_timeout 10m;
  ssl_prefer_server_ciphers on;
  ssl_ciphers ${SSL_CIPHERS}};
> end

# injected nginx_admin_* directives
> for _, el in ipairs(nginx_admin_directives) do
  ${el.name} ${el.value};
> end

location / {
  default_type application/json;
  content_by_lua_block {
    Kong.serve_admin_api()
  }
}

location /nginx_status {
  internal;
  access_log off;
  stub_status;
}

location /robots.txt {
  return 200 'User-agent: *\nDisallow: /';
}
}
> end
]]

```

## kong.yaml

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: kong
  namespace: kong
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kong
  template:
    metadata:
      labels:
        name: kong
        app: kong

```

```

spec:
  tolerations:
  - key: "node.kubernetes.io/unreachable"
    operator: "Exists"
    effect: "NoExecute"
    tolerationSeconds: 60
  - key: "node.kubernetes.io/not-ready"
    operator: "Exists"
    effect: "NoExecute"
    tolerationSeconds: 60
  terminationGracePeriodSeconds: 0          # 异常立即删除
  initContainers:
  # hack to verify that the DB is up to date or not
  # TODO remove this for Kong >= 0.15.0
  - name: wait-for-migrations
    image: hub.sky-cloud.net/k8s/kong:1.0.3
    imagePullPolicy: IfNotPresent
    command: [ "kong", "migrations", "list" ]
    env:
    - name: KONG_ADMIN_LISTEN
      value: 'off'
    - name: KONG_PROXY_LISTEN
      value: 'off'
    - name: KONG_PROXY_ACCESS_LOG
      value: "/dev/stdout"
    - name: KONG_ADMIN_ACCESS_LOG
      value: "/dev/stdout"
    - name: KONG_PROXY_ERROR_LOG
      value: "/dev/stderr"
    - name: KONG_ADMIN_ERROR_LOG
      value: "/dev/stderr"
    - name: KONG_PG_HOST
      value: postgres-0.postgres.postgres
    - name: KONG_PG_PORT
      value: '5432'
    - name: KONG_PG_DATABASE
      value: kong
    - name: KONG_PG_USER
      value: kong
    - name: KONG_PG_PASSWORD
      value: kong
    - name: KONG_PLUGINS
      value: jwt-keycloak,oidc,acl,aws-lambda,azure-functions,basic-auth,bot-detection,correlation-id,cors,datadog,file-log,hmac-auth,http-log,ip-restriction,jwt,key-auth,ldap-auth,loggly,oauth2,post-function,pre-function,prometheus,rate-limiting,request-size-limiting,request-termination,request-transformer,statsd,syslog,tcp-log,udp-log,zipkin
    - name: TZ
      value: Asia/Shanghai
  resources:
    limits:
      memory: 1024Mi
    requests:
      memory: 1024Mi
  volumeMounts:

```



```
- name: tz
  mountPath: /etc/localtime
hostAliases:
- ip: "192.168.1.225"
  hostnames:
  - "nginx-service"
  - "nap.sky-cloud.net"
containers:
- name: kong-proxy
  image: hub.sky-cloud.net/k8s/kong:1.0.3
  imagePullPolicy: IfNotPresent
  env:
  - name: KONG_PG_HOST
    value: postgres-0.postgres.postgres
  - name: KONG_PG_PORT
    value: '5432'
  - name: KONG_PG_DATABASE
    value: kong
  - name: KONG_PG_USER
    value: kong
  - name: KONG_PG_PASSWORD
    value: kong
  - name: KONG_PROXY_ACCESS_LOG
    value: "/dev/stdout"
  - name: KONG_PROXY_ERROR_LOG
    value: "/dev/stderr"
  - name: KONG_ADMIN_LISTEN
    value: '0.0.0.0:8001, 0.0.0.0:8444 ssl'
  - name: KONG_PLUGINS
    value: jwt-keycloak,oidc,acl,aws-lambda,azure-functions,basic-auth,bot-detection,correlation-id,cors,datadog,file-log,hmac-auth,http-log,ip-restriction,jwt,key-auth,ldap-auth,loggly,oauth2,post-function,pre-function,prometheus,rate-limiting,request-size-limiting,request-termination,request-transformer,statsd,syslog,tcp-log,udp-log,zipkin
  ports:
  - name: proxy
    containerPort: 8000
    protocol: TCP
  - name: proxy-ssl
    containerPort: 8443
    protocol: TCP
  - name: kong-admin
    containerPort: 8001
livenessProbe:
  failureThreshold: 3
  httpGet:
    path: /status
    port: 8001
    scheme: HTTP
  initialDelaySeconds: 30
  periodSeconds: 10
  successThreshold: 1
  timeoutSeconds: 1
readinessProbe:
  failureThreshold: 3
```

```
  httpGet:
    path: /status
    port: 8001
    scheme: HTTP
  periodSeconds: 10
  successThreshold: 1
  timeoutSeconds: 1
  volumeMounts:
  - name: tz
    mountPath: /etc/localtime
  volumeMounts:
  - name: nginx
    subPath: nginx_kong.lua
    mountPath: /usr/share/lua/5.1/kong/templates/nginx_kong.lua
  volumes:
  - name: tz
    hostPath:
      path: /etc/localtime
  - name: nginx
    configMap:
      name: kong-config
    items:
    - key: nginx_kong.lua
      path: nginx_kong.lua
```

## svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: kong
  namespace: kong
spec:
  externalIPs:      # 暴露Service到外部IP
  - 192.168.30.51   # IP
  ports:
  - name: kong-proxy
    port: 8000
    targetPort: 8000
    protocol: TCP
  - name: kong-proxy-ssl
    port: 8443
    targetPort: 8443
    protocol: TCP
  - name: kong-admin
    port: 8001
    targetPort: 8001
    protocol: TCP
  selector:
    app: kong
```

## konga

```
---
apiVersion: v1
kind: Service
metadata:
  name: konga
  namespace: kong
spec:
  externalIPs:      # 暴露Service到外部IP
  - 192.168.30.51   # IP
  ports:
  - name: konga
    port: 1337
    targetPort: 1337
    protocol: TCP
  selector:
    app: konga
---
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: konga
  namespace: kong
spec:
  replicas: 1
  selector:
    matchLabels:
      app: konga
  template:
    metadata:
      labels:
        name: konga
        app: konga
    spec:
      tolerations:
      - key: "node.kubernetes.io/unreachable"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 60
      - key: "node.kubernetes.io/not-ready"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 60
      terminationGracePeriodSeconds: 0      # 异常立即删除
    containers:
    - name: konga
      image: hub.sky-cloud.net/k8s/konga:0.14.7
      imagePullPolicy: IfNotPresent
      env:
      - name: DB_ADAPTER
        value: "postgres"
      - name: DB_HOST
        value: "postgres-0.postgres.postgres" //配置上你数据库的地址
      - name: DB_PORT
```

```

value: "5432"
- name: DB_USER
value: "konga"
- name: DB_PASSWORD
value: "r00tme"
- name: DB_DATABASE
value: "konga"
- name: NODE_ENV
value: "production"
ports:
- name: konga
containerPort: 1337
protocol: TCP
volumeMounts:
- name: tz
mountPath: /etc/localtime
volumes:
- name: tz
hostPath:
path: /etc/localtime

```

直接启动 `kubectl apply -f`.

查看启动情况

NAME	READY	STATUS	RESTARTS	AGE
kong-7554fd84f9-64hcc	1/1	Running	23	28d
konga-67cd5887fc-9nlms	1/1	Running	0	25h
konga-67cd5887fc-9nlms	1/1	Running	0	28d

[root@paas-51 kong]#

## 打开konga

The screenshot shows the Kong dashboard interface. The left sidebar contains navigation options: DASHBOARD, API GATEWAY, INFO, SERVICES, ROUTES, CONSUMERS, PLUGINS, UPSTREAMS, CERTIFICATES, APPLICATION, USERS, CONNECTIONS, SNAPSHOTS, and SETTINGS. The main content area displays several panels:

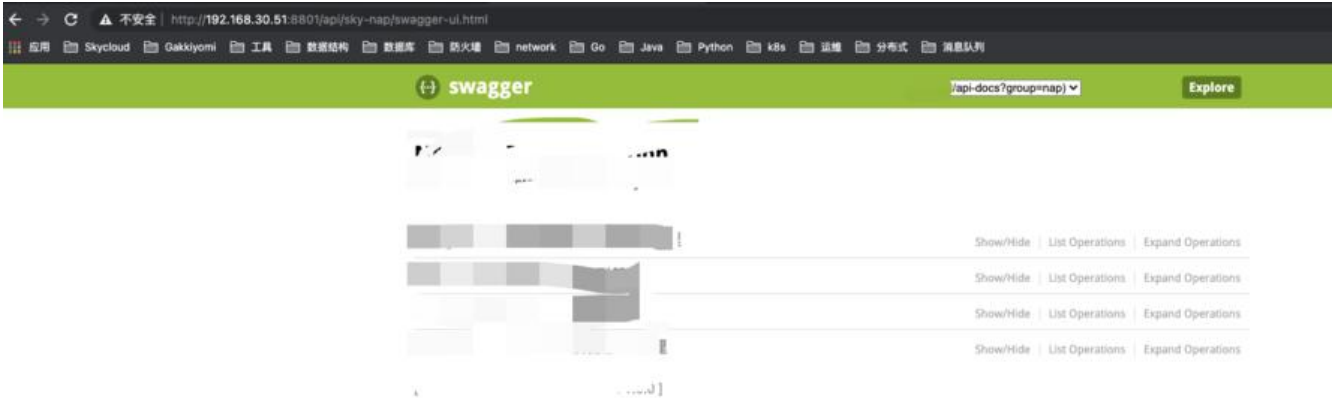
- CONNECTIONS:** Shows a summary of connection states: ACTIVE (3), READING (0), WRITING (2), WAITING (1), ACCEPTED (131K+), and HANDLED (131K+). Total Requests: 141K+.
- NODE INFO:** Displays host details for 'kong-7554fd84f9-64hcc', including Tag Line ('Welcome to kong'), Version (1.0.3), LUA Version (LuaJIT 2.1.0-beta3), and Admin listen address ([\*0.0.0.0:8001,\*0.0.0.0:8444 ssl]).
- TIMERS:** A graph showing 'Pending' and 'Running' request counts over time.
- DATASTORE INFO:** Shows database configuration: DBMS (postgres), Host (postgres-0.postgres.postgres), Database (kong), User (kong), and Port (5432).
- PLUGINS:** A list of installed plugins including: cors, request-size-limiting, fault-detection, correlation-id, jwt-keycloak, jwt, cors, request-termination, basic-auth, key-auth, ldap-auth, syslog, http-log, stand, hmac-auth, request-transformer, datadog, tcp-keep-alive, zipkin, post-function, rate-limiting, api, prometheus, pre-function, azure-functions, ldap-sig, file-log, aws-lambda, loggly, @-resonance, and @auth-0.

# 路由服务

我们启动了一个8801端口的后端服务

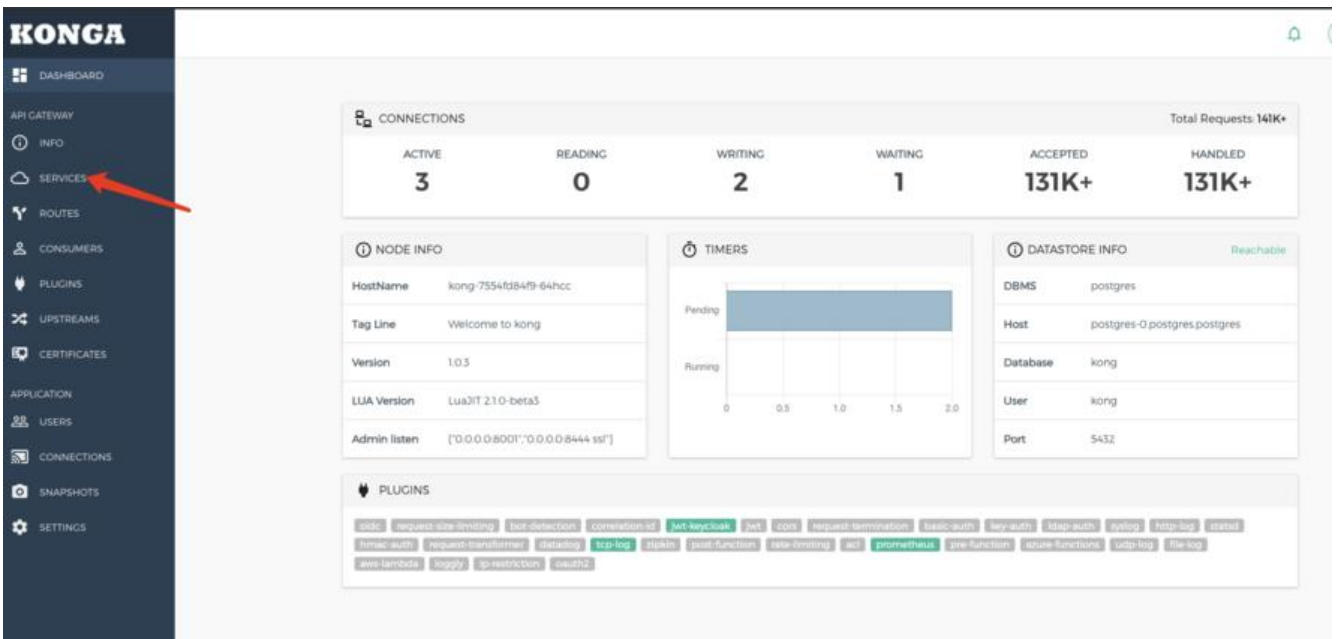
在浏览器输入<http://192.168.30.51:8801/api/sky-nap/swagger-ui.html>

我们该如何使用kong来路由他呢?



## 第一步

点击Service

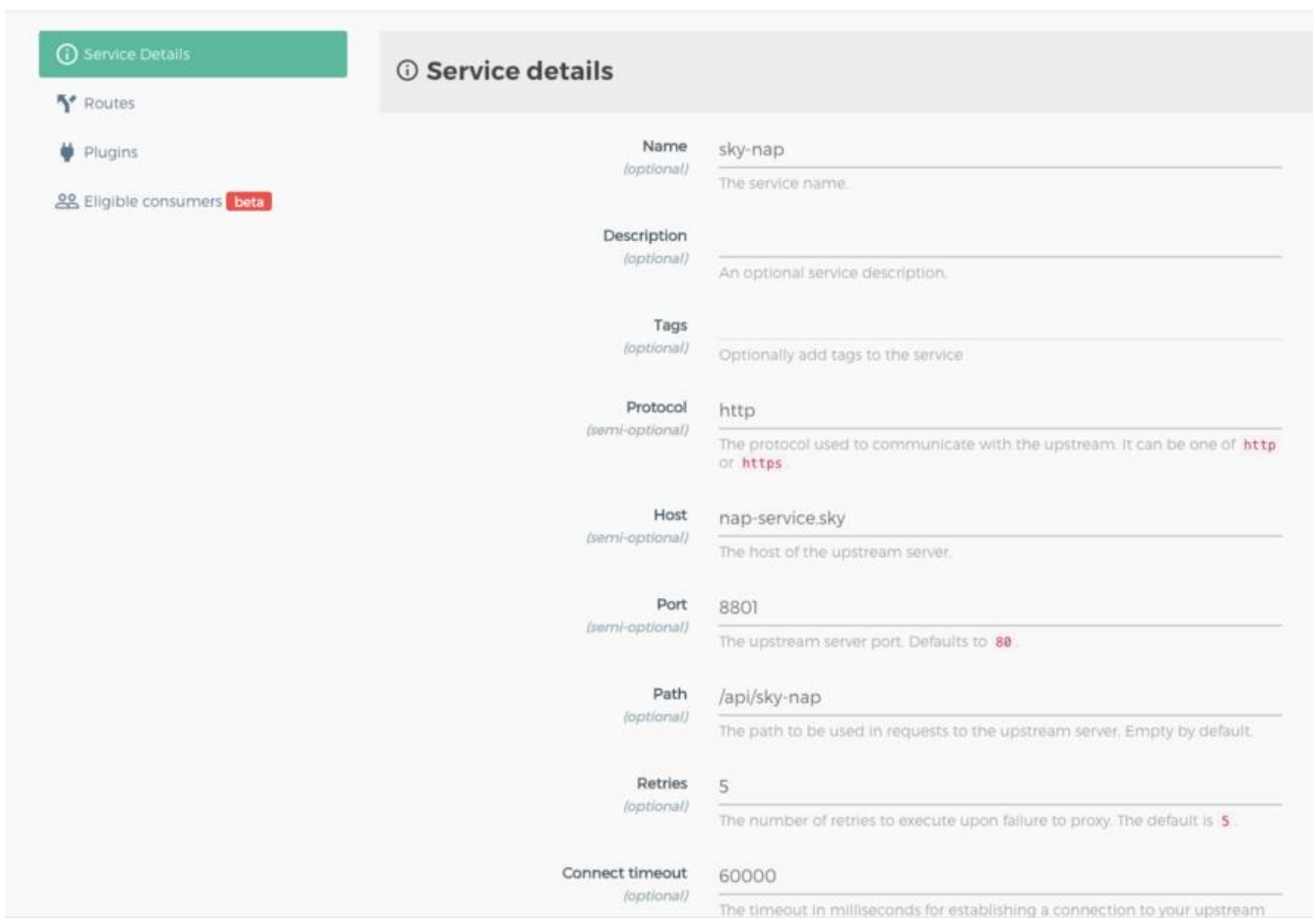


点击新增service



## 第二步

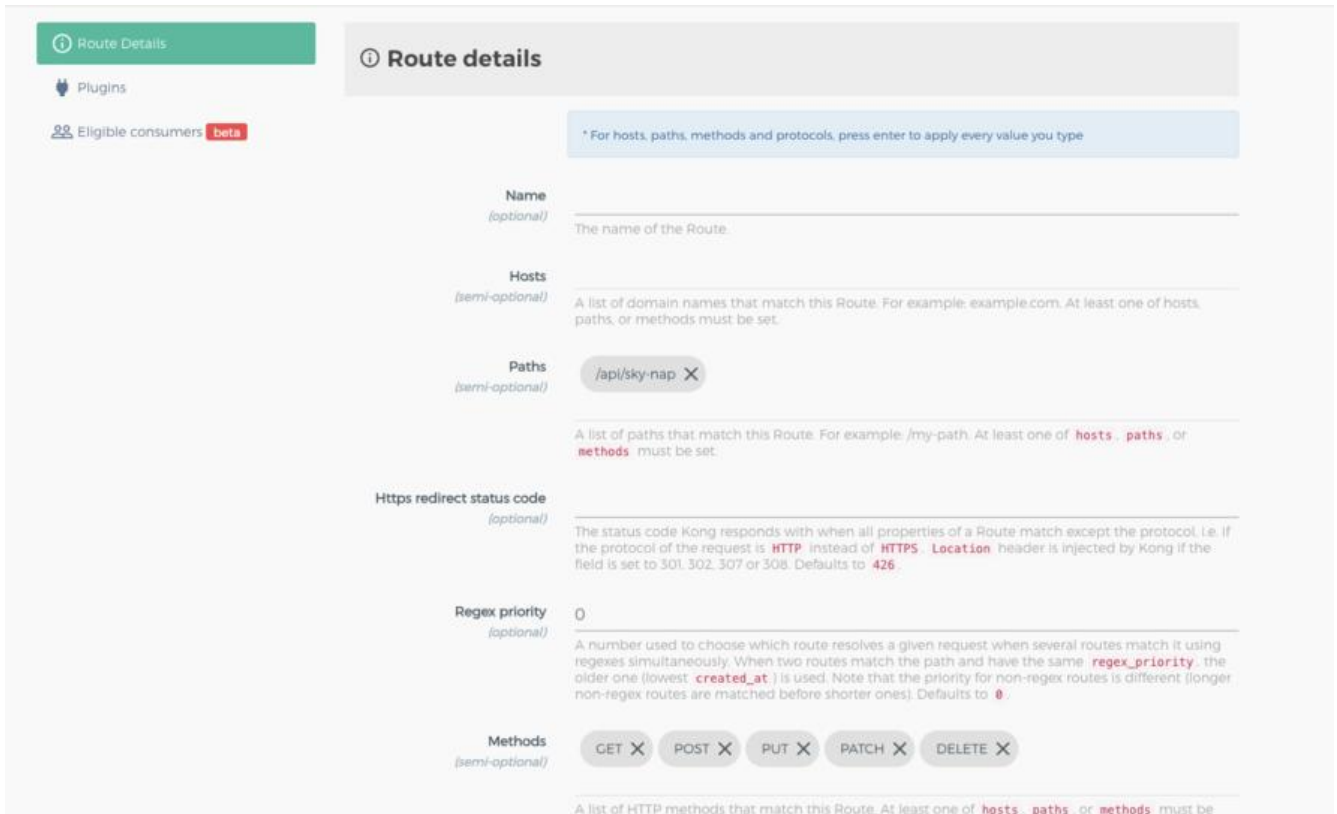
注册服务，包括这个服务的host,端口等。



## 第三步

点击上图的Routes,将这个服务路由到一个url路径

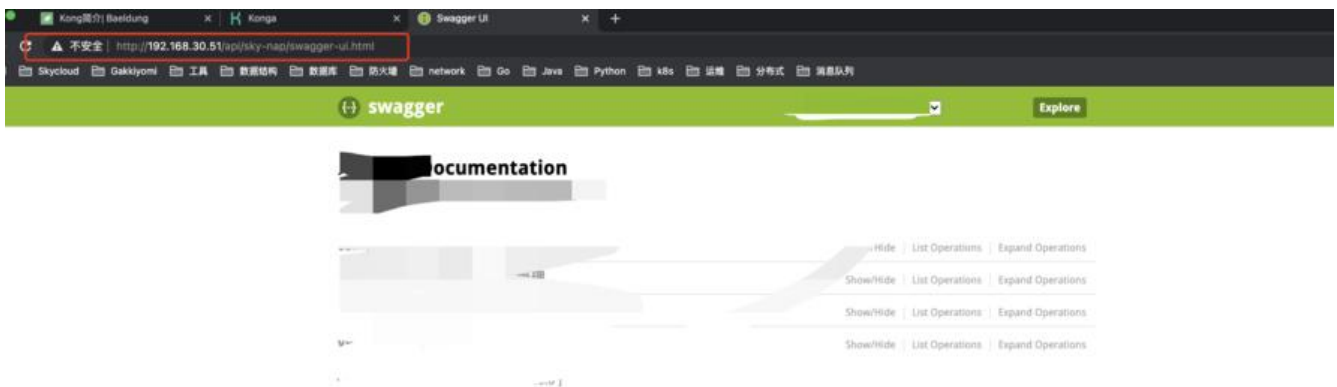
填写好Paths之后，我们就可以通过填写的这个路径来路由了



## 第四步

检验

在浏览器输入<http://192.168.30.51/api/sky-nap/swagger-ui.html>



这样简单的使用kong来路由服务就完成了joy