Kong 教程梳理

By huangnx

# Kong官方文档地址

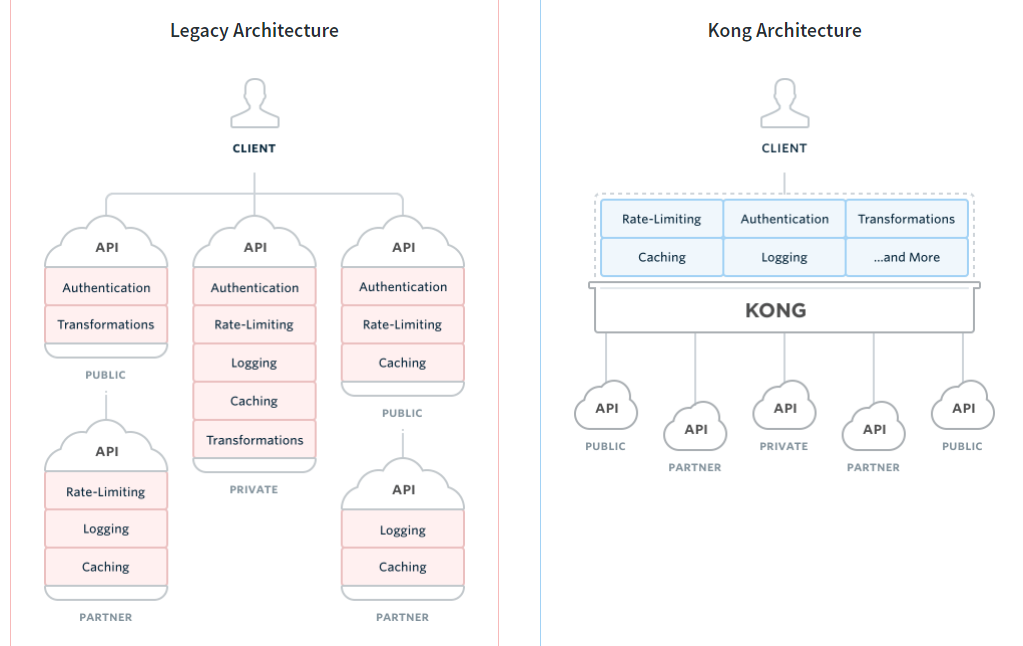
<https://docs.konghq.com/>

# Kong概述

## 简介

Kong 是在客户端和（微）服务间转发API通信的API网关和API服务管理层,通过插件扩展功能。我们可以  
可以通过增加更多 Kong Server 机器对 Kong 服务进行水平扩展，通过前置的负载均衡器向这些机器分发请求。

* 官网:https://getkong.org/
* github:https://github.com/Mashape/kong
* kong-dashboard(开源UI):https://www.npmjs.com/package/kong-dashboard



## 特点

* Kong核心基于OpenResty构建，实现了请求/响应的Lua处理化
* Kong通过Restful API提供了API/API消费者/插件/upstream/sni/证书的管理;
* 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；
* 缓存机制(为了避免每次查询数据库，Kong 在第一次请求之行时，尝试在本地的内存里做尽可能多的数据缓存。)、kong集群使kong具备高性能高可用.

## 结构

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

## 功能

* 服务注册及路由
* 负载均衡
* 安全控制
* 流量控制
* 审计日志
* 等等

# 常用的Kong组件介绍

PS：0.13之后API拆成了service+route。

## Service

Service 顾名思义就是我们自己定义的上游服务，通过Kong匹配到相应的请求要转发的地方。

Service 可以与下面的Route进行关联，一个Service可以有很多Route，匹配到的Route就会转发到Service中，当然中间也会通过Plugin的处理，增加或者减少一些相应的Header或者其他信息。

Service可以是一个实际的地址，也可以是Kong内部提供的upstream object。

## Route：

Route字面意思就是路由，实际就是我们通过定义一些规则来匹配客户端的请求，每个路由都会关联一个Service, 并且Service可以关联多个Route，当匹配到客户端的请求时，每个请求都会被代理到其配置的Service中。

Route作为客户端的入口，通过将Route和Service的松耦合，可以通过hosts path等规则的配置，最终让请求到不同的Service中。

例如，我们规定api.example.com 和 api.service.com的登录请求都能够代理到123.11.11.11:8000端口上，那我们可以通过hosts和path来路由

首先，创建一个Service s1，其相应的host和port以及协议为http://123.11.11.11:8000

然后，创建一个Route，关联的Service为s1，其hosts为[api.service.com, api.example.com], path为login

最后，将域名api.example.com和api.service.com的请求转到到我们的Kong集群上，也就是我们上面一节中通过Nginx配置的请求地址

那么，当我们请求api.example.com/login和api.service.com/login时，其通过Route匹配，然后转发到Service，最终将会请求我们自己的服务。

## Upstream：

这是指您自己的API /服务位于Kong后面，客户端请求被转发到该服务器。 相当于Kong提供了一个负载的功能，基于Nginx的虚拟主机的方式做的负载功能

当我们部署集群时，一个单独的地址不足以满足我们的时候，我们可以使用Kong的upstream来进行设置

首先在service中指定host的时候，可以指定为我们的upstream定义的hostname

我们在创建upstream时指定名字，然后指定solts (暂时不确定具体作用)，upstream可以进行健康检查等系列操作。这里先不开启（还没有研究）

然后我们可以再创建target类型，将target绑定到upstream上，那么基本上我们部署集群时，也可以使用

## Target：

target 就是在upstream进行负载均衡的终端，当我们部署集群时，需要将每个节点作为一个target，并设置负载的权重，当然也可以通过upstream的设置对target进行健康检查。

当我们使用upstream时，整个路线是 Route >> Service >> Upstream >> Target

## Consumer：

Consumer 可以代表一个服务，可以代表一个用户，也可以代表消费者，可以根据我们自己的需求来定义

可以将一个Consumer对应到实际应用中的一个用户，也可以只是作为一个Service的请求消费者

Consumer具体可以在Plugin使用时再做深入了解

## Plugin：

在请求被代理到上游API之前或之后执行Kong内的动作的插件。

例如，请求之前的Authentication或者是请求限流插件的使用

Plugin可以和Service绑定，也可以和Route以及Consumer进行关联。 具体的使用可以根据在创建Plugin以及后面的修改时，具体与Consumer，Service，Route绑定关系时，可参考

# 安装

PS：

支持多平台，但不支持windows

## 1．安装Kong（官网）：

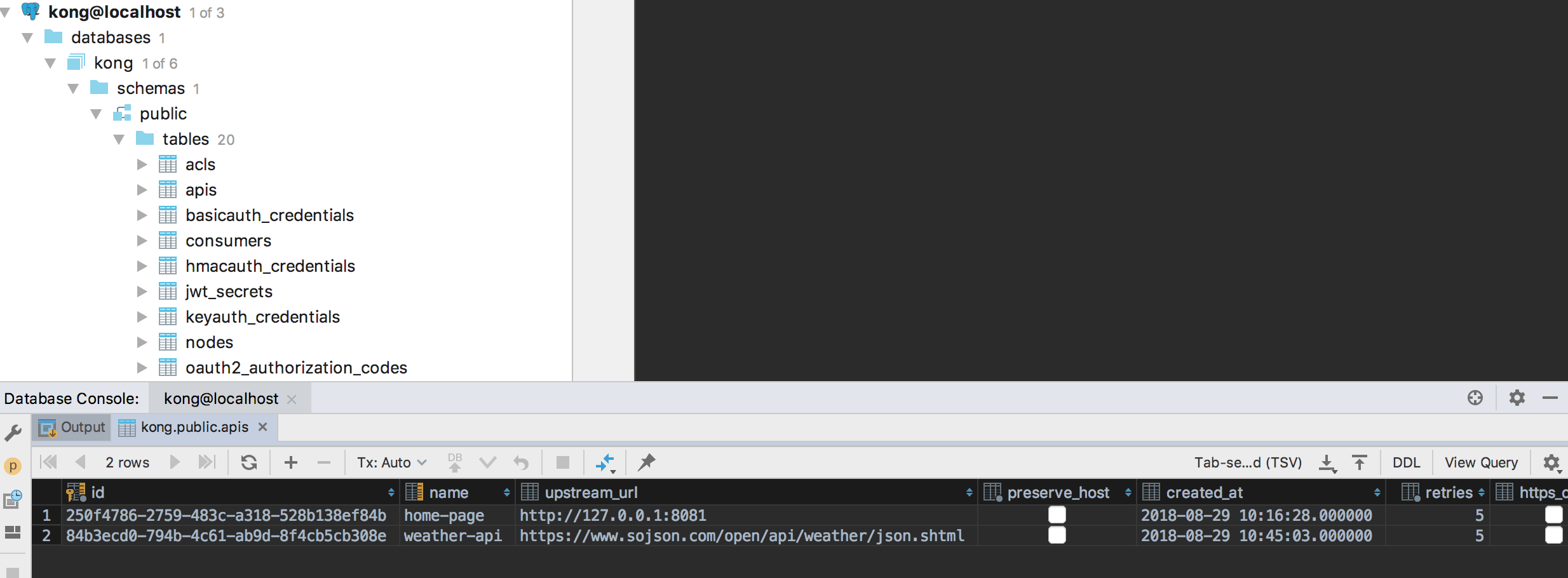
[https://getkong.org/install/](https://link.jianshu.com/?t=https://getkong.org/install/" \t "_blank)

## 2．安装postgresql

PS：支持windows、macOS、linux

[https://www.postgresql.org/download/](https://www.postgresql.org/download/" \t "_blank)

自带psql命令行工具，也可用数据库客户端可视化工具连接访问



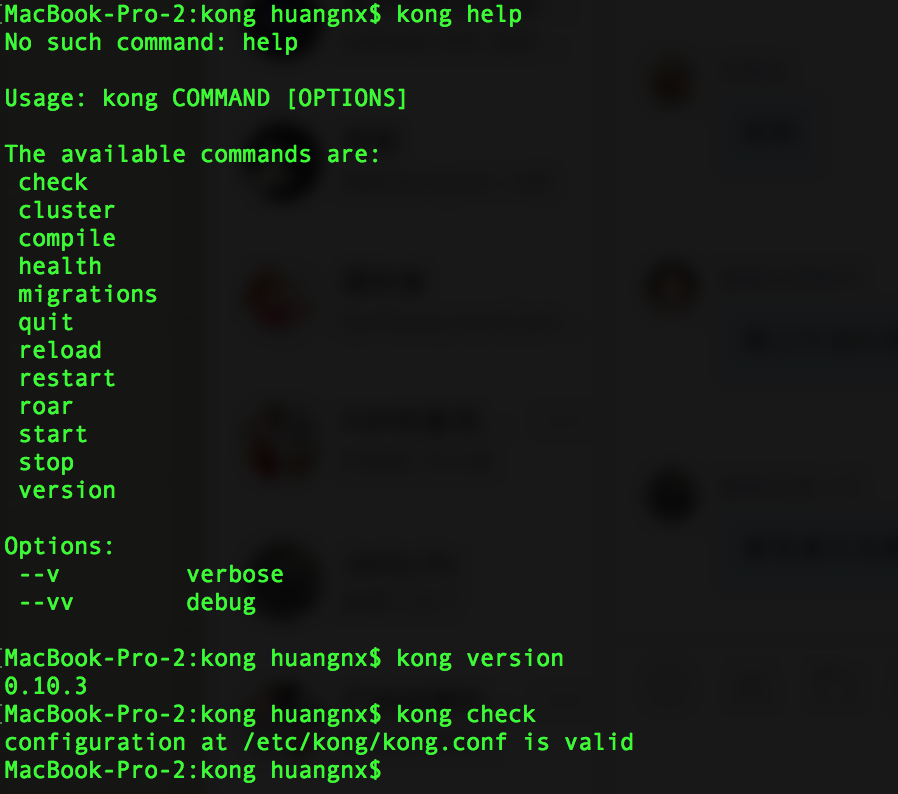
## 3．创建数据库及用户并赋权

CREATE USER kong WITH PASSWORD kong;

CREATE DATABASE kong OWNER kong;

GRANT ALL PRIVILEGES ON DATABASE kong to kong;

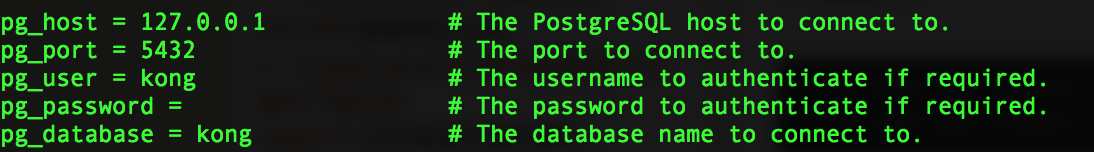
## 4．校验安装是否成功



# Kong初始化配置

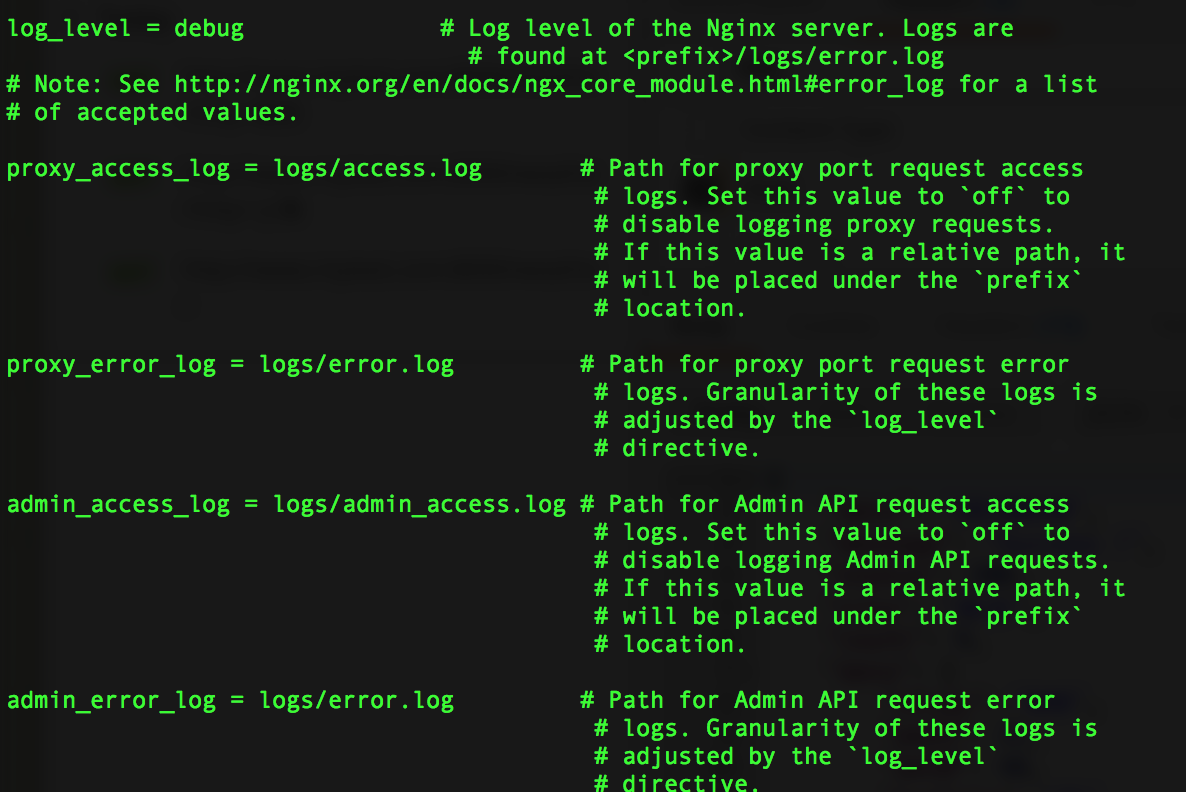
## 参考官方文档：<https://docs.konghq.com/0.9.x/configuration/>

## 数据库配置： /etc/kong/kong.conf 中如下配置必须设置，才能启动kong



也可用Cassandra作为数据库。见官方文档配置

## kong的日志配置



## nginx配置

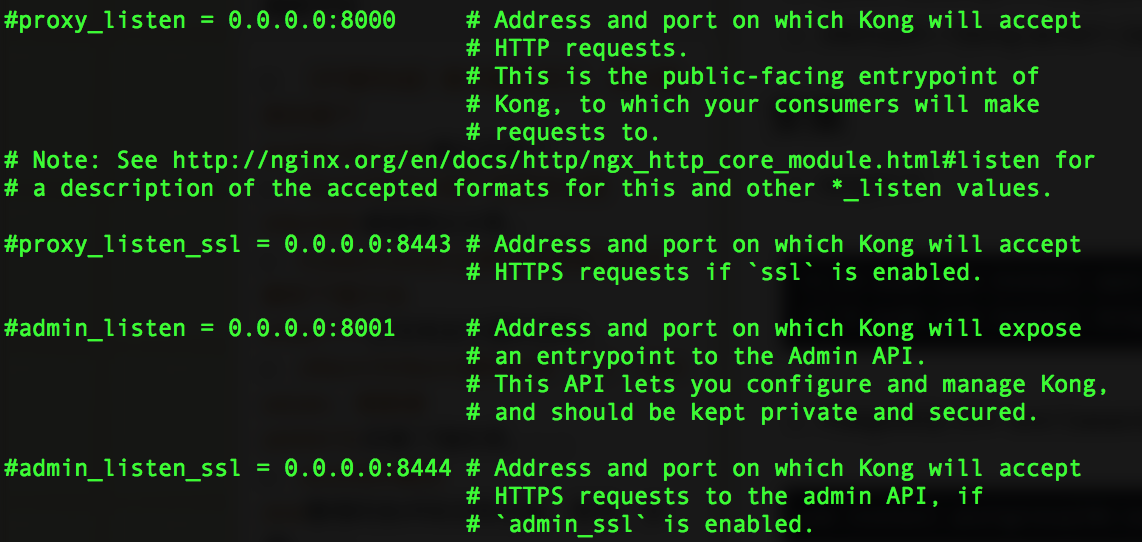
kong是基于nginx的。启用的nginx配置路径为：/usr/local/kong/nginx.conf。可以在其中引用自定义的配置



## 端口配置：

：默认监听代理8000端口的http请求，进行代理

：默认8001端口为admin端口



## 集群部署配置

kong可以集群化部署，这里先不介绍。用到的时候再说

# Kong使用及应用配置

## Kong命令

参见：<https://docs.konghq.com/0.14.x/cli/>

|  |  |  |
| --- | --- | --- |
| 启动 | kong start -c /etc/kong/kong.conf |  |
| 停止 | kong stop |  |
| 重新加载配置文件 | kong reload |  |
| 检查配置文件有效性 | kong check |  |
| 检查kong启用健康状态 | kong health |  |
| kong数据库重置或初始化 | kong migrations | up、list、reset |
| kong重启 | kong restart |  |

## Kong Admin Api

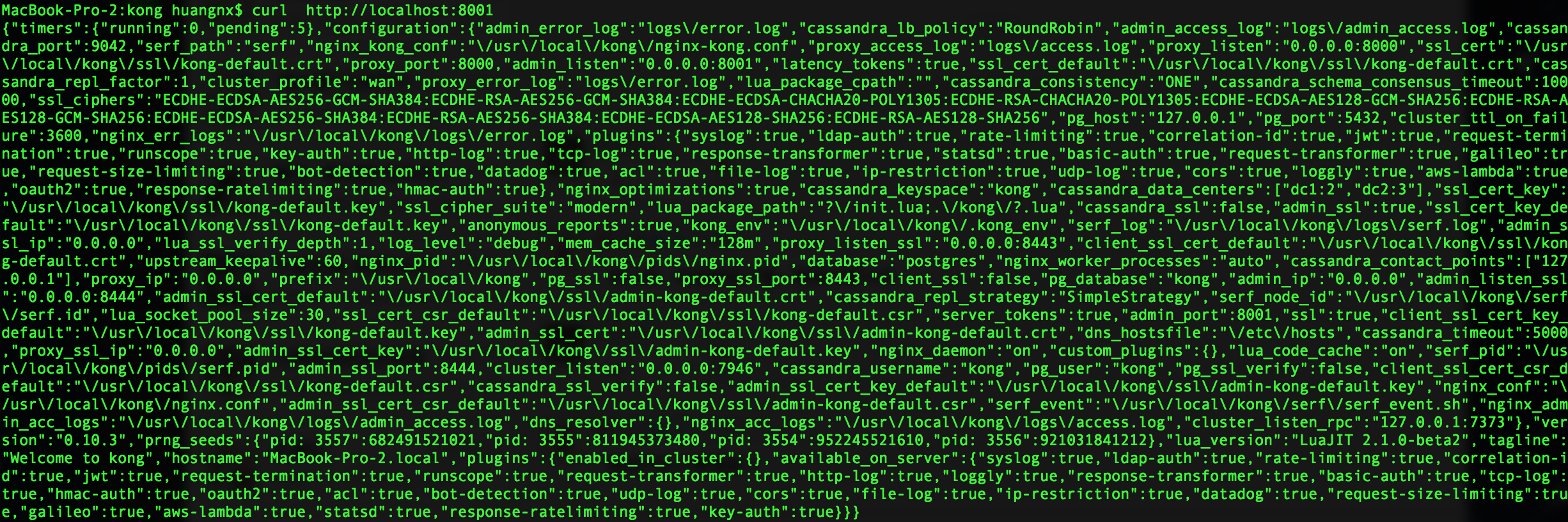
参见：<https://docs.konghq.com/0.14.x/admin-api/>

有相关字段的含义说明

### Kong信息获取

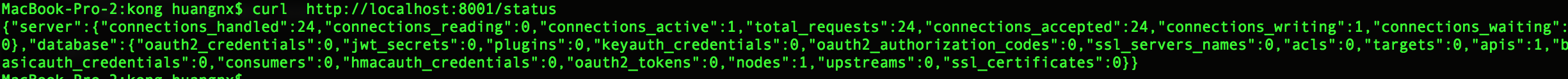
#### 查看Kong节点信息：

<http://localhost:8001>



#### 查看Kong节点的运行信息

<http://localhost:8001/status>



### API操作

#### 增加一个接入的API

* + **Example**

curl -i -X POST \

--url http://localhost:8001/apis/ \

--data 'name=weather-api' \

--data 'hosts=mytest.com,www.mytest.com,127.0.0.1 ' \

--data 'uris=/ \

--data 'upstream\_url=http://127.0.0.1:8081'

* + **Request param**

|  |  |  |
| --- | --- | --- |
| 属性 | 约束 | 描述 |
| name | required | 接入的API名称 |
| hosts | semi-optional | 逗号分割的接入API的域名列表 |
| uris | semi-optional | 逗号分割的接入API的前缀path，即指定uri用户通过kong刚问要加上这个path |
| methods | semi-optional | 逗号分割的接入API的HTTP method，如get /post/ put/delete/.. |
| upstream\_url | required | 代理的上游API Server |
| strip\_uri | optional,default:true | 当匹配到uris前缀时，去掉请求的upstream\_url中匹配的uris；即uris是挂载在kong的路径下，不是上游接口的path |
| preserve\_host | optional,default:false | Kong默认将上游请求的Host头设置为从API的upstream\_url中提取的主机名，当通过hosts来匹配API时，确保hosts能转发到上游服务 |
| retries | optional,default:5 | 代理失败时重试的次数 |
| upstream\_connect\_timeout | optional,default:60000ms | 建立与上游连接的超时时间(ms) |
| upstream\_send\_timeout | optional,default:60000ms | 在发送请求到上游服务的两个连续写入操作之间的超时时间(ms) |
| upstream\_read\_timeout | optional,default:60000ms | 在发送请求到上游服务的两个连续读取操作之间的超时时间(ms) |
| https\_only | optional,default:false | 如果希望仅通过HTTPS转发API（默认8443端口），则启用 |
| http\_if\_terminated | optional,default:false | 仅在https限流时才考虑设置X-Forwarded-Proto头部 |

#### 查看接入的所有api

curl -i -X GET --url <http://localhost:8001/apis/>

#### 查看指定api

curl -i -X GET \ --url http://localhost:8001/apis/${apiNameOrId}

#### 根据name或id更新一个API

* + Example

curl -i -X PATCH \

--url http://localhost:8001/apis/weather-api \

--data 'name=weather-api-1' \

--data 'retries=6'

* + Request param
    - 同‘增加一个接入的API’

#### 根据name或id删除一个API

curl -i -X DELETE \

--url [http://localhost:8001/apis/${apiNameOrId](http://localhost:8001/apis/$%7bapiNameOrId)}

### 关于service、router

Kong 0.13版本后建议用这两种替代API。我们用的是0.11版本。此处不做介绍

### 消费者操作

#### 创建消费者

* Example

curl -i -X POST \

--url http://localhost:8001/consumers/ \

--data “username=test “\

--data “custom\_id=000001”

* **Requst Body**

|  |  |  |
| --- | --- | --- |
| 属性 | 约束 | 描述 |
| username | semi-optional | 消费者的唯一用户名，和custom\_id至少有一个必须指定 |
| custom\_id | semi-optional | 用户存储的唯一id, 用来和现有数据库中的用户一一映射，若需要权限管理，必须将此字段或用户名与请求一起发送。 |

#### 查询所有消费者列表

curl -i -X GET \ --url <http://localhost:8001/consumers>

#### 查询指定消费者

curl -i -X GET \ --url [http://localhost:8001/consumers/${idOrName](http://localhost:8001/consumers/$%7bidOrName)}

#### 更新指定消费者

curl -i -X PATCH --url <http://localhost:8001/consumers/test>--data "username=test001"

#### 删除指定消费者

curl -i -X DELETE \ --url [http://localhost:8001/consumers/${idOrName](http://localhost:8001/consumers/$%7bidOrName)}

### upstream操作

### 证书操作

### SNI操作

### 插件操作

和上面大同小异，不再详细列举。

参考官方文档：<https://docs.konghq.com/0.14.x/admin-api/>

参考中文博客：<https://www.cnblogs.com/zhoujie/p/kong2.html>

## Kong 插件

参见：<https://konghq.com/plugins/>

### 官方插件列表

| **类别** | **免费/收费** | **name** | **插件名** | **使用场景** |
| --- | --- | --- | --- | --- |
| 认证 | 免费 | basic-auth | Basic Authentication | 对于服务或者路由提供用户名/密码基本认证机制 |
| 认证 | 免费 | key-auth | Key Authentication | 对于服务或者路由提供用关键字认证机制 |
| 认证 | 收费 | - | OpenID Connect | 提供与三方OpenID的集成方式 |
| 认证 | 免费 | oauth2 | OAuth 2.0 Authentication | 添加OAuth 2.0认证 |
| 认证 | 收费 | - | OAuth 2.0 Introspection | 提供与三方OAuth 2.0认证服务器的集成方式 |
| 认证 | 免费 | hmac-auth | HMAC Authentication | 提供HMAC（Hashed Message Authentication Code）签名认证方式 |
| 认证 | 免费 | jwt | JWT | 提供JWT(JSON WEB Token)的认证方式 |
| 认证 | 免费 | ldap-auth | LDAP Authentication | 提供了与LDAP认证服务器的集成方式 |
| 安全 | 免费 | acl | ACL | 通过ACL(Access Control List)的组名称对服务或者路由进行黑白名单的访问控制 |
| 安全 | 免费 | cors | CORS | 对服务或者路由提供CORS支持 |
| 安全 | 免费 | ip-restriction | IP Restriction | 通过IP地址对服务或者路由进行黑白名单的访问控制 |
| 安全 | 免费 | bot-detection | Bot Detection | 对服务或者路由提供机器人检出并能进行黑白名单的设定 |
| 安全 | 免费 | ssl | Dynamic SSL | 对路由或者API提供ssl支持 |
| 流量控制 | 收费 | - | EE Canary Release | 提供金丝雀发布支持 |
| 流量控制 | 收费 | - | EE Forward Proxy | 提供企业内网中前向代理的支持 |
| 流量控制 | 收费 | - | EE Proxy Caching | 提供代理缓存的加速功能 |
| 流量控制 | 免费 | request-size-limiting | Request Size Limiting | 可以对请求的body的size进行限制 |
| 流量控制 | 免费 | rate-limiting | Rate Limiting | 提供对给定时间段HTTP请求数目的控制功能 |
| 流量控制 | 收费 | - | EE Rate Limiting | CE版本Rate Limiting的强化 |
| 流量控制 | 免费 | response-ratelimiting | Response Rate Limiting | 基于用户响应Header的信息，提供对HTTP请求数目的控制功能 |
| 流量控制 | 免费 | request-termination | Request Termination | 根据响应的状态码和信息，可停止对某个服务或者路由的流量 |
| Serverless | 免费 | aws-lambda | AWS Lambda | 调用和管理AWS Lambda函数 |
| Serverless | 免费 | openwhisk | Apache OpenWhisk | 调用和管理OpenWhisk |
| 分析&监控 | 免费 | zipkin | Zipkin | 提供兼容zipkin的方式对用户请求进行追踪 |
| 分析&监控 | 免费 | datadog | Datadog | 可将API的指标信息在datadog上进行可视化地展示 |
| 分析&监控 | 免费 | runscope | Runscope | 结合runscope进行API性能测试和监控 |
| 转换 | 免费 | request-transformer | Request Transformer | 通过kong对请求进行转换 |
| 转换 | 收费 | - | EE request Transformer | request Transformer的强化版 |
| 转换 | 免费 | response-transformer | Response Transformer | 通过kong对响应进行转换 |
| 转换 | 免费 | correlation-id | Correlation ID | 通过kong实现请求和响应之间的关联 |
| 日志 | 免费 | tcp-log | TCP | 发送请求和响应日志到TCP服务器 |
| 日志 | 免费 | udp-log | UDP | 发送请求和响应日志到UDP服务器 |
| 日志 | 免费 | http-log | HTTP | 发送请求和响应日志到HTTP服务器 |
| 日志 | 免费 | file-log | File | 发送请求和响应日志到磁盘的文件中 |
| 日志 | 免费 | statsd | StatsD | 发送请求和响应日志到StatsD服务器 |
| 日志 | 免费 | syslog | Syslog | 发送请求和响应日志到syslog中 |
| 日志 | 免费 | syslog | Loggly | 发送请求和响应日志到Loggly服务器 |

### Kong插件作用对象

可以通过五种不同的方式添加插件：

* 对于每个Service/Route and Consumer，不要设置consumer\_id并同时设置service\_id或route\_id。
* 对于每个Service/Route和特定的Consumer，只设置consumer\_id。
* 针对每个Consumer和特定Service，只设置service\_id（警告：一些插件只允许设置他们的route\_id）
* 针对每个 Consumer和特定Route，只设置route\_id（警告：一些插件只允许设置他们的service\_id）
* 针对特定的Service/Route/Consumer，同时设置service\_id / route\_id和consumer\_id。

请注意，并非所有插件都允许指定consumer\_id

### Kong插件开发

不详细介绍。

参考别人插件开发的博客：

<https://www.jianshu.com/p/2dae283b0395>

<https://www.jianshu.com/p/186accacda84>

## Kong 集群

### 0.11版本：

当一个节点发生更新到数据库后，其他节点需要间隔db\_update\_frequency后，更新缓存失效字段。在使用Cassandra数据库db\_update\_propagation（数据节点传播延时）必须配置。

为防止db\_update\_frequency后错过一个失效事件，db\_cache\_ttl全量更新cache

db\_update\_frequency = 5

db\_update\_propagation = 0

db\_cache\_ttl = 3600

### 0.10版本：

通过 Kong 集群，每个节点能够知道其它节点的存在。并且当一个Kong 节点有对数据更新，该节点有责任通知这个变化给集群里的其他节点，

通知其其他节点把本地内存中的缓存无效，重新从数据库中获取更新后的数据。

cluster\_listen 通信

cluster\_listen\_rpc 代理通信

cluster\_advertise

cluster\_encrypt\_key base64编码16字节加密集群通信

cluster\_keyring\_file

cluster\_ttl\_on\_failure //失败节点超时后停止被链接；

cluster\_profile //local, lan, wan.

--数据库node表来存储节点信息：

node

VM\_3\_3\_centos\_0.0.0.0:7946\_58b54877bc4a47d884b9986e71f49d8b | 10.100.3.3:7946 | 2017-08-07 04:29:05

VM\_3\_2\_centos\_0.0.0.0:7946\_e4147c192bcc4401acf698ca6374d59d | 10.100.3.2:7946 | 2017-06-12 10:19:24

# Kong-dashboard

Kong的网关的应用配置有可视化工具，可部署该可视化工具在其上做应用配置。

很easy，就不细讲了。

参见：<https://github.com/PGBI/kong-dashboard>