目录

[一、kong介绍 1](#_Toc11143133)

[1、Kong的概述 1](#_Toc11143134)

[2、功能特性 1](#_Toc11143135)

[二、使用yaml文件在kubernets上部署kong及konga 3](#_Toc11143136)

[2.1、前置条件 3](#_Toc11143137)

[2.2、kong部署 4](#_Toc11143138)

[三、konga配置 5](#_Toc11143139)

[3.1、konga初始化设置 5](#_Toc11143140)

[3.2、将kubernets服务接入kong 7](#_Toc11143141)

[四、排障及总结 8](#_Toc11143142)

# 一、kong介绍

## 1、Kong的概述

Kong是一个clould-native、快速的、可扩展的、分布式的微服务抽象层（也称为API网关、API中间件或在某些情况下称为服务网格）框架。Kong作为开源项目在2015年推出，它的核心价值是高性能和可扩展性。Kong被广泛用于从初创企业到全球5000家公司以及政府组织的生产环境中。

如果构建Web、移动或IoT（物联网）应用，可能最终需要使用通用的功能来实现这些应用。Kong充当微服务请求的网关（或侧车），通过插件能够提供负载平衡、日志记录、身份验证、速率限制、转换等能力。



## 2、功能特性

* 云本土化(Cloud-Native)：Kong可以在Kubernetes或物理环境上运行；
* 动态负载平衡(Dynamic Load Balancing)：跨多个上游服务的负载平衡业务。
* 基于哈希的负载平衡(Hash-based Load Balancing)：一致的散列/粘性会话的负载平衡。
* 断路器(Circuit-Breaker)：智能跟踪不健康的上游服务。
* 健康检查(Health Checks)：主动和被动监控您的上游服务。
* 服务发现(Service Discovery)：解决如Consul等第三方DNS解析器的SRV记录。
* 无服务器(Serverless)：从Kong中直接调用和保证AWS或OpenWhisk函数安全。
* WebSockets：通过WebSockets与上游服务进行通信。
* OAuth2.0：轻松的向API中添加OAuth2.0认证。
* 日志记录(Logging)：通过HTTP、TCP、UDP记录请求或者相应的日志，存储在磁盘中。
* 安全(Security)：ACL，Bot检测，IPs白名单/黑名单等。
* 系统日志(Syslog)：记录信息到系统日志。
* SSL：为基础服务或API设置特定的SSL证书。
* 监视(Monitoring)：能够实时对关键负载和性能指标进行监控。
* 转发代理(Forward Proxy)：使端口连接到中间透明的HTTP代理。
* 认证(Authentications)：支持HMAC，JWT和BASIC方式进行认证等等。
* 速率限制(Rate-limiting)：基于多个变量的阻塞和节流请求。
* 转换(Transformations)：添加、删除或操作HTTP请求和响应。
* 缓存(Caching)：在代理层进行缓存和服务响应。
* 命令行工具(CLI)：能够通过命令行控制Kong的集群。
* REST API：可以通过REST API灵活的操作Kong。
* GEO复制：在不同的区域，配置总是最新的。
* 故障检测与恢复(Failure Detection & Recovery)：如果Cassandra节点失效，Kong并不会受影响。
* 群集(Clustering)：所有的Kong节点会自动加入群集，并更新各个节点上的配置。
* 可扩展性(Scalability)：通过添加节点，实现水平缩放。
* 性能(Performance)：通过缩放和使用Nigix，Kong能够轻松处理负载。
* 插件(Plugins)：基于插件的可扩展体系结构，能够方便的向Kong和API添加功能。

3、Kong依赖的技术

Kong部署在Nginx和Apache Cassandra或PostgreSQL等可靠技术之上，并提供了易于使用的RESTful API来操作和配置系统。下面是Kong的技术逻辑图。基于这些技术，Kong提供相关的特性支持：

**Nginx**：

* 经过验证的高性能基础；
* HTTP和反向代理服务器；
* 处理低层级的操作。

**OpenRestry**：

* 支持Lua脚本；
* 拦截请求/响应生命周期；
* 基于Nginx进行扩展。

**Clustering&Datastore**：

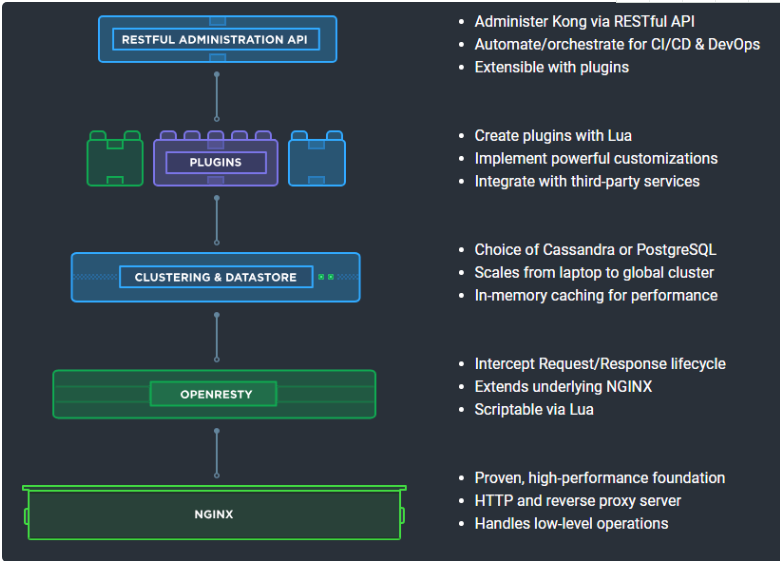
* 支持Cassandra或PostgreSQL数据库；
* 内存级的缓存；
* 支持水平扩展。

**Plugins**：

* 使用Lua创建插件；
* 功能强大的定制能力；
* 与第三方服务实现集成。

**Restful Administration API**：

* 通过Restful API管理Kong；
* 支持CI/CD&DevOps；
* 基于插件的可扩展。



# 二、使用yaml文件在kubernets上部署kong及konga

## 2.1、前置条件

* Kubernets集群（kube-dns、ingress-controller）

说明：初始化postgress的时候kong使用svc访问postgres，因此kube-dns必须部署好，最后kong-proxy以及后面的konga都需要对外暴露，因此ingress-controller也必须部署好。

* 相关镜像

按照相关文档，postgres和kong以及konga版本都是有要求的。

|  |  |  |
| --- | --- | --- |
| 镜像 | Tag | 说明 |
| postgres | 10.0 | 数据库镜像 |
| kong | 1.1-centos | Kong服务镜像 |
| pantsel/konga | 0.14.1 | 提供kong web管理界面 |

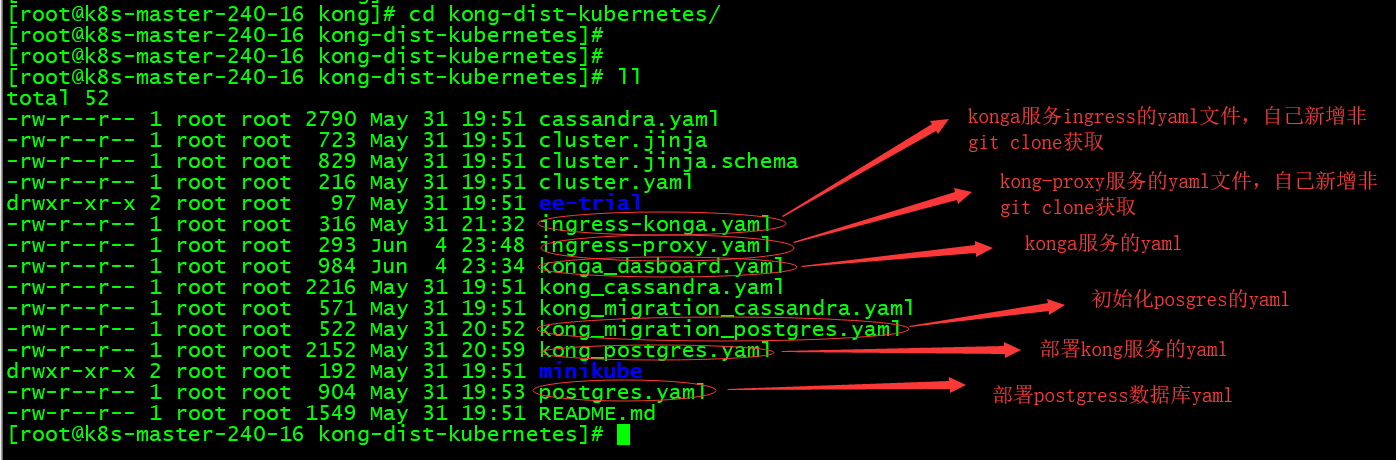
## 2.2、kong部署

克隆代码（代码会更新）

[root@k8s-master-240-16 /]# cd /usr/local/k8s/kong/

[root@k8s-master-240-16 kong]# git clone <https://github.com/Kong/kong-dist-kubernetes.git>

由于在搭建kong使用postgres，因此用到cassandra数据库的yaml文件都忽略掉，我们使用如下标识的yaml文件，git clone后对原始的yaml文件进行了修改。



本次部署使用到的yaml文件





关于kong部署yaml文件kong\_postgres.yaml说明：

在部署过程中kong启动了三个副本，有一个成功两个失败，将副本数量改为1就失败，进一步排查发现pod分配到其中一个节点是成功的，因此断定kubernest集群节点有问题，因此加如下配置将pod调度到一个节点上

部署服务

[root@k8s-master-240-16 kong-dist-kubernetes]# kubectl create -f postgres.yaml

[root@k8s-master-240-16 kong-dist-kubernetes]# kubectl create -f kong\_migration\_postgres.yaml

[root@k8s-master-240-16 kong-dist-kubernetes]# kubectl create -f kong\_postgres.yaml

[root@k8s-master-240-16 kong-dist-kubernetes]# kubectl create -f konga\_dasboard.yaml

[root@k8s-master-240-16 kong-dist-kubernetes]# kubectl create -f ingress-konga.yaml ###暴露konga服务

[root@k8s-master-240-16 kong-dist-kubernetes]# kubectl create -f ingress-proxy.yaml ###暴露kong-proxy服务

查看pod

[root@k8s-master-240-16 kong-dist-kubernetes]# kubectl get pod

NAME READY STATUS RESTARTS AGE

kong-migration-dx5cv 0/1 Completed 0 4d5h

kong-rc-657798c4f-2n52h 1/1 Running 2 4d5h

kong-rc-657798c4f-mnt56 1/1 Running 2 4d5h

kong-rc-657798c4f-qw9jp 1/1 Running 2 4d5h

konga-d545c55f-dd77z 1/1 Running 0 152m

konga-d545c55f-lp44s 1/1 Running 0 152m

postgres-nsgqc 1/1 Running 2 4d6h

查看svc

[root@k8s-master-240-16 kong-dist-kubernetes]# kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kong-admin ClusterIP 10.254.119.175 <none> 8001/TCP 4d5h

kong-admin-ssl ClusterIP 10.254.195.55 <none> 8444/TCP 4d5h

kong-proxy ClusterIP 10.254.202.197 <none> 8000/TCP 4d5h

kong-proxy-ssl ClusterIP 10.254.44.84 <none> 8443/TCP 4d5h

konga ClusterIP 10.254.86.38 <none> 1337/TCP 153m

kubernetes ClusterIP 10.254.0.1 <none> 443/TCP 165d

postgres ClusterIP 10.254.37.44 <none> 5432/TCP 4d6h

查看ingress

[root@k8s-master-240-16 kong-dist-kubernetes]#

[root@k8s-master-240-16 kong-dist-kubernetes]# kubectl get ingress

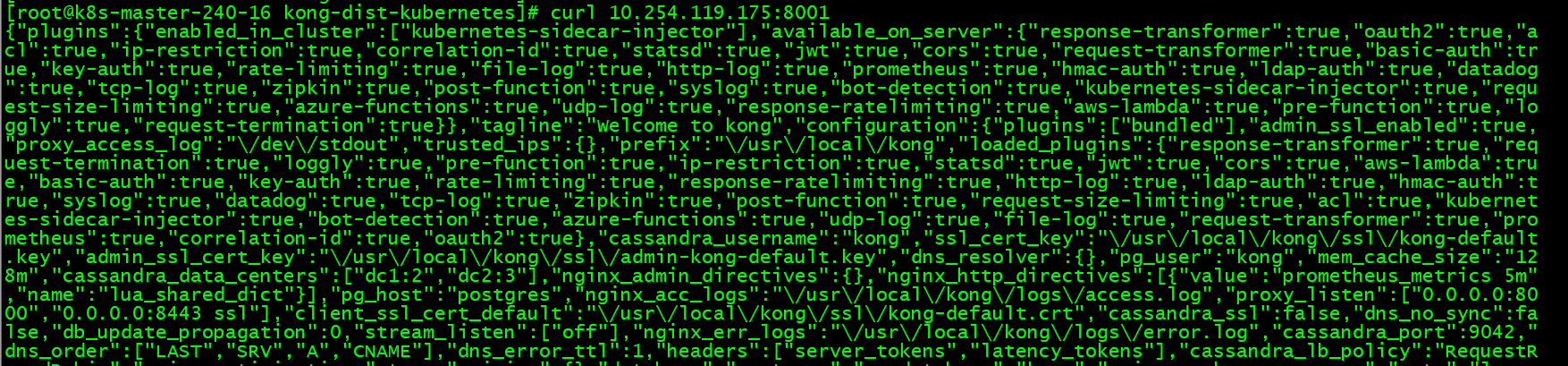
NAME HOSTS ADDRESS PORTS AGE

ingress-pengixn konga.pengxin.com 80 37d

ingress-proxy proxy.pengxin.com 80 140m

[root@k8s-master-240-16 kong-dist-kubernetes]#

验证kong-admin是否有数据，如下为正常。

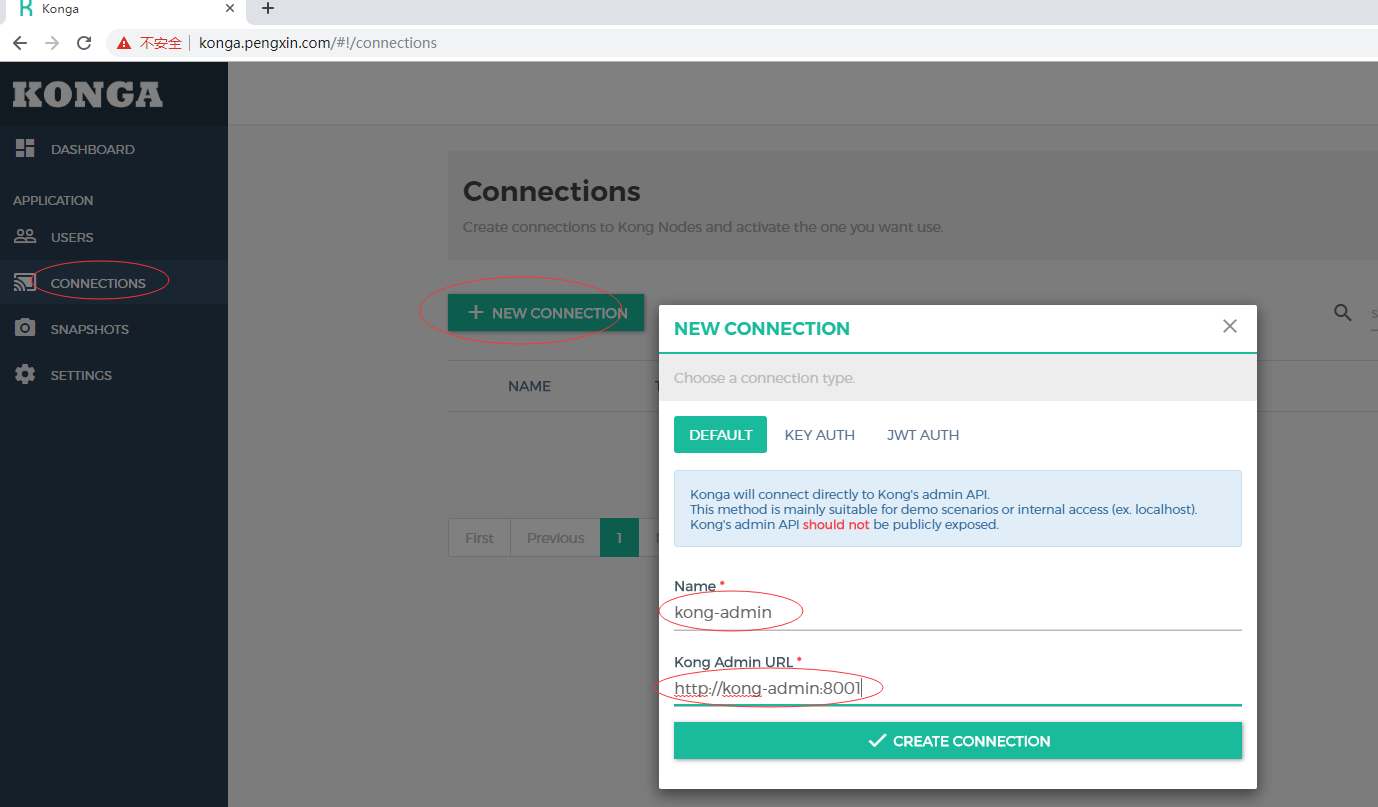


# 三、konga配置

## 3.1、konga初始化设置

在上面步骤中已经通过ingress暴露了konga的域名，在本地绑定host（192.168.240.16 konga.pengxin.com

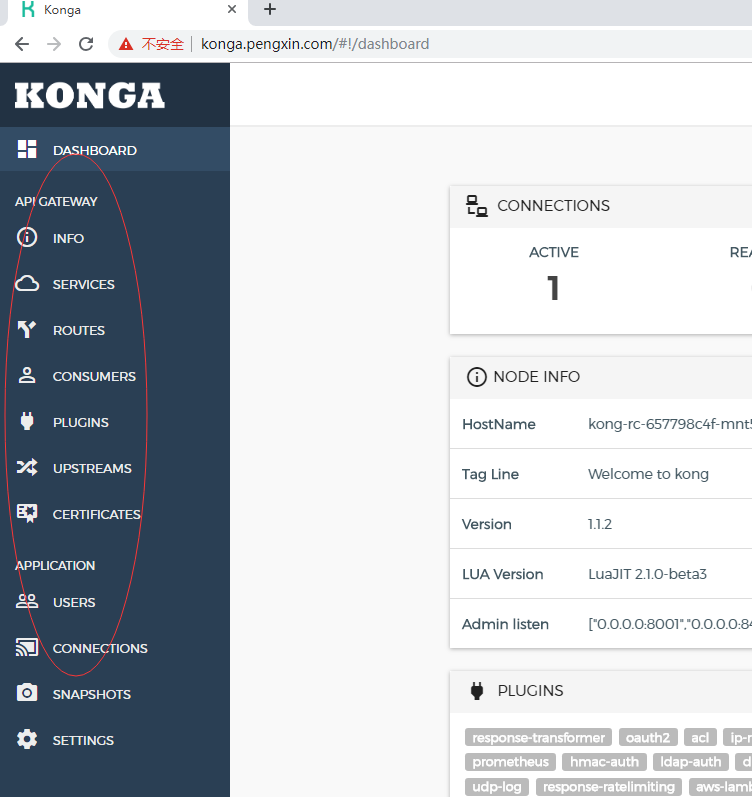
）即可使用域名访问,并做相应设置：



激活配置：



激活后左侧菜单栏多了菜单选项



## 3.2、将kubernets服务接入kong

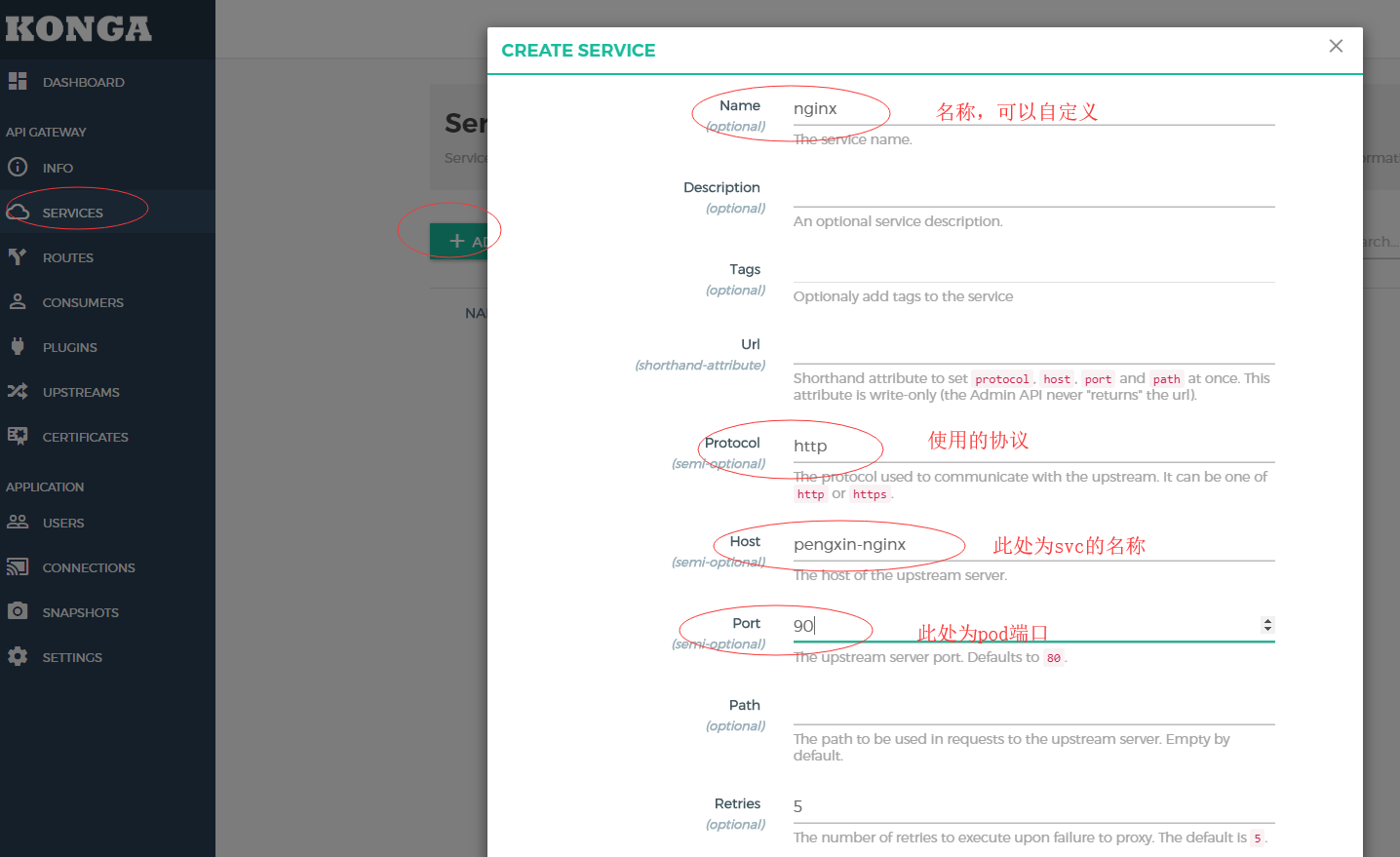
创建一个nginx的应用（步骤略）

[root@k8s-master-240-16 kong-dist-kubernetes]# kubectl get svc | grep pengxin-nginx

pengxin-nginx ClusterIP 10.254.141.219 <none> 90/TCP 14s

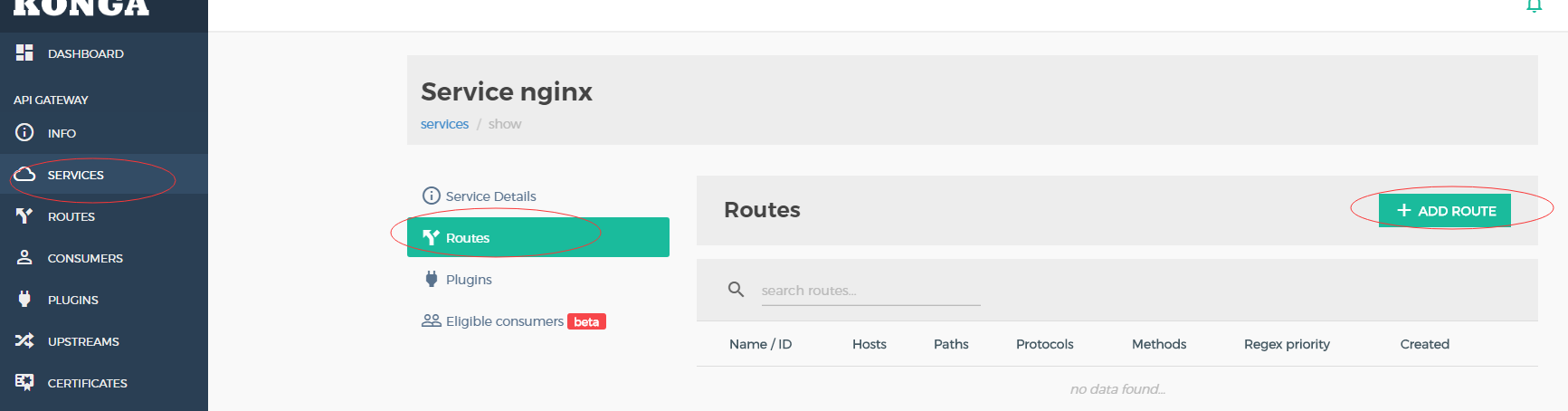
kong配置

* 配置service，其他默认

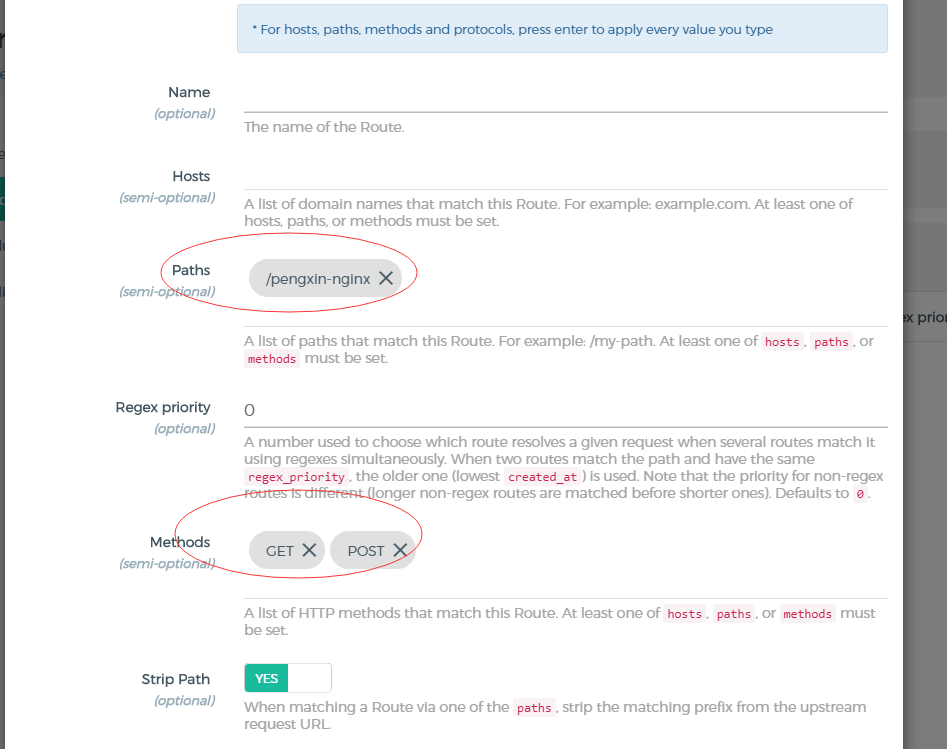


* 配置routing

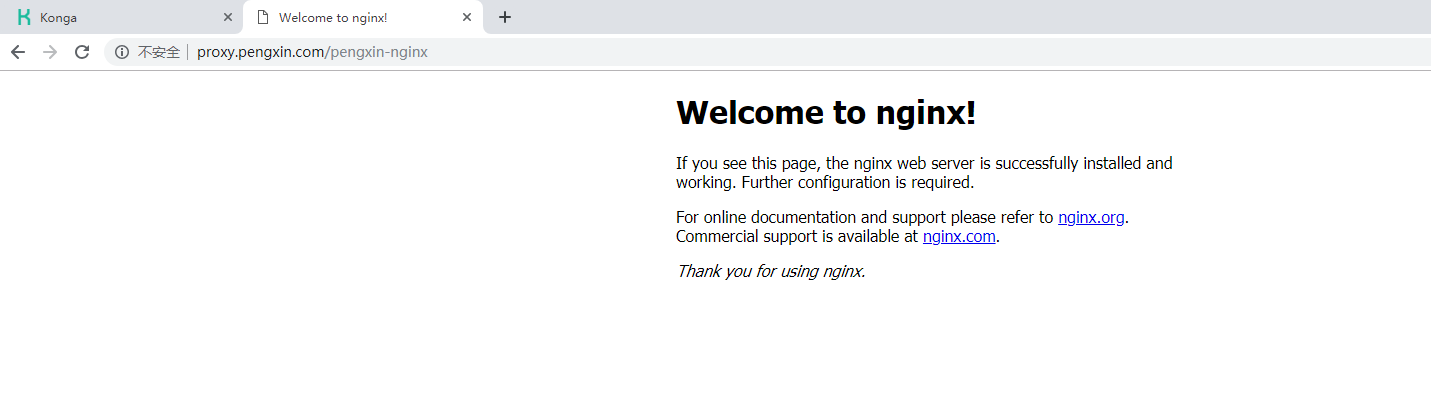
选择刚才的svc配置routing



Path为通过kong-proxy域名访问后面跟的路径，methods限定请求方法，此处设置好相应配置后回车



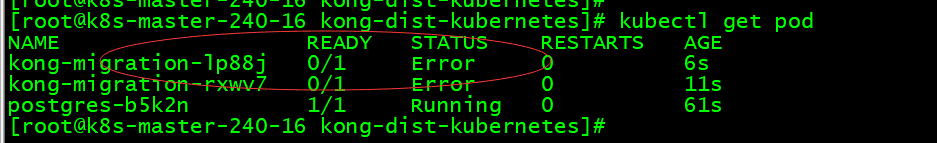
访问<http://proxy.pengxin.com/pengxin-nginx>验证

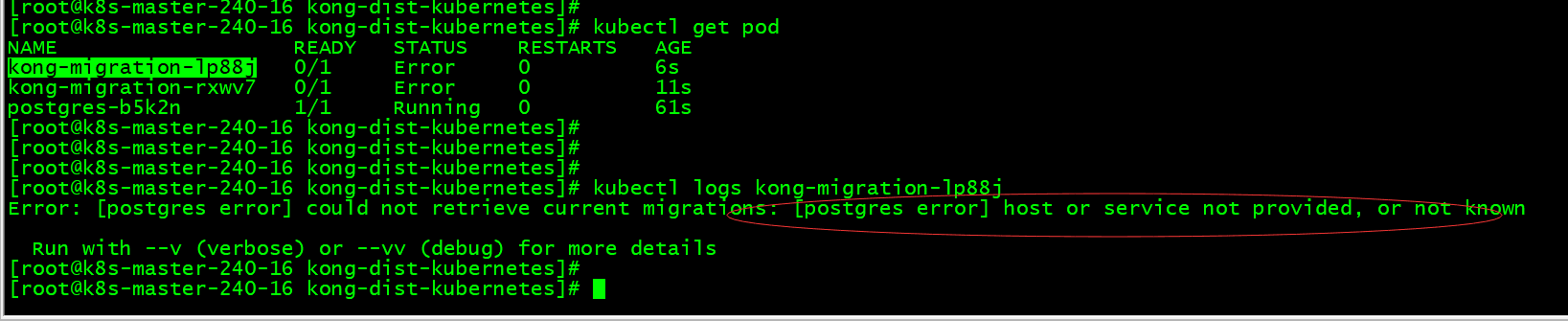


# 四、排障及总结

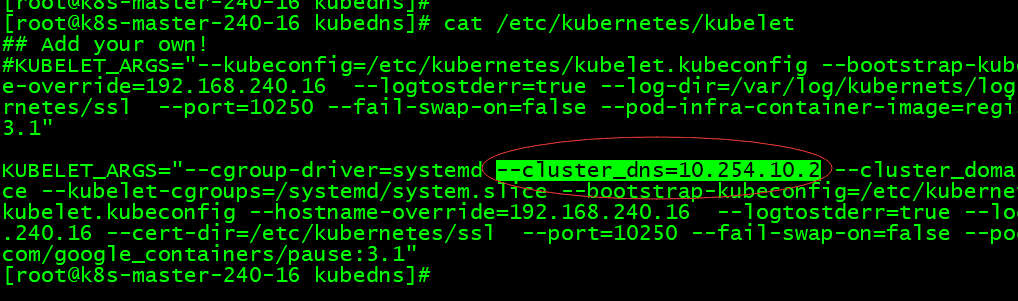
在kong部署过程中，主要遇到了三个问题：

* 在初始化postgres的时候，job一直起不来，通过kubectl logs报错提示DNS问题，后来发现是kubelet组件的配置有问题没有指定





添加如下配置就ok

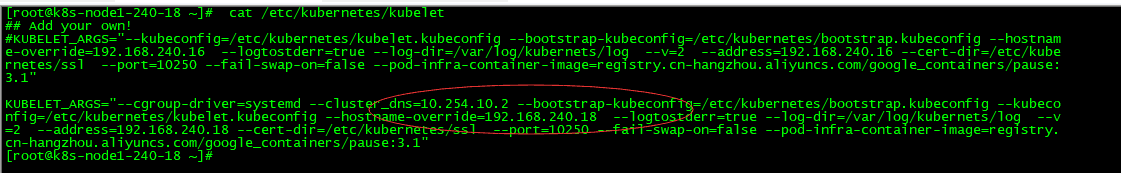


* 在启动kong后，三个副本有两个异常，一个正常，经过再三排查，确定集群中两个节点有问题，因此改yaml文件将kong调度到一个node节点后正常。

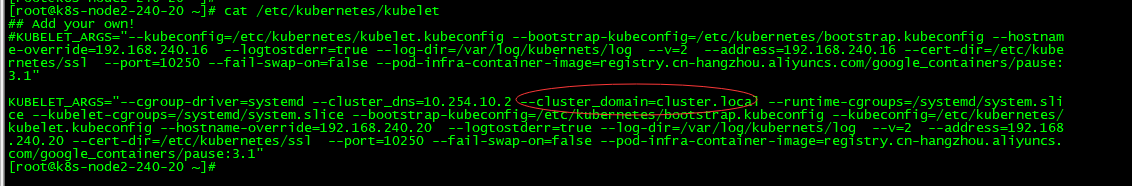
经过再三排查，另外两个节点的问题找到了，还是kubelet配置文件里面dns没有配置好，将dns配置好，然后将kong\_postgres.yaml里面的nodeselector去掉。



更正前：

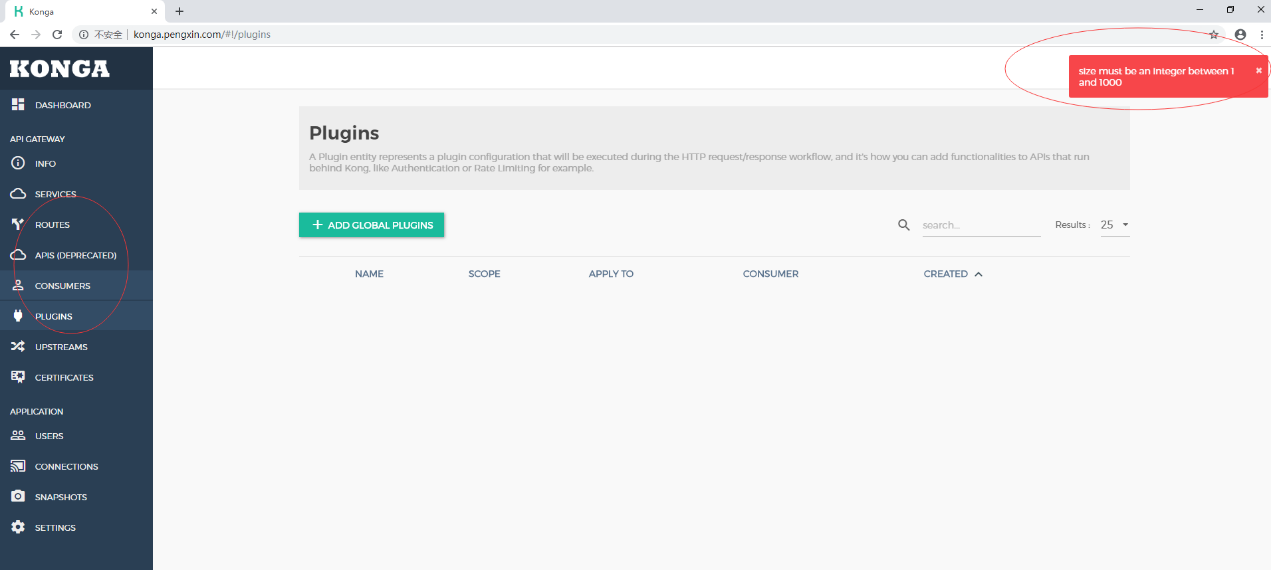


更正后：



* Konga部署好后，登录web界面，点相关菜单有报错如下，于是在github上提交issue，得到了konga作者的回复，建议将konga版本升级，从0.13.0升级到0.14.1后解决。

https://github.com/pantsel/konga/issues/390



参考文献：

<https://www.cnblogs.com/geekmao/p/9100350.html>

<https://github.com/Kong/kong>

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

<https://github.com/pantsel/konga>

<https://github.com/pantsel/konga/tree/master/charts/konga>

参数说明：

Kong将其所有数据（如API，用户和插件）存储在Cassandra或PostgreSQL中。 属于同一集群的所有Kong节点必须连接到同一个数据库。

　　　　database：配置此节点来指定KONG使用哪个数据库（PostgreSQL或Cassandra）作为其数据存储。可选的数据库只有postgres和cassandra。默认为 postgres。

　　　　　　Postgres的设置：

　　　　　　　　pg\_host：Postgres的服务器的主机地址

　　　　　　　　pg\_port：Postgres的服务器的端口

　　　　　　　　pg\_user：Postgres用户名

　　　　　　　　pg\_password：Postgres的用户密码

　　　　　　　　pg\_database：要连接的数据库实例名，必须存在

　　　　　　　　pg\_ssl：是否启用与服务器的SSL连接

　　　　　　　　pg\_ssl\_verify：如果启用了pg\_ssl，则切换服务器证书验证。请参阅lua\_ssl\_trusted\_certificate设置。

　　　　　　Cassandra的设置：

　　　　　　　　cassandra\_contact\_points：集群名称列表，以逗号分隔

　　　　　　　　cassandra\_port：您的节点正在监听的端口

　　　　　　　　cassandra\_keyspace：您在群集中使用的密钥空间，如果不存在将被自动创建

　　　　　　　　cassandra\_consistency：设置读写操作的一致性

　　　　　　　　cassandra\_timeout：读写操作的超时设定，单位为毫秒ms

　　　　　　　　cassandra\_ssl：配置启用SSL连接

　　　　　　　　cassandra\_ssl\_verify：如果启用cassandra\_ssl，则切换服务器证书验证。请参阅lua\_ssl\_trusted\_certificate设置

　　　　　　　　cassandra\_username：使用PasswordAuthenticator方案时的用户名

　　　　　　　　cassandra\_password：使用PasswordAuthenticator方案时的用户密码

　　　　　　　　cassandra\_consistency：读/写Cassandra群集时使用的一致性设置

　　　　　　　　cassandra\_lb\_policy：在Cassandra群集中分发查询时使用的负载均衡策略。接受的值是RoundRobin和DCAwareRoundRobin。当且仅当您使用多数据中心集群时方可配置，此时，请同时配置cassandra\_local\_datacenter选项

　　　　　　　　cassandra\_local\_datacenter：当使用DCAwareRoundRobin策略时，必须在KONG节点中指定本地集群的名称

　　　　　　　　cassandra\_repl\_strategy：如果是首次创建密钥空间，请指定复制策略

　　　　　　　　cassandra\_repl\_factor：指定SimpleStrategy的复制条件

　　　　　　　　cassandra\_data\_centers：指定NetworkTopologyStrategy的数据中心

Kong侦听四个端口的请求，默认情况是：

　　8000：此端口是Kong用来监听来自客户端的HTTP请求的，并将此请求转发到您的上游服务。这也是本教程中最主要用到的端口。

　　8443：此端口是Kong监听HTTP的请求的端口。该端口具有与8000端口类似的行为，但是它只监听HTTPS的请求，并不会产生转发行为。可以通过配置文件来禁用此端口。

　　8001：用于管理员对KONG进行配置的端口。

　　8444：用于管理员监听HTTPS请求的端口。

　　在本文中，我们将介绍Kong的路由功能，并详细说明8000端口上的客户端请求如何根据请求头、URI或HTTP被代理到配置中的上游服务。

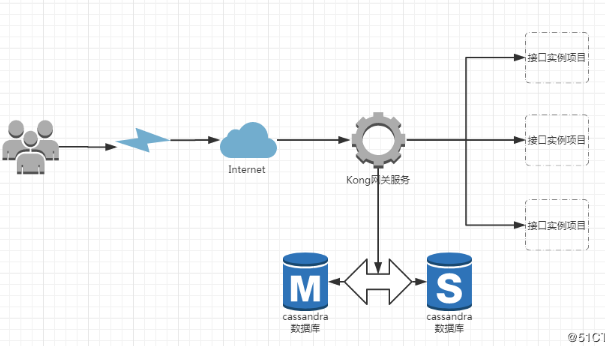
**基础术语**：

　　API：指Kong的API实例。您可以通过管理员身份配置您的API。

　　Plugin：他指的是Kong的“插件”，它们是在代理生命周期中运行的业务逻辑。可以通过管理员身份进行全局配置，也可针对每个API进行分别配置。

　　Client：指向Kong的代理端口发出请求的下游客户，即第三方客户端。

Upstream service：指的是位于Kong后面的您自己的API服务，客户端请求被转发的最终目的地。nginx中的上游服务器。

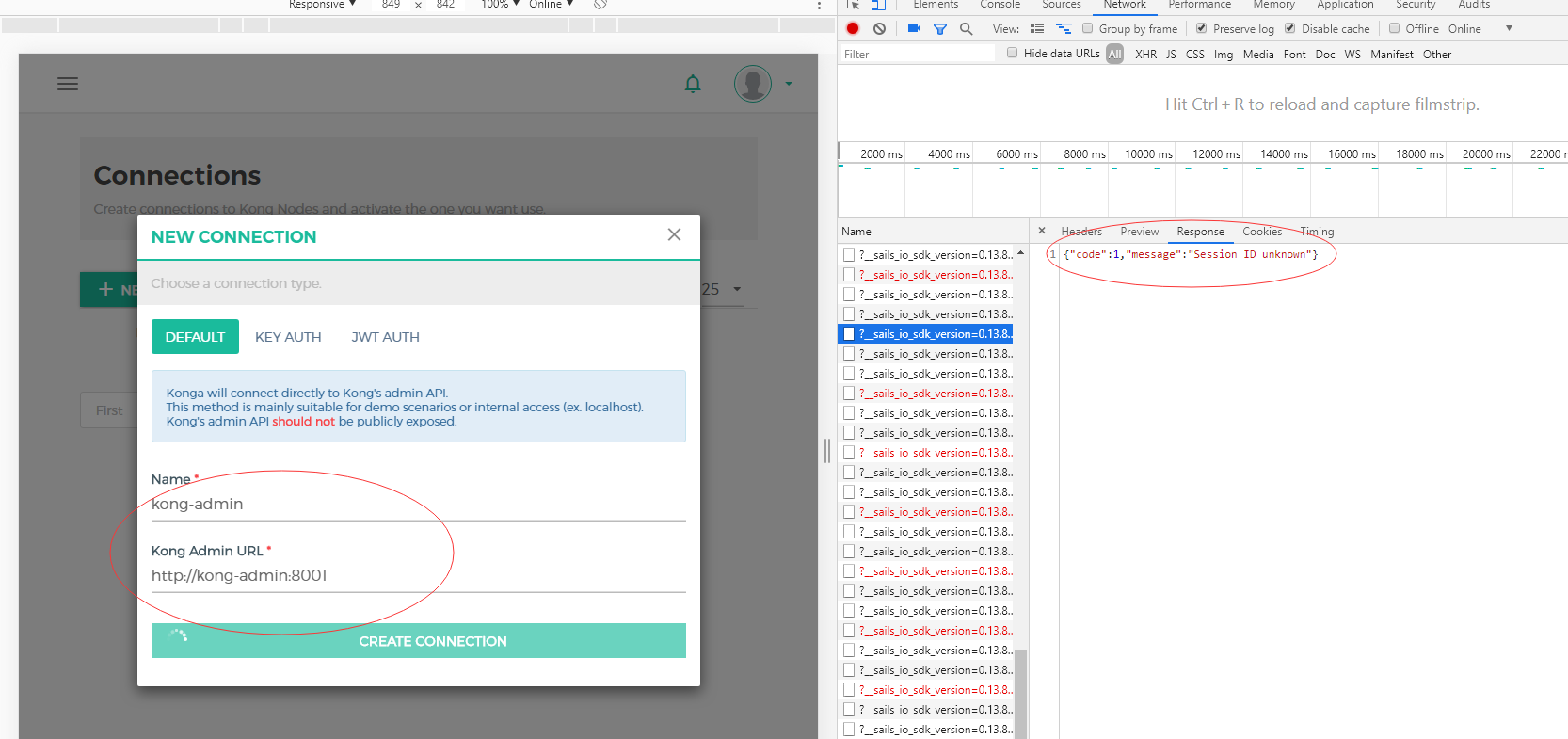


云客预发布：



云客预发布部署遇到的坑：

Konga连接kong-admin的时候一直报错，如下：



在konga的ingress里面增加回话保持的配置就好了