**Kong 1.0 升级解读**

#### [GA Version：](https://konghq.com/blog/kong-1-0-ga/)

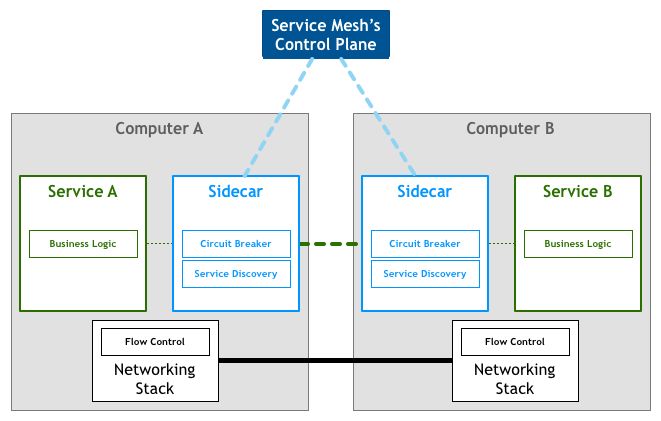
#### ****Service Mesh****

In 1.0, users can now deploy Kong not only as an API gateway but also as a standalone [service-mesh](https://konghq.com/solutions/service-mesh/) proxy. Kong plugins provide key functionality for service mesh out of the box and integrations with other cloud-native technologies including Prometheus, Zipkin, health checks, canary, blue-green and much more.

基于插件化的扩展能力，引入独立的Service-mesh代理来作为Api-GW的选项，更好的支持Cloud-native Tech Stack，相关插件包括Prometheus、Zipkin、健康状态检查、金丝雀测试以及蓝绿测试等等。

[Service Mesh：](http://philcalcado.com/2017/08/03/pattern_service_mesh.html)

# 竟品：**[Istio](https://istio.io/zh/)**



#### ****Mutual TLS (mTLS) and TCP****

In 1.0, the Kong cluster creates a Certificate Authority which Kong nodes can use to establish mutual TLS authentication with each other. Additionally, Kong can now route raw TCP traffic which means Kong can now balance traffic from mail servers and other TCP-based applications, all the way from L7 to L4.

Kong，目前继续扩展Nginx 实现对原始TCP（TLS：Transport Layer Security Protocol位于TCP上层） 流量进行路由，可供其在Kong集群内创建 Certificate Authority （基于TLS 层），实现了对基于TCP协议的应用进行负载均衡。

#### ****gRPC (basic)****

Kong 1.0 now supports primitive, passive, proxy-pass of gRPC traffic in addition to REST. For this initial iteration, no Kong plugins can be applied to manipulate gRPC traffic. gRPC is built on top of HTTP/2, and this initial support provides another option for Kong users looking to connect east-west traffic with low overhead and latency. This is particularly helpful in enabling Kong users to open more mesh deployments in hybrid environments.

gRPC是由Google主导开发的RPC（Remote Process Call Protocol）框架，要解决的问题有Call ID映射（函数指针），序列化和反序列化（protobuf），网络传输（Http2）。目的是低成本，低延迟地实现东西流量地连接，更好支持Service Mesh 在混合模式下的应用。

#### ****New Migrations Framework****

Kong 1.0 introduces a new Database Abstraction Object (DAO), which eases migrations from one database schema to another with near-zero downtime. The new DAO allows users to upgrade their Kong cluster all at once, without requiring manual intervention to upgrade each node.

属于Kong 性能上的重大优化，引入新的DAO实现，简化了从一种数据库模式的到另一种数据库模式的歉意流程。使得停机时间几乎为0，并能够一次性完成Kong 集群的升级，无需手动干预。

#### ****Plugin Development Kit (********PDK)****

The PDK is a set of Lua functions and variables that can be used by custom-plugins to implement their own logic on Kong. Though it was released in 0.14.0, changes in 1.0 fulfill the promise that plugins built with the PDK will be compatible with Kong versions 1.0 and higher.

PDK 是为插件开发提供的一个SDK，进行了些更新，具体看文档。

#### ****100+ Features and Fixes****

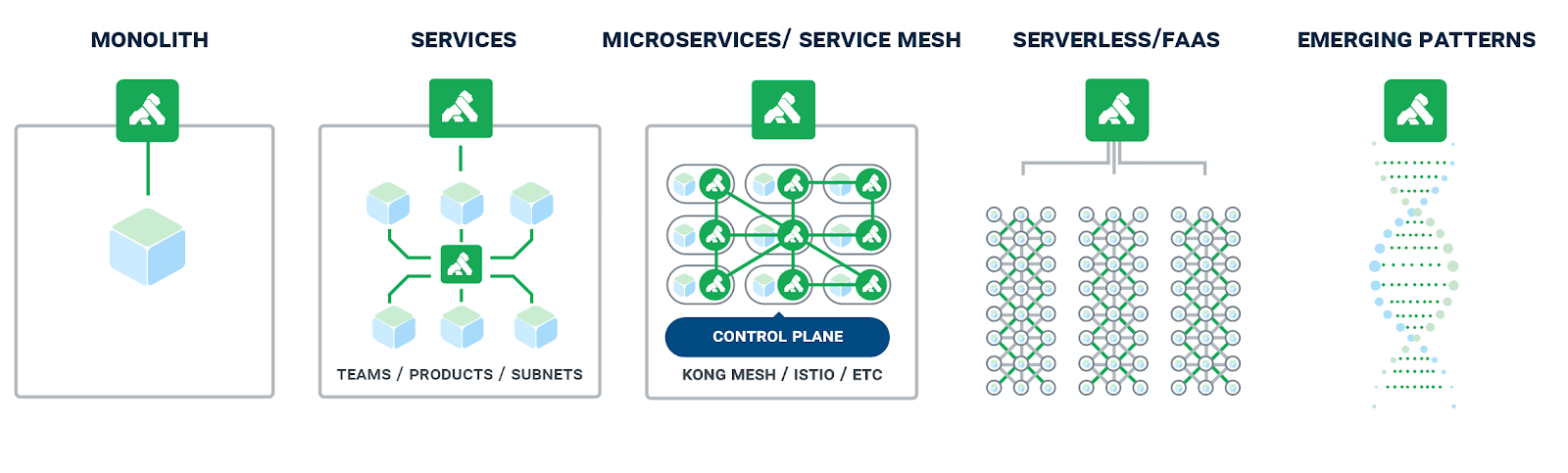
You can find the full list of changes in all Kong releases in the [Changelog](https://github.com/Kong/kong/blob/master/CHANGELOG.md#100). There are a number of breaking changes in this release, so please be sure to read the suggested [upgrade path](https://github.com/Kong/kong/blob/master/UPGRADE.md#upgrade-to-100) for 1.0.

吹个牛逼。

**Vision**

**平台架构进行了重大调整，未来将以Cloud Native的混合环境下支持Service Mesh 能力作为方向。**

*“Kong was built with the vision of a hybrid world in mind, and Kong 1.0 represents a critical step towards that vision. Together with our Community, we’ve made key changes to the architecture of the platform, including the ability to support service mesh, that will give our users the ability to handle any deployment across vendors, environments, and ecosystems. Moving forward, be assured that we’re deepening our commitment to support even more emerging ecosystems”, – Kong CTO, Marco Palladino.*



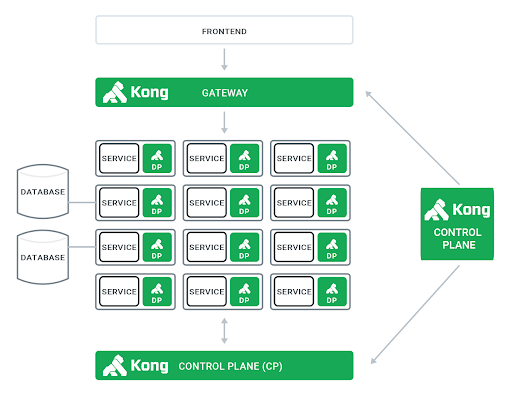
1，引入Sidecar 代理模式， 在实现混合环境下，支持网格当中的服务，接入来自各类环境、平台以及供应商的其它服务。利用Kong，用户能够桥接云原生设计以及传统架构模式之间的鸿沟，且完全无需变更服务代码。

2，天生强大的插件化支持，带来的灵活性和关键功能的开箱即用，以及陆续开发的支持同其它云原生技术的无缝集成的插件，具体包括Prometheus以及Zipkin等等。

3，基于Nginx ，Openresty的性能优化，以及Kong代码的不断演进，实现服务间更低延迟。

4，Kong对于Service Mesh的支持，是建立在各Kong实例之间相互传输层安全性（简称TLS）以及对插件运行循环（RunLoop）的修改基础之上的。正是这些调整，使得Kong项目能够对服务中的每个实例进行逐一部署、在服务之间进行信息代理，同时随着这些服务的扩展而自动扩展。Kong集群会建立一个证书颁发机构，Kong节点将利用它来彼此建立相互TLS身份验证机制。

### 6，数据平面与控制平面相互分离，在以往的版本当中，大家需要单独配置各个集群的数据与控制平面，但现在用户已经可以立足单一中心位置进行变更，且该位置将能够反映在多个Kong集群当中。这种数据与控制平面彼此独立的配置方式，允许Kong用户更好地控制大型部署体系。



[Upgrade guide](https://docs.konghq.com/1.0.x/upgrading/)

This version introduces **a new schema format for plugins**, **changes in Admin API endpoints**, **database migrations**, **Nginx configuration changes**, and **removed configuration properties**.

In this release, the **API entity is removed**, along with its related Admin API endpoints.

#### ****Content：****

**1,** **Preliminary Checks**

**2，****Breaking changes**  should be aware of when upgrading

**3，Steps** to obtain a **no-downtime** migration in different upgrade scenarios.

### **1. Preliminary Checks**

If your cluster is running a version lower than 0.14, you need to upgrade to 0.14.1 first instead. Upgrading from a pre-0.14 cluster straight to Kong 1.0 is **not** supported.

If you still use the deprecated API entity to configure your endpoints and upstream services (via /apis) instead of using Routes for endpoints (via /routes) and Services for upstream services (via /services), now is the time to do so. Kong 1.0 will refuse to run migrations if you have any entity configured using /apis in your datastore. Create equivalent Routes and Services and delete your APIs. (Note that Kong does not do this automatically because the naive option of creating a Route and Service pair for each API would miss the point of the improvements brought by Routes and Services; the ideal mapping of Routes and Services depends on your microservice architecture.)

If you use additional plugins other than the ones bundled with Kong, make sure they are compatible with Kong 1.0 prior to upgrading. See the section above on Plugins for information on plugin compatibility.

## **2. Breaking Changes**

### **Configuration**部分：

a，使用 plugins代替 custom\_plugins，新的指令不仅能启用新的插件还能卸载不需的kong自带插件。

b，指令cassandra\_lb\_policy 的默认值从 RoundRobin 改到了RequestRoundRobin

c，Nginx 的配置文件发生了更新，如果使用了自定义配置模版需要进行更新升级。

### **Core**部分：

a，API 实体以及相关的概念全部被移除 比如/apis endpoints。

新的服务模式改为 Routes 实体（用来配置供Consumer 消费的endpoinst）和Services（用来配置Pulisher 部署的上游服务） 实体的组合。

b，老版本代码中的 kong.dao 全部移除。这个改动会影响到Plugins, Upstreams and Targets这些实体，以及老版本自定义插件开发中使用的DAO实现。

### **Plugins**部分：

主要是Plugin Development Kit (PDK)的升级

**Api-GW 演进：**

1. 重构现有Api-GW的打包，部署，插件安装/启用等部分，考虑使用CRD，helm等工具来工程化运作。添加Infra 的Unit测试，Integet测试等。
2. 插件开发模式的重构，集成测试，Unit测试，CI/CD化。使用CRD 改变插件的投产。
3. Service Mesh Pattern的探索。