# API7 网关技术白皮书

**（版本：2022-02）**

## 一. 整体介绍

支流科技 API 网关产品（以下简称 API7）是基于 Apache 软件基金会顶级项目 Apache APISIX 构建的，包含了 API 网关、ManagerAPI 与 Dashboard 控制面板 3 个组件。

API 网关作为微服务架构中重要组件，是流量的核心出入口，用于统一处理和业务相关的请求，可有效解决海量请求、恶意访问等问题，以保障业务安全性与稳定性。

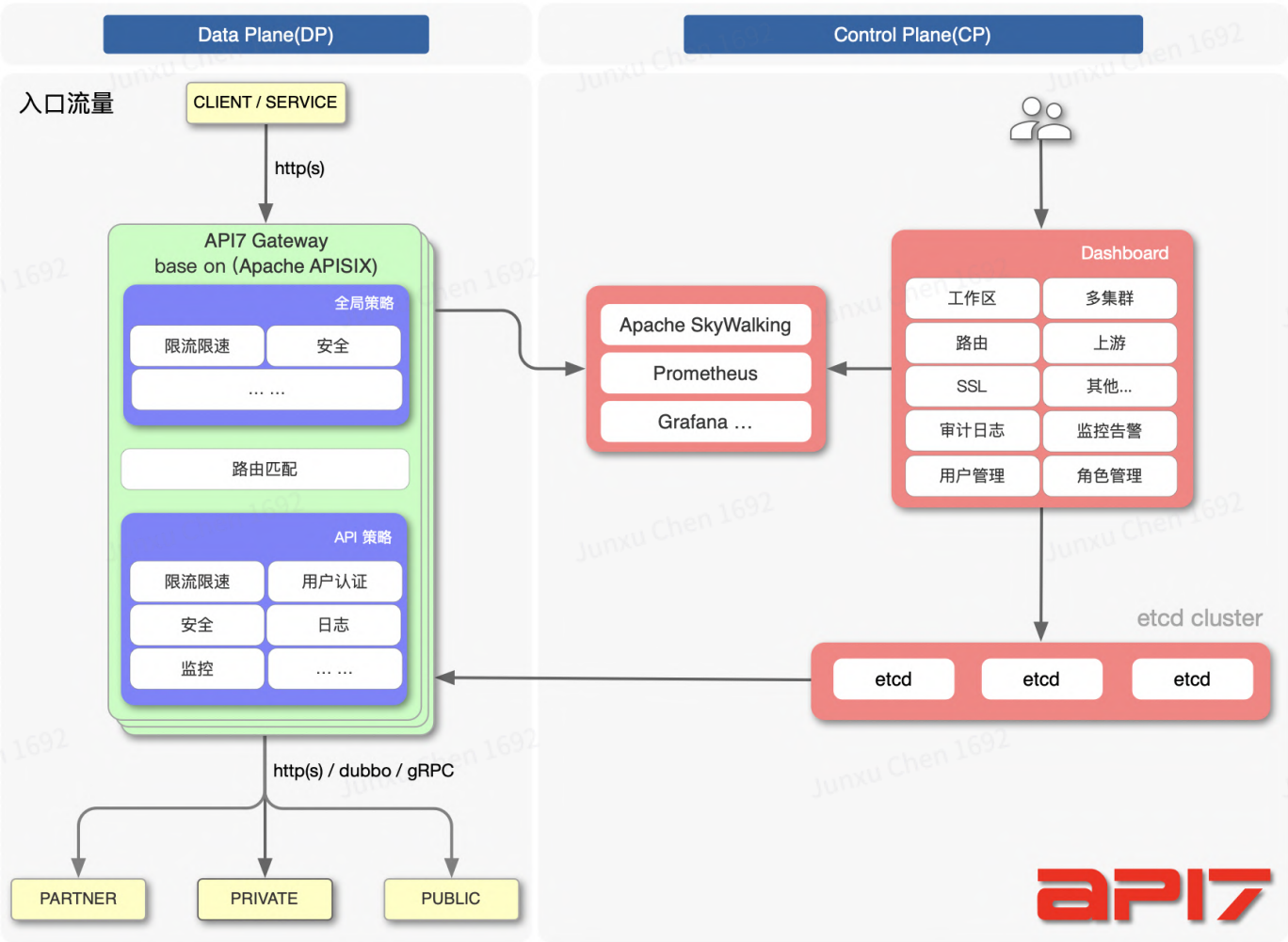


**图 1-1 API7 架构图**

上图为 API7 产品中控制平面（简称 CP）与数据平面（简称 DP）的架构示意图，并包含了3个部分：

### 1. API 网关

用于承载并处理业务流量，管理员在配置路由规则后，网关将根据预设规则将请求转发至上游服务。此外，借助 API7 内置的 50 多种插件，可实现身份验证、安全防护、流量控制、分析监控、请求/响应

转换等常见业务需求；若内置插件无法满足需求，我们也支持使用 **Lua**、Java、Go、Python 语言自定义插件，可作用于请求进入、上游响应各个阶段。

2. **Manager API**

用于管理 API 网关，通过访问其暴露的 RESTful API 接口以实现对路由、上游、证书、全局插件、消费者等资源的管理。

3. **控制面板**

为了简化网关管理，管理员可以通过 Dashboard 控制面板以可视化形式操作网关，支持监控分析、日志审计、多租户管理、多集群切换、多工作分区等能力。

## 1.1 技术架构

1. **数据平面**

数据平面用于接收并处理调用方请求，使用 Lua 与 Nginx 动态控制请求流量。当请求进入时，将根据预设路由规则进行匹配，匹配到的请求将被网关转发至对应上游服务。在此过程中，网关有能力根据预设规则中不同插件的配置，使用一系列插件对请求从进入到离开的各个阶段进行操作。例如：请求可能会经过身份认证（避免重放攻击、参数篡改等）、请求审计（请求来源信息、上游处理时长等）、路由处理（根据预设规则获取最终上游服务地址）、请求转发（网关将请求转发至上游目标节点）、请求响应（上游处理完成后，网关将结果返回给调用方）等几个步骤。

2. **控制平面**

控制平面包含了 ManagerAPI 与默认配置中心 ETCD。管理员在访问并操作控制台时，控制台将调用 ManagerAPI 下发配置到 ETCD，借助 ETCD Watch 机制，配置将在网关中实时生效。例如：管理员可增加一条路由，并配置限速插件，当触发到限速阈值后，网关将会暂时阻止后续匹配到该路由的请求进入。借助 ETCD 的 Watch 机制，当管理员在控制面板更新配置后，API7 将在毫秒级别内通知到各个网关节点。

3. **其它**

从图 1-1 可见，API7 采用了数据平面与控制平面分离的架构方式，通过配置中心接收、下发配置，使得数据平面不会受到控制平面影响。配置中心默认为 ETCD，但也支持 Consul、Nacos、Eureka 等，可根据您的实际情况进行选择。此外，企业用户只需关注业务本身，与业务无关的大部分功能交给 API7 内置插件即可实现，如身份验证、性能分析等。

## 1.2 技术亮点

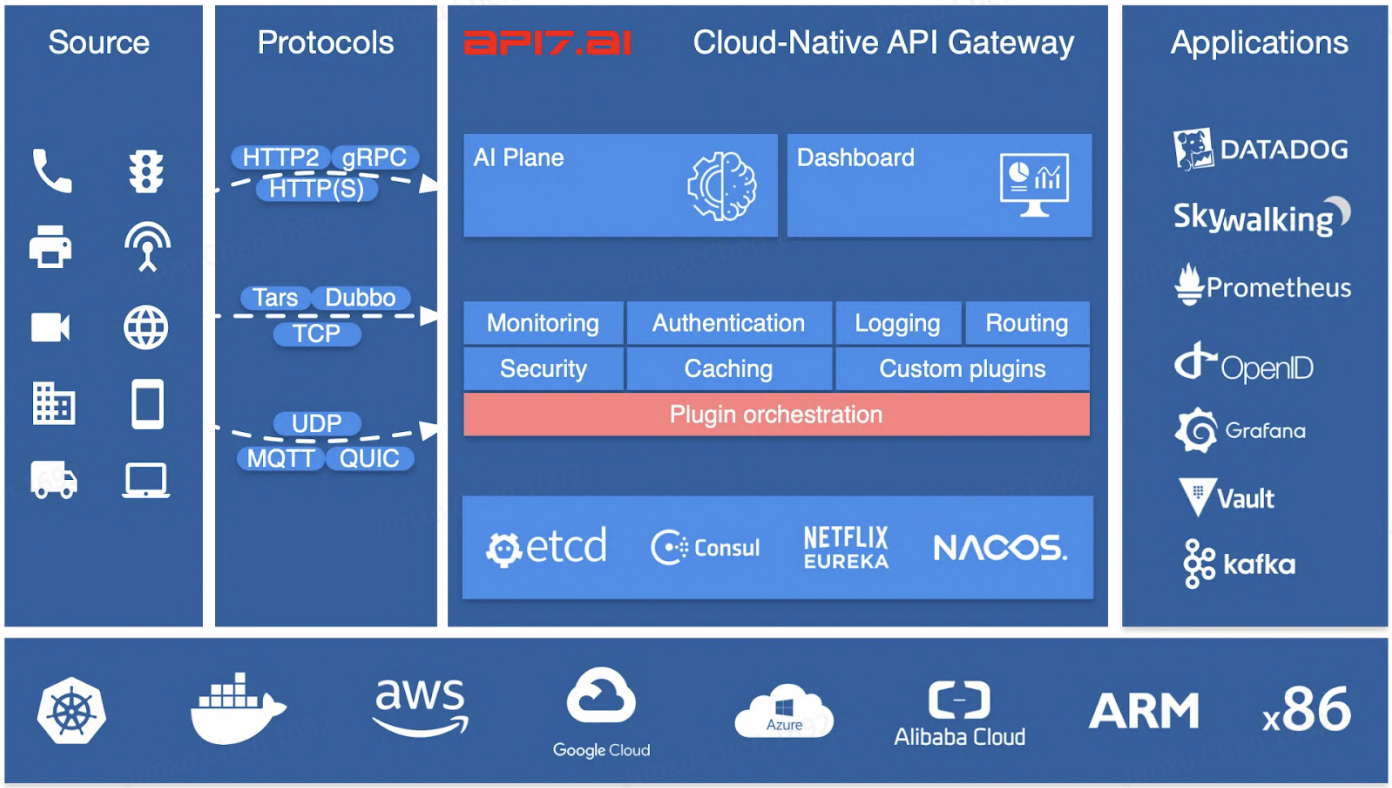


**图 1-2 API7 技术亮点**

1. **云原生**

API7 是一个云原生网关，与平台无关，没有供应商锁定的风险。它支持裸金属、虚拟机、Kubernetes、OpenShift、ARM64 等。此外，API7 也可轻松与其它组件对接，如 SkyWalking、Prometheus、Kafka、Zipkin 等，共同为企业赋能；

2. **高可用**

API7 默认选用 ETCD 作为配置中心，ETCD 天然支持分布式、高可用，并且在 K8s 等领域有大量实践经验，使得 API7 可以轻松支持毫秒级配置更新、支撑数千网关节点；网关节点无状态，可任意扩容或缩容；

3. **协议转换**

支持丰富的协议类型，如 TCP/UDP、Dubbo、MQTT、gRPC、SOAP、WebSocket 等；

4. **安全防护**

内置多种身份验证与安全防护能力，如 Basic Auth、JSON Web Token、IP 黑白名单、OAuth 等；

5. **性能极高**

API7 使用 Radixtree 算法实现高性能、灵活路由，在 AWS 8 核心服务器中，QPS 约为 140K，延迟约为 0.2 ms；

6. **全动态能力**

修改网关配置、增加或修改插件等，无需重启网关服务即可实时生效；支持动态加载 SSL 证书；

7. **扩展能力强**

借助灵活的插件机制，可针对内部业务完成功能定制；支持自定义负载均衡算法与路由算法，不受限于 API 网关实现；通过运行时动态执行用户自定义函数方式来实现 Serverless，使网关边缘节点更加灵活；

8. **治理能力丰富**

如故障隔离、熔断降级、限流限速等；在启用主动健康检查后，网关将支持智能跟踪不健康上游节点的能力，并自动过滤不健康节点，以提高整体服务稳定性。

## 1.3 功能架构

**API 网关主要包含了如下功能模块：**

- **用户系统：**借助用户系统，管理员将对系统内各个用户进行权限资源划分，用户不可越权访问资源。支持账户密码登陆与 SSO 登陆；
- **权限系统：**内置基于角色的权限管理系统（RBAC），管理员可借助控制台创建不同角色，通过将用户与角色绑定，可实现针对某类用户细粒度的权限管控；
- **多租户（多工作分区）：**支持基于工作分区隔离的多租户模型，管理员可创建不同的工作分区，并指定哪些用户对工作分区有哪些资源的访问权限；
- **多环境：**支持多 ETCD 集群，集群之间数据不共享；
- **身份验证：**包含多种认证类插件，如 basic-auth、jwt-auth、key-auth、wolf-rbac 等。此外，借助内置的 HMAC 插件，可使用 AK/SK 对请求参数进行签名与校验，以实现请求防篡改、请求防重放的需求，并能够达到鉴权的目的；
- **服务路由：**API7 基于 Radixtree 实现高效的路由匹配，是目前匹配路由速度最快的 API 网关。它支持全路径匹配、前缀匹配，也支持使用 Nginx 内置变量作为匹配条件，以此实现精细化路由。此外，API7 支持流量镜像与高级路由匹配功能，可实现灰度发布等精细化路由管理功能。此外，它也支持服务发现与多种注册中心，并有能力根据请求中 Header、Query、Cookie 等参数进行分流；
- **协议转换：**API7 支持丰富的协议，如 TCP/UDP、Dubbo、MQTT、gRPC、WebSocket 等，并能够实现 HTTP 协议到后端服务其它协议的转换。API 网关对外暴露统一 HTTP 入口，管理员可通过控制台界面完成协议转换设置，支持请求与后端服务的参数映射；
- **服务治理：**API7 支持熔断、限流、限速、IP 黑白名单、故障隔离等能力，通过控制台可视化面板，可方便、清楚地完成相关功能设置；
- **自定义插件：**API7 内置了50多种插件，涵盖安全防护、流量控制、日志记录等各个分类，可满足绝大多数企业需求。对于特定业务，API7 目前支持 Lua、Java、Go、Python 编写自定义插件，且插件可以作用于流量进出的各个阶段。得益于全动态能力，新增、修改插件无需停机重启，可实时生效，避免中断业务；
- **分析监控：**API7 内置了请求审计、监控告警、统计报表等分析监控功能，API 网关将记录所有节点每个请求的信息，并进行成功请求、异常请求统计，可在控制台查看请求成功数、请求失败数、错误码、请求延迟等指标。此外，借助 Grafana 的能力，可满足更多维度地分析监控需求；
- **全生命周期管理：**API7 支持 API 版本管理、API 分组、API 上下线、在线调试等功能，并兼容 OpenAPI 3.0 标准，实现 API 文档生成、API 导入导出等特性，方便用户进行数据迁移操作。

## 1.4 功能列表

| 分类 | 功能模块 | 功能点 | API7 | Kong | Zuul 2 | Nginx | Spring Cloud Gateway |
|---|---|---|---|---|---|---|---|
| API 和服务治理 | 协议支持 | HTTP/1.1、HTTP 2 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | | HTTP/3 | ✅ | ✅ | ❌ | ❌ | ✅ |
| | | TLS / HTTPS | ✅ | ✅ | ✅ | ✅ | ✅ |
| | | MQTT | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | TCP | ✅ | ✅ | ❌ | ✅ | ❌ |
| | | UDP | ✅ | ✅ | ❌ | ✅ | ❌ |
| | | HTTP gRPC/Dubbo 协议转换 | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | Websocket | ✅ | ✅ | ✅ | ✅ | ✅ |
| | | Dubbo | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | 自定义四层、七层协议 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | 平台支持 | 裸金属 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | | 虚拟机 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | | Kubernetes | ✅ | ✅ | ✅ | ✅ | ✅ |
| | | ARM64 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | | 鲲鹏（通过华为云认证） | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | AWS、GCP、阿里云、腾讯云等公有云 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | 精细化路由 | URI 参数匹配 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | | HTTP 请求头匹配 | ✅ | ✅ | ❌ | ✅ | ✅ |
| | | HTTP 请求方法匹配 | ✅ | ✅ | ❌ | ✅ | ✅ |
| | | 支持所有 Nginx 变量匹配 | ✅ | ❌ | ❌ | ✅ | ❌ |
| | | 支持 Lua 代码段实现自定义匹配 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | 支持条件表达式 | ✅ | ❌ | ❌ | ❌ | ✅ |
| | | 支持IPv6 | ✅ | ✅ | ✅ | ✅ | ✅ |

| | | | | | | |
|---|---|---|---|---|---|---|
| | GeoIP 地理位置匹配 | ✅ | ✅ | ❌ | ❌ | ❌ |
| | 路由存活时间（TTL） | ✅ | ❌ | ❌ | ❌ | ❌ |
| | 优先级匹配 | ✅ | ✅ | ✅ | ❌ | ✅ |
| **负载均衡** | Round Robin | ✅ | ✅ | ✅ | ✅ | ✅ |
| | Weighted Round Robin | ✅ | ✅ | ✅ | ✅ | ✅ |
| | Chash（一致性哈希） | ✅ | ✅ | ❌ | ✅ | ❌ |
| | Sticky Session（会话保持） | ✅ | ✅ | ❌ | ❌ | ❌ |
| | Least Connections | ✅ | ✅ | ❌ | ✅ | ❌ |
| | EWMA | ✅ | ❌ | ❌ | ❌ | ❌ |
| | 支持自定义负载均衡算法 | ✅ | ❌ | ✅ | ❌ | ✅ |
| **请求改写** | URI 改写 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | 新增、修改和删除 HTTP 请求头 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | 301、302 重定向 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | 强制跳转到 HTTPS | ✅ | ✅ | ❌ | ✅ | ❌ |
| **响应改写** | 新增、修改和删除 HTTP 响应头 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | 修改 HTTP 响应码 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | 修改响应体 | ✅ | ✅ | ✅ | ✅ | ✅ |
| **服务发现和注册** | 默认 ETCD，并支持 ETCD 集群 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | Consul | ✅ | ❌ | ❌ | ❌ | ✅ |
| | Eureka | ✅ | ❌ | ✅ | ❌ | ✅ |
| | Nacos | ✅ | ❌ | ❌ | ❌ | ✅ |
| | Redis | ✅ | ❌ | ❌ | ❌ | ❌ |
| | 限流 / 集群限流 | ✅ | ✅ | ❌ | ❌ | ✅ |
| | 限速 | ✅ | ✅ | ❌ | ✅ | ✅ |
| | 限制并发 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | 上游主动健康检查 | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **容错和降级** | 上游被动健康检查 | ✅ | ✅ | ❌ | ✅ | ❌ |
| | | 服务熔断 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | | 服务降级 | ✅ | ✅ | ✅ | ❌ | ✅ |
| | | API 熔断 | ✅ | ✅ | ❌ | ❌ | ✅ |
| | | 超时 | ✅ | ✅ | ✅ | ✅ | ✅ |
| | **流量管理** | 灰度发布 | ✅ | ✅ | ❌ | ✅ | ❌ |
| | | 蓝绿发布 | ✅ | ✅ | ❌ | ✅ | ❌ |
| | | 流量镜像 | ✅ | ❌ | ❌ | ✅ | ❌ |
| | | 故障注入 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | **API 管理** | 多 API 聚合 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | 版本管理 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | 上线和下线 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | Swagger 和 OpenAPI | ✅ | ❌ | ❌ | ❌ | ✅ |
| | | 生成 SDK 和文档 | ✅ | ❌ | ❌ | ❌ | ✅ |
| | **插件管理** | 动态新增、修改和删除插件 | ✅ | ❌ | ✅ | ❌ | ❌ |
| | | 插件编排（低代码） | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | 支持 Lua、Java 和 Go 编写自定义插件 | ✅ | ✅ | ❌ | ❌ | ❌ |
| **安全** | **用户相关** | RBAC | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | 多租户 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | 多工作分区 | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | SSL 证书管理 | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | 通过 Admin API Key 和 IP 限制来控制访问 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | **通信加密** | mTLS | ✅ | ✅ | ✅ | ✅ | ❌ |
| | | SSL 证书自动轮转 | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | 支持国密 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | IP 黑白名单 | ✅ | ✅ | ❌ | ✅ | ❌ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 防止攻击 | URI 黑白名单 | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | 防 ReDOS 攻击 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | 防重放攻击 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | 身份认证 | key-auth | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | basic-auth | ✅ | ✅ | ❌ | ✅ | ❌ |
| | | JWT | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | API 签名校验（HMAC） | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | OAuth2 | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | SSO | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | 对接 Auth0、Okta 等 | ✅ | ✅ | ❌ | ❌ | ✅ |
| 可观测性 | Metrics | Prometheus | ✅ | ✅ | ❌ | ❌ | ✅ |
| | Tracing | SkyWalking | ✅ | ❌ | ✅ | ❌ | ✅ |
| | | Zipkin | ✅ | ✅ | ✅ | ❌ | ✅ |
| | | OpenTracing | ✅ | ✅ | ✅ | ❌ | ✅ |
| | 日志 | Kakfa | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | HTTP Logger | ✅ | ❌ | ❌ | ✅ | ❌ |
| | | TCP Logger | ✅ | ❌ | ❌ | ✅ | ❌ |
| | | UDP Logger | ✅ | ❌ | ❌ | ❌ | ❌ |
| 集群和高可用 | QPS | 单核心性能 | 极高 | 高 | 低 | 极高 | 低 |
| | 延迟 | 每请求最低延迟 | 极好 | 中 | 低 | 极好 | 低 |
| | 部署 | 数据平面无状态 | ✅ | ✅ | ✅ | ❌ | ✅ |
| | | 配置中心支持集群 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | 集群管理 | 支持多集群的配置和管理 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | 支持不同集群间权限隔离 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | 全球化部署，跨网关集群协作 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | 拓扑网络下自动选择最优路径 | ✅ | ❌ | ❌ | ❌ | ❌ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 多层网络 | 自定义多层网络下插件 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | | 分离式部署，支持原生开源版本 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | 动态和热更新 | 所有变更都是热更新，实时生效 | ✅ | ❌ | ❌ | ✅ | ❌ |
| | | 插件热更新 | ✅ | ❌ | ✅ | ❌ | ❌ |
| | | 程序自身热更新 | ✅ | ✅ | ❌ | ✅ | ❌ |
| 运维 | CLI | 命令行工具 | ✅ | ✅ | ❌ | ✅ | ❌ |
| | Admin API | 使用 REST API 来控制，方便集成 | ✅ | ✅ | ❌ | ❌ | ❌ |
| | 单机模式 | 用yaml文件来定义所有规则 | ✅ | ✅ | ✅ | ❌ | ✅ |
| | 回滚 | 支持操作的无限回滚 | ✅ | ❌ | ❌ | ❌ | ❌ |
| | Helm charts | 更方便k8s下的运维 | ✅ | ❌ | ❌ | ✅ | ❌ |
| | 全局插件 | 简化操作 | ✅ | ✅ | ✅ | ❌ | ✅ |
| | 健康状态 | 数据平面节点的版本和运行监控 | ✅ | ✅ | ✅ | ❌ | ✅ |
| | | 配置中心状态和版本信息 | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | 节点负载状态监控 | ✅ | ✅ | ❌ | ❌ | ✅ |
| | 服务可观测性 | 服务调用拓扑 | ✅ | ✅ | ❌ | ❌ | ❌ |
| | | 数据吞吐量 | ✅ | ✅ | ✅ | ❌ | ✅ |
| | | 响应时间统计 | ✅ | ✅ | ✅ | ❌ | ✅ |
| | | 上游响应时间统计 | ✅ | ✅ | ✅ | ❌ | ❌ |
| | | 状态码统计 | ✅ | ✅ | ✅ | ❌ | ✅ |
| | | API 调用次数统计 | ✅ | ✅ | ❌ | ❌ | ✅ |

**表 1-1 API7 功能列表**

## 1.5 功能亮点

1. **API 全生命周期管理**

涵盖 API 设计、创建、测试、部署、管理、运维、下线等阶段，可进一步帮助企业优化 API 管理流程、提高企业价值。借助于 OpenAPI 3.0 标准，可方便地完成 API 导入导出以及文档生成工作，更多

地发挥 API 能力。

2. **多租户能力（多工作区）**

API7 支持通过工作分区进行项目隔离，以支持多租户能力。结合用户系统与权限管理，不同用户对不同工作区下的资源有不同权限，可精细化地对资源进行权限管控。

3. **多协议转换**

从 Dubbo、gRPC 到 MQTT 物联网协议，API 网关均支持对外统一暴露 RESTful API，减少内部服务协议改造。

4. **全动态能力**

借助于 API 网关全动态能力，从网关配置到插件修改，无需重启服务即可实时生效，避免服务中断影响业务流量而产生无法意料的结果。此外，API7 也支持动态加载 SSL 证书。

5. **自定义插件**

API 网关内置 50 多种常见插件，通过对插件的组合使用可满足常见的绝大多数网关需求。

结合支流科技 API 网关特有的低代码能力，通过绘制流程图的方式将插件进行组合，可实现更高级地插件使用方式。

此外，若现有插件无法满足您的特殊需求，通过使用简单易懂的 Lua 语言，即可快速编写定制插件；它可以作用于请求从进入到响应返回的各个阶段，例如 init、rewrite、access、balancer、header filter、body filter、log 等。

6. **分析监控**

API7 集成 Prometheus 以获取详细 API 调用数据，包括但不限于访问来源、成功率、95 值、99 值、成功/失败响应码分布、QPS 等指标。

7. **Dashboard**

内置控制面板与 ManagerAPI，前者方便用户通过可视化面板进行规则配置，后者方便用户使用自动化工具或集成到内部业务中，以控制网关节点。

# 二. 功能摘要介绍

## 2.1 插件列表

API7 内置50多种常用插件，涵盖身份验证、安全防护、流量控制、分析监控、请求/响应转换等多个分类。下表中列出了一些流行的插件。

| 分类 | 名称 | 作用 | 场景 |
| --- | --- | --- | --- |
| | 身份验证插件可以有效地保护 Route、Service，以避免非法、无权限的访问。 | | |
| | **authz-keycloak** | 启用该插件后，将支持配合 KeyCloak 认证服务完成身份鉴权。 | |

| | | | |
|---|---|---|---|
| 身份验证 | basic-auth | 启用该插件后，客户端访问时需使用正确的账户、密码。 | 身份认证 |
| | hmac-auth | 启用该插件后，除了验证客户端身份有效性，其端请求参数也将被签名验证，避免参数被篡改或二次访问（重放攻击）。 | |
| | jwt-auth | 启用该插件后，将使用 JSON Web Token 方式进行有效性验证，客户端访问时，需要在 HTTP 请求头中增加正确的 Token 内容。 | |
| | key-auth | 启用该插件后，在访问资源时，客户端需要在请求头或查询字符串中携带正确的密钥。 | |
| | wolf-rbac | 启用该插件后，网关将支持基于 wolf 的认证及授权特性。 | |
| | openid-connect | 启用该插件后，网关将支持身份验证与令牌自省功能。 | |
| 安全防护 | api-breaker | 启用该插件后，网关将根据配置判断上游是否异常，若异常，则直接返回预设的错误码，一定时间内不再访问上游。 | 熔断 |
| | consumer-restriction | 启用该插件后，若设置了白名单，则白名单外的消费者将被网关拒绝请求；若设置了黑名单，黑名单内的消费者将被网关拒绝请求。 | |
| | cors | 通过启用 CORS 插件，以支持浏览器向服务发出请求。 | |
| | fault-injection | 启用故障注入插件后，将直接返回指定 HTTP 状态码与响应内容给传入的请求，以实现服务维护的需求。 | 故障注入 |
| | ip-restriction | 通过将 IP 地址列入白名单或黑名单来限制对服务的访问，可以设置单个或者多个地址，或以 CIDR 的方式设置 IP 范围。 | |
| | referer-restriction | 启用该插件后，将通过请求头中 Referer 信息，以判断是否需要限制该请求。 | |
| | request-validation | 在网关将请求转发至上游时，该插件使用 JSONSchema 校验请求头与请求体，校验失败的请求将被拒绝。 | |
| | uri-blocker | 启用该插件后，当请求路径匹配到预设规则后，网关将返回指定的状态码。 | |
| 流量控制 | limit-conn | 启用该插件后，将限制请求并发数量。 | 限流限速 |
| | limit-count | 启用该插件后，在一个固定时间窗口内，超过预设值的请求将被拒绝。 | |
| | limit-req | 启用该插件后，将使用漏桶算法限制请求速率。 | |
| | traffic-split | 该插件允许我们动态控制指向不同上游服务的流量比。 | 灰度 |

| | | | |
|---|---|---|---|
| **Serverless** | Serverless 插件在网关 access 阶段动态执行 Lua 代码，以实现无服务环境下执行 FaaS 函数。 | | |
| | **serverless-post-function** | 该插件中配置的函数，将在其它插件之前运行。 | |
| | **serverless-pre-function** | 该插件中配置的函数，将在其它插件之后运行。 | |
| **可观测性** | **error-log-logger** | 该插件将使用 TCP 协议把网关产生的 error.log 文件内容推送至指定的服务器。 | |
| | **http-logger** | 该插件用于将请求数据、响应数据以及上下文信息发送至 HTTP 服务器。 | |
| | **kafka-logger** | 该插件将把日志数据推送至 Kafka。 | |
| | **prometheus** | 该插件将以 Prometheus 的数据格式暴露网关的相关度量指标。 | |
| | **request-id** | 该插件将为网关处理过的每一个请求增加 request-id 请求头，用于标识 API 请求。 | |
| | **skywalking** | SkyWalking 是一个可观测性分析平台，该插件将向 SkyWalking 主动上报数据，方便我们通过 SkyWalking 查看网关状态。 | |
| | **sls-logger** | 该插件用于将请求数据、响应数据以及上下文信息发送至阿里云 SLS 日志服务。 | |
| | **syslog** | 该插件用于将请求数据、响应数据以及上下文信息发送至 Syslog。 | |
| | **tcp-logger** | 该插件用于将 access-log 数据以 TCP 的形式发送至指定服务器 | |
| | **udp-logger** | 该插件用于将 access-log 数据以 UDP 的形式发送至指定服务器，由于 UDP 无需三次握手，因此它传输效率高，具有较好的实时性。 | |
| | **zipkin** | 该插件将向 Zipkin 报告网关时序与跟踪数据，包括但不限于 TraceID、节点信息、请求信息、延迟等，这可以帮助我们通过 Zipkin 定位网关遇到的问题。 | |
| | **batch-requests** | 该插件将支持使用 Pipeline 的形式接收多个请求，并发送到对应上游服务，其响应内容是多个请求的响应内容组合。这对于客户端希望访问多个 API 时很有帮助。 | |

| | | | |
|---|---|---|---|
| 其它 | grpc-transcode | 该插件将支持将 RESTful API 请求发送至 gRPC 上游服务中。 | |
| | proxy-cache | 该插件将支持缓存上游服务响应内容，当客户端所请求的内容已经存在于缓存，则直接从缓存返回内容，无需再次请求上游服务。这将有效减轻上游服务的压力。此外，当上游节点故障时，也可以暂时返回缓存内容，而无需返回错误页，以提升用户体验。 | |
| | proxy-mirror | 该插件支持对请求进行镜像复制，以便更好地进行旁路的请求分析。 | |
| | proxy-rewrite | 客户端发送的请求在到达上游服务前，该插件将按照指定规则对请求进行修改，包含但不限于请求体、请求头、请求路径等参数。 | |
| | response-rewrite | 上游服务的响应在到达客户端之前，该插件将按照指定规则修改响应内容，包含但不限于响应体、响应头等参数。 | |

**表 2-1 API7 常用插件列表**

## 2.2 认证鉴权

API 网关内置了 key-auth、basic-auth、jwt-auth 等认证鉴权插件，以 hmac 插件为例，我们可配合 AK/SK 对请求参数进行加密，以保证请求未被篡改。

请求参数包括 Request Header、Request Path、Request Query String、时间戳、签名算法等，以避免请求被篡改、重放。

## 2.3 灰度发布

路由作为 API 网关的核心功能，用于对经过 API 网关的请求进行路由匹配，并转发至对应的上游服务中。当上游服务完成处理后，结果将被返回至客户端。若请求未匹配到路由时，网关将返回 404 状态码，这是由于路由未被发布到网关或未配置相关路由。

借助 API7 强大的路由功能，也可以实现灰度发布、蓝绿部署的需求，以便企业稳定地进行服务平滑升级。此外，API 网关将请求转发至上游服务时，将携带一些 HTTP 请求头，用于标记这些流量来自于网关。

以灰度发布为例：在开始灰度发布后，首先启动新版本服务（应用），并交给测试人员对新版本进行测试。如果测试正常，那可以将少量的流量切换到新版本中，接着对新版本进行运行状态检查，并收集各种数据。当确认新版本运行良好后，再逐步将更多的流量切换到新版本中。直到将 100% 流量全部切换到新版本中，再关闭旧版本服务，这样便完成了灰度发布。如果在灰度发布过程中发现了新版本有问题，就立即将流量切回旧版本中，这样，负面影响将会控制在最小范围内。

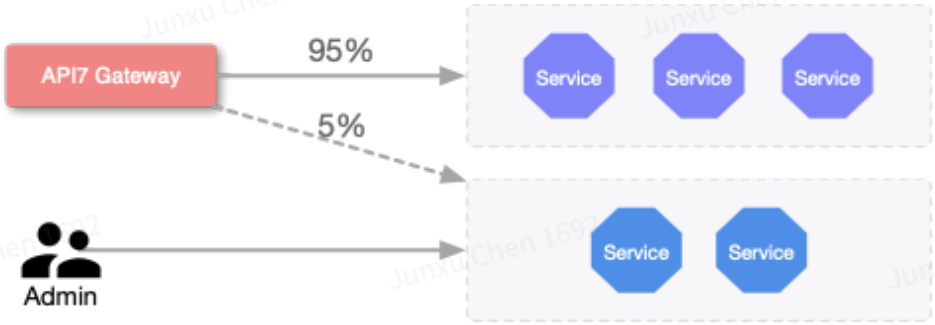


**图 2-1 灰度发布示意图**

## 2.4 服务治理

API7 内置了限流限速、熔断、IP 黑白名单、故障隔离等服务治理功能。

1. **限流限速**

API 网关基于漏桶算法实现了限流限速，内置了 limit-count、limit-req、limit-conn 三个限流限速插件：

| 名称 | 描述 |
| --- | --- |
| limit-count | 基于固定窗口实现限速 |
| limit-req | 基于漏桶原理实现请求限速 |
| limit-conn | 限制并发请求 |

**表 2-2 API7 限流限速插件列表**

以 limit-req 为例，它包含了如下参数：

| 参数名 | 类型 | 必选 | 值范围 | 描述 |
| --- | --- | --- | --- | --- |
| rate | 整数型 | 是 | > 0 | 允许的最大请求速率。大于 rate 但小于 rate + burst 的请求将被延迟处理。单位秒。 |
| burst | 整数型 | 是 | >= 0 | 允许被延迟处理的请求速率。 |
| key | 字符串 | 是 | remote_addr,server_addr,http_x_real_ip,http_x_forwarded_for,consumer_name | 用于限制请求速率的关键字（请求计数依据）。 |
| rejected_code | 整数型 | 否 | 200 ~ 599 | 当请求速率超过 rate + burst 后，将返回该状态码。默认为 503。 |

**表 2-3 API7 limit-req 插件参数列表**

通过控制面板创建路由后，为其绑定 limit-req 插件，假设设置 rate 为 1、burst 为 2，rejected_code 为 503，这表示表示每秒速率为1，当速率超过1但小于3时，请求将被延迟；当速率超过3时，请求将被拒绝，并返回 rejected_code，如 503。

2. **IP 黑白名单**

API 网关内置 IP 黑白名单插件，允许管理员通过控制面板进行设置。通过设置黑名单 IP 列表或者白名单 IP 列表，实现对路由、服务的资源访问控制。

3. **熔断**

当请求到达 API 网关后，会存在如下3种情况：

- 请求正常、响应正常；
- 请求正常、响应异常；
- 请求异常。

当海量请求到达时，若上游服务无法及时响应请求而处于阻塞状态时，将存在上游服务被打垮的情况。作为 API 网关，应当能及时发现并将异常问题处理，以避免更严重问题发生。 此时，API 网关服务降级将发挥作用。

## 2.5 日志审计

API7 内置了日志审计模块，通过集中采集信息系统中的系统安全事件、管理员操作记录、系统运行日志、系统运行状态等各类信息，经过规范化、过滤、归并等处理后，以统一格式的日志形式进行集中存储和管理，结合丰富的日志统计汇总及关联分析功能，实现对信息系统日志的全面审计。通过事后分析和报表系统，管理员可以方便高效地对信息系统进行有针对性的安全审计；当遇到特殊安全事件或配置故障，日志审计系统可以帮助管理员进行配置快速定位、回滚。只有具有权限的管理员可以进行操作回滚。

## 2.6 精细化路由

API 网关会将匹配到的请求，按照预设的权重与参数进行分流。

1. **按照权重分流**

管理员通过控制面板创建各个上游对象，在配置过程中，允许对每个上游服务实例设置权重值（weight），若该值为 0，表示不分配流量到该示例。此外，上游支持带权重的轮询调度（round robin）、一致性哈希（chash）、指数加权移动平均法(ewma) 等算法。

2. **按照参数分流**

API 网关支持根据请求的各个参数及其值来进行分流，例如：

| 参数名 | 参数值 |
| --- | --- |
| Request Header | { |

| | |
|---|---|
| | "vars": [["http_user_agent", "~*", "android"]], // 请求头 User-Agent 是否匹配 android<br>"uri": "/hello",<br>"upstream_id": "1"<br>} |
| Request Host | {<br>"hosts": ["www.my.com"], // host 是否为 "www.my.com"<br>"uri": "/hello",<br>"upstream_id": "1"<br>} |
| Request Path | {<br>"uri": "/hello", // path 是否为 "/hello"<br>"upstream_id": "1"<br>} |
| Request Query String | {<br>"vars": [["arg_theme", "==", "light"]], // query string 的 theme 参数是否为 light<br>"uri": "/hello",<br>"upstream_id": "1"<br>} |
| Request Cookie | {<br>"vars": [["cookie_token", "==", "1234"]], // cookie 的 token 字段是否为 “1234”<br>"uri": "/hello",<br>"upstream_id": "1"<br>} |

**表 2-4 API7 分流参数列表**

当请求与各参数匹配后，流量将被分配到对应上游服务。

## 2.7 监控告警

API 网关会记录每个请求的基本信息与状态，管理员借助控制面板的统计报表页，可以直观看到各个服务调用情况、状态码分布、成功数、失败数、95值、99值等信息，方便管理员了解系统健康程度。此外，数据平面会定时将流量处理情况进行上报，管理员可通过控制面板查看某时间段内的网关运行状态与其它指标，如错误率、请求数、状态码分布等。当管理员通过控制面板预设告警规则后，若网

关上报的流量情况与规则匹配后，将触发预设的策略，如发送站内信、邮件提醒、短信与 Webhook 通知等。

## 2.8 协议转换

API 网关对外统一暴露 RESTful API，管理员可在控制面板进行设定。这些 API 与企业内微服务/上游服务相对应，除代理常见的 HTTP 服务外，也支持对 Dubbo、gRPC、WebServices、MQTT 等协议的代理。

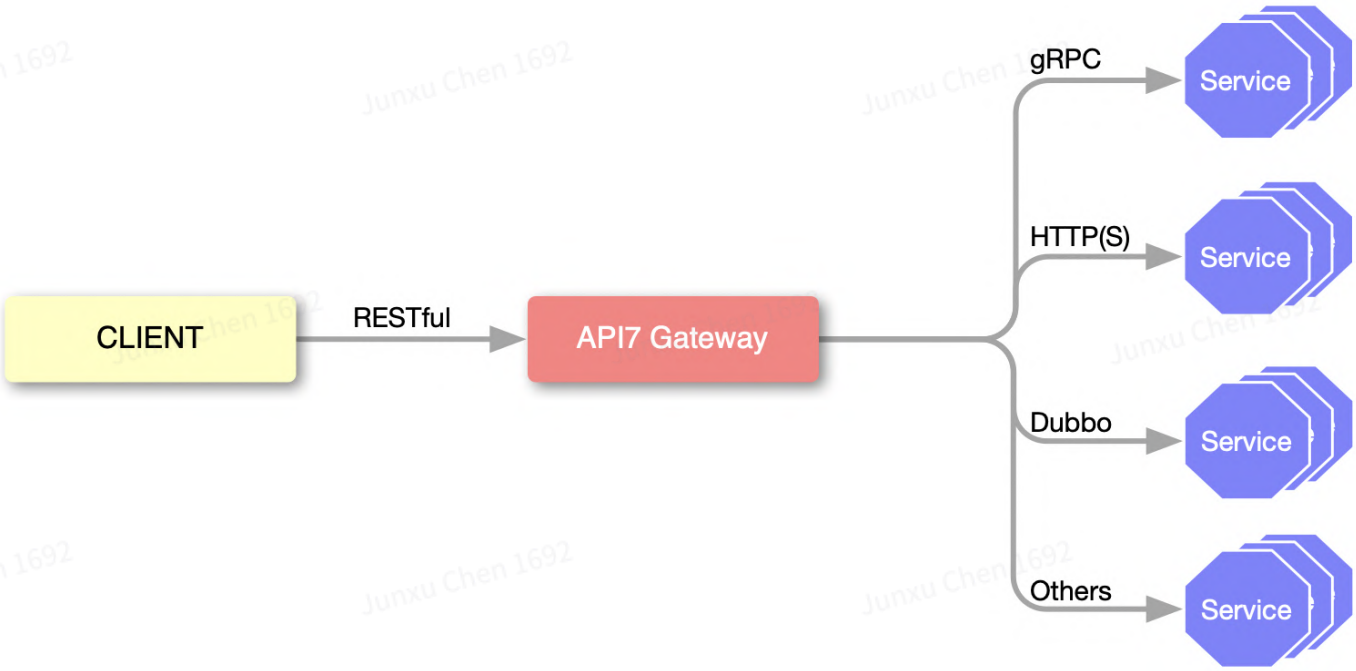


**图 2-2 协议转换**

控制面板允许管理员创建不同协议的上游服务，并支持创建路由对象与上游服务绑定，这些路由即调用方要使用的 API。在配置路由的过程中，管理员需设置该路由监听的 HTTP 方法（如 GET、POST、PATCH 等）、HTTP 主机名等参数，用以作为规则而匹配请求。

当路由配置完毕并发布后，请求经过 API 网关处理时，API 网关将根据各个路由规则匹配出对应的请求、构造不同协议的请求内容并转发至上游服务。

## 2.9 多租户和多工作分区

API7 内置了工作区模块，超级管理员需创建多个工作区，接着创建普通用户并分配不同的权限（在配置权限时，可绑定工作区与资源权限），这样结合用户系统与权限管理，可以实现不同用户在不同工作区时，对不同的资源有不同的权限，以实现对资源进行精细化权限管控。

## 2.10 性能

API7 从路由匹配、JSONSchema 校验、插件运行等各个环节，都采用了性能优秀的解决方案。

以路由匹配为例，API7 采用自研的 radixtree（由支流科技开源）算法路由，该算法在路由数量非常多时，效率并不会降低，因为其时间复杂度为 O(K)（K 为路由字符串长度，和路由数量无关）。

下图是在 10000 rps 的情况下，API7 和 Kong 企业版的延迟对比。

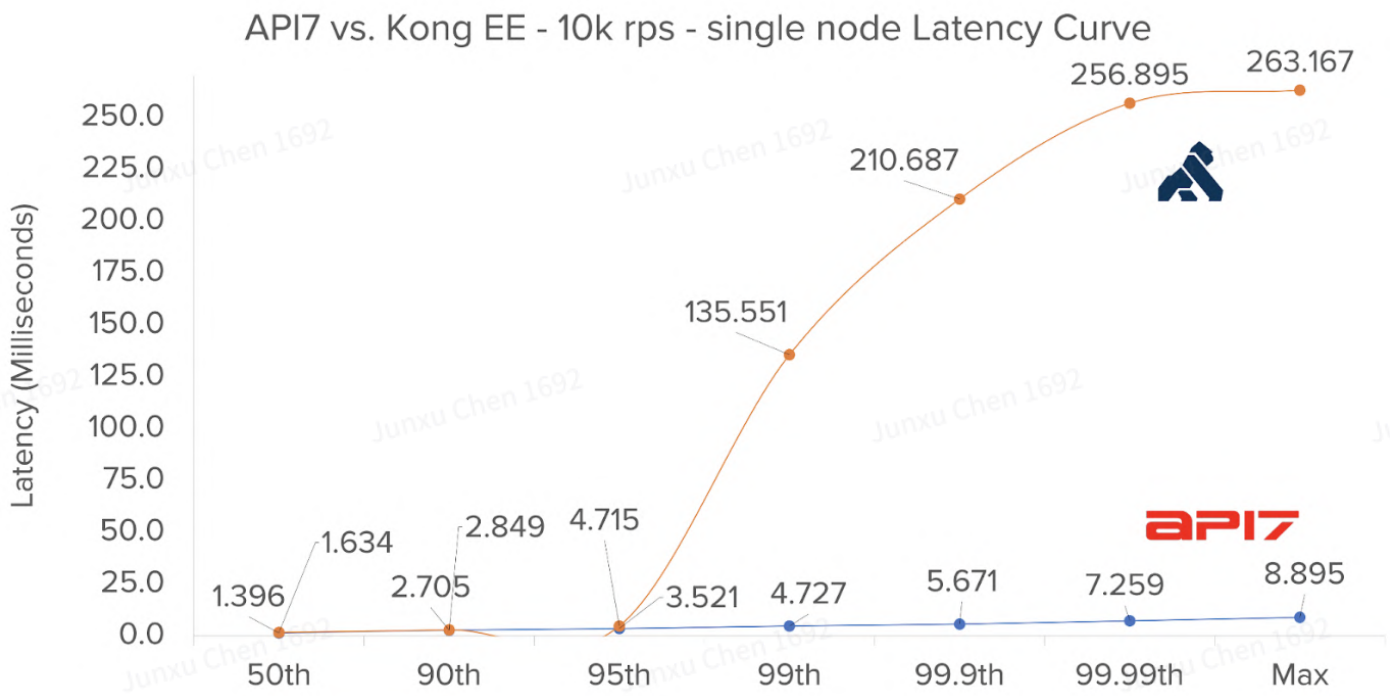


**图 2-3 在单节点和10krps 的情况下，API7 与 Kong 企业版性能对比**

从中可以看出，API7的延迟表现非常稳定，99.9% 的请求都在 6 毫秒内完成了处理；而 Kong 企业版的延迟是 API7 的几十倍。

下图是开启了 JWT 认证插件后，同样的 10000 rps 的延迟表现：

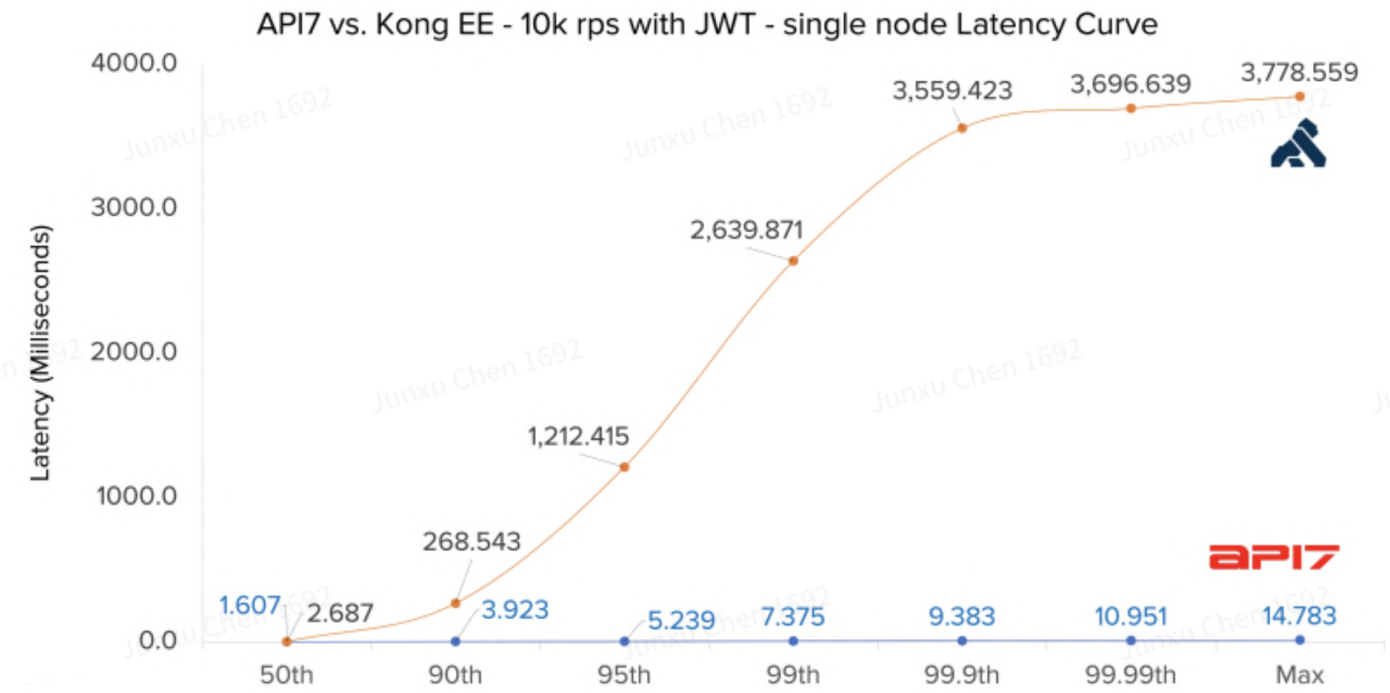


**图 2-4 在开启 JWT 插件和10krps 的情况下，API7 与 Kong 企业版性能对比**

可以看到，在开启 JWT 插件后 API7 的请求延迟表现依然稳定，而 Kong 企业版的延迟是 API7 的数百倍，差距非常明显。