Apache APISIX API Gateway Architecture

Apache APISIX API Gateway Architecture – Data Plane, Control Plane, and API Types

Apache APISIX is a **dynamic, real-time, high-performance API gateway** and **cloud-native** platform that provides rich traffic management features such as load balancing, dynamic upstream, authentication, observability, and more.

Apache APISIX is a **high-performance, cloud-native API gateway** built on a robust stack of technologies designed for **scalability, dynamic configuration, and extensibility**. Below is a detailed breakdown of its core components and architecture:

**1. Core Technologies Behind APISIX**

**a) Nginx (OpenResty)**

* **Foundation**: APISIX is built on top of **OpenResty**, which extends **Nginx** with Lua scripting.
  + **Why Nginx?**
    - Handles **high concurrency** with low resource usage (event-driven architecture).
    - Supports **L4/L7 traffic** (TCP/UDP/HTTP/HTTPS, gRPC, WebSockets, etc.).
  + **Why OpenResty?**
    - Adds **LuaJIT** for dynamic scripting inside Nginx.
    - Enables **plugin system** (APISIX plugins are mostly written in Lua).

**b) etcd (Configuration & Service Discovery)**

* **Role**: APISIX uses **etcd** (a distributed key-value store) as its **primary configuration backend**.
  + **Why etcd?**
    - Provides **real-time updates** (no reloads needed for config changes).
    - Supports **high availability** (distributed consensus via Raft).
    - Used for storing:
      * Routes
      * Upstreams (load balancing configs)
      * Plugin configurations
      * SSL certificates

**c) Lua & Plugin System**

* **Lua** is the primary scripting language for APISIX plugins.
  + **Why Lua?**
    - Lightweight, fast (JIT-compiled via LuaJIT).
    - Seamless integration with Nginx/OpenResty.
  + **Plugins** can modify requests/responses, add auth, logging, etc.

**d) Additional Dependencies**

* **Prometheus** (metrics collection).
* **Zipkin/SkyWalking** (distributed tracing).
* **Redis** (rate limiting, caching).
* **Kafka** (log streaming).

**2. APISIX Architecture (Deep Dive)**

**a) Data Plane vs. Control Plane**

| **Component** | **Role** | **Technology** |
| --- | --- | --- |
| **Data Plane** | Handles actual API traffic (L4/L7 proxy). | Nginx + Lua |
| **Control Plane** | Manages configs (routes, plugins, etc.). | etcd + Admin API |

**b) How APISIX Processes a Request**

1. **Request hits APISIX** → Matched against **routes** (stored in etcd).
2. **Plugins execute** (auth, rate limiting, transformations).
3. **Load balancing** → Forwarded to upstream (dynamic or static).
4. **Response returned** with optional logging/metrics.

**c) Dynamic Updates (No Reloads!)**

* Traditional Nginx requires reloads (nginx -s reload) for config changes.
* **APISIX avoids this** by:
  + Storing configs in **etcd** (real-time sync).
  + Using **Lua code** to apply changes on-the-fly.

**3. Why This Stack?**

**a) Performance**

* **Nginx** (C + LuaJIT) → Handles **millions of requests/sec** with low latency.
* **etcd** → Ensures configs are distributed and consistent.

**b) Extensibility**

* **Lua plugins** → Custom logic without recompiling.
* **Multi-language plugins** (Go, Java, Python via WASM).

1. Planes in APISIX

Data Plane (DP)

- Location in Diagram: Left side

- Purpose: Handles all actual API request and response processing.

- Responsibilities:

- Receives requests from clients/services

- Applies plugins (rate limiting, authentication, logging, etc.)

- Proxies and routes traffic to backend services

- Returns responses to clients

Control Plane (CP)

- Location in Diagram: Right side

- Purpose: Manages configuration, orchestration, and observability of the API gateway.

- Responsibilities:

- Stores and manages configuration (routes, plugins, credentials)

- Provides admin UI (dashboard/Manager API)

- Synchronizes configuration to Data Plane nodes

- Exports data to monitoring/observability tools

2. Data Plane Components

- Client/Service:

End-users or client applications making API requests.

- Apache APISIX Gateway:

The main API gateway engine (orange box), built on top of NGINX, handling incoming traffic, enforcing policies, and proxying requests.

- Plugin Layer:

- Rate Limit: Controls request rates.

- Auth: JWT, OAuth2, Key-auth, etc.

- Security: Access controls, IP restrictions.

- Logging: Captures API access logs.

- Custom Plugins: Extend APISIX functionality as needed.

- Public / Private / Partner:

Categories of APIs/services exposed via APISIX, each with different access and security policies.

3. Control Plane Components

- etcd Cluster:

Distributed, highly available storage for all APISIX configuration and metadata.

- Dashboard / Manager API:

Admin interfaces to configure and manage APISIX, which send updates to etcd.

- Observability & Monitoring Stack:

- Apache Skywalking, Prometheus, Grafana:

For tracing, monitoring, and visualizing logs/metrics collected from APISIX.

4. Flow Overview

API Request Flow:

1. Clients send API requests to APISIX.

2. APISIX processes the request via enabled plugins.

3. Requests are routed to the appropriate backend service (public, private, or partner APIs).

Configuration Flow:

1. Admins manage configs via Dashboard/Manager API.

2. Configurations are stored in etcd.

3. APISIX dynamically pulls configs from etcd—no restart needed.

Monitoring Flow:

1. APISIX exports monitoring data to Prometheus, Grafana, and Skywalking for real-time operational visibility.

5. Public, Private, and Partner APIs in APISIX

| API Type | Exposed To | Example Security | Example Use |

|-----------|-----------------------|--------------------------|------------------------------|

| Public | Anyone/external users | Basic Auth, Rate Limit | Open data, public search |

| Private | Internal teams/apps | mTLS, JWT, LDAP | Microservices, internal ops |

| Partner | Trusted third parties | API Key, OAuth, Quotas | B2B integration, supply chain|

How APISIX Handles These:

- Routing: Defines which APIs are public, private, or partner.

- Policy Enforcement: Applies appropriate plugins for auth, rate limiting, logging, etc., for each API category.

- Access Control: Ensures only authorized users or partners can access non-public APIs.

6. Summary Table

| Plane | Primary Role | Typical Components |

|---------------|----------------------------------|-------------------------------------|

| Data Plane | Handles API requests & policies | APISIX Gateway, Plugins |

| Control Plane | Config & monitoring management | etcd, Dashboard/API, Monitoring tools|

7. Summary Points

- Data Plane: Real API traffic processing and enforcement of policies/plugins.

- Control Plane: Configuration, orchestration, and monitoring of gateway nodes.

- etcd: Central configuration store.

- Observability: Real-time monitoring and visualization via Skywalking, Prometheus, and Grafana.

One-Line Summary

Apache APISIX is a next-gen cloud-native API gateway, built on NGINX and etcd, supporting dynamic routing, hot plugin loading, and deep integration with monitoring stacks.

(Use as documentation, speaker notes, or slide content as needed.)

**What is etcd?**

* **etcd** is a fast, distributed, highly-available key-value store.
* It’s open-source and widely used for storing configuration and metadata in cloud-native systems (like Kubernetes, APISIX).

**Why is etcd used in APISIX?**

**1. Central Configuration Store**

* All API gateway configurations—routes, upstreams, plugin settings, consumer info—are stored in etcd.
* This allows for **dynamic, real-time configuration** (APISIX doesn’t need to restart when configs change).

**Why is etcd Required by Both Data Plane (DP) and Control Plane (CP)?**

**Control Plane (CP):**

* The **CP** (e.g., Dashboard, Admin API) is responsible for writing or updating config in etcd.
* Whenever an admin adds, edits, or deletes a route or plugin, that information is written into etcd.

**Data Plane (DP):**

* The **DP** (the APISIX nodes handling live traffic) **reads config from etcd**.
* The DP watches etcd for any config changes and applies them instantly, without downtime.

**What is etcd?**

* **etcd** is a fast, distributed, highly-available key-value store.
* It’s open-source and widely used for storing configuration and metadata in cloud-native systems (like Kubernetes, APISIX).

**Why is etcd used in APISIX?**

**1. Central Configuration Store**

* All API gateway configurations—routes, upstreams, plugin settings, consumer info—are stored in etcd.
* This allows for **dynamic, real-time configuration** (APISIX doesn’t need to restart when configs change).

**Why is etcd Required by Both Data Plane (DP) and Control Plane (CP)?**

**Control Plane (CP):**

* The **CP** (e.g., Dashboard, Admin API) is responsible for writing or updating config in etcd.
* Whenever an admin adds, edits, or deletes a route or plugin, that information is written into etcd.

**Data Plane (DP):**

* The **DP** (the APISIX nodes handling live traffic) **reads config from etcd**.
* The DP watches etcd for any config changes and applies them instantly, without downtime.