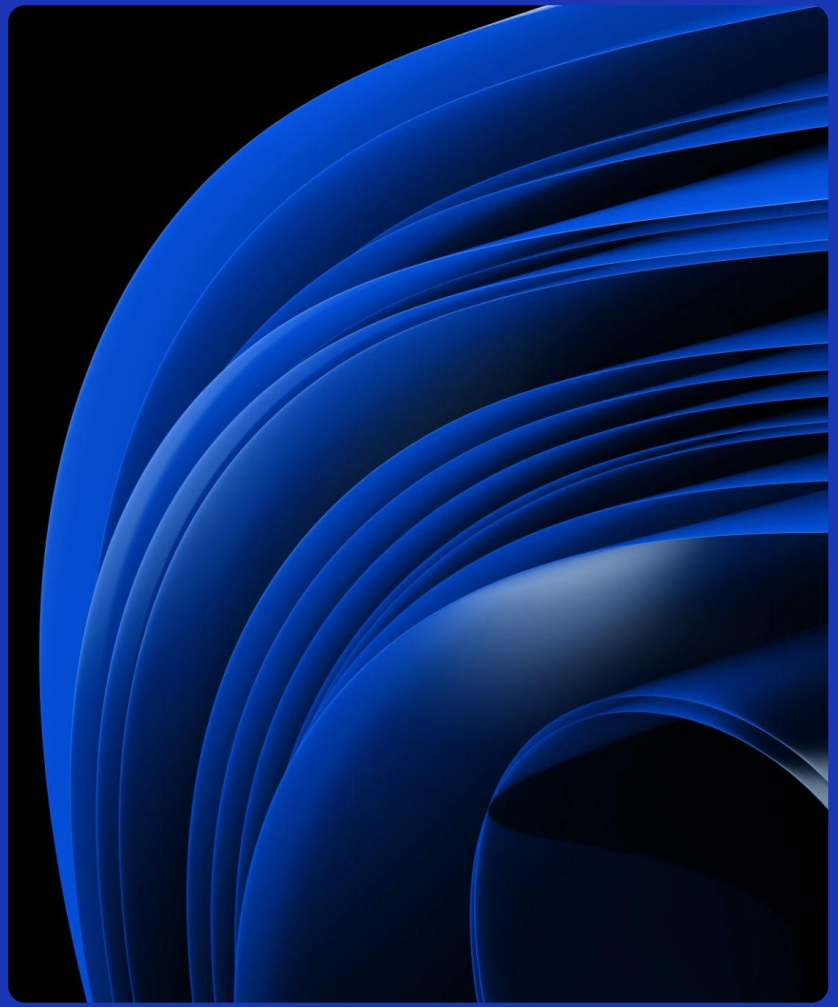
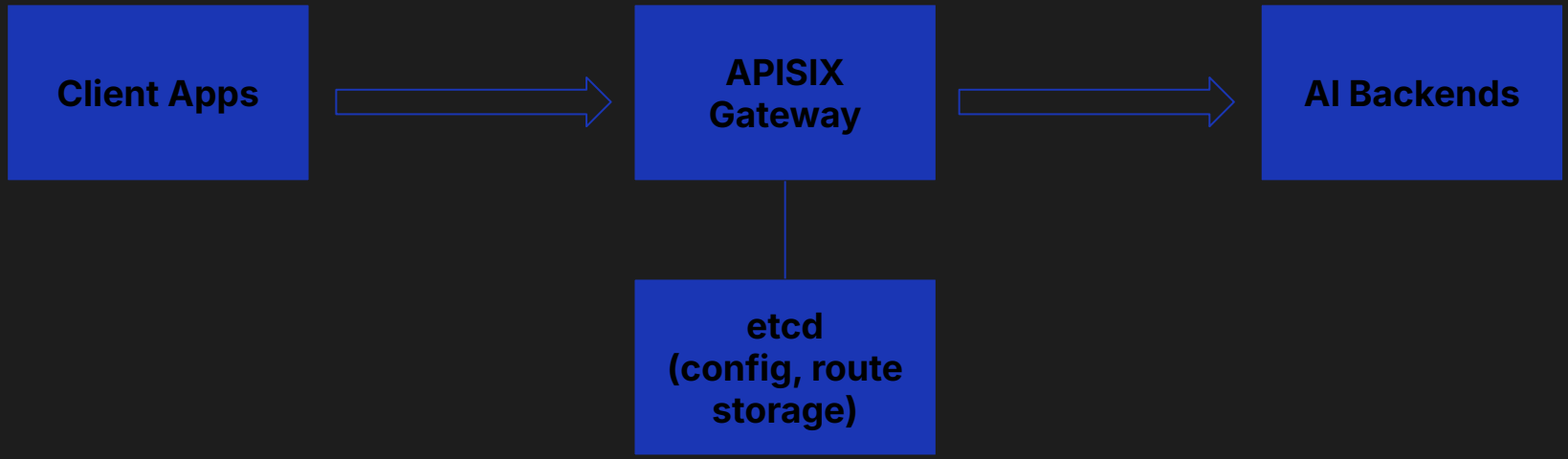
APACHE APISIX

AI Gateway

Derrick M.
Jovytta E. Y.





Architecture Overview

docker-compose.yml

docker-compose.yml

```yaml
services:
  apisix:
    image: apache/apisix:latest
    ports:
      - "9080:9080"  # HTTP port
      - "9180:9180"  # Admin API port
    environment:
      - ETCD_HOST=etcd  # "etcd" is the service name
      - ETCD_PORT=2379
    network_mode: host
    depends_on:
      - etcd

  etcd:
    image: bitnami/etcd:latest
    ports:
      - "2379:2379"
    environment:
    - ALLOW_NONE_AUTHENTICATION=yes
    network_mode: host
```

- Images used: etcd:latest & apisix:latest
- **apisix**:
  - Port **9080** for handling HTTP requests
  - Port **9180** for Admin API for configuration commands (setting up routes, plugins, etc)
  - Uses **etcd** to store the configurations, hence the "**environment**" settings
  - **network_mode**: host, exposes the container ports to the host's local ports
- **etcd**:
  - **ALLOW_NONE_AUTHENTICATION=yes**, allows configurations without the need of authentication

Plugin: ai-proxy-multi

Connects local route to AI backend services.

You can specify which model from your backend service that you want.

You can specify the weight of each endpoint to decide how much of the traffic they get.

```
"ai-proxy-multi": {
        "instances":[
          { "name": "llama1",
            "provider": "openai-compatible",
            "weight": 3,
            "auth": {
              "header": {"api-key": "blablabal"}
            },
            "options": {
              "model": "nvidia/Llama-3_3-Nemotron-Super-49B-v1"
            },
            "override": {
              "endpoint":
"http://172.18.212.157:31236/v1/chat/completions"
            }
          },
          { "name": "llama2",
            "weight": 2,
            "provider": "openai-compatible",
            "auth": {
              "header": {"api-key": "blablabla"}
            },
            "options": {
              "model": "LLAMA 3.3 70B"
            },
            "override": {
              "endpoint":
"http://172.18.246.168:31199/v1/chat/completions"
            }
          }
        ]
      }
```

Plugin: ai-rate-limiting

```
"ai-rate-limiting": {
  "instances":[
    {
      "name": "llama1",
      "limit": 100,
      "time_window": 60
    },
    {
      "name": "llama2",
      "limit": 200,
      "time_window": 90
    }
  ],
  "rejected_code": 429,
  "rejected_msg": "Exceeded Rate Limit :(",
  "limit_strategy": "prompt_tokens"
}
```

- If we were using **ai-proxy**, or wish to implement uniform rate limit to all models, the use of **instances** is not necessary, **limit** and **time_window** will suffice.
- **instances** allow custom limiting specific to each models included in **ai-proxy-multi**
- **limit** is calculated in tokens, and **time_window** in seconds.
- **limit_strategy**: accepted values are “prompt_tokens”, “completion_tokens” and “total_tokens”
- **rejected_code** and **rejected_msg** are NOT necessary, but provide extra context

In case of rate-limit-exceeded, returns **all upstream servers tried** when used with **ai-proxy-multi**. (error code caught by handler, not printed to stdout)
Only returns exact **code** and **msg** when paired with a**i-proxy**

Plugin: ai-prompt-decorator

```
"ai-prompt-decorator":{
  "prepend": [
    {
      "role": "system",
      "content": "You are a helpful
      and knowledgeable assistant
      that DOES NOT blurt out
      random things."
    }
  ],
  "append": [
    {
      "role": "system",
      "content": "Keep it concise"
    }
  ]
},
```

Modifies the message:
"messages": [{ "role": "user", "content": "What is TLS Handshake?" }]
into:

```
"messages": [
   {
     "role": "system",
     "content": "You are a helpful and knowledgeable
assistant that DOES NOT blurt out random things."
   },
   { "role": "user", "content": "What is TLS Handshake?"
},
   {
     "role": "system",
     "content": "Keep it concise."
   }
 ]
```

! Uses up tokens faster when used with **ai-rate-limiting**

Plugin: ai-prompt-guard

Safeguards AI endpoints by inspecting and validating incoming prompts.

match_all_roles: true → checks prompts from all roles. Default is false → only checks user prompts.

deny_patterns: array of unallowed strings

allow_patterns: when provided, prompt must match at least on pattern to be considered valid.

```
"ai-prompt-guard": {
    "match_all_roles": true,
    "deny_patterns": ["badword"],
    "allow_patterns": ["goodword"]
 }
```

Thank you