Lesson 7 -> API gateways

What is an API Gateway?

An **API Gateway** is a **central entry point for all client-facing API requests**.

Clients never call backend services directly.
They call the API Gateway, and the gateway decides **what to do with the request**.

At a high level:

> An API Gateway is a **specialized reverse proxy designed for APIs**.

What problems does an API Gateway solve?

As systems grow:

- Many microservices
- Many teams
- Many versions of APIs
- Many clients (web, mobile, partners)

Without a gateway:

- Clients must know every service
- Security logic is duplicated
- Versioning becomes chaos

The API Gateway centralizes all of this.

Core Responsibilities of an API Gateway

1) Request Routing

Routes requests to the correct backend service.

Example:

- /api/v1/users → User Service
- /api/v1/orders → Order Service
- /api/v1/recommend → ML Service

Clients see **one endpoint**, not many services.

---

2) Authentication & Authorization

API Gateway usually handles:

- JWT validation
- OAuth tokens
- API keys

Backend services can **trust** the gateway and remain simpler.

---

3) Rate Limiting & Throttling

Very important for ML systems.

Example:

- Free users → 10 requests/sec
- Premium users → 100 requests/sec
- ML inference → hard limits

Prevents abuse and cost explosions.

---

4) Request/Response Transformation

Gateway can:

- Modify headers
- Transform payloads
- Aggregate responses

Example:
Client makes one call → gateway calls 3 services → combines response.

---

5) API Versioning

Handles:

- /v1/

- /v2/
- backward compatibility

Clients don’t break when services evolve.

---

6) Observability & Control

Because all traffic passes through the gateway:

- Central logging
- Metrics
- Tracing
- Error handling

This is critical for debugging production ML issues.

How is an API Gateway related to a Reverse Proxy?

This is the key insight:

> **Every API Gateway is a reverse proxy, but not every reverse proxy is an API Gateway.**

Reverse Proxy (General-Purpose)

Primary focus:

- Routing
- Load balancing
- SSL termination
- Caching

Examples:

- Nginx
- HAProxy
- Envoy

Used for:

- Web traffic

- Static content
- Generic backend routing

API Gateway (API-Specific)

Built on top of reverse proxy concepts, but adds:

- Authentication
- Authorization
- Rate limiting
- API versioning
- Request aggregation

Examples:

- Kong
- AWS API Gateway
- Apigee
- Azure API Management

Used for:

- Client-facing APIs
- External consumers
- ML inference APIs

Side-by-Side Mental Model

Reverse Proxy:
“I forward traffic efficiently”

API Gateway:
“I manage APIs safely and intelligently”

Where They Sit in Architecture

Typical modern architecture:

Client
→ API Gateway

→ Reverse Proxy / Service Mesh
→ Microservices / ML Services

Sometimes:

- API Gateway itself uses a reverse proxy internally

API Gateway in ML Systems (Very Important)

ML inference is:

- Expensive
- Latency-sensitive
- Abuse-prone

API Gateway protects ML by:

- Limiting request rates
- Authenticating clients
- Routing traffic to correct model versions
- Enabling A/B testing and canary releases

Example:

- 90% traffic → old model
- 10% traffic → new model

All controlled at gateway level.

Do You Always Need an API Gateway?

No.

You need an API Gateway if:

- You have multiple services
- You have multiple clients
- You expose public APIs
- You serve ML models externally

You may skip it if:

- Small internal system

- Single service
- No external consumers

Downsides of API Gateways (Important)

1) Additional latency

One more hop in request path.

2) Configuration complexity

Misconfigurations can break many services at once.

3) Potential single point of failure

Must be deployed redundantly (just like reverse proxy).

Interview-Ready Explanation

An API Gateway is a specialized reverse proxy that acts as a single entry point for client-facing APIs, handling routing, authentication, rate limiting, and versioning, while backend services focus only on business logic.