Learning Consolidation

Create a Single-entry Point to Route the Request Coming for Different Microservices Using Spring Cloud

Learning Objectives

- Explore the Microservices Design Pattern
- Define the API Gateway Pattern

Slide Note

Microservices Design Patterns

- Microservices design patterns are software design patterns that generate reusable autonomous services.
- The goal for developers using microservices is to accelerate application releases.
- By using microservices, developers can deploy each individual microservice independently, if desired.
- The design pattern helps developers with certain principles when developing individual microservices.

Slide Note

• It might have other responsibilities such as authentication, monitoring, load balancing, caching, request shaping and management, and static response handling.

• An API Gateway is an API management tool that sits between a client and a collection of backend services.

• The API Gateway encapsulates the internal system architecture and provides an API that is tailored to each client.

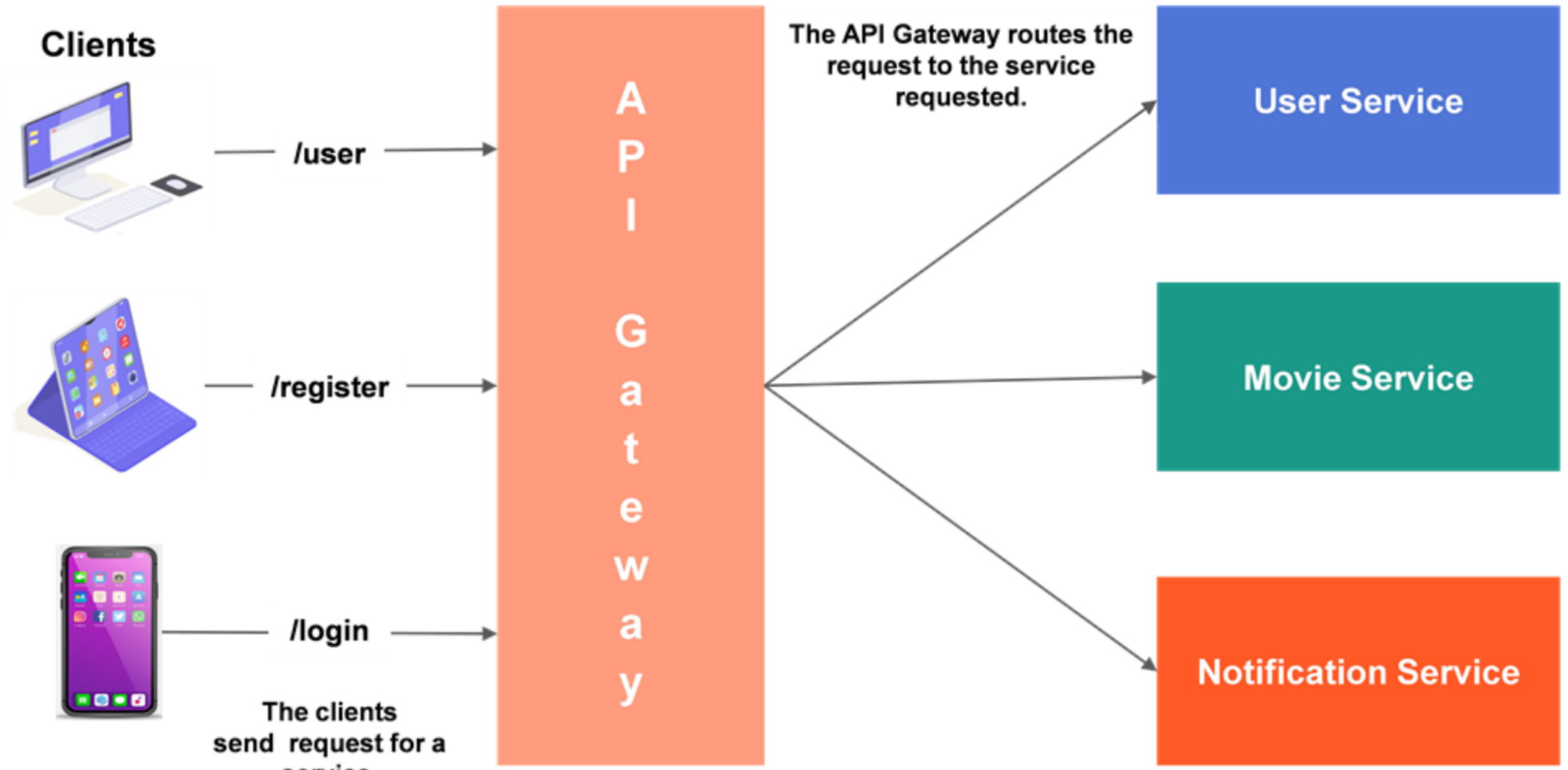
• It's common for API gateways to handle common tasks that are used across a system of API services, such as user authentication, rate limiting, and statistics.

What Is an API Gateway?

- An API Gateway is a server that is the single-entry point into the system.
- It is a tool that sits between a client and a collection of backend services.
- An API gateway acts as a reverse proxy to:
  - Accept all application programming interface (API) calls
  - Aggregate the various services required to fulfill them
  - Return the appropriate result back to the client
- Most enterprise APIs are deployed via API Gateways.

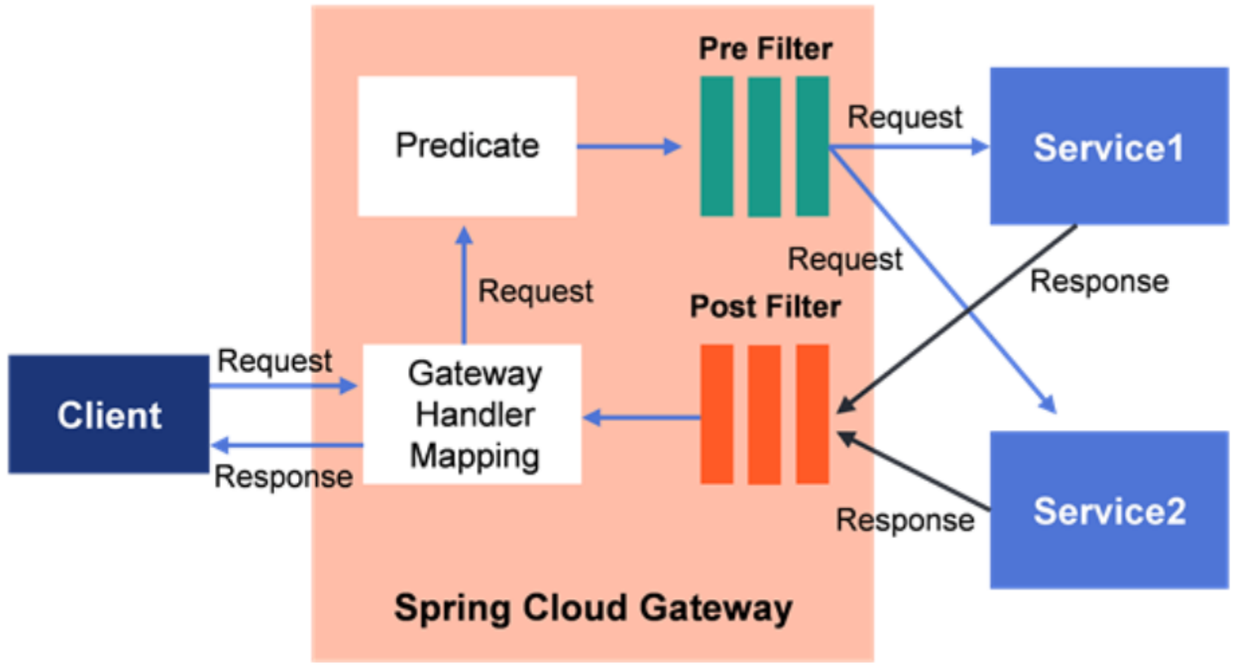
Slide Note

How Does an API Gateway Work?



Slide Note

Spring Cloud API Gateway Architecture



Spring Cloud API Gateway:

- Is built on top of the Spring ecosystem.
- Aims to provide a simple, yet effective way to route to the APIs.
- Consists of the following:
  - Route
  - Predicate
  - Filter

Slide Note

Spring Cloud API Gateway Architecture

- **Route**: The route is the basic building block of the gateway. It is defined by an ID, a destination URI, a collection of predicates, and a collection of filters. A route is matched if the aggregate predicate is true.
  - **Predicate**: This is a Java 8 Function Predicate. It is an object in Spring Cloud gateway that tests whether the given request fulfills a given condition. For each route, you can define one or more predicates that, if satisfied, will accept requests for the configured backend.
  - **Filter**: These are instances of Spring Framework Gateway Filter that have been constructed with a specific factory. Here, you can modify requests and responses before or after sending the downstream request.

Slide Note

Step 2 – Configure the Routes

- Create a Java class as a configuration file for configuring the routes to the APIs in the application.
- Build the routes using the classes below:
  - `RouteLocator` – To obtain route information:
    - `path` – the rest endpoint patterns
    - `uri` – the uri at which the service is currently running
  - `RouteLocatorBuilder` – Used to create routes
- Here, `UserAuthenticationService` runs on port `8086`.
- And `UserMovieService` runs on port `8081`.

```
@Bean
public RouteLocator myRoutes(RouteLocatorBuilder builder) {
    return builder.routes()
            .route(p -> p
                    .path( ...patterns: "/api/v1/**")
                    .uri("http://localhost:8086/"))

            .route(p->p
                    .path( ...patterns: "/api/v2/**")
                    .uri("http://localhost:8081/"))
            .build();
}
```

Slide Note

1. REST call is made to the API Gateway.

2. The API Gateway routes the request to the appropriate service.

3. The call is routed to the UserMovieService to register the new user.

4. Once the user is registered, he/she will be directed to login, thus we need to save the user in the UserAuthenticationService.

5. Then the user logs into the system and a JSON web token is generated.

6. Using the JWT the user can perform all the CRUD operations.

7. Finally, the entire application is dockerized.

How Does This Application Work?