

API-GETEWAY

In a microservices application, there are many small services running separately each handling a specific task like user login, payments, or showing product details. Now, imagine a user wants to use your application. Instead of the user directly calling each microservice, we give them **one single entry point**. That entry point is called the **API Gateway**.

Netflix Zuul is a tool used to create this API Gateway in a Spring Boot project. It acts like a **smart gate** or **traffic controller** that receives all user requests and then sends them to the correct microservice behind the scenes. It can also add security, log requests, and handle errors.

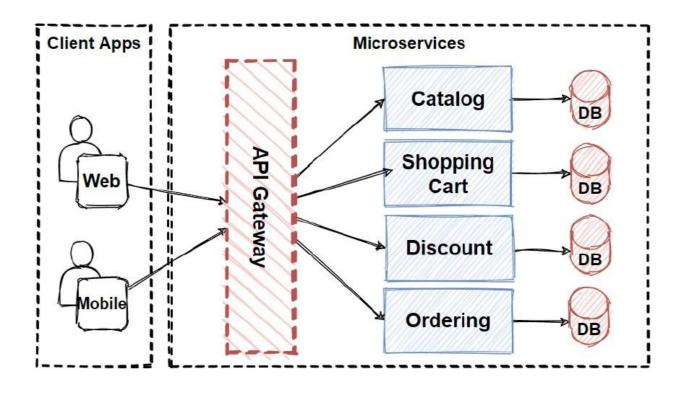
Using Netflix Zuul, we can make our system simpler, more secure, and easier to manage.

Pointwise Explanation (Very Simple)

- 1. **Microservices** = Small parts of a big application, each doing one job (like login, payment, order, etc.).
- 2. **Problem** = User has to call each microservice directly messy and hard to manage.
- 3. **Solution** = Use an **API Gateway** one place where all user requests come in.
- 4. **Netflix Zuul** = A tool that helps create the API Gateway in a Spring Boot project.
- 5. What Zuul Does:
 - o Receives requests from users
 - Sends them to the correct microservice
 - o Returns the response to the user
 - o Can also handle logging, authentication, and errors

6. Why Use Zuul:

- Easier to manage microservices
- o More secure (you can add security rules in one place)
- Better control over traffic
- 7. **Spring Boot** + **Zuul** = Simple way to build and run an API Gateway quickly.

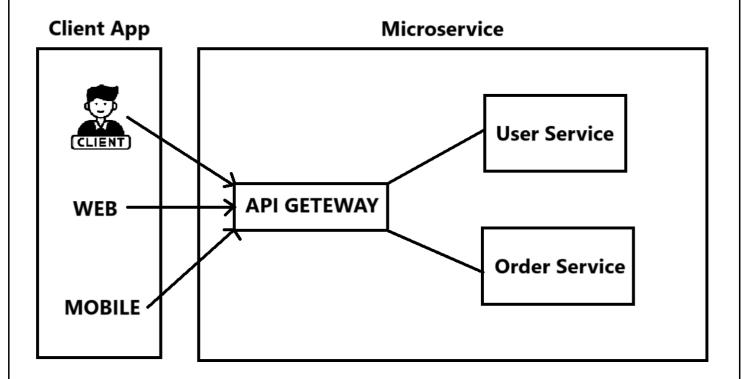


Netflix Zuul API Gateway Example with Spring Boot

We have two microservices,

- 1. User-Service
- 2. Order-Service,

integrated with an API-Gateway



1. User Service:

Project Structure

- - - →

 ⊕ com.app.userservice
 - > UserServiceApplication.java
 - v # com.app.userservice.controller
 - > <a> UserController.java
 - ∨ # src/main/resources

 - src/test/java
 - > March JRE System Library [JavaSE-11]
 - > Maven Dependencies
 - # target/generated-sources/annotations
 - # target/generated-test-sources/test-annotations
 - > 🕭 src
 - > 🗁 target

 - mvnw
 - mvnw.cmd
 - pom.xml

Class: UserServiceApplication.Java

```
package com.app.userservice;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class UserServiceApplication {

public static void main(String[] args) {
    SpringApplication.run(UserServiceApplication.class, args);
}

}
```

Class: UserController

```
1 package com.app.userservice.controller;
 3 import org.springframework.web.bind.annotation.GetMapping;
10
11 @RestController
12 @RequestMapping("/users")
13 public class UserController {
14
       @GetMapping("/{id}")
15⊜
16
       public Map<String, Object> getUser(@PathVariable String id) {
           Map<String, Object> user = new HashMap<>();
17
           user.put("id", id);
18
           user.put("name", "User " + id);
19
           user.put("email", "user" + id + "@example.com");
20
           user.put("service", "user-service");
21
           return user;
22
23
       }
24
25⊝
       @GetMapping
       public Map<String, Object> getAllUsers() {
26
27
           Map<String, Object> response = new HashMap<>();
           response.put("message", "All users from user-service");
28
           response.put("service", "user-service");
29
30
           response.put("port", "8081");
           return response;
31
32
       }
33 }
```

YAML: application.yml

Pom.xml

```
http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)
1 ⟨?xml version="1.0" encoding="UTF-8"?>
  2⊖ cproject xmlns="http://maven.apache.org/POM/4.0.0"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  4
  5
             http://maven.apache.org/xsd/maven-4.0.0.xsd">
  6
        <modelVersion>4.0.0</modelVersion>
  7
  8
        <groupId>com.example
  9
        <artifactId>user-service</artifactId>
 10
        <version>1.0-SNAPSHOT</version>
 11
        <packaging>jar</packaging>
 12
 13Θ
        <parent>
 14
            <groupId>org.springframework.boot
 15
            <artifactId>spring-boot-starter-parent</artifactId>
 16
            <version>2.3.12.RELEASE
 17
            <relativePath/>
 18
        </parent>
 19
 20⊝
        properties>
 21
            <java.version>11</java.version>
 22
        </properties>
 23
 24⊝
        <dependencies>
 25⊜
            <dependency>
 26
                <groupId>org.springframework.boot
 27
                <artifactId>spring-boot-starter-web</artifactId>
            </dependency>
 28
 29
        </dependencies>
 30
        <build>
 31⊖
 32⊝
            <plugins>
 33⊝
                <plugin>
 34
                    <groupId>org.springframework.boot
 35
                    <artifactId>spring-boot-maven-plugin</artifactId>
 36
                </plugin>
 37
            </plugins>
 38
        </build>
 39 k/project>
```

2. Order Service

Project Structure

```
∨ 🕭 src/main/java

→ 

⊕ com.app.orderservice

        > 🛮 OrderServiceApplication.java

→ 

⊕ com.app.orderservice.controller

        > 🗓 OrderController.java

√ 

// src/main/resources

       application.yml
    src/test/java
  ⇒ Mark JRE System Library [JavaSE-11]
  Maven Dependencies
    # target/generated-sources/annotations
    # target/generated-test-sources/test-annotations
  > 🗁 src
  > 🗁 target

    HELP.md

     mvnw
    mvnw.cmd
    pom.xml
```

Class: OrderServiceApplication.java

```
PorderServiceApplication.java x

1 package com.app.orderservice;
2
3*import org.springframework.boot.SpringApplication;
5
6 @SpringBootApplication
7 public class OrderServiceApplication {
8
9*    public static void main(String[] args) {
        SpringApplication.run(OrderServiceApplication.class, args);
11    }
12
13 }
```

Class: OrderController.java

```
☑ OrderController.java ×
 1 package com.app.orderservice.controller;
 3⊕import org.springframework.web.bind.annotation.GetMapping;
10
11 @RestController
12 @RequestMapping("/orders")
13 public class OrderController {
14
15⊜
       @GetMapping("/{id}")
       public Map<String, Object> getOrder(@PathVariable String id) {
16
17
            Map<String, Object> order = new HashMap<>();
            order.put("id", id);
18
            order.put("product", "Product " + id);
19
            order.put("amount", 100.00);
20
            order.put("service", "order-service");
21
            return order;
22
23
       }
24
25⊜
       @GetMapping
26
       public Map<String, Object> getAllOrders() {
27
            Map<String, Object> response = new HashMap<>();
            response.put("message", "All orders from order-service");
28
            response.put("service", "order-service");
29
            response.put("port", "8082");
30
31
            return response;
32
       }
33 }
```

YAML: application.yml

```
application.yml ×
  1 server:
  2
     port: 8082
  3
 4⊖spring:
  5⊜
     application:
        name: order-service
  6
  7
  8 logging:
     level:
  9⊜
        com.example.orderservice: DEBUG
10
```

Pom.xml

```
order-service/pom.xml ×
    http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)
% 1 <?xml version="1.0" encoding="UTF-8"?>
  2⊖cproject xmlns="http://maven.apache.org/POM/4.0.0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  4
  5
            http://maven.apache.org/xsd/maven-4.0.0.xsd">
  6
        <modelVersion>4.0.0</modelVersion>
  7
  8
        <groupId>com.example
  9
        <artifactId>order-service</artifactId>
        <version>1.0-SNAPSHOT
 10
 11
        <packaging>jar</packaging>
 12
 13⊖
        <parent>
            <groupId>org.springframework.boot</groupId>
 14
 15
            <artifactId>spring-boot-starter-parent</artifactId>
 16
            <version>2.3.12.RELEASE
            <relativePath/>
 17
        </parent>
 18
 19
 20⊖
        properties>
 21
            <java.version>11</java.version>
 22
        </properties>
 23
 249
        <dependencies>
 25⊖
           <dependency>
 26
                <groupId>org.springframework.boot
 27
                <artifactId>spring-boot-starter-web</artifactId>
            </dependency>
 28
        </dependencies>
 29
 30
 31⊖
        <build>
           <plugins>
 32⊖
 33⊖
                   <groupId>org.springframework.boot
 34
 35
                    <artifactId>spring-boot-maven-plugin</artifactId>
 36
                </plugin>
 37
            </plugins>
        </build>
 38
 39
```

After developing two services finally we have to develop API-GETEWAY, Lets do it...

API-GETEWAY

```
v 🕭 src/main/java

    # com.app.geteway

       ApiGatewayApplication.java
  ∨ # src/main/resources
      # src/test/java
  ⇒ Mark System Library [JavaSE-11]
  > Maven Dependencies
    ## target/generated-sources/annotations
    # target/generated-test-sources/test-annotations
  > 🔝 src
  > 🗁 target

    HELP.md

    mvnw
    mvnw.cmd
    M pom.xml
```

Class: ApiGatewayApplication.java

```
PapiGatewayApplicationjava x

1 package com.app.geteway;

2
3*import org.springframework.boot.SpringApplication;
6
7 @SpringBootApplication
8 @EnableZuulProxy
9 public class ApiGatewayApplication {
10
11 public static void main(String[] args) {
    SpringApplication.run(ApiGatewayApplication.class, args);
    }
14
15 }
```

YAML: application.yml

```
application.yml ×
 1⊖ server:
    port: 8080
 3
 4⊕ spring:
 5⊖ application:
 6 name: api-gateway
 7
 8 # Zuul Configuration
 9⊖ zuul:
 10⊖ routes:
      user-service:
 11⊕
      path: /users/**
 12
        url: http://localhost:8081
 13
 14
        strip-prefix: false
 15
 16⊖ order-service:
      path: /orders/**
 17
         url: http://localhost:8082
 18
 19
         strip-prefix: false
 20
 21
    # Global settings
 22⊖ host:
    connect-timeout-millis: 20000
 23
      socket-timeout-millis: 20000
 24
 25
 26 # Remove sensitive headers
 27 sensitive-headers: Cookie, Set-Cookie, Authorization
 28
29@logging:
 30⊖ level:
31 com.netflix.zuul: DEBUG
332
       org.springframework.cloud.netflix.zuul: DEBUG
```

Pom.xml

```
    □ api-gateway/pom.xml ×

    http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)

  1 <?xml version="1.0" encoding="UTF-8"?>

  20 cproject xmlns="http://maven.apache.org/POM/4.0.0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
 4
  5
            http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
  6
 7
 8
        <groupId>com.example
 9
        <artifactId>api-gateway</artifactId>
        <version>1.0-SNAPSHOT</version>
 10
 11
        <packaging>jar</packaging>
 12
 13⊜
        <parent>
 14
            <groupId>org.springframework.boot
 15
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>2.3.12.RELEASE
 16
            <relativePath/>
 17
 18
        </parent>
 19
 20⊝
        properties>
            <java.version>11</java.version>
 21
            <spring-cloud.version>Hoxton.SR12
 22
 23
        </properties>
 2/1
       <dependencies>
25⊜
269
           <dependency>
27
               <groupId>org.springframework.boot
               <artifactId>spring-boot-starter-web</artifactId>
28
29
           </dependency>
30
31⊖
           <dependency>
32
               <groupId>org.springframework.cloud
33
               <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
           </dependency>
34
35
369
           <dependency>
37
               <groupId>org.springframework.cloud
38
               <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
39
           </dependency>
       </dependencies>
40
41
429
       <dependencyManagement>
43⊜
           <dependencies>
44⊜
               <dependency>
                   <groupId>org.springframework.cloud
45
                   <artifactId>spring-cloud-dependencies</artifactId>
46
47
                   <version>${spring-cloud.version}
48
                   <type>pom</type>
49
                   <scope>import</scope>
50
               </dependency>
           </dependencies>
51
       </dependencyManagement>
52
```

```
549
      <build>
          <plugins>
55⊖
              <plugin>
56⊖
                   <groupId>org.springframework.boot
57
                   <artifactId>spring-boot-maven-plugin</artifactId>
58
59
              </plugin>
          </plugins>
60
       </build>
61
62 </project>
```

How to Run

Start the services in order

Terminal 1 - Start User Service

cd user-service mvn spring-boot:run

Terminal 2 - Start Order Service

cd order-service mvn spring-boot:run

Terminal 3 - Start API Gateway

cd api-gateway mvn spring-boot:run

Test the API Gateway

Access User Service through Gateway

http://localhost:8080/users http://localhost:8080/users/123

Access Order Service through Gateway

http://localhost:8080/orders http://localhost:8080/orders/456

How API Gateway Works

- 1. **Single Entry Point**: All client requests go through the API Gateway (port 8080)
- 2. **Request Routing**: Zuul routes requests based on path patterns:
 - /users/** → User Service (port 8081)
 - $/orders/** \rightarrow Order Service (port 8082)$

Key Zuul Concepts

- @EnableZuulProxy: Enables Zuul proxy functionality
- **Routes**: Define how requests are mapped to backend services
- Filters: Pre, Route, Post, and Error filters for request processing
- Load Balancing: Built-in load balancing with Ribbon
- Circuit Breaker: Integration with Hystrix for fault tolerance

Than	ık you
------	--------