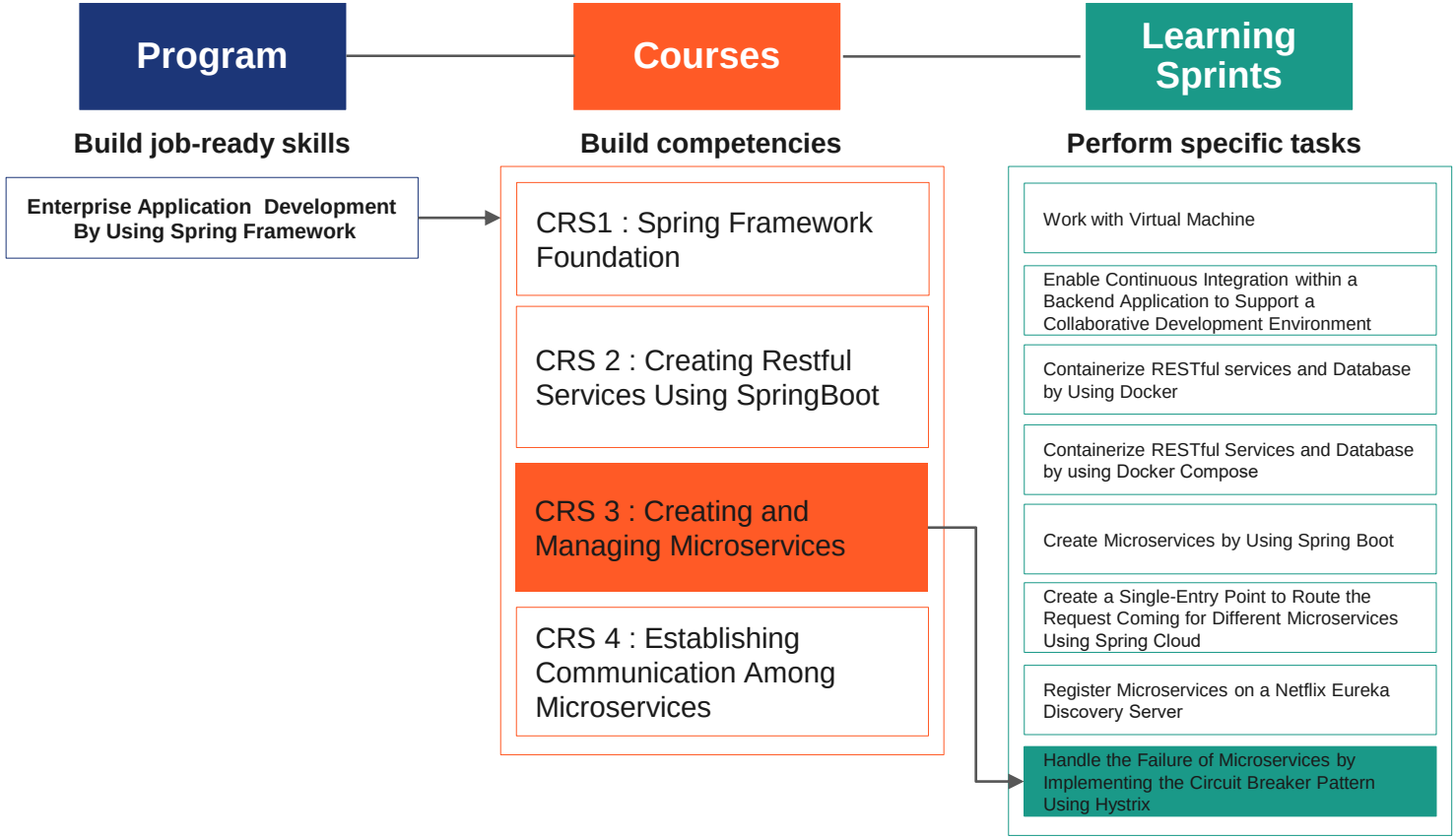
Backend Program: Course 3: Structure



Think and Tell

If a service is under maintenance and all instances of the service are not running. A request is sent for that service by a client.

- What do you think the API Gateway will do?
- Will it keep searching for an instance of the service in the Eureka registry?
- Will the API Gateway get a meaningful message stating the service is under maintenance?
- Will the API Gateway be able to convey this back to the client?

Handle the Failure of Microservices by Implementing the Circuit Breaker Pattern Using Hystrix

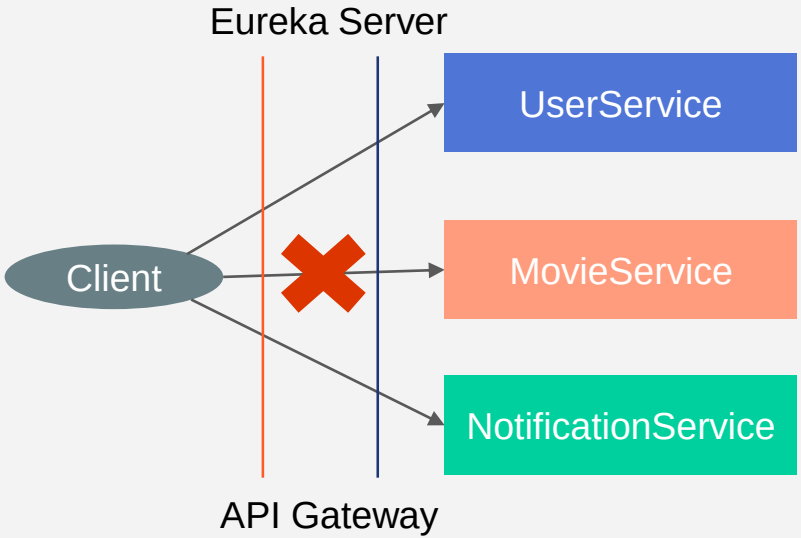
Learning Objectives

- Define the microservices circuit breaker design pattern
- Implement Hystrix for circuit breaking

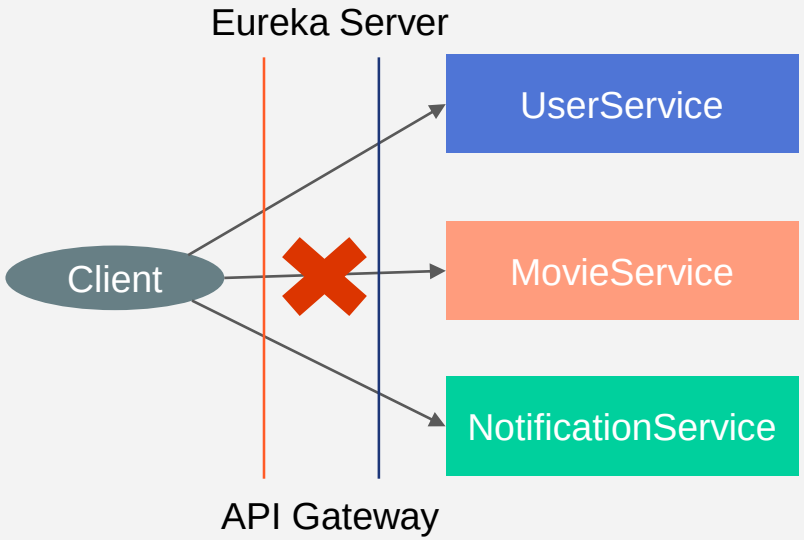
Microservices Design Patterns - Circuit Breaker Design Pattern

Microservices Failure

- The microservice architecture contains many small microservices.
- The instances of microservices may go up and down frequently. The chances of its failure are also most likely.
- When a client invokes a service, there are chances that the service might become unavailable.
- The failure of one service can potentially cascade to other services throughout the application.



Microservices Failure

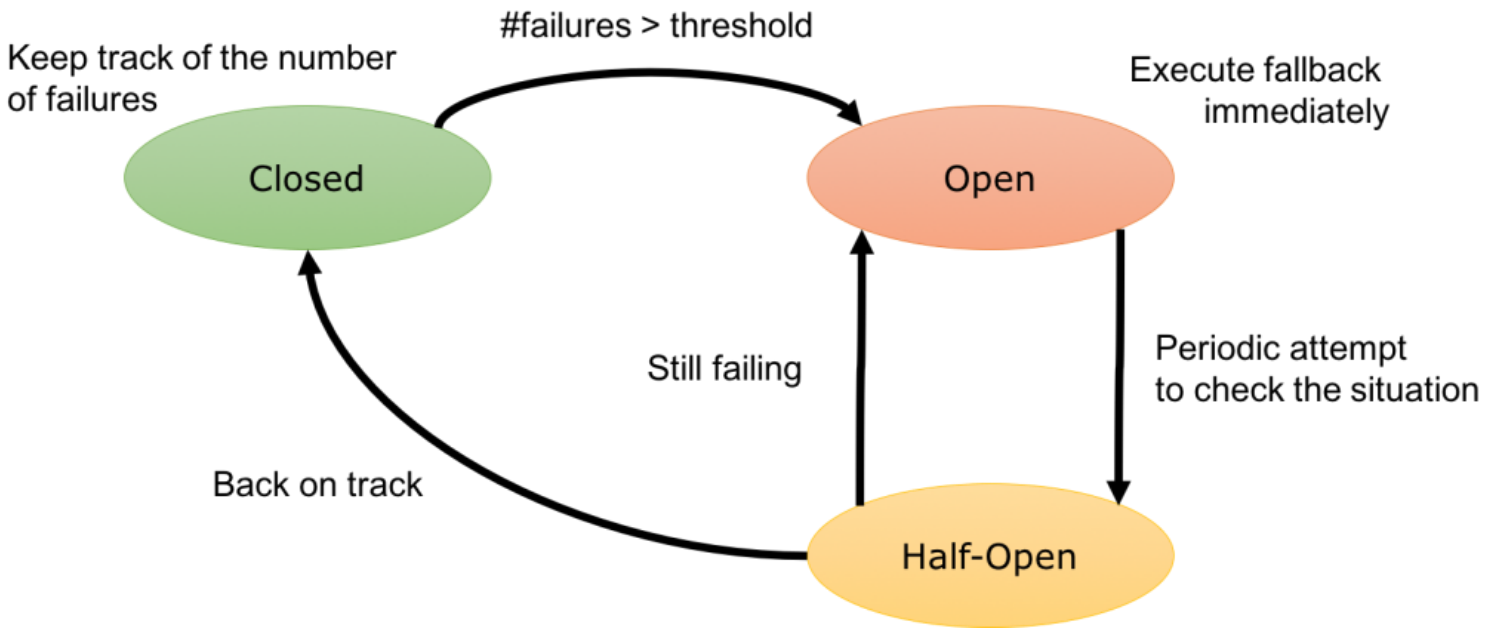


- A MovieService might fail or go down due to a network error, or when the service is under maintenance for a short time.
- In this period, if a client makes a service request, then the request will have to wait in a queue to get processed. This is for all requests that follow.
- The requests will keep waiting until the service is up. Once the service resumes, it will have to process multiple requests simultaneously which might lead to failure or overload.
- The circuit breaker helps in breaking the wait time by setting a time out. If the requests are not processed within the time frame all requests return with an error.
- This prevents overload or failure of the service.

Circuit Breaker

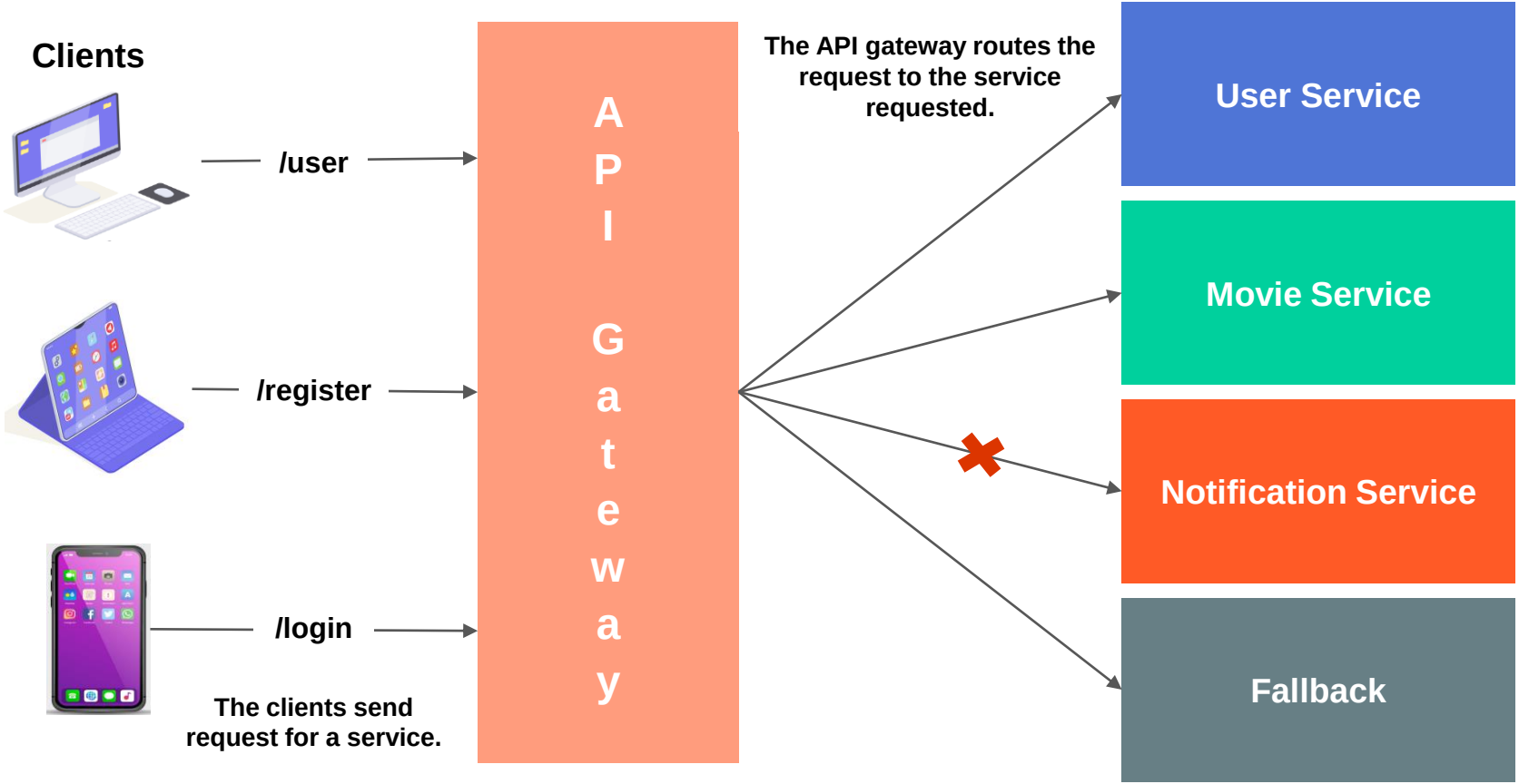
- Fault tolerance can be achieved in microservices with the help of a circuit breaker.
- It is a pattern that wraps requests to external services and detects when they fail.
- If a failure is detected, the circuit breaker opens.
- All the subsequent requests immediately return an error instead of making requests to the unhealthy service.
- It monitors and detects the service which is down and misbehaves with other services. It rejects calls until it becomes healthy again.

Circuit Breaker



A Circuit Breaker can have three states: Open, Closed, and Half-Open.

Circuit Breaker - Hystrix



Implementing Circuit Breaker Using Hystrix

Hystrix in a Spring Boot Application

- Add the `spring-cloud-starter-netflix-hystrix` Maven dependency.

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
    <version>2.2.9.RELEASE</version>
</dependency>
```

- Add the `@EnableHystrix` to the Application class.

```
@SpringBootApplication
@EnableEurekaClient
@EnableHystrix
public class UserAuthenticationServiceApplication {

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

}
```

Hystrix in a Spring Boot Application

- Add `@HystrixCommand` to the method that requires circuit breaking.

```
@PostMapping("/login")
@HystrixCommand(fallbackMethod = "fallbackLogin",commandKey = "loginKey",groupKey = "login")
@HystrixProperty(name = "execution.isolation.thread.timeoutInMilliseconds", value = "1000")
public ResponseEntity<?> loginUser(@RequestBody User user) throws InvalidCredentialsException
{
    User retrievedUser = userService.findByEmailAndPassword(user.getEmail(),user.getPassword());

    if(retrievedUser==null)
    {
        throw new InvalidCredentialsException();
    }
    Map<String,String> map = securityTokenGenerator.generateToken(user);
    return new ResponseEntity<>(map,HttpStatus.OK);
}
```

- The `@HystrixProperty` can be used to set the timeout value.

Hystrix in a Spring Boot Application

- Configure the Hystrix Behavior.

```
public ResponseEntity<?> fallbackLogin(@RequestBody User user){
    String msg = "login failed";
    return new ResponseEntity<>(msg,HttpStatus.GATEWAY_TIMEOUT);
}
```

- Note that the fallback method must have the same signature as the method to which circuit breaking is applied.
- An optional feature of Hystrix is the ability to monitor its status on a dashboard.
- To enable it, add spring-cloud-starter-hystrix-dashboard and spring-boot-starter-actuator in the pom.xml of the spring boot application.

Quick Check

How does the service discovery server know that a service is down?

1. Heartbeat
2. Pulse
3. Time out
4. Discovery

Quick Check: Solution

How does the service discovery server know that a service is down?

1. Heartbeat
2. Pulse
3. **Time out**
4. Discovery

Streaming Application

Consider a streaming application that enables users to watch movies on any smart device. The application provides multiple features to all its registered users. A user needs to register with the application in order to access some of its features. Let us create multiple microservices for the streaming application.

1. A user must first register with the application.
2. Use credentials such as id, password to login.
3. Access the features provided by the streaming application, like adding favourites, compiling a watch later list, etc.

Let us create a parent project called **MovieApplication**.
This will contain the **UserAuthenticationService** and the **UserMovieService** as microservices.
Create a circuit breaker on the client side in the **UserAuthenticationService** . **Dockerize the application.**

DEMO

Key Takeaways

- Failure in Microservices
- Circuit Breaker Pattern
- Circuit Breaker using Hystrix