Concept	Definition	Real-Time Example	Tools Used	Implementation (Simple Steps)
Failover	Automatically switches to a backup instance when one fails.	If one **payment service** instance goes down, another instance handles the request.	☑ Spring Cloud Load Balancer ☑ Eureka (Service Discovery) ☑ OpenFeign (Declarative Rest Client)	1 Register services in **Eureka** 2 Use **Load Balanced RestTemplate** or **Feign Client** 3 If one instance fails, request goes to another
Fault Tolerance	Ensures system keeps working despite failures by handling errors gracefully.	If **all** payment service instances are down, return a default message instead of breaking the system.	☑ Resilience4J (Circuit Breaker) ☑ Spring Retry ☑ Hystrix (Deprecated)	1 Add **Resilience4J** dependency 2 Use **@CircuitBreaker** annotation with a fallback method 3 If a service fails, return a **default response** instead of an error
Netflix OSS	A set of tools by Netflix for building resilient microservices.	Used by **Netflix, Amazon, Uber** for scalable systems.	☑ Eureka ☑ Hystrix ☑ Zuul (API Gateway)	1 Use **Eureka** for service discovery 2 Use **Resilience4J** for circuit breaking
Spring Cloud	A framework that helps build **microservices easily** in Spring Boot.	Used to develop large **e-commerce apps** like Amazon.	☑ Spring Cloud Eureka ☑ Spring Cloud Gateway ☑ Spring Cloud Load Balancer	1 Register services in **Eureka** 2 Use **Spring Cloud Gateway** for API management
Eureka (Service Discovery)	Keeps track of running microservices and their availability.	When a **new microservice** starts, it registers itself in Eureka.	☑ Eureka Server ☑ Eureka Client	1 Start **Eureka Server** 2 Register services in **Eureka** 3 Use Eureka for **service discovery**

Concept	Definition	Real-Time Example	Tools Used	Implementation (Simple Steps)
Spring Cloud Load Balancer	Distributes requests across multiple service instances to prevent overload.	If **Order Service** calls **Payment Service**, traffic is distributed across available instances.	☑ Spring Cloud Load Balancer ☑ Ribbon (Deprecated)	1️⃣ Enable **@LoadBalanced RestTemplate** 2️⃣ Use **Feign Client** for automatic balancing
Implementing Microservices with Eureka & Load Balancer	Uses Eureka to register services and Load Balancer to distribute requests.	An **Order Service** finds a **Payment Service** instance through Eureka.	☑ Eureka ☑ Spring Cloud Load Balancer	1️⃣ Register services in **Eureka** 2️⃣ Use **Feign Client** for service calls 3️⃣ Load Balancer automatically distributes traffic
OpenFeign (Declarative REST Client)	A simpler way to call REST APIs without writing boilerplate code.	Instead of manually calling `RestTemplate`, just define an **interface**.	☑ OpenFeign ☑ Spring Cloud Load Balancer	1️⃣ Add `@FeignClient(name="SERVICE-NAME")` 2️⃣ Define API methods inside the interface 3️⃣ Feign automatically handles API calls
Circuit Breaker (Resilience4J)	Prevents **cascading failures** by stopping calls to failing services.	If a service **keeps failing**, the circuit **opens** and stops further calls.	☑ Resilience4J ☑ Spring Boot Actuator	1️⃣ Add `@CircuitBreaker(name="service", fallbackMethod="fallback")` 2️⃣ Return a **default response** if the service is down
Monitoring Circuit Breakers (Prometheus)	Tracks failures and uptime of services.	**Admin dashboards** show how many times a service failed.	☑ Prometheus ☑ Spring Boot Actuator	1️⃣ Enable **Spring Boot Actuator** 2️⃣ Expose **/metrics** endpoint 3️⃣ Connect Prometheus for monitoring
Spring Cloud Config (Server & Client)	Centralized configuration management for microservices.	**All services fetch config from one central location.**	☑ Spring Cloud Config Server ☑ Spring Cloud Config Client	1️⃣ Set up **Config Server** 2️⃣ Clients fetch configs from the server 3️⃣ Use **@RefreshScope** to apply changes dynamically

Concept	Definition	Real-Time Example	Tools Used	Implementation (Simple Steps)
Spring Cloud Gateway (API Gateway)	A single entry point for all microservices.	**Filters requests**, manages authentication, and routes traffic.	☑ Spring Cloud Gateway ☑ Spring Security	1️⃣ Define **routes in application.yml** 2️⃣ Add **global filters** for security and logging
Distributed Log Tracing (Sleuth & Zipkin)	Tracks requests across multiple microservices.	**Find which service is slow** or failing.	☑ Spring Cloud Sleuth ☑ Zipkin ☑ Prometheus	1️⃣ Enable **Sleuth** in each microservice 2️⃣ Send logs to **Zipkin for visualization**
Spring Security - Basic Auth	Protects APIs using **username & password**.	Login required for **admin pages**.	☑ Spring Security	1️⃣ Add `spring.security.user.name` & `password` in properties 2️⃣ API calls require authentication
Spring Security - CORS	Controls which domains can call an API.	Allows **only trusted domains** to access APIs.	☑ Spring Security	1️⃣ Enable CORS in `SecurityConfig` 2️⃣ Specify allowed origins & methods
Spring Security - JWT (JSON Web Token)	Uses tokens for authentication instead of session-based login.	**Mobile apps & SPAs** use JWT for login.	☑ Spring Security ☑ jjwt library	1️⃣ Generate JWT on login 2️⃣ Pass JWT in request headers 3️⃣ Validate JWT before processing requests
Spring Security - OAuth 2.0	Allows authentication via **Google, Facebook, GitHub, etc.**	**Login with Google, Facebook, etc.**	☑ Spring Security OAuth2 ☑ Keycloak	1️⃣ Configure OAuth 2.0 provider 2️⃣ Redirect users for authentication 3️⃣ Use access tokens for API calls
SSL/TLS Certificates	Secures communication between services.	**HTTPS instead of HTTP** for security.	☑ Self-signed SSL ☑ Let's Encrypt	1️⃣ Generate **SSL certificate** 2️⃣ Configure Spring Boot to use HTTPS 3️⃣ Deploy with **TLS encryption**

🔥 Key Takeaways

- ✔ **Failover** = Switch to another instance when failure occurs (Eureka + Load Balancer).
- ✔ **Fault Tolerance** = Handle failures **gracefully** using **circuit breakers** (Resilience4J).
- ✔ **Netflix OSS & Spring Cloud** = Tools for **building scalable microservices**.
- ✔ **Spring Security** = Protect APIs with **Basic Auth, JWT, OAuth 2.0, and SSL/TLS**.
- ✔ **Spring Cloud Gateway & Distributed Tracing** = API management + Debugging tools.