**What Are Microservices?**

**Microservices** is an architectural style where a large application is composed of small, independently deployable services. Each service is:

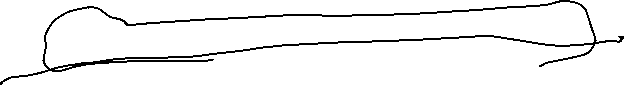
* Loosely coupled
* Focused on a single business capability
* Can be developed, deployed, and scaled independently

**🚀 Why Spring for Microservices?**

Spring ecosystem provides powerful tools to build microservices:

* ✅ **Spring Boot** – Simplifies RESTful API development
* ✅ **Spring Cloud** – Enables distributed systems patterns like config management, service discovery, circuit breakers, etc.
* ✅ **Spring Cloud Gateway** – API gateway for routing and filtering
* ✅ **Spring Security** – Authentication & authorization

**🧱 Microservices Architecture Components**



| **Component** | **Spring Tool** | **Description** |
| --- | --- | --- |
| **Service Registry** | Spring Cloud Netflix Eureka | Keeps track of services |
| **API Gateway** | Spring Cloud Gateway / Zuul | Routes requests to services |
| **Config Server** | Spring Cloud Config | Centralized configuration |
| **Load Balancer** | Spring Cloud LoadBalancer | Distributes requests across instances |
| **Circuit Breaker** | Resilience4j / Hystrix | Prevents cascading failures |
| **Tracing** | Sleuth + Zipkin | Distributed tracing |
| **Security** | Spring Security + JWT | Secures endpoints |
| **Messaging** | Spring Cloud Stream + Kafka/RabbitMQ | Async communication |

**🧩 Sample Microservices Scenario**

Imagine building **Verizon Telecom Microservices**:

* 📱 sms-service → Handles SMS sending & history
* 📞 call-service → Manages call logs
* 👤 user-service → Manages users/subscribers
* 🌐 api-gateway → Unified entry point
* 🧾 billing-service → Handles billing info

**🛠️ Tools & Tech Stack**

| **Layer** | **Tool** |
| --- | --- |
| Backend Framework | Spring Boot |
| Microservices Infra | Spring Cloud |
| API Gateway | Spring Cloud Gateway |
| Config Management | Spring Cloud Config |
| Service Discovery | Eureka |
| Communication | REST, Kafka, RabbitMQ |
| Database | MySQL, MongoDB |
| Auth | Spring Security + JWT |
| Monitoring | Actuator, Zipkin, Sleuth |

**📁 Folder Structure (Microservice Setup)**

verizon-telecom/

├── api-gateway/

├── config-server/

├── discovery-server/

├── sms-service/

├── call-service/

├── user-service/

├── billing-service/

Each module is a separate Spring Boot app with:

* Its own pom.xml
* Own application.yml
* Separate DB if needed

**🔐 Secure Your Microservices**

Use Spring Security with JWT:

* Gateway verifies token
* Internal services trust gateway
* Avoid exposing microservices directly