Spring Boot and Microservices

What Are Microservices?

- An architecture style where applications are built as a collection of independent, loosely coupled services
- Each service:
  - Has a single responsibility
  - Runs in its own process
  - Communicates over HTTP/REST or messaging

Benefits of Microservices

- Easier to develop and deploy independently
- Enables scalability per component
- Teams can work on different services in parallel
- Services can use different tech stacks if needed

Challenges with Microservices

- Service discovery: how do services find each other?
- Communication failures: one service might be down
- Monitoring and debugging becomes harder
- Security, configuration, and consistency across services

Spring Boot for Microservices

- Spring Boot is ideal for microservices:
  - Lightweight and easy to deploy
  - Embedded server (Tomcat/Jetty)
  - RESTful APIs with @RestController
- Used to build individual services quickly

Spring Cloud for Distributed Systems

- Spring Cloud provides tools for:
  - Service discovery
  - API gateway
  - Circuit breakers
  - Central configuration
  - Distributed tracing
- Works on top of Spring Boot

Types of Microservices in Spring Cloud

- API Gateway - Single entry point, routing, filtering
- Service Discovery - Services register & discover each other
- Circuit Breaker - Handle service failure gracefully
- Config Server - Centralized configuration
- Tracing & Monitoring - Track requests across services

API Gateway

- Entry point to your microservices system
- Responsibilities:
  - Routing requests to correct services
  - Rate limiting, authentication, logging

Circuit Breaker

- Prevents cascading failure when a service is down
- Instead of failing repeatedly, fallback logic is used

Service Discovery with Eureka

- Services register themselves on Eureka Server
- Clients query Eureka to find service locations dynamically
- Avoids hardcoding service URLs

Summary — Spring Boot + Spring Cloud

- Use Spring Boot to build each microservice
- Use Spring Cloud to handle:
  - Discovery
  - Gateway
  - Circuit Breaking
  - Configuration
- Build resilient, scalable distributed systems

Demo 1 - API Gateway

- Spring Boot application acting as a API Gateway