MICROSERVICES COURSE

Module 1: Introduction to Microservices

1.1 Understanding Microservices Architecture

1.1.1 What are microservices?
1.1.2 Contrasting monolithic and microservices architectures.
Benefits and challenges of microservices.

1.2. Microservices Principles and Patterns

1.2.1 Key principles of microservices.
1.2.2 Common architectural patterns (e.g., API Gateway, Service Discovery).
1.2.3 Microservices communication (e.g., REST, gRPC).

Module 2: Designing Microservices

2.1. Domain-Driven Design (DDD)

2.1.1 Identifying bounded contexts.
2.1.2 Aggregates, entities, and value objects.
2.1.3 Building a domain model.

2.2. Microservices Decomposition

2.2.1Strategies for breaking down monoliths.
2.2.2 Service granularity.
2.2.3 Data management strategies (polyglot persistence).

2.3. Microservices Data Management

2.3.1 Database per service.
2.3.2 Event sourcing and CQRS (Command Query Responsibility Segregation).
2.3.2 Data consistency and transactions.

Module 3: Developing Microservices

3.1. Microservices Development Tools and Frameworks

3.1.1Choice of programming languages.
3.1.2 Containerization with Docker.
3.1.3 Service frameworks (e.g., Spring Boot, Node.js).

3.2. Microservices Testing

3.2.1 Unit testing and integration testing.
3.2.2 Contract testing and consumer-driven contracts.
3.2.3 Testing in isolation and with dependencies.

3.3. Continuous Integration and Deployment (CI/CD)

3.3.1 Building CI/CD pipelines for microservices.
3.3.2 Automating deployments with Kubernetes or other orchestration tools.

Module 4: Microservices Communication and Orchestration

4.1. Inter-Service Communication

4.1.1 RESTful APIs and HTTP.
4.1.2 Message brokers (e.g., RabbitMQ, Kafka).
4.1.3 RPC for high-performance communication.

4.2. Service Discovery and Load Balancing

4.2.1 Service registration and discovery.
4.2.2 Load balancing strategies.
4.2.3 Circuit breakers and fault tolerance.

Module 5: Monitoring and Logging

5.1. Microservices Observability

5.1.1 Logging, tracing, and metrics.
5.1.2 Tools like Prometheus, Grafana, and ELK stack.

5.1.3Distributed tracing with tools like Zipkin and Jaeger.

5.2. Performance Optimization

5.2.1 Identifying performance bottlenecks.
Scaling strategies.
5.2.2 Auto-scaling and resource management.

Module 6: Security and Authentication

6.1. Microservices Security Principles

6.1.1 Securing communication.
6.1.2 Authorization and authentication.
6.1.3 API security best practices.

6.2. OAuth and JWT

6.1 Implementing OAuth for API security.
6.2 Using JWT for authentication.

Module 7: Microservices Deployment and Management

7.1. Container Orchestration

7.1.1 Kubernetes and container orchestration basics.

Deployment strategies and rolling updates.

7.2. Service Mesh

7.1.1 Introduction to service mesh (e.g., Istio, Linkerd).

7.1.2 Traffic management and security with service mesh.

Module 8: Best Practices and Case Studies

8.1. Microservices Best Practices

8.1.1 Lessons learned from real-world microservices projects.

8.1.2 Common pitfalls and how to avoid them.

8.2. Case Studies

8.2.1 Examining successful microservices implementations.

8.2.2 Architecture review of well-known microservices applications.