Microservices Architecture - Study Guide

1. What is Microservices Architecture?

Microservices Architecture is a software design pattern where a large application is broken down into small, independent services. Each service performs a specific business function and communicates with other services via APIs or messaging systems. These services are independently deployable and scalable.

2. Key Principles of Microservices

- Single Responsibility Principle
- Decentralized Data Management
- Independent Deployment
- Technology Agnostic Services
- Resilience and Fault Isolation

3. Microservices Architecture Diagram

Client -> API Gateway -> [Student Service, Result Service, Notification Service] -> Databases

Each service communicates with its own DB and may publish/consume messages through a queue (RabbitMQ/Kafka).

4. Communication Types

- Synchronous: HTTP REST, gRPC
- Asynchronous: RabbitMQ, Kafka (event-driven architecture)

5. Tools & Technologies

- API Gateway: Ocelot, YARP
- Messaging: RabbitMQ, Kafka
- Containerization: Docker
- Orchestration: Kubernetes
- Monitoring: Prometheus, Grafana
- Logging: Serilog, ELK Stack

Microservices Architecture - Study Guide

6. .NET Core for Microservices

- ASP.NET Core Web API for services
- EF Core for data access
- MassTransit for message handling
- Ocelot for API Gateway
- Docker & Docker Compose for containerization

7. Security in Microservices

- Authentication using IdentityServer4, JWT
- HTTPS enforcement
- Rate Limiting, API Keys at Gateway level

8. Deployment & CI/CD

- Docker images for services
- Kubernetes for orchestration
- GitHub Actions, Azure DevOps for pipelines

9. Monitoring and Observability

- Health checks
- Centralized logging
- Distributed tracing

10. Learning Resources

- Microsoft Learn: .NET Microservices
- Official Docs: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/
- GitHub Repositories & Sample Projects