**Dotnet Micro service course content**

**Duration: 5 Days**

**Prerequisite**

**Participants should have strong knowledge on ASP.NET 8**

**Lab Setup**

**Visual Studio 2022 Enterprise**

**SQL Server 2018 and Management Studio**

**Docker Desktop with Kubernetes**

**🧭 Training Objectives:**

**By the end of this training, participants will be able to:**

1. **Understand Microservices Architecture**
   * **Core concepts, advantages, and trade-offs**
   * **Monolith vs Microservices**
2. **Design Microservices using .NET Core**
   * **Create RESTful APIs using ASP.NET Core**
   * **Implement API Gateway patterns and service discovery**
   * **Use Entity Framework Core with Microservices**
3. **Containerize Microservices using Docker**
   * **Write Dockerfiles for .NET applications**
   * **Build, run, and debug containers locally**
   * **Manage multi-container apps using Docker Compose**
4. **Implement CI/CD using GitLab**
   * **Understand GitLab pipelines, runners, and YAML scripting**
   * **Build Docker images and push to GitLab Container Registry**
   * **Automate testing and deployment with GitLab CI/CD**
5. **Configure Service Communication and Observability**
   * **REST, gRPC, and messaging via RabbitMQ/Kafka**
   * **Logging with Serilog and monitoring with Prometheus + Grafana**
6. **Secure and Deploy Microservices**
   * **Implement authentication and authorization with IdentityServer or JWT**
   * **Use GitLab environments to deploy to dev/staging/production**
   * **Optional: Deploy to Kubernetes (basic intro)**

**🎯 Training Outcome:**

**After completing this training, participants will:**

* **Be able to build microservices in .NET Core from scratch**
* **Know how to containerize and run services using Docker**
* **Create and configure GitLab CI/CD pipelines for building, testing, and deploying microservices**
* **Understand real-world DevOps practices with GitLab and Docker**
* **Be ready to deploy services to cloud or on-prem environments**
* **Apply monitoring, logging, and security best practices to microservices**

**👥 Target Audience:**

* **.NET Developers looking to transition to microservices architecture**
* **DevOps Engineers wanting to understand .NET microservices deployment**
* **Software Architects designing scalable and container-based systems**
* **QA/Test Automation Engineers integrating test stages in CI/CD pipelines**
* **Technical Leads & Engineers managing or migrating legacy systems**
* **Cloud/Platform Engineers deploying .NET services with GitLab**

**Day 1 and Day2**

A screenshot of a computer

Description automatically generated

1. Microservices: building blocks (done)
2. Microservices: Solver of problems architecture (done)
3. Microservices and cloud-native
4. Microservices Core Advance Concepts
5. Microservices Advanced Concepts

A screenshot of a computer

Description automatically generated with medium confidence

Day 3

1. Asynchronous Communications (Kafka), (Rabbitmq)
2. Hybrid architectures: Hierarchy and service-based
3. Making Architecture Choices
4. Testing for microservices
5. Secure microservices (JWT)

Day 4

1. Monitor microservices (Prometheus and Grafana)
2. Deep framework implementation microservices
3. Design resilient distributed microservices systems (Polly)
4. Deploy effective microservices
5. Why CI/CD?
   * Overview of GitLab’s CI/CD architecture and terminology (pipelines, jobs, stages)
   * **Building Your First Pipeline**
     + Anatomy of .gitlab-ci.yml: stages, jobs, scripts
     + Defining stages such as build, test, deploy
   * **Hands-On:** Write a simple pipeline that builds, tests, and reports status
   * **Managing Variables and Secrets**
     + Protected vs. masked variables
     + Project-level and group-level scopes
   * **Hands-On:** Store database credentials and API keys as masked variables and consume them in jobs
   * **Artifacts and Caching**
     + Persisting build outputs (artifacts) between stages
     + Speeding up builds with dependency caches (e.g., NPM, Maven)
   * **Hands-On:** Configure cache for package dependencies and collect build artifacts

Day 5

1. Embracing DevOps (Docker and Kubernetes)

• Docker Workflows in CI

o Building Docker images within pipelines

o Pushing images to GitLab’s Container Registry

• Hands-On: Create a job that builds your sample app’s Docker image and pushes it to the registry

* **GitOps-Style Continuous Delivery**
  + Principles of GitOps and infrastructure as code
  + Triggering Argo CD or Terraform from .gitlab-ci.yml
* **Hands-On:** Write a pipeline job that applies Terraform changes to a dev cluster

1. CI /Continuous Delivery as a requirement
   * GitLab
   * **Hands-On:**
   * Create and include reusable pipeline templates
   * Configure dynamic review environments (review/$CI\_COMMIT\_REF\_NAME)
   * Implement a manual “Promote to Production” job
   * Secure credentials via masked variables