**Dotnet Micro service course content**

**Duration: 5 Days**

**Day 1**

.NET Microservices: Architecture for Containerized .NET Applications

Introduction to Containers and Docker

What is Docker?

Docker terminology

Docker containers, images, and registries

Choosing Between NET Core and NET Framework for Docker Containers

General guidance

When to choose .NET Core for Docker containers

When to choose .NET Framework for Docker containers

Decision table: .NET frameworks to use for Docker

What OS to target with .NET containers

Official .NET Docker images

Architecting Container and Microservice Based Applications

Containerizing monolithic applications

State and data in Docker applications

**Day 2**

Service-oriented architecture

Microservices architecture

Data sovereignty per microservice

Logical architecture versus physical architecture

Challenges and solutions for distributed data management

Identifying domain-model boundaries for each microservice

Communication between microservices

Asynchronous message-based communication

Creating, evolving, and versioning microservice APIs and contracts

Microservices addressability and the service registry

Creating composite UI based on microservices, including visual UI shape and layout

generated by multiple microservices

Resiliency and high availability in microservices

**Day 3**

Orchestrating microservices and multi-container applications for high scalability and

availability

Using Azure Service Fabric

Development Process for Docker Based Applications

Development workflow for Docker apps

Deploying Single Container Based NET Core Web Applications on Linux or Windows

Nano Server Hosts

Migrating Legacy Monolithic NET Framework Applications to Windows Containers

Designing and Developing Multi Container and Microservice Based NET Applications

Designing a microservice-oriented application

Creating a simple data-driven CRUD microservice

Defining your multi-container application with docker-compose.yml

Using a database server running as a container

Implementing event-based communication between microservices (integration

events)

Implementing an event bus with RabbitMQ for the development or test environment

Subscribing to events

**Day 4**

Testing ASP.NET Core services and web apps

Tackling Business Complexity in a Microservice with DDD and CQRS Patterns

Applying simplified CQRS and DDD patterns in a microservice

Applying CQRS and CQS approaches in a DDD microservice in eShopOnContainers

Implementing reads/queries in a CQRS microservice

Designing a DDD-oriented microservice

Designing a microservice domain model

Implementing a microservice domain model with .NET Core

Seedwork (reusable base classes and interfaces for your domain model)

Implementing value objects

Using Enumeration classes instead of enum types

Designing validations in the domain model layer

Client-side validation (validation in the presentation layers)

Domain events: design and implementation

Designing the infrastructure persistence layer

Implementing the infrastructure persistence layer with Entity Framework Core

Using NoSQL databases as a persistence infrastructure

Designing the microservice application layer and Web API

Implementing the microservice application layer using the Web API

Implementing Resilient Applications

Handling partial failure

Strategies for handling partial failure

Implementing retries with exponential backoff

Implementing resilient Entity Framework Core SQL connections

Implementing custom HTTP call retries with exponential backoff

Implementing HTTP call retries with exponential backoff with Polly

Implementing the Circuit Breaker pattern

Health monitoring

**Day 5**

Securing NET Microservices and Web Applications

About authorization in .NET microservices and web applications

Storing application secrets safely during development

Using Azure Key Vault to protect secrets at production time

Key takeaways