Architecting Cloud-Native .NET Apps for Azure

Duration: 40 hours

Targeted Audience profile:

The audience for this course is mainly developers, development leads, and architects who are interested in learning how to build applications designed for the cloud.

A secondary audience is technical decision-makers who plan to choose whether to build their applications using a cloud-native approach.

How this will benefit you:

This course begins by defining cloud native and introducing a reference application built using cloud-native principles and technologies. Beyond these first two chapters, the rest of the course is broken up into specific chapters focused on topics common to most cloud-native applications. You can learn about cloud-native approaches to:

- Data and data access
- Communication patterns
- Scaling and scalability
- Application resiliency
- Monitoring and health
- Identity and security
- DevOps

Content

1. **Introduction to Cloud-Native Applications**
    - Introduction to Cloud-native computing
    - What is Cloud Native?
    - The pillars of cloud native
        - The cloud
        - Modern design
        - Microservices
        - Containers
        - Backing services
        - Automation
    - Candidate apps for cloud native
        - Modernizing legacy apps

2. **Introducing eShopOnContainers Reference App**
    - Features and requirements
    - Overview of the code
    - Understanding microservices
    - Mapping eShopOnContainers to Azure Services
        - Container orchestration and clustering
        - API Gateway
        - Data
        - Event Bus
        - Resiliency
    - Deploying eShopOnContainers to Azure
        - Azure Kubernetes Service
        - Deploying to Azure Kubernetes Service using Helm
        - Azure Functions and Logic Apps (Serverless)
    - Centralized configuration
        - Azure App Configuration
        - Azure Key Vault
        - Configuration in eShop
3. **Scaling Cloud-Native Applications**
    - Leveraging containers and orchestrators
        - Challenges with monolithic deployments
        - What are the benefits of containers and orchestrators?
        - What are the scaling benefits?
        - What scenarios are ideal for containers and orchestrators?
        - When should you avoid using containers and orchestrators?
        - Development resources
    - Leveraging serverless functions
        - What is serverless?
        - What challenges are solved by serverless?
        - What is the difference between a microservice and a serverless function?

        - What scenarios are appropriate for serverless?
        - When should you avoid serverless?
    - Combining containers and serverless approaches
    - Deploying containers in Azure
        - Azure Container Registry
        - ACR Tasks
        - Azure Kubernetes Service
        - Azure Bridge to Kubernetes
    - Scaling containers and serverless applications
        - The simple solution: scaling up
        - Scaling out cloud-native apps
    - Other container deployment options
4. **Cloud-Native Communication Patterns**
    - Communication considerations
    - Front-end client communication
        - Simple Gateways
        - Azure Application Gateway
        - Azure API Management
        - Real-time communication
    - Service-to-service communication
        - Queries
        - Commands
        - Events
    - gRPC
        - What is gRPC?
        - gRPC Benefits
        - Protocol Buffers
        - gRPC support in .NET
        - gRPC usage
        - gRPC implementation
        - Looking ahead

5. **Cloud-Native Data Patterns**
    - Database-per-microservice, why?
    - Cross-service queries
    - Distributed transactions
    - High volume data
        - CQRS
        - Event sourcing
    - Relational vs. NoSQL data
        - The CAP theorem
        - Considerations for relational vs. NoSQL systems
        - Database as a Service
        - Azure relational databases
        - Azure SQL Database
        - Open-source databases in Azure
        - NoSQL data in Azure
        - Data migration to the cloud
    - Caching in a cloud-native app
        - Why?
        - Caching architecture
        - Azure Cache for Redis
    - Elasticsearch in a cloud-native app
6. **Cloud-Native Resiliency**
    - Application resiliency patterns
        - Retry pattern
        - Circuit breaker pattern
        - Testing for resiliency
    - Azure platform resiliency
        - Design with resiliency
        - Design with redundancy
        - Design for scalability
        - Built-in retry in services

- Resilient communications
    - Service mesh
    - Istio and Envoy
    - Integration with Azure Kubernetes Services

7. **Monitoring and Health**
    - Observability patterns
        - When to use logging
        - Challenges with detecting and responding to potential app health issues
        - Challenges with reacting to critical problems in cloud-native apps
    - Logging with Elastic Stack
        - Elastic Stack
        - What are the advantages of Elastic Stack?
        - Logstash
        - Elasticsearch
        - Visualizing information with Kibana web dashboards
        - Installing Elastic Stack on Azure
    - Monitoring in Azure Kubernetes Services
        - Azure Monitor for Containers
    - Log.Finalize()
    - Azure Monitor
        - Gathering logs and metrics
        - Reporting data
        - Dashboards
        - Alerts
    - References
8. **Cloud-Native Identity**
    - Authentication and authorization in cloud-native apps
    - Azure Active Directory
    - IdentityServer for cloud-native applications
        - Common web app scenarios
        - Getting started

        - Configuration
        - JavaScript clients
9. **Cloud-Native Security**
    - Azure security for cloud-native apps
    - Threat modeling
    - Principle of least privilege
    - Penetration testing
    - Monitoring
    - Securing the build
    - Building secure code
    - Built-in security
    - Azure network infrastructure
    - Role-based access control for restricting access to Azure resources
    - Securing secrets
        - Azure Key Vault
        - Kubernetes
    - Encryption in transit and at rest
        - In transit
        - At rest
    - Keeping secure
10. **DevOps**
    - Azure DevOps
    - GitHub Actions
    - Source control
        - Repository per microservice
        - Single repository
        - Standard directory structure
    - Task management
    - CI/CD pipelines
        - Azure Builds
        - Azure DevOps releases

    - Everybody gets a build pipeline
    - Versioning releases
- Feature flags
    - Implementing feature flags
- Infrastructure as code
    - Azure Resource Manager templates
    - Terraform
    - Azure CLI Scripts and Tasks
- Cloud Native Application Bundles
- DevOps Decisions