Azure DevOps and Cloud Capstone Project: Full Lifecycle Overview

Project Goal:

The goal of this Azure DevOps and Cloud Capstone Project is to equip engineers with the skills and experience needed to design, implement, and manage a complete end-to-end DevOps lifecycle in a real-world cloud environment.

Through this project, engineers will:

- Establish a strong DevOps culture focusing on collaboration, automation, and continuous improvement.
- Implement version control strategies using Git within Azure DevOps.
- Build secure, scalable infrastructure on Azure using Infrastructure as Code (Terraform), including the creation of production-grade AKS clusters, Azure Container Registry (ACR), and resilient networking setups.
- Design and deploy a **high availability (HA)** and **disaster recovery (DR)** based architecture across multiple Azure regions using Active-Passive AKS cluster design.
- Develop and automate **CI/CD pipelines** for application build, containerization, deployment, and release management using Azure DevOps Pipelines, GitHub Actions, and Terraform.
- Enforce robust **security and compliance** practices leveraging Azure Key Vault, Azure Defender, Network Security Groups, and Azure Policy.
- Implement **monitoring, logging, and alerting** with Azure Monitor and Application Insights, ensuring observability into application and infrastructure health.
- Define and track **SLA, SLO, and SLI** targets following **Site Reliability Engineering (SRE)** standards.
- Simulate failover scenarios to validate the disaster recovery design.
- Document all strategies, configurations, processes, and establish effective **communication and feedback loops** across teams.

Project Details

Phase 1: Planning and Version Control

1.1 DevOps Principles & Culture

- Define a culture emphasizing **collaboration, automation, continuous improvement**.

- Establish **clear communication channels** and **feedback loops**.
- Document the DevOps strategy including:
    - Deployment frequency, lead time, MTTR, and change failure rate.
    - KPIs and performance goals.

1.2 DevOps vs Traditional SDLC

- Contrast DevOps with waterfall/agile SDLC.
- Highlight benefits: **speed**, **quality**, **resilience**, **customer feedback loops**.

1.3 Site Reliability Engineering (SRE)

- Define **SLOs (Service Level Objectives)** and **SLIs (Service Level Indicators)** for the web app.
- Set up **basic monitoring and alerting** based on the SLOs/SLIs.

1.4 Azure DevOps Overview

- Use core Azure DevOps components:
    - **Boards** (Agile project management)
    - **Repos** (Source control)
    - **Pipelines** (CI/CD)
    - **Artifacts** (Package management)

1.5 Git and Gitflow

- Implement **Gitflow** for feature, release, and hotfix management.
- Set up:
    - Code review and pull request workflows.
    - Git hooks for pre-commit checks.
    - Branch policies in Azure Repos (e.g., require PR validation).
    - Pull request linking with work items (track features & bugs).
- Define CI/CD pipeline's
- Integrate **static code analysis** into the CI build.

Phase 2: Infrastructure as Code (IaC)

2.1 GitHub Integration

- Integrate GitHub with **Azure AD** for centralized identity.
- Explore **Changelogs** to track infrastructure changes.

2.2 Terraform IaC

- Provision critical Azure resources:
    - **AKS Clusters**
    - **Azure Container Registry (ACR)**
    - **VNets**, **Subnets**, **NAT Gateway**, **Azure Bastion**.
- Implement **Terraform linting** for code quality.

2.3 Azure Pipelines for Infra

- Automate Terraform deployments using Azure Pipelines.

✨ **Assignment 1: Identity and Access Management**

- **Create a Budget** in Azure.
- Create user roles with **Owner, Contributor, Reader** access at:
    - Subscription scope
    - Resource Group scope
    - Resource Level scope

✨ **Assignment 2: VNet and Connectivity**

Create a **Terraform Module** to:

- Deploy a **VNet** with:
    - Public and private subnets
    - NAT Gateway connection for private subnet
    - Azure Bastion Host for secure management

Use the module to:

- Create **two VNets** in different regions.
- **Peer** them together (VNet peering).
- Open **only necessary ports** for minimal exposure.

Create a **third VNet** to simulate an **on-premises environment**:

- Configure **site-to-site VPN** between Azure and simulated on-prem VNet.
- Take backup of the "on-prem" environment in Azure (snapshot, or backup service).

Phase 3: Continuous Integration and Continuous Deployment (CI/CD)

3.1 Azure DevOps CI/CD Pipelines

- Create **YAML pipelines** for:
    - **Build**: application code, infrastructure modules.
    - **Release**: deploy to multiple AKS clusters.
- Explore:
    - Self-hosted build agents.
    - Build Triggers (CI, scheduled).
    - Retention Policies.
    - Containerized build agents for isolated builds.

3.2 Deployment Options

- Containerize the application.
- Push Docker images to **ACR** (Azure Container Registry).
- Pull images into **AKS** clusters securely.

3.3 External Integrations

- Integrate:
    - Azure Key Vault for secrets management.
    - Testing Frameworks (Unit Tests, Static Code Analysis, Code Coverage).
- Connect:
    - GitHub Actions to trigger releases.
    - Jenkins for hybrid CI/CD pipelines (optional).

✨ **Assignment 3: AKS Cluster Setup and Disaster Recovery**

- Create **two AKS clusters** (Active-Passive DR Model):
    - Active Cluster in Region A.
    - Passive Cluster in Region B.
- Configure:
    - Failover Simulation (manually or using Traffic Manager/Front Door).
    - Define:
        - **SLA (Service Level Agreement)**
        - **SLO (Service Level Objective)**
        - **SLI (Service Level Indicators)**

- Build Azure Pipelines to deploy **simultaneously** to both clusters.
- Implement DR failover policies and rehearse the failover procedure.

Phase 4: Monitoring, Logging, and Compliance

4.1 Azure Monitoring Setup

- Application Insights for:
    - Application logging
    - Custom metrics
    - Dependency tracking
- Azure Monitor Logs:
    - Log aggregation for AKS, App Insights, Infrastructure.
- RBAC:
    - Secure access to monitoring data.
- Crash Analytics:
    - Set up and analyze crash reports.
- Terraform:
    - Provision Monitoring resources (Dashboards, Alerts).

4.2 Monitoring SLA Compliance

- Track SLA/SLO breaches using:
    - Azure Monitor Alerts
    - Service Health Dashboard
- Implement:
    - Auto-scaling based on application load.
    - Alerting on critical issues.

Phase 5: Security and Compliance

5.1 Azure Security Services

- Azure Defender for:
    - Container Security (AKS)
    - Registry Scans (ACR)
- Network Security:

    - Minimum ports open (NSG rules).
    - Just-in-time access for Bastion Hosts.
- Compliance:
    - Azure Policy for:
        - Image scan enforcement
        - Regulatory Compliance Reporting
- Secrets Management:
    - Store all secrets in Azure Key Vault.
    - Rotate keys automatically.
- Alerts:
    - Setup right alerts when compliance or security is compromised.