**Solution Document: CI/CD Workflow for .NET Application Deployment to AKS**

**Objective**

This document describes the flow for developing a .NET application, building and pushing its container image to Artifactory, scanning the code with SonarQube, scanning the image with Trivy, and deploying it to AKS.

**Workflow Steps**

1. **Develop the Application**:

• Develop a .NET web application with coding best practices.

• Add a Dockerfile for containerization and configuration files for deployment.

2. **Build and Test**:

• Build the .NET application and execute unit tests in a CI/CD pipeline.

3. **Containerize and Push to Artifactory**:

• Build a Docker image of the application.

• Push the image to a private container registry (e.g., JFrog Artifactory or Azure Container Registry).

4. **Code Quality Scan with SonarQube**:

• Use SonarQube to scan the source code for vulnerabilities and quality issues.

5. **Image Security Scan with Trivy**:

• Use Trivy to scan the Docker image for known vulnerabilities and misconfigurations.

6. **Deploy to AKS**:

• Use Helm charts or Kubernetes manifests to deploy the application to Azure Kubernetes Service (AKS).

7. **Monitor and Manage**:

• Use Azure Monitor, Prometheus, or Grafana for application and infrastructure monitoring.

Infrastructure using Terrafrom.

├── [main.tf](http://main.tf/)

├── [variables.tf](http://variables.tf/)

├── [outputs.tf](http://outputs.tf/)

├── modules

│   ├── aks

│   │   ├── [main.tf](http://main.tf/)

│   │   ├── [variables.tf](http://variables.tf/)

│   │   ├── [outputs.tf](http://outputs.tf/)

│   ├── key\_vault

│   │   ├── [main.tf](http://main.tf/)

│   │   ├── [variables.tf](http://variables.tf/)

│   │   ├── [outputs.tf](http://outputs.tf/)

│   ├── log\_analytics

│   │   ├── [main.tf](http://main.tf/)

│   │   ├── [variables.tf](http://variables.tf/)

│   │   ├── [outputs.tf](http://outputs.tf/)

│   ├── virtual\_network

│   │   ├── [main.tf](http://main.tf/)

│   │   ├── [variables.tf](http://variables.tf/)

│   │   ├── [outputs.tf](http://outputs.tf/)

├── environments

│   ├── dev.tfvars

│   ├── prod.tfva

**Flowchart**

+-------------------------+

|   Develop Application   |

|      (.NET Codebase)    |

+-------------------------+

             |

             v

+-------------------------+

|      Build & Test       |

|   (Run Unit Tests)      |

+-------------------------+

             |

             v

+-------------------------+

|   Build Docker Image    |

|     (Containerize)      |

+-------------------------+

             |

             v

+-------------------------+

| Push Image to Registry  |

|   (e.g., Artifactory)   |

+-------------------------+

             |

             v

+-------------------------+

| Code Scan: SonarQube    |

|  (Static Code Analysis) |

+-------------------------+

             |

             v

+-------------------------+

| Image Scan: Trivy       |

|  (Check Vulnerabilities)|

+-------------------------+

             |

             v

+-------------------------+

| Deploy to AKS Cluster   |

|  (Helm or Manifests)   🡺 Gitops with ArgoCD |

+-------------------------+

             |

             v

+-------------------------+

| Monitor & Manage        |

| (Logs, Alerts, Metrics) Azure monitor/Prometheus/grafana |

+-------------------------+

Repo details:

1: Infrastructure setup using Terraform and gitactions to provision the resources in Azure: [ushabapat/Infrasetup](https://github.com/ushabapat/Infrasetup)

2. NET application build , code scan , image scan and push the image. I have commented out few tasks due to unavailability of the target resource.

The deployment to AKS will be triggered by calling the below mentioned repo, where it will update the image tag and application deployed via helm chart using ARGOCD : [ushabapat/buildnaddeploy](https://github.com/ushabapat/buildnaddeploy)

3. Sample helm based application deployment using argocd : [ushabapat/IAAC at master](https://github.com/ushabapat/IAAC/tree/master)

NOTE: The secret or credentials needs to be updated else it wont work.

**Tools Used**

1. **Development**:

• Language: .NET Core/.NET Framework

2. **Build and Test**:

• CI/CD Platform: GitHub Actions

3. **Containerization**:

• Docker

4. **Registry**:

• JFrog Artifactory, Azure Container Registry (ACR), or Docker Hub – It cant be configured based on the project requirement or standard

5. **Code Scanning**:

• SonarQube

6. **Image Scanning**:

• Trivy

7. **Deployment**:

• AKS with Helm or Kubernetes manifests

8. **Monitoring**:

• Azure Monitor, Prometheus, Grafana

**Notes**

• Automate the entire pipeline to ensure faster delivery and consistency.

• Secure the CI/CD process by applying access controls to Artifactory and SonarQube.

• Incorporate rollback mechanisms in the deployment strategy for safer releases.

Let me know if you’d like a specific focus on any section!