# DevOps Project: Comprehensive CI/CD Implementation with Monitoring

## Project Overview

This project will guide students through implementing a complete CI/CD pipeline, integrating tools like Azure DevOps, SonarQube, Docker, and Ansible, while setting up monitoring with Prometheus and Grafana. Students will also implement alerts for resource usage.

## Project Objectives

1. Azure Boards: Create and manage work items and enforce commit policies related to work items.

2. Azure Repos: Create repositories, define branching strategies, and integrate work items.

3. CI Pipeline:

- Build the source code.

- Integrate unit tests and static code analysis (SonarQube).

- Produce binary packages and Docker images.

- Scan Docker images for vulnerabilities.

- Automate email notifications for SonarQube and image hardening reports.

4. CD Pipeline:

- Deploy the application to the IIS server using **Azure DevOps** and **Ansible PLAYBOOKS**.

5. Monitoring:

- Install and configure Prometheus and Grafana.

- Monitor disk and CPU usage with alerts.

6. Version Control: Maintain all scripts and configurations in an Azure DevOps repository.

7. Terraform: Create an Ubuntu LTS VM on VirtualBox to serve as the Ansible server.

## 1. Azure Boards

1. Create a Work Item:

- Define a user story to track the CI/CD pipeline development.

Example: 'As a developer, I want a CI/CD pipeline to automate building, testing, and deploying my application.'

2. Set Policies:

- Ensure that no commits are allowed without linking to a work item.

- Configure branch policies in Azure Repos to enforce this rule.

## 2. Azure Repos

1. Create Repositories:

- Set up a repository for the project.

- Push the source code of a chosen application with separate front-end and back-end parts.

2. Branch Strategies:

- Discuss and implement a branching model (e.g., Gitflow, trunk-based, or GitHub Flow).

- Set policies for pull requests and merges.

## 3. CI Pipeline

1. Pipeline Setup:

- Use an Azure DevOps YAML file for the pipeline configuration.

- Define tasks to build and test the application.

2. Unit Tests Integration:

- Include unit test execution in the pipeline.

3. Static Code Analysis:

- Integrate SonarQube for SAST.

- Generate a quality gate report.

4. Build Outputs:

- Binary Package: Push to Azure Artifacts with semantic versioning.

- Docker Image: Build and push to a container registry (Docker Hub, Harbor, or Nexus).

- Scan Docker Images: Use security tools to harden images.

5. Email Notification:

- Automate sending reports for SonarQube analysis and image hardening via email.

## 4. CD Pipeline

1. IIS Deployment:

- Deploy the application on an IIS server using Azure DevOps.

- Automate the configuration of the application pool, feature activation, and website creation.

2. Ansible Deployment:

- Create an Ansible playbook to deploy the application.

- Include prerequisites installation (e.g., Java, .NET, Python).

- Fetch binary packages from Azure Artifacts and deploy to IIS.

## 5. Monitoring

1. Setup Prometheus and Grafana:

- Install Prometheus and Grafana on a monitoring server.

- Monitor IIS, your windows vm, and Ansible servers.

2. Create Alerts:

- Disk Usage Alert: Trigger an alert when disk usage exceeds 80%.

- CPU Usage Alert: Trigger an alert when CPU usage exceeds 60%.

3. Integrate Alerts:

- Configure Prometheus rules and Grafana notifications (e.g., email or Slack).

## 6. Version Control

1. Store Scripts:

- Maintain all scripts (YAML, Ansible playbook, monitoring configurations) in an Azure DevOps repository.

- Ensure proper documentation and tagging for better traceability.

## 7. Terraform: Create Ansible Server

1. Objective:

- Use Terraform to provision an Ubuntu LTS virtual machine on VirtualBox. The hardware specifications of the VM will be dynamically adjusted based on the capabilities of your laptop.

2. Steps:

- Install Terraform and VirtualBox on your local machine.

- Write a Terraform configuration file to define the Ubuntu LTS VM.

- Include variables for CPU, memory, and disk size to make the configuration adaptable.

- Apply the Terraform configuration to create the VM.

- Verify the VM creation and ensure SSH access is enabled.

3. Ansible Server Setup:

- Configure the VM as the Ansible control node.

- Install necessary software (e.g., Ansible, Python).

- Test Ansible connectivity to ensure the VM is ready for deployment tasks.

## Deliverables

1. Azure Boards with properly linked work items.

2. Configured Azure Repos with branch strategies and source code.

3. CI pipeline YAML file:

- Includes unit tests, SonarQube, binary and Docker image outputs, and email notifications.

4. CD pipeline YAML file:

- Automates deployment to IIS using Azure DevOps and Ansible.

5. Ansible playbook for IIS server setup.

6. Monitoring setup with Prometheus and Grafana:

- Dashboard showcasing server metrics.

- Alerting rules for disk and CPU usage.

7. Documentation for all steps.