**Capstone Project (4 days)**

**Title:** CI/CD Pipeline for Containerized Application Deployment Using Azure DevOps with Branch Management

**Objective**: Build a CI/CD pipeline that manages code through Azure Repos, allowing developers to push code, create Docker images, and deploy to Azure Kubernetes Service (AKS) using Azure DevOps and Azure Container Registry (ACR).

Table of Contents

[Day 1: Setup and Preparation 3](#_Toc182747823)

[Environment Setup: 3](#_Toc182747824)

[Create Azure Resources: 3](#_Toc182747825)

[Application Selection: 3](#_Toc182747826)

[Azure Organization and Billing Configuration: 3](#_Toc182747827)

[Azure Organization Setup 3](#_Toc182747828)

[Billing Setup 3](#_Toc182747829)

[User Creation for Development and Testing Teams 4](#_Toc182747830)

[User Permissions and Access 4](#_Toc182747831)

[Git Repository Setup and Branch Management: 4](#_Toc182747832)

[Creating Local Repository in Visual Studio Code 4](#_Toc182747833)

[Branch Creation and Workflow 4](#_Toc182747834)

[Developer Workflow 4](#_Toc182747835)

[Code Review and Merging Process 4](#_Toc182747836)

[Pull Request and Conflict Resolution 4](#_Toc182747837)

[Git Merge Workflow in Visual Studio Code 4](#_Toc182747838)

[Branch Management Setup: 5](#_Toc182747839)

[Day 2: Code Development and Dockerization 5](#_Toc182747840)

[Developing Code in Visual Studio Code: 5](#_Toc182747841)

[Azure Boards Setup for Capstone Project: 5](#_Toc182747842)

[1. EPIC: Set Up Development Environment and Team 5](#_Toc182747843)

[2. EPIC: Set Up Git Repository and Version Control Workflow 5](#_Toc182747844)

[3. EPIC: Setup Continuous Integration and Automated Pipelines 6](#_Toc182747845)

[4. EPIC: Bug Tracking and Issue Management 7](#_Toc182747846)

[Shared Queries for Work Items: 7](#_Toc182747847)

[Push Code to Azure Repos: 8](#_Toc182747848)

[Code Review Process: 8](#_Toc182747849)

[Dockerize the Application: 9](#_Toc182747850)

[Push to Azure Container Registry: 9](#_Toc182747851)

[Day 3: Setting Up Azure DevOps Pipeline 9](#_Toc182747852)

[Create Azure DevOps Pipeline: 9](#_Toc182747853)

[Pipeline Configuration: 9](#_Toc182747854)

[Testing the Pipeline: 9](#_Toc182747855)

[Day 4: Monitoring and Cleanup 9](#_Toc182747856)

[Monitor the Deployment: 9](#_Toc182747857)

[Documentation: 10](#_Toc182747858)

[Cleanup: 10](#_Toc182747859)

[Presentation 10](#_Toc182747860)

# Day 1: Setup and Preparation

## Environment Setup:

* + **Linux/Windows:** Set up your local environment on Windows or a Linux VM.
  + **Install Visual Studio Code/Visual Studio 2022:** Ensure you have the necessary extensions for Azure, Docker, and Git.

## Create Azure Resources:

* + Set up Azure Container Registry (ACR).
  + Create an Azure Kubernetes Service (AKS) cluster.
  + Configure Azure CLI for resource management.

## Application Selection:

* + Choose a sample application MVC donetcore
  + create a new one in Azure Repos.

# Azure Organization and Billing Configuration:

## Azure Organization Setup

* + How would you create an Azure Organization with the name yourname\_Ascendion using the Azure DevOps portal?
  + What process template would you select for your organization, and why would you choose the "Agile" process for this setup?

## Billing Setup

* + How would you configure billing for your Azure DevOps organization to use **1 parallel job** with the currently active subscription? Explain the steps involved in setting this up.

## User Creation for Development and Testing Teams

* + What are the steps to create two users, **Dev1** and **Dev2**, and assign them to a **Development** team in Azure DevOps?
  + How would you create two users, **Test1** and **Test2**, and assign them to a **Testing** team in Azure DevOps?

## User Permissions and Access

* + What are the best practices for setting up user roles and permissions for **Dev1**, **Dev2**, **Test1**, and **Test2** in Azure DevOps to ensure proper access control for development and testing activities?

# Git Repository Setup and Branch Management:

## Creating Local Repository in Visual Studio Code

* + How would you create a new local Git repository for an MVC project using **Visual Studio Code**? Describe the steps from initializing the repository to creating the first commit.

## Branch Creation and Workflow

* + How do you create three branches in your local Git repository: dev1, dev2, and main? Describe the steps to create, switch between, and manage branches in Git using Visual Studio Code.

## Developer Workflow

* + Explain the workflow where **Dev1** and **Dev2** will each submit code to their respective branches (dev1 and dev2). How would they perform commits and push changes to the remote repository?

## Code Review and Merging Process

* + After **Dev1** and **Dev2** have submitted their changes to their respective branches, what steps would be involved in conducting a **code review**? Describe how to merge their changes into the main branch after the code review process.

## Pull Request and Conflict Resolution

* + What is the process for creating a **Pull Request (PR)** to merge code from dev1 and dev2 into main? How would you handle any **merge conflicts** that might arise during the PR process?

## Git Merge Workflow in Visual Studio Code

* + After the code review, describe the steps to merge the changes from the dev1 and dev2 branches into the main branch using **Visual Studio Code** and Git commands. What commands would you use to merge and push the final changes?

## Branch Management Setup:

* + Create branches in Azure Repos through local git repository:
    - **Dev1 Branch:** For ongoing development by Dev1.
    - **Dev2 Branch:** For ongoing development by Dev2.
    - **Main Branch:** For stable code, containing the latest release.

# Day 2: Code Development and Dockerization

## Developing Code in Visual Studio Code:

* + Developer works on the **Dev** branch using Visual Studio Code.
  + Implement features or fixes, committing changes locally.

## Azure Boards Setup for Capstone Project:

### 1. EPIC: Set Up Development Environment and Team

* **Description**: This EPIC covers setting up the development environment, configuring Azure DevOps for your team, and setting up users and permissions.
* **User Stories**:
  1. **Create an Azure Organization**:
     + **Task**: Set up the organization named yourname\_Ascendion in Azure DevOps.
  2. **Configure Billing and Parallel Jobs**:
     + **Task**: Set up billing with 1 parallel job using the active subscription.
  3. **Create Users for Development and Testing Teams**:
     + **Task**: Add Dev1, Dev2 users to the Development team and Test1, Test2 users to the Testing team.
  4. **Assign Roles and Permissions**:
     + **Task**: Assign appropriate roles to Dev1, Dev2, Test1, and Test2 for proper access control.

### 2. EPIC: Set Up Git Repository and Version Control Workflow

* **Description**: This EPIC covers setting up the Git repository, branching strategy, and integrating with Visual Studio Code.
* **User Stories**:
  1. **Initialize Git Repository**:
     + **Task**: Initialize a Git repository for the MVC project using Visual Studio Code.
  2. **Create and Manage Branches**:
     + **Task**: Create three branches: dev1, dev2, and main in the Git repository.
  3. **Implement Developer Workflow**:
     + **Task**: Dev1 and Dev2 commit changes to their respective branches.
  4. **Code Review Process**:
     + **Task**: Review code for Dev1 and Dev2 before merging into the main branch.
  5. **Merge Changes into Main Branch**:
     + **Task**: Merge Dev1 and Dev2 branches into the main branch after code review.
  6. **Handle Merge Conflicts**:
     + **Task**: Resolve any conflicts during the merge process.

### 3. EPIC: Setup Continuous Integration and Automated Pipelines

* **Description**: This EPIC involves setting up automated CI/CD pipelines to integrate with the Azure DevOps workflow.
* **User Stories**:
  1. **Setup CI Pipeline**:
     + **Task**: Create a Continuous Integration pipeline that triggers on commits to the main branch.
  2. **Setup Build and Test Automation**:
     + **Task**: Set up automated build and test steps in the pipeline for main branch merges.
  3. **Setup Parallel Jobs**:
     + **Task**: Configure the pipeline to run with 1 parallel job as per the billing setup.
  4. **Automated Deployment**:
     + **Task**: Integrate automated deployment to staging or production once code is merged into the main branch.

### 4. EPIC: Bug Tracking and Issue Management

* **Description**: This EPIC is dedicated to tracking bugs and managing issues for the Capstone project.
* **User Stories**:
  1. **Bug Reporting and Tracking**:
     + **Task**: Set up a process for creating and tracking bugs in Azure DevOps.
  2. **Bug Fix Workflow**:
     + **Task**: Assign bugs to Test1/Test2 and Dev1/Dev2, fix them, and verify resolutions.
  3. **Track Open Issues**:
     + **Task**: Create a process for monitoring open issues throughout the sprint.
  4. **Close Issues and Mark Resolved**:
     + **Task**: Close issues after testing and review the resolution process.

### Shared Queries for Work Items:

**Shared Queries** are an essential part of Azure DevOps for tracking and managing work items across your team. Below are some common queries you might create to track various aspects of your project.

**Shared Queries:**

1. **Query to Track Open Issues**
   * **Query Name**: **Open Issues**
   * **Query Type**: **Work Items and Direct Links**
   * **Conditions**:
     + **State** = "Active"
     + **Work Item Type** = "Bug" or "Issue"
   * **Purpose**: To show all issues or bugs that are still open and being worked on.
2. **Query to Track Closed Issues**
   * **Query Name**: **Closed Issues**
   * **Query Type**: **Work Items and Direct Links**
   * **Conditions**:
     + **State** = "Closed"
     + **Work Item Type** = "Bug" or "Issue"
   * **Purpose**: To display all issues or bugs that have been resolved and closed.
3. **Query to Track Active Bugs**
   * **Query Name**: **Active Bugs**
   * **Query Type**: **Work Items and Direct Links**
   * **Conditions**:
     + **State** = "Active"
     + **Work Item Type** = "Bug"
   * **Purpose**: To show all currently active bugs.
4. **Query to Track Assigned Bugs**
   * **Query Name**: **Bugs Assigned to Developers**
   * **Query Type**: **Work Items and Direct Links**
   * **Conditions**:
     + **State** = "Active"
     + **Work Item Type** = "Bug"
     + **Assigned To** = [Dev1, Dev2, Test1, Test2] (for specific team members)
   * **Purpose**: To show all bugs currently assigned to the developers or testers.
5. **Query to Track Recently Closed Work Items**
   * **Query Name**: **Recently Closed Work Items**
   * **Query Type**: **Work Items and Direct Links**
   * **Conditions**:
     + **State** = "Closed"
     + **Changed Date** = "Last 7 Days"
   * **Purpose**: To see the work items that have been recently closed within the last week.

## Push Code to Azure Repos:

* + Push changes to the **Dev** branch in Azure Repos.
  + Create pull requests (PRs) for code review when ready to merge changes into **Main**.

## Code Review Process:

* + Code owner reviews the PR.
  + After approval, merge changes from **Dev** to **main**.

## Dockerize the Application:

* + Create a Dockerfile for the application.
  + Build the Docker image locally (if needed for testing).

## Push to Azure Container Registry:

* + Log in to ACR using Azure CLI.
  + Tag the Docker image for ACR.
  + Push the image to ACR after a successful build.

# Day 3: Setting Up Azure DevOps Pipeline

## Create Azure DevOps Pipeline:

* + Set up a CI/CD pipeline in Azure DevOps.
  + Configure triggers to automatically build the Docker image when changes are pushed to the **Master** branch.

## Pipeline Configuration:

* + Steps in the pipeline should include:
    - Building the Docker image.
    - Pushing the image to ACR.
    - Deploying the application to AKS using kubectl commands.

## Testing the Pipeline:

* + Run the pipeline after merging to **Master** to ensure that the new Docker image is built and deployed without errors.

# Day 4: Monitoring and Cleanup

## Monitor the Deployment:

* + Access the AKS cluster to ensure the application is running smoothly.
  + Use Azure Monitor or Kubernetes dashboard for monitoring performance.

## Documentation:

* + Document the process, including:
    - Branching strategy.
    - Pipeline configurations and triggers.
    - Code review procedures.

## Cleanup:

* + Delete the resources created during the project (AKS, ACR, VMs, etc.) to avoid unnecessary costs.

# Presentation

* Prepare a presentation that showcases the entire workflow, including:
  + Branch management strategy.
  + Code review process.
  + CI/CD pipeline automation.