Assignment 3

Part 1

Step 1: Set up Jenkins Environment

Install Jenkins and necessary plugins: Make sure Jenkins is installed on your server. Install the following plugins:

Docker Plugin: This plugin is required for Docker integration.

MSBuild Plugin: If you're building a .NET Core app, you might need this plugin for building the project.

Step 2: Configure Docker

Ensure Docker is installed on the Jenkins server and Jenkins user has the necessary permissions to run Docker commands.

Verify that the Docker daemon is running.

Step 3: Create a Freestyle Jenkins Job

Log in to the Jenkins web interface.

Click on "New Item" to create a new Jenkins job.

Enter a name for your job (e.g., "BuildAndDeployDotNetApp").

Select "Freestyle project" as the job type, and click "OK."

A screenshot of a computer

Description automatically generated

Step 4: Configure Source Code Management

Under the "Source Code Management" section, select your version control system (e.g., Git).

Enter the repository URL and credentials if needed.

A screenshot of a computer

Description automatically generated

Step 5: Build

Under the "Build" section, add build steps as needed for your .NET Core project. Typically, this involves running dotnet build to build the project.

You may also need to restore dependencies using dotnet restore.

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step 6: Build and Push Docker Image**

1. Add a build step to create a Docker image for your .NET Core app. Use the Dockerfile for your app.

docker build -t your-image-name:tag .

2. Add another build step to push the Docker image to a Docker registry (e.g., Docker Hub or a private registry).

docker push your-registry/your-image-name:tag

**Step 7: Deployment**

1. Depending on your deployment target, you can use tools like Docker Compose, Kubernetes, or custom shell scripts to deploy your Docker container.

**Step 8: Post-Build Actions**

1. Under the "Post-build Actions" section, you can configure actions such as sending email notifications or triggering other Jenkins jobs upon successful deployment.

**Step 9: Save and Run the Jenkins Job**

1. Click "Save" to save your Jenkins job configuration.
2. Manually run the job to test the pipeline.

**Step 10: Monitor and Improve**

1. Monitor the Jenkins job and pipeline for any issues or errors.
2. Continuously improve your pipeline by adding more stages or automation as needed.