Task: Setting up Jenkins for Continuous Integration/Continuous Deployment (CI/CD)

Description:

Configure Jenkins to automate the build, test, and deployment processes for your application.

Jenkins Installation and Configuration:

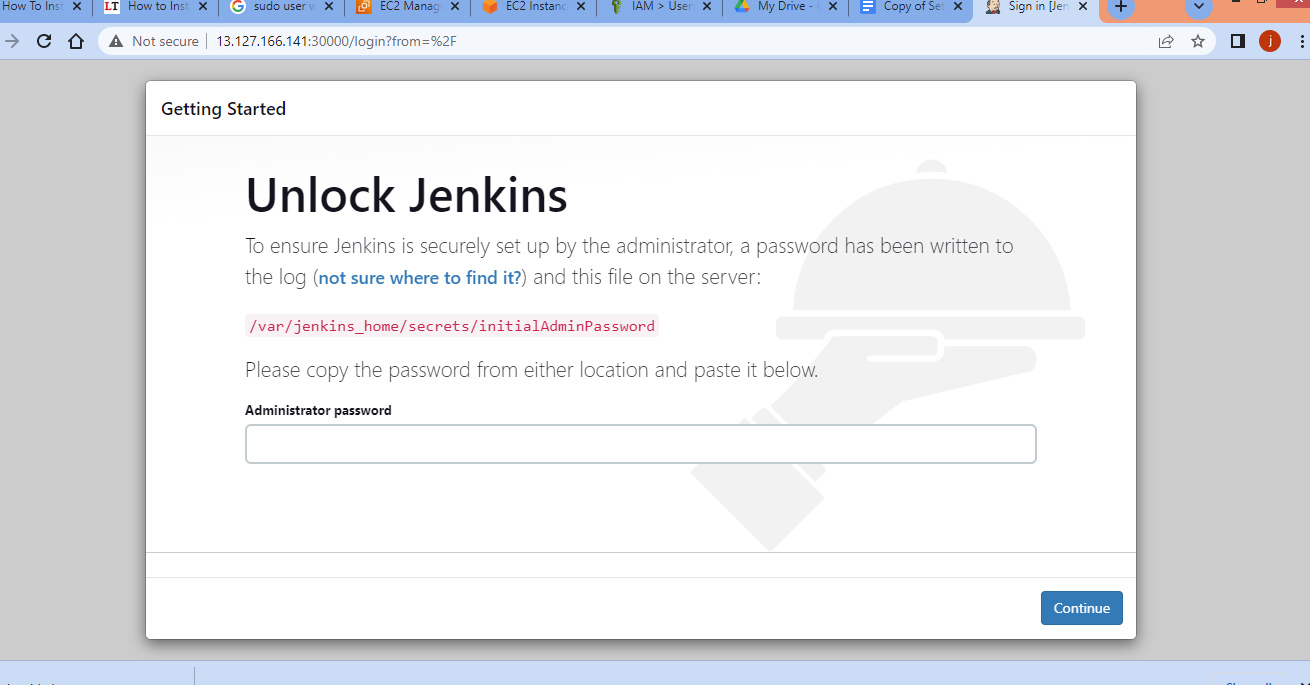
Test Case 1:

Verify that Jenkins is successfully installed on the designated server.

Expected Outcome: Jenkins should be installed without errors, and the web interface should be

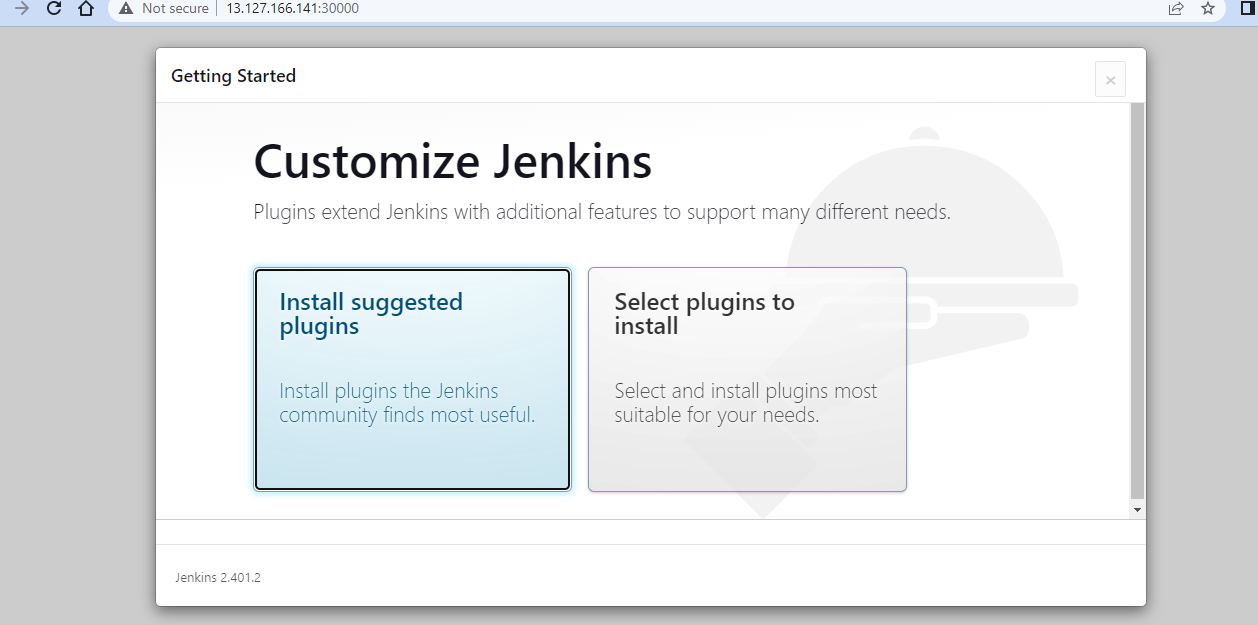
accessible.

Verify that Jenkins is successfully installed on the designated server.

* Install Jenkins on the designated server. You can follow the official Jenkins installation guide for your operating system:
* # This script installs Jenkins on an Ubuntu server
* # Update package lists
* sudo apt-get update
* # Install Java 11
* sudo apt-get install -y openjdk-11-jdk
* # Download Jenkins key and add it to system
* curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
* # Add Jenkins to system package source list
* sudo sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null'
* # Update package lists again to include Jenkins and install it
* sudo apt-get update
* sudo apt-get install -y jenkins
* # Verify Jenkins is installed and working
* sudo systemctl status jenkins
* Output:
* 
* Once installed, ensure that Jenkins starts without errors.
* Access the Jenkins web interface using a web browser and navigate to http://your\_server\_ip:8080 (replace your\_server\_ip with the actual IP address or hostname).
* Follow the on-screen instructions to complete the Jenkins setup, including installing recommended plugins.
* Log in to Jenkins with the credentials created during the setup.

Navigate to /var/lib/jenkins/secrets/initialAdminPassword and copy the password and unlock Jenkins.

Then Install the suggested plugins:



Test Case 2:

Trigger a sample commit in the version control system and ensure that the CI pipeline is automatically executed.

* Set up a version control system (e.g., Git) for your project and push a sample commit.
* Create a new Jenkins pipeline job:
  + Click on "New Item" on the Jenkins dashboard.
  + Choose "Pipeline" and give it a name (e.g., "MyCIJob").
  + In the pipeline configuration, select "Pipeline script from SCM" and configure the repository URL.
* Configure the SCM Polling:
  + In the pipeline configuration, under "Build Triggers," select "Poll SCM" and set a schedule (e.g., \* \* \* \* \* for polling every minute).
* Save the job configuration.
* Make a change in your version control system and commit it.
* Jenkins should automatically detect the change and trigger the CI pipeline.
* CD Pipeline and Deployment

Test Case 3:

Manually trigger the CD pipeline and confirm that the application is deployed to a staging environment.

* 1.Create a new Jenkins pipeline job for your CD pipeline:
  + Follow the steps similar to creating the CI job, but this time set up the deployment stages in the pipeline script.
* 2.Add a manual trigger for the CD pipeline:
  + In the pipeline script, use an input step to pause the pipeline and wait for manual approval:

stage('Deploy to Staging') {

steps {

script {

// Deployment steps

}

}

post {

failure {

echo "Deployment to staging failed!"

}

success {

input "Deploy to Staging?"

}

}

}

3.Save the job configuration.

4.Run the CD pipeline manually:

* + After running the CI pipeline, go to the Jenkins dashboard and click on the CD job.
  + Click on "Build Now" or manually trigger the pipeline.

5.Confirm the deployment:

* + Monitor the Jenkins console output to ensure that the deployment steps are executed without errors.
  + Access the staging environment and verify that the latest changes have been deployed successfully.

These steps provide a basic setup for CI/CD with Jenkins. Depending on your project and technology stack, you may need to customize the pipeline script and configurations. s