

Experiment 2: Creating a Jenkins Pipeline with a Jenkinsfile

Objective: Create a Jenkins pipeline using a Jenkinsfile that builds a simple project, runs tests, and deploys the project to a designated Environment.

Prerequisites:

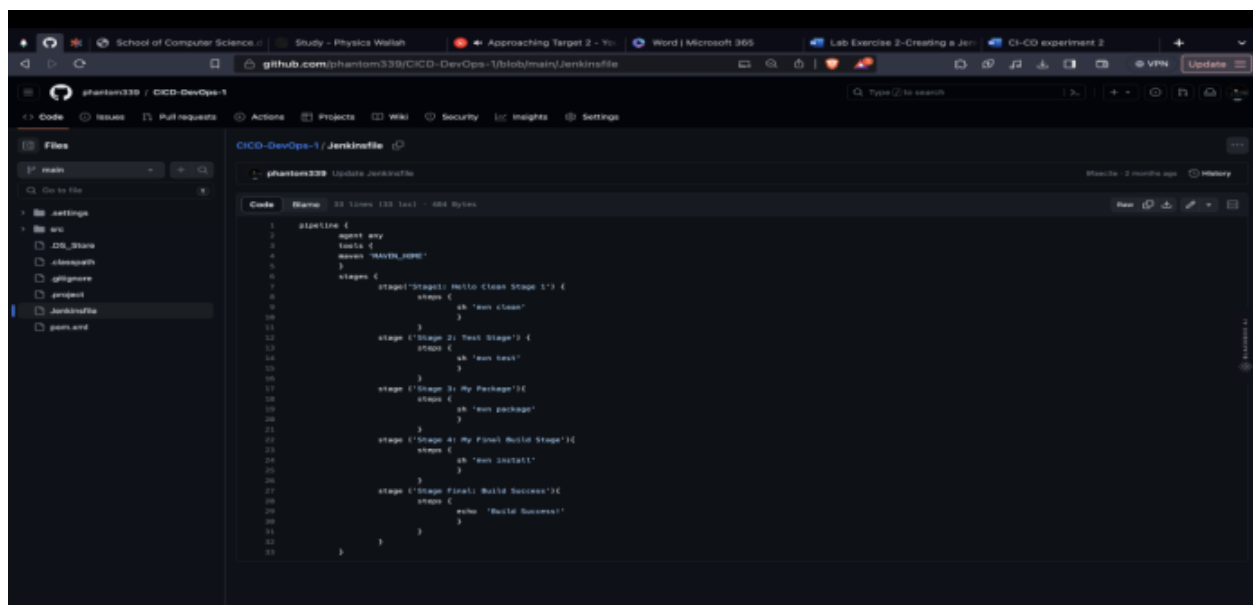
1. Jenkins server up and running.
2. A sample project hosted in a version control repository (e.g., Git).

Steps:

Jenkins Configuration:

- Ensure that Jenkins is installed and accessible.
- Install necessary plugins: Pipeline and any plugins specific to your version control system (e.g., Git Plugin).
- Setting Up the Project:
Create a sample project (e.g., a simple web application) and host it on a version control repository (e.g., GitHub).
- Creating a Jenkinsfile: In the root of your project repository, create a file named Jenkinsfile.
- Defining the Pipeline:
Open the Jenkinsfile and define the pipeline stages using the declarative pipeline syntax.

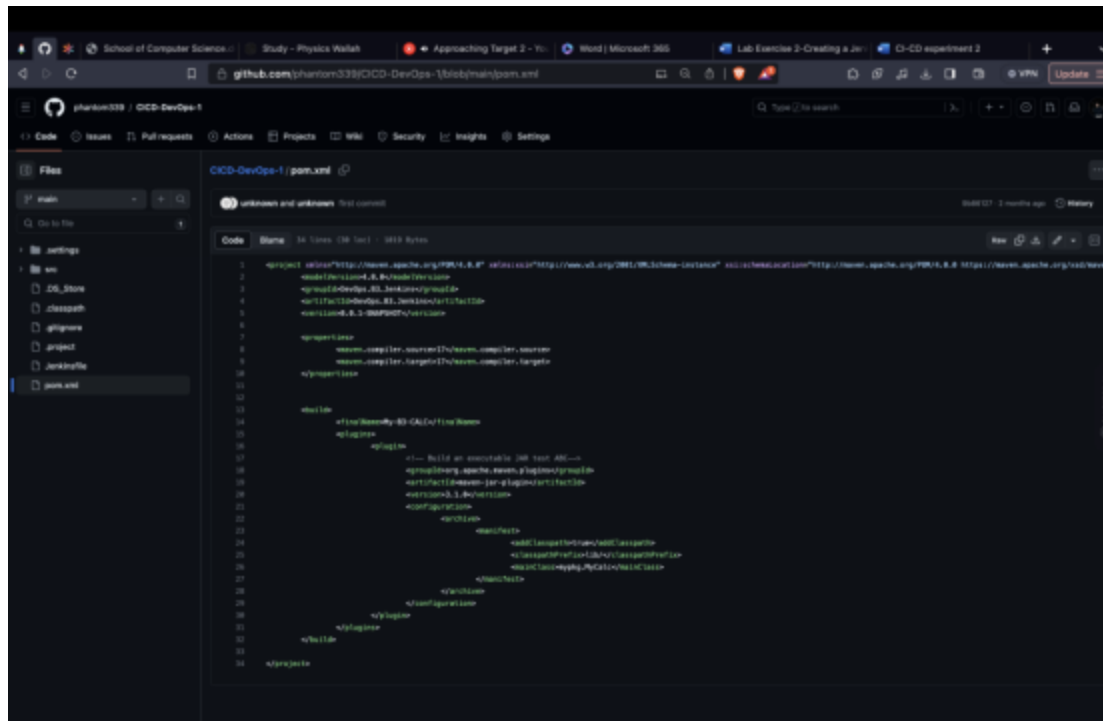
Here's an example Jenkinsfile with basic stages:



The screenshot shows a web browser displaying a GitHub repository named 'phantom339 / CI-CD-DevOps-1'. The 'Jenkinsfile' is open in the code editor. The pipeline is defined with the following stages:

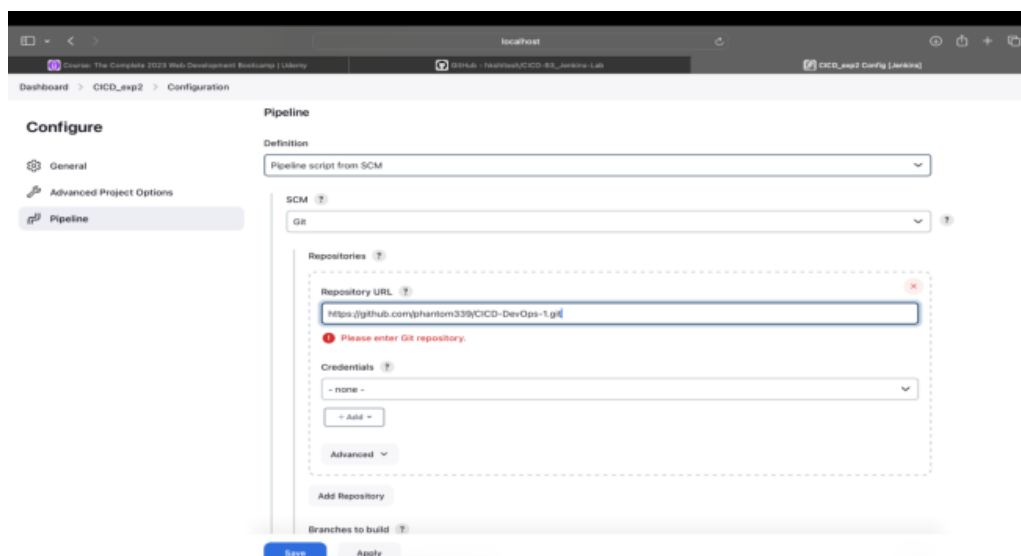
```
1 pipeline {
2   agent any
3   tools {
4     jdk 'JDK_11'
5   }
6   stages {
7     stage('Stage 1: Hello Clean Stage 1') {
8       steps {
9         sh 'echo "Hello Clean Stage 1"'
10      }
11    }
12    stage('Stage 2: Test Stage') {
13      steps {
14        sh 'echo "Test Stage"'
15      }
16    }
17    stage('Stage 3: My Package') {
18      steps {
19        sh 'echo "My Package"'
20      }
21    }
22    stage('Stage 4: My Final Build Stage') {
23      steps {
24        sh 'echo "Final Build"'
25      }
26    }
27    stage('Stage Final: Build Success') {
28      steps {
29        echo 'Build Success!'
30      }
31    }
32  }
33 }
```

```
pipeline {
  agent any
  stages {
    stage('Checkout') {
      steps {
        checkout scm
      }
    }
    stage('Build') {
      steps {
        sh 'your-build-command-here'
      }
    }
    stage('Test') {
      steps {
        sh 'your-test-command-here'
      }
    }
    stage('Deploy') {
      steps {
        sh 'your-deployment-command-here'
      }
    }
  }
  post {
    success {
      echo 'Pipeline succeeded! Project built and deployed.'
    }
    failure {
      echo 'Pipeline failed! Check logs for details.'
    }
  }
}
```



Configuring the Pipeline in Jenkins:

- In Jenkins, create a new pipeline job.
- Link the job to your version control repository (e.g., provide the repository URL).
- Choose the option to use a Jenkinsfile from the repository and specify the path to your Jenkinsfile (usually the root directory).



Running the Pipeline:

- Trigger the pipeline manually or set up a webhook to trigger it automatically on repository changes.

Observing the Results:

- Observe the pipeline execution on the Jenkins dashboard.
- Check the console output of each stage for any errors or issues.

This lab experiment will give you hands-on experience in creating a Jenkins pipeline using a Jenkinsfile. You can extend this experiment by adding more stages, integrating with other tools, and handling more complex build and deployment scenarios.

Stage View

	Declarative: Checkout SCM	Declarative: Tool Install	Stage1: Hello Clean Stage 1	Stage 2: Test Stage	Stage 3: My Package	Stage 4: My Final Build Stage	Stage Final: Build Success
Average stage times: (Average full run time: ~11s)	1s	77ms	1s	1s	1s	1s	136ms
Nov 16 23:47 1 commit	1s	92ms	1s	1s	1s	1s	137ms
Sep 19 09:26 No Changes	1s	63ms	1s	1s	1s	1s	136ms

Build History

Filter builds...

- #2 17-Nov-2023 3:17 am TLT
- #1 19-Sep-2023 12:54 am TLT

Atom feed for all Atom feed for failures

Permalinks

- Last build (#2), 7.9 sec ago
- Last stable build (#1), 1 mo 28 days ago
- Last successful build (#1), 1 mo 28 days ago
- Last completed build (#1), 1 mo 28 days ago