Lab 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:

```
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:

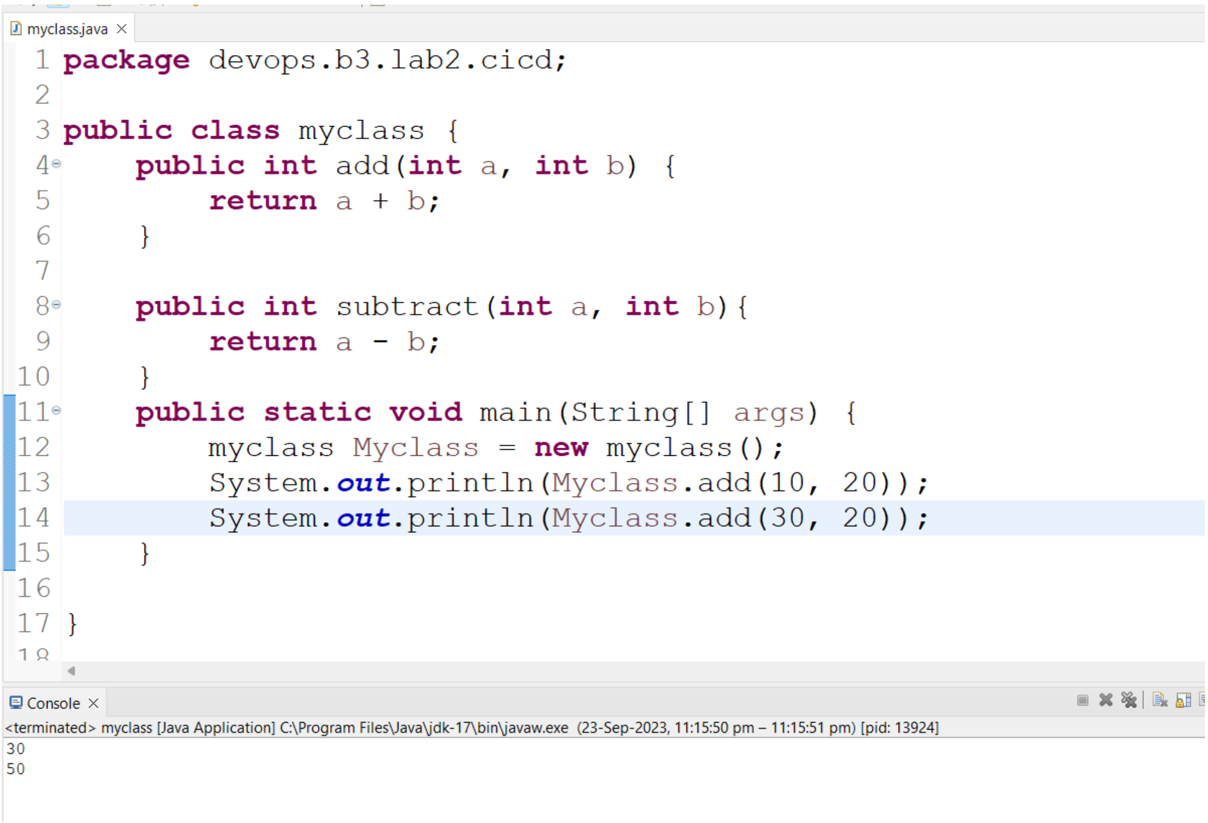
- Trigger the pipeline manually or set up a webhook to trigger it automatically on repository changes.

Observing the Results:

- Observe the pipeline execution on the Jenkins dashboard.

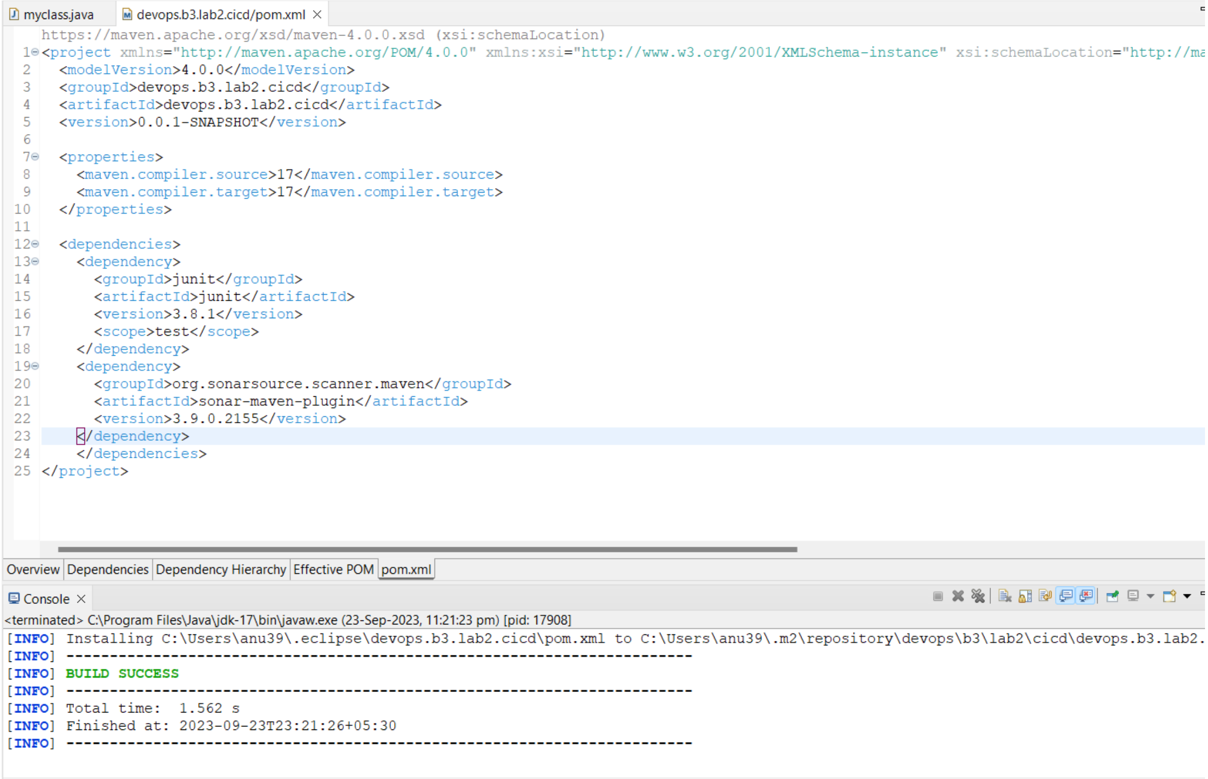
- Check the console output of each stage for any errors or issues.

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.



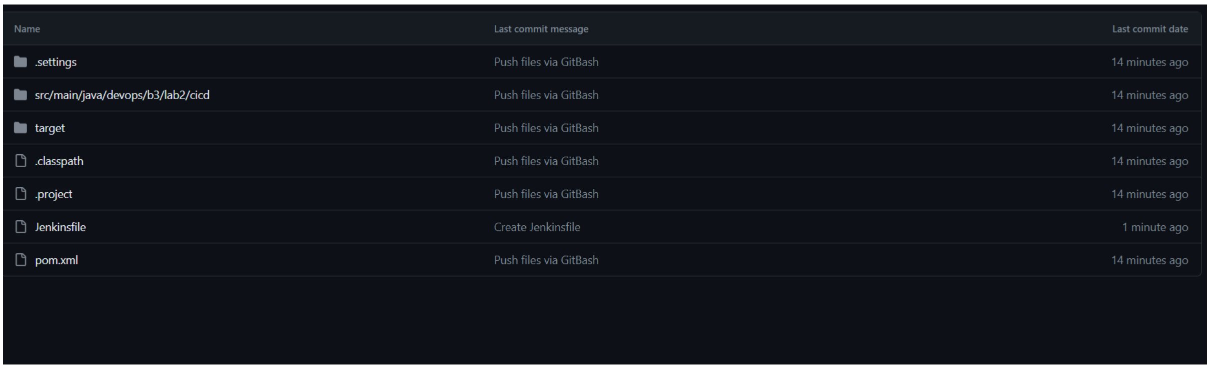
```java
package devops.b3.lab2.cicd;

public class myclass {
    public int add(int a, int b) {
        return a + b;
    }

    public int subtract(int a, int b){
        return a - b;
    }
    public static void main(String[] args) {
        myclass Myclass = new myclass();
        System.out.println(Myclass.add(10, 20));
        System.out.println(Myclass.add(30, 20));
    }

}
```

```
<terminated> myclass [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe  (23-Sep-2023, 11:15:50 pm – 11:15:51 pm) [pid: 13924]
30
50
```

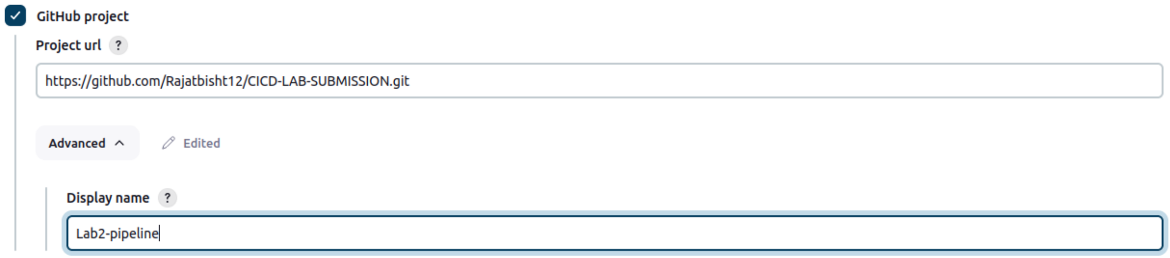


```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://ma
  <modelVersion>4.0.0</modelVersion>
  <groupId>devops.b3.lab2.cicd</groupId>
  <artifactId>devops.b3.lab2.cicd</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.sonarsource.scanner.maven</groupId>
      <artifactId>sonar-maven-plugin</artifactId>
      <version>3.9.0.2155</version>
    </dependency>
    </dependencies>
</project>
```

```
<terminated> C:\Program Files\Java\jdk-17\bin\javaw.exe (23-Sep-2023, 11:21:23 pm) [pid: 17908]
[INFO] Installing C:\Users\anu39\.eclipse\devops.b3.lab2.cicd\pom.xml to C:\Users\anu39\.m2\repository\devops\b3\lab2\cicd\devops.b3.lab2.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.562 s
[INFO] Finished at: 2023-09-23T23:21:26+05:30
[INFO] ------------------------------------------------------------------------
```



Name	Last commit message	Last commit date
.settings	Push files via GitBash	14 minutes ago
src/main/java/devops/b3/lab2/cicd	Push files via GitBash	14 minutes ago
target	Push files via GitBash	14 minutes ago
.classpath	Push files via GitBash	14 minutes ago
.project	Push files via GitBash	14 minutes ago
Jenkinsfile	Create Jenkinsfile	1 minute ago
pom.xml	Push files via GitBash	14 minutes ago



☑ GitHub project

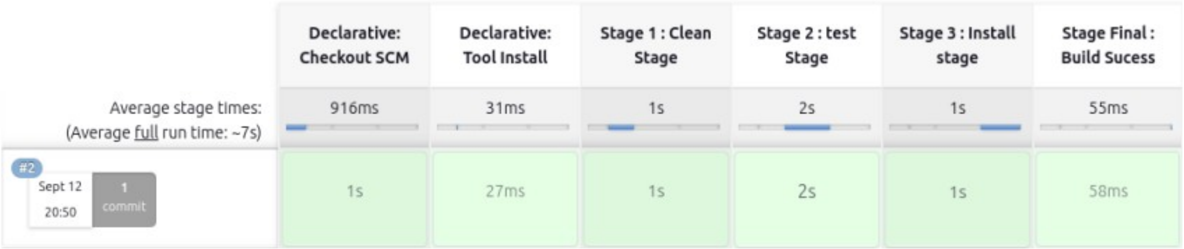
Project url ?

https://github.com/Rajatbisht12/CICD-LAB-SUBMISSION.git

Display name ?

Lab2-pipeline

5. An automatic build is triggered



	Declarative: Checkout SCM	Declarative: Tool Install	Stage 1 : Clean Stage	Stage 2 : test Stage	Stage 3 : Install stage	Stage Final : Build Sucess
Average stage times: (Average full run time: ~7s)	916ms	31ms	1s	2s	1s	55ms
#2 Sept 12 20:50 1 commit	1s	27ms	1s	2s	1s	58ms