

Name of Student: Pushkar Sane		
Roll Number: 45		Lab Assignment Number: 5
Title of Lab Assignment: To implement Jenkins pipeline using scripted/ declarative pipeline.		
DOP: 21-02-2024		DOS: 23-02-2024
CO Mapped: CO3	PO Mapped: PO2, PO3, PO5, PSO1, PSO2	Signature:

Practical No. 5

Aim: To implement Jenkins pipeline using scripted/ declarative pipeline.

Steps to implement Jenkins pipeline using declarative pipeline:

Introduction:

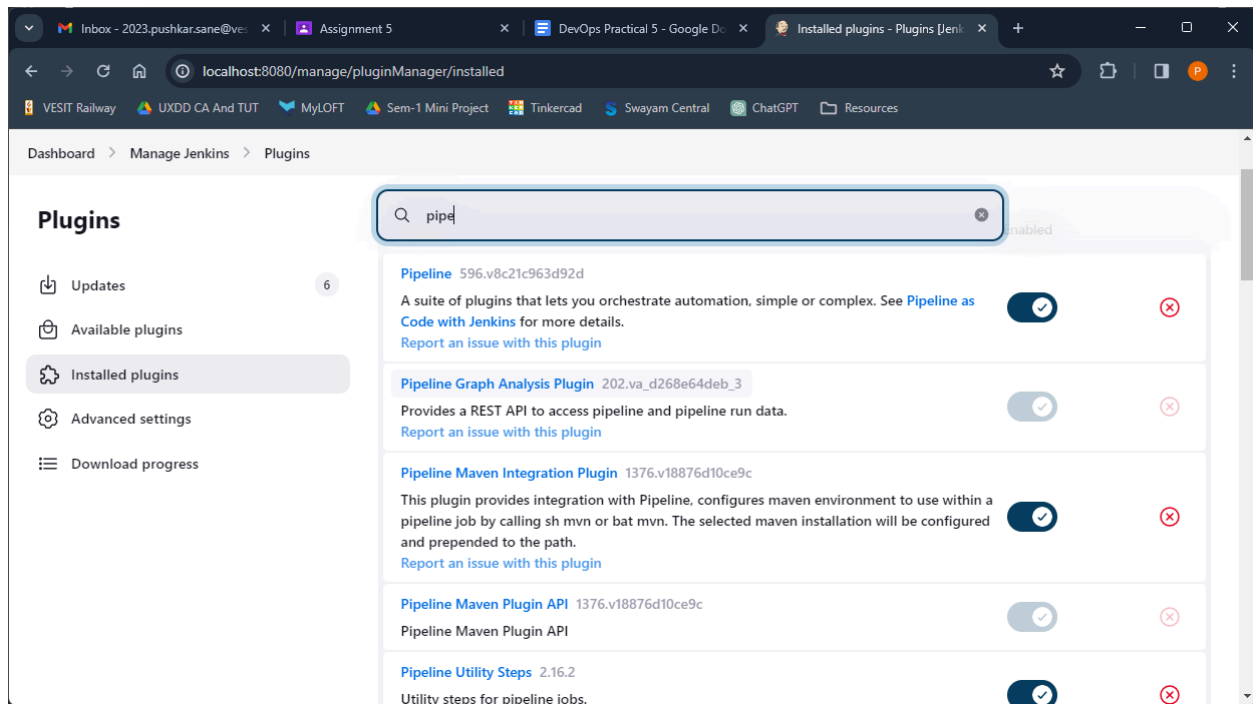
Jenkins pipeline is a powerful tool for automating software delivery pipelines. It allows users to define their entire software delivery process as code, enabling continuous integration and continuous delivery (CI/CD). This detailed guide aims to walk users through the process of implementing Jenkins pipelines using both scripted and declarative syntaxes.

Step 1: Setting up Jenkins:

1. Install Jenkins on your system by downloading the latest version from the official Jenkins website or using a package manager.
2. Access the Jenkins dashboard by navigating to <http://localhost:8080> (or the appropriate URL if Jenkins is hosted elsewhere).
3. Follow the instructions to complete the initial setup, including creating an admin user and installing recommended plugins.

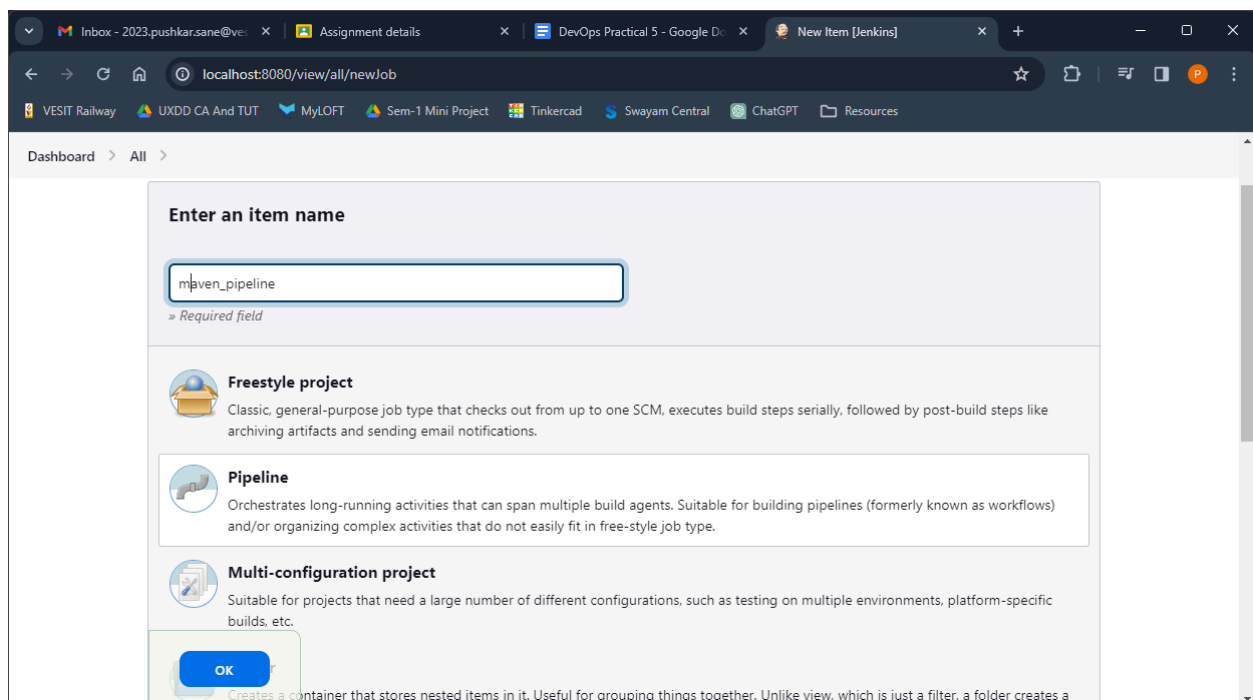
Step 2: Installing Pipeline Plugins:

1. Navigate to the Jenkins dashboard and click on "Manage Jenkins" in the left sidebar.
2. Select "Manage Plugins" from the dropdown menu.
3. Go to the "Available" tab and search for "Pipeline" plugins.
4. Install the following plugins:
 - Pipeline
 - Pipeline: GitHub Integration
 - Pipeline Maven Integration
 - Pipeline Utility
 - Maven integration



Step 3: Creating a New Pipeline Job and Configuration:

1. Click on "New Item" in the Jenkins dashboard.
2. Enter a name for your pipeline job and select "Pipeline" as the job type.
3. Click on "OK" to create the job.



The screenshot shows the Jenkins Configuration page for a job named 'maven_pipeline'. The browser address bar indicates the URL is 'localhost:8080/job/maven_pipeline/configure'. The left sidebar shows the 'Configure' section with tabs for 'General', 'Advanced Project Options', and 'Pipeline'. The 'General' tab is active. The 'Description' field contains the text 'This is a maven pipeline'. Below this, there are three unchecked checkboxes: 'Discard old builds', 'Do not allow concurrent builds', and 'Do not allow the pipeline to resume if the controller restarts'. The 'GitHub project' checkbox is checked. The 'Project url' field contains the value 'https://github.com/PrasadCodes1520/JENKINS_PIPE'. At the bottom, there are 'Save' and 'Apply' buttons.

Dashboard > maven_pipeline > Configuration

Configure

General

Advanced Project Options

Pipeline

Description

This is a maven pipeline

Plain text [Preview](#)

☐ Discard old builds ?

☐ Do not allow concurrent builds

☐ Do not allow the pipeline to resume if the controller restarts

☒ GitHub project

Project url ?

https://github.com/PrasadCodes1520/JENKINS_PIPE

Save Apply

The screenshot shows the Jenkins Configuration page for a job named 'maven_pipeline', specifically the 'Build Triggers' section. The browser address bar indicates the URL is 'localhost:8080/job/maven_pipeline/configure'. The left sidebar shows the 'Configure' section with tabs for 'General', 'Advanced Project Options', and 'Pipeline'. The 'General' tab is active. The 'Build Triggers' section contains several checkboxes: 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling' (checked), 'Poll SCM', 'Quiet period', and 'Trigger builds remotely (e.g., from scripts)'. Below this, there is an 'Advanced Project Options' section with a dropdown menu set to 'Advanced'. At the bottom, there are 'Save' and 'Apply' buttons.

Dashboard > maven_pipeline > Configuration

Configure

General

Advanced Project Options

Pipeline

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

☐ Quiet period ?

☐ Trigger builds remotely (e.g., from scripts) ?

Advanced Project Options

Advanced

Save Apply

The screenshot shows the Jenkins Pipeline Configuration page for a job named 'maven_pipeline'. The 'Definition' dropdown is set to 'Pipeline script from SCM'. The 'SCM' dropdown is set to 'Git'. The 'Repositories' section contains one repository with the URL 'https://github.com/PrasadCodes1520/JENKINS_PIPE.git' and 'Credentials' set to '- none -'. The 'Save' button is highlighted.

Dashboard > maven_pipeline > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories:

- Repository URL: `https://github.com/PrasadCodes1520/JENKINS_PIPE.git`
- Credentials: - none -

+ Add

Save Apply

The screenshot shows the Jenkins Pipeline Configuration page for a job named 'maven_pipeline'. The 'Branch Specifier' is set to '*/main'. The 'Repository browser' is set to '(Auto)'. The 'Script Path' is set to 'Jenkinsfile'. The 'Save' button is highlighted.

Dashboard > maven_pipeline > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

Branch Specifier (blank for 'any'): */main

Add Branch

Repository browser: (Auto)

Additional Behaviours: Add

Script Path: Jenkinsfile

Save Apply

Step 4: Writing a Scripted Pipeline:

1. In the pipeline job configuration page, scroll down to the "Pipeline" section.
2. Select "Pipeline script" from the definition dropdown.
3. Write your scripted pipeline code in the Script box.

4. Example of scripted pipeline:

```
pipeline {
  agent any
  stages {
    stage('Build') {
      steps {
        echo 'Building the project...'
      }
    }
    stage('Test') {
      steps {
        echo 'Running tests...'
      }
    }
    stage('Deploy') {
      steps {
        echo 'Deploying to production...'
      }
    }
  }
}
```

Step 5: Writing a Declarative Pipeline:

1. In the pipeline job configuration page, select "Pipeline script from SCM" from the definition dropdown.
2. Choose your SCM repository (e.g., Git) and provide the necessary repository URL and credentials.
3. Create a Jenkinsfile in your repository with the declarative pipeline syntax.
4. For example:

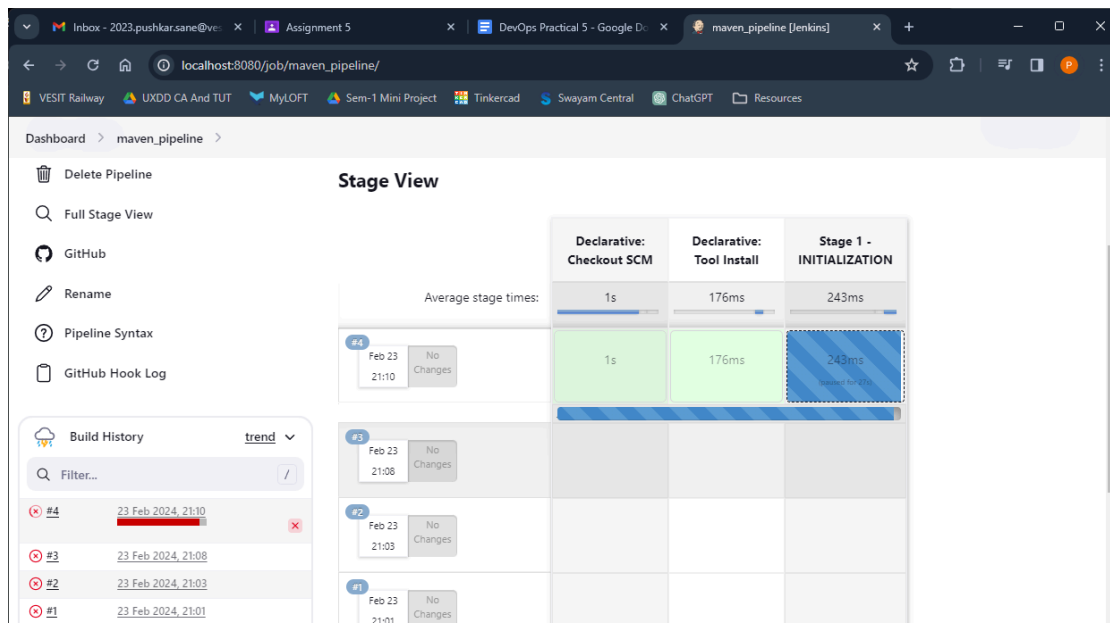
```

1 pipeline {
2   agent any
3   tools {
4     maven "maven_home"
5     jdk "jdk_home"
6   }
7   stages {
8     stage('Stage 1 - INITIALIZATION') {
9       steps {
10        input('Do you want to proceed?')
11        echo 'THIS IS THE END OF MAVEN PIPELINE USING JENKINS'
12      }
13    }
14    stage('Stage 2 - COMPILER CODE') {
15      steps {
16        bat "mvn compile"
17      }
18    }
19    stage('Stage 3 - UNIT TEST') {
20      steps {
21        echo "Running Unit Test"
22        bat "mvn test"
23      }
24    }
25    stage('Stage 4 - INTEGRATION TEST') {
26      steps {
27        echo "Running Integration Test"
28        bat "mvn verify"
29      }
30    }
31    stage('Stage 5 - CREATE BUILD') {
32      steps {
33        echo "Creating a build"
34        bat "mvn package"
35      }
36    }
37  }
38  post {
39    failure {
40      echo "Email has been sent for Jenkins build failure"
41    }
42  }
43 }

```

Step 6: Running the Pipeline:

1. Save your pipeline job configuration.
2. Trigger a build for the pipeline job.
3. Monitor the pipeline execution in the Jenkins dashboard.
4. View the console output and logs to track the progress of the pipeline stages.



Dashboard > maven_pipeline >

Delete Pipeline

Full Stage View

GitHub

Rename

Pipeline Syntax

GitHub Hook Log

Stage View

Average stage times:

	Declarative: Checkout SCM	Declarative: Tool Install	Stage 1 - INITIALIZATION
#4	1s	176ms	243ms
#3			
#2			
#1			

Do you want to proceed?

Proceed Abort

243ms (average for this job)

Build History

Filter...

#4	23 Feb 2024, 21:10
#3	23 Feb 2024, 21:08
#2	23 Feb 2024, 21:03
#1	23 Feb 2024, 21:01

Dashboard > maven_pipeline >

<> Changes

Build Now

Configure

Delete Pipeline

Full Stage View

GitHub

Rename

Pipeline Syntax

GitHub Hook Log

Stage View

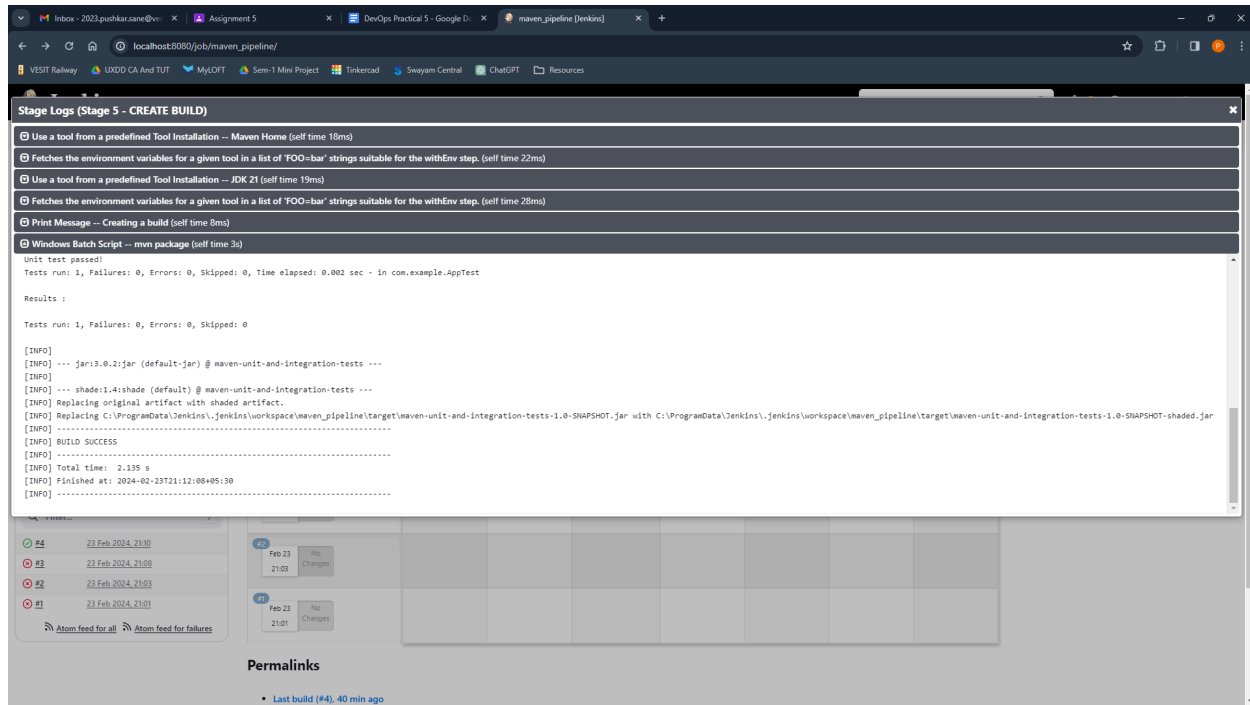
Average stage times: (Average full run time: ~2min 0s)

	Declarative: Checkout SCM	Declarative: Tool Install	Stage 1 - INITIALIZATION	Stage 2 - COMPILE CODE	Stage 3 - UNIT TEST	Stage 4 - INTEGRATION TEST	Stage 5 - CREATE BUILD
#4	1s	176ms	325ms (average for this job)	10s	7s	8s	3s
#3							
#2							
#1							

Permalinks

- Last build (#3), 2 min 4 sec ago
- Last failed build (#3), 2 min 4 sec ago
- Last unsuccessful build (#3), 2 min 4 sec ago
- Last completed build (#3), 2 min 4 sec ago

Click on the Logs to check the pipeline logs if there are any failures



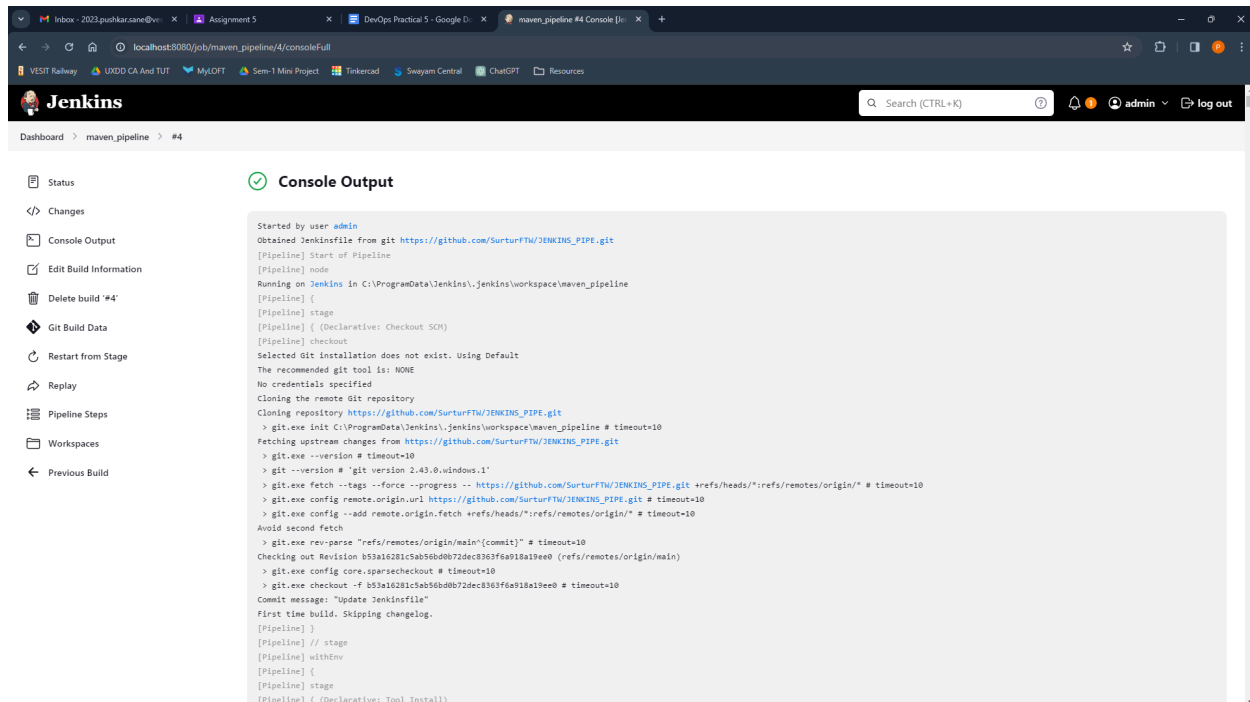
The screenshot shows the Jenkins web interface with the 'Stage Logs (Stage 5 - CREATE BUILD)' window open. The logs display the following steps and their durations:

- Use a tool from a predefined Tool Installation -- Maven Home (self time 18ms)
- Fetches the environment variables for a given tool in a list of 'FOO-bar' strings suitable for the withEnv step. (self time 22ms)
- Use a tool from a predefined Tool Installation -- JDK 21 (self time 19ms)
- Fetches the environment variables for a given tool in a list of 'FOO-bar' strings suitable for the withEnv step. (self time 28ms)
- Print Message -- Creating a build (self time 8ms)
- Windows Batch Script -- mvn package (self time 3s)

The logs also show the results of the 'mvn package' command, indicating that the build was successful. The 'Unit test passed' section shows that all tests passed.

Below the logs, there is a table showing the build history for the 'maven_pipeline' job. The table has columns for build number, status, and time. The last build (build #4) is shown as successful and completed 40 minutes ago.

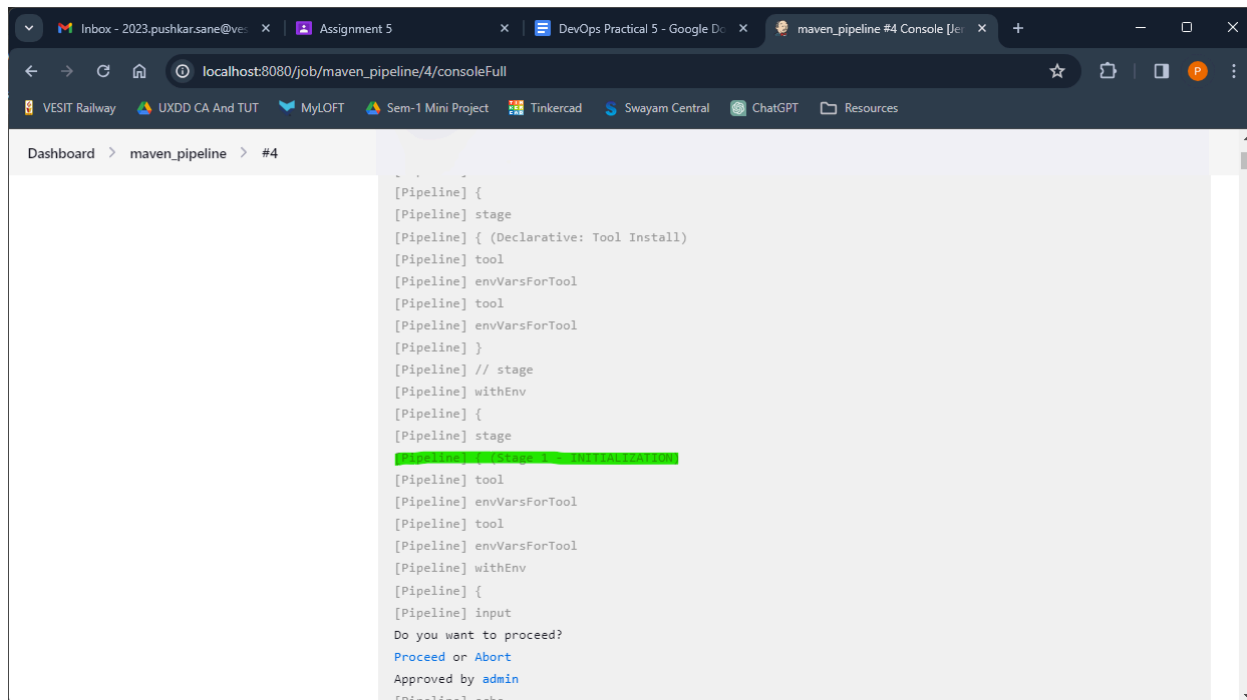
Console Output:



The screenshot shows the Jenkins web interface with the 'Console Output' window open for build #4. The output displays the following steps and their durations:

- Started by user admin
- Obtained Jenkinsfile from git https://github.com/SurturFTW/JENKINS_PIPE.git
- [Pipeline] Start of Pipeline
- [Pipeline] node
- Running on Jenkins in C:\ProgramData\Jenkins\jenkins\workspace\maven_pipeline
- [Pipeline] {
- [Pipeline] stage
- [Pipeline] { (Declarative: Checkout SCM)
- [Pipeline] checkout
- Selected Git installation does not exist. Using Default
- The recommended git tool is: NONE
- No credentials specified
- Cloning the remote Git repository
- Cloning repository https://github.com/SurturFTW/JENKINS_PIPE.git
- > git.exe init C:\ProgramData\Jenkins\jenkins\workspace\maven_pipeline # timeout=10
- Fetching upstream changes from https://github.com/SurturFTW/JENKINS_PIPE.git
- > git.exe --version # timeout=10
- > git --version # 'git version 2.43.0.windows.1'
- > git.exe fetch --tags --force --progress -- https://github.com/SurturFTW/JENKINS_PIPE.git +refs/heads/*:refs/remotes/origin/* # timeout=10
- > git.exe config remote.origin.url https://github.com/SurturFTW/JENKINS_PIPE.git # timeout=10
- > git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
- Avoid second fetch
- > git.exe rev-parse 'refs/remotes/origin/main:(commit)' # timeout=10
- Checking out Revision b53a16281c5ab56bd0b72dec8363f6a918a19ee0 (refs/remotes/origin/main)
- > git.exe config core.sparsecheckout # timeout=10
- > git.exe checkout -f b53a16281c5ab56bd0b72dec8363f6a918a19ee0 # timeout=10
- Commit message: "Update Jenkinsfile"
- First time build. Skipping changelog.
- [Pipeline] }
- [Pipeline] // stage
- [Pipeline] withEnv
- [Pipeline] {
- [Pipeline] stage
- [Pipeline] { (Declarative: Tool Install)

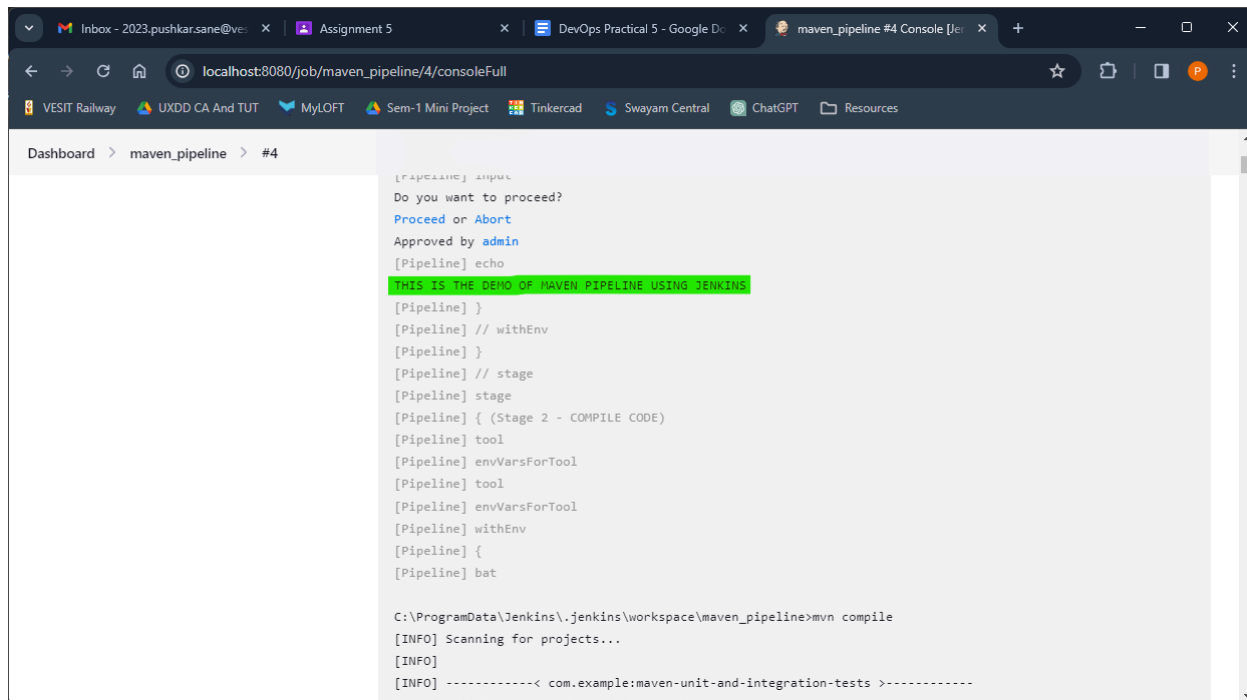
Initialization Stage:



The screenshot shows the Jenkins console output for the 'Initialization' stage of a Maven pipeline. The output includes the following text:

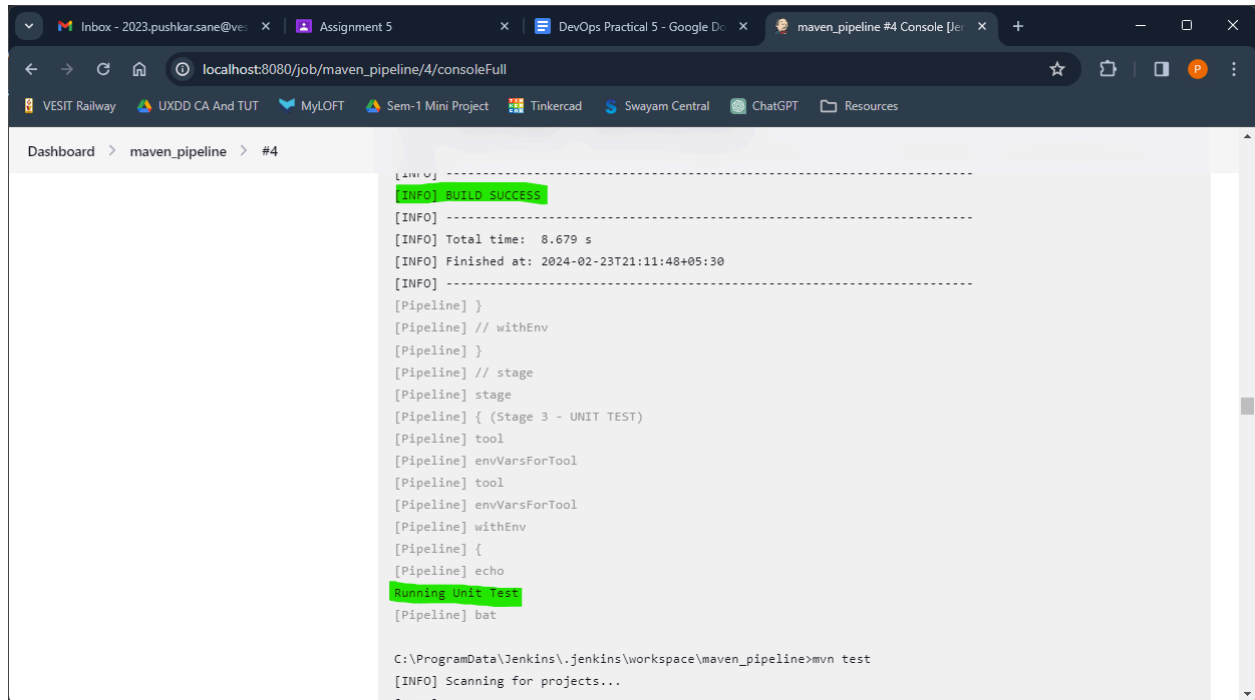
```
[Pipeline] {  
[Pipeline] stage  
[Pipeline] { (Declarative: Tool Install)  
[Pipeline] tool  
[Pipeline] envVarsForTool  
[Pipeline] tool  
[Pipeline] envVarsForTool  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] withEnv  
[Pipeline] {  
[Pipeline] stage  
[Pipeline] { (Stage 1 - INITIALIZATION)  
[Pipeline] tool  
[Pipeline] envVarsForTool  
[Pipeline] tool  
[Pipeline] envVarsForTool  
[Pipeline] withEnv  
[Pipeline] {  
[Pipeline] input  
Do you want to proceed?  
Proceed or Abort  
Approved by admin  
[Pipeline] echo
```

Compilation Stage:



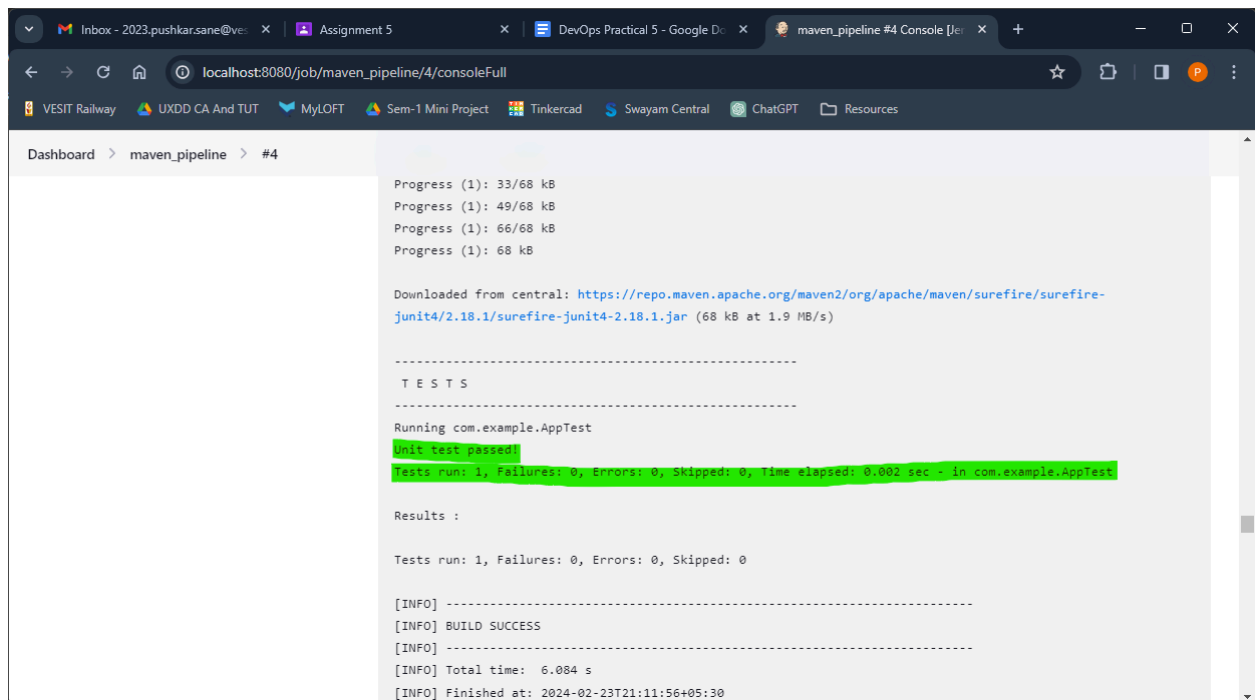
The screenshot shows the Jenkins console output for the 'Compilation' stage of a Maven pipeline. The output includes the following text:

```
[Pipeline] echo  
Do you want to proceed?  
Proceed or Abort  
Approved by admin  
[Pipeline] echo  
THIS IS THE DEMO OF MAVEN PIPELINE USING JENKINS  
[Pipeline] }  
[Pipeline] // withEnv  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] stage  
[Pipeline] { (Stage 2 - COMPILE CODE)  
[Pipeline] tool  
[Pipeline] envVarsForTool  
[Pipeline] tool  
[Pipeline] envVarsForTool  
[Pipeline] withEnv  
[Pipeline] {  
[Pipeline] bat  
  
C:\ProgramData\Jenkins\jenkins\workspace\maven_pipeline>mvn compile  
[INFO] Scanning for projects...  
[INFO]  
[INFO] -----< com.example:maven-unit-and-integration-tests >-----  
[INFO] Building maven-unit-and-integration-tests 1.0-SNAPSHOT
```

Unit Test Stage:

```
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 8.679 s
[INFO] Finished at: 2024-02-23T21:11:48+05:30
[INFO] -----
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Stage 3 - UNIT TEST)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] echo
Running Unit Test
[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\maven_pipeline>mvn test
[INFO] Scanning for projects...
```



```
Progress (1): 33/68 kB
Progress (1): 49/68 kB
Progress (1): 66/68 kB
Progress (1): 68 kB

Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit4/2.18.1/surefire-junit4-2.18.1.jar (68 kB at 1.9 MB/s)

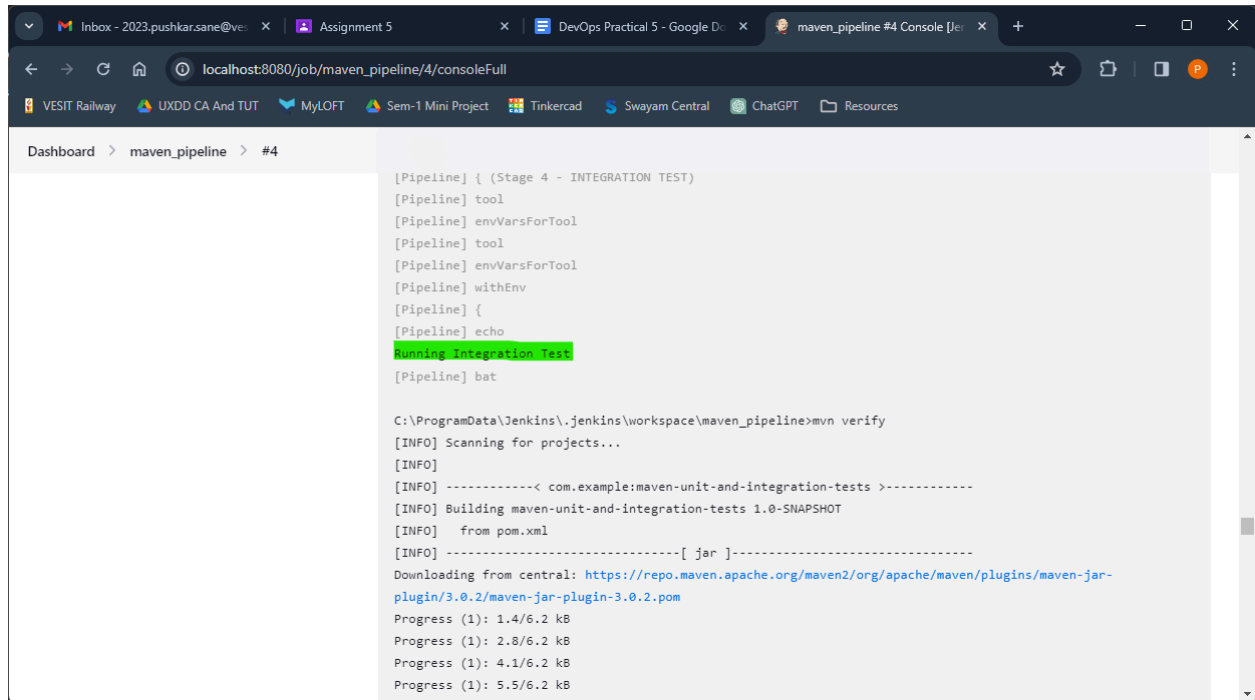
-----
T E S T S
-----
Running com.example.AppTest
Unit test passed!
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, time elapsed: 0.002 sec - in com.example.AppTest

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 6.084 s
[INFO] Finished at: 2024-02-23T21:11:56+05:30
```

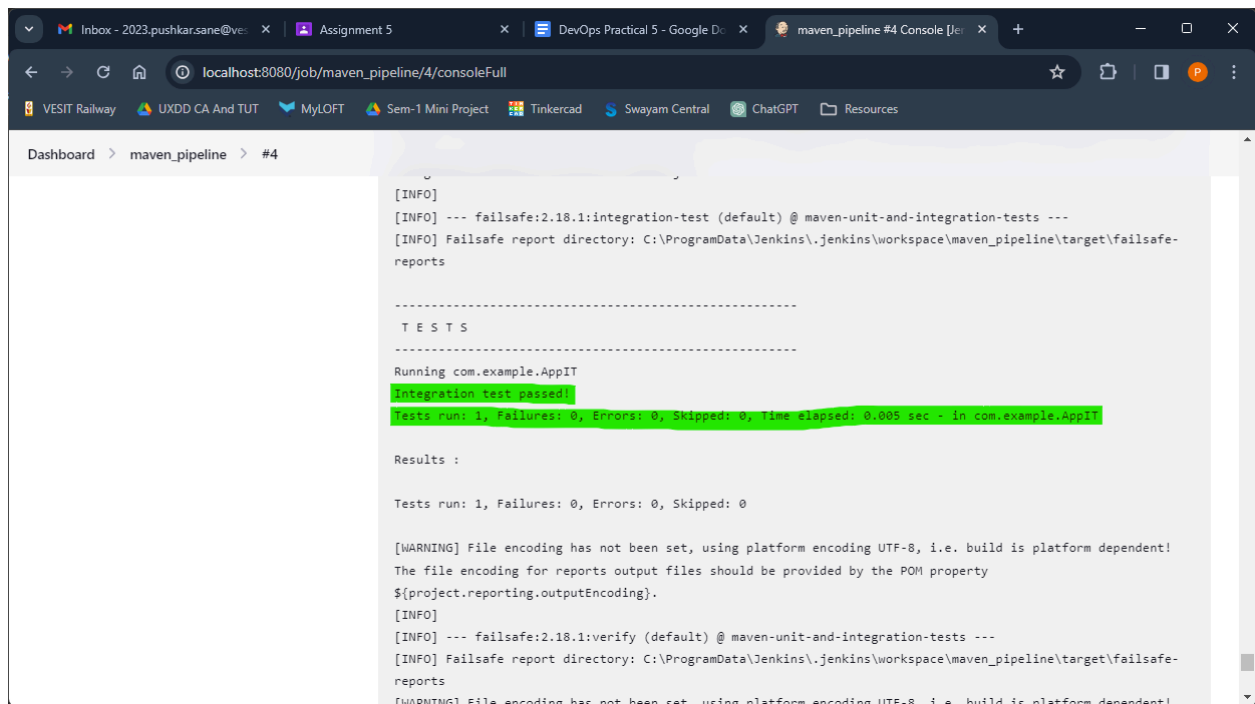
Integration Test Stage:



The screenshot shows the Jenkins console output for the 'Integration Test' stage of a pipeline. The output includes the pipeline configuration, the execution of 'mvn verify', and the progress of downloading Maven plugins. The 'Running Integration Test' line is highlighted in green.

```
[Pipeline] { (Stage 4 - INTEGRATION TEST)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] echo
Running Integration Test
[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\maven_pipeline>mvn verify
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:maven-unit-and-integration-tests >-----
[INFO] Building maven-unit-and-integration-tests 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-jar-
plugin/3.0.2/maven-jar-plugin-3.0.2.pom
Progress (1): 1.4/6.2 kB
Progress (1): 2.8/6.2 kB
Progress (1): 4.1/6.2 kB
Progress (1): 5.5/6.2 kB
```



The screenshot shows the continuation of the Jenkins console output. It displays the execution of 'mvn verify' with various informational messages, including the failsafe report directory and the results of the integration tests. The 'integration test passed' line is highlighted in green.

```
[INFO]
[INFO] --- failsafe:2.18.1:integration-test (default) @ maven-unit-and-integration-tests ---
[INFO] Failsafe report directory: C:\ProgramData\Jenkins\.jenkins\workspace\maven_pipeline\target\failsafe-
reports

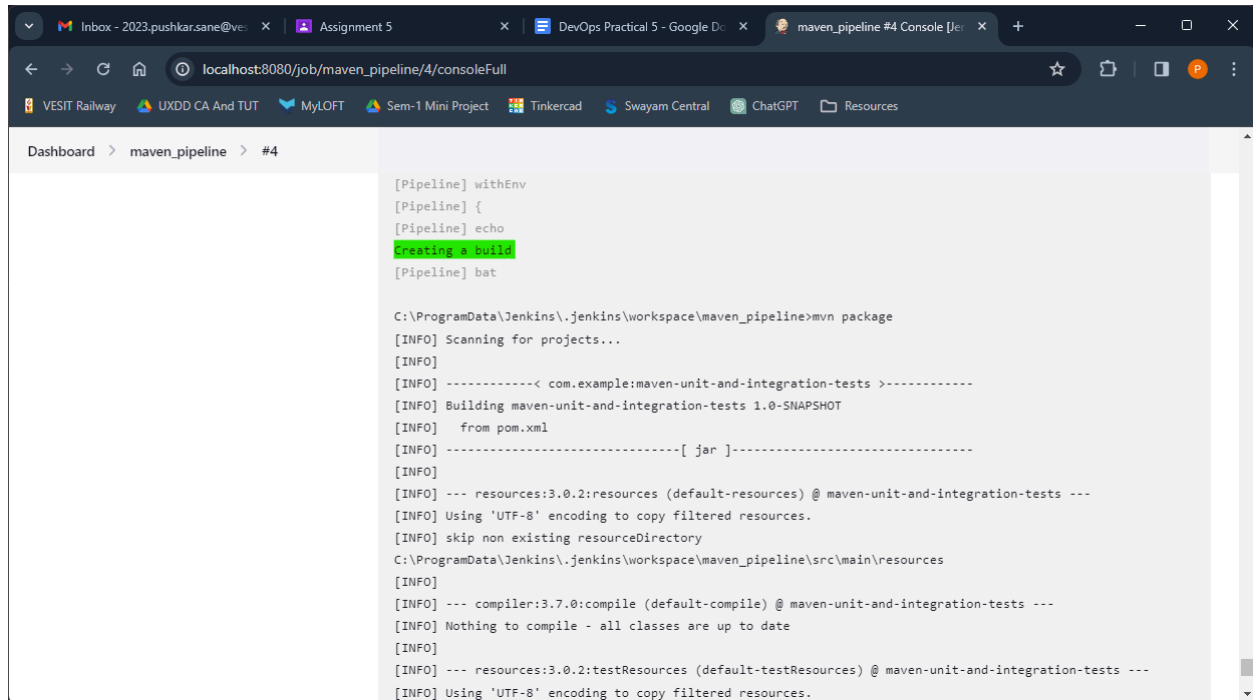
-----
T E S T S
-----
Running com.example.AppIT
integration test passed
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.003 sec - in com.example.AppIT

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
The file encoding for reports output files should be provided by the POM property
${project.reporting.outputEncoding}.
[INFO]
[INFO] --- failsafe:2.18.1:verify (default) @ maven-unit-and-integration-tests ---
[INFO] Failsafe report directory: C:\ProgramData\Jenkins\.jenkins\workspace\maven_pipeline\target\failsafe-
reports
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
```

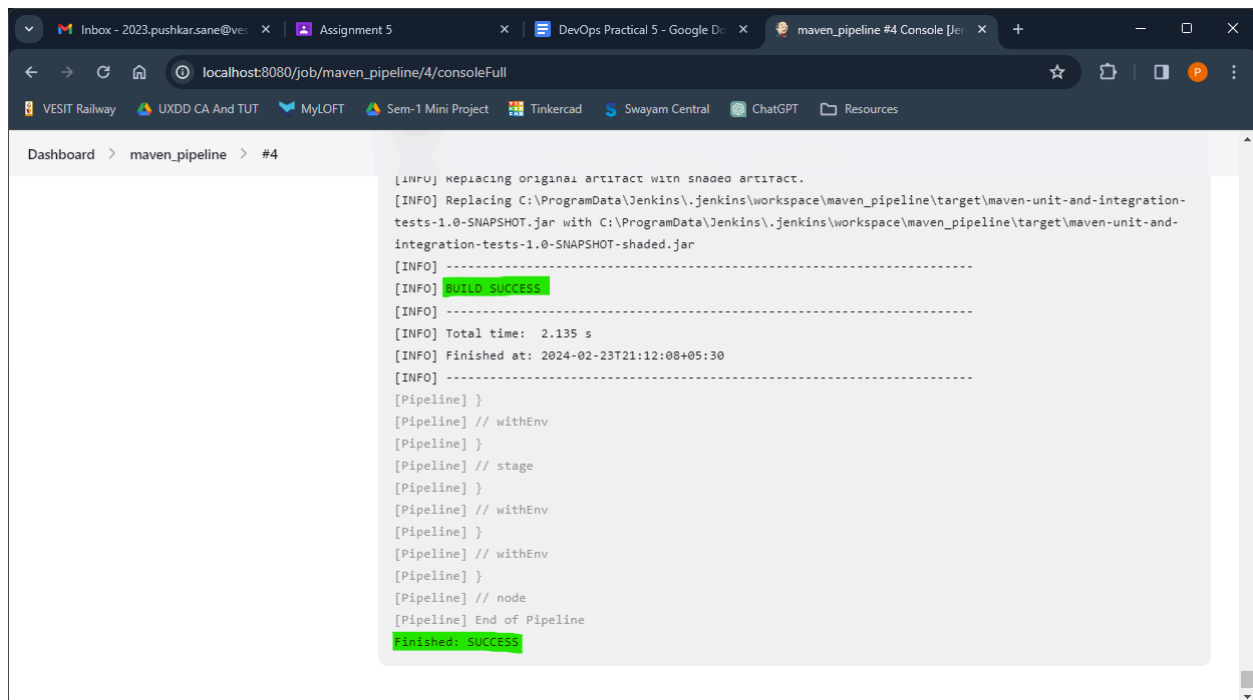
Creating Build Stage:



The screenshot shows the Jenkins console output for stage #4. The output begins with the pipeline configuration and the execution of the 'mvn package' command. It shows the scanning for projects, building the 'maven-unit-and-integration-tests 1.0-SNAPSHOT' from the 'pom.xml' file, and the execution of the 'jar' goal. The output also shows the resources being used, the compiler version, and the test resources. The build is successful, and the output ends with the 'Finished: SUCCESS' message.

```
[Pipeline] withEnv
[Pipeline] {
[Pipeline] echo
Creating a build
[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\maven_pipeline>mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:maven-unit-and-integration-tests >-----
[INFO] Building maven-unit-and-integration-tests 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- resources:3.0.2:resources (default-resources) @ maven-unit-and-integration-tests ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
C:\ProgramData\Jenkins\.jenkins\workspace\maven_pipeline\src\main\resources
[INFO]
[INFO] --- compiler:3.7.0:compile (default-compile) @ maven-unit-and-integration-tests ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- resources:3.0.2:testResources (default-testResources) @ maven-unit-and-integration-tests ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
```



The screenshot shows the final steps of the Jenkins build. It includes the replacement of the original artifact with the shaded artifact, the successful completion of the build, and the final 'Finished: SUCCESS' message. The output also shows the total time taken for the build and the finished timestamp.

```
[INFO] Replacing original artifact with shaded artifact.
[INFO] Replacing C:\ProgramData\Jenkins\.jenkins\workspace\maven_pipeline\target\maven-unit-and-integration-tests-1.0-SNAPSHOT.jar with C:\ProgramData\Jenkins\.jenkins\workspace\maven_pipeline\target\maven-unit-and-integration-tests-1.0-SNAPSHOT-shaded.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 2.135 s
[INFO] Finished at: 2024-02-23T21:12:08+05:30
[INFO]
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Conclusion:

Successfully implemented the Jenkins pipeline using Jenkins declarative pipeline.