Week 9: Pipeline Creation using script

Procedure

General :

Description :- provide the description of the project

Build triggers: here we can provide with build triggers of you choice

Advance project option:

Definition:

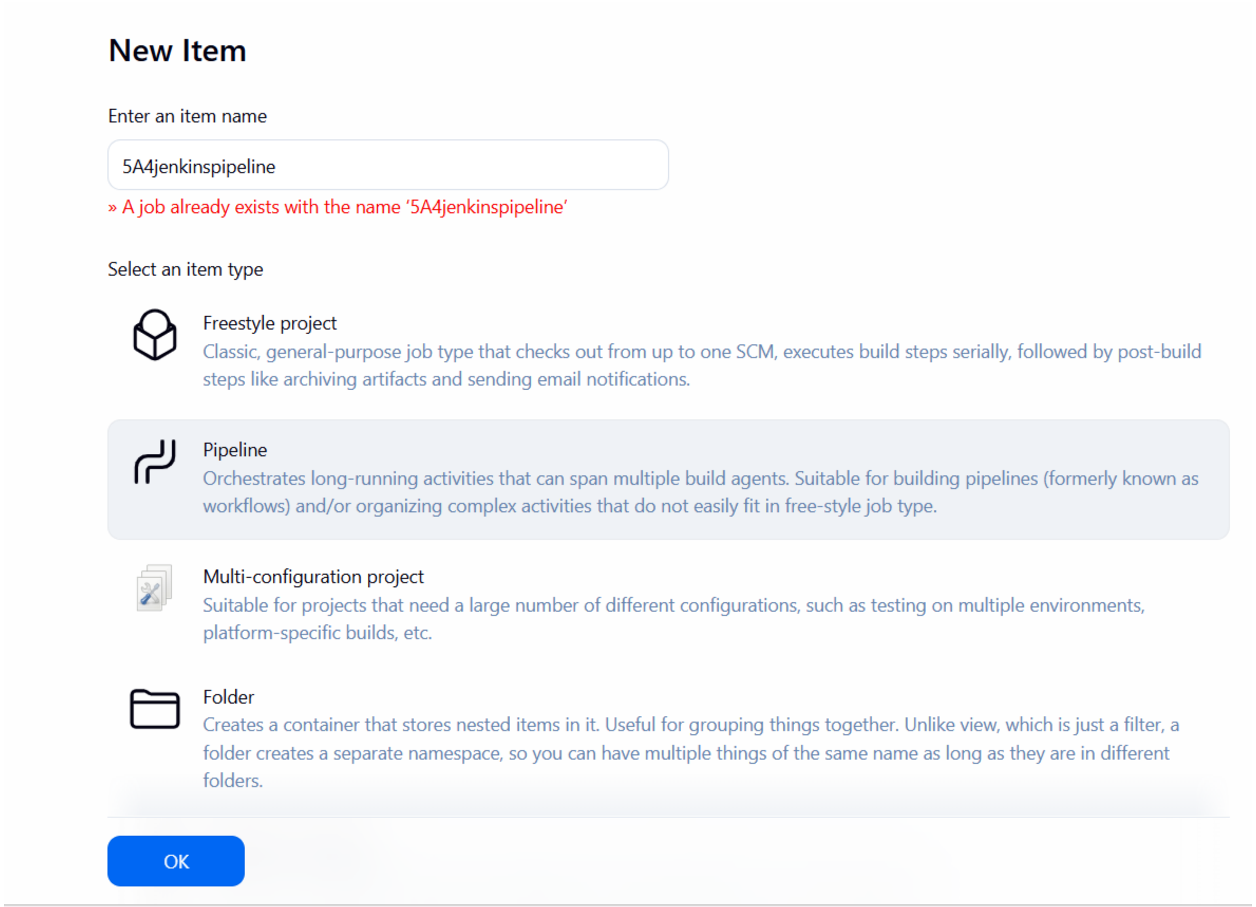
Choose pipeline script

Here code for pipeline -script

Copy the code there

```
pipeline {
   agent any
   tools{
      maven 'MAVEN-HOME'
   }
   stages {
      stage('git repo & clean') {
         steps {
            //bat "rmdir  /s /q mavenjava"
            bat "git clone provide your github link"
            bat "mvn clean -f mavenjava"
         }
      }
      stage('install') {
         steps {
            bat "mvn install -f mavenjava" #project name#
         }
      }
      stage('test') {
         steps {
            bat "mvn test -f mavenjava"
         }
      }
      stage('package') {
            steps {
                bat "mvn package -f mavenjava"
            }
        }
    }
}
```

Apply and save



New Item

Enter an item name

5A4jenkinspipeline

» A job already exists with the name '5A4jenkinspipeline'

Select an item type

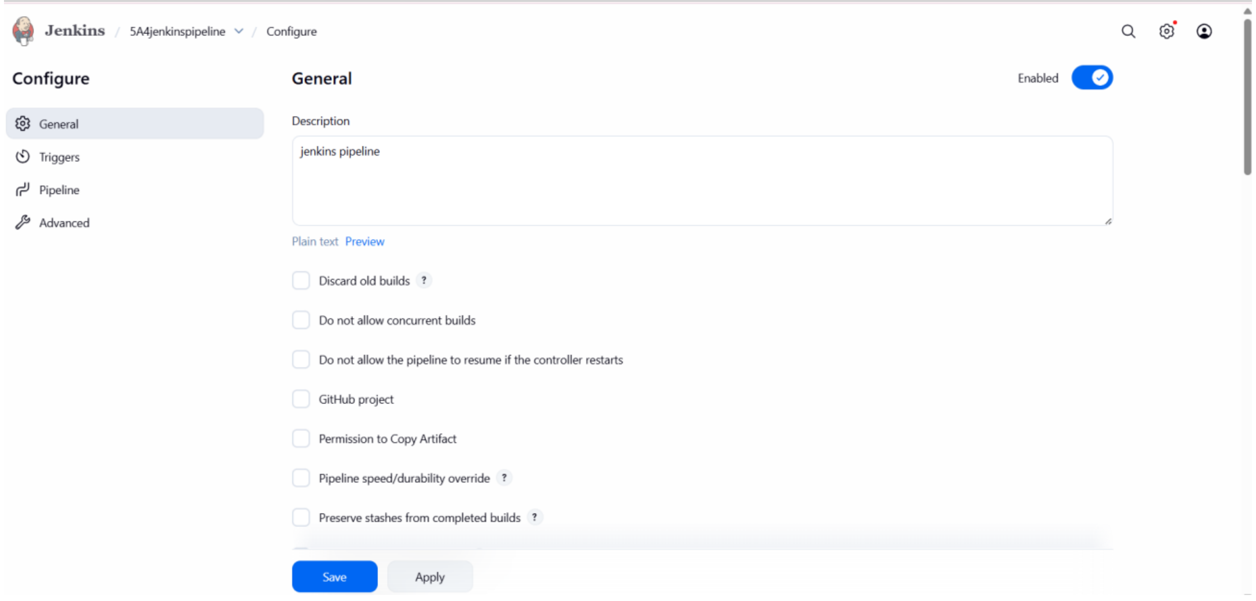
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK



Jenkins / 5A4jenkinspipeline / Configure

Configure

- General
- Triggers
- Pipeline
- Advanced

General

Enabled

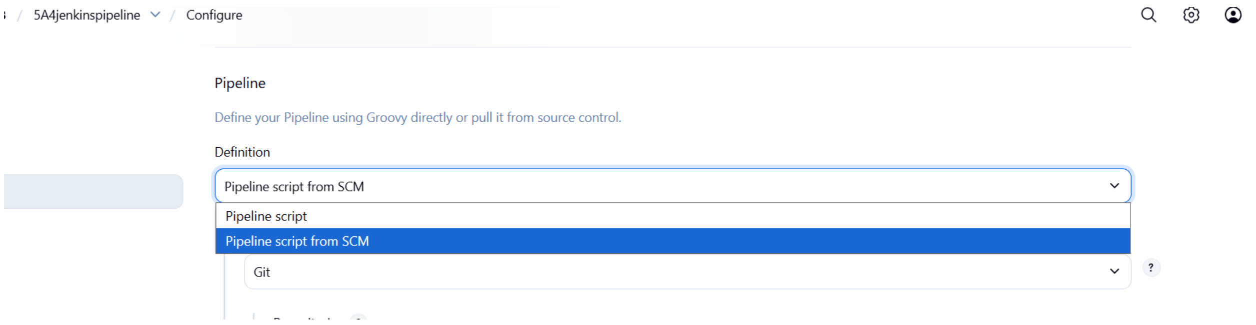
Description

jenkins 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
- [ ] Permission to Copy Artifact
- [ ] Pipeline speed/durability override ?
- [ ] Preserve stashes from completed builds ?

Save Apply


5A4jenkinspipeline / Configure

Pipeline

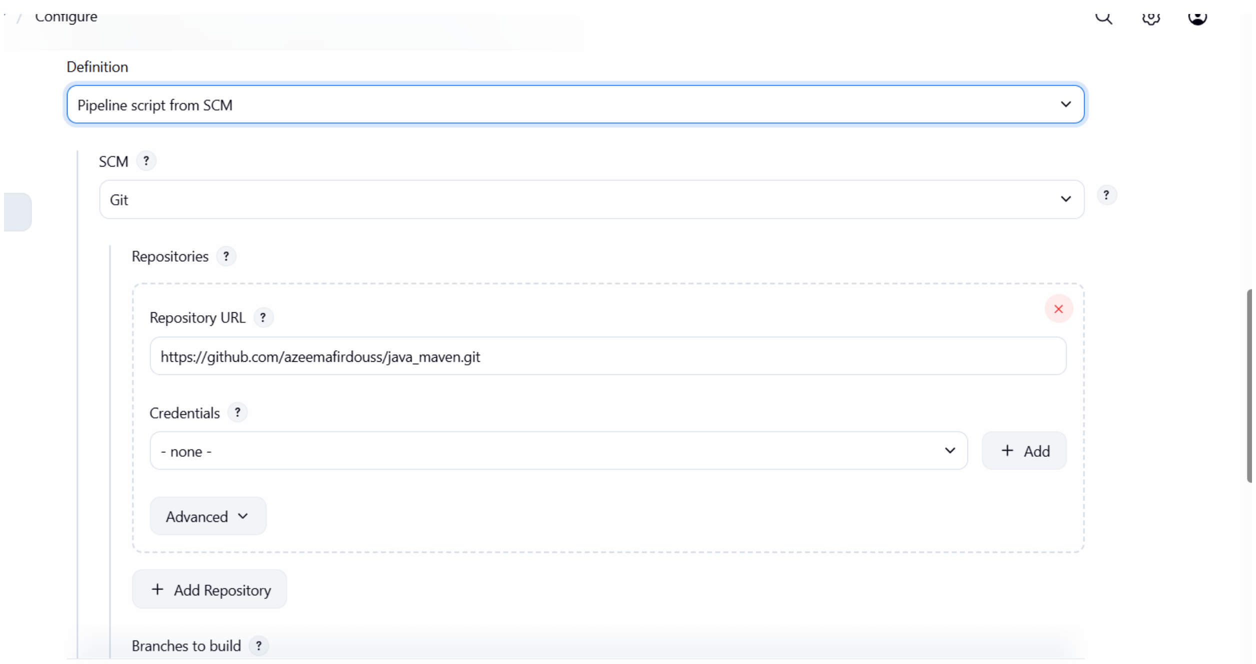
Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script from SCM

- Pipeline script
- Pipeline script from SCM

Git


Configure

Definition

Pipeline script from SCM

SCM

Git

Repositories

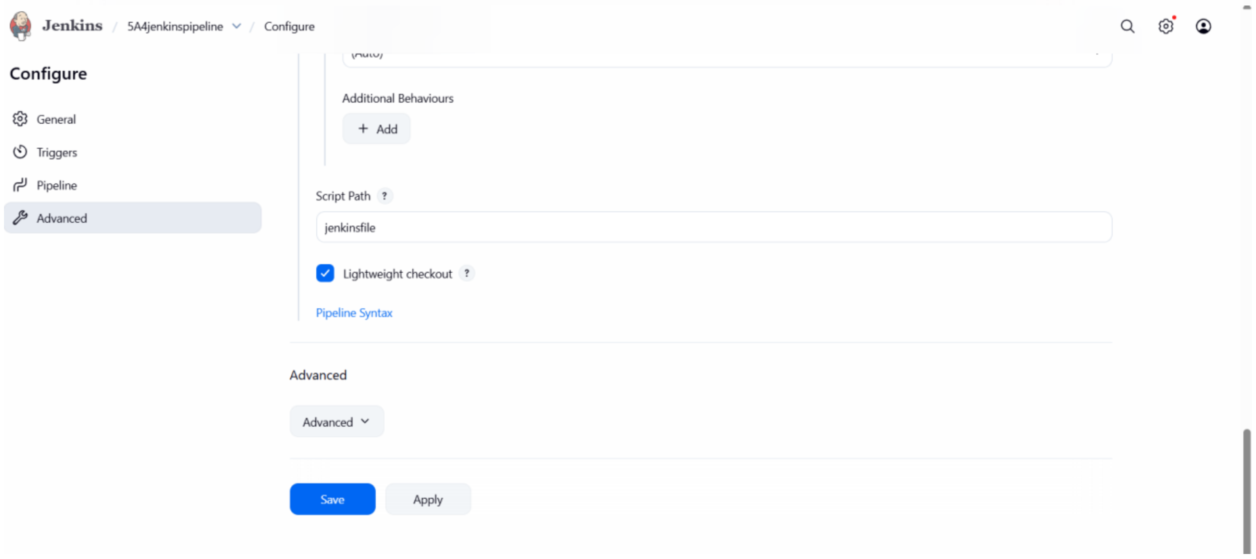
Repository URL

https://github.com/azeemafirdouss/java_maven.git

Credentials

- none -

Branches to build


Jenkins / 5A4jenkinspipeline / Configure

Additional Behaviours

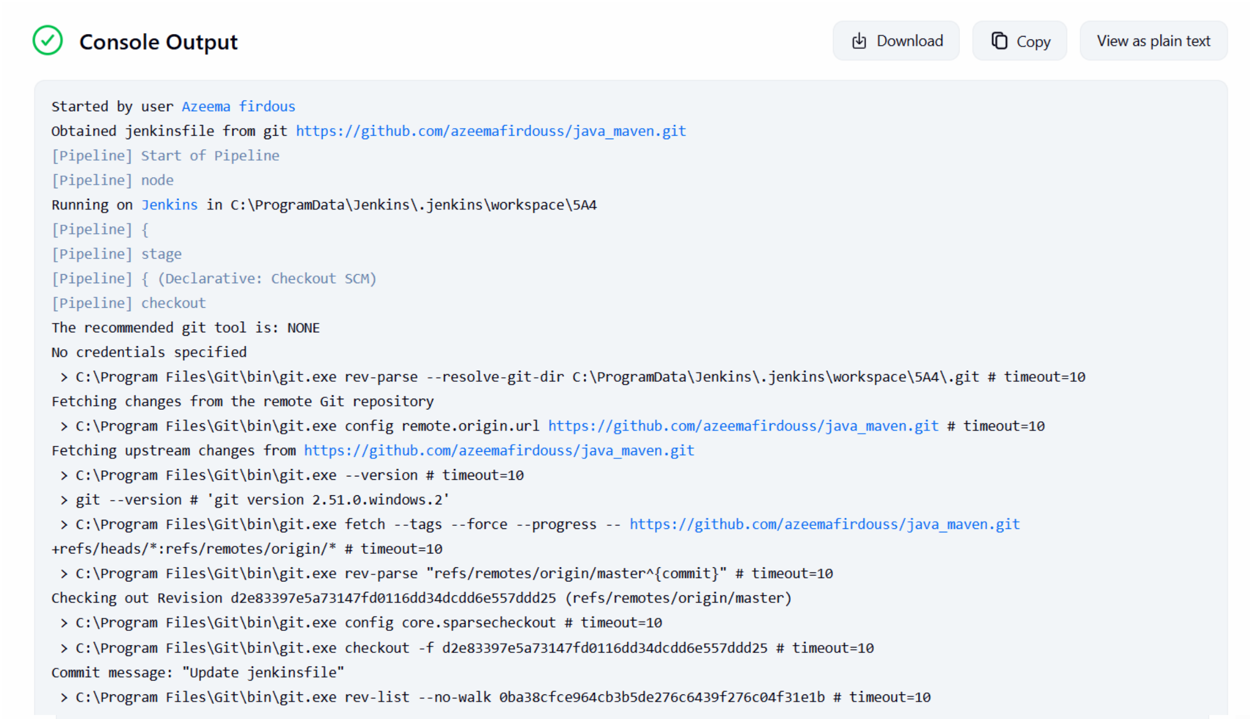
Script Path

jenkinsfile

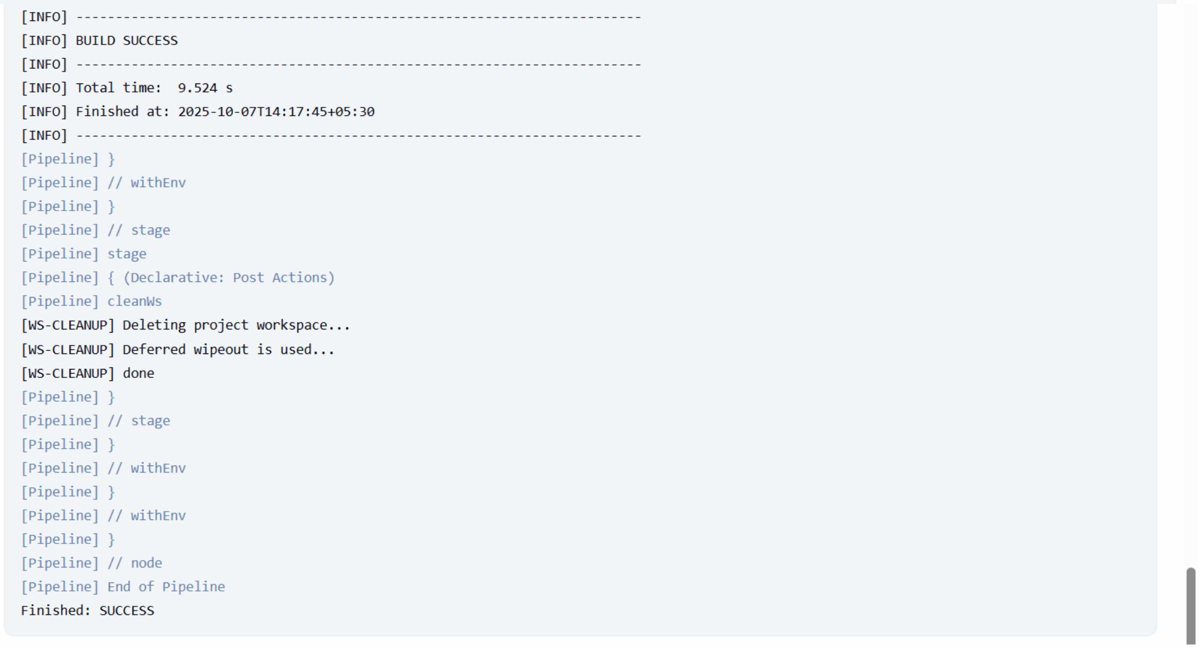
[✓] Lightweight checkout

Pipeline Syntax

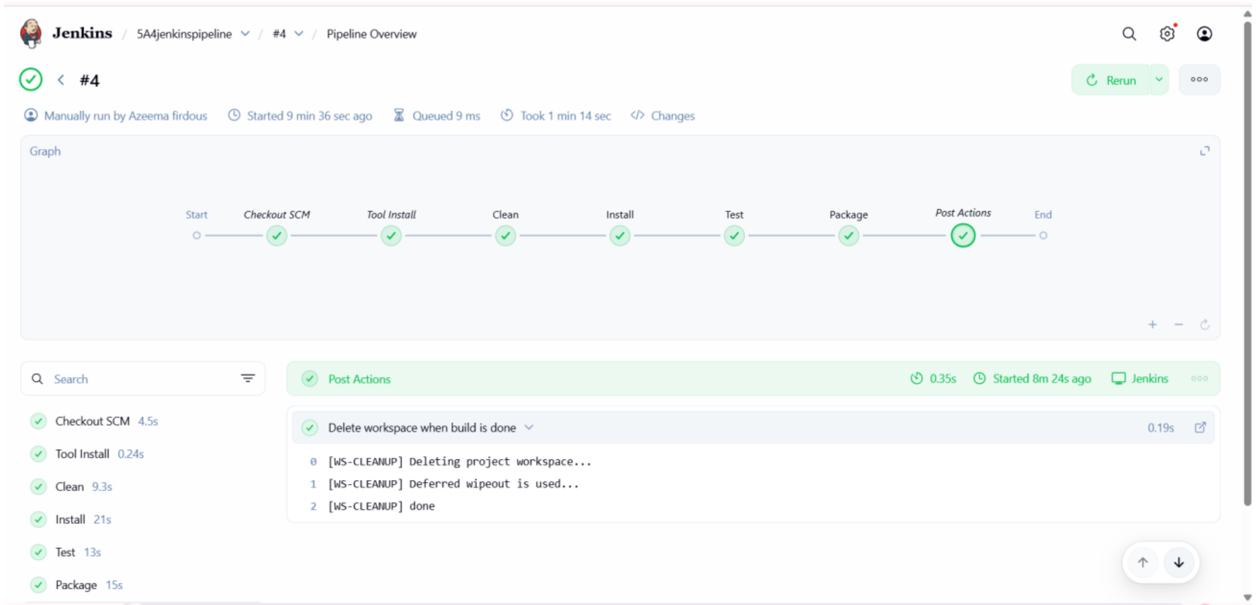
Advanced



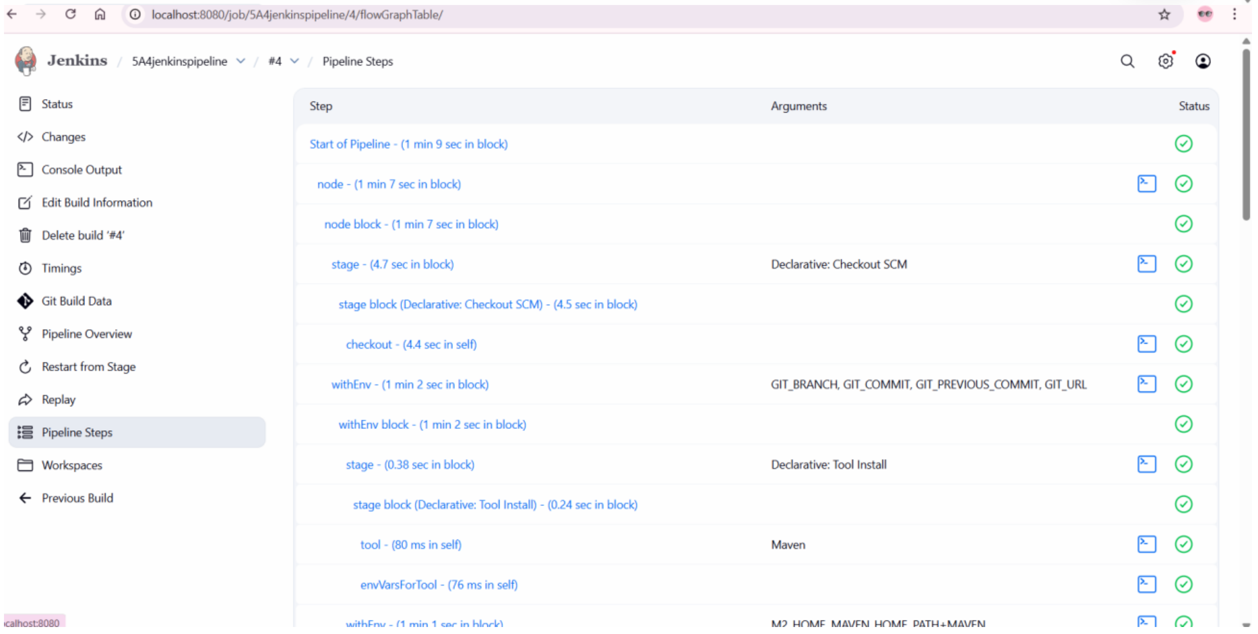
```
Started by user Azeema firdous
Obtained jenkinsfile from git https://github.com/azeemafirdouss/java_maven.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\5A4
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
The recommended git tool is: NONE
No credentials specified
 > C:\Program Files\Git\bin\git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\5A4\.git # timeout=10
Fetching changes from the remote Git repository
 > C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/azeemafirdouss/java_maven.git # timeout=10
Fetching upstream changes from https://github.com/azeemafirdouss/java_maven.git
 > C:\Program Files\Git\bin\git.exe --version # timeout=10
 > git --version # 'git version 2.51.0.windows.2'
 > C:\Program Files\Git\bin\git.exe fetch --tags --force --progress -- https://github.com/azeemafirdouss/java_maven.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > C:\Program Files\Git\bin\git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision d2e83397e5a73147fd0116dd34dcdd6e557ddd25 (refs/remotes/origin/master)
 > C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
 > C:\Program Files\Git\bin\git.exe checkout -f d2e83397e5a73147fd0116dd34dcdd6e557ddd25 # timeout=10
Commit message: "Update jenkinsfile"
 > C:\Program Files\Git\bin\git.exe rev-list --no-walk 0ba38cfce964cb3b5de276c6439f276c04f31e1b # timeout=10
```



```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  9.524 s
[INFO] Finished at: 2025-10-07T14:17:45+05:30
[INFO] ------------------------------------------------------------------------
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```


Jenkins / 5A4jenkinspipeline / #4 / Pipeline Overview
#4
Rerun
Manually run by Azeema firdous · Started 9 min 36 sec ago · Queued 9 ms · Took 1 min 14 sec · Changes
Graph
Start
Checkout SCM
Tool Install
Clean
Install
Test
Package
Post Actions
End
Search
Checkout SCM 4.5s
Tool Install 0.24s
Clean 9.3s
Install 21s
Test 13s
Package 15s
Post Actions · 0.35s · Started 8m 24s ago · Jenkins
Delete workspace when build is done · 0.19s

```
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
```



Step	Arguments	Status
Start of Pipeline - (1 min 9 sec in block)		✓
node - (1 min 7 sec in block)		✓
node block - (1 min 7 sec in block)		✓
stage - (4.7 sec in block)	Declarative: Checkout SCM	✓
stage block (Declarative: Checkout SCM) - (4.5 sec in block)		✓
checkout - (4.4 sec in self)		✓
withEnv - (1 min 2 sec in block)	GIT_BRANCH, GIT_COMMIT, GIT_PREVIOUS_COMMIT, GIT_URL	✓
withEnv block - (1 min 2 sec in block)		✓
stage - (0.38 sec in block)	Declarative: Tool Install	✓
stage block (Declarative: Tool Install) - (0.24 sec in block)		✓
tool - (80 ms in self)	Maven	✓
envVarsForTool - (76 ms in self)		✓
withEnv - (1 min 1 sec in block)	M2_HOME, MAVEN_HOME, PATH+MAVEN	✓

SBQ's:

1. **Your manager asks you to clean the workspace before building — which stage in this pipeline takes care of it?**

   You can add `cleanWs()` in the `post` block **or** create a dedicated stage before build. Currently: *Not present*, but it **should be added as the first step** or in a `post { always { cleanWs() } }`.

2. **The GitHub repository link is missing in the script — where exactly do you provide it?**

In the **Jenkins project configuration**, under **"Pipeline"** → **"SCM"** → **"Git"**, you provide the repo URL like:
`https://github.com/azeemafirdouss/java_maven.git`

3. **If Maven is not configured globally in Jenkins, which section of the pipeline will fail first?**

   The tools { maven 'Maven' } section — the **first Maven command in any stage** (like mvn clean) will fail with "mvn not found".

4. **A teammate complains the pipeline is not creating .war files — which stage is responsible?**

   The **package** stage.
   If the project is not set to generate a .war in pom.xml, it will produce a .jar instead.

5. **Your test cases failed, but the pipeline still continued — how will Jenkins behave in the test stage?**

   By default, if mvn test fails (non-zero exit code), Jenkins **will fail the stage** and **stop the pipeline**, unless it's wrapped in catchError or script.

6. **Instead of running nightly builds, you want this pipeline to trigger only when GitHub changes occur — where will you configure it?**

   In the Jenkins job settings:
   "Build Triggers" → Check 'GitHub hook trigger for GITScm polling'
   And configure a **webhook in your GitHub repo** under **Settings → Webhooks**.

7. **If you replace mvn clean with mvn compile, what difference will it make to the project build?**

   mvn clean: Deletes target/ directory.
   mvn compile: Compiles code but doesn't delete previous builds.
   Replacing clean with compile may cause stale artifacts to remain.

8. **The project folder name is not mavenjava but studentapp — which parts of the script must you edit?**

   If paths like `java_maven/pom.xml` are used, change them to:
   `mvn clean -f studentapp/pom.xml`

9. **If the install stage fails, will the test and package stages still run in this pipeline?**

   No.
   Jenkins pipeline **fails fast by default** — subsequent stages are **skipped** if a previous stage fails.

10. **A student asks where to add deployment steps to Tomcat after packaging — which is the best place in this script?**

    Add a new **`deploy`** stage **after `package`**. Example:

```
stage('Deploy') {
  steps {
          bat 'copy target/*.war path\\to\\tomcat\\webapps\\'
        }
    }
```

11. **Why is tools { maven 'MAVEN-HOME' } used in this pipeline?**

    To tell Jenkins to install or use a configured Maven version (from Global Tool Config) and **add it to the PATH** for that job.

12. **If GitHub credentials are private, how can you secure them in the git repo & clean stage?**

    Use Jenkins **credentials binding** and refer to them like this:

```
  git credentialsId: 'github-creds-id', url: 'https://github.com/your-
repo.git'
```

13. **Which Jenkins plugin is needed to recognize the pipeline { ... } structure in this script?**

    Pipeline Plugin (also known as **Declarative Pipeline Plugin**)

14. **In Windows, the script uses bat commands — what change would you make if Jenkins runs on Linux?**

    Replace bat with sh:

    sh 'mvn clean'

15. **How will you modify the pipeline to stop execution if the GitHub clone command fails?**

Your project lead asks you to print a welcome message in Jenkins to confirm the pipeline is working — how would you script it?

Jenkins already stops on failure. But if you're using a manual git clone, ensure you check errors:

bat 'git clone https://repo.git || exit 1'

✅ Better: Use Jenkins **SCM checkout** with error handling.

16. **A teammate forgot to pull the latest GitHub code before building; how can you fix this in your pipeline?**

    Use Jenkins' **checkout scm** or enable **"Git SCM" → "Wipe workspace and fetch"** in the job config to force fresh pull.

17. **Your Java file Hello.java must be compiled every time code is committed — how will you add this step?**

    Add a new stage before test:

    stage('Compile Java') {

       steps {

          bat 'javac src/main/java/Hello.java'

       }

    }

    Or rely on mvn compile.

18. **Tests should stop the pipeline if they fail — where will you put the mvn test command?**

    In the `test` stage — it already is there:

```
stage('Test') {
      steps {
          bat 'mvn test'
      }
 }
```

    If test fails, Jenkins will stop unless `catchError` or `continueOnFailure` is used.

19. **Every evening at 6 PM your pipeline should run automatically — how can you set this in Jenkins?**

    Go to job → **Configure → Build Triggers** → Enable **"Build periodically"**
    Use cron syntax:

    H 18 * * *

20. **You only want to package the project if compilation succeeds — how would you connect the stages?**

    Just define compile and package as separate stages.
    Jenkins **runs sequentially**, so package only runs if compile succeeds.

21. **Your professor wants a .jar file generated for submission — what pipeline stage will you add?**

    No extra stage needed if pom.xml uses packaging: jar.
    But for clarity:

```
 stage('Build Jar') {
   steps {
       bat 'mvn package'
   }
}
```

Output will be in target/java_maven-0.0.1-SNAPSHOT.jar.

22. **A Git clone step is failing due to wrong credentials — how would you secure and use them in your script?**

    Use Jenkins Credentials:

    ```
    git url: 'https://github.com/azeemafirdouss/java_maven.git', credentialsId: 'your-cred-id'
    ```

    Store the credentials in Jenkins → **Manage Credentials**.

23. **A teammate wants to see the build number inside console logs — how do you print it in the pipeline?**

    Add:

    ```
    echo "Build number is ${env.BUILD_NUMBER}"
    ```

    Or

    ```
    bat "echo Build number is %BUILD_NUMBER%"
    ```

Azeema firdous

23BD1A05A4

CSE-F