Lab Exercise 20

Creating a Pipeline Script

Objective: To create a pipeline script for automating build processes in Jenkins

Tools required: Jenkins

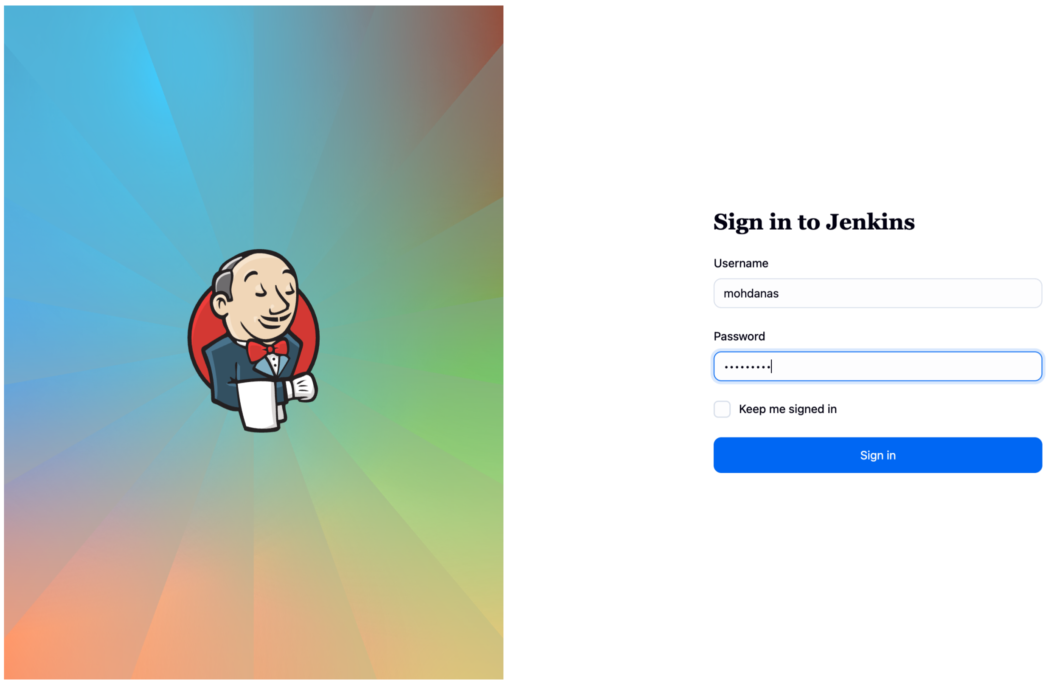
Prerequisites: None

Steps to be followed:

1. Log in to the Jenkins CI tool and create a pipeline script

Step 1: Log in to the Jenkins CI tool and create a pipeline script

1. Open the browser, go to the Jenkins **Dashboard** by typing **localhost:8080** in your browser, provide the credentials, and click the **Sign in** button



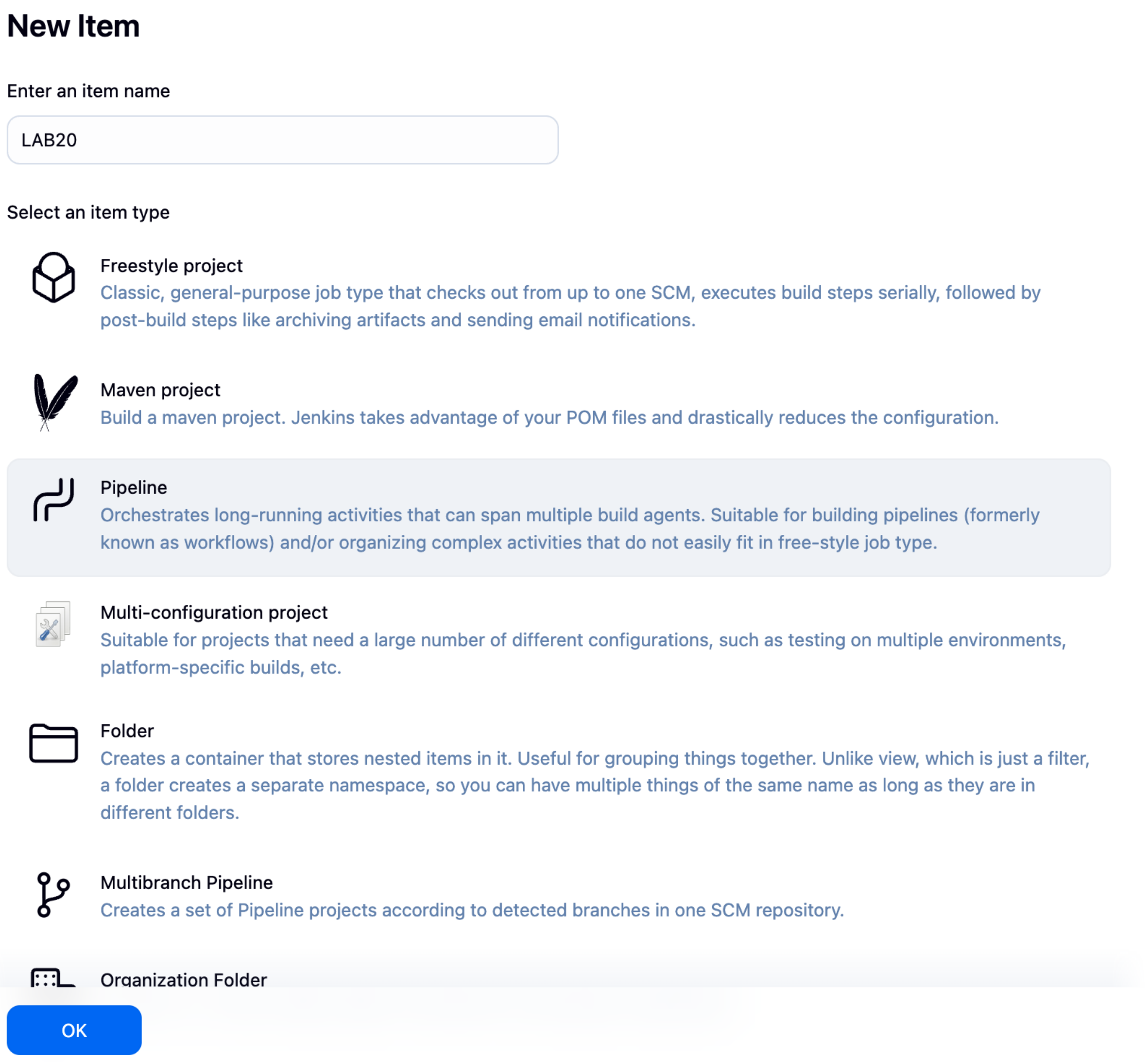
Sign in to Jenkins

Username: mohdanas

Password: •••••••••

2. Click on the **New Item** option as shown in the screenshot below:

3. Enter a desired name for the project, select **Pipeline**, and then click on **OK** as shown in the screenshot below:



New Item

Enter an item name

LAB20

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.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

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.

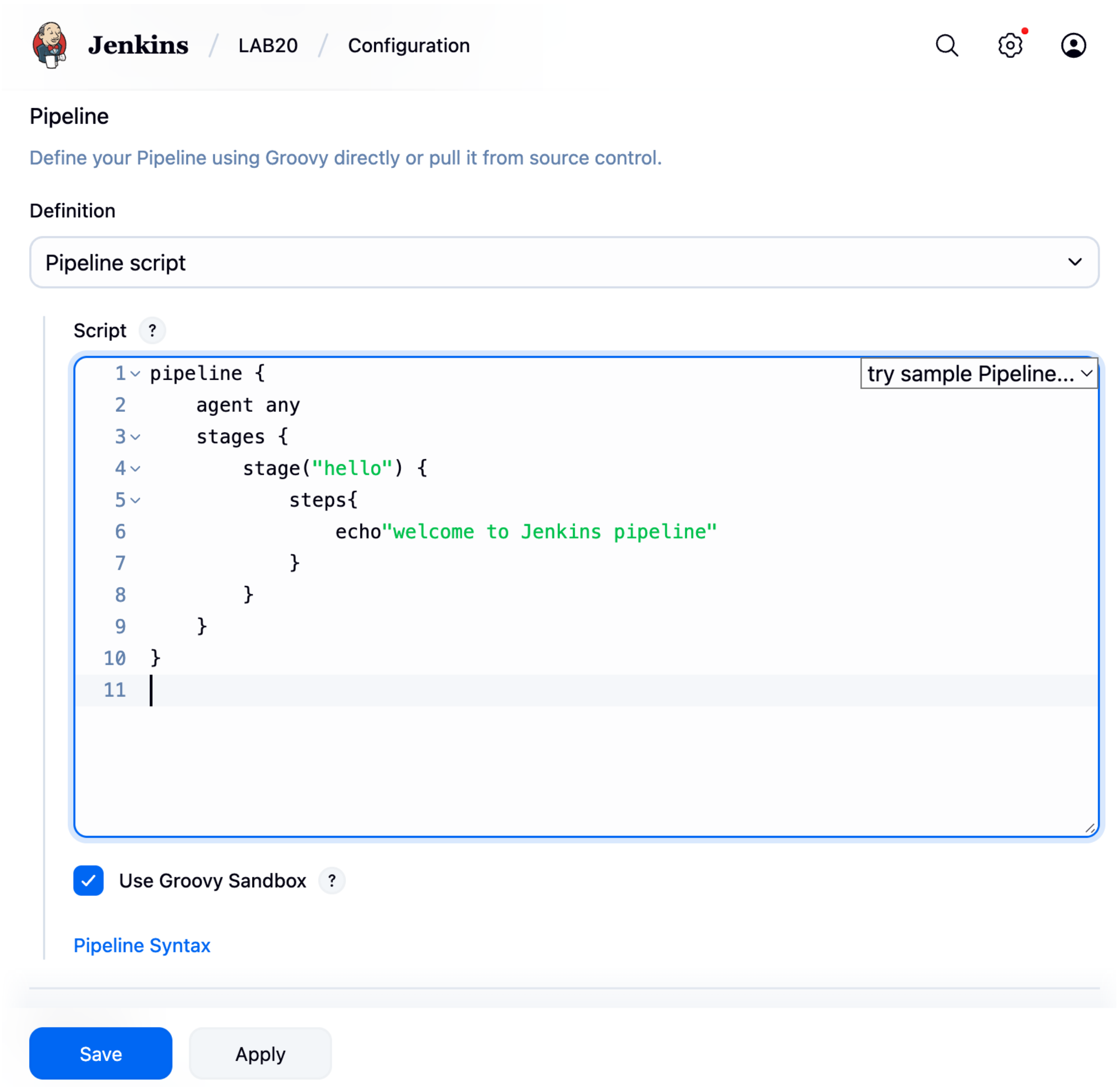
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder

4. Click on **Pipeline** as shown in the screenshot below:

5. Enter the following pipeline script in the script editor and click on **Save** as shown in the screenshot below:

```
pipeline {
  agent any
  stages {
    stage("hello") {
      steps{
        echo"welcome to Jenkins pipeline"
      }
    }
  }
}
```



Pipeline

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

Definition

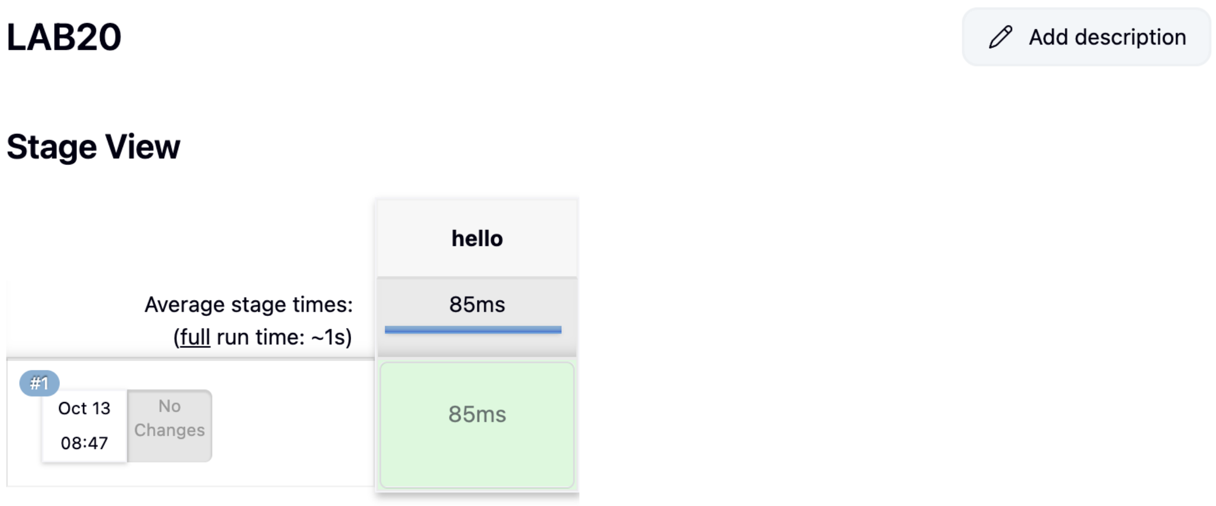
Pipeline script

Script

```
pipeline {
    agent any
    stages {
        stage("hello") {
            steps{
                echo"welcome to Jenkins pipeline"
            }
        }
    }
}
```

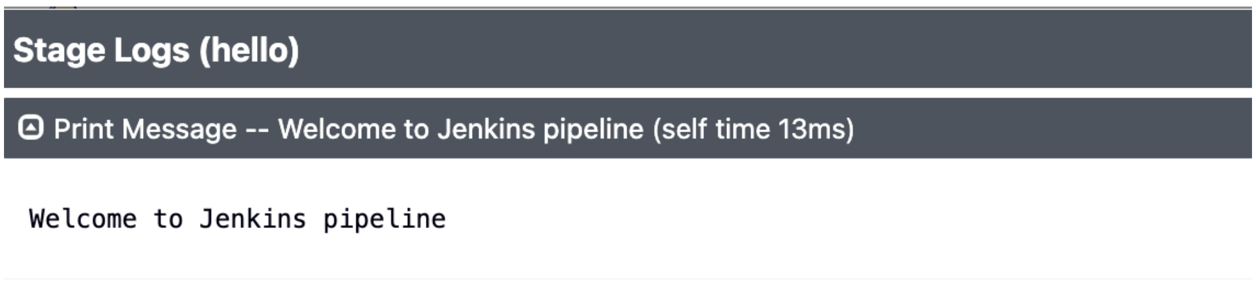
6. Click on **Build Now** to run the pipeline script as shown in the screenshot below:

7. Hover over the milliseconds number next to the build stage name as shown in the screenshot below:



8. Click on **Logs** as shown in the screenshot below:

9. Check for the message in the top-left corner to confirm the successful execution of the pipeline stage as shown in the screenshot below:



Stage Logs (hello)

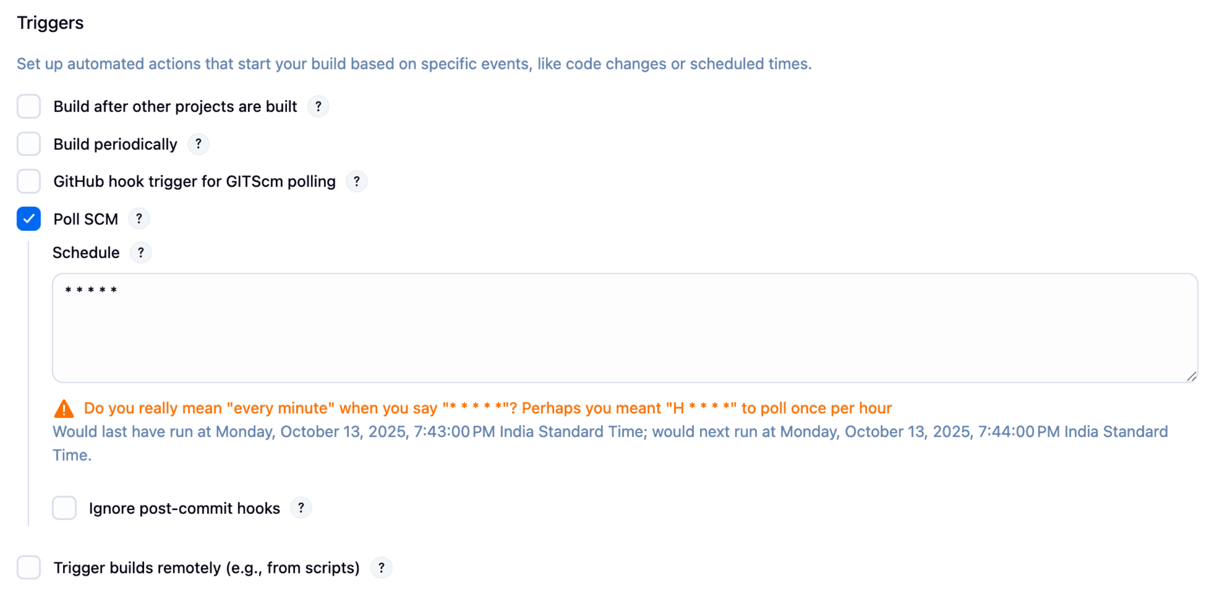
Print Message -- Welcome to Jenkins pipeline (self time 13ms)

```
Welcome to Jenkins pipeline
```

By following these steps, you have successfully created a pipeline script for automating build processes in Jenkins.

Polling from SCM

Step 1:
Go to your Jenkins project and select **“Configure.”**
Under **Build Triggers**, check the option **“Poll SCM.”**
Step 2:
In the **Schedule** field, enter the cron syntax



Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub hook trigger for GITScm polling
- [✓] Poll SCM

Schedule

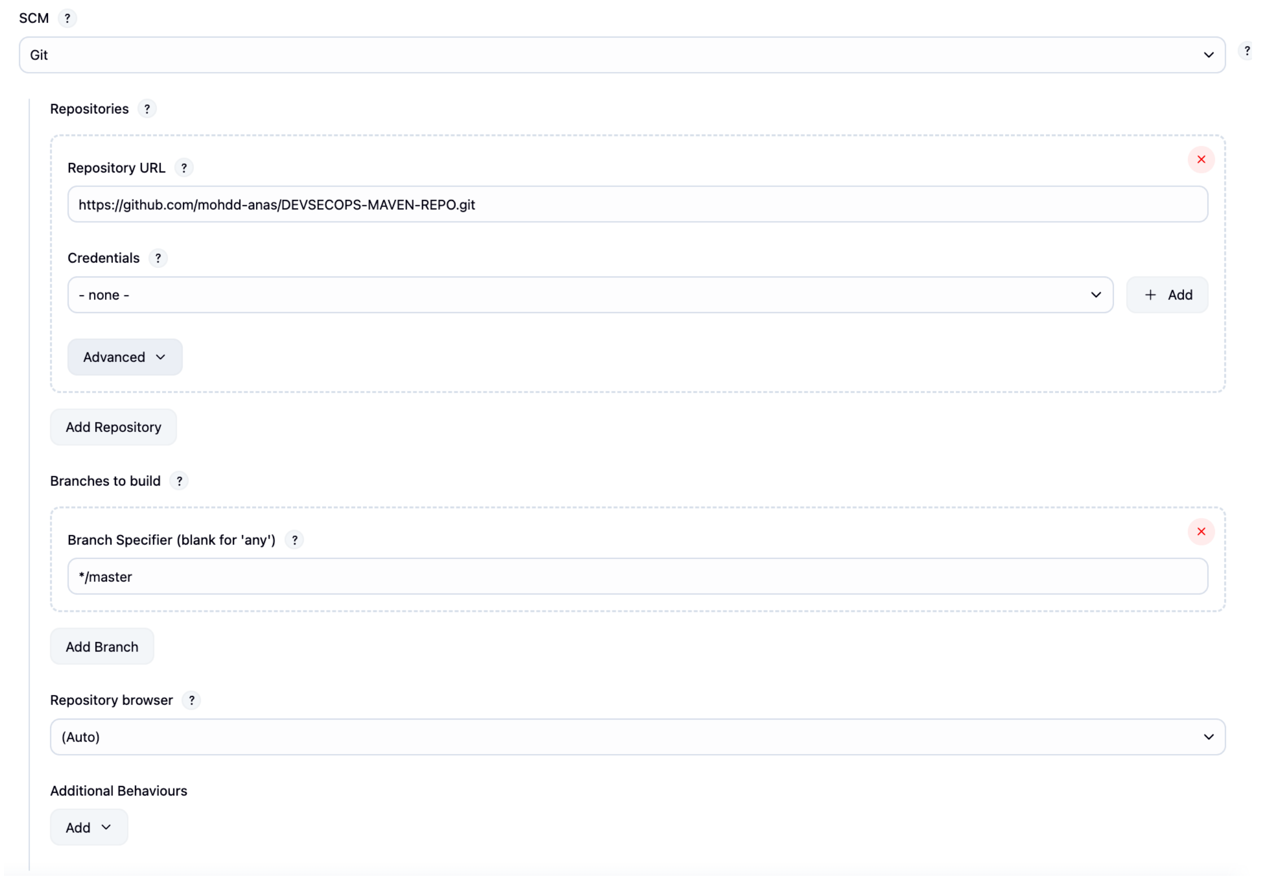
```
* * * * *
```

Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour
Would last have run at Monday, October 13, 2025, 7:43:00 PM India Standard Time; would next run at Monday, October 13, 2025, 7:44:00 PM India Standard Time.

- [ ] Ignore post-commit hooks
- [ ] Trigger builds remotely (e.g., from scripts)

Step 3:
Scroll down to the **Pipeline** section.
Select **“Pipeline script from SCM.”**

- Choose **Git** as the SCM.
- Enter your **Repository URL** (e.g., `https://github.com/mohdd-anas/DEVSECOPS-MAVEN-REPO.git`).
- Enter branch name as `*/master`.



SCM

Git

Repositories

Repository URL

https://github.com/mohdd-anas/DEVSECOPS-MAVEN-REPO.git

Credentials

- none -

Branches to build

Branch Specifier (blank for 'any')

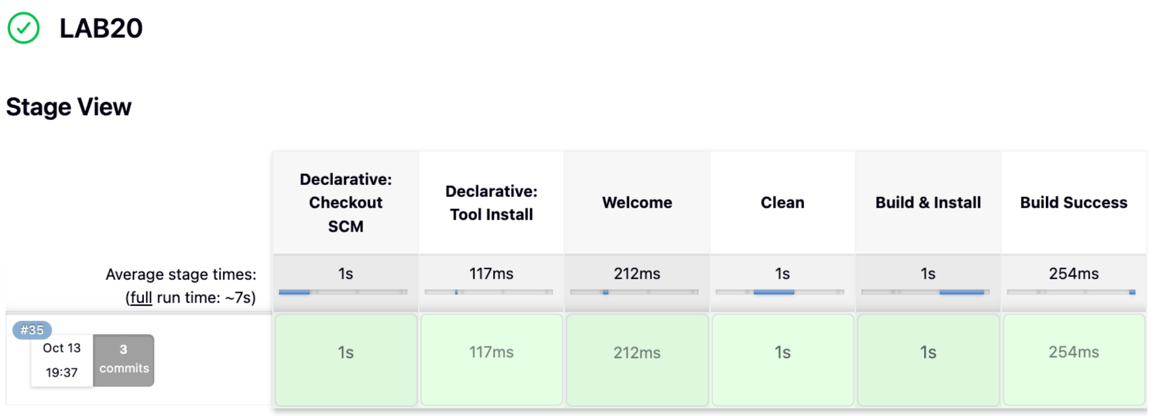
*/master

Repository browser

(Auto)

Additional Behaviours

Step 4:
Make a commit/push in your GitHub repository.
Jenkins will automatically detect the change and **trigger a new build.**

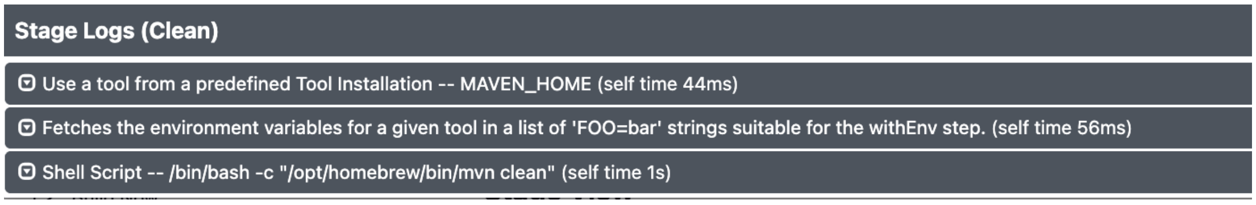


LAB20

Stage View

	Declarative: Checkout SCM	Declarative: Tool Install	Welcome	Clean	Build & Install	Build Success
Average stage times: (full run time: ~7s)	1s	117ms	212ms	1s	1s	254ms
#35 Oct 13 19:37 3 commits	1s	117ms	212ms	1s	1s	254ms

Step 5:
Go to **Build History → Console Output** to **check the build logs** and confirm that the polling triggered the build.



Stage Logs (Clean)

```
Use a tool from a predefined Tool Installation -- MAVEN_HOME (self time 44ms)
Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 56ms)
Shell Script -- /bin/bash -c "/opt/homebrew/bin/mvn clean" (self time 1s)
```