LAB 1: Setting Up Jenkins CI/CD Pipeline

TO CREATE A SIMPLE CI/CD PIPELINE IN JENKINS

STEP 1: Install Git Bash

- Download and install Git Bash from the official Git website: Git Website.

STEP 2: Set Up Git Bash

- Open Git Bash and configure your Git username and email:

git config --global user.name "Your Name"

git config --global user.email "you@example.com"

STEP 3: Initialize a Git Repository

- Navigate to your project directory:

cd /path/to/your/project

- Initialize the Git repository:

git init

```
ibmtr@DivyaMurugan MINGW64 ~/Downloads/Arsha
$ git init
Initialized empty Git repository in C:/Users/ibmtr/Downloads/Arsha/.git/
```

STEP 4: Add Files to Git

- Add all files to the staging area:

git add .

STEP 5: Commit the Files

- Commit the files with a message:

git commit -m "Initial commit"

```
ibmtr@DivyaMurugan MINGW64 ~/Downloads/Arsha (master)
$ git commit -m "Initial commit"
[master (root-commit) 917c9ca] Initial commit
 117 files changed, 79690 insertions(+)
```

STEP 6: Create a Repository on GitHub

- Go to GitHub, sign in, and create a new repository.
  (Leave the repository blank without adding any README or .gitignore files.)

STEP 7: Add GitHub Repository as Remote

- Add the GitHub repository as a remote origin:

git remote add origin https://github.com/username/repository.git

```
ibmtr@DivyaMurugan MINGW64 ~/Downloads/Arsha (master)
$ git remote add origin https://github.com/DivyaMuruganIBM/Arshasample.git
```

STEP 8: Push the Project to GitHub

- Push the local project to the GitHub repository:

git push -u origin master

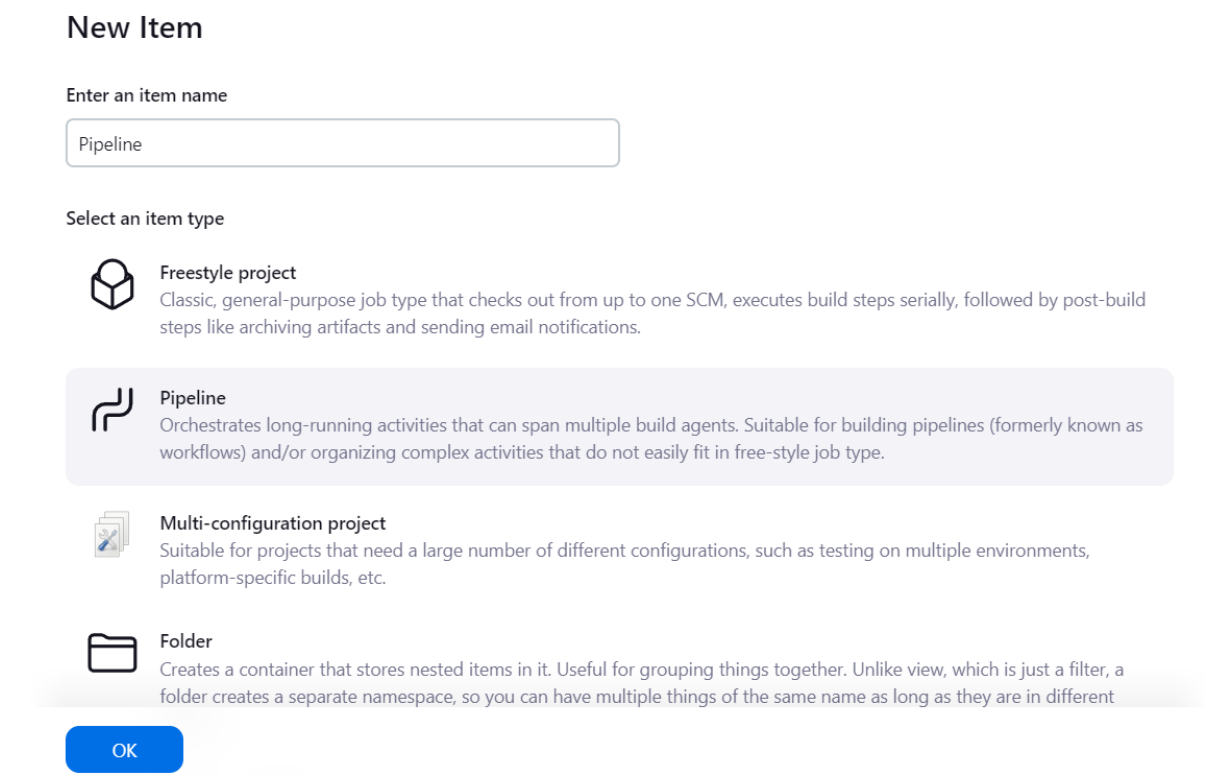
```
ibmtr@DivyaMurugan MINGW64 ~/Downloads/Arsha (master)
$ git push -u origin master
Enumerating objects: 145, done.
Counting objects: 100% (145/145), done.
Delta compression using up to 12 threads
Compressing objects: 100% (140/140), done.
Writing objects: 100% (145/145), 2.65 MiB | 267.00 KiB/s, done.
Total 145 (delta 34), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (34/34), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/DivyaMuruganIBM/Arshasample/pull/new/master
remote:
To https://github.com/DivyaMuruganIBM/Arshasample.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

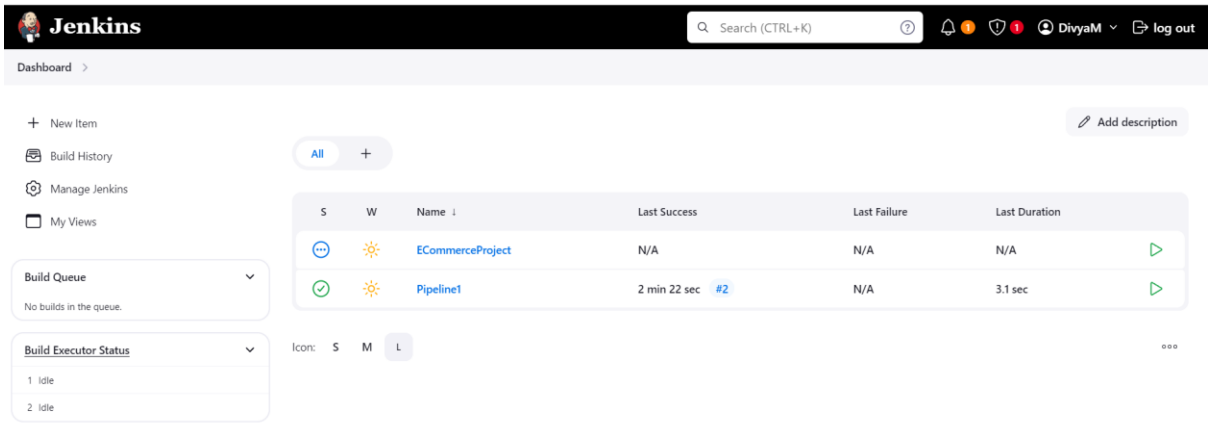
STEP 9: Install Jenkins Plugins (if not already installed):

- Go to Jenkins Dashboard → Manage Jenkins → Manage Plugins.
- In the Available tab, search for and install:
  - **Pipeline**
  - **Git** (if using Git as your version control system)

STEP 10: Create a Pipeline Job:

1. In Jenkins, click on New Item in the Jenkins dashboard.
2. Enter a name for your job, select **Pipeline** as the project type, and click OK.





---

3. Under the Pipeline section, choose **Pipeline script**.

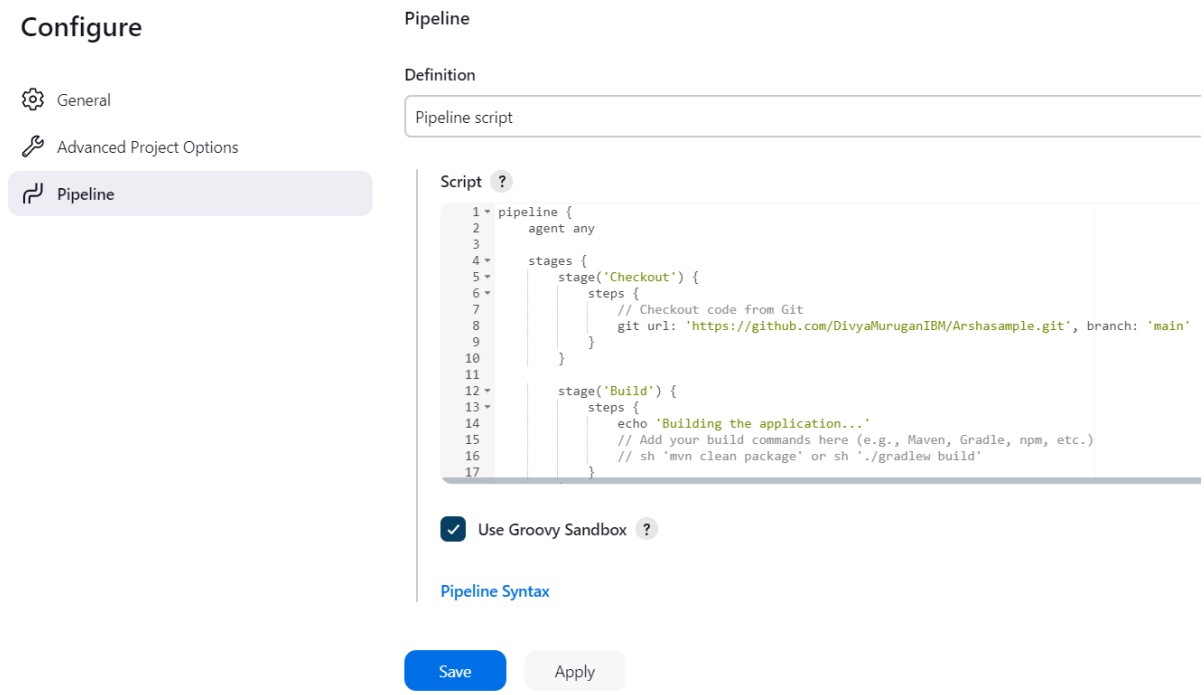
STEP 11: Write the Jenkins Pipeline Script:

You can write a simple declarative pipeline that showcases multiple stages like build, test, and deploy.

Here's a basic example of a Jenkins pipeline script:

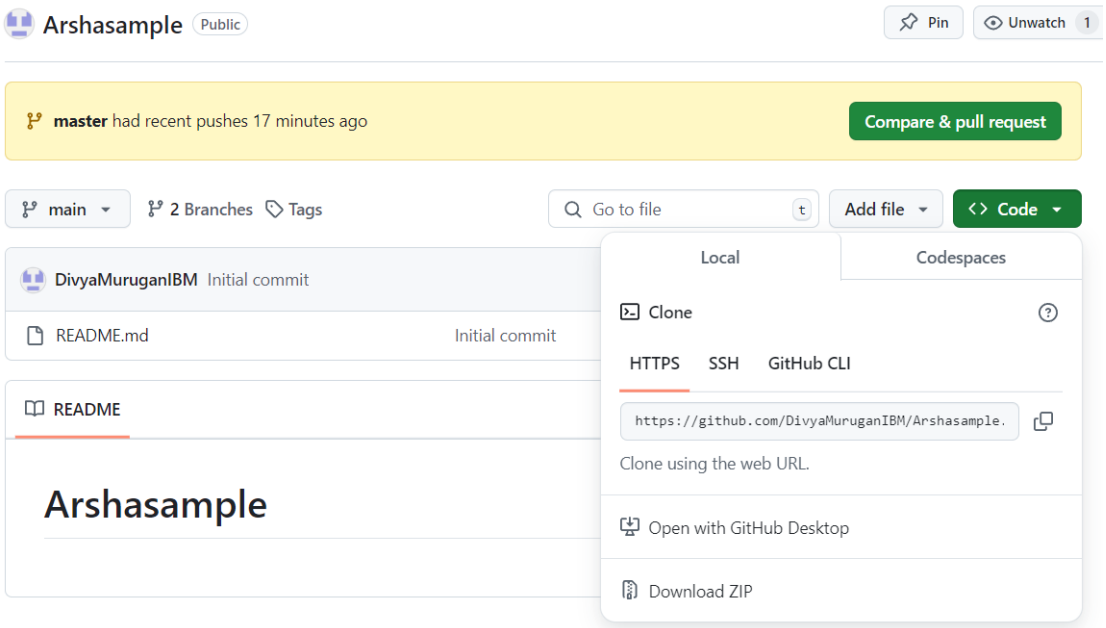
```
pipeline {
    agent any
    stages {
        stage('Checkout') {
            steps {
                // Checkout code from Git
                git url: 'https://github.com/your-repo.git', branch: 'main'
            }
        }
        stage('Build') {
            steps {
                echo 'Building the application...'
                // Add your build commands here (e.g., Maven, Gradle, npm, etc.)
                // sh 'mvn clean package' or sh './gradlew build'
            }
        }
        stage('Test') {
            steps {
                echo 'Running tests...'
                // Add your test commands here (e.g., unit tests, integration tests)
                // sh 'mvn test' or sh './gradlew test'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying the application...'
                // Add your deploy commands (e.g., deployment scripts, Docker, etc.)
                // sh 'docker-compose up -d' or sh 'kubectl apply -f deployment.yaml'
            }
        }
    }
```

```
    post {
        success {
            echo 'Pipeline succeeded!'
        }
        failure {
            echo 'Pipeline failed!'
        }
    }
}
```



STEP 12: Configure Git Repository:

Replace the placeholder Git repository URL (https://github.com/your-repo.git) with your actual repository URL.



STEP 13: Save the Pipeline:

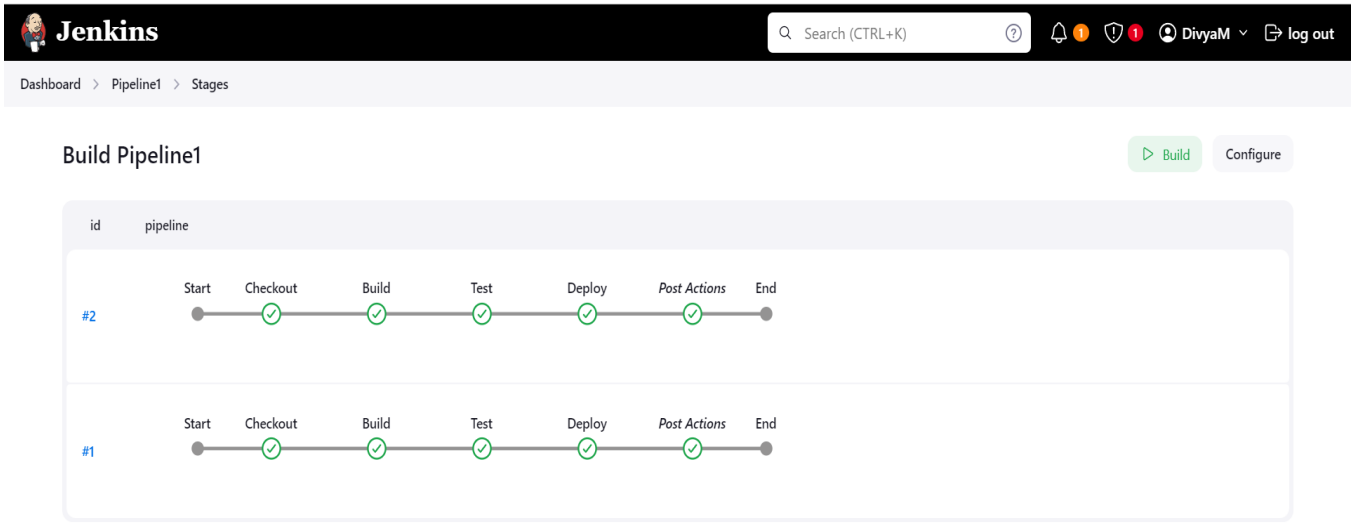
After writing the script, click Save.

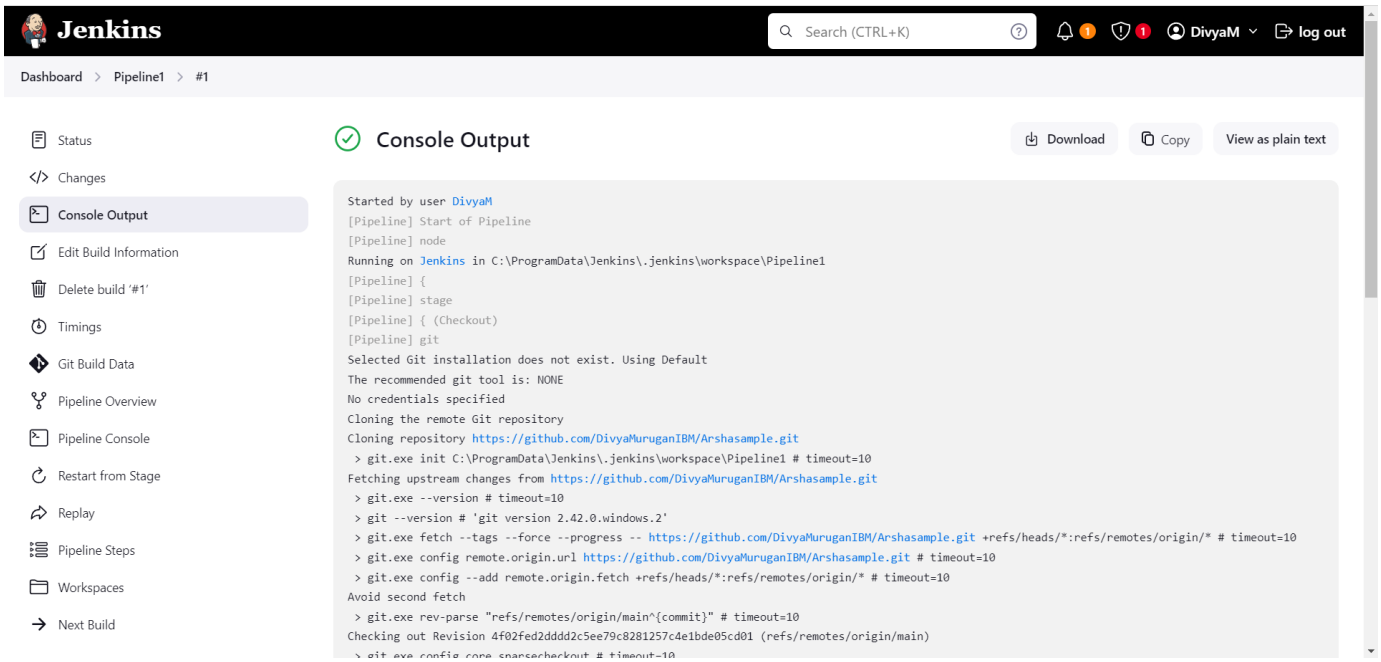
STEP 14: Run the Pipeline:

- Go back to the Jenkins dashboard and select your newly created pipeline job.
- Click Build Now to trigger the pipeline.

STEP 15: Check Pipeline Execution:

- As the pipeline runs, you'll be able to see each stage (Checkout, Build, Test, Deploy) being executed.
- You can view the progress by clicking on the Build Number in the build history and selecting Console Output.


Jenkins
Search (CTRL+K)
DivyaM
log out
Dashboard
Pipeline1
Stages
Build Pipeline1
Build
Configure
id
pipeline
#2
Start
Checkout
Build
Test
Deploy
Post Actions
End
#1
Start
Checkout
Build
Test
Deploy
Post Actions
End


Jenkins
Search (CTRL+K)
DivyaM
log out
Dashboard
Pipeline1
#1
Status
Changes
Console Output
Edit Build Information
Delete build '#1'
Timings
Git Build Data
Pipeline Overview
Pipeline Console
Restart from Stage
Replay
Pipeline Steps
Workspaces
Next Build
Console Output
Download
Copy
View as plain text
Started by user DivyaM
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\Pipeline1
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Checkout)
[Pipeline] git
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/DivyaMuruganIBM/Arshasample.git
> git.exe init C:\ProgramData\Jenkins\.jenkins\workspace\Pipeline1 # timeout=10
Fetching upstream changes from https://github.com/DivyaMuruganIBM/Arshasample.git
> git.exe --version # timeout=10
> git --version # 'git version 2.42.0.windows.2'
> git.exe fetch --tags --force --progress -- https://github.com/DivyaMuruganIBM/Arshasample.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe config remote.origin.url https://github.com/DivyaMuruganIBM/Arshasample.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 4f02fed2dddd2c5ee79c8281257c4e1bde05cd01 (refs/remotes/origin/main)