IBM Assignment 2 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 "Rameshk84"

git config --global user.email " karamesh410@gmail.com*"*

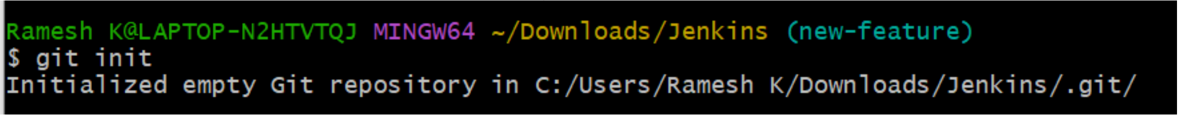
STEP 3: Initialize a Git Repository

- Navigate to your project directory:

cd /C:\Users\Ramesh K\Downloads\IBM-Jenkins-Assignment-2

- Initialize the Git repository:

git init



```
Ramesh K@LAPTOP-N2HTVTQJ MINGW64 ~/Downloads/Jenkins (new-feature)
$ git init
Initialized empty Git repository in C:/Users/Ramesh K/Downloads/Jenkins/.git/
```

STEP 4: Add Files to Git

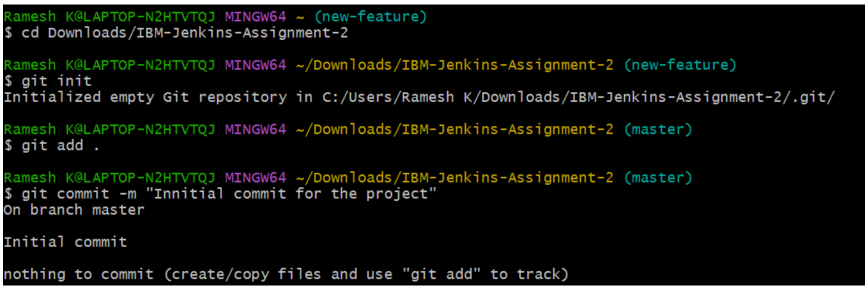
- Add all files to the staging area:

git add .

STEP 5: Commit the Files

- Commit the files with a message:

git commit -m "Initial commit"



```
Ramesh K@LAPTOP-N2HTVTQJ MINGW64 ~ (new-feature)
$ cd Downloads/IBM-Jenkins-Assignment-2

Ramesh K@LAPTOP-N2HTVTQJ MINGW64 ~/Downloads/IBM-Jenkins-Assignment-2 (new-feature)
$ git init
Initialized empty Git repository in C:/Users/Ramesh K/Downloads/IBM-Jenkins-Assignment-2/.git/

Ramesh K@LAPTOP-N2HTVTQJ MINGW64 ~/Downloads/IBM-Jenkins-Assignment-2 (master)
$ git add .

Ramesh K@LAPTOP-N2HTVTQJ MINGW64 ~/Downloads/IBM-Jenkins-Assignment-2 (master)
$ git commit -m "Innitial commit for the project"
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
```

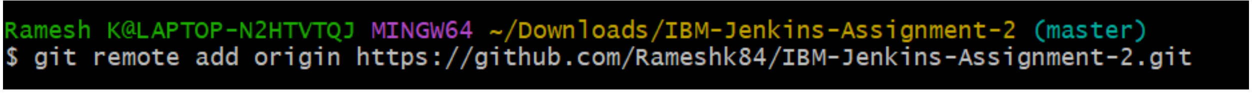
STEP 6: Create a Repository on GitHub

- 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

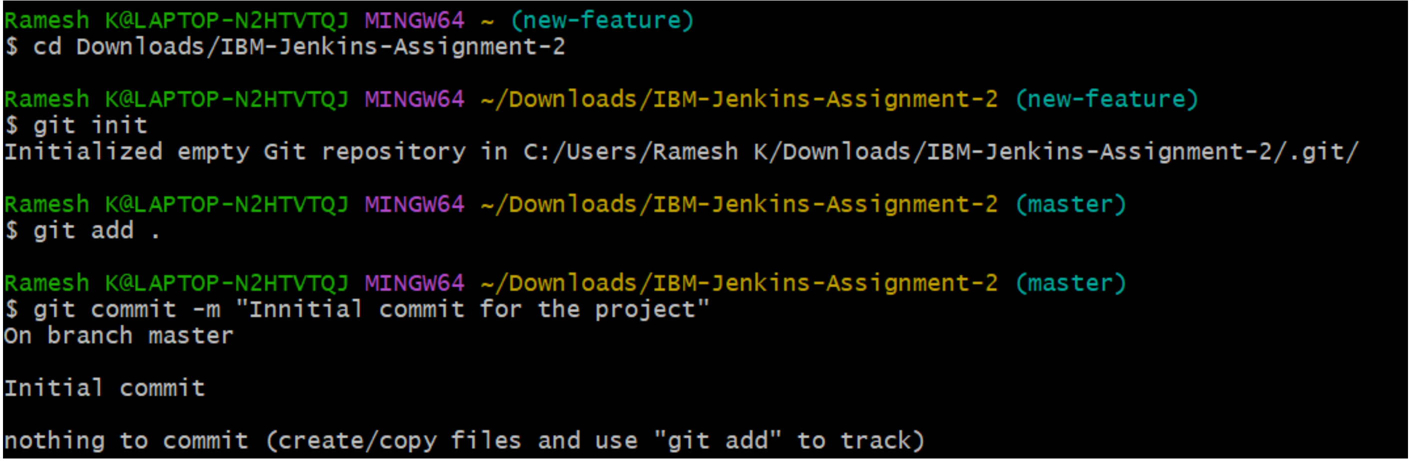


```
Ramesh K@LAPTOP-N2HTVTQJ MINGW64 ~/Downloads/IBM-Jenkins-Assignment-2 (master)
$ git remote add origin https://github.com/Rameshk84/IBM-Jenkins-Assignment-2.git
```

STEP 8: Push the Project to GitHub

- Push the local project to the GitHub repository:

git push -u origin master



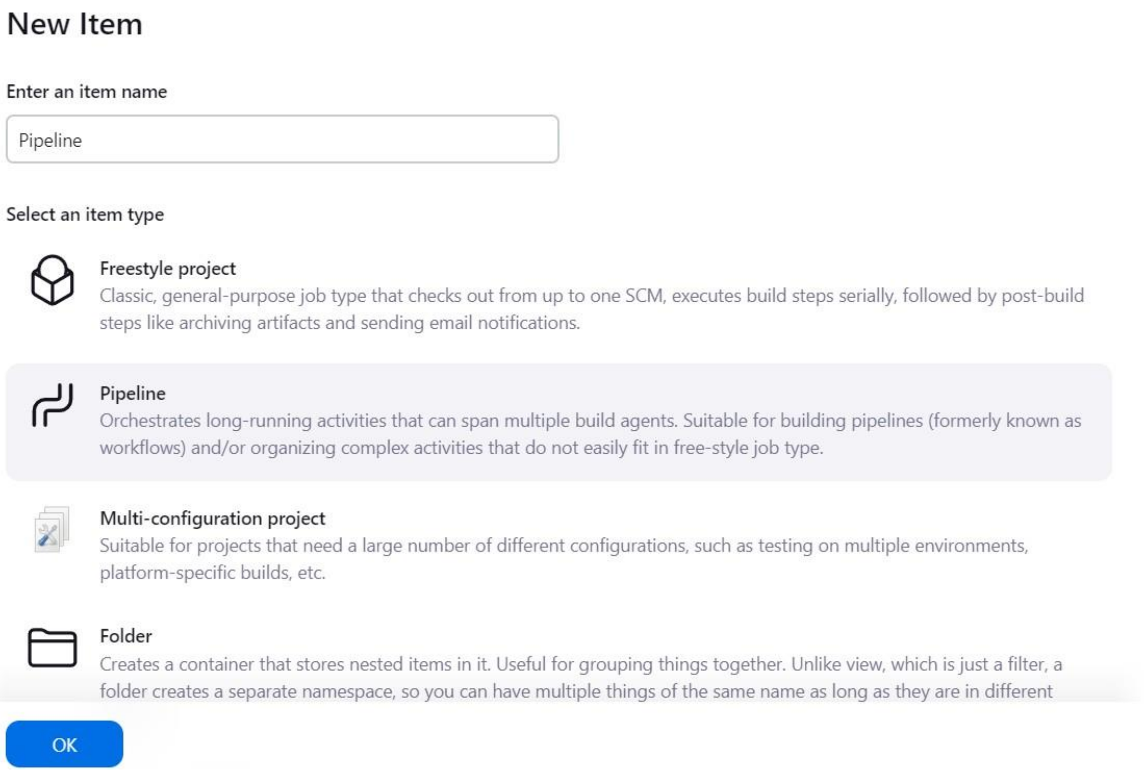
```
Ramesh K@LAPTOP-N2HTVTQJ MINGW64 ~ (new-feature)
$ cd Downloads/IBM-Jenkins-Assignment-2

Ramesh K@LAPTOP-N2HTVTQJ MINGW64 ~/Downloads/IBM-Jenkins-Assignment-2 (new-feature)
$ git init
Initialized empty Git repository in C:/Users/Ramesh K/Downloads/IBM-Jenkins-Assignment-2/.git/

Ramesh K@LAPTOP-N2HTVTQJ MINGW64 ~/Downloads/IBM-Jenkins-Assignment-2 (master)
$ git add .

Ramesh K@LAPTOP-N2HTVTQJ MINGW64 ~/Downloads/IBM-Jenkins-Assignment-2 (master)
$ git commit -m "Innitial commit for the project"
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
```

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.



New Item

Enter an item name

Pipeline

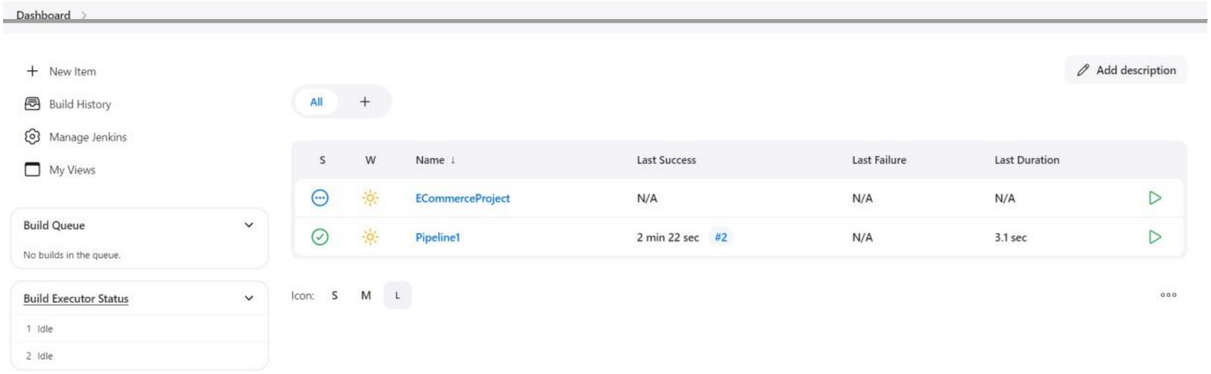
Select an item type

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



Build Queue
No builds in the queue.

Build Executor Status
1 Idle
2 Idle

S	W	Name ↓	Last Success	Last Failure	Last Duration
		ECommerceProject	N/A	N/A	N/A
		Pipeline1	2 min 22 sec #2	N/A	3.1 sec

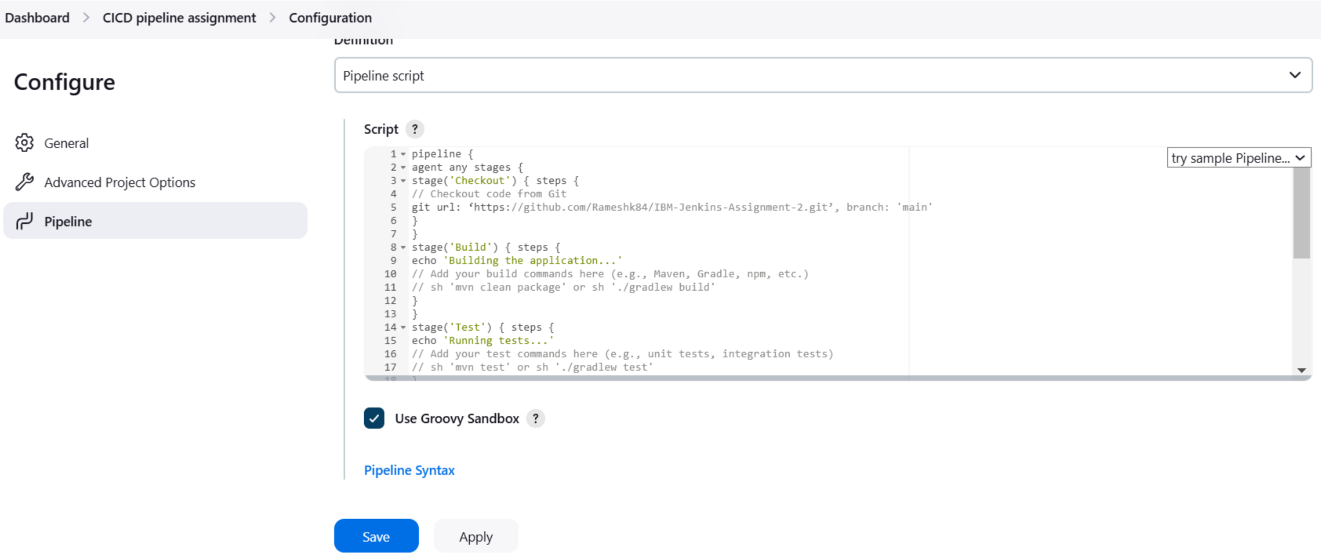
3. Under the Pipeline section, choose **Pipeline script**.

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/Rameshk84/IBM-Jenkins-Assignment-2.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!'
        }
    }
}
```



Definition

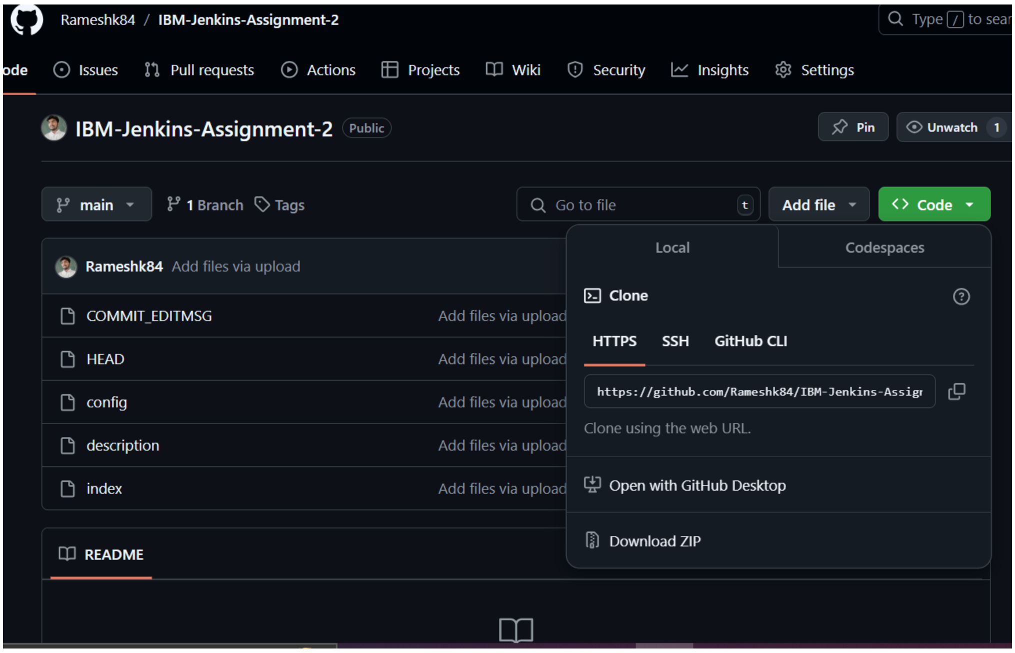
Pipeline script

```
pipeline {
agent any stages {
stage('Checkout') { steps {
// Checkout code from Git
git url: ‘https://github.com/Rameshk84/IBM-Jenkins-Assignment-2.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'
```

☑ Use Groovy Sandbox

STEP 12: Configure Git Repository:

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



Rameshk84 Add files via upload

COMMIT_EDITMSG	Add files via upload
HEAD	Add files via upload
config	Add files via upload
description	Add files via upload
index	Add files via upload

```
https://github.com/Rameshk84/IBM-Jenkins-Assign
```

Clone using the web URL.

STEP 13: Save the Pipeline:

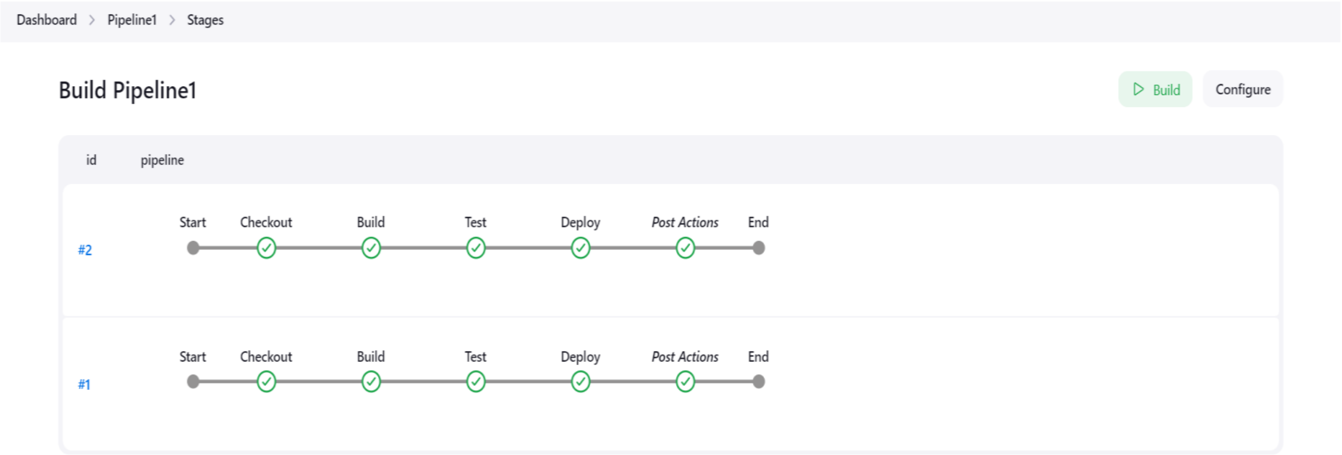
After writing the script, click Save.

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.

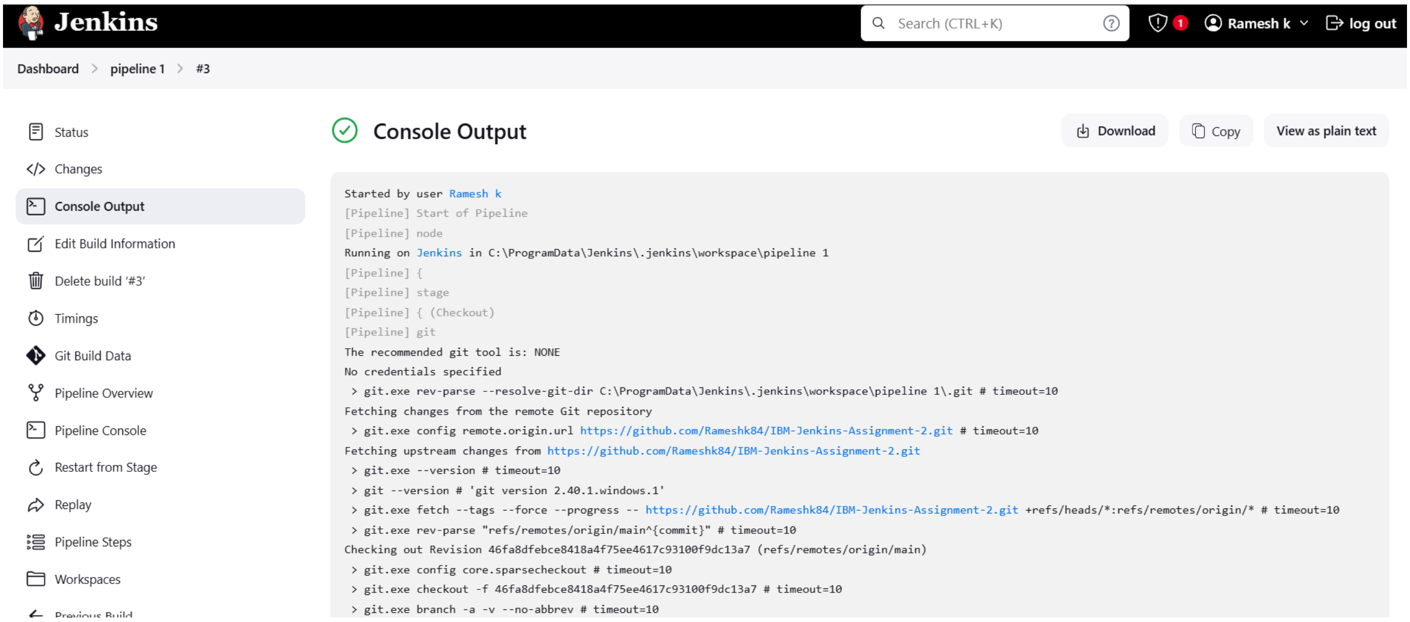
STEP 15: Check Pipeline Execution:

- 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.



Build Pipeline1

id	pipeline
#2	Start ● — Checkout ✓ — Build ✓ — Test ✓ — Deploy ✓ — *Post Actions* ✓ — End ●
#1	Start ● — Checkout ✓ — Build ✓ — Test ✓ — Deploy ✓ — *Post Actions* ✓ — End ●



Console Output

```
Started by user Ramesh k
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\pipeline 1
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Checkout)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\pipeline 1\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/Rameshk84/IBM-Jenkins-Assignment-2.git # timeout=10
Fetching upstream changes from https://github.com/Rameshk84/IBM-Jenkins-Assignment-2.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.40.1.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/Rameshk84/IBM-Jenkins-Assignment-2.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10
Checking out Revision 46fa8dfebce8418a4f75ee4617c93100f9dc13a7 (refs/remotes/origin/main)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 46fa8dfebce8418a4f75ee4617c93100f9dc13a7 # timeout=10
 > git.exe branch -a -v --no-abbrev # timeout=10
```