

**Practice for Lesson 5:  
Understanding of Parallel  
Jenkins Jobs and Jenkins  
Slave on AWS**

## Practices for Lesson 5

---

### Overview

In these practices, you will learn how to Build and Deploy an Application to Webserver using Jenkins Pipeline. Further create a parallel Agent Pipeline Job on Jenkins.

.

## Practice 5-3: Create a DevOps Pipeline Job on Jenkins

---

### Overview

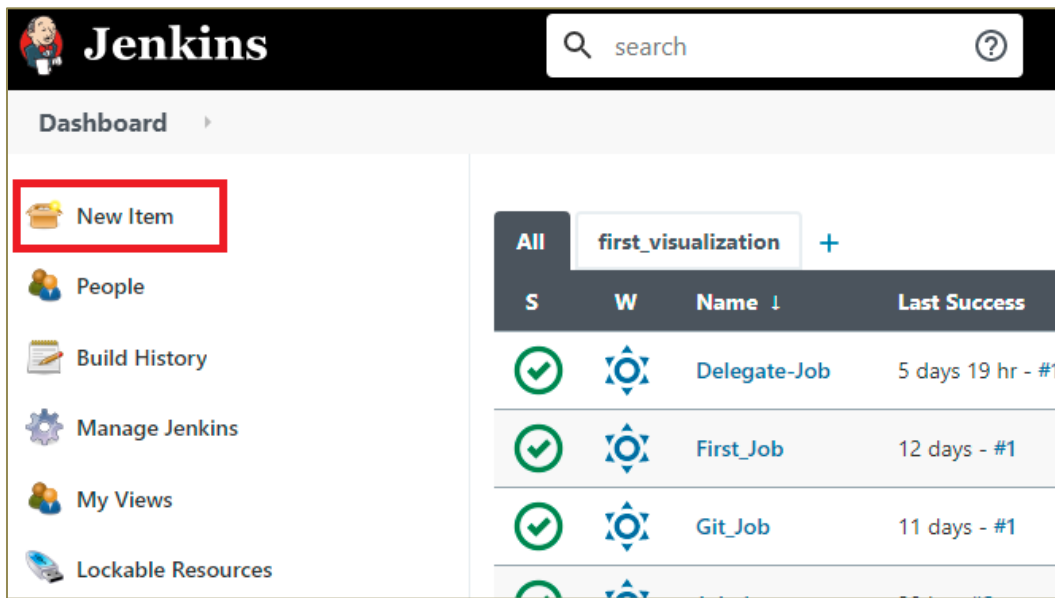
In this practice, you will learn how to create the DevOps Pipeline Job on Jenkins instance using as sample example.

### Assumptions

You should have completed the Practice of Lesson 5-1.

### Tasks

1. Create a DevOps Pipeline Job on Jenkins.
  - a. In the Jenkins Dashboard, navigate to main menu and select **New Item** to create a Parallel Agent Pipeline as shown below.





- b. Provide the **name** for the DevOps Pipeline, further select **Pipeline** and click **OK** as shown below.


**Enter an item name**


DevOps\_Pipeline

» Required field


**Freestyle project**  
 This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.


**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.


**External job**  
 This job allows you to record the execution of a process run outside Jenkins, even on a remote system. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

OK

- c. Navigate to **Pipeline**, select **Pipeline script from SCM** under **Definition** as shown below and select **Git** under **SCM** (Source Code Management).

General Build Triggers Advanced Project Options **Pipeline**

**Pipeline**

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

Please enter Git repository.

Credentials

- none - Add


Advanced...




Add Repository


Save Apply




- d. The GitHub link is provided below which is in the public domain, consisting of the Jenkins file.

**Link:** [oralabuser/PipelineScript \(github.com\)](https://github.com/oralabuser/PipelineScript)




[Pulls](#) [Issues](#) [Marketplace](#) [Explore](#)   


 [oralabuser](#) / [PipelineScript](#)


 Unwatch 1  Star 0  Fork 0







[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) ...

 main


[Go to file](#) [Add file](#) [Code](#)

[About](#) 

 [oralabuser](#) Update Jenkinsfile ... 32 minutes ago 7

 Build.sh	Create Build.sh	2 months ago
 Deploy.sh	Create Deploy.sh	2 months ago
 Jenkinsfile	Update Jenkinsfile	32 minutes ago
 Quality.sh	Create Quality.sh	2 months ago
 README.md	Initial commit	2 months ago
 Unit.sh	Create Unit.sh	2 months ago

No description, website, or topics provided.

 [Readme](#)

---

### Releases

No releases published  
[Create a new release](#)

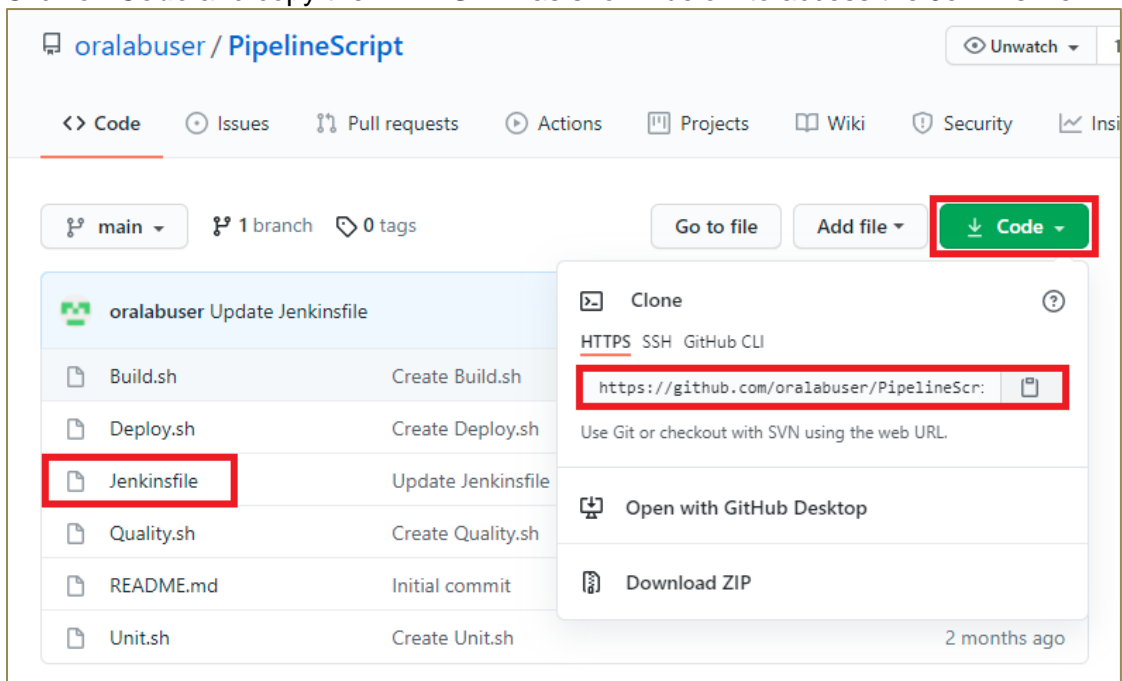
---

### Packages

The screenshot shows the GitHub interface for the file `PipelineScript/Jenkinsfile` in the `oralabuser` repository. The file is 38 lines long and 570 bytes. The code is a Jenkins pipeline script with the following structure:

```
1 pipeline {
2   agent none
3   stages {
4     stage('Non-Parallel Stage') {
5       agent {
6         label "master"
7       }
8       steps {
9         echo "This stage will be executed first";
10      }
11    }
12    stage('Run Tests') {
13      steps {
14        echo "Building the checked-out project!";
15      }
16    }
17    stage('Unit-Test') {
18      parallel {
19        stage('Test on Agent') {
20          agent {
21            label "AWS_instance_Node"
22          }
23          steps {
24            echo "Task1 on Agent";
25          }
26        }
27        stage('Test On master') {
28          agent {
29            label "master"
30          }
31          steps {
32            echo "Task1 on Master";
33          }
34        }
35      }
36    }
37  }
38 }
```

e. Click on **Code** and copy the **HTTPS** link as shown below to access the Jenkins file.



- f. Paste the HTTPS GitHub link copied in the **Repository URL** as shown below.

The screenshot shows the Jenkins Pipeline configuration interface. The 'Pipeline' tab is active. In the 'Definition' section, 'Pipeline script from SCM' is selected. Under the 'SCM' section, 'Git' is chosen. In the 'Repositories' section, the 'Repository URL' is set to 'https://github.com/oralabuser/PipelineScript.git'. The 'Credentials' dropdown is set to '- none -'. There are 'Advanced...' and 'Add Repository' buttons at the bottom right.

- g. Provide the **Branch** as **main** and in **Script Path** provide the File name to access the code as shown below. Click **Apply** and **Save**.

General Build Triggers Advanced Project Options **Pipeline**

**Branches to build** ?

Branch Specifier (blank for 'any') X ?

**\*/main**

Add Branch

**Repository browser** ?

(Auto) v

**Additional Behaviours**

Add v

**Script Path** ?

**Jenkinsfile**

☒ Lightweight checkout ?

Pipeline Syntax

**Save** **Apply**

- h. Connect to **Agent node** server and install **Git** in it as shown below.

```
[ec2-user@ip-172-31-36-198 ~]$ sudo yum install -y git
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core | 3.7 kB 00:00
Resolving Dependencies
--> Running transaction check
--> Package git.x86_64 0:2.23.4-1.amzn2.0.1 will be installed
--> Processing Dependency: perl-Git = 2.23.4-1.amzn2.0.1 for package: git-2.23.4-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core-doc = 2.23.4-1.amzn2.0.1 for package: git-2.23.4-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core = 2.23.4-1.amzn2.0.1 for package: git-2.23.4-1.amzn2.0.1.x86_64
--> Processing Dependency: emacsfilesystem >= 25.2 for package: git-2.23.4-1.amzn2.0.1.x86_64
```

- i. Select **Build Now** from the menu to execute the Pipeline and **view** the **stages** of **DevOps\_pipeline** getting executed and click on the link under **Build History** as shown below.



Dashboard > DevOps\_Pipeline >

Back to Dashboard
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
Build History trend
find
#1 Jun 18, 2021 11:09 AM

## Pipeline DevOps\_Pipeline

add description
Disable Project

Recent Changes

### Stage View

	Non-Parallel Stage	Run Tests	Unit-Test	Test on Agent	Test On master
Average stage times: (Average full run time: ~6s)	710ms	41ms	48ms	1min 18s	556ms
Jun 18 16:39 No Changes	577ms	37ms	44ms	4s	546ms

- j. Click on **Console Output** to view the execution of the **DevOps\_Pipeline** Job as shown below.

Dashboard > DevOps\_Pipeline > #5

Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#5'
Git Build Data
Restart from Stage
Replay
Pipeline Steps
Workspaces
Previous Build

## Console Output

Started by user oralabuser  
Obtained Jenkinsfile from git <https://github.com/oralabuser/PipelineScript.git>  
Running in Durability level: MAX\_SURVIVABILITY  
[Pipeline] Start of Pipeline  
[Pipeline] stage  
[Pipeline] { (Non-Parallel Stage)  
[Pipeline] node  
Running on Jenkins in /var/lib/jenkins/workspace/DevOps\_Pipeline  
[Pipeline] {  
[Pipeline] checkout  
Selected Git installation does not exist. Using Default  
The recommended git tool is: NONE  
No credentials specified  
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/DevOps\_Pipeline/.git # timeout=10  
Fetching changes from the remote Git repository  
> git config remote.origin.url https://github.com/oralabuser/PipelineScript.git # timeout=10  
Fetching upstream changes from https://github.com/oralabuser/PipelineScript.git  
> git --version # timeout=10  
> git --version # 'git version 2.23.4'  
> git fetch --tags --force --progress -- https://github.com/oralabuser/PipelineScript.git +refs/heads/\*:refs/remotes/origin/\* # timeout=10  
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10  
Checking out Revision 6af9247de1fd91ffc3ddfdcd771d238c5a824a25 (refs/remotes/origin/main)  
> git config core.sparsecheckout # timeout=10  
> git checkout -f 6af9247de1fd91ffc3ddfdcd771d238c5a824a25 # timeout=10  
Commit message: "Update Jenkinsfile"  
> git rev-list --no-walk 6af9247de1fd91ffc3ddfdcd771d238c5a824a25 # timeout=10  
[Pipeline] withEnv

[Test On master]	<pre> [Pipeline] { [Pipeline] echo Task1 on Master [Pipeline] } [Pipeline] // withEnv [Pipeline] } [Pipeline] // node [Pipeline] } [Pipeline] // stage [Pipeline] } </pre>
[Test on Agent]	<pre> Avoid second fetch Checking out Revision 6af9247de1fd91ffcf3ddfd771d238c5a824a25 (refs/remotes/origin/main) Cloning repository https://github.com/oralabuser/PipelineScript.git &gt; git init /home/ec2-user/jenkins/workspace/DevOps_Pipeline # timeout=10 Fetching upstream changes from https://github.com/oralabuser/PipelineScript.git &gt; git --version # timeout=10 &gt; git --version # 'git version 2.23.4' &gt; git fetch --tags --force --progress -- https://github.com/oralabuser/PipelineScript.git +refs/heads/*:refs/remotes/origin/* # timeout=10 &gt; git config remote.origin.url https://github.com/oralabuser/PipelineScript.git # timeout=10 &gt; git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10 &gt; git rev-parse refs/remotes/origin/main^{commit} # timeout=10 &gt; git config core.sparsecheckout # timeout=10 &gt; git checkout -f 6af9247de1fd91ffcf3ddfd771d238c5a824a25 # timeout=10 Commit message: "Update Jenkinsfile" [Pipeline] withEnv [Pipeline] { [Pipeline] echo Task1 on Agent [Pipeline] } [Pipeline] // withEnv [Pipeline] } [Pipeline] // node [Pipeline] } [Pipeline] // stage [Pipeline] } [Pipeline] // parallel [Pipeline] } [Pipeline] // stage [Pipeline] End of Pipeline Finished: SUCCESS </pre>

2. Close the terminal, Logout from the AWS Management console and Jenkins Dashboard.