

## Devops Capstone Project 1

You have been hired as a Sr. DevOps Engineer in Abode Software. They want to implement DevOps Lifecycle in their company. You have been asked to implement this lifecycle as fast as possible. Abode Software is a product-based company and their product is available on this GitHub link.

<https://github.com/hshar/website.git>

Following are the specifications of the lifecycle:

1. Install the necessary software on the machines using a configuration management tool
2. Git workflow has to be implemented
3. CodeBuild should automatically be triggered once a commit is made to master branch or develop branch.
  - a. If a commit is made to master branch, test and push to prod
  - b. If a commit is made to develop branch, just test the product, do not push to prod
4. The code should be containerized with the help of a Dockerfile. The Dockerfile should be built every time there is a push to GitHub. Use the following pre-built container for your application: hshar/webapp  
The code should reside in '/var/www/html'
5. The above tasks should be defined in a Jenkins Pipeline with the following jobs:
  - a. Job1 : build
  - b. Job2 : test
  - c. Job3 : prod

Problem (1) Solution: Install the necessary software on the machines using a configuration management tool

A. Create the Three Instances Named as Master, Slave1 & Slave2

EC2 > Instances > Launch an instance

### Name and tags

Name: Ansible

**Application and OS Images (Amazon Machine Image)**

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.

Search our full catalog including 1000s of application and OS images

Recent OS Images: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux

Browse more AMIs: Including AMIs from AWS, Marketplace and the Community

**Amazon Machine Image (AMI)**

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type  
ami-0866a3c8686eaeeba (64-bit (x86)) / ami-0325498274077fac5 (64-bit (Arm))

Free tier eligible

**Summary**

Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more  
ami-0866a3c8686eaeeba

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the future) in the free tier eligible region.

Cancel, Launch instance, Preview code

### Instance type

Instance type: t2.medium

Family: t2 - 2 vCPUs - 4 GiB Memory - Current generation: true

On-Demand Ubuntu Pro base pricing: 0.0499 USD per Hour

On-Demand Linux base pricing: 0.0464 USD per Hour

On-Demand RHEL base pricing: 0.0752 USD per Hour

On-Demand Windows base pricing: 0.0644 USD per Hour

On-Demand SUSE base pricing: 0.1464 USD per Hour

All generations, Compare instance types

**Key pair (login)**

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required: Demo-1

Create new key pair

**Summary**

Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more  
ami-0866a3c8686eaeeba

Virtual server type (instance type): t2.medium

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the future) in the free tier eligible region.

Cancel, Launch instance, Preview code

### Additional charges apply when outside of free tier allowance

#### Firewall (security groups)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group, Select existing security group

Common security groups: Practice sg-0358306caa46f41e9

Select security groups: Practice sg-0358306caa46f41e9

Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

### Configure storage

Root volume (Not encrypted): 1x 8 GiB gp3

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Add new volume

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance.

**Summary**

Number of instances: 3

When launching more than 1 instance, consider EC2 Auto Scaling

Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more  
ami-0866a3c8686eaeeba

Virtual server type (instance type): t2.medium

Firewall (security group): Practice

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the future) in the free tier eligible region.

Cancel, Launch instance, Preview code

Instances (3) Info								
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Last updated
Ansible_Master	i-0392b74f9dc400d4c	Running	t2.medium	Initializing	View alarms +	us-east-1d		less than a minute ago
Ansible_Slave1	i-04d9782c46ccc1497	Running	t2.medium	Initializing	View alarms +	us-east-1d		
Ansible_Slave2	i-057cc850bfacd6735	Running	t2.medium	Initializing	View alarms +	us-east-1d		

## B. Install “Ansible” Over the Master Machine using the “install.sh” file

Step 1: Choose the “Master” machine & click on “Connect”

create an “install.sh” file: `sudo nano install.sh`.

```
ubuntu@Master:~$ sudo nano install.sh
```

Paste these commands in the “install.sh” file.

```
sudo apt-add-repository ppa:ansible/ansible
sudo apt update
sudo apt install ansible -y
```

Run the “install.sh” file using the command: `bash install.sh`

```
ubuntu@Master:~$ bash install.sh |
```

Create an inventory file to connect the slave nodes to master

```
ubuntu@Master:~$ vi inventory|
```

Give Slave1 as Test and Slave2 as Prod

```
[Test]
172.31.0.167 ansible_ssh_private_key_file="/home/ubuntu/Demo-1.pem" ansible_ssh_common_args="-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null"
[Prod]
172.31.6.94 ansible_ssh_private_key_file="/home/ubuntu/Demo-1.pem" ansible_ssh_common_args="-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null"
```

Add your Private key in your home path to make connection

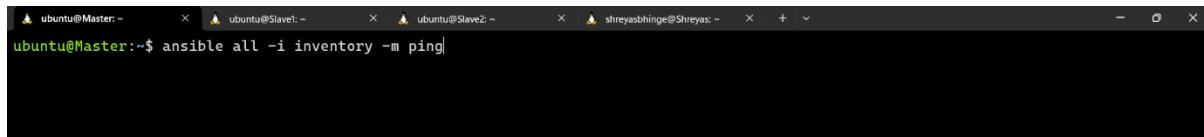
```
ubuntu@Master:~$ vi Demo-1.pem|
```

Give read permission

```
3 ubuntu@Master:~$ sudo chmod 600 Demo-1.pem |
```

Run this ansible command to ping the machines:

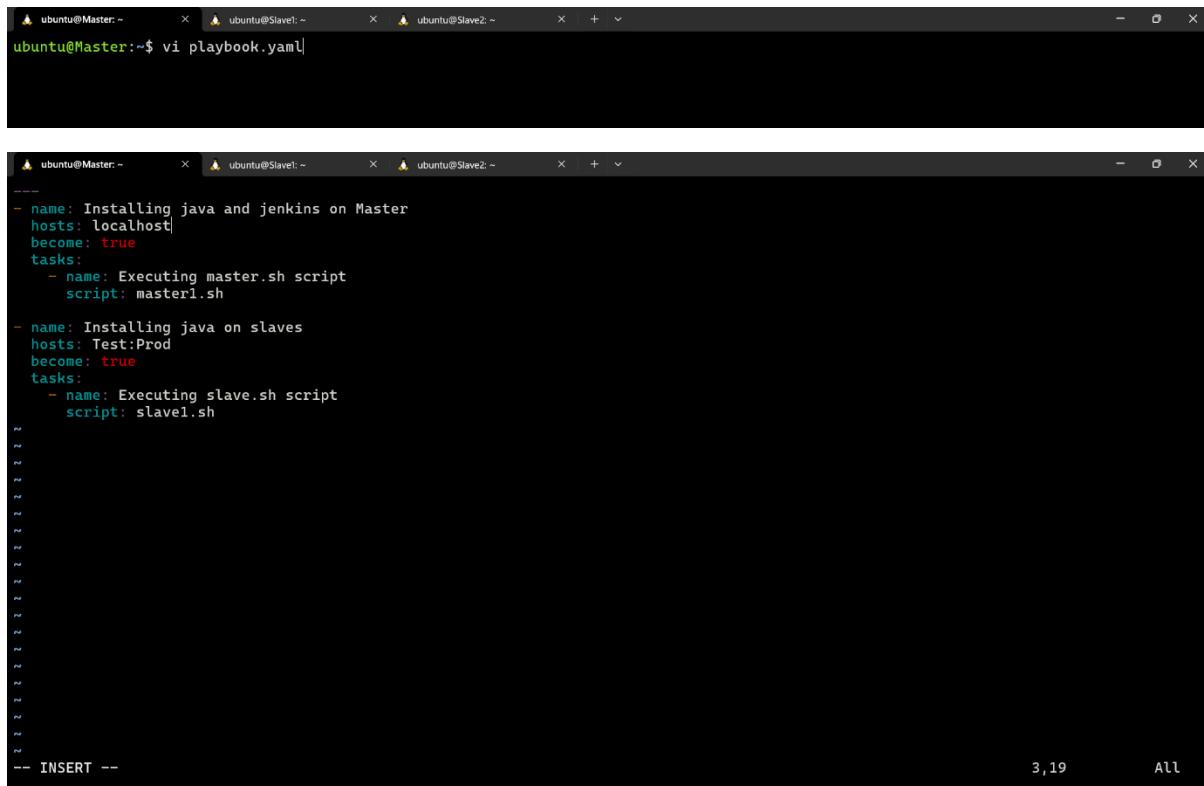
```
ansible all -i inventory -m ping
```



```
ubuntu@Master:~$ ansible all -i inventory -m ping|
```

Create a YAML file to run “**master1.sh**” & “**slave1.sh**” File to Install Much Needed Tools on Instances

Run this command to create an “**playbook.yaml**” file to execute the “**master1.sh**” & “**slave1.sh**” through “Ansible”.



```
ubuntu@Master:~$ vi playbook.yaml|
```

```
---
- name: Installing java and jenkins on Master
  hosts: localhost
  become: true
  tasks:
    - name: Executing master.sh script
      script: master1.sh

- name: Installing java on slaves
  hosts: Test:Prod
  become: true
  tasks:
    - name: Executing slave.sh script
      script: slave1.sh
```

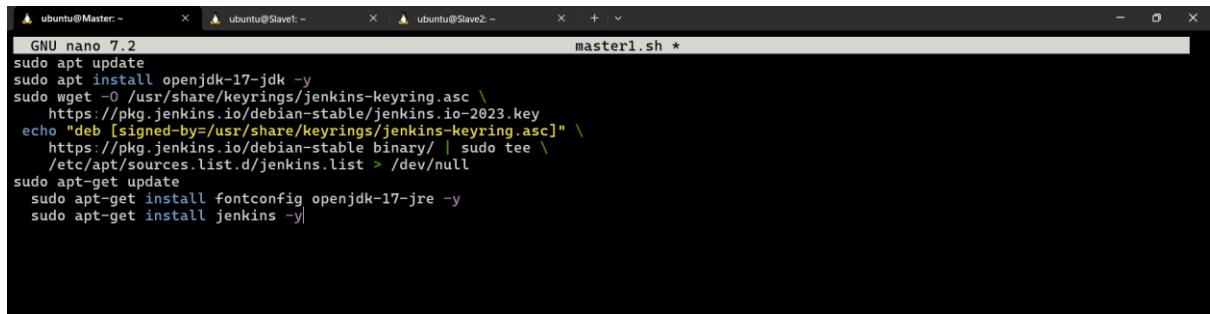
-- INSERT --

3,19 All

Create “**master1.sh**” & “**slave1.sh**” File to Install Much Needed Tools on Instances

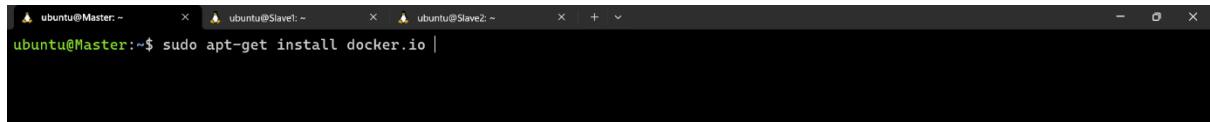
```
ubuntu@Master:~$ sudo nano master1.sh|
```

Create a “**master.sh**” file & paste the below commands here to install the tools such as “**Jenkins**”, “**Docker**”, & “**Java**”



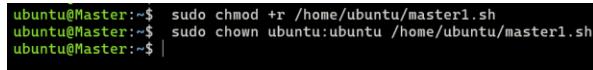
```
GNU nano 7.2                                     master1.sh *
sudo apt update
sudo apt install openjdk-17-jdk -y
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install fontconfig openjdk-17-jre -y
sudo apt-get install jenkins -y
```

Docker we will install separately



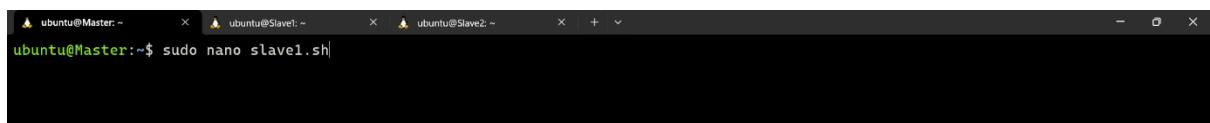
```
ubuntu@Master:~$ sudo apt-get install docker.io |
```

Give the file read permission



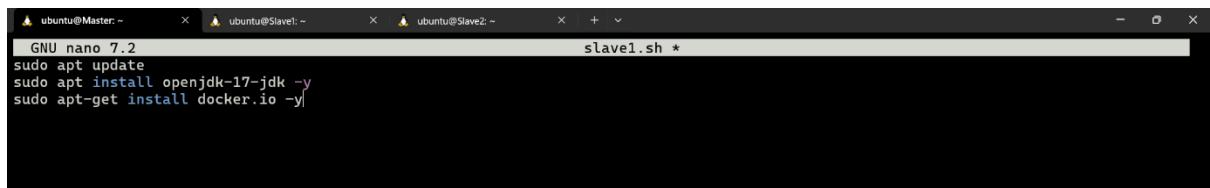
```
ubuntu@Master:~$ sudo chmod +r /home/ubuntu/master1.sh
ubuntu@Master:~$ sudo chown ubuntu:ubuntu /home/ubuntu/master1.sh
ubuntu@Master:~$ |
```

Create a “slave1.sh” file for installing the much-needed tools on the “Test” and “Prod” server.

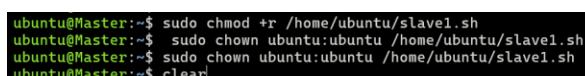


```
ubuntu@Master:~$ sudo nano slave1.sh |
```

We can add sudo install docker.io command in it

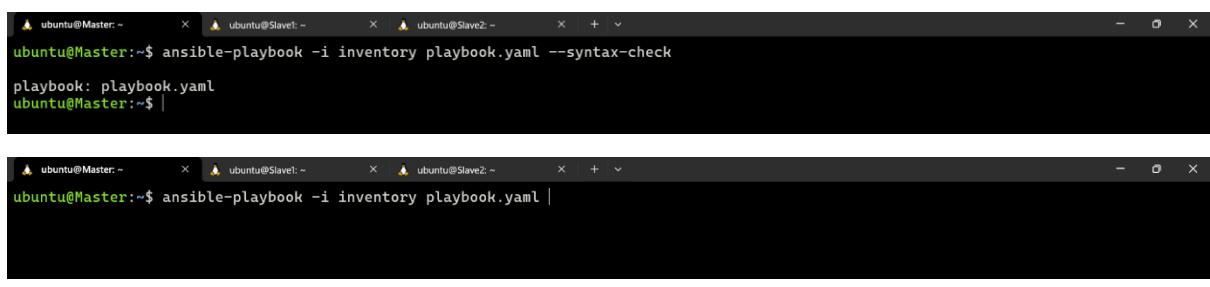


```
GNU nano 7.2                                     slave1.sh *
sudo apt update
sudo apt install openjdk-17-jdk -y
sudo apt-get install docker.io -y |
```



```
ubuntu@Master:~$ sudo chmod +r /home/ubuntu/slave1.sh
ubuntu@Master:~$ sudo chown ubuntu:ubuntu /home/ubuntu/slave1.sh
ubuntu@Master:~$ sudo chown ubuntu:ubuntu /home/ubuntu/slave1.sh
ubuntu@Master:~$ clear |
```

Execute the Playbook



```
ubuntu@Master:~$ ansible-playbook -i inventory playbook.yaml --syntax-check
playbook: playbook.yaml
ubuntu@Master:~$ |
```

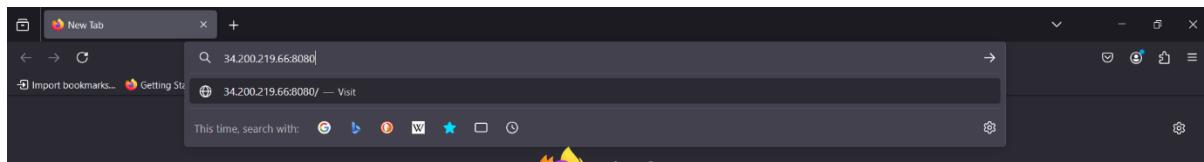
```
ubuntu@Master:~$ ansible-playbook -i inventory playbook.yaml |
```

```

ubuntu@Master:~$ ansible-playbook -i inventory playbook.yaml
PLAY [Installing java and jenkins on Master] ****
TASK [Gathering Facts] ****
ok: [localhost]
TASK [Executing master.sh script] ****
changed: [localhost]
PLAY [Installing java on slaves] ****
TASK [Gathering Facts] ****
[WARNING]: Platform linux on host 172.31.6.94 is using the discovered Python interpreter at /usr/bin/python3.12, but future
installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.6.94]
[WARNING]: Platform linux on host 172.31.0.167 is using the discovered Python interpreter at /usr/bin/python3.12, but future
installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.0.167]
TASK [Executing slave.sh script] ****
changed: [172.31.6.94]
changed: [172.31.0.167]
PLAY RECAP ****
172.31.0.167 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.6.94  : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
localhost     : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
ubuntu@Master:~$ |

```

## Setup Jenkins Dashboard Over “Master” Server



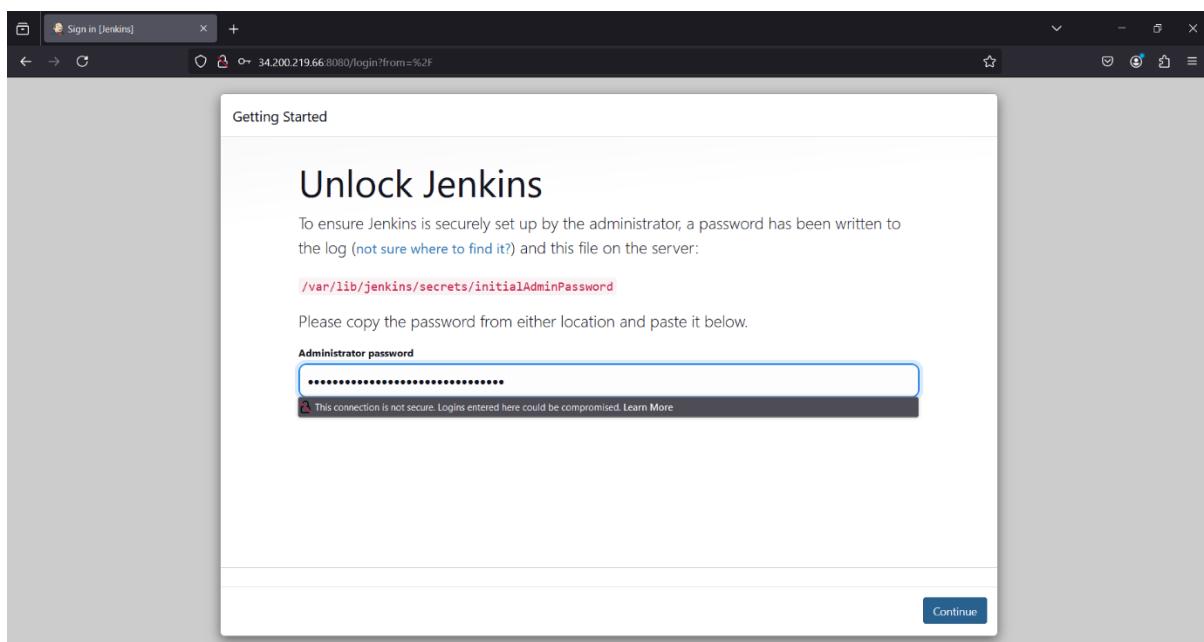
**Copy this token from here.**

```

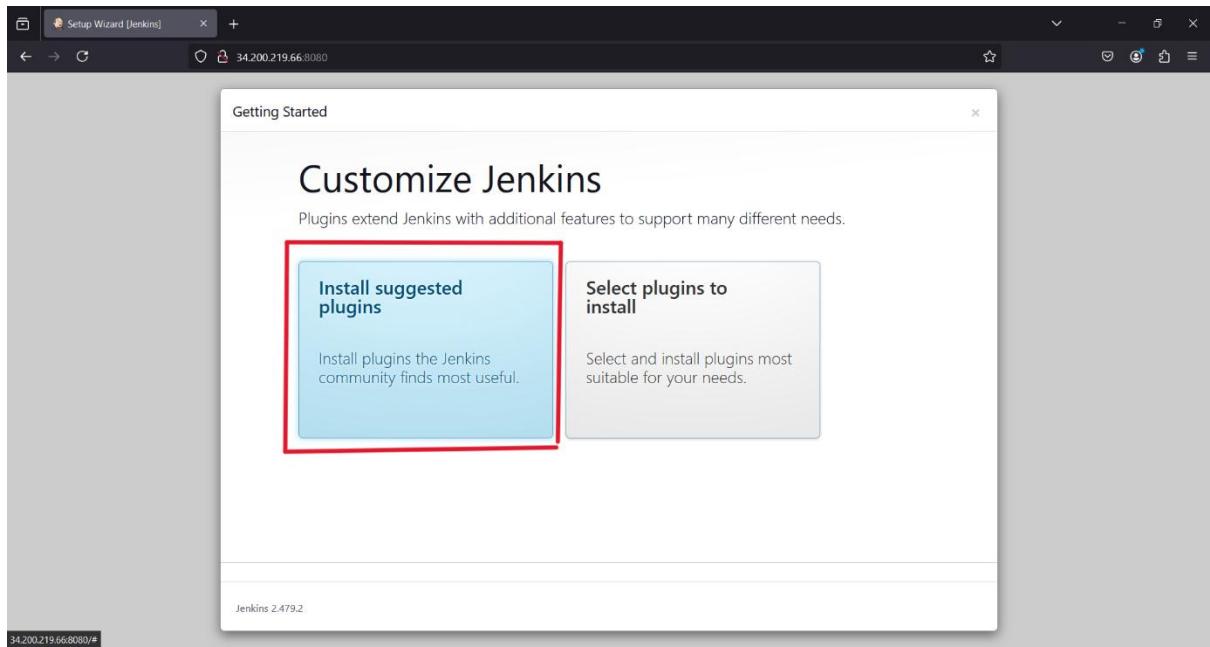
ubuntu@Master:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
f29fd8768ad243bd9edec3308d7b8f69
ubuntu@Master:~$ |

```

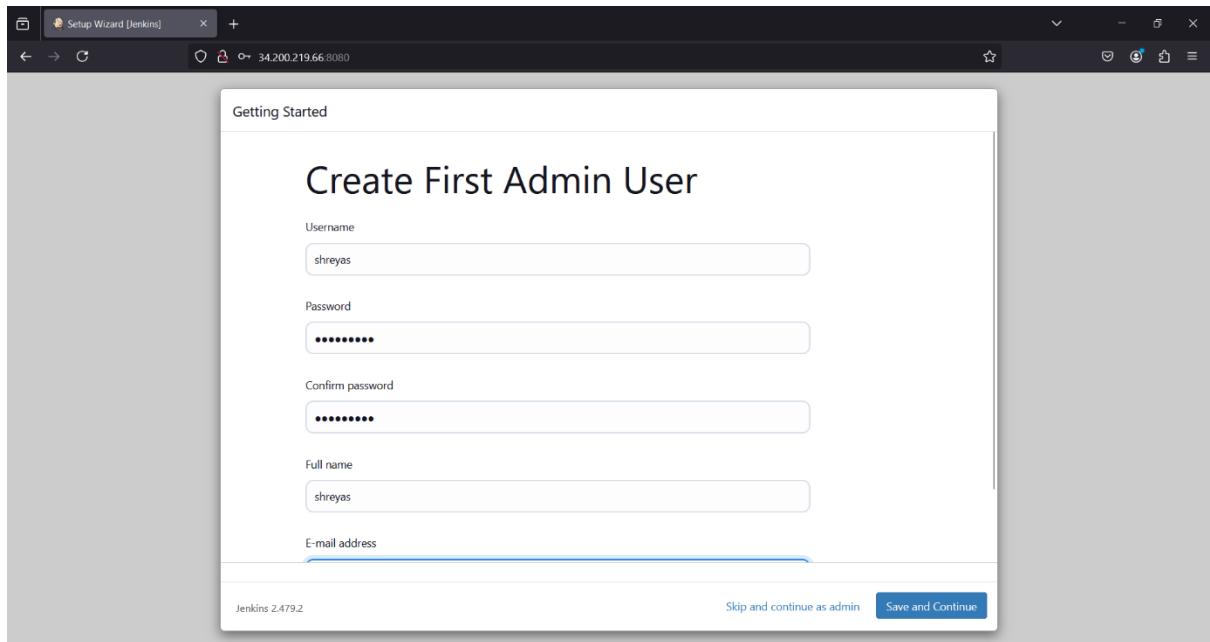
**Paste the token in “Dashboard” & click on “Continue”.**



**Click on “Install Suggested Plugins”.**



**Add Info**



Getting Started

## Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.479.2      Not now      **Save and Finish**

Setup Wizard [Jenkins]    34.200.219.66:8080

Getting Started

## Jenkins is ready!

Your Jenkins setup is complete.

**Start using Jenkins**

**"Jenkins Dashboard" will be ready.**

Dashboard >

+ New item      Welcome to Jenkins!

Build History      Set up a distributed build

Manage Jenkins      Create a job

My Views

Build Queue      +

No builds in the queue.

Build Executor Status      0/2

Set up an agent

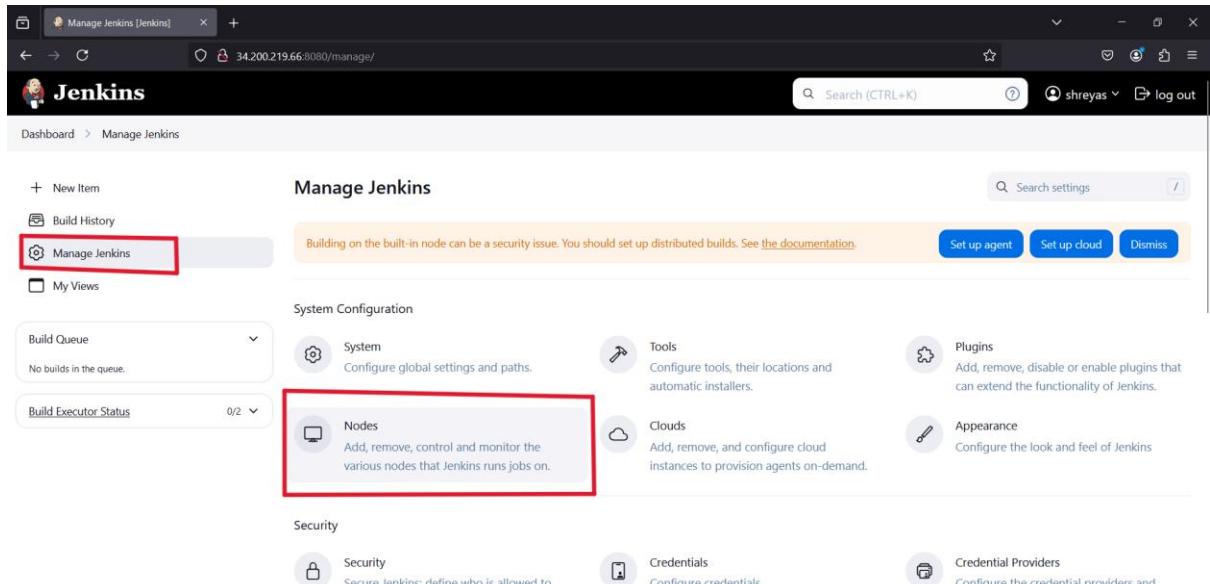
Configure a cloud

Learn more about distributed builds

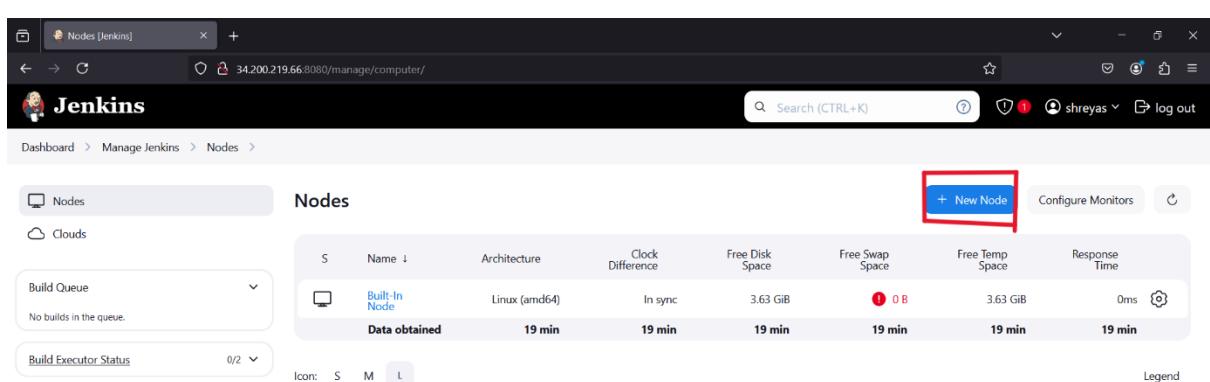
34.200.219.66:8080/manage      REST API      Jenkins 2.479.2

## Add “Test” in “Jenkins”

Step 1: Go to “Jenkins Master” & click on “New Node”.

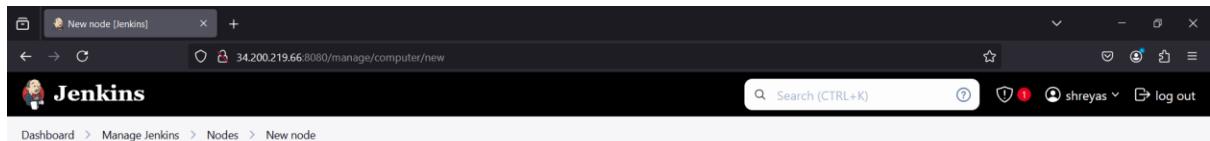


The screenshot shows the Jenkins Manage Jenkins interface. The 'Nodes' section is highlighted with a red box. The 'Manage Jenkins' tab is also highlighted with a red box. Other sections like 'System Configuration', 'Security', and 'Clouds' are visible but not highlighted.



The screenshot shows the Jenkins Nodes page. The '+ New Node' button is highlighted with a red box. The table header includes columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. A single node entry is shown: Built-In Node, Linux (amd64), In sync, 3.63 GiB, 0 B, 3.63 GiB, 0ms, and 19 min.

Step 2: Write “Node name” as “Slave1”. Choose “Type” as “Permanent Agent”.



### Step 3: Choose the following options here:

**Name:** Test

**Number of executors:** 1

**Remote root directory:** /home/ubuntu/jenkins

Name ?  
Test

Description ?  
Plain text Preview

Number of executors ?  
1

Remote root directory ?  
/home/ubuntu/jenkins

Labels ?  
Save

**Label:** Write “Test” here because “Jenkins” identifies the node using this feature.

**Launch Method:** Launch agents via SSH

Credentials: Click on “Add”. Again, click on “Jenkins”.

The screenshot shows the Jenkins Node configuration interface. The 'Host' field is filled with '172.31.0.167'. A red error message 'The Host must be specified' is displayed below the field. The 'Credentials' dropdown is set to '- none -'. The 'Save' button is visible at the bottom.

**Credentials** ?  
- none -  
Jenkins Credentials Provider  
Jenkins not be found

**Host Key Verification Strategy** ?  
Known hosts file Verification Strategy

**Availability** ?

**Save**

In “Add Credentials”, choose “kind” as “SSH Username with private key”.

**Scope: Global (Jenkins, nodes, items, all child items, etc)** — Remain as it is.

**ID & Description : PEM**

**Username:** ubuntu

The screenshot shows the 'Jenkins Credentials Provider: Jenkins' configuration page. The 'Kind' dropdown is set to 'SSH Username with private key'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'ID' field contains 'PEM'. The 'Description' field is empty. The 'Username' field contains 'ubuntu'. The 'Treat username as secret' checkbox is unchecked. The 'Save' button is visible at the bottom.

**Jenkins Credentials Provider: Jenkins**

**Kind**  
SSH Username with private key

**Scope** ?  
Global (Jenkins, nodes, items, all child items, etc)

**ID** ?  
PEM

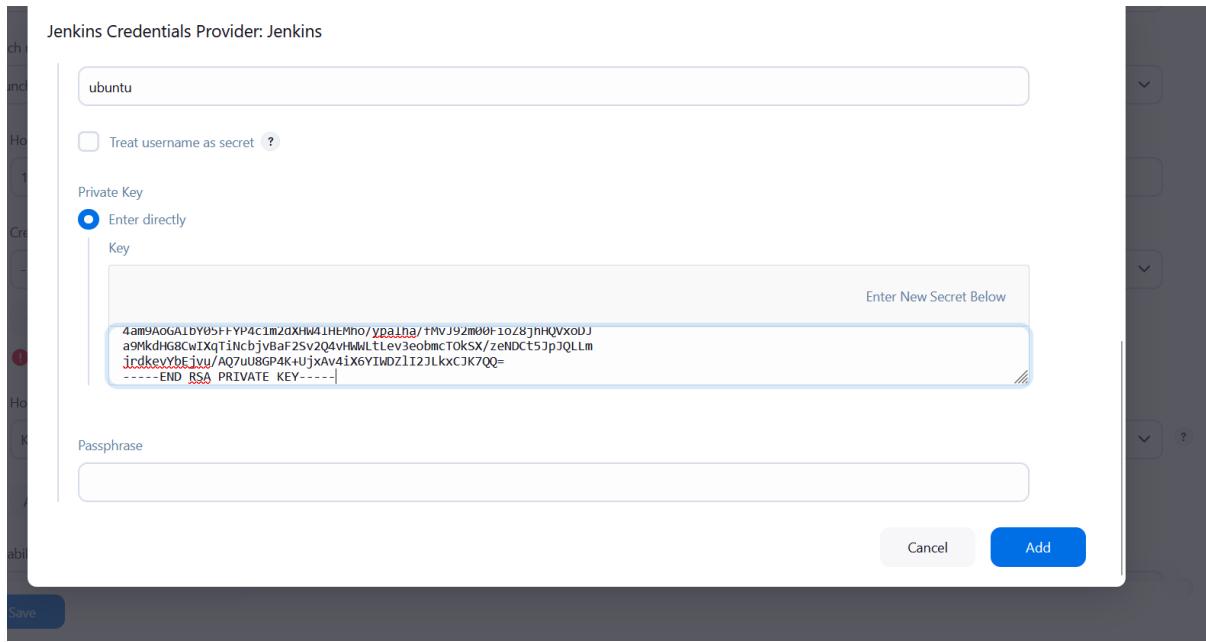
**Description** ?

**Username**  
ubuntu

Treat username as secret ?

**Save**

In “Private Key”, choose “Enter Directly”. Click on “Add”

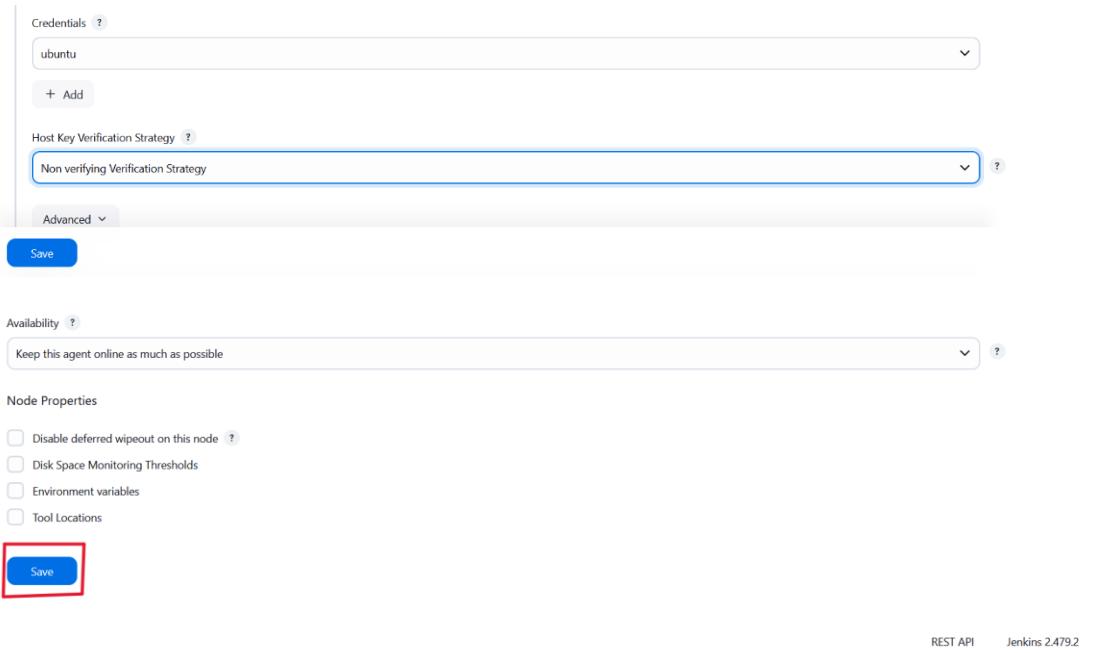


**Click on “Add”.**

**Step 4: Choose “Credentials” as “ubuntu”.**

**Choose “Host Key Verification Strategy” as “Non verifying Verification Strategy”.**

**Click on “Save”.**



**Step 5: “Test Node” has been successfully launched like this:**

The screenshot shows the Jenkins 'Nodes' page. On the left, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (two entries: 'built-in Node' and 'Test'). The main area is titled 'Nodes' and contains a table with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. It lists two nodes: 'Built-In Node' (Linux (amd64), In sync, 3.63 GiB free disk, 0.0 B free swap, 3.63 GiB free temp, 0ms response) and 'Test' (N/A for all metrics). A legend at the bottom indicates icons for Status (S), Model (M), and Label (L).

## Add “Prod” in “Jenkins”

**Step 1: Go to “Jenkins Master” & click on “New Node”.**

**Step 2: Write “Node name” as “Prod”. Choose “Type” as “Copy Existing Node”. In “Copy Existing Node”, choose “Test”.**

The screenshot shows the 'New node' creation form. The 'Node name' field is filled with 'Prod'. The 'Type' section has 'Copy Existing Node' selected, with 'Test' listed in the dropdown. At the bottom is a 'Create' button.

REST API Jenkins 2.479.2

**Click on “Create”.**

**Step 3: Fill the following options here:**

**Labels:** Prod

**Host:** 172.31.6.94

***Remain all entries are as it is.***

**Click on “Save”.**

Labels ?  
Prod

Usage ?  
Use this node as much as possible

Launch method ?  
Launch agents via SSH

Host ?  
172.31.6.94

Credentials ?  
ubuntu

+ Add

Host Key Verification Strategy ?

**Save**

#### Step 4: The “Prod” Node has been successfully launched.

The screenshot shows the Jenkins interface for managing nodes. The top navigation bar includes links for Nodes [Jenkins], Manage Jenkins, and Nodes. The main content area is titled "Nodes" and displays a table of currently available nodes:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
1	Built-In Node	Linux (amd64)	In sync	3.07 GiB	1 0 B	3.07 GiB	0ms
2	Prod	Linux (amd64)	In sync	3.98 GiB	1 0 B	3.98 GiB	40ms
3	Test	Linux (amd64)	In sync	3.98 GiB	1 0 B	3.98 GiB	32ms
	Data obtained	11 sec	11 sec	10 sec	10 sec	10 sec	10 sec

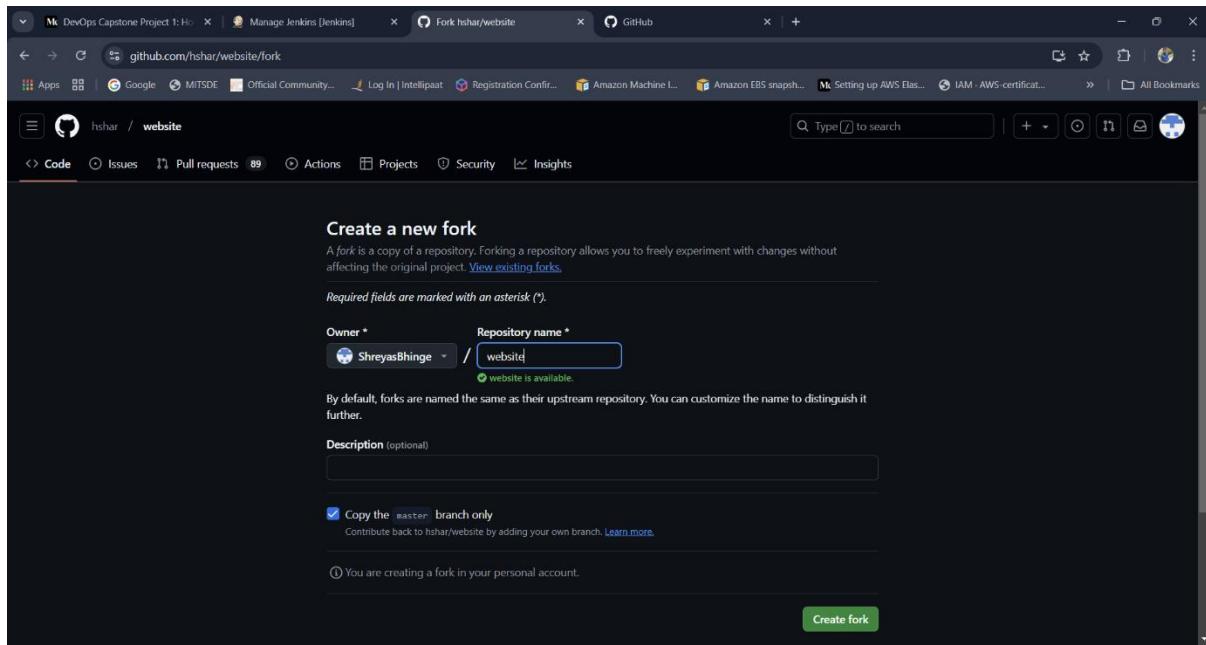
On the left sidebar, there are sections for Build Queue (empty), Build Executor Status (listing Built-In Node, Prod, and Test), and Clouds (empty). A legend at the bottom right indicates icons for Small (S), Medium (M), and Large (L) nodes.

#### Problem (2) Solution: Git workflow has to be implemented

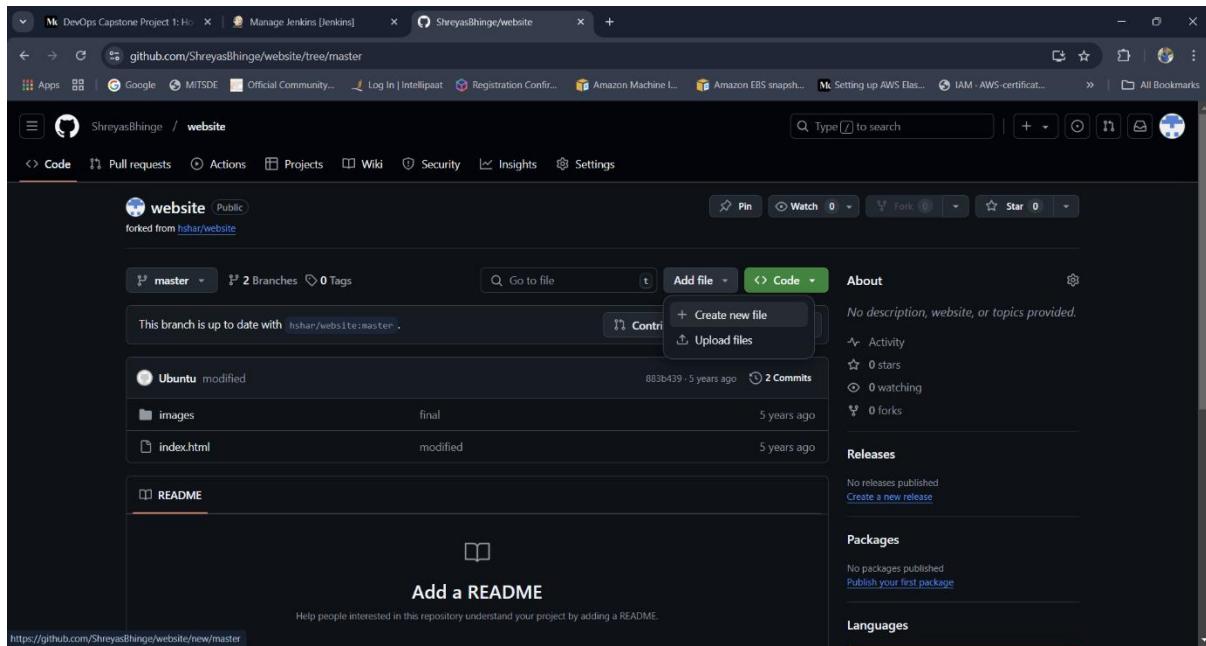
##### A. Fork the Given Repository

**Step 1:** First, fork the given repository to your “GitHub Account” to perform this assignment.

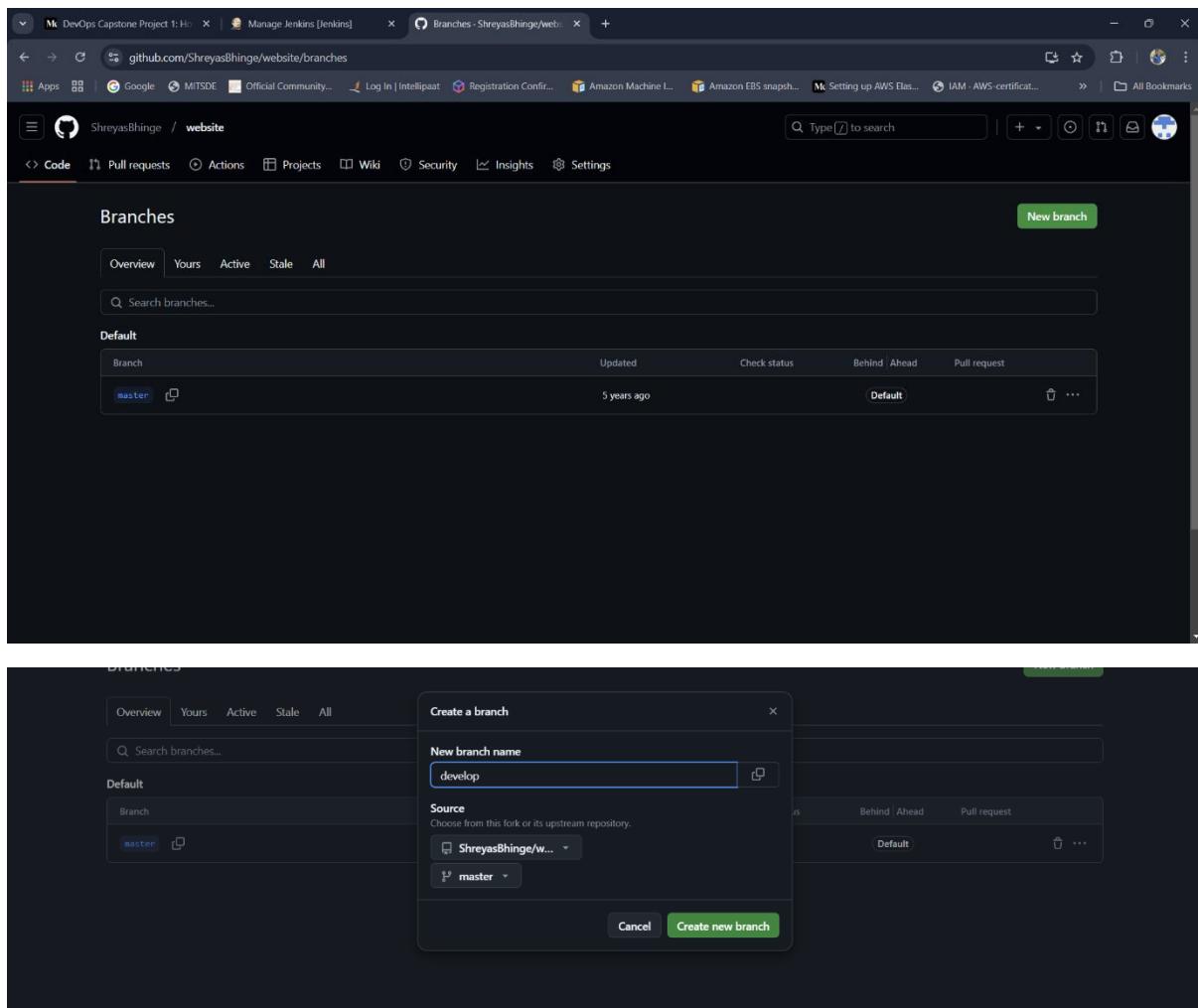
**Now, the “Fork” option will be shown. Click on “Fork”. It will automatically fetch “website” as the “Repository name”.**



Given that “Repository [website]” has been successfully forked.



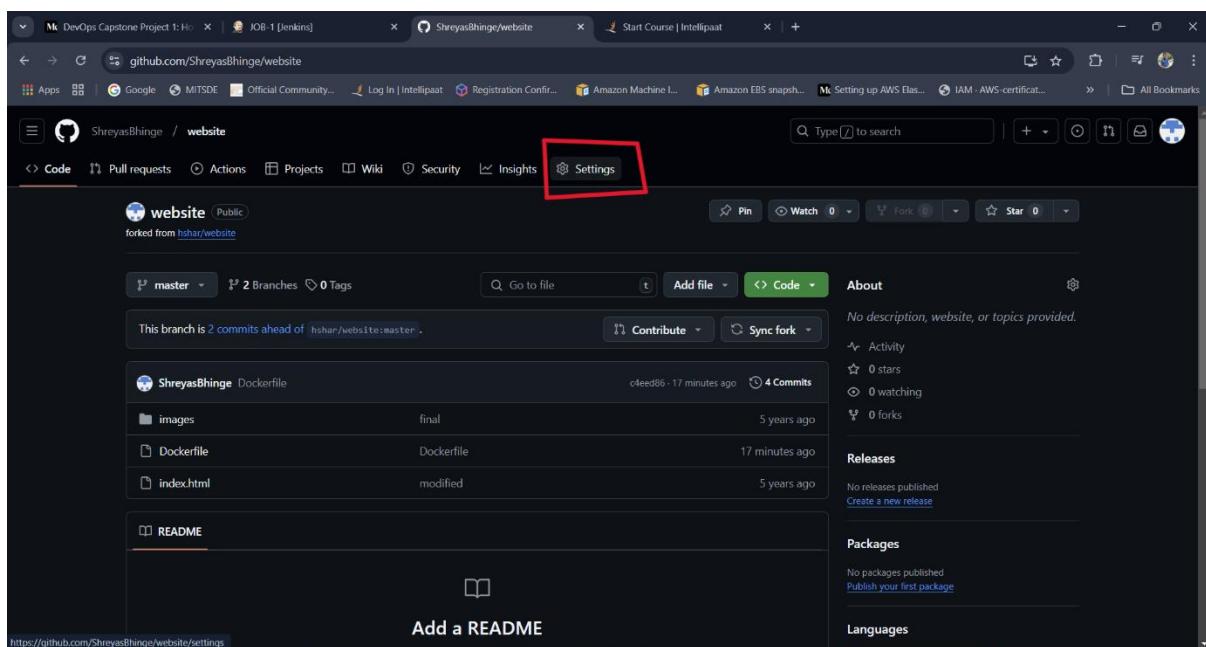
check the branches. The “develop” branch will be successfully shown here.



The image shows two screenshots of the GitHub interface. The top screenshot displays the 'Branches' page for the repository 'ShreyasBhinge/website'. It lists a single branch, 'master', which was last updated 5 years ago and is set as the 'Default'. A green 'New branch' button is visible in the top right. The bottom screenshot shows a 'Create a branch' modal window. In the 'New branch name' field, 'develop' is typed. The 'Source' dropdown shows 'ShreyasBhinge/w...' and 'master' is selected. At the bottom of the modal are 'Cancel' and 'Create new branch' buttons.

## Create a Webhook for Trigger the Jobs

### Step 1: Go to “Settings”.



The image shows the 'Settings' page for the 'ShreyasBhinge/website' repository. The 'Settings' tab is highlighted with a red box. The page includes sections for 'About', 'Releases', 'Packages', and 'Languages'. The 'About' section notes 'No description, website, or topics provided.' The 'Releases' section says 'No releases published' and 'Create a new release'. The 'Languages' section indicates 'No packages published' and 'Publish your first package'.

The screenshot shows the GitHub repository settings page for 'ShreyasBhinge / website'. The 'General' tab is selected. On the left sidebar, the 'Webhooks' option under the 'Actions' section is highlighted with a red box. The main content area shows the 'Default branch' configuration, which is set to 'master'. There is also a 'Social preview' section where an image can be uploaded.

#### Step 4: Choose the following options here:

**Payload URL:** <http://3.110.160.39:8080/github-webhook/>

**Which events would you like to trigger this webhook?:** Send me everything

The screenshot shows the 'Webhooks / Add webhook' form. In the 'Payload URL' field, the value 'http://3.239.239.8:8080/github-webhook/' is entered. The 'Content type' dropdown is set to 'application/x-www-form-urlencoded'. The 'Secret' field is empty. The 'SSL verification' section indicates that SSL certificates are verified by default. The 'Add webhook' button is visible at the bottom of the form.

**Click on “Add webhook”.**

The screenshot shows the final step of adding a webhook. It includes a note about SSL verification, a list of event triggers ('Just the push event.', 'Send me everything.', 'Let me select individual events.'), and a checked 'Active' checkbox with a note about delivering event details. The 'Add webhook' button is present at the bottom.

**Step 5: Your webhook will be successfully created.**

The screenshot shows the GitHub settings page for a repository named 'website'. The 'Webhooks' tab is selected. A single webhook is listed with the URL 'http://3.239.239.8:8080/github-webhook/push/'. The status indicates 'Last delivery was successful.'

**Problem (4) Solution:** The code should be containerized with the help of a Dockerfile. The Dockerfile should be built every time there is a push to GitHub. Use the following pre-built container for your application: hshar/webapp The code should reside in '/var/www/html'

#### A. Create a Dockerfile

The screenshot shows the GitHub repository page for 'website'. The 'Code' tab is selected. A prominent 'Create new file' button is highlighted with a red box.

The screenshot shows the GitHub repository page for 'website'. The 'Code' tab is selected, and the 'Dockerfile' file is open. The Dockerfile content is as follows:

```

1 FROM ubuntu
2 RUN apt-get update
3 RUN apt-get install apache2 -y
4 ADD index.html /var/www/html/
5 ENTRYPOINT apache2 -D FOREGROUND
6

```

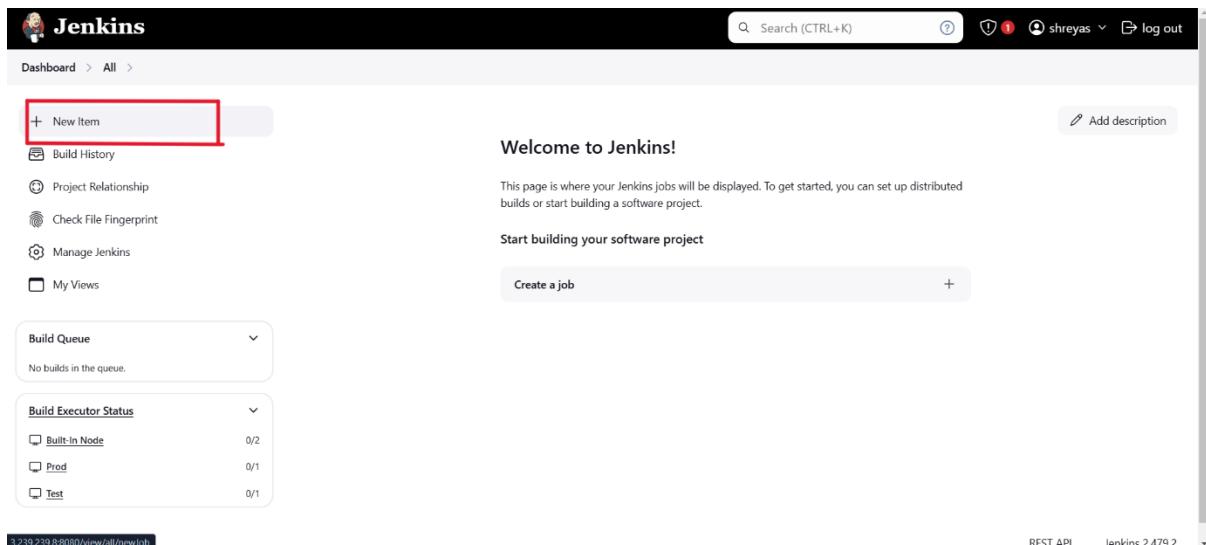
**Problem (5) Solution:** The above tasks should be defined in a Jenkins Pipeline with the following jobs:

Create Job-1 in develop branch lable as Test

Create JOB-2 in master branch and lable as Test

Create JOB-3 in master branch and lable as Prod

#### A. Create a JOB-1



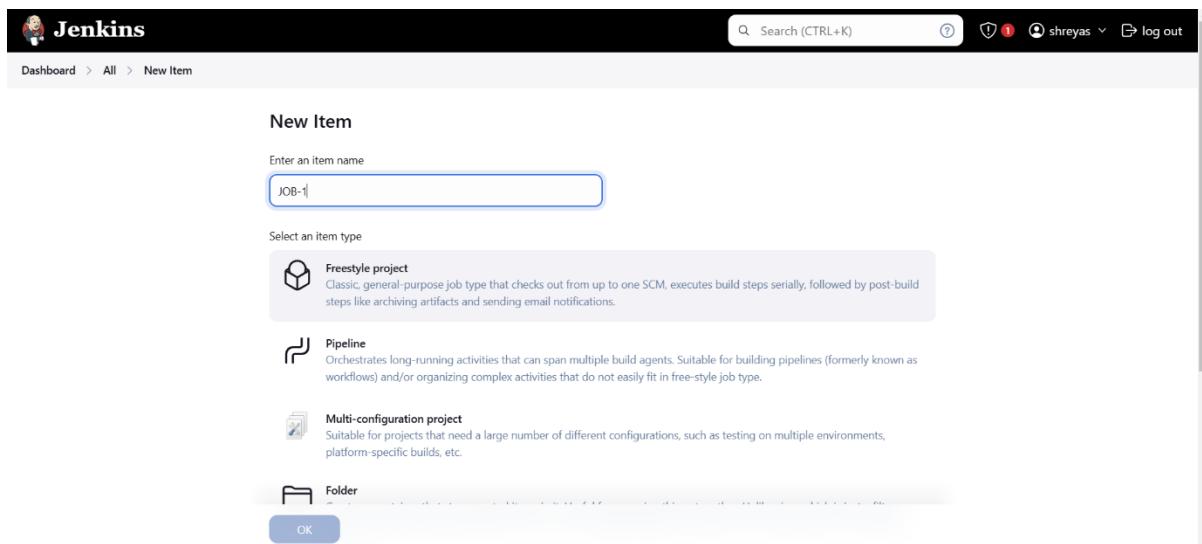
The screenshot shows the Jenkins dashboard. At the top, there's a navigation bar with links for 'Dashboard', 'All', 'New Item' (which is highlighted with a red box), 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', 'My Views', 'Add description', 'Create a job', and a '+' button. Below the navigation is a section titled 'Welcome to Jenkins!' with a message: 'This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.' There are two sections: 'Start building your software project' and 'Create a job'. On the left, there are collapsed sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (Built-In Node: 0/2, Prod: 0/1, Test: 0/1). At the bottom, it shows the URL '3.239.239.8:8080/view/all/newJob' and the Jenkins version '2.479.2'.

**Step 2: Choose the following options here:**

**Enter an item name: JOB-1**

**Option: Freestyle project**

**Click on “OK”.**



The screenshot shows the 'New Item' creation page. The title is 'New Item'. It has a field 'Enter an item name' containing 'JOB-1'. Below it, there's a section 'Select an item type' with three options: 'Freestyle project' (selected), 'Pipeline', and 'Multi-configuration project'. Each option has a small icon and a brief description. At the bottom is a blue 'OK' button.

**Step 3: Choose “Description” as “JOB-1”**

**Step 4: Go to the “GitHub” account & click on Code. Go to the “HTTPS” section & copy the link.**

**While Choose “Label Expression” as “Slave1”.**

**Step 5: In “Source Code Management”, choose “Git”. Put “Repo URL” in the “Repository URL” section.**

**choose “Credentials” as “none”.**

**Step 6: Choose “Branch” as “develop” in “Branch Specifier”.**

**Step 7: Choose “GitHub hook trigger for GITScm polling” in “Build Triggers”.**

**Step 8: Click on “Apply”. After clicking on “Apply”, choose “Save”.**

**Step 9: Your “JOB-1” has been successfully created.**

The screenshot shows the Jenkins web interface. At the top, there are several browser tabs: "Mr DevOps Capstone Project 1: H...", "JOB-1 [Jenkins]", "Not secure 3.239.239.8:8080/job/JOB-1/", "ShreyasBhinge/website", "Start Course | Intellipaat", and others. The main content area is titled "JOB-1". On the left, there's a sidebar with options like "Status", "Changes", "Build scheduled", "Build Now", "Configure", "Delete Project", "GitHub", and "Rename". Below that is a "Builds" section showing a single build entry: "#1 09:35" with a progress bar. The main panel displays a table of builds for "JOB-1", showing columns for Status (S), Workstation (W), Name, Last Success, Last Failure, and Last Duration. The first row shows a green checkmark for Status, a sun icon for Workstation, "JOB-1" for Name, "4 min 19 sec" for Last Success, "N/A" for Last Failure, and "7.8 sec" for Last Duration. At the bottom of the page, the URL "3.239.239.8:8080/job/JOB-1/build?delay=0sec" is shown, along with "REST API" and "Jenkins 2.479.2".

## Creating JOB-2

The screenshot shows the "New Item" dialog in Jenkins. The title is "New Item". There is a text input field labeled "Enter an item name" containing "JOB-2". Below it is a section titled "Select an item type" with three options: "Freestyle project", "Pipeline", and "Multi-configuration project". The "Freestyle project" option is selected and described as "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.". The "Pipeline" option is described as "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.". The "Multi-configuration project" option is described as "Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.". At the bottom of the dialog is a blue "OK" button.

**Jenkins**

Dashboard > JOB-2 > Configuration

**Configure**

**General**

Description: JOB-2

Discard old builds ?

GitHub project  
Project url: https://github.com/ShreyasBhinge/website.git

**Advanced**

**Save** **Apply**

**Enabled**

This screenshot shows the Jenkins job configuration for 'JOB-2'. The 'General' tab is selected. It includes a 'Description' field containing 'JOB-2', a 'Discard old builds' checkbox, and a 'GitHub project' section where 'https://github.com/ShreyasBhinge/website.git' is specified. An 'Enabled' toggle switch is turned on.

## Label as Test

**Configure**

**General**

Restrict where this project can be run ?  
Label Expression: Test

Label Test matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

**Source Code Management**

None

Git ?  
Repositories ?

Repository URL: https://github.com/ShreyasBhinge/website.git

**Credentials** ?  
ubuntu

**Save** **Apply**

**Configure**

**Source Code Management**

**Credentials** ?  
ubuntu

**Add**

**Advanced**

**Add Repository**

**Branches to build** ?

Branch Specifier (blank for 'any') ?  
\*/master

**Add Branch**

**Repository browser** ?

(Autumn)

**Save** **Apply**

This screenshot shows the Jenkins job configuration for 'JOB-2' under the 'Source Code Management' tab. It uses a 'Git' repository with 'https://github.com/ShreyasBhinge/website.git' and a credential named 'ubuntu'. The 'Credentials' dropdown also lists 'ubuntu'.

Save the job

**Configure**

General  
Source Code Management  
**Build Triggers**  
Build Environment  
Build Steps  
Post-build Actions

Repository browser ?  
(Auto)

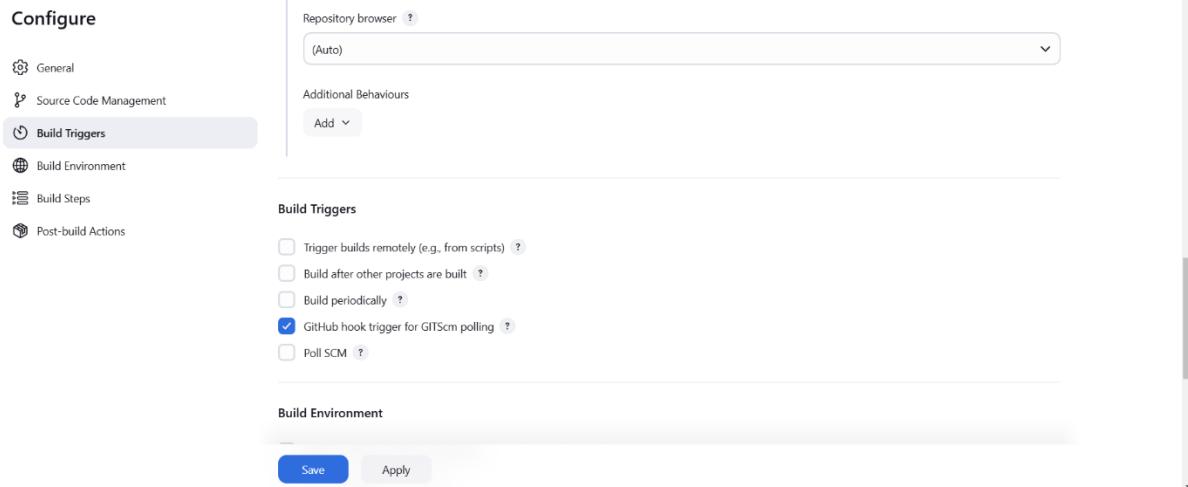
Additional Behaviours  
Add ▾

**Build Triggers**

Trigger builds remotely (e.g., from scripts) ?  
 Build after other projects are built ?  
 Build periodically ?  
 GitHub hook trigger for GITScm polling ?  
 Poll SCM ?

**Build Environment**

Save Apply



**Enter an item name:** JOB-3

**Option:** Freestyle project

Jenkins

Dashboard > All > New Item

New Item

Enter an item name  
JOB-3

Select an item type

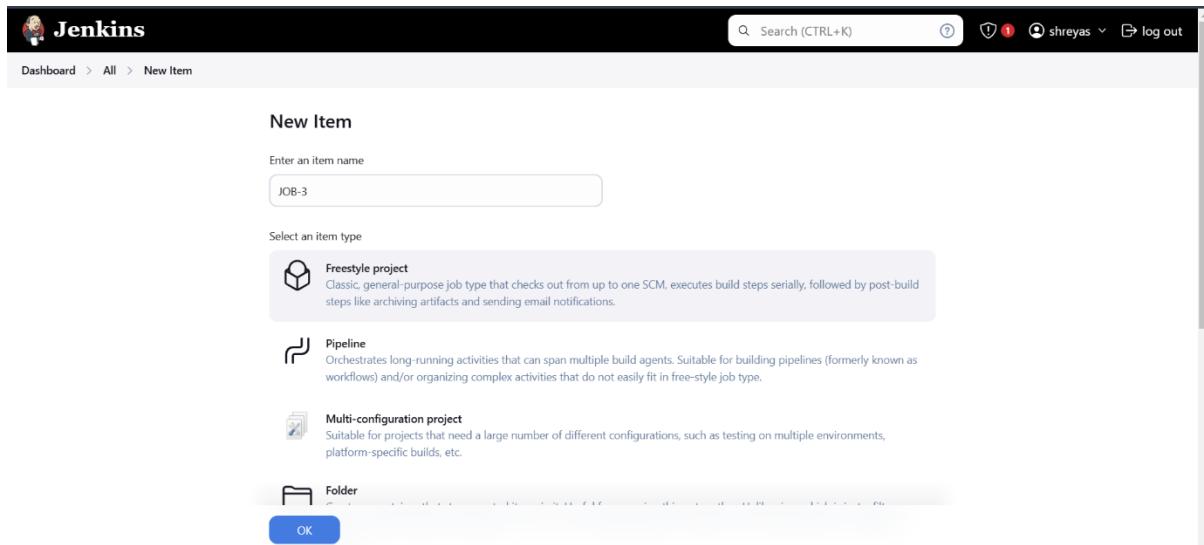
**Freestyle project**  
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

**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.

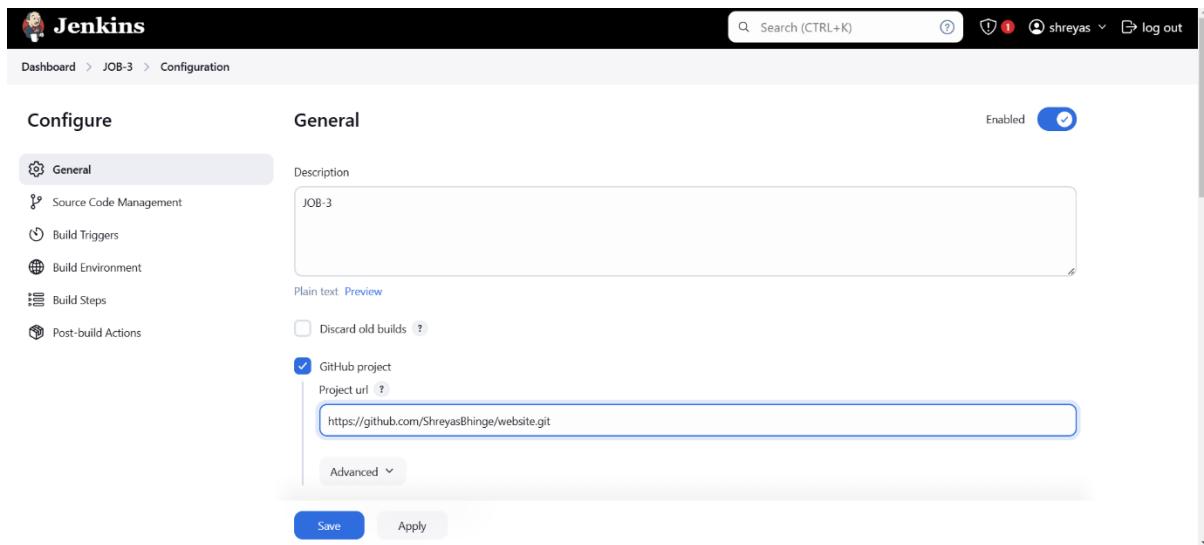
**Folder**

OK

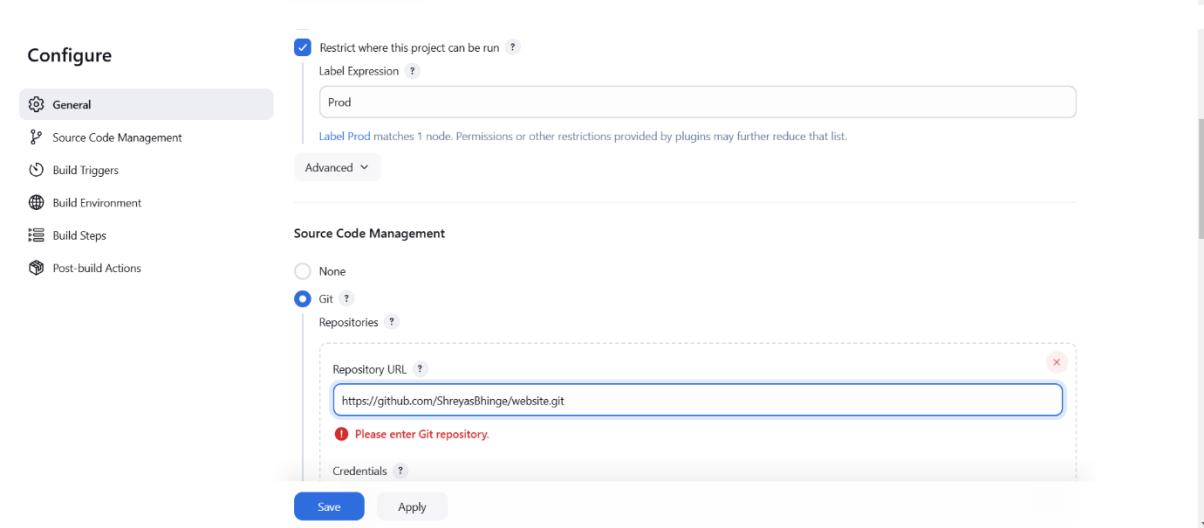


In “General”, put the description as “JOB-3”

Choose “Label Expression” as “Prod” in “Restrict where this project can be run”.

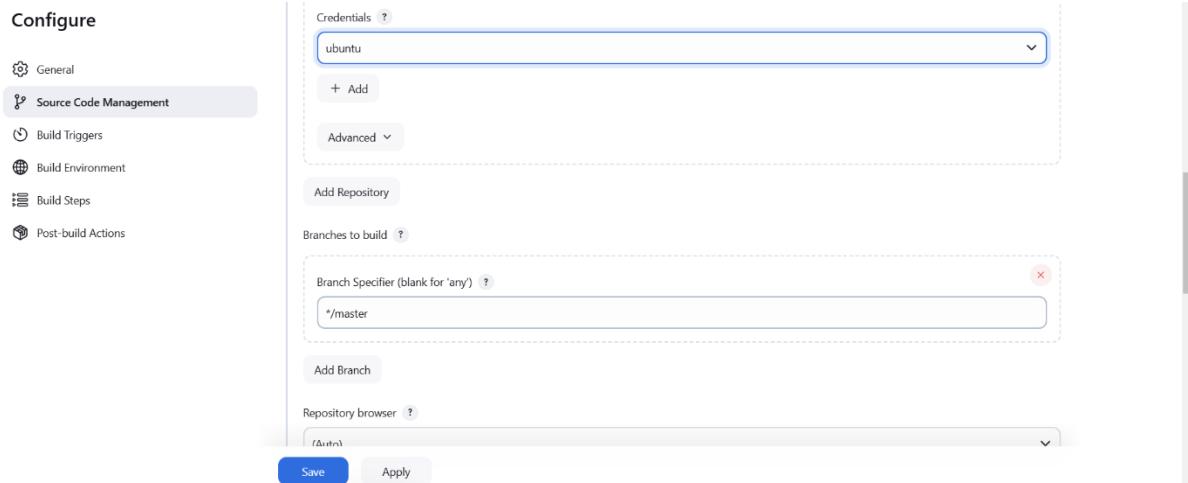


The screenshot shows the Jenkins job configuration for 'JOB-3'. The 'General' tab is selected. The 'Description' field contains 'JOB-3'. Under 'Source Code Management', the 'GitHub project' option is checked, and the 'Project url' is set to 'https://github.com/ShreyasBhinge/website.git'. The 'Save' and 'Apply' buttons are at the bottom.

The screenshot shows the Jenkins job configuration for 'JOB-3'. The 'Source Code Management' tab is selected. Under 'Source Code Management', the 'Git' option is selected. The 'Repository URL' field is set to 'https://github.com/ShreyasBhinge/website.git', but there is an error message: 'Please enter Git repository.'. The 'Save' and 'Apply' buttons are at the bottom.

**Remain other settings as it is. Choose “Branch Specifier (blank for ‘any’) as “\*/master”.**



The screenshot shows the Jenkins job configuration for 'JOB-3'. The 'Source Code Management' tab is selected. Under 'Source Code Management', the 'Git' option is selected. In the 'Credentials' section, 'ubuntu' is listed. In the 'Branches to build' section, 'Branch Specifier (blank for ‘any’)' is set to '\*/master'. The 'Save' and 'Apply' buttons are at the bottom.

**Click on “Build Steps”.**

**Click on “Add build step”.**

**Click on “Execute Shell”**

**Configure**

**Build Environment**

- General
- Source Code Management
- Build Triggers
- Build Environment**
- Build Steps
- Post-build Actions

**Post-build Actions**

Add post-build action ▾

**Save** **Apply**

**Put these commands to build a container on port 81:**

**Configure**

**Build Steps**

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

**Post-build Actions**

**Save** **Apply**

**Click on “Apply>Save”**

**The “Public IP Address” in the “Browser Address Bar” & Press “enter” from the keyboard.**

Instances | EC2 | us-east-1

98.80.135.178:81

Course Mod 98.80.135.178:81

98.80.135.178:81 - Bing Search

M DevOps Capstone Project JOB-3 [Jenkins] ShreyasBhingl/website Intellipaat

Not secure 98.80.135.178:81

Hello world!

