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:

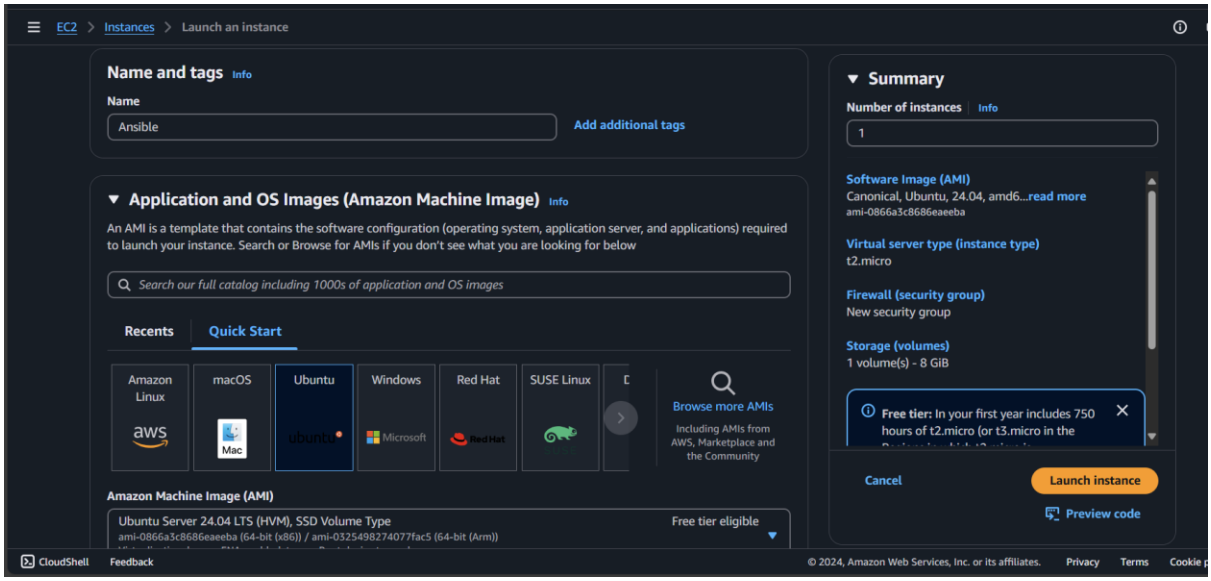
a. Job1 : build

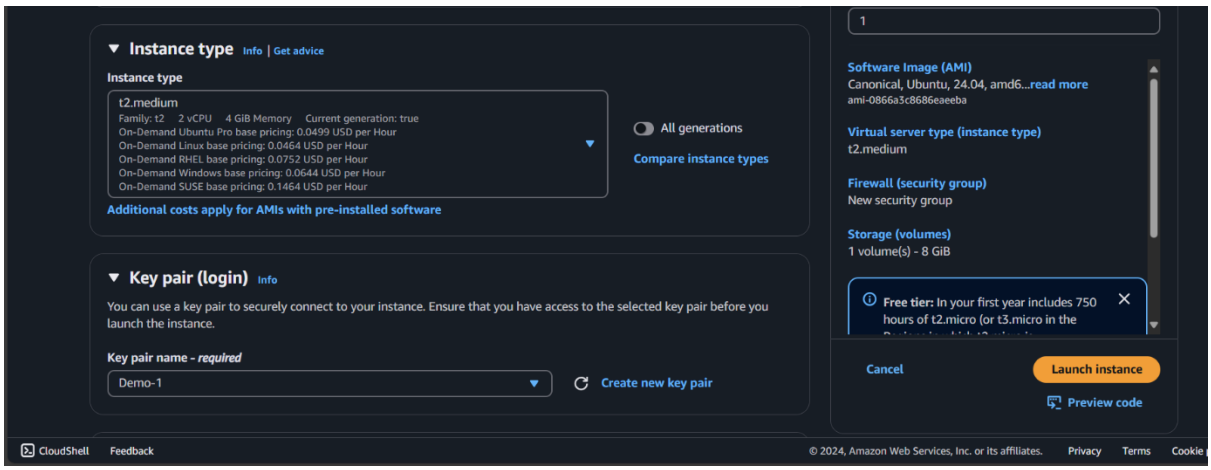
b. Job2 : test

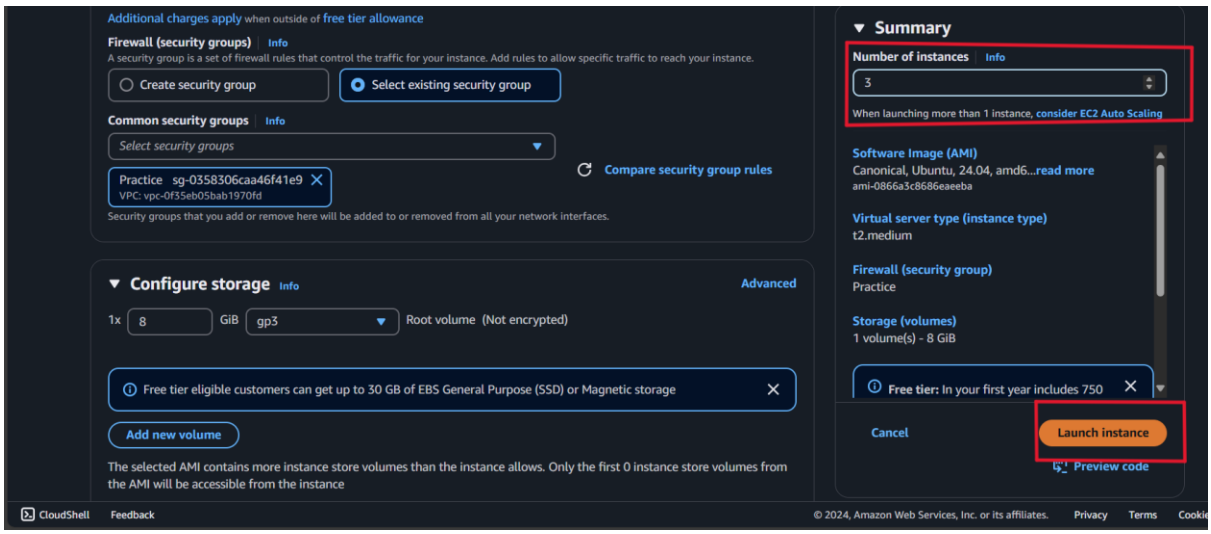
c. Job3 : prod

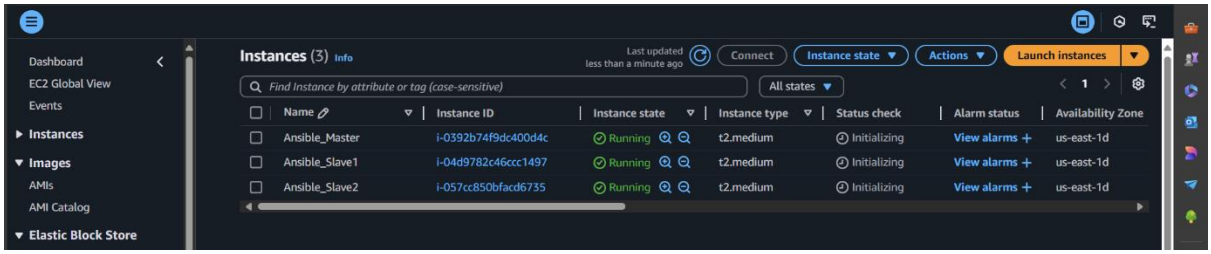
Problem (1) Solution: Install the necessary software on the machines using a configuration management tool

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


EC2 > Instances > Launch an instance
Name and tags Info
Name
Ansible
Add additional tags
Application and OS Images (Amazon Machine Image) Info
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
Recents
Quick Start
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 Info
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
Cancel
Launch instance
Preview code
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms


Instance type Info | Get advice
Instance type
t2.medium
Family: t2 2 vCPU 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
Additional costs apply for AMIs with pre-installed software
Key pair (login) Info
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
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
Cancel
Launch instance
Preview code
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms

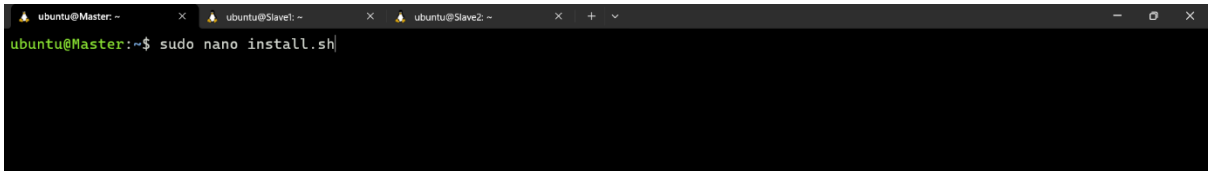

Additional charges apply when outside of free tier allowance
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
Common security groups Info
Select security groups
Practice sg-0358306caa46f41e9
VPC: vpc-0f35eb05bab1970fd
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 Info
Advanced
1x 8 GiB gp3 Root volume (Not encrypted)
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 Info
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
Cancel
Launch instance
Preview code
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms



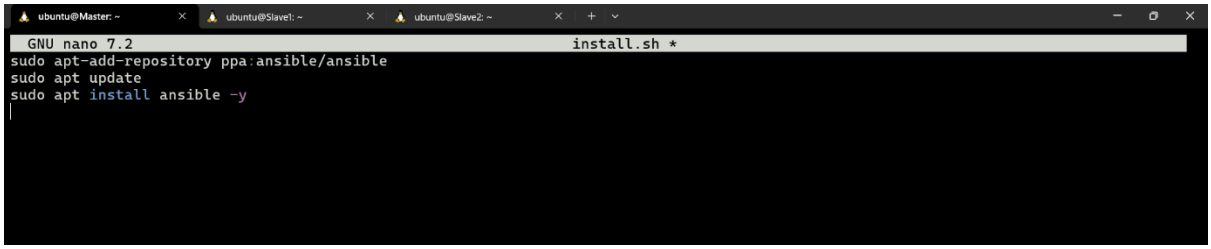
B. Install "Ansible" Over the Master Machine using the "install.sh" file

Step 1: Choose the "Master" machine & click on "Connect"

create an **"install.sh" file: sudo nano install.sh.**

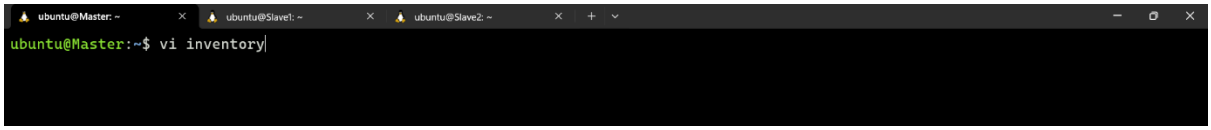


Paste these **commands** in **the "install.sh" file.**

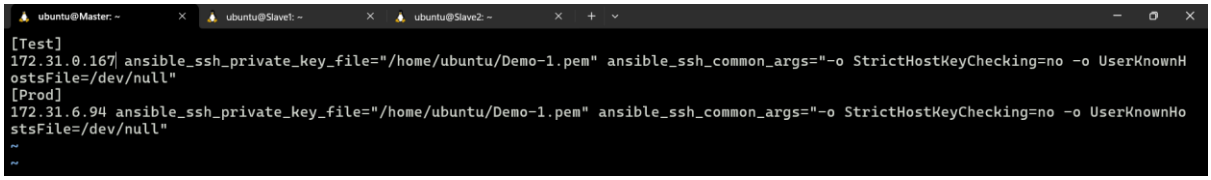


Run the **"install.sh"** file **using** the **command: bash install.sh**

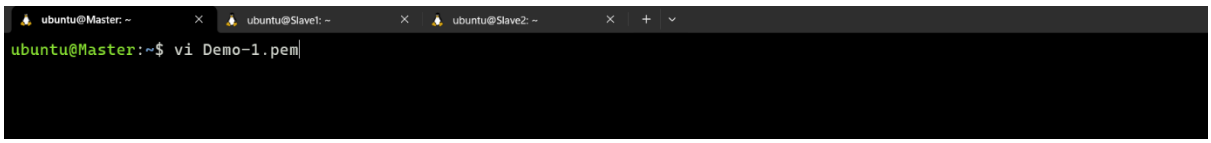
Create an inventory file to connect the slave nodes to master



Give Slave1 as Test and Slave2 as Prod



Add your Private key in your home path to make connection



Give read permission

```
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
```

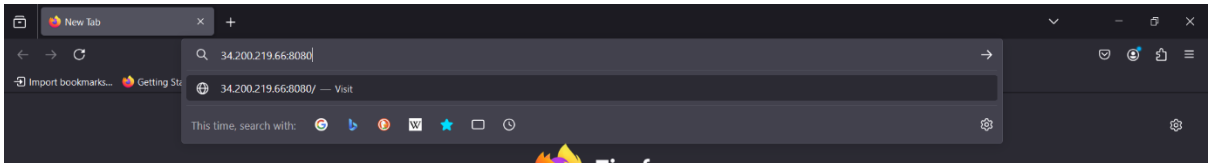
Execute the Playbook

```
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:~$
```

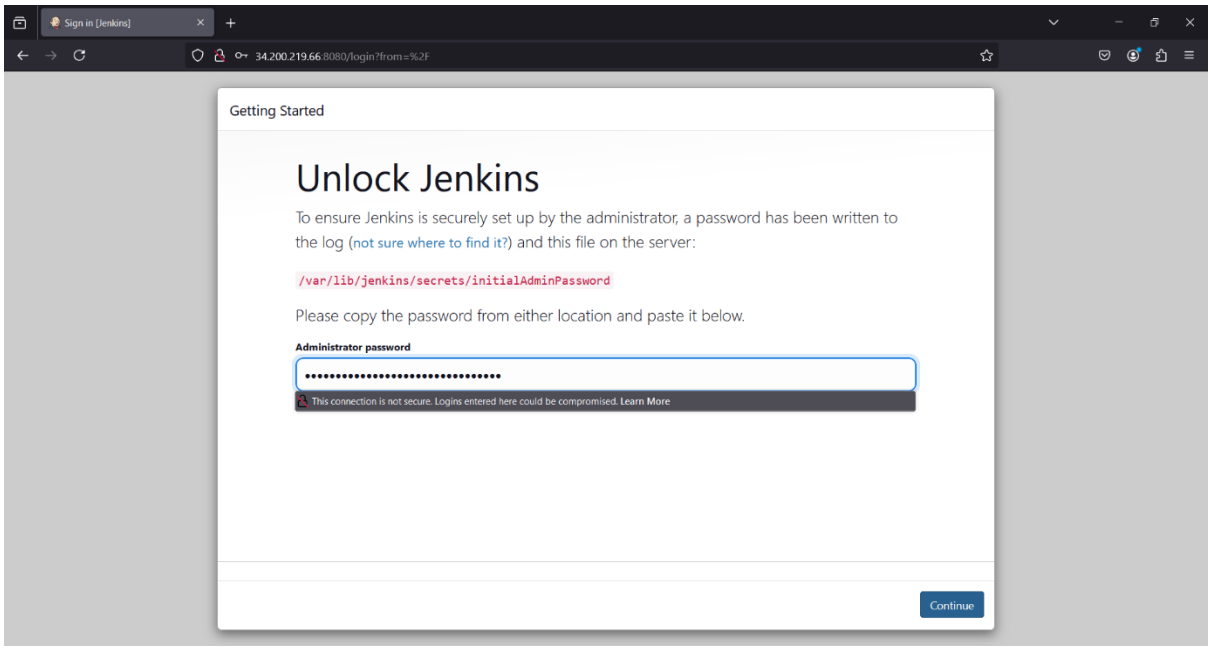
Setup Jenkins Dashboard Over “Master” Server



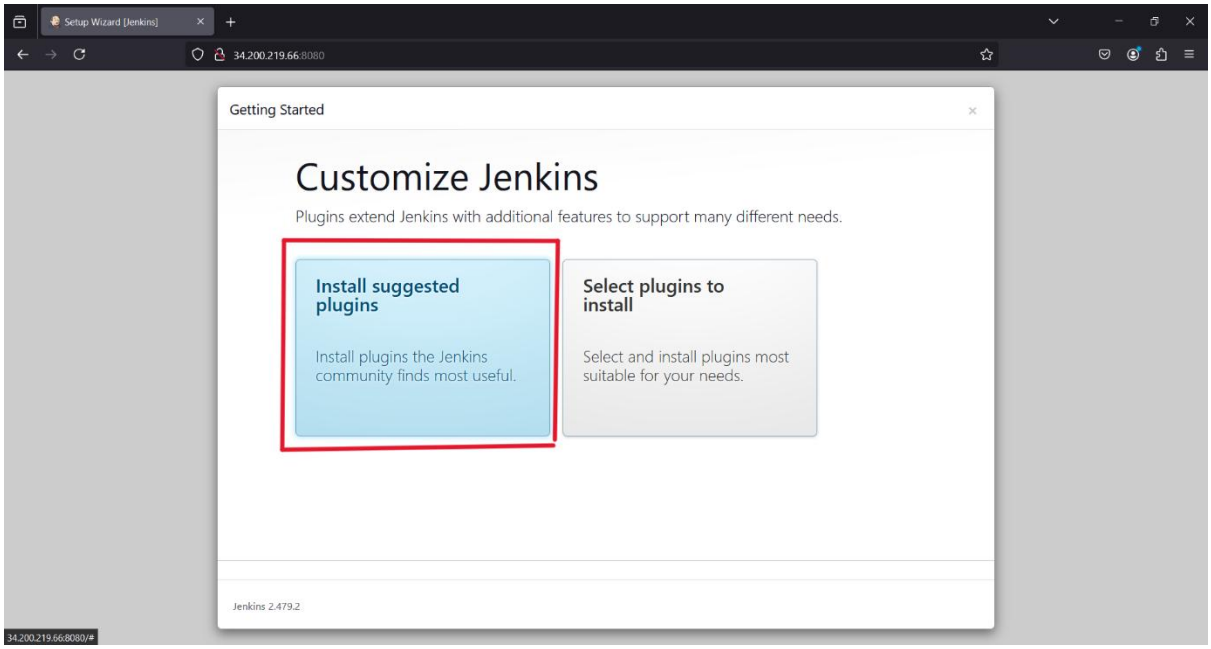
Copy this token from here.

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

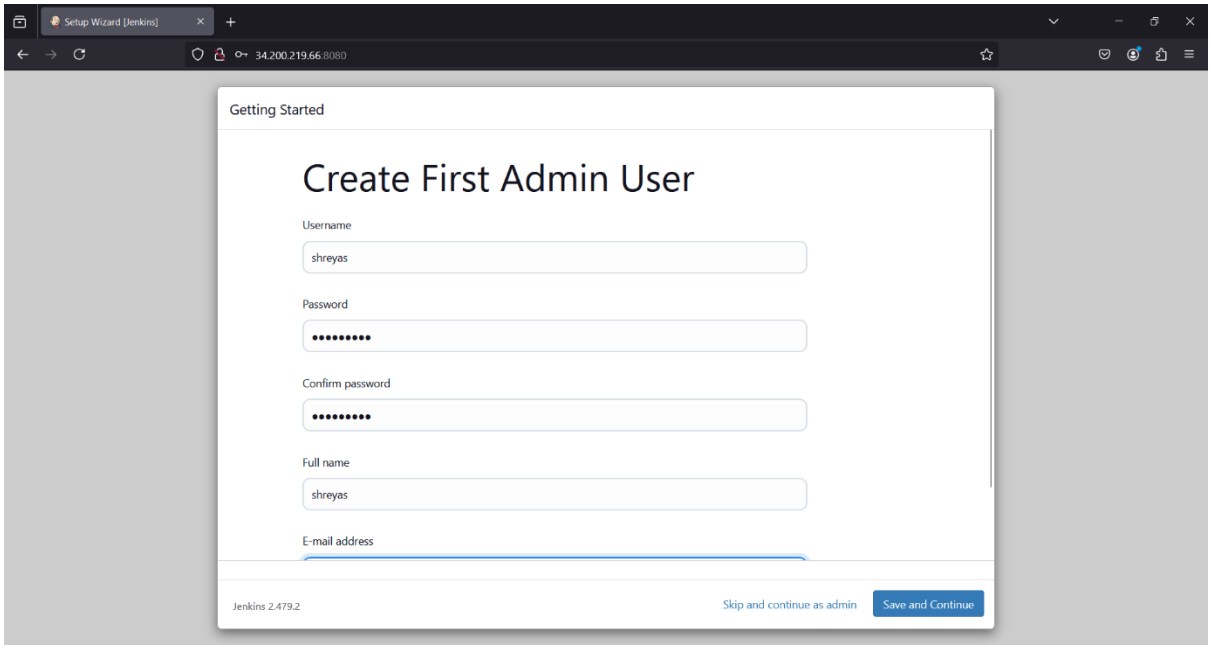
Paste the **token** in **“Dashboard”** & **click** on **“Continue”.**

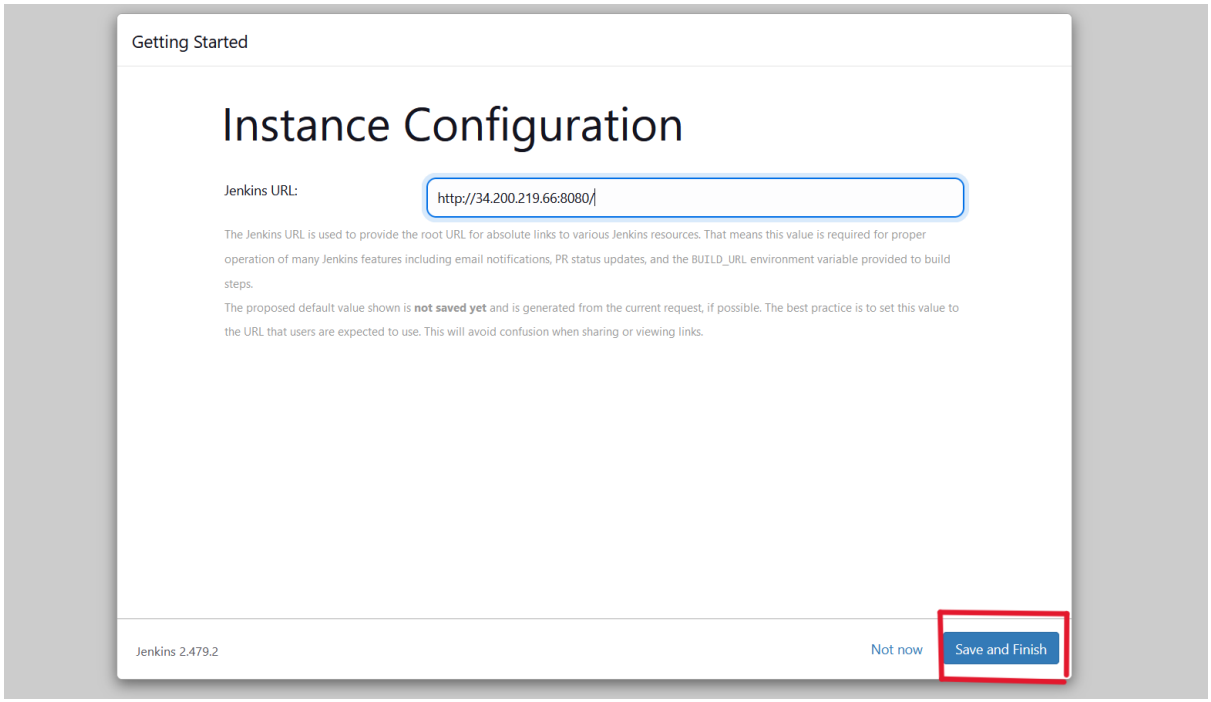


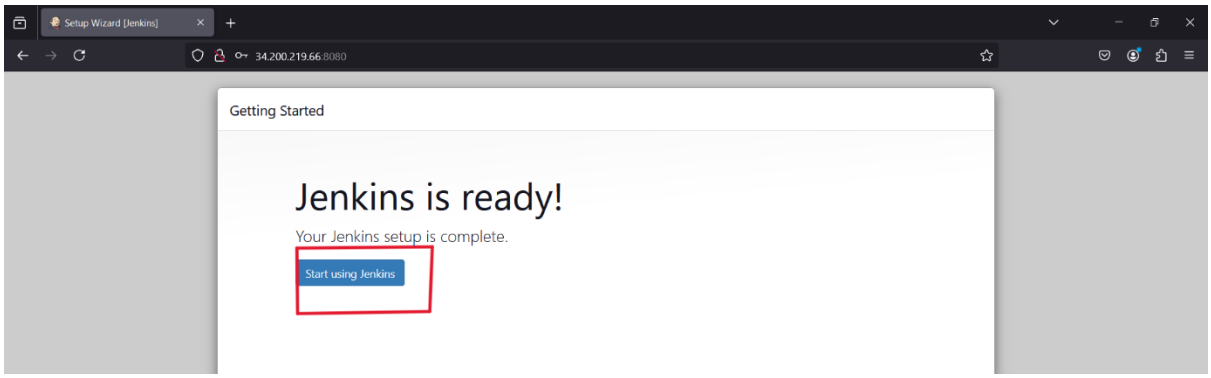
Click on **“Install Suggested Plugins”.**



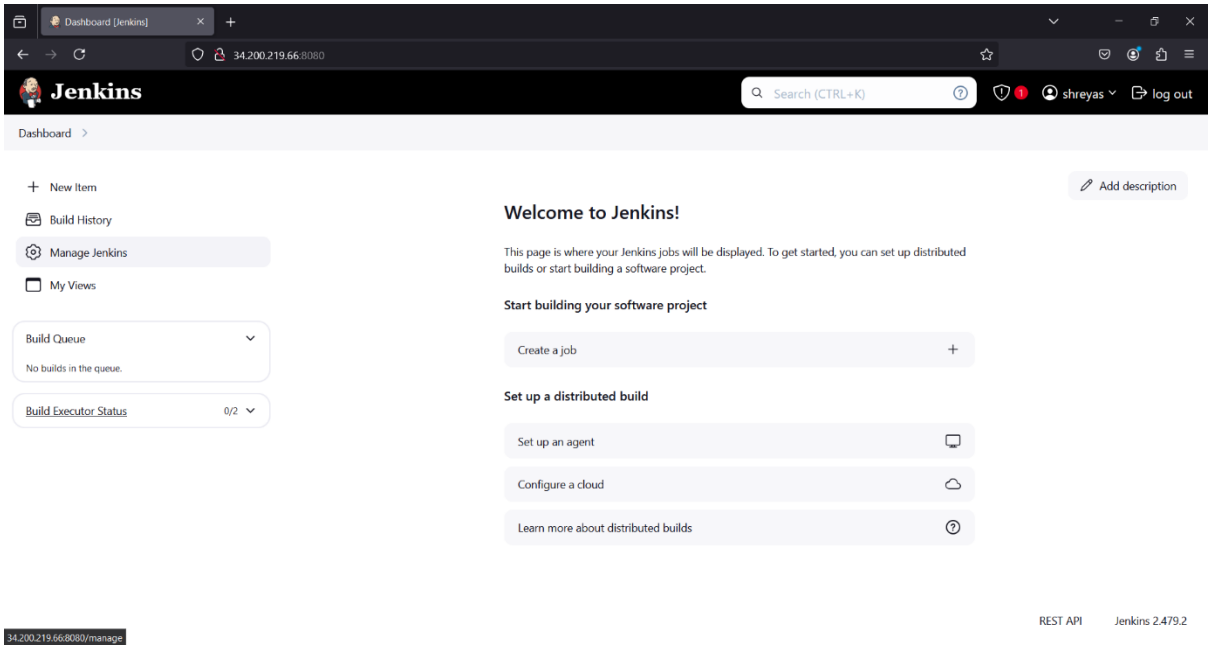
Add Info





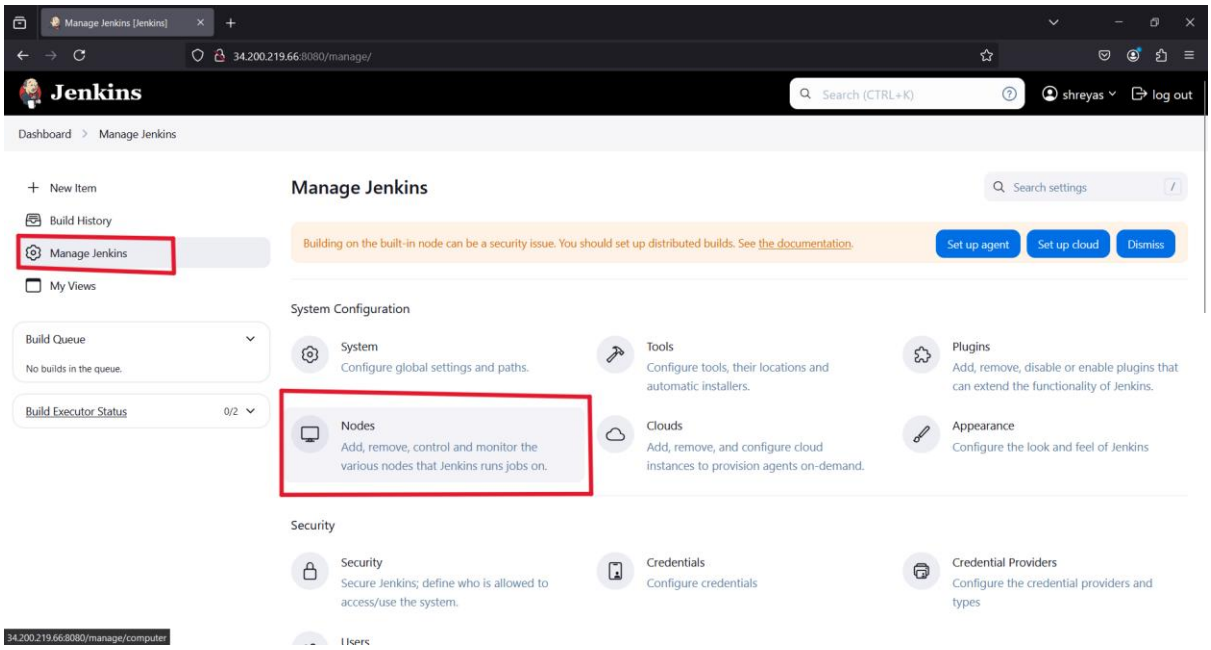


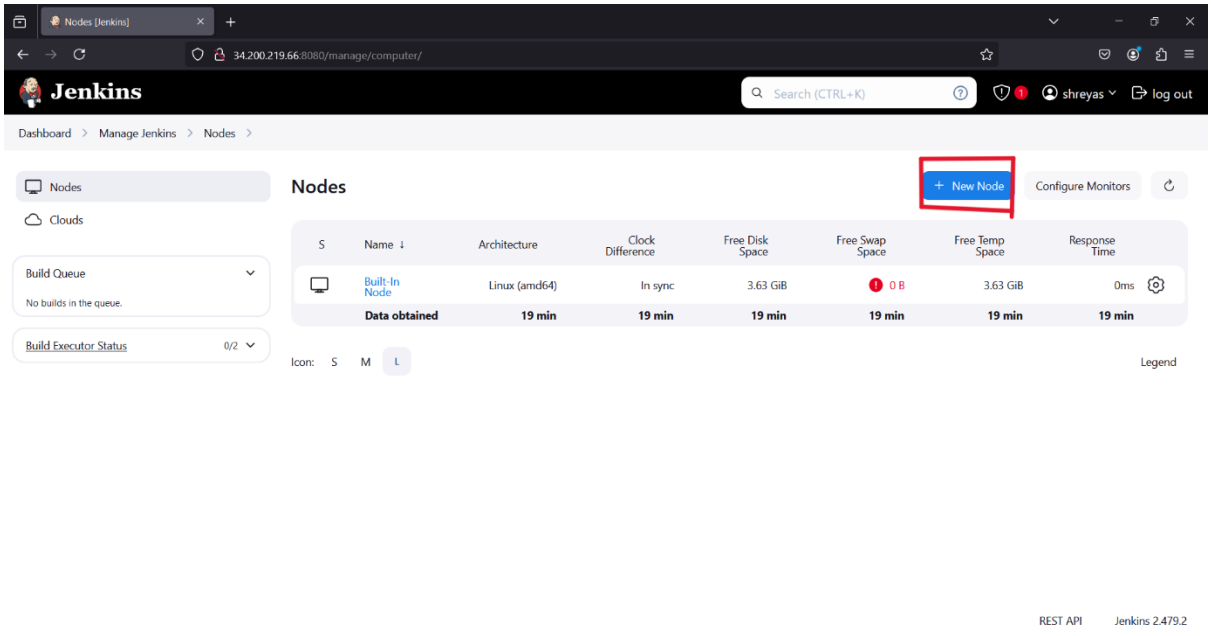
“Jenkins Dashboard” will be **ready.**



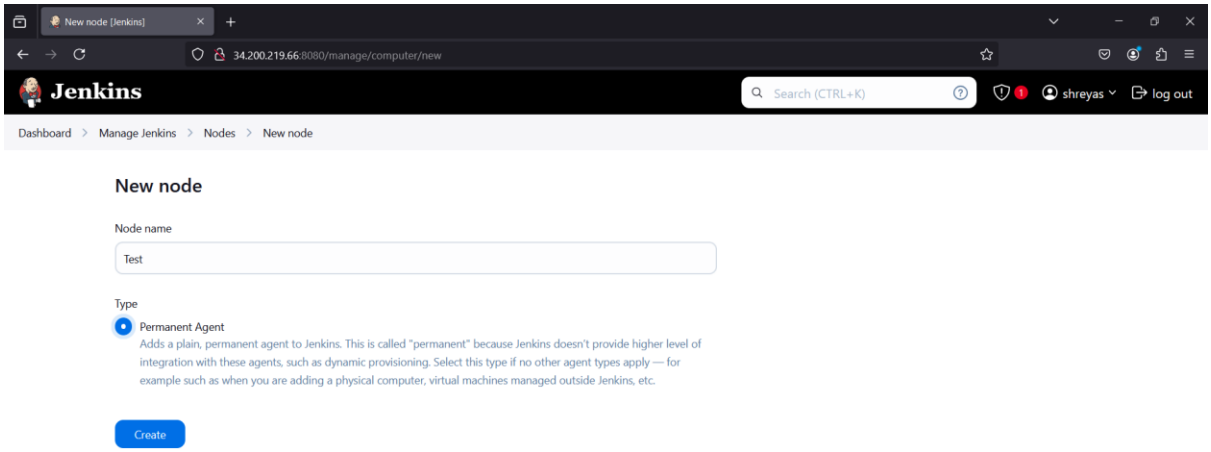
Add “Test” in “Jenkins”

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





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

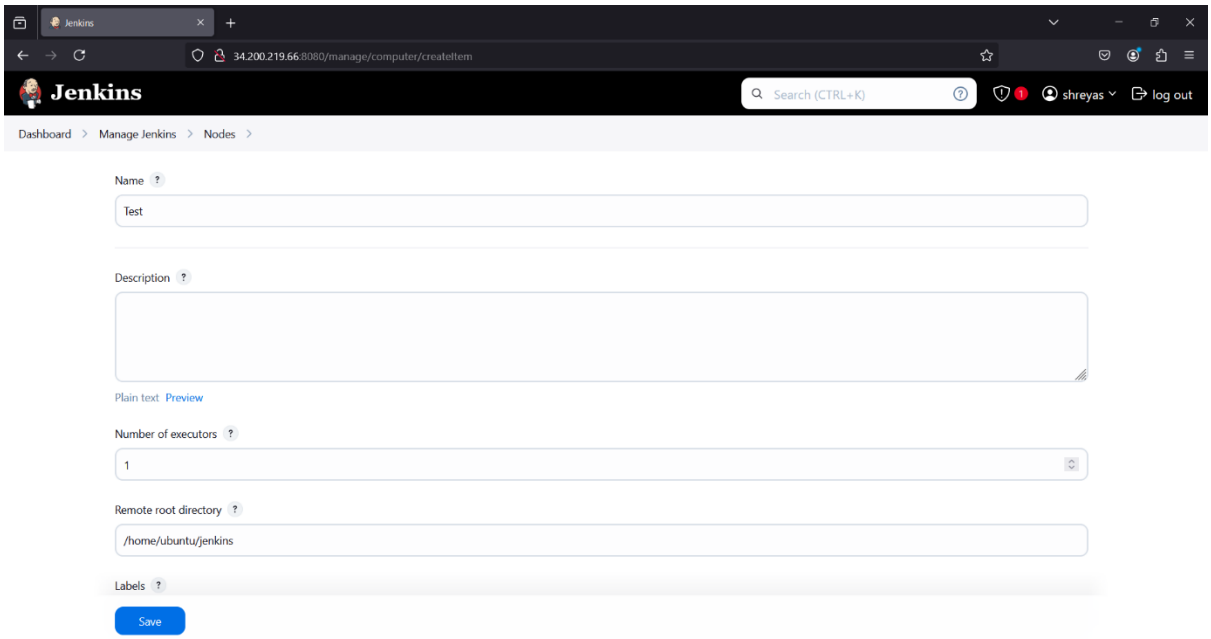


Step 3: Choose the following options here:

Name: Test

Number of executors: 1

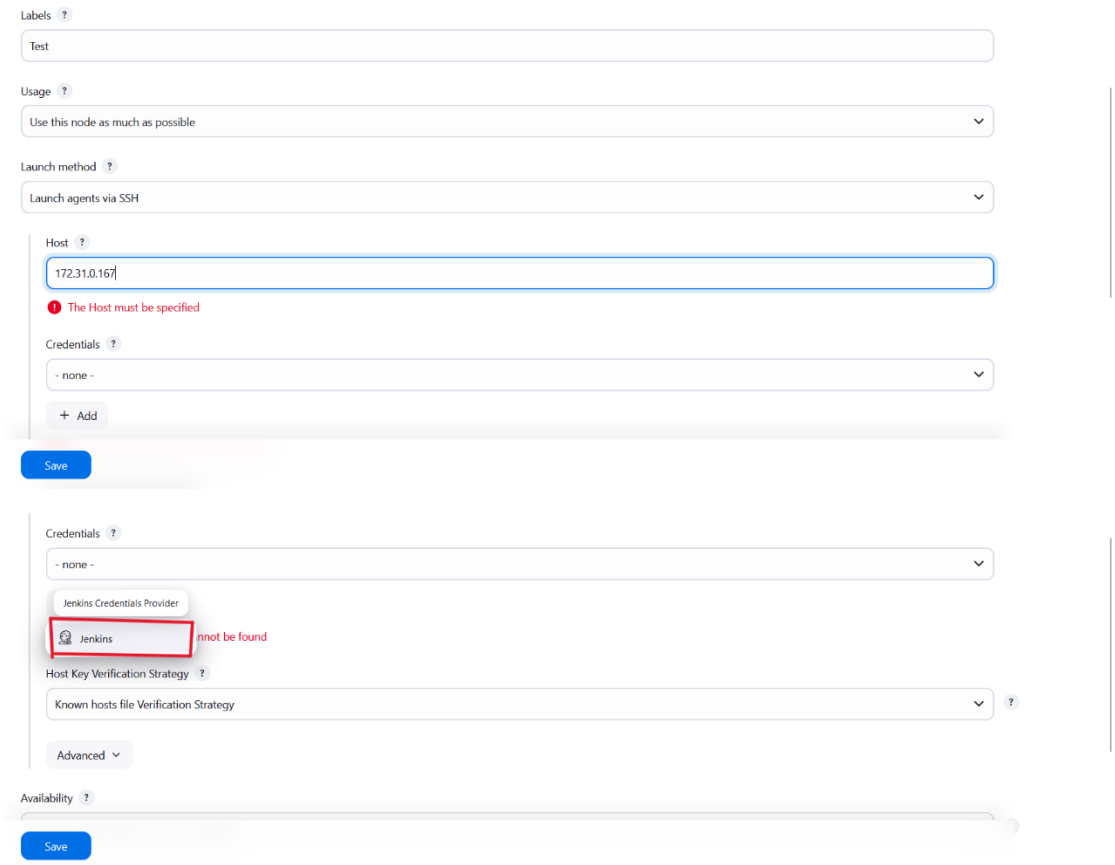
Remote root directory: /home/ubuntu/jenkins



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".



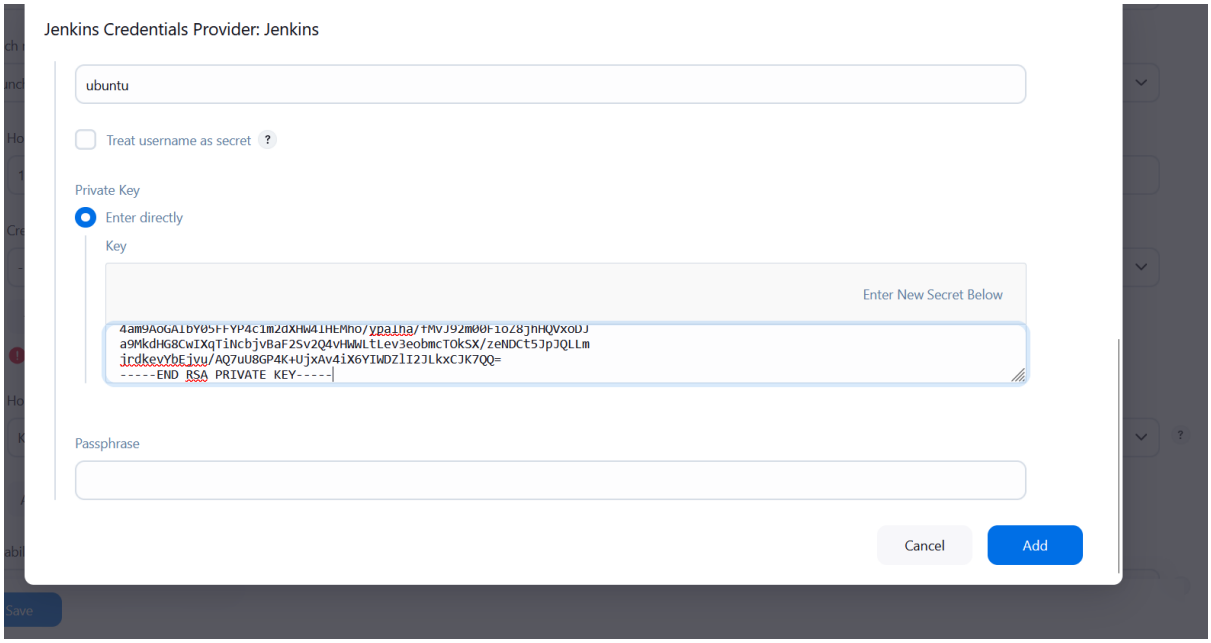
In **“Add Credentials”**, choose **“kind”** as **“SSH Username with private key”.**

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

ID & Description : PEM

Username: ubuntu

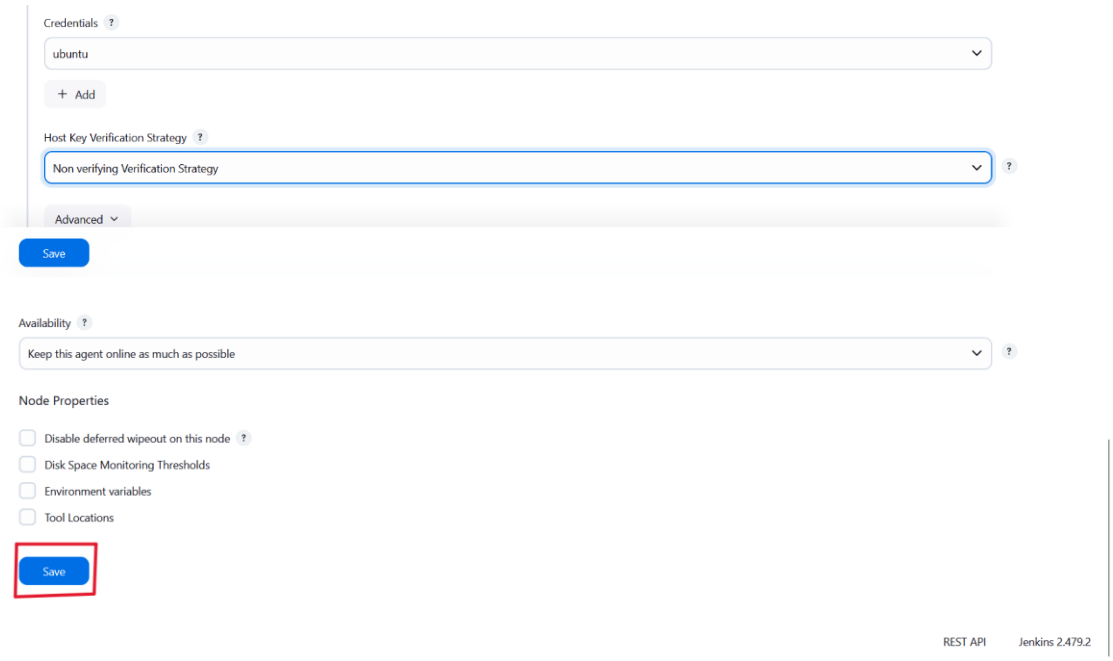
In **“Private Key”,** choose **“Enter Directly”. Click** on **“Add”**



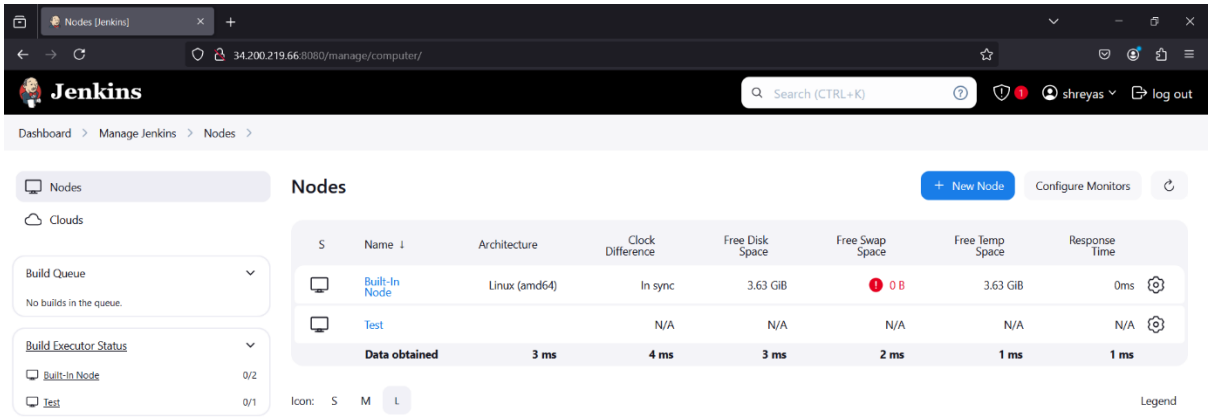
Click on **“Add”.**

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

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



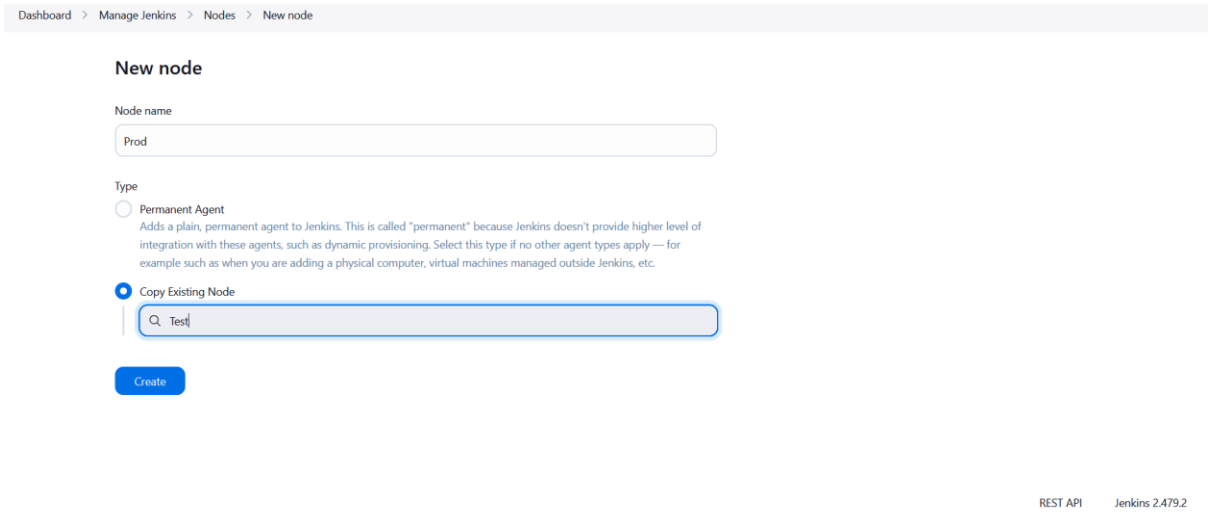
Step 5: “Test Node” has been **successfully launched** like this:



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”.



Click on **“Create”.**

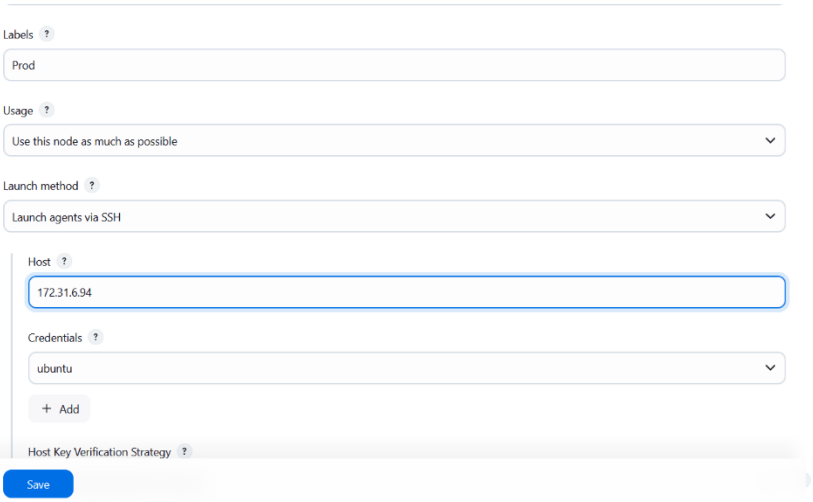
Step 3: Fill the following options here:

Labels: Prod

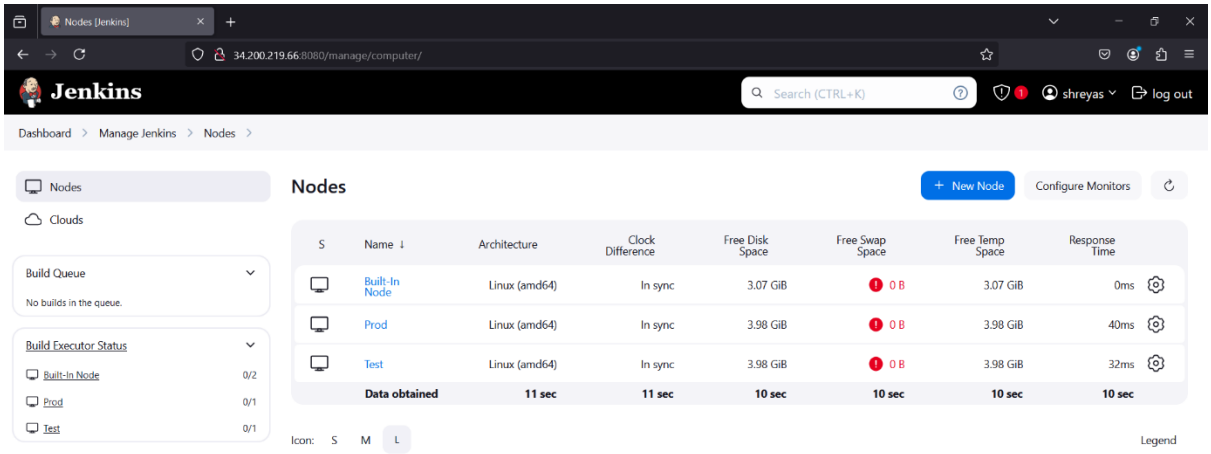
Host: 172.31.6.94

Remain all entries are as it is.

Click on **“Save”.**



Step 4: The "Prod" Node has been successfully launched.

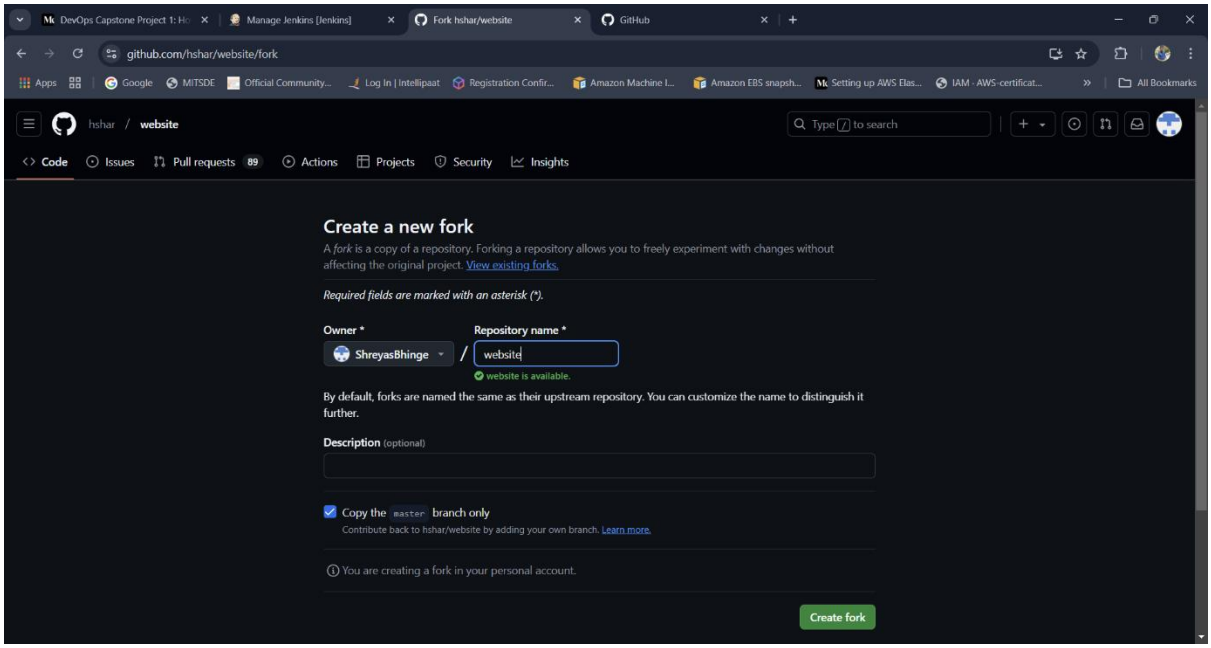


Problem (2) Solution: Git workflow has to be implemented

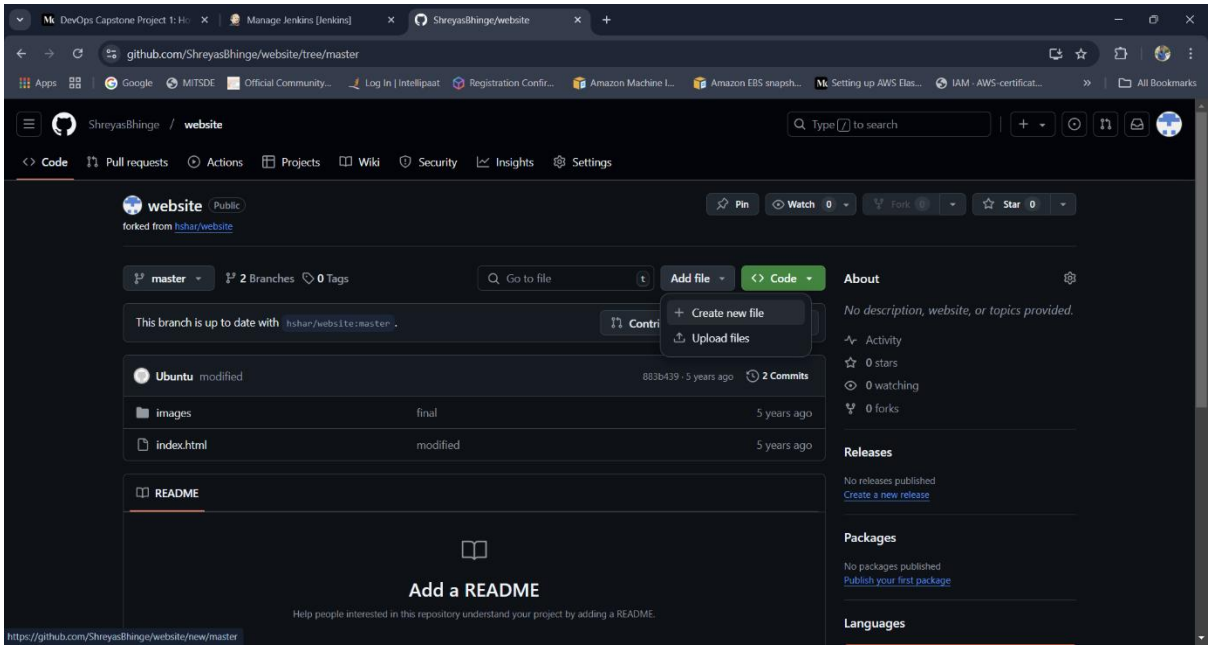
A. Fork the Given Repository

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

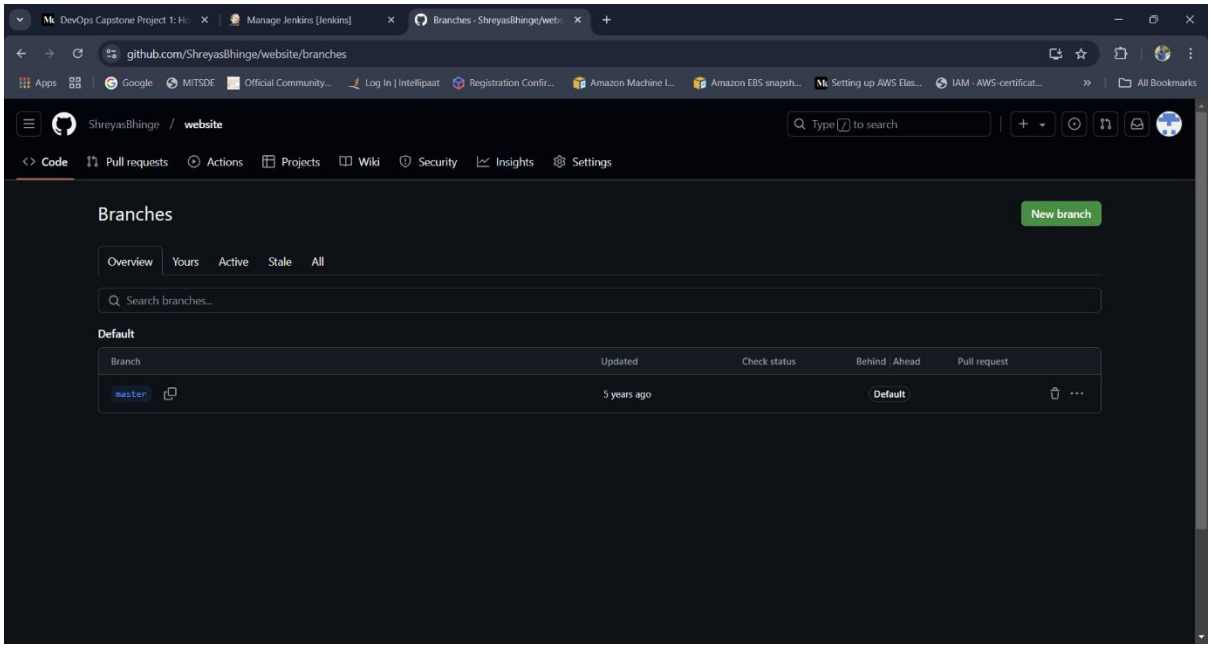
Now, the **"Fork" option** will be **shown. Click** on **"Fork". It** will **automatically fetch "website"** as **the "Repository name".**

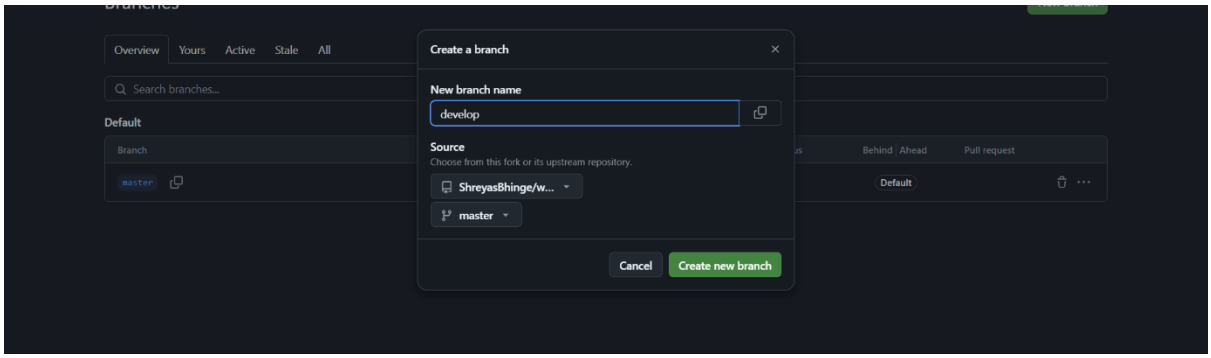


Given that **“Repository [website]”** has been **successfully forked.**



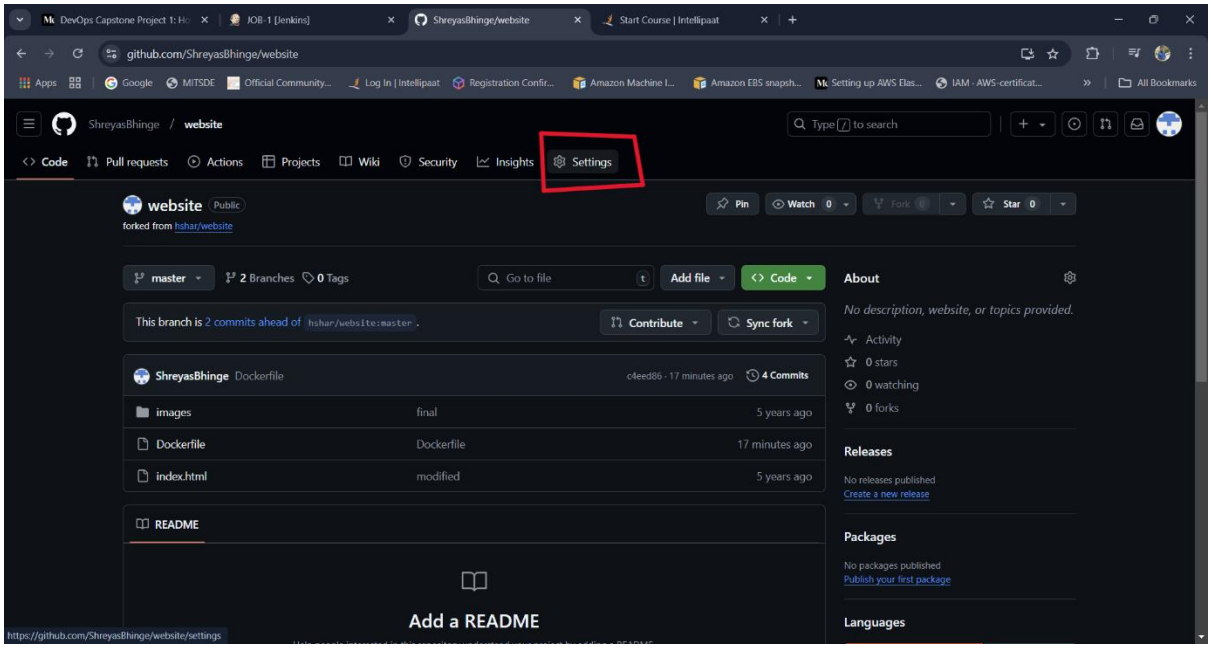
check the **branches. The “develop” branch** will be **successfully shown** here.

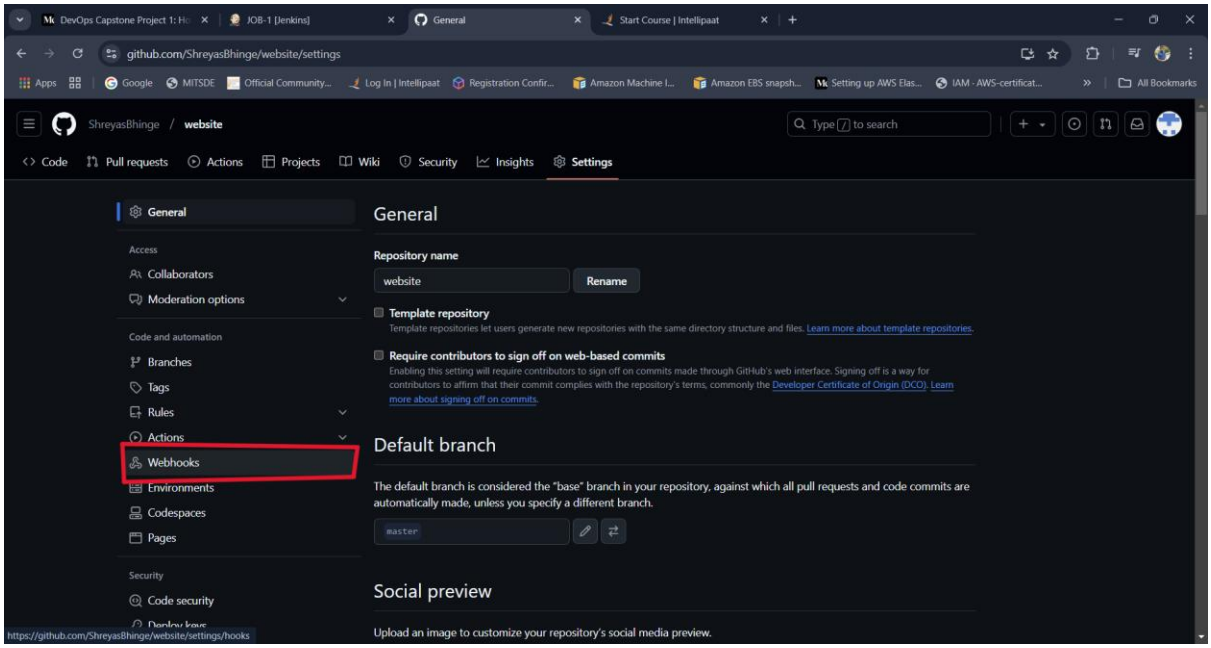




Create a Webhook for Trigger the Jobs

Step 1: Go to “Settings”.

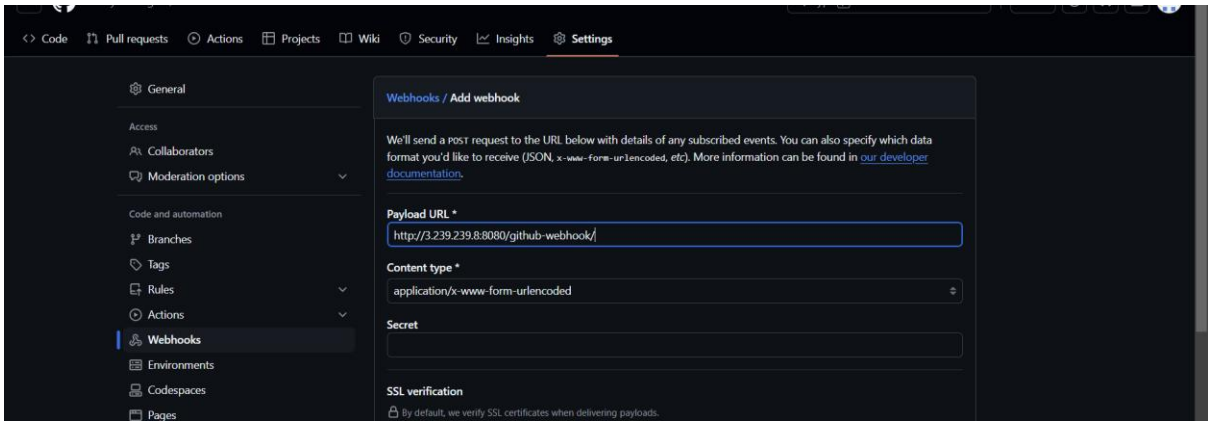




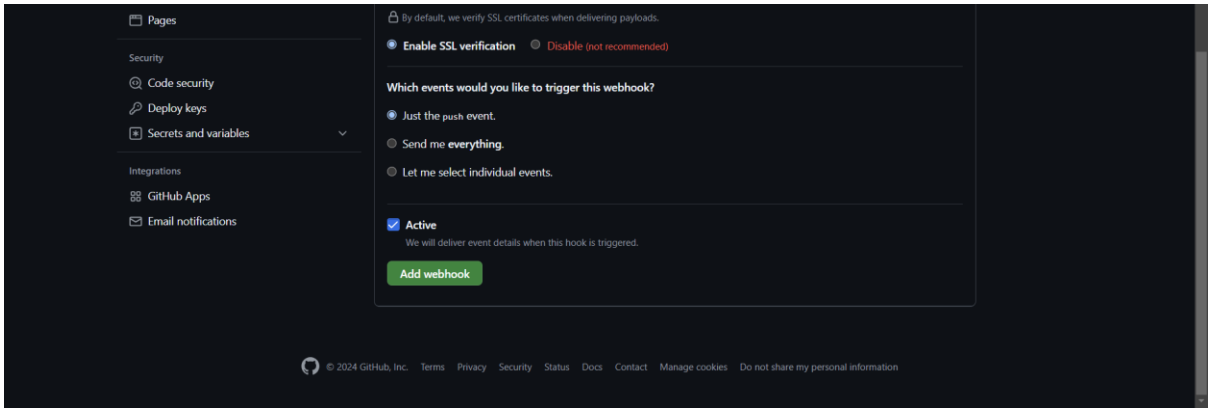
Step 4: Choose the following options here:

Payload URL: http://3.110.160.39:8080/github-webhook/

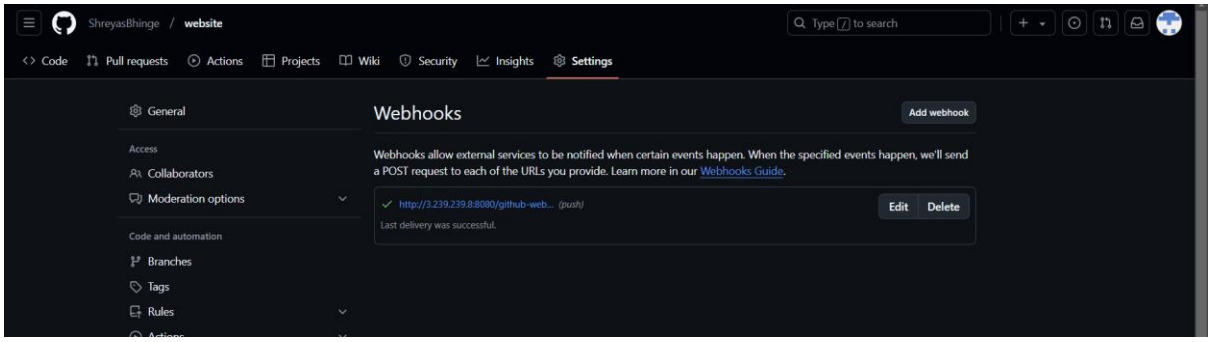
Which events would you like to trigger this webhook?: Send me everything



Click on **"Add webhook".**

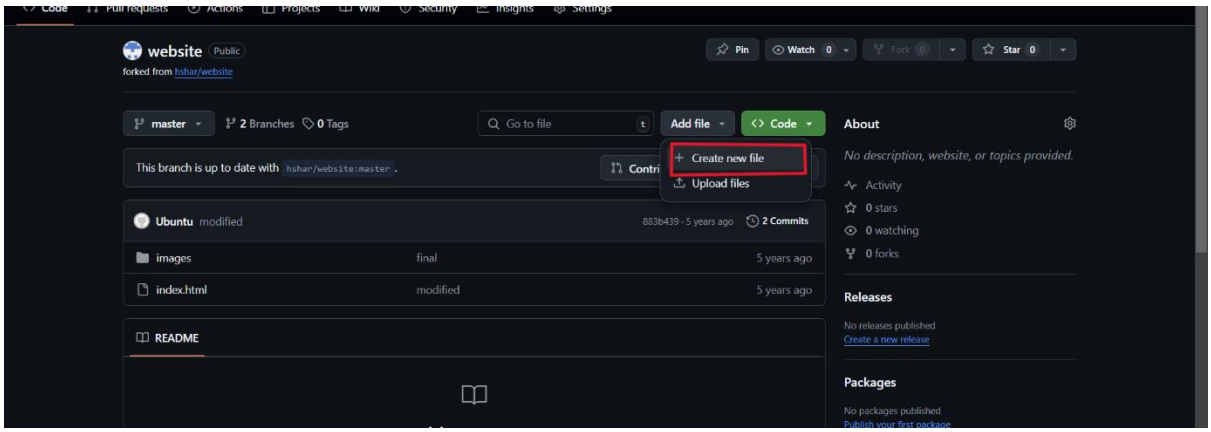


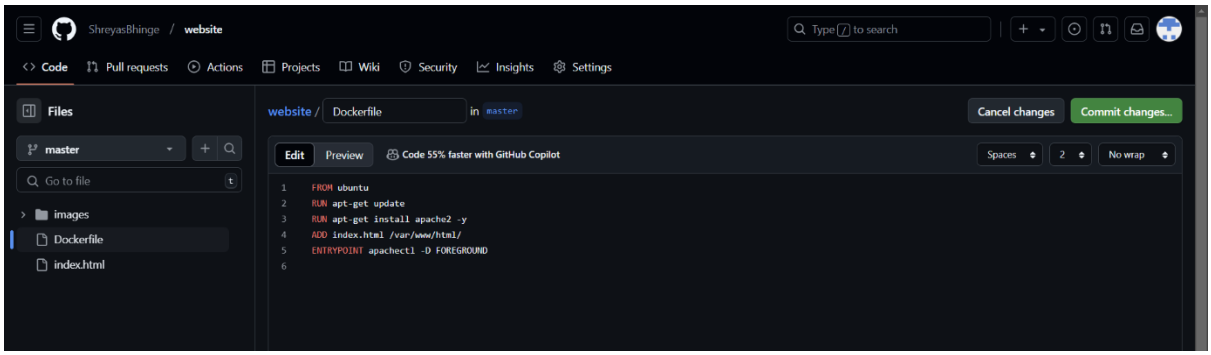
Step 5: Your webhook will be **successfully created.**



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





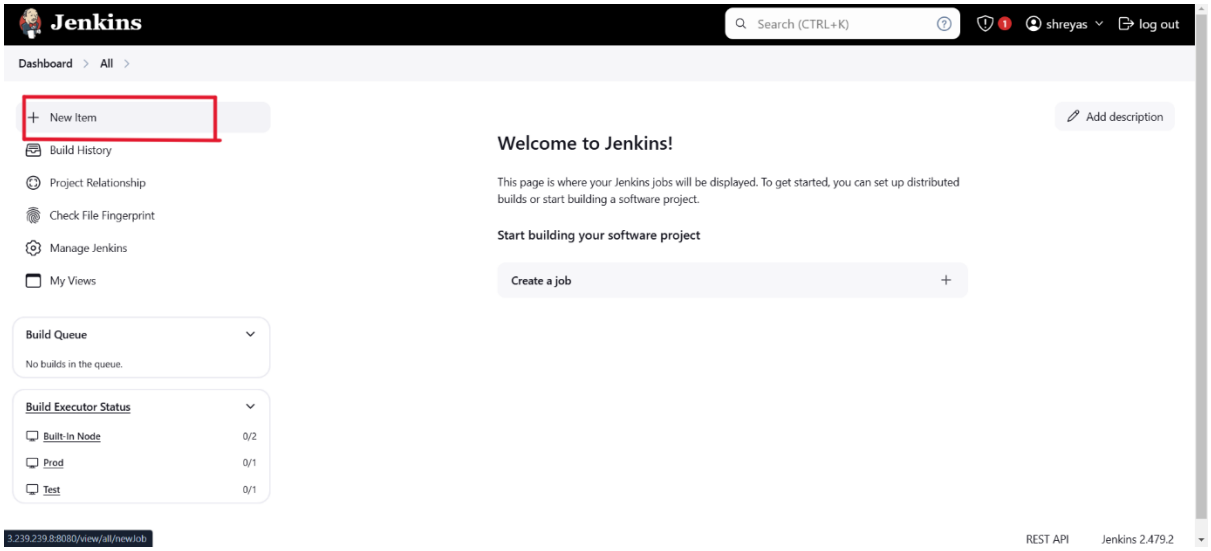
Problem (5) Solution: The above tasks should be defined in a Jenkins Pipeline with the following jobs:

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**

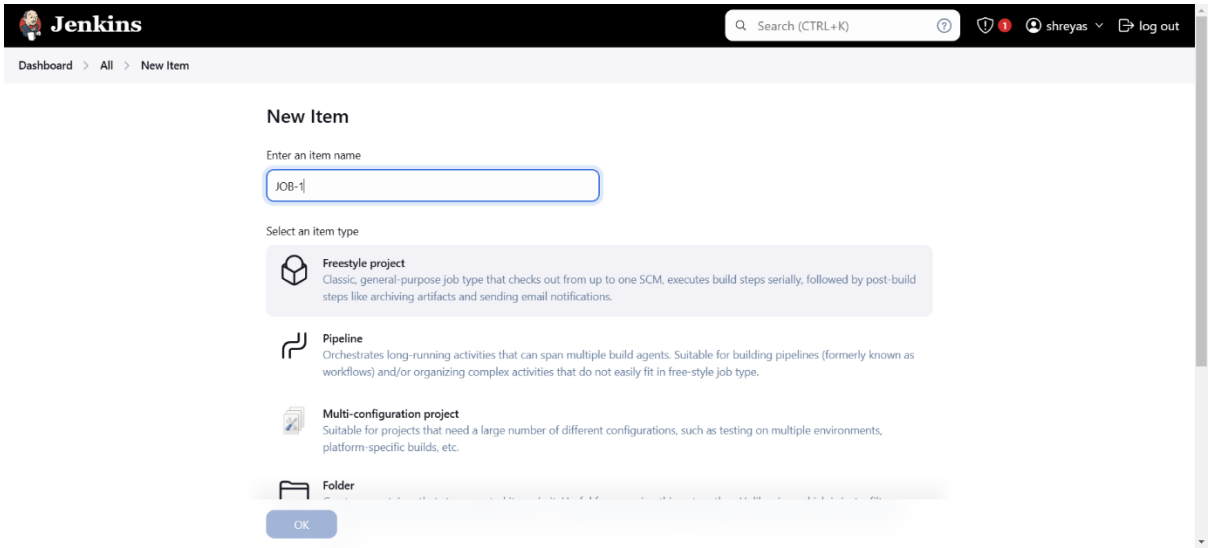


Step 2: Choose the following options here:

Enter an item name: JOB-1

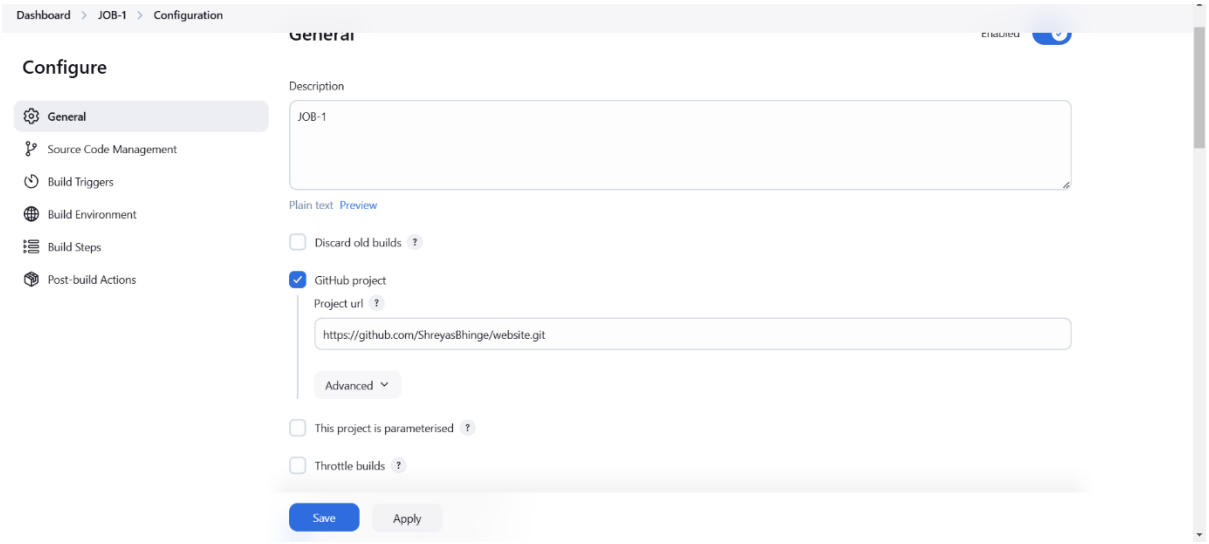
Option: Freestyle project

Click on "OK".



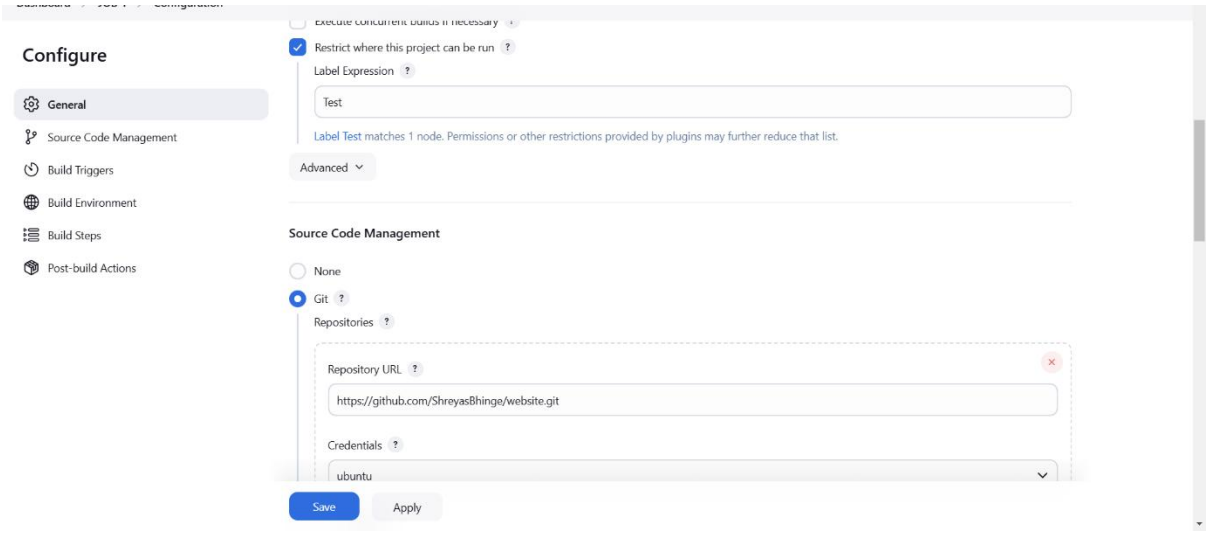
Step 3: Choose "Description" as "JOB-1"

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



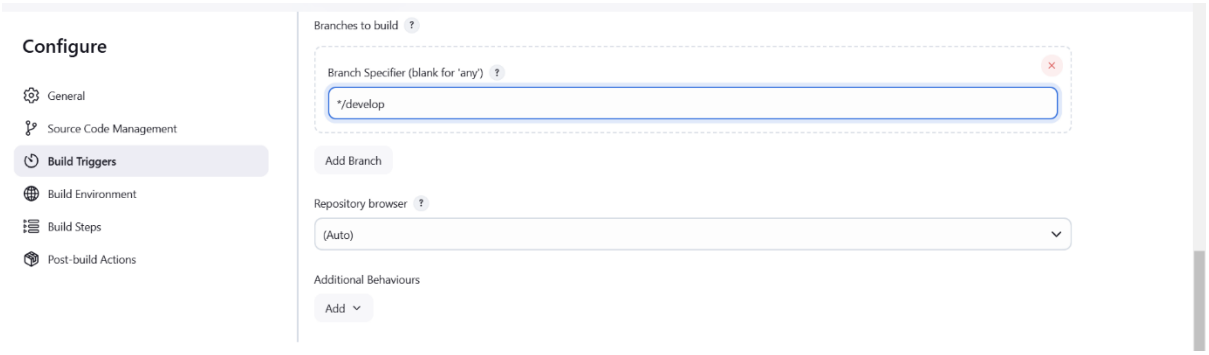
While Choose “Label Expression” as “Slave1”.

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



hoose “Credentials” as “none”.

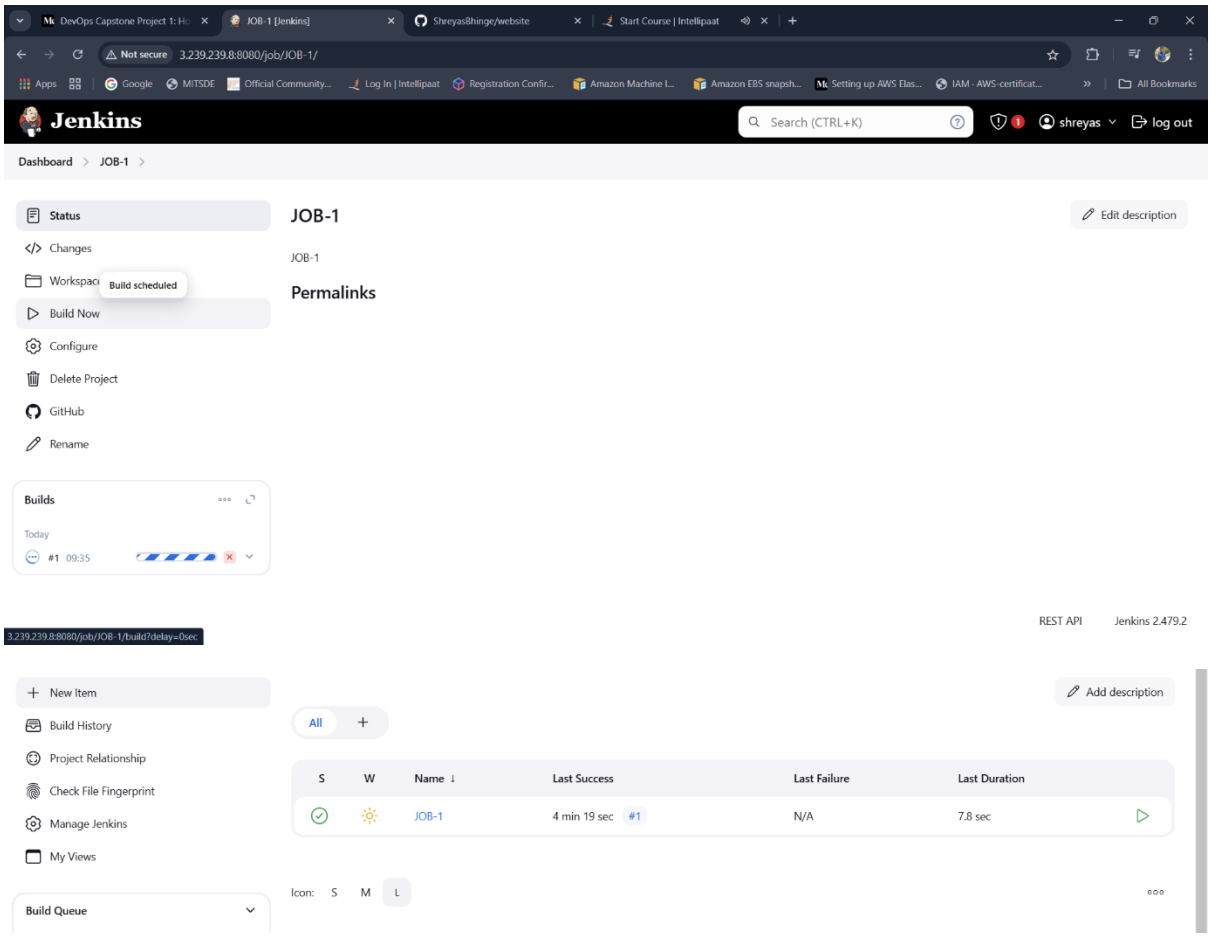
Step 6: Choose “Branch” as “develop” in “Branch Specifier”.



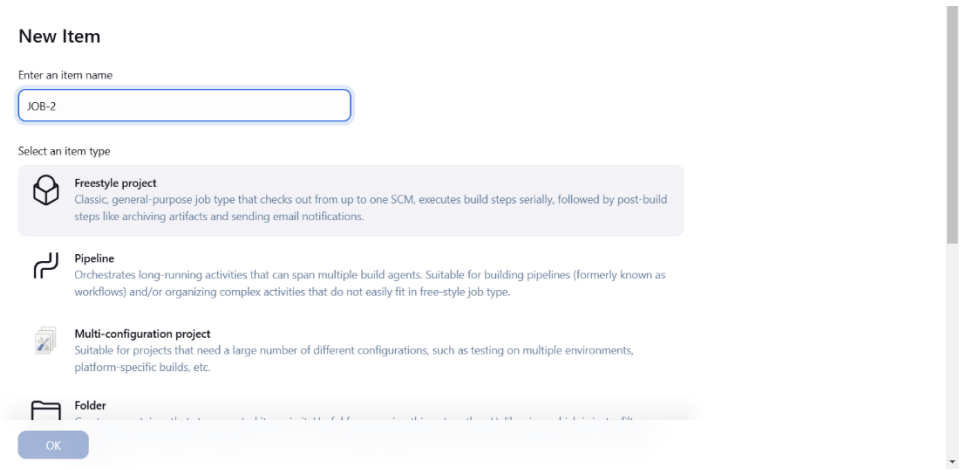
Step 7: Choose “GitHub hook trigger for GITScm polling” in “Build Triggers”.

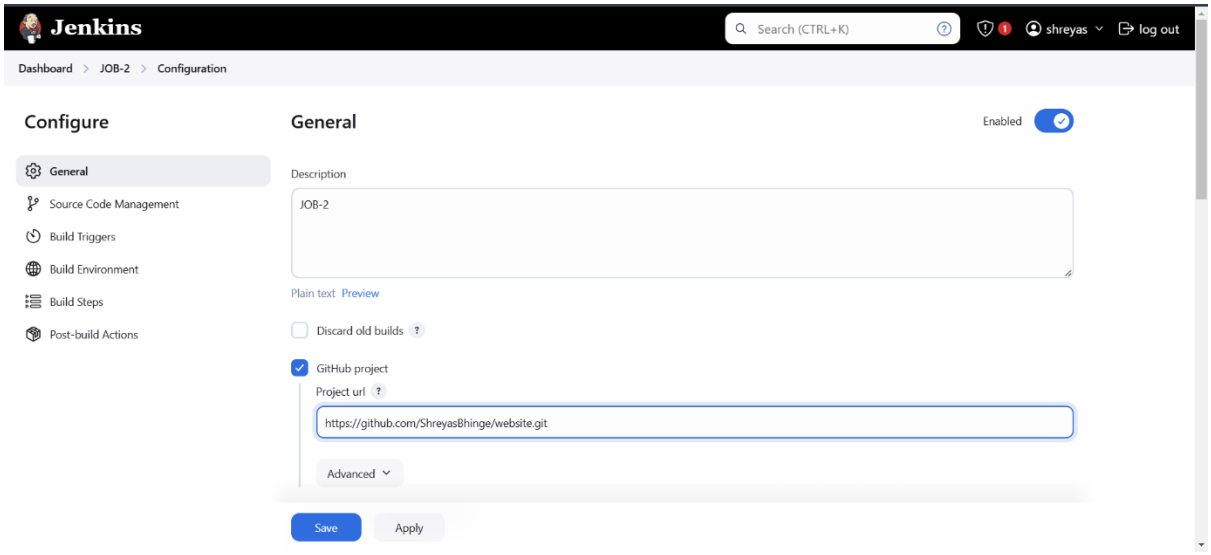
Step 8: Click on “Apply”. After clicking on “Apply”, choose “Save”.

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

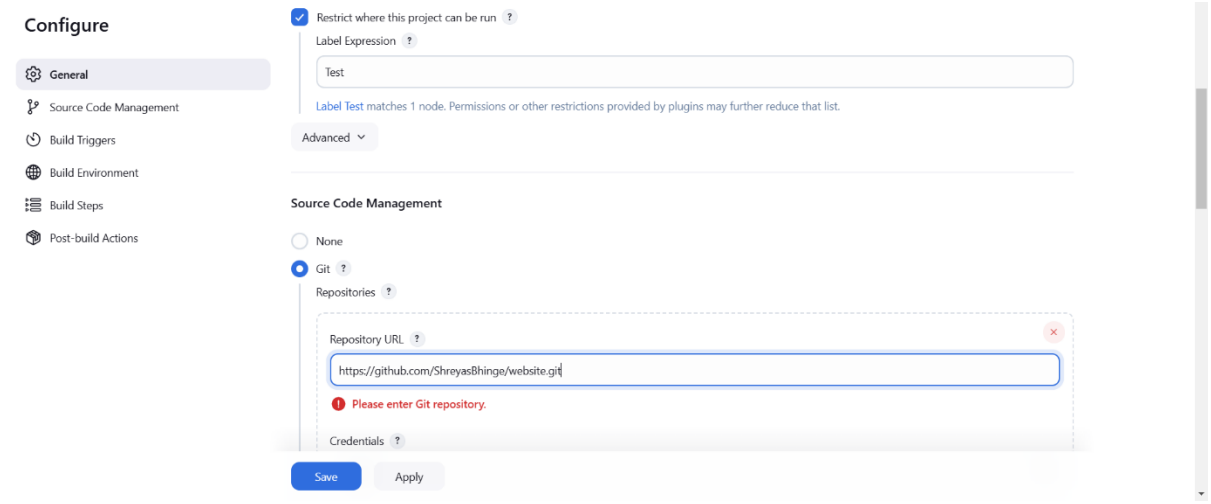


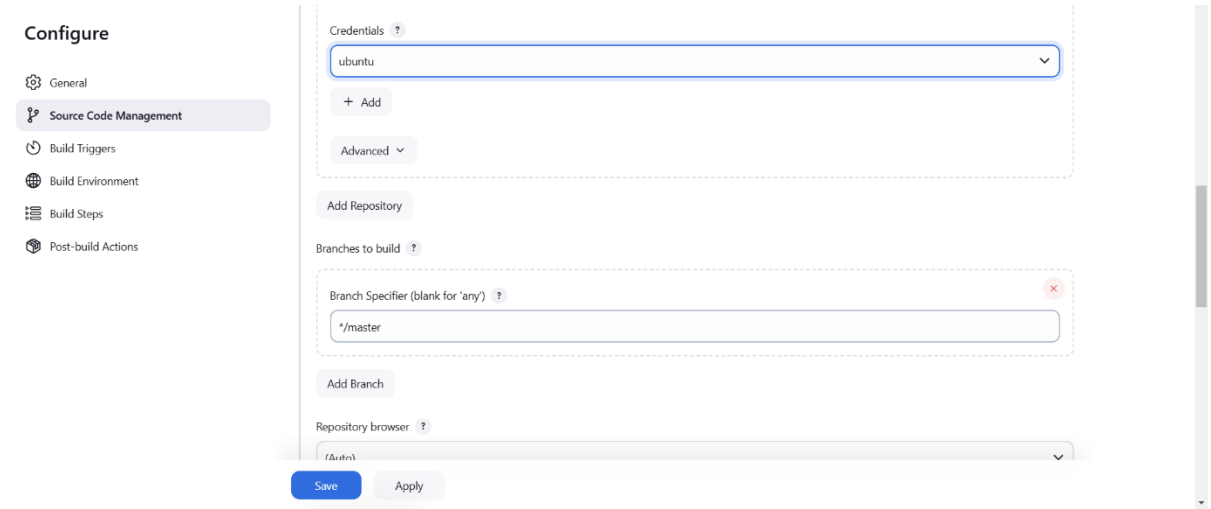
Creating JOB-2



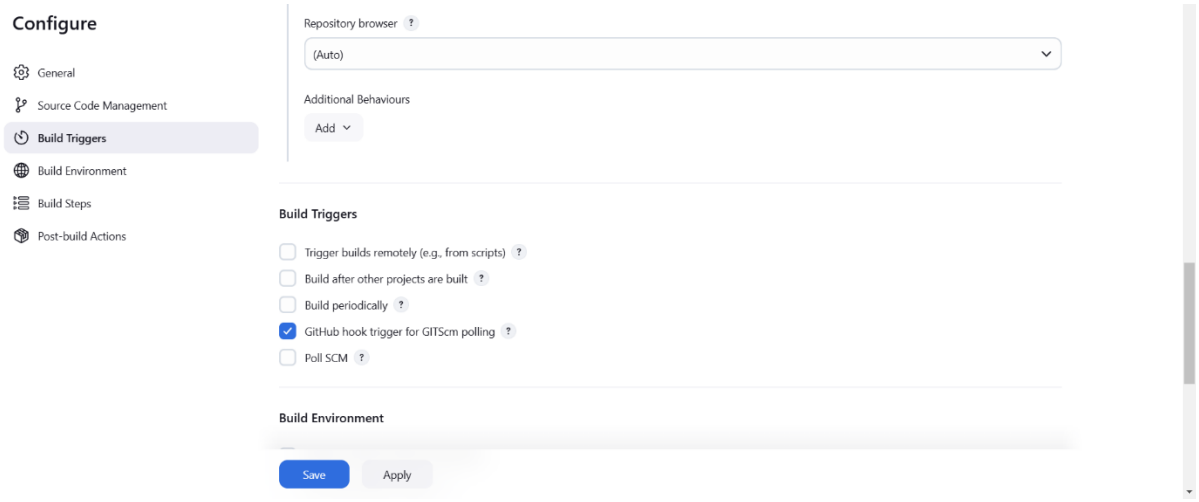


Label as Test



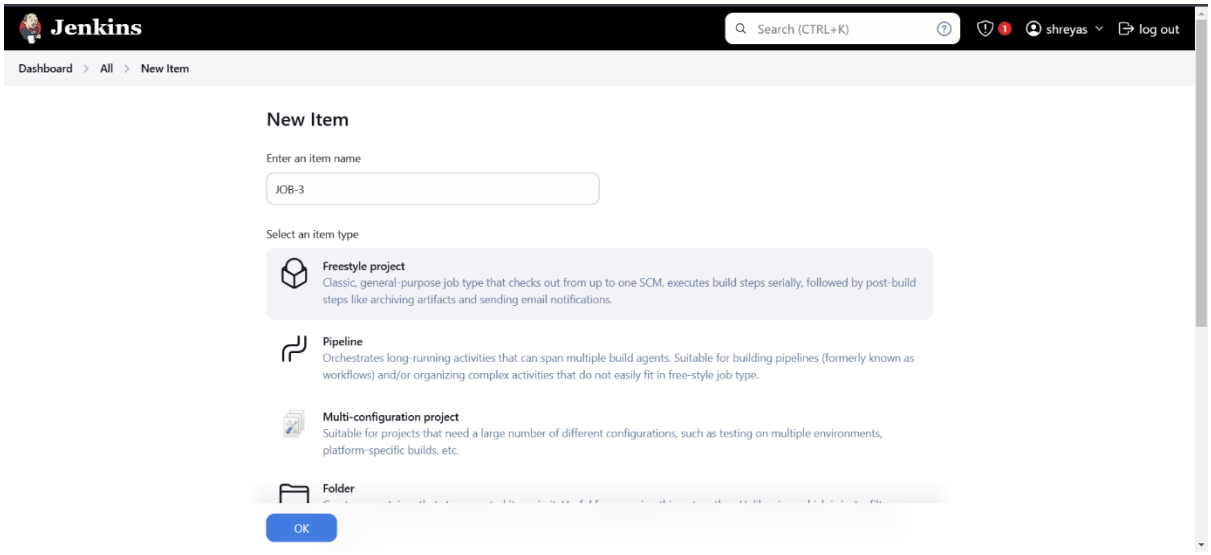


Save the job



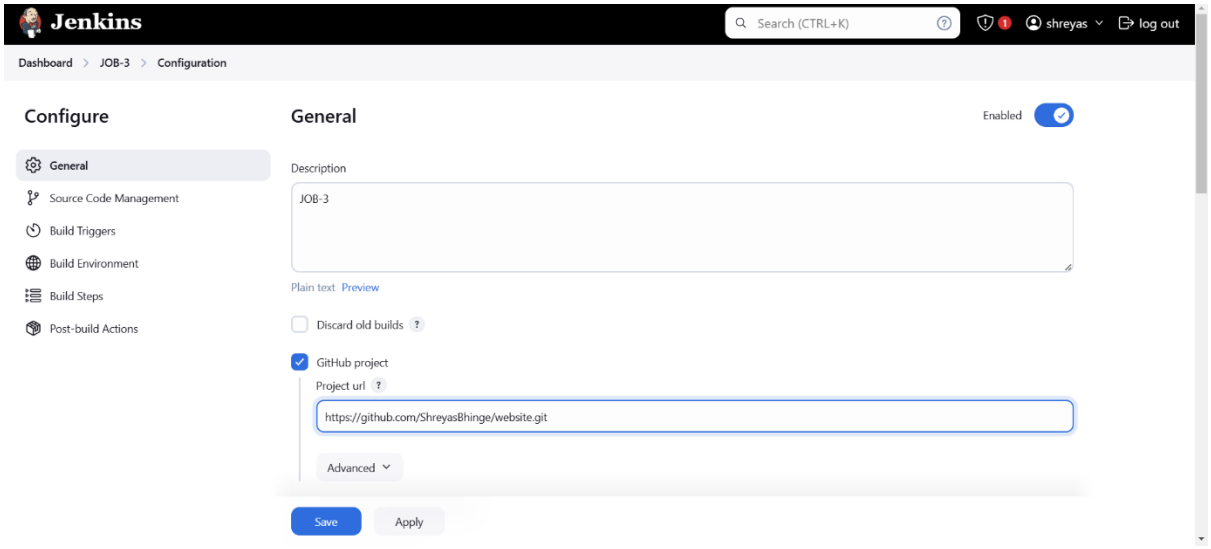
Enter an item name: JOB-3

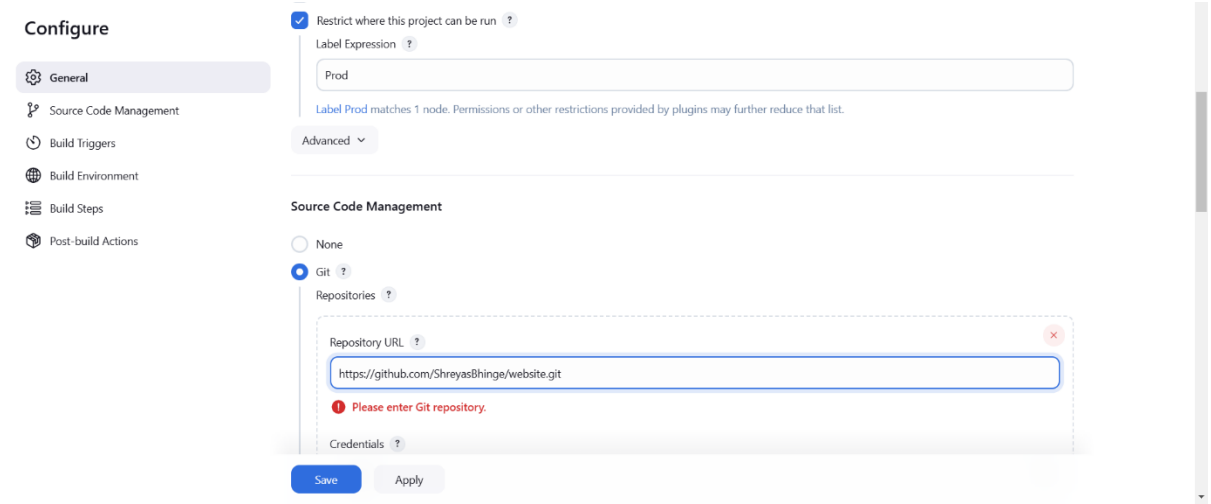
Option: Freestyle project



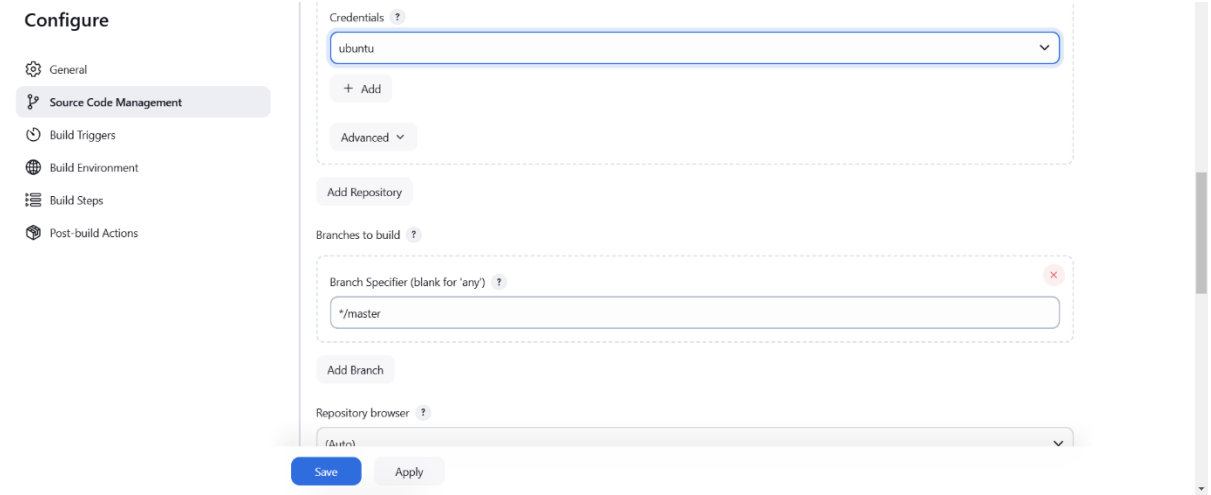
In "General", put the **description** as **"JOB-3"**

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





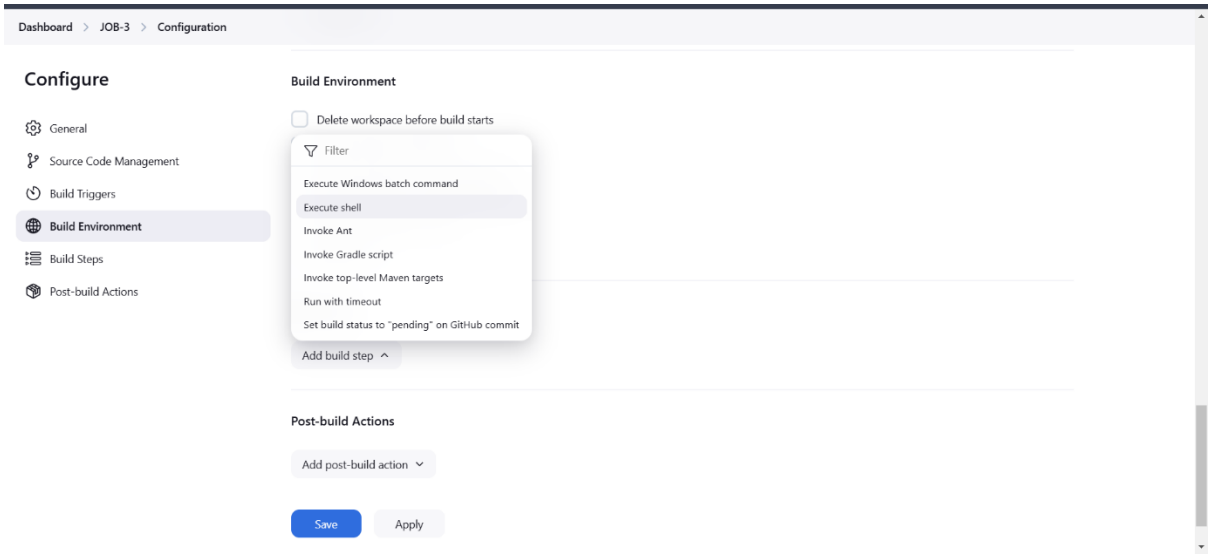
Remain other settings as **it is. Choose "Branch Specifier (blank for 'any')** as **"*/master".**



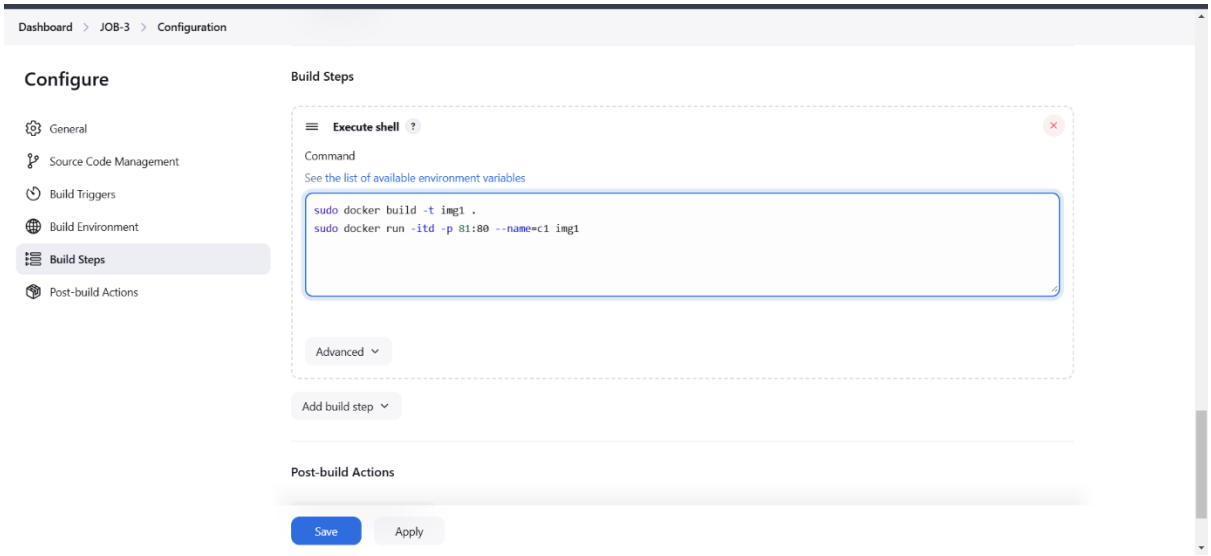
Click on "Build Steps".

Click on **"Add build step".**

Click on **"Execute Shell"**

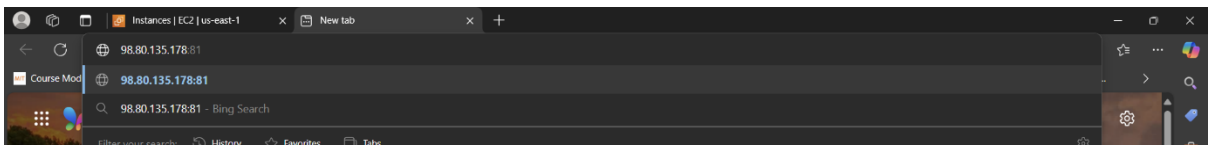


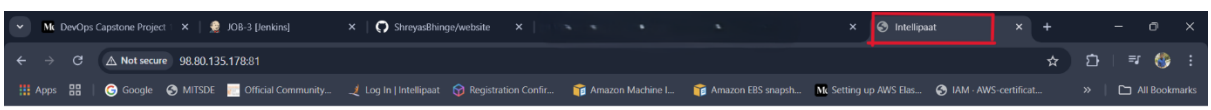
Put these **commands** to **build** a **container** on **port 81:**



Click on **“Apply>Save”**

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





Hello world!