Report on the CI/CD implementation in the NextJS web application

Sanjeeb Nepal – C0923568

Nabin Shrestha – C0939681

Diwash Shrestha – C0942206

Divas Kumar Mahato – C0941399

Divash Raj Upreti – C0940995

DevOps for Cloud Computing, Lambton College

CBD 2244- DevOps Fundamentals for Canadian Enterprises

Mohammad Salim

December 13, 2024

CI/CD implementation in the NextJS web application

In this project, we set out to implement the DevOps tools and techniques to deploy a NextJS application. DevOps means reducing the time required to commit to deployment while maintaining high quality. So, we are going to implement those tools and techniques like automated testing and the addition of a staging environment in between the development and production environment.

The GitHub link for the project is:

https://github.com/sanjeebnepal/DEVOPS-final-2.git

The tools used:

- Vagrant
- Node
- Jenkins
- Docker
- Kubernetes
- Git
- Github
- Cyprus
- Chromium

Vagrant

For the first step, we will set up vagrant machines. To do that, we use the Vagrant file along with scripts to set up the following VMs with Debian OS:

1. 1 VM for running Jenkins
2. 3 VMs for staging environment
3. 3 VMs for production environment.

We have tried to create the exact replica environment between staging and production.

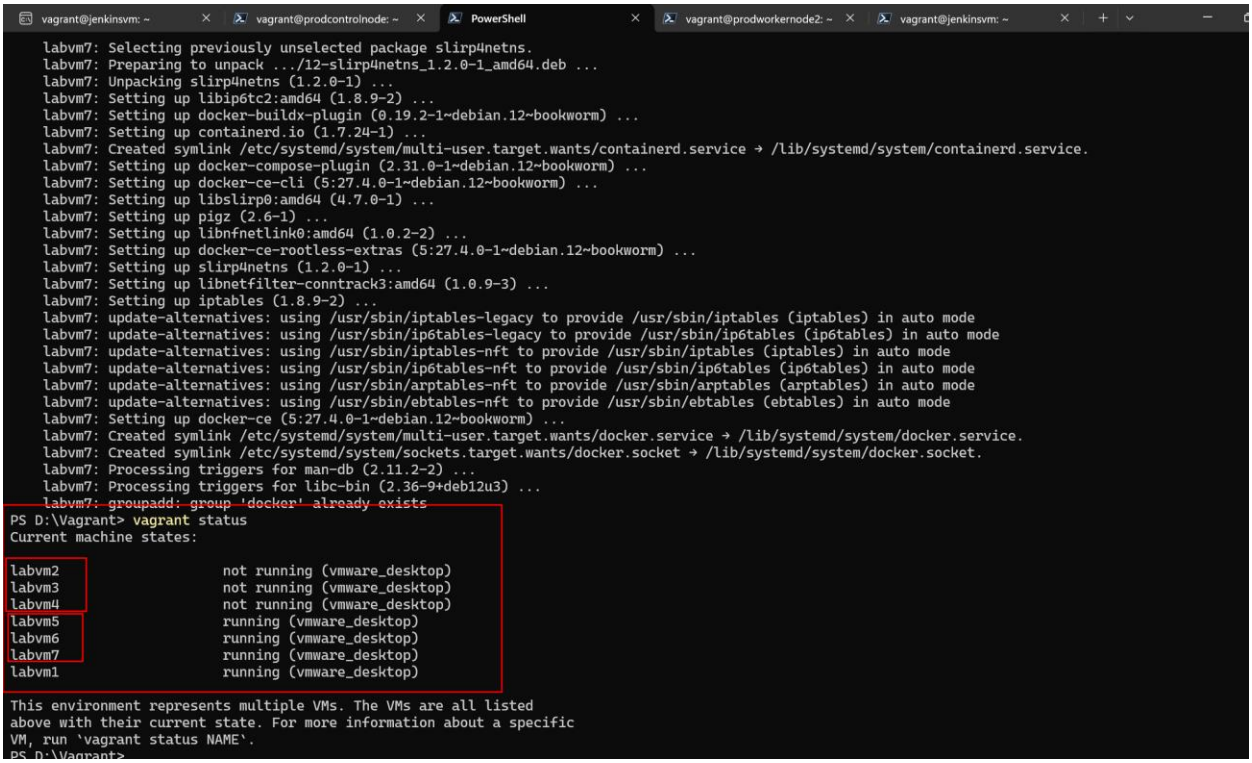
We have the required files for vagrant VMs setup in the GitHub link:
https://github.com/sanjeebnepal/Vagrant-script-jenkinsdockerkubernetes.git

To create the Virtual Machines, we have to clone the repository, install VMware Workstation in our system, and use the command:

vagrant init

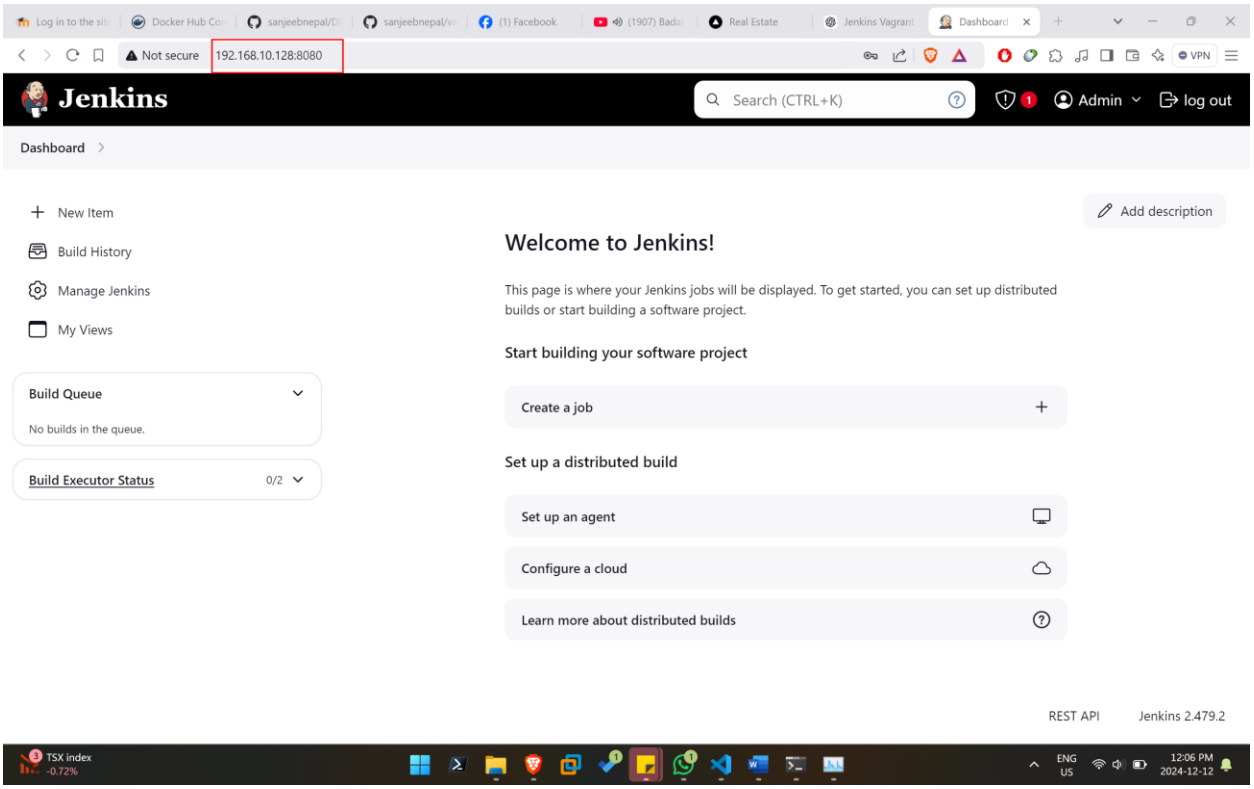
vagrant up.

We can check if they were created successfully by using the command **vagrant status**.

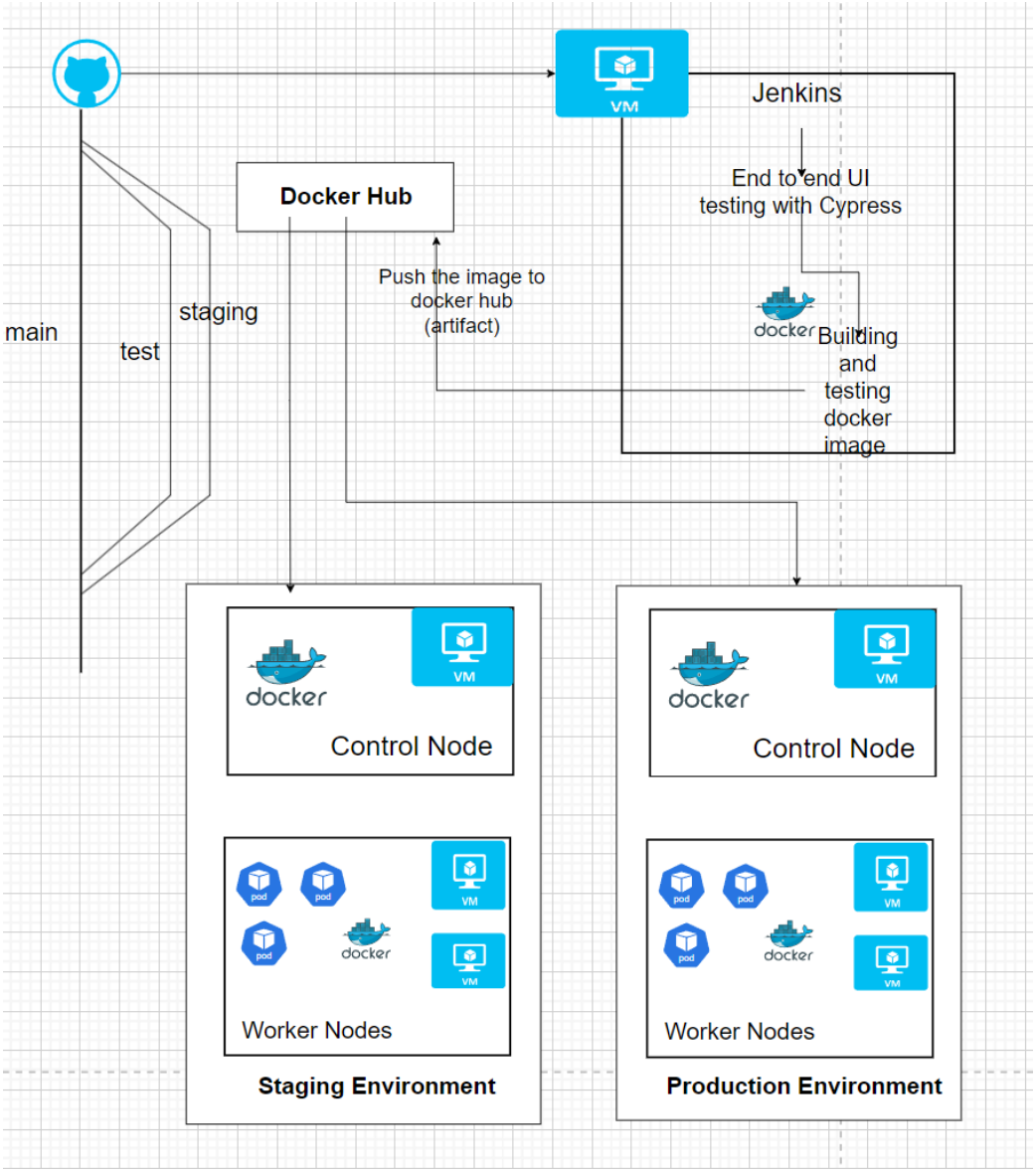


Jenkins

Now, we check the Jenkins setup. To do that, we enter the http://Jenkins-VM-IP:8080.


Not secure
192.168.10.128:8080
Jenkins
Search (CTRL+K)
Admin
log out
Dashboard
New Item
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
0/2
Add description
Welcome to Jenkins!
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.
Start building your software project
Create a job
Set up a distributed build
Set up an agent
Configure a cloud
Learn more about distributed builds
REST API
Jenkins 2.479.2

Architecture



Pipelines

In this project, we are developing three pipelines:

1. **Test Pipeline**: Here, we will do end-to-end testing of the code using the Cyprus tool, then build docker image, run it locally, test it and if successful push it to docker hub. Also, this pipeline triggers the staging pipeline on its completion.

2. **Staging Pipeline**: In this pipeline, we will test if the docker image can run in among the Kubernetes worker and control node. It is the replica of the production environment.
3. **Production pipeline**: We will deploy the app using the docker image from the docker hub in three nodes using Kubernetes after we have confirmed that everything works in staging pipeline.

Git setup

SSH authentication

Before we build the pipelines, we should set up SSH authentication from GitHub to the Jenkins VM. For this, we need to create a new SSH key.

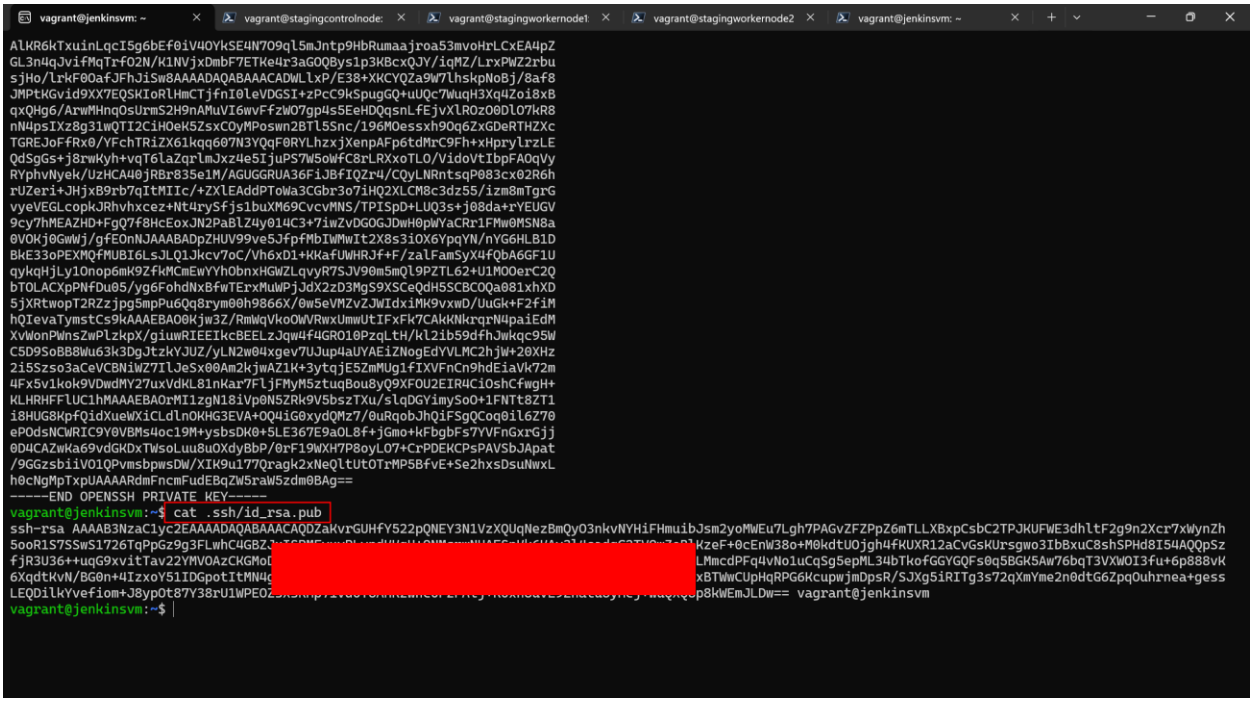
ssh-keygen -t rsa -b 4096 "email@gmail.com"

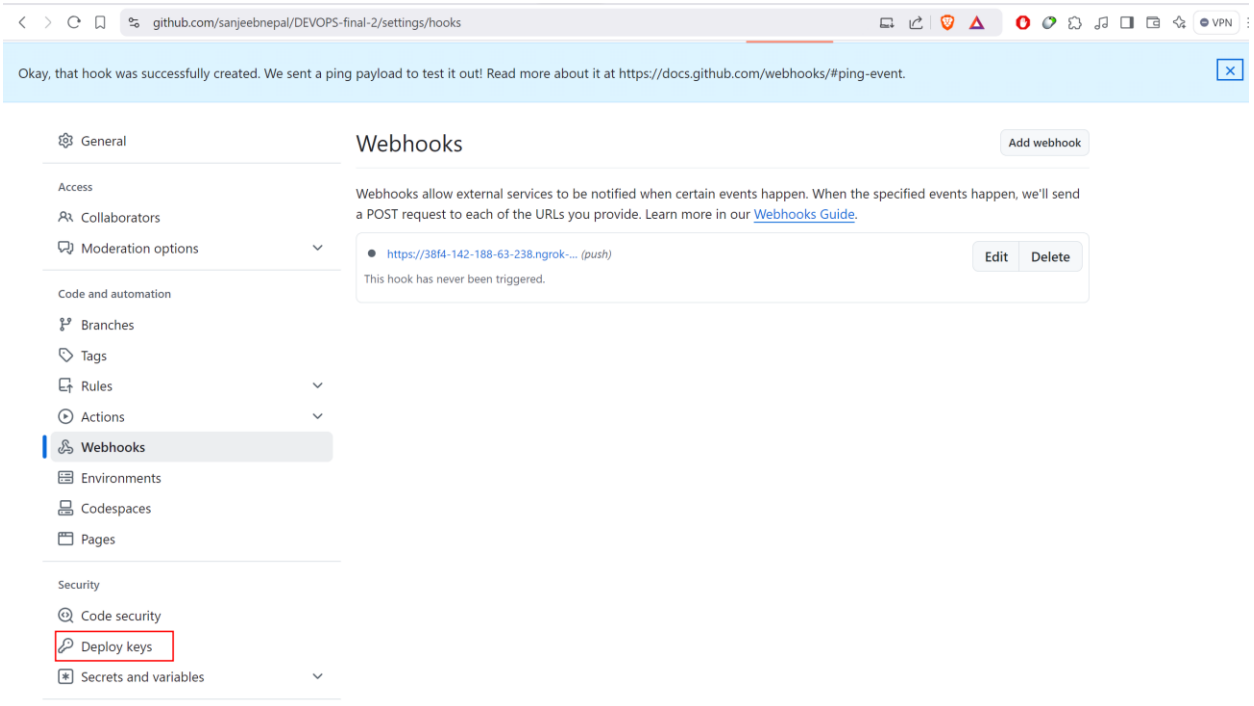
We will execute this command in VM running Jenkins. We have to copy the content of the id_rsa.pub from the .ssh directory in Jenkins VM and copy it to deploy keys in my GitHub repository.

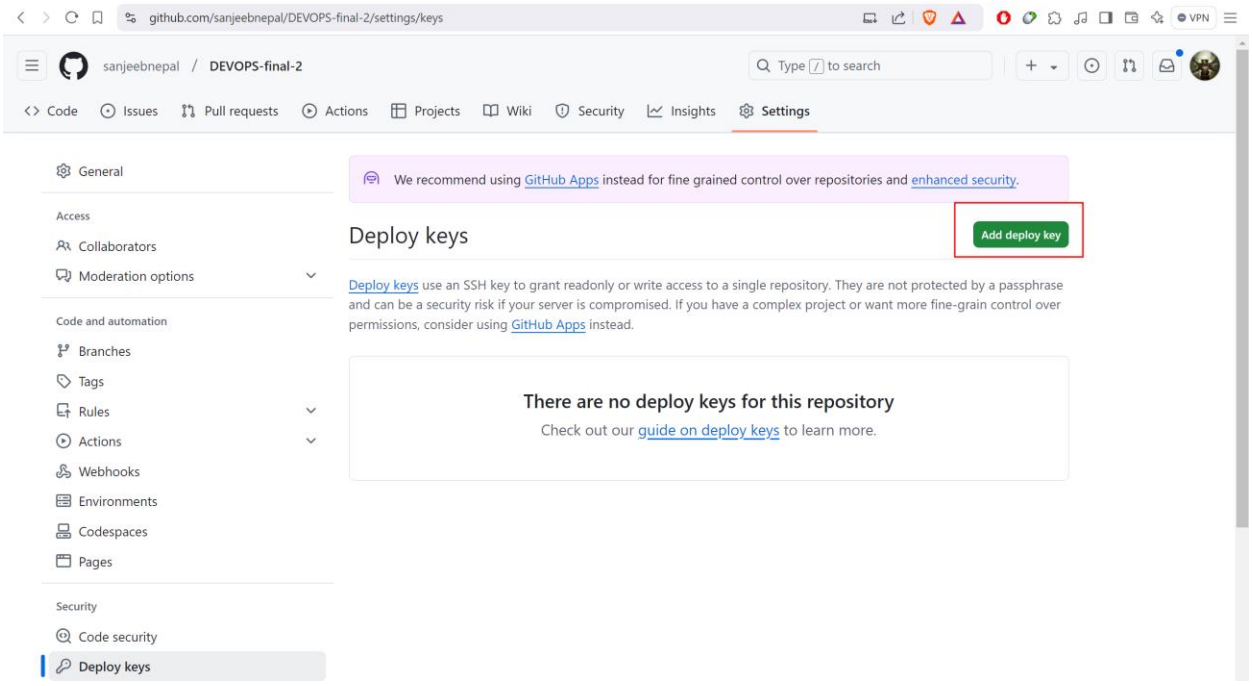
We will verify the connection with the command:

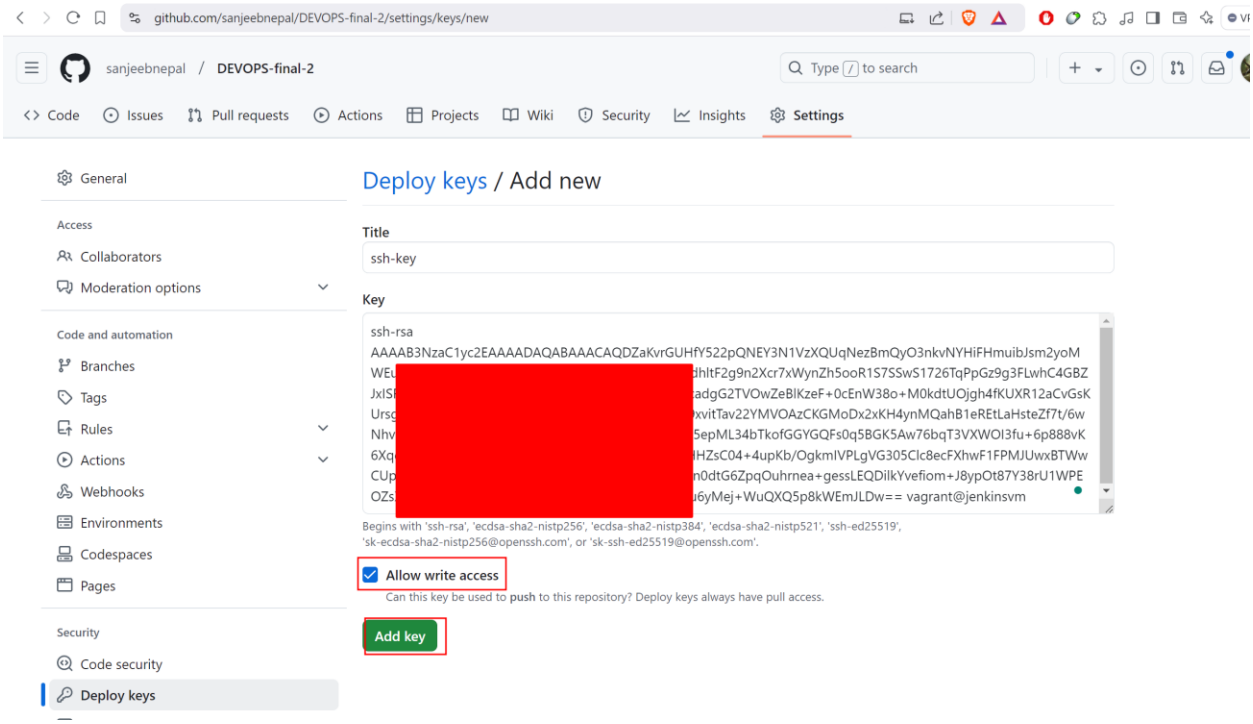
ssh -T git@github.com

in the VM running Jenkins.


vagrant@jenkinsvm: ~
vagrant@stagingcontrolnode:
vagrant@stagingworkernode1
vagrant@stagingworkernode2
vagrant@jenkinsvm: ~
AlkR6kTxuinLqcI5g6bEf0iV4OYkSE4N7O9ql5mJntp9HbRumaajroa53mvoHrLCxEA4pZ
GL3n4qJvifMqTrfO2N/K1NVjxDmbF7ETKe4r3aGOQBys1p3KBcxQJY/iqMZ/LrxPWZ2rbu
sjHo/lrkF0OafJFhJiSw8AAAADAQABAAACADWLlxP/E38+XKCYQZa9W7lhskpNoBj/8af8
JMPtKGvid9XX7EQSkIoRlHmCTjfnI0leVDGSI+zPcC9kSpugGQ+uUQc7WuqH3Xq4Zoi8xB
qxQHg6/ArwMHnqOsUrmS2H9nAMuVI6wvFfzWO7gp4s5EeHDQqsnLfEjvXlROzO0DlO7kR8
nN4psIXz8g31wQTI2CiHOeK5ZsxCOyMPoswn2BTl5Snc/196MOessxh9Oq6ZxGDeRTHZXc
TGREJoFfRx0/YFchTRiZX61kqq607N3YQqF0RYLhzxjXenpAFp6tdMrC9Fh+xHprylrzLE
QdSgGs+j8rwKyh+vqT6laZqrlmJxz4e5IjuPS7W5oWfC8rLRXxoTLO/VidoVtIbpFAOqVy
RYphvNyek/UzHCA40jRBr835e1M/AGUGGRUA36FiJBfIQZr4/CQyLNRntsqP083cx02R6h
rUZeri+JHjxB9rb7qItMIIc/+ZXlEAddPToWa3CGbr3o7iHQ2XLCM8c3dz55/izm8mTgrG
vyeVEGLcopkJRhvhxcez+Nt4rySfjs1buXM69CvcvMNS/TPISpD+LUQ3s+j08da+rYEUGV
9cy7hMEAZHD+FgQ7f8HcEoxJN2PaBlZ4y014C3+7iwZvDGOGJDwH0pWYaCRr1FMw0MSN8a
0VOKj0GwWj/gfEOnNJAAABADpZHUV99ve5JfpfMbIWMwIt2X8s3iOX6YpqYN/nYG6HLB1D
BkE33oPEXMQfMUBI6LsJLQ1Jkcv7oC/Vh6xD1+KKafUWHRJf+F/zalFamSyX4fQbA6GF1U
qykqHjLy1Onop6mK9ZfkMCmEwYYhObnxHGWZLqvyR7SJV90m5mQl9PZTL62+U1MOOerC2Q
bTOLACXpPNfDu05/yg6FohdNxBfwTErxMuWPjJdX2zD3MgS9XSCeQdH5SCBCOQa081xhXD
5jXRtwopT2RZzjpg5mpPu6Qq8rym00h9866X/0w5eVMZvZJWIdxiMK9vxwD/UuGk+F2fiM
hQIevaTymstCs9kAAAEBAO0kjw3Z/RmWqVkoOWVRwxUmwUtIFxFk7CAkKNkrqrN4paiEdM
XvWonPWnsZwPlzkpX/giuwRIEEIkcBEELzJqw4f4GRO10PzqLtH/kl2ib59dfhJwkqc95W
C5D9SoBB8Wu63k3DgJtzkYJUZ/yLN2w04xgev7UJup4aUYAEiZNogEdYVLMC2hjW+20XHz
2i5Szso3aCeVCBNiWZ7IlJeSx00Am2kjwAZ1k+3ytqjE5ZmMUg1fIXVFnCn9hdEiaVk72m
4Fx5v1kok9VDwdMY27uxVdKL81nKar7FljFMyM5ztuqBou8yQ9XFOU2EIR4CiOshCfwgH+
KLHRHFFlUC1hMAAAEBAOrMI1zgN18iVp0N5ZRk9V5bszTXu/slqDGYimySoO+1FNTt8ZT1
i8HUG8KpfQidXueWXiCLdlnOKHG3EVA+OQ4iG0xydQMz7/0uRqobJhQiFSgQCoq0il6Z70
ePOdsNCWRIC9Y0VBMs4oc19M+ysbsDK0+5LE367E9aOL8f+jGmo+kFbgbFs7YVFnGxrGjj
0D4CAZwKa69vdGKDxTWsoLuu8uOXdyBbP/0rF19WXH7P8oyLO7+CrPDEkCPsPAVSbJApat
/9GGzsbiiVO1QPvmsbpwsDW/XIk9u177Qragk2xNeQltUtOTrMP5BfvE+Se2hxsDsuNwxL
h0cNgMpTxpUAAAARdmFncmFudEBqZW5raW5zdm0BAg==
-----END OPENSSH PRIVATE KEY-----
vagrant@jenkinsvm:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDZaKvrGUHfY522pQNEY3N1VzXQUqNezBmQyO3nkvNYHiFHmuibJsm2yoMWEu7Lgh7PAGvZFZPpZ6mTLLXBxpCsbC2TPJKUFWE3dhltF2g9n2Xcr7xWynZh
5ooR1S7SSwS1726TqPpGz9g3FLwhC4GBZ
fjR3U36++uqG9xvitTav22YMVOAzCKGMoD
6XqdtKvN/BG0n+4IzxoY51IDGpotItMN4
LEQDilkYvefiom+J8ypOt87Y38rU1WPEO
vagrant@jenkinsvm:~$


github.com/sanjeebnepal/DEVOPS-final-2/settings/hooks
Okay, that hook was successfully created. We sent a ping payload to test it out! Read more about it at https://docs.github.com/webhooks/#ping-event.
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
Code security
Deploy keys
Secrets and variables
Webhooks
Add webhook
Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.
https://38f4-142-188-63-238.ngrok-... (push)
This hook has never been triggered.
Edit
Delete


github.com/sanjeebnepal/DEVOPS-final-2/settings/keys
sanjeebnepal / DEVOPS-final-2
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
Code security
Deploy keys
We recommend using GitHub Apps instead for fine grained control over repositories and enhanced security.
Deploy keys
Add deploy key
Deploy keys use an SSH key to grant readonly or write access to a single repository. They are not protected by a passphrase and can be a security risk if your server is compromised. If you have a complex project or want more fine-grain control over permissions, consider using GitHub Apps instead.
There are no deploy keys for this repository
Check out our guide on deploy keys to learn more.


github.com/sanjeebnepal/DEVOPS-final-2/settings/keys/new
sanjeebnepal / DEVOPS-final-2
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
Code security
Deploy keys
Deploy keys / Add new
Title
ssh-key
Key
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDZaKvrGUHfY522pQNEY3N1VzXQUqNezBmQyO3nkvNYHiFHmuibJsm2yoM
[redacted]
vagrant@jenkinsvm
Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'.
Allow write access
Can this key be used to push to this repository? Deploy keys always have pull access.
Add key

```
LEQDilkYvefiom+J8ypOt87Y38rU1WPEOZsXsRMp7ivdoY5AHKzWncoFzFAlj+Koxn8uvE9Znatu6yMej+WuQXQ5p8kWEmJLDw== vagrant@jenkinsvm
vagrant@jenkinsvm:~$ ssh -T git@github.com
The authenticity of host 'github.com (140.82.112.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi sanjeebnepal/DEVOPS-final-2! You've successfully authenticated, but GitHub does not provide shell access.
vagrant@jenkinsvm:~$
```

```
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi sanjeebnepal/DEVOPS-final-2! You've successfully authenticated, but GitHub does not provide shel
vagrant@jenkinsvm:~$ git clone git@github.com:sanjeebnepal/DEVOPS-final-2.git
Cloning into 'DEVOPS-final-2'...
remote: Enumerating objects: 276, done.
remote: Counting objects: 100% (276/276), done.
remote: Compressing objects: 100% (151/151), done.
remote: Total 276 (delta 111), reused 272 (delta 107), pack-reused 0 (from 0)
Receiving objects: 100% (276/276), 181.93 KiB | 1.31 MiB/s, done.
Resolving deltas: 100% (111/111), done.
vagrant@jenkinsvm:~$
```

Webhook

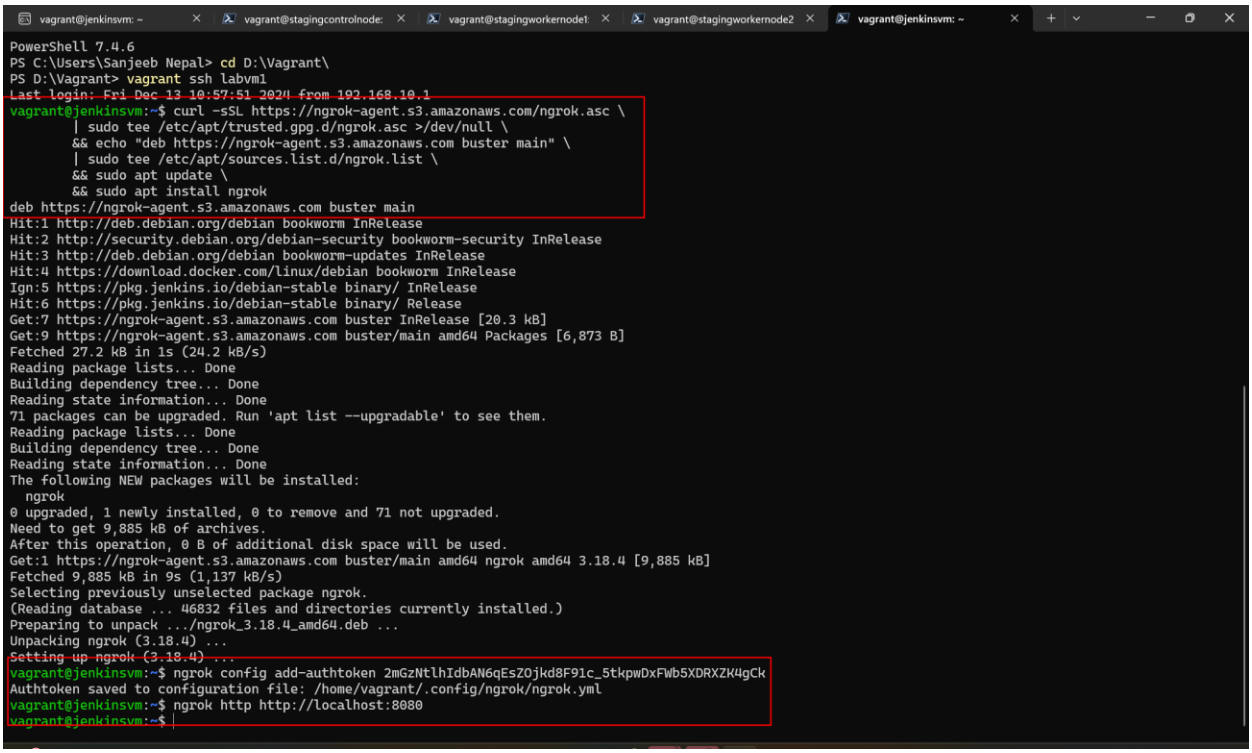
We can also set up a webhook beforehand to trigger the pipelines automatically when we push to GitHub.

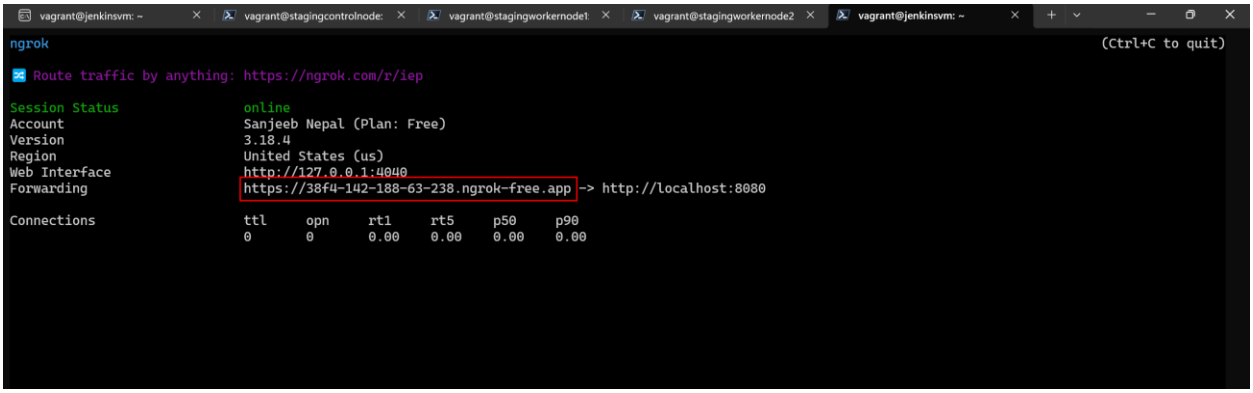
Since we are using Vagrant, we can set up ngrok in Jenkins for this purpose.

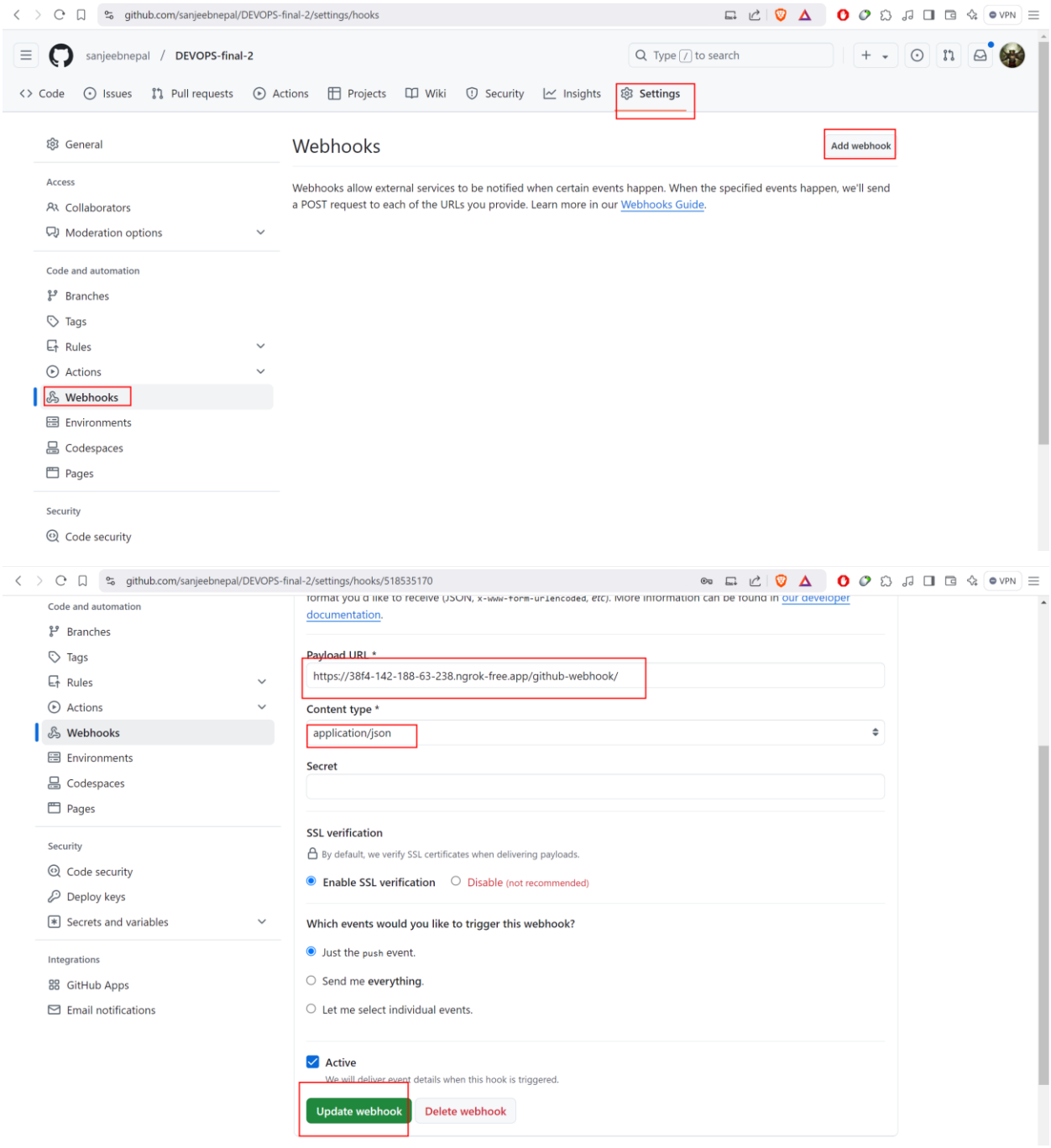
Go to https://ngrok.com/

Sign up, select the Linux option, install ngrok in Jenkins VM, and start it.

The public DNS is then copied to the webhook section of GitHub.


vagrant@jenkinsvm: ~
vagrant@stagingcontrolnode:
vagrant@stagingworkernode1:
vagrant@stagingworkernode2
vagrant@jenkinsvm: ~
PowerShell 7.4.6
PS C:\Users\Sanjeeb Nepal> cd D:\Vagrant\
PS D:\Vagrant> vagrant ssh labvm1
Last login: Fri Dec 13 10:57:51 2024 from 192.168.10.1
vagrant@jenkinsvm:~$ curl -sSL https://ngrok-agent.s3.amazonaws.com/ngrok.asc \
| sudo tee /etc/apt/trusted.gpg.d/ngrok.asc >/dev/null \
&& echo "deb https://ngrok-agent.s3.amazonaws.com buster main" \
| sudo tee /etc/apt/sources.list.d/ngrok.list \
&& sudo apt update \
&& sudo apt install ngrok
deb https://ngrok-agent.s3.amazonaws.com buster main
Hit:1 http://deb.debian.org/debian bookworm InRelease
Hit:2 http://security.debian.org/debian-security bookworm-security InRelease
Hit:3 http://deb.debian.org/debian bookworm-updates InRelease
Hit:4 https://download.docker.com/linux/debian bookworm InRelease
Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:6 https://pkg.jenkins.io/debian-stable binary/ Release
Get:7 https://ngrok-agent.s3.amazonaws.com buster InRelease [20.3 kB]
Get:9 https://ngrok-agent.s3.amazonaws.com buster/main amd64 Packages [6,873 B]
Fetched 27.2 kB in 1s (24.2 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
71 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
ngrok
0 upgraded, 1 newly installed, 0 to remove and 71 not upgraded.
Need to get 9,885 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 https://ngrok-agent.s3.amazonaws.com buster/main amd64 ngrok amd64 3.18.4 [9,885 kB]
Fetched 9,885 kB in 9s (1,137 kB/s)
Selecting previously unselected package ngrok.
(Reading database ... 46832 files and directories currently installed.)
Preparing to unpack .../ngrok_3.18.4_amd64.deb ...
Unpacking ngrok (3.18.4) ...
Setting up ngrok (3.18.4) ...
vagrant@jenkinsvm:~$ ngrok config add-authtoken 2mGzNtlhIdbAN6qEsZOjkd8F91c_5tkpwDxFWb5XDRXZK4gCk
Authtoken saved to configuration file: /home/vagrant/.config/ngrok/ngrok.yml
vagrant@jenkinsvm:~$ ngrok http http://localhost:8080
vagrant@jenkinsvm:~$


vagrant@jenkinsvm: ~
vagrant@stagingcontrolnode:
vagrant@stagingworkernode1:
vagrant@stagingworkernode2
vagrant@jenkinsvm: ~
ngrok
(Ctrl+C to quit)
Route traffic by anything: https://ngrok.com/r/iep
Session Status online
Account Sanjeeb Nepal (Plan: Free)
Version 3.18.4
Region United States (us)
Web Interface http://127.0.0.1:4040
Forwarding https://38f4-142-188-63-238.ngrok-free.app -> http://localhost:8080
Connections ttl opn rt1 rt5 p50 p90
0 0 0.00 0.00 0.00 0.00



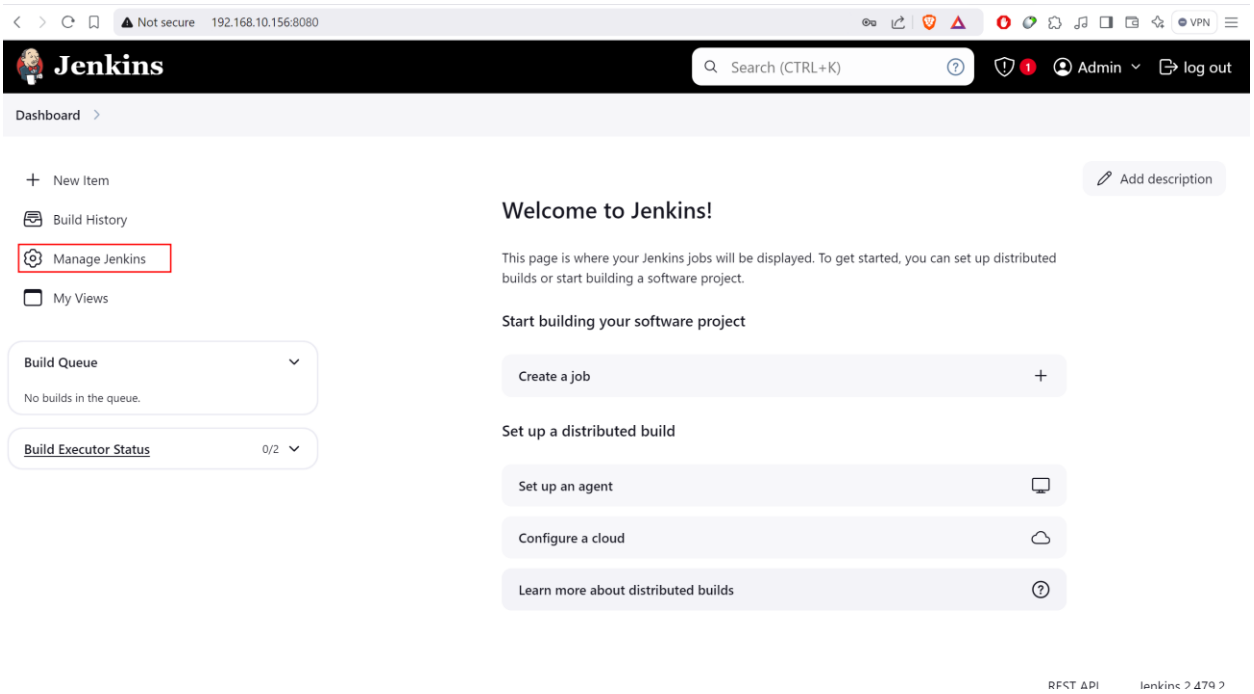
Test Pipeline (Development Environment)

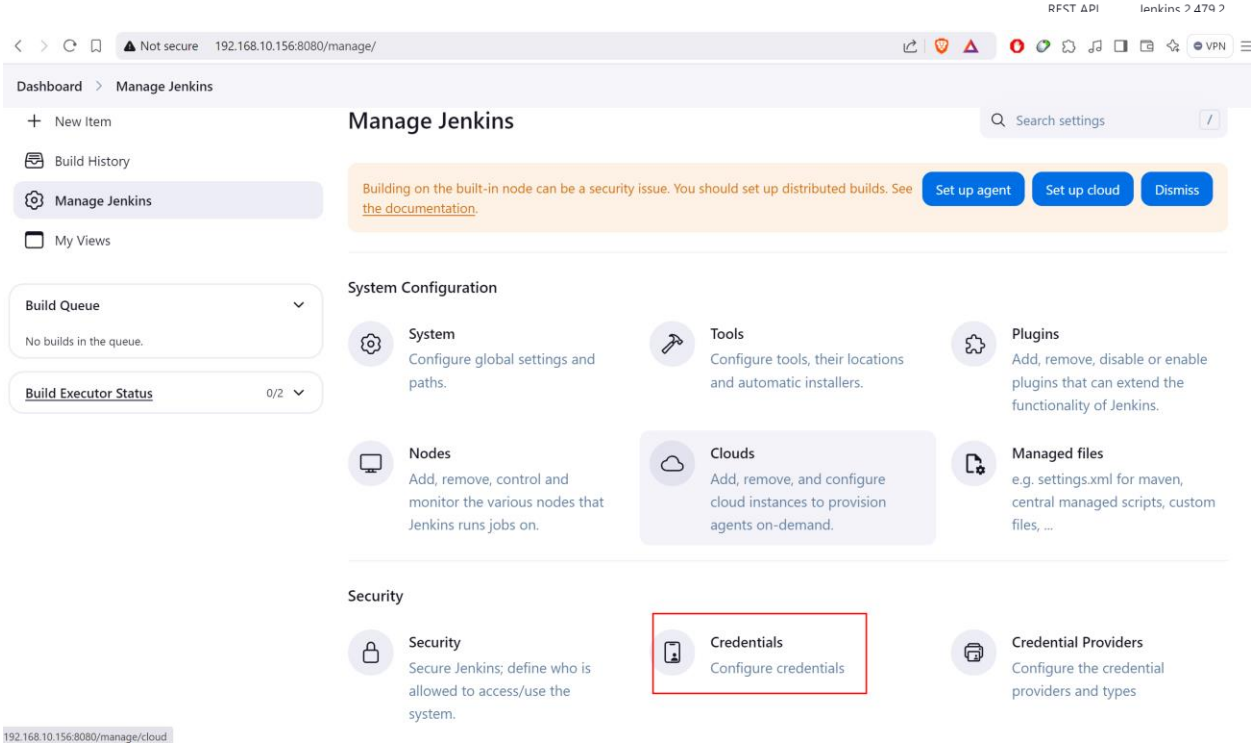
The test pipeline is built to test the code itself. We have included the Cypress tool, docker, and curl for that purpose.

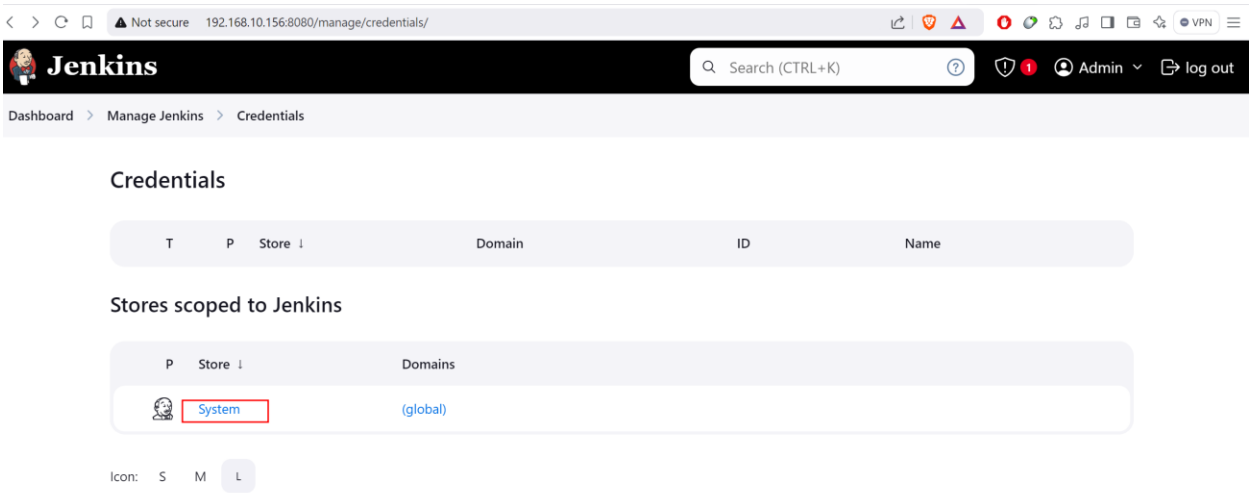
Cypress will do end-to-end testing of the web app to ensure that the components of the application exist. Then, we use Docker to build a docker image, which we will test locally by

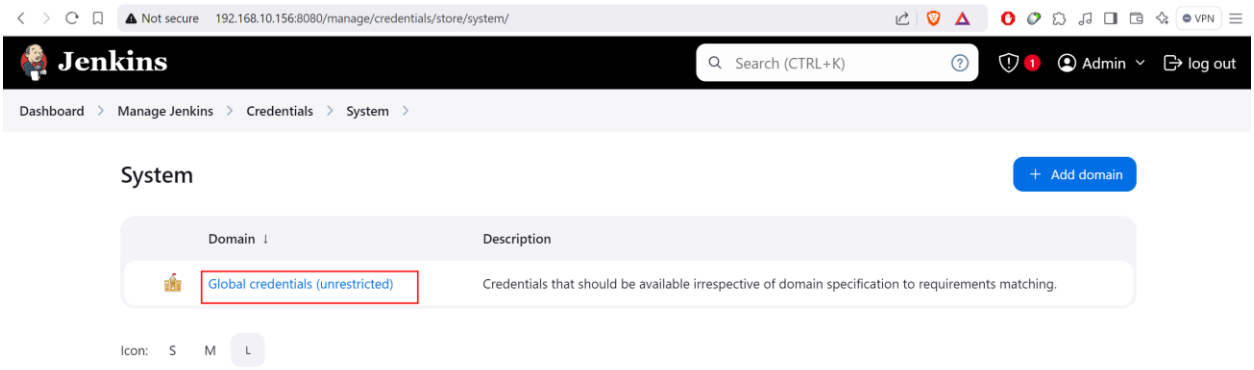
running a container from that image and testing the port accessibility using the curl command in Jenkinsfile. Finally, the successful completion of the pipeline then pushes the image to the docker hub and triggers the staging pipeline.

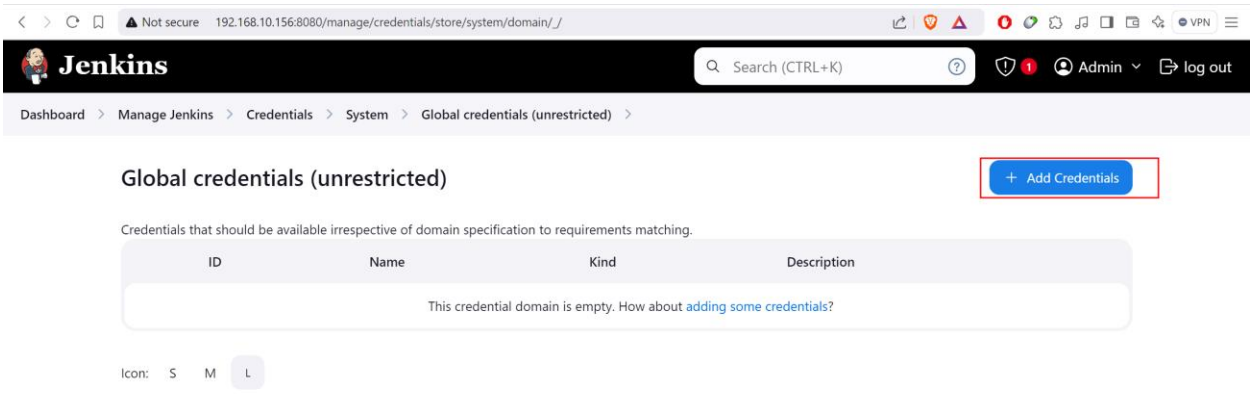
One thing to consider is we are going to push the images to the docker hub, so the credentials to log in must be saved in the credentials manager of Jenkins.

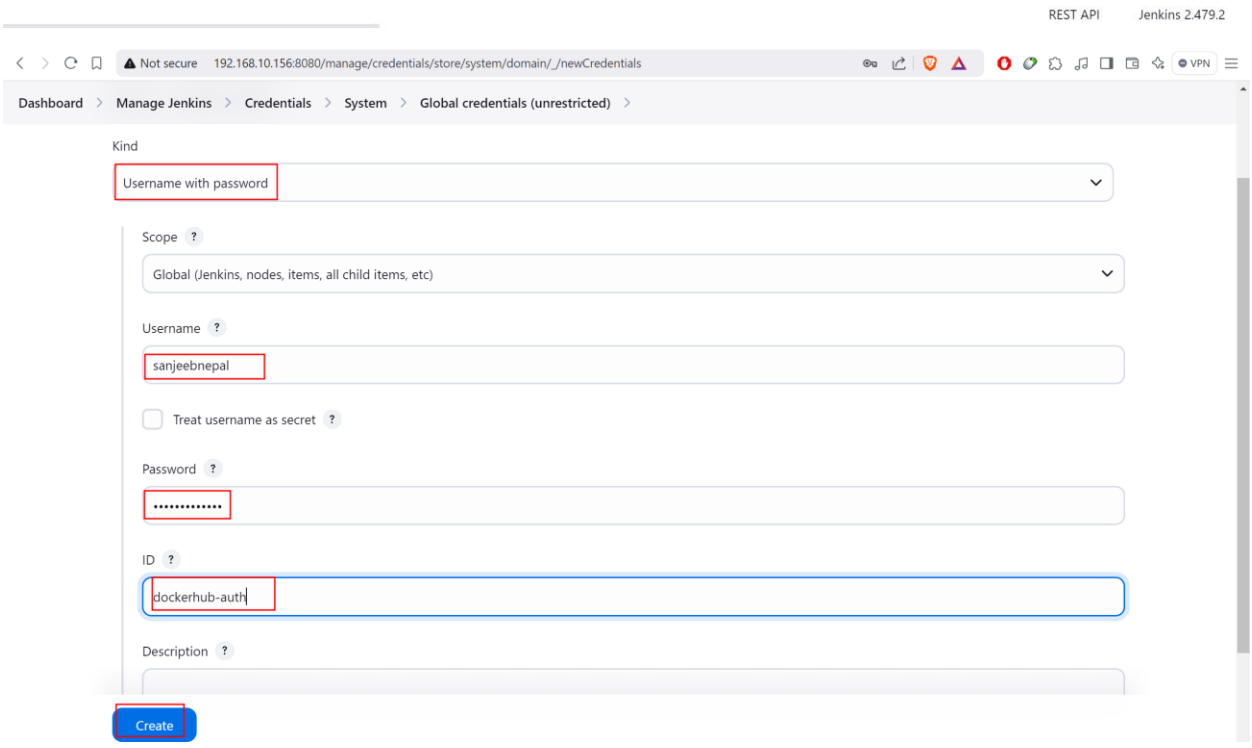

Not secure 192.168.10.156:8080
Jenkins
Search (CTRL+K)
Admin
log out
Dashboard
New Item
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status 0/2
Add description
Welcome to Jenkins!
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.
Start building your software project
Create a job
Set up a distributed build
Set up an agent
Configure a cloud
Learn more about distributed builds


Not secure 192.168.10.156:8080/manage/
Dashboard > Manage Jenkins
New Item
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status 0/2
Manage Jenkins
Search settings
Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.
Set up agent
Set up cloud
Dismiss
System Configuration
System
Configure global settings and paths.
Tools
Configure tools, their locations and automatic installers.
Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
Clouds
Add, remove, and configure cloud instances to provision agents on-demand.
Managed files
e.g. settings.xml for maven, central managed scripts, custom files, ...
Security
Security
Secure Jenkins; define who is allowed to access/use the system.
Credentials
Configure credentials
Credential Providers
Configure the credential providers and types
192.168.10.156:8080/manage/cloud

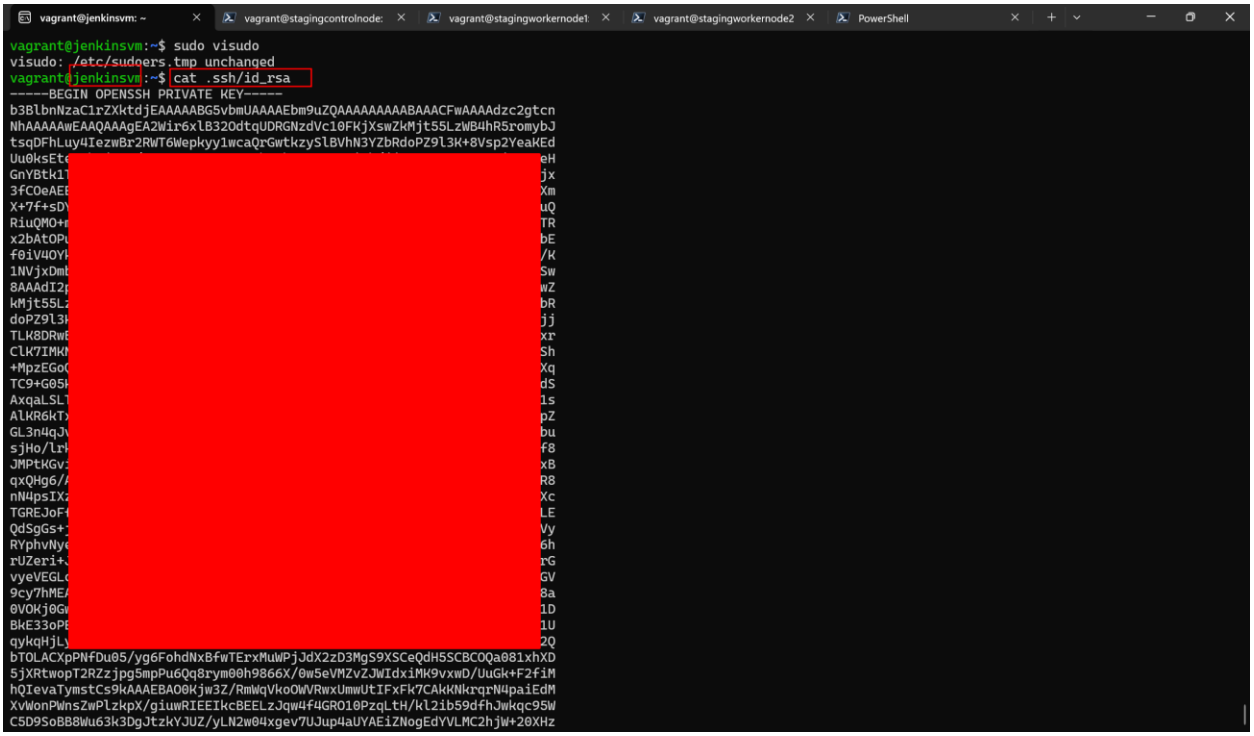

Not secure 192.168.10.156:8080/manage/credentials/
Jenkins
Search (CTRL+K)
Admin
log out
Dashboard > Manage Jenkins > Credentials
Credentials
T P Store ↓ Domain ID Name
Stores scoped to Jenkins
P Store ↓ Domains
System
(global)
Icon: S M L

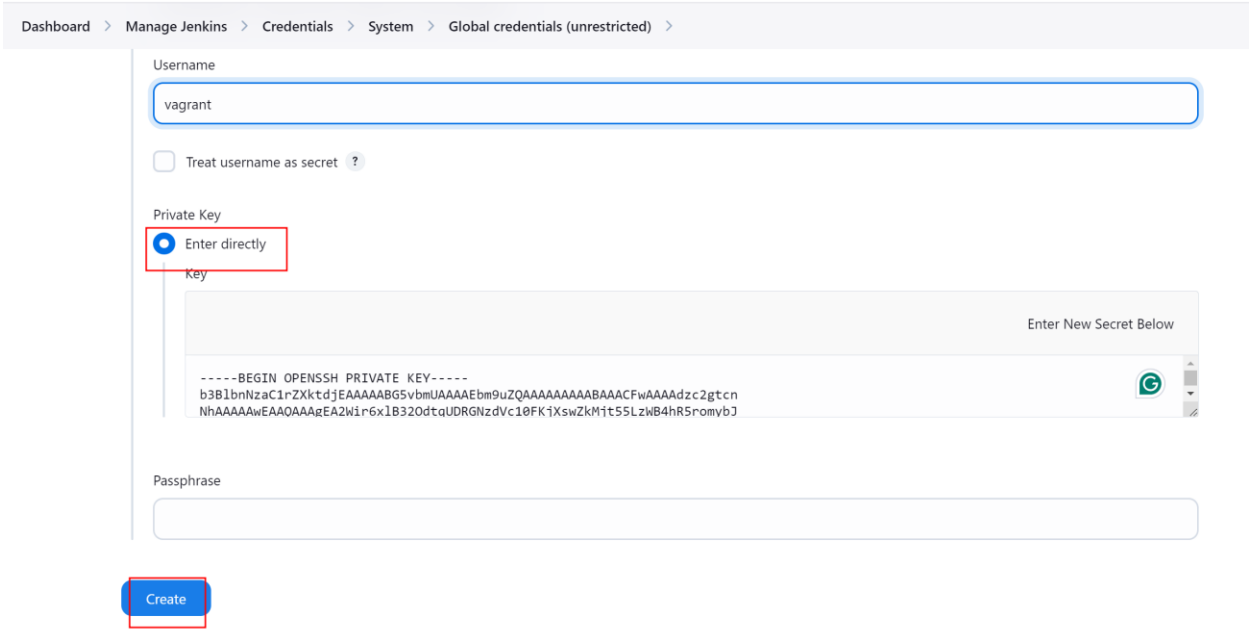

Not secure 192.168.10.156:8080/manage/credentials/store/system/
Jenkins
Search (CTRL+K)
Admin
log out
Dashboard > Manage Jenkins > Credentials > System >
System
+ Add domain
Domain ↓ Description
Global credentials (unrestricted)
Credentials that should be available irrespective of domain specification to requirements matching.
Icon: S M L

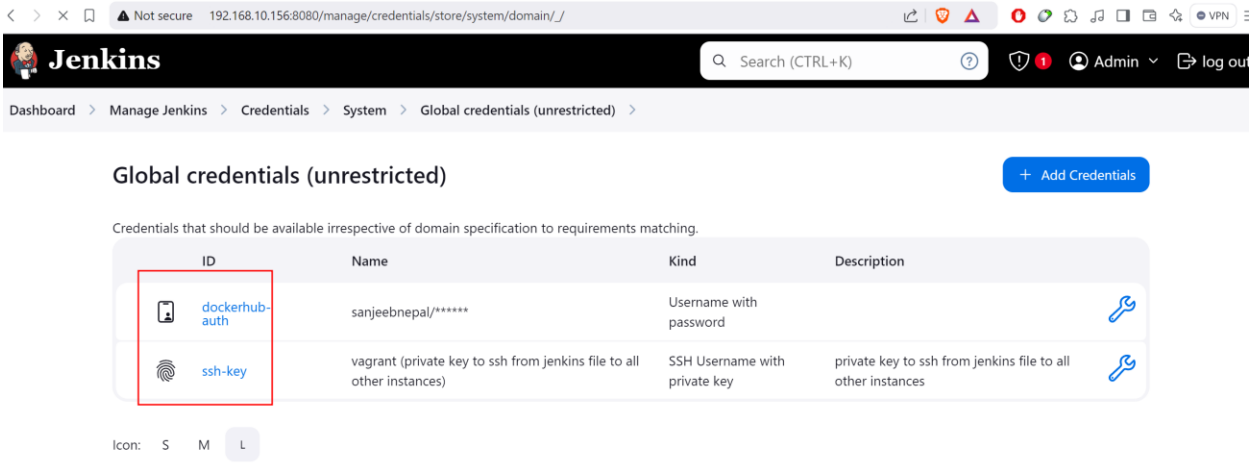




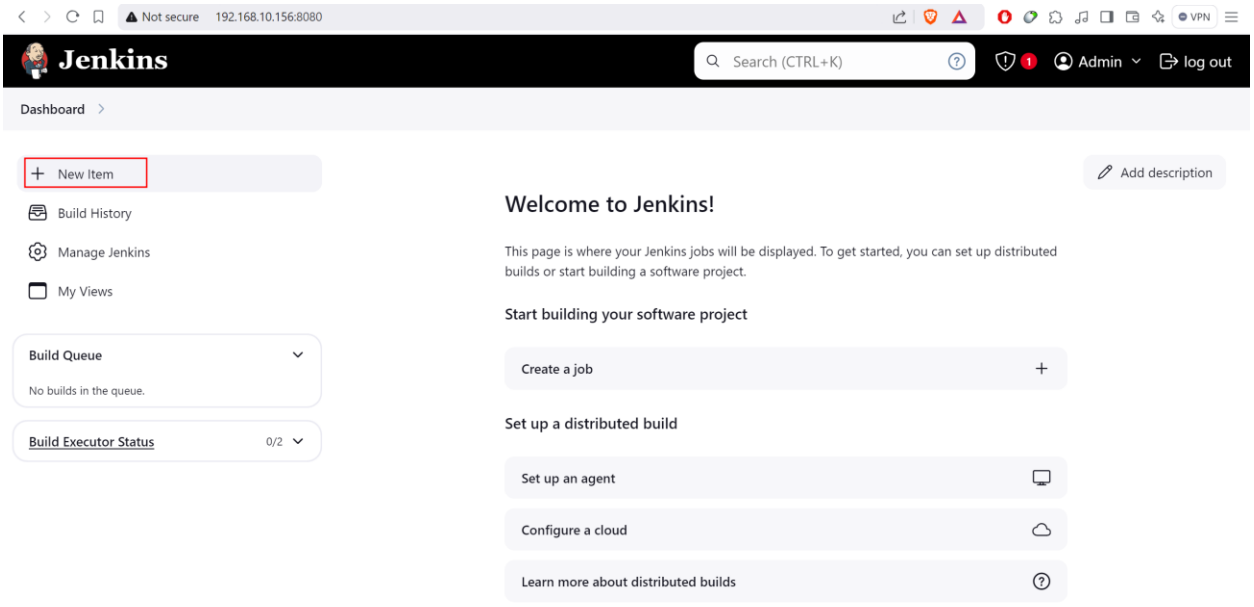
We can also add the private SSH key that we need to login to the other VMs of the staging environment so that we can automatically trigger that pipeline after the test pipeline completes. For that, we go to the Jenkins VM and navigate to the .ssh directory from home, then copy the id_rsa key.

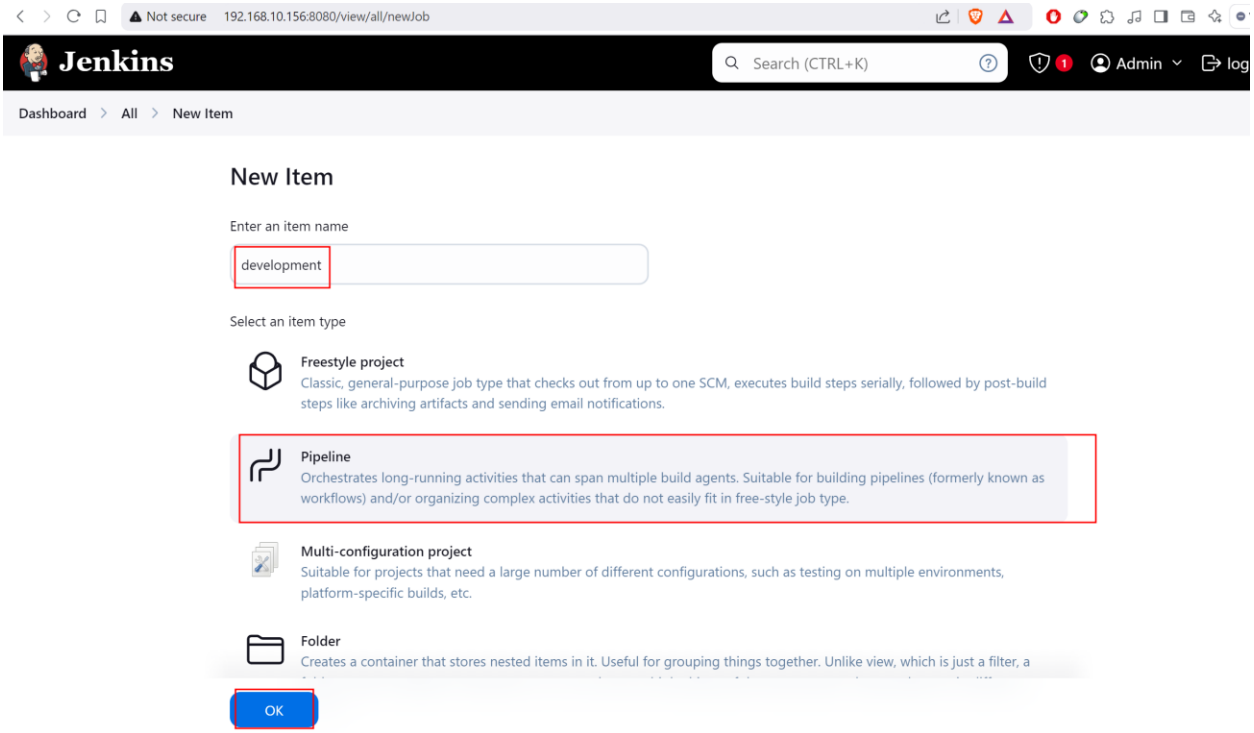

vagrant@jenkinsvm: ~
vagrant@stagingcontrolnode:
vagrant@stagingworkernode1
vagrant@stagingworkernode2
PowerShell
vagrant@jenkinsvm:~$ sudo visudo
visudo: /etc/sudoers.tmp unchanged
vagrant@jenkinsvm:~$ cat .ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAACFwAAAAdzc2gtcn
NhAAAAAwEAAQAAAgEA2Wir6xlB32OdtqUDRGNzdVc10FkjXswZkMjt55LzWB4hR5romybJ
tsqDFhLuy4IezwBr2RWT6Wepkyy1wcaQrGwtkzySlBVhN3YZbRdoPZ9l3K+8Vsp2YeaKEd
Uu0ksEt
GnYBtk1
3fCOeAE
X+7f+sD
RiuQMO+
x2bAtOPu
f0iV4OY
1NVjxDmb
8AAAdI2
kMjt55Lz
doPZ9l3
TLK8DRwE
ClK7IMKN
+MpzEGoC
TC9+G05k
AxqaLSLT
AlKR6kTx
GL3n4qJV
sjHo/lrk
JMPtKGvz
qxQHg6/A
nN4psIXz
TGREJoFf
QdSgGs+j
RYphvNyc
rUZeri+x
vyeVEGLc
9cy7hMEA
0VOKj0Gw
BkE33oPE
qykqHjLy
bTOLACXpPNfDu05/yg6FohdNxBfwTErxMuWPjJdX2zD3MgS9XSCeQdH5SCBCOQa081xhXD
5jXRtwopT2RZzjpg5mpPu6Qq8rym00h9866X/0w5eVMZvZJWIdxiMk9vxwD/UuGk+F2fiM
hQIevaTymstCs9kAAAEBAO0Kjw3Z/RmWqVkoOWVRwxUmwUtIFxFk7CAkkNkrqrN4paiEdM
XvWonPWnsZwPlzkpX/giuwRIEEIkcBEELzJqw4f4GRO10PzqLtH/kl2ib59dfhJwkqc95W
C5D9SoBB8Wu63k3DgJtzkYJUZ/yLN2w04xgev7UJup4aUYAEiZNogEdYVLMC2hjW+20XHz
Not secure
192.168.10.156:8080/manage/credentials/store/system/domain/_/newCredentials
VPN
Dashboard
Manage Jenkins
Credentials
System
Global credentials (unrestricted)
New credentials
Kind
SSH Username with private key
Scope
Global (Jenkins, nodes, items, all child items, etc)
ID
ssh-key
Description
private key to ssh from jenkins file to all other instances
Username
vagrant
Treat username as secret
Create

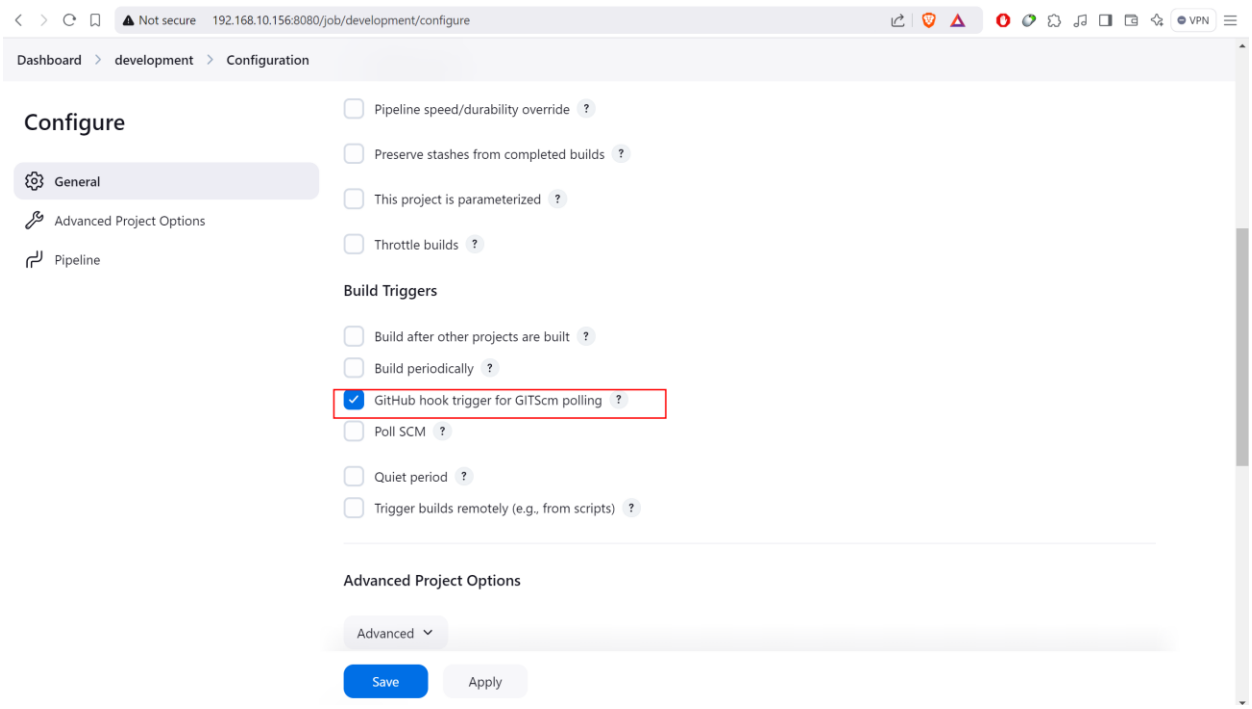


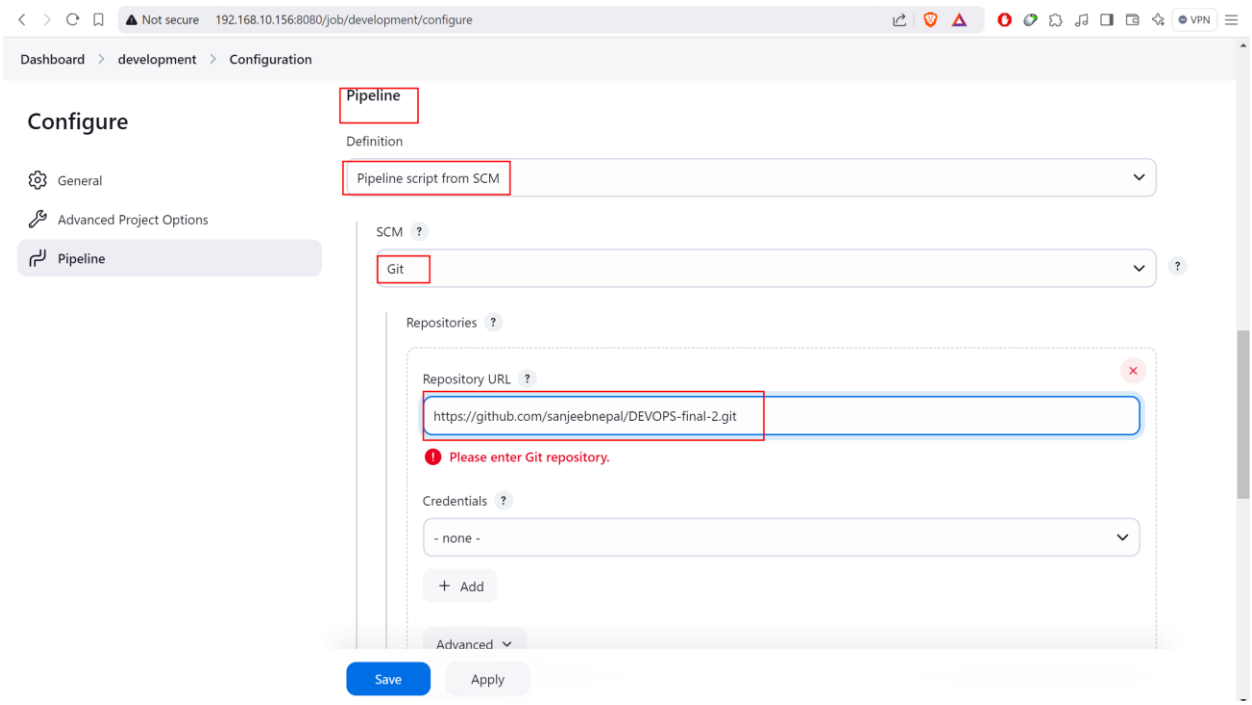


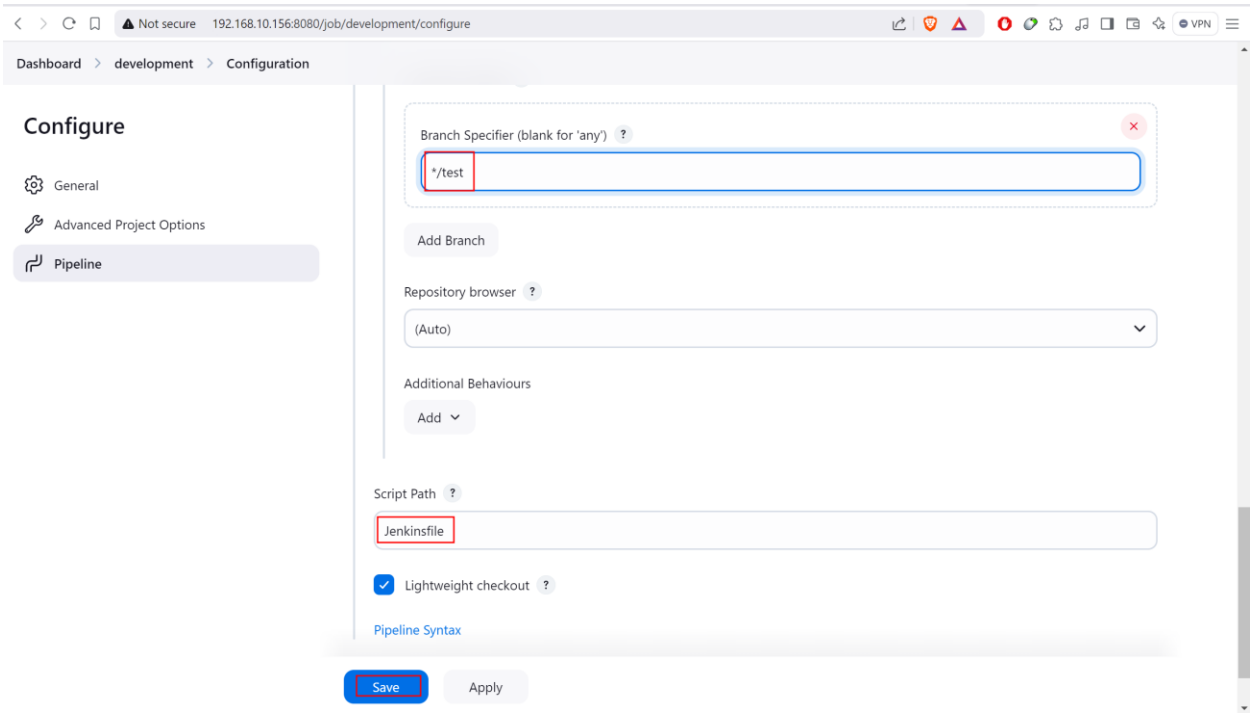
Next, we create the pipeline itself. In this process, we used the pipeline option and connected the GitHub repository to the pipeline. We also mentioned the specific branch we want to connect to and the name of the Jenkinsfile the pipeline pipeline is supposed to run.


Not secure 192.168.10.156:8080
Jenkins
Search (CTRL+K)
Admin
log out
Dashboard
New Item
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status 0/2
Add description
Welcome to Jenkins!
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.
Start building your software project
Create a job
Set up a distributed build
Set up an agent
Configure a cloud
Learn more about distributed builds

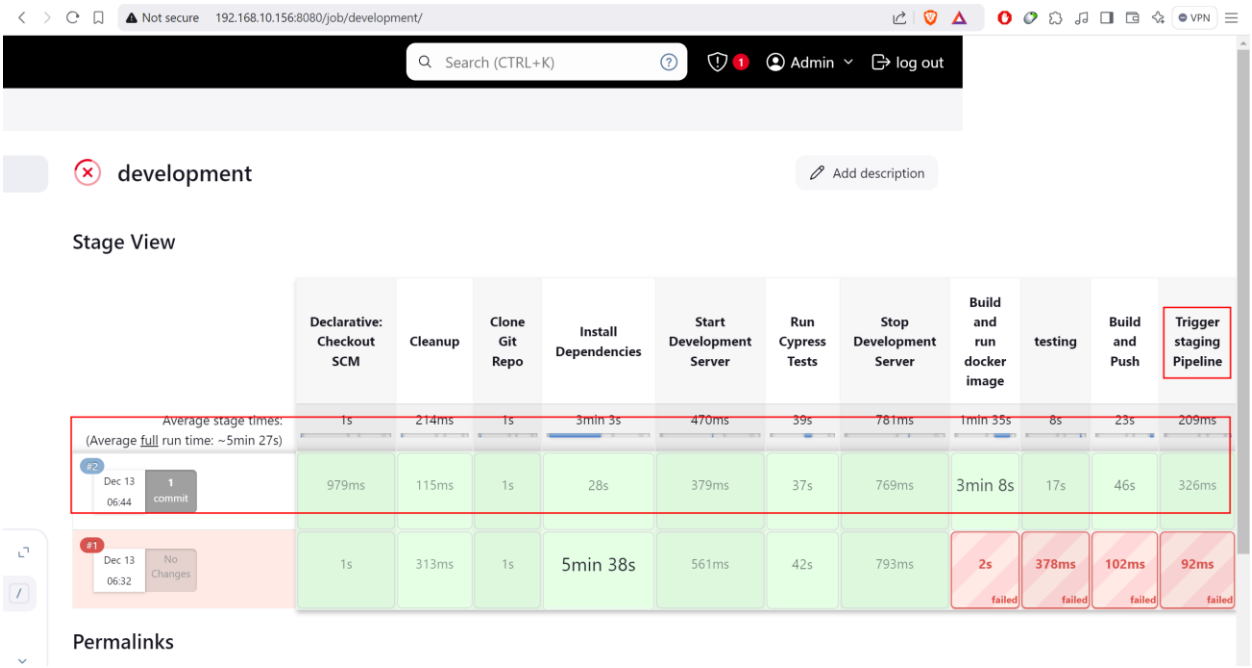

Not secure 192.168.10.156:8080/view/all/newJob
Jenkins
Search (CTRL+K)
Admin
Dashboard > All > New Item
New Item
Enter an item name
development
Select an item type
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build 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
OK


Not secure 192.168.10.156:8080/job/development/configure
VPN
Dashboard > development > Configuration
Configure
General
Advanced Project Options
Pipeline
Pipeline speed/durability override
Preserve stashes from completed builds
This project is parameterized
Throttle builds
Build Triggers
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Quiet period
Trigger builds remotely (e.g., from scripts)
Advanced Project Options
Advanced
Save
Apply

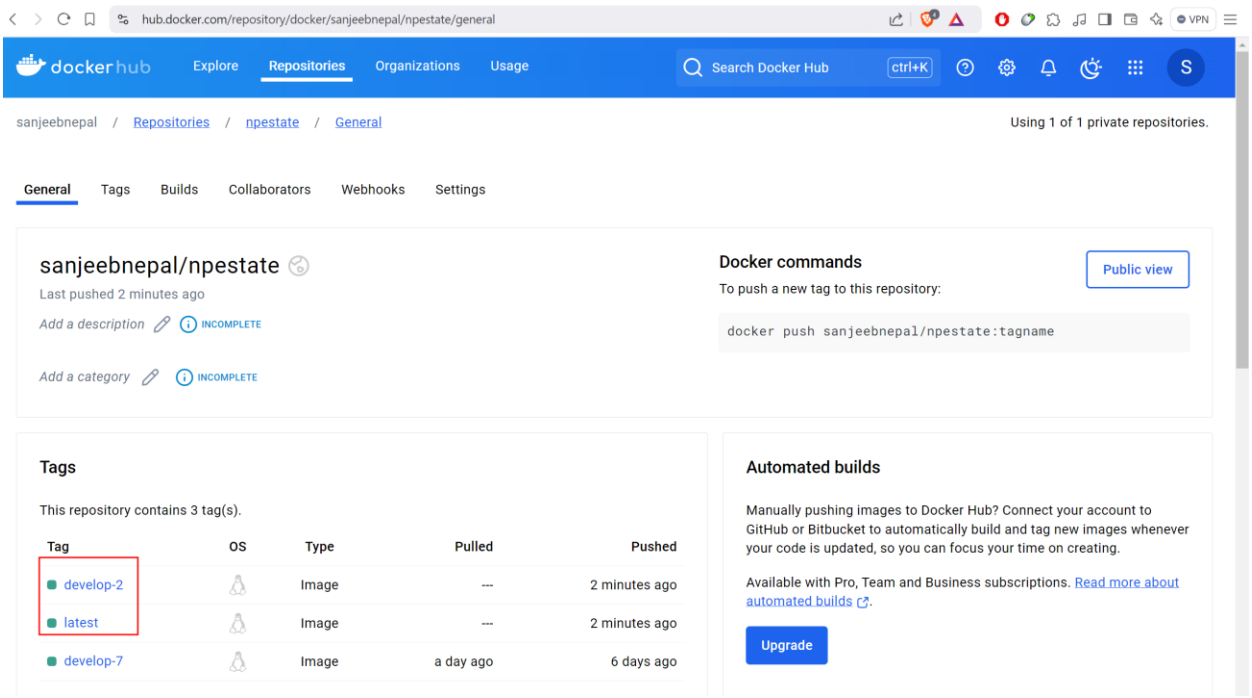

Not secure 192.168.10.156:8080/job/development/configure
VPN
Dashboard > development > Configuration
Configure
General
Advanced Project Options
Pipeline
Pipeline
Definition
Pipeline script from SCM
SCM
Git
Repositories
Repository URL
https://github.com/sanjeebnepal/DEVOPS-final-2.git
Please enter Git repository.
Credentials
- none -
Add
Advanced
Save
Apply



After we push some changes to GitHub in the test branch, the pipeline is automatically triggered as below.

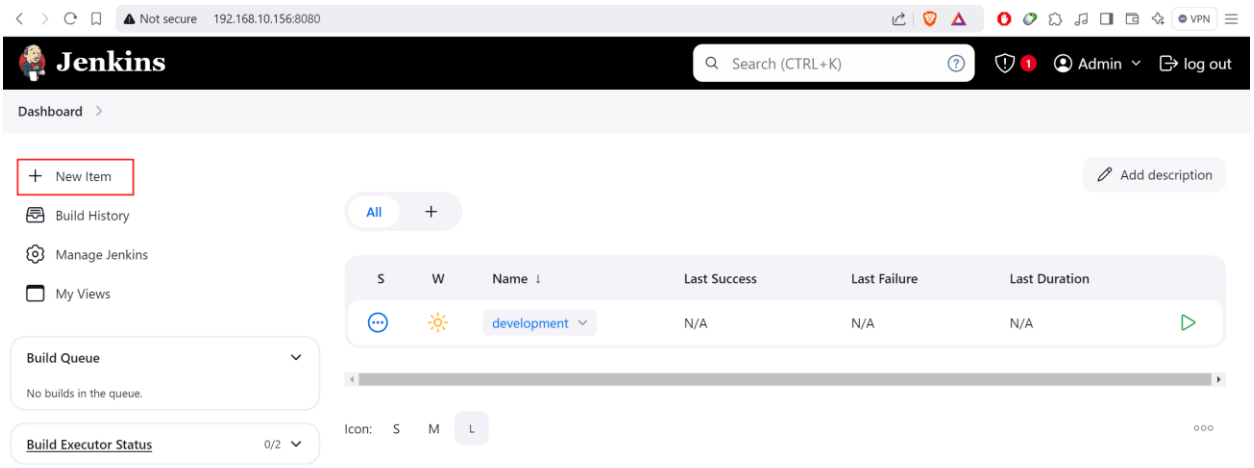


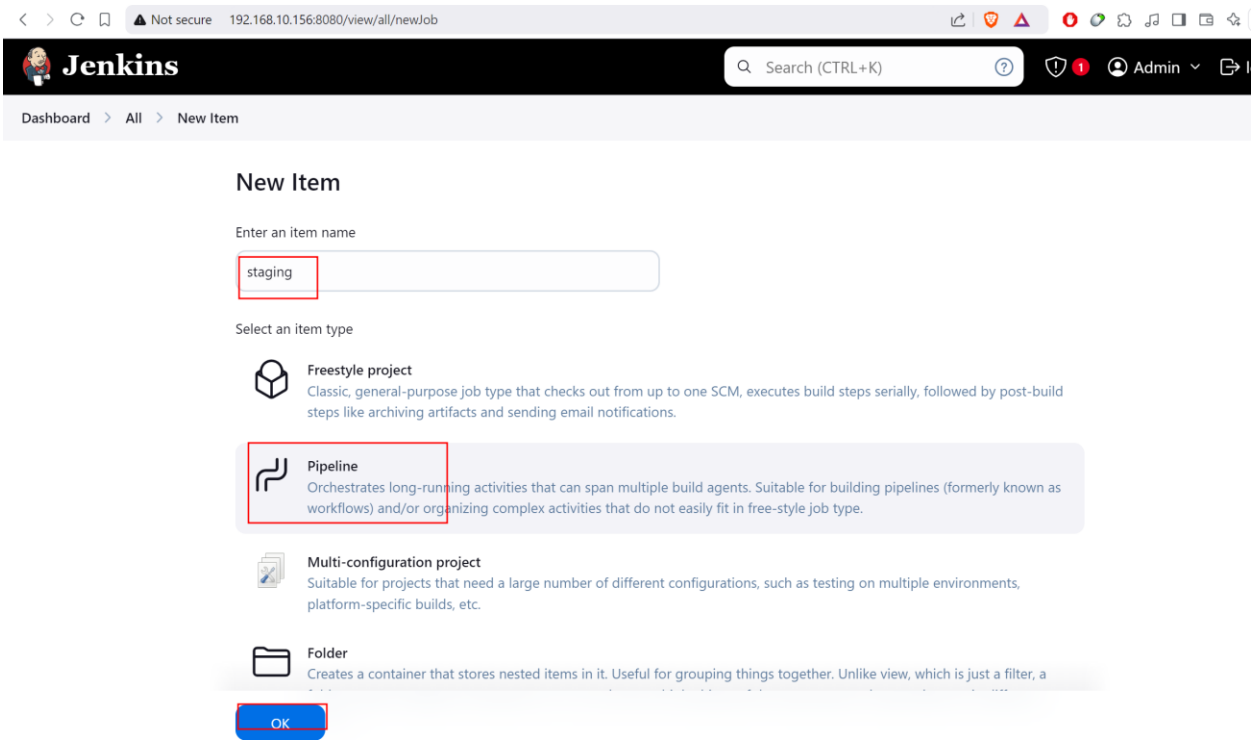
We can verify the images that are pushed to docker hub.

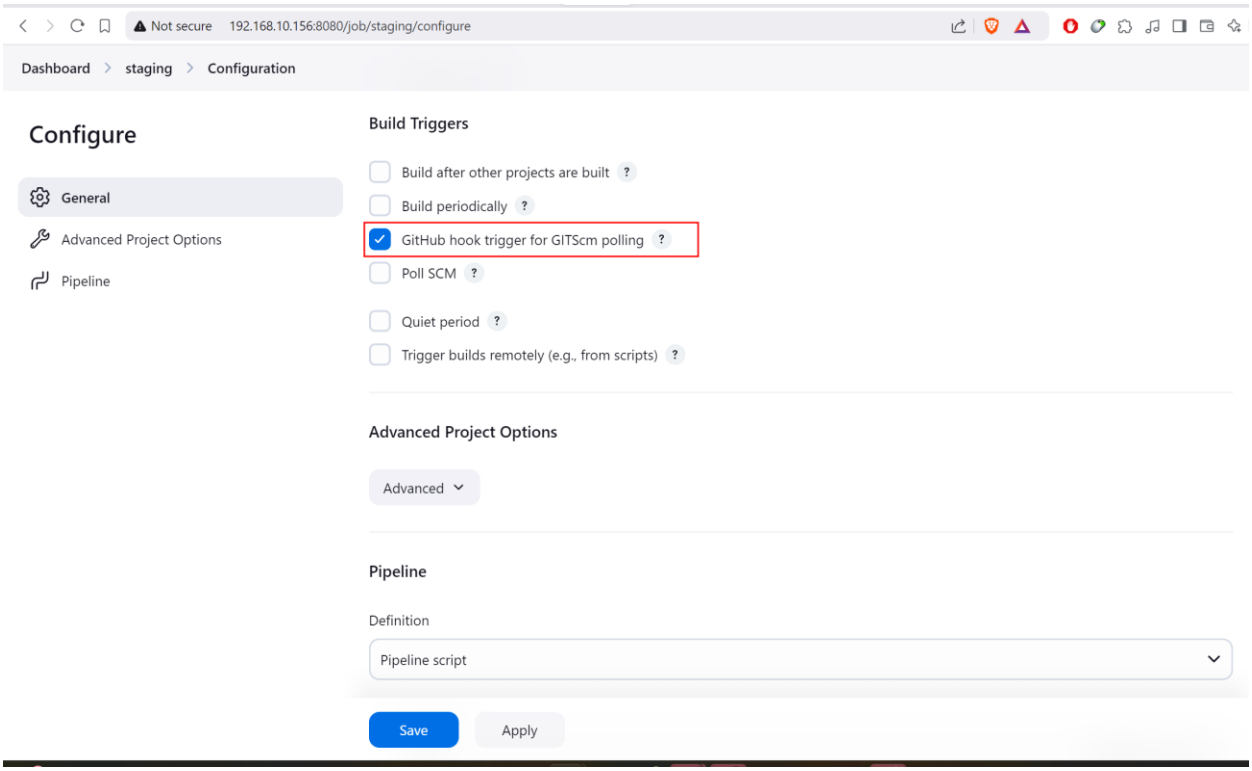


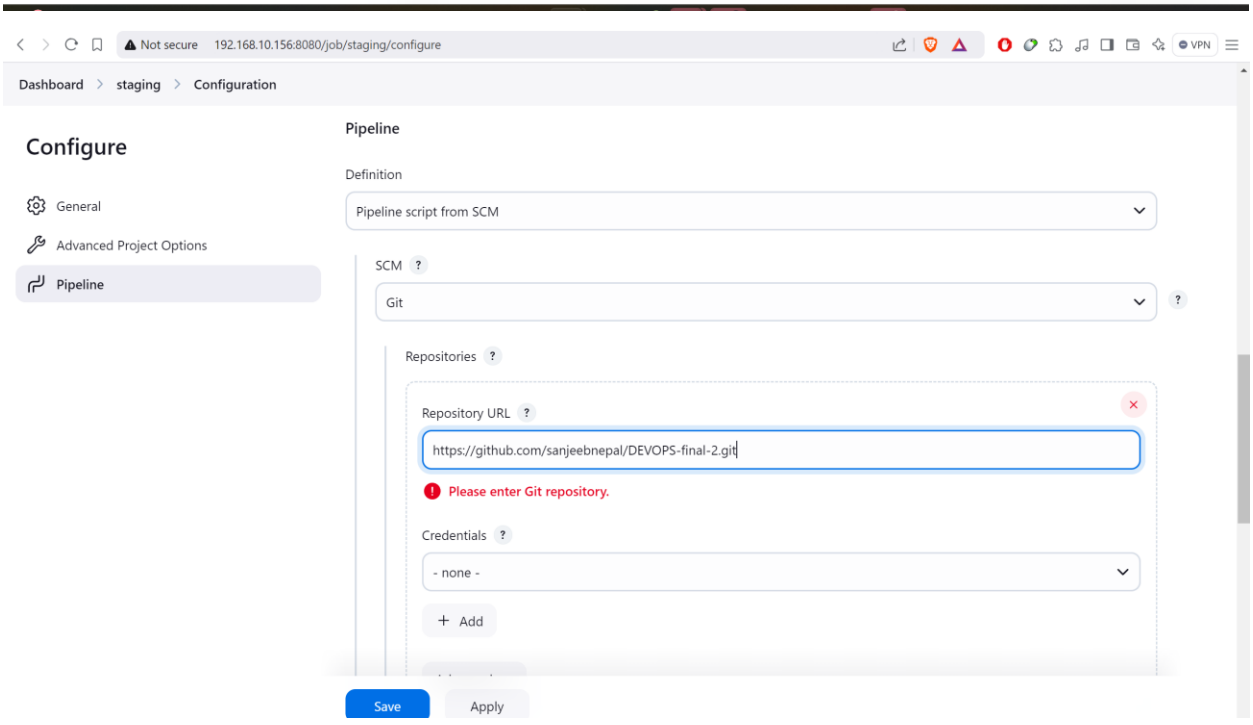
Staging Pipeline (Staging Environment)

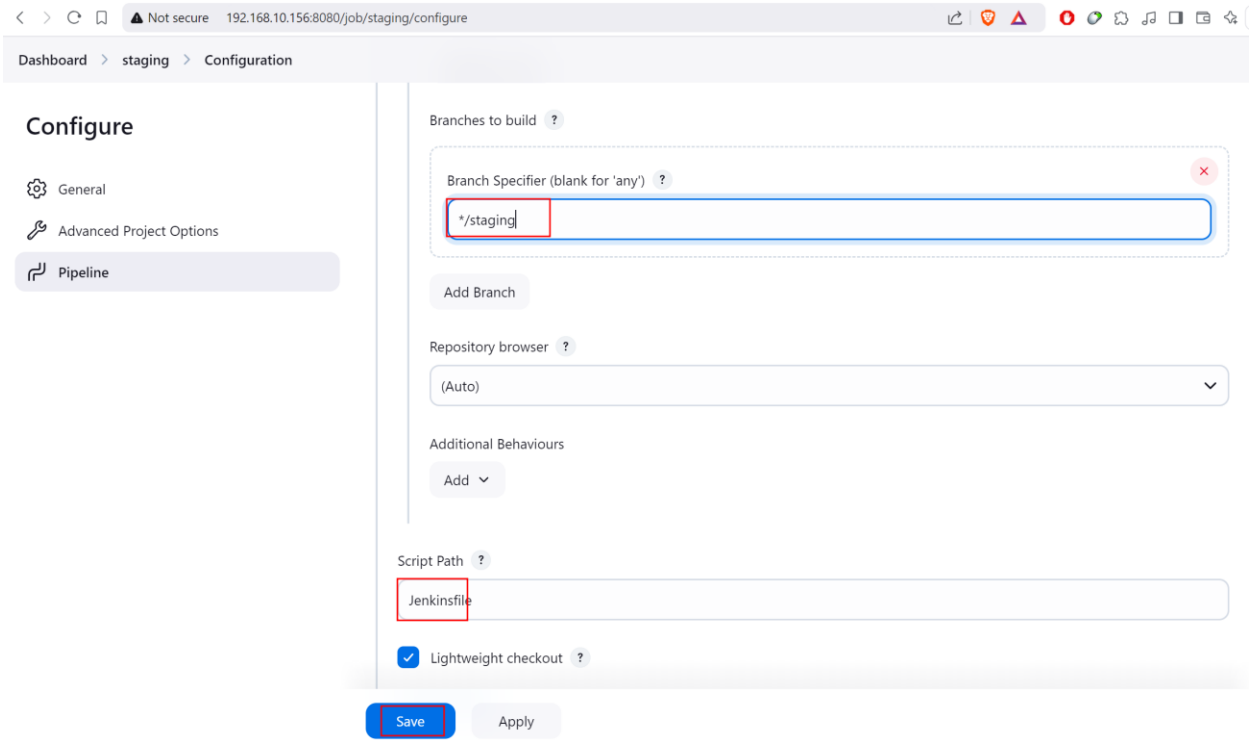
Staging pipeline is triggered automatically after the completion of test pipeline. In here, we have set up three virtual machines for deployment purposes. The orchestration is done using Kubernetes. In the control node, the K3S is setup and a token is generated. That token is used to set up K3S in the other two worker nodes. Here, nodes mean Virtual Machines. That is possible from Jenkins VM which uses private key stored in credentials manager and establishes SSH connection to the three nodes. We use the private key to store in Jenkins and public key in each of the nodes.

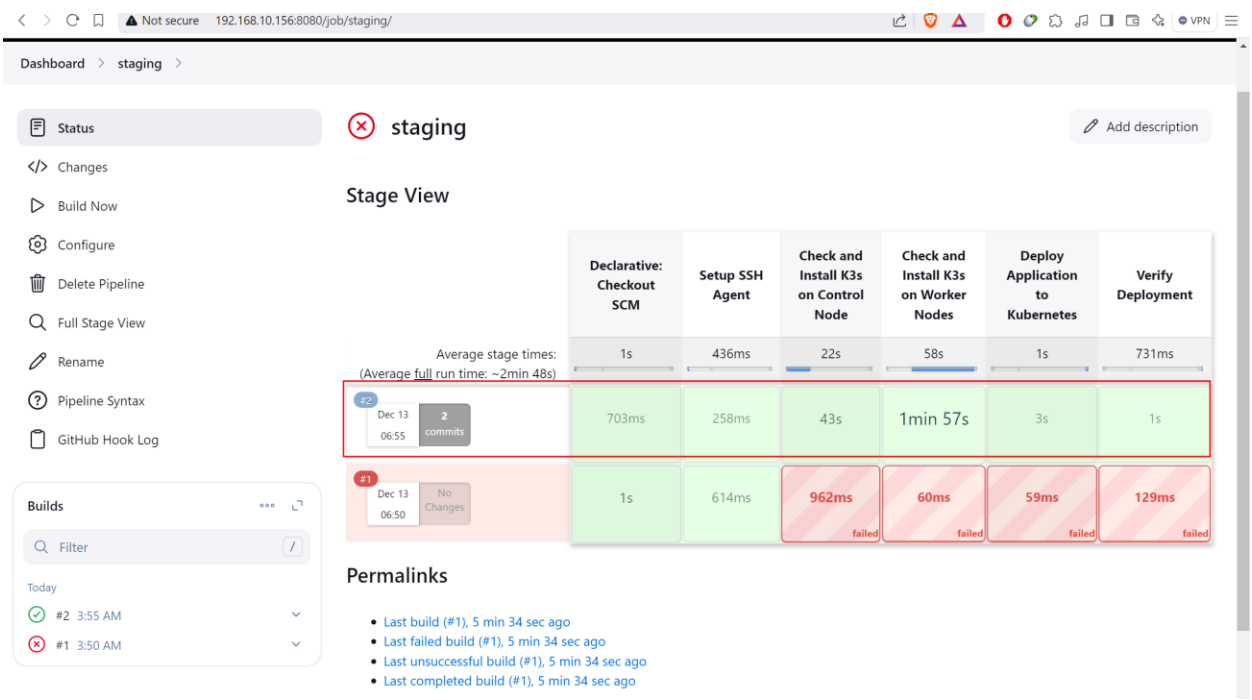

Not secure 192.168.10.156:8080
Jenkins
Search (CTRL+K)
Admin
log out
Dashboard
New Item
Build History
Manage Jenkins
My Views
Add description
All
S
W
Name
Last Success
Last Failure
Last Duration
development
N/A
N/A
N/A
Build Queue
No builds in the queue.
Build Executor Status
0/2
Icon: S M L

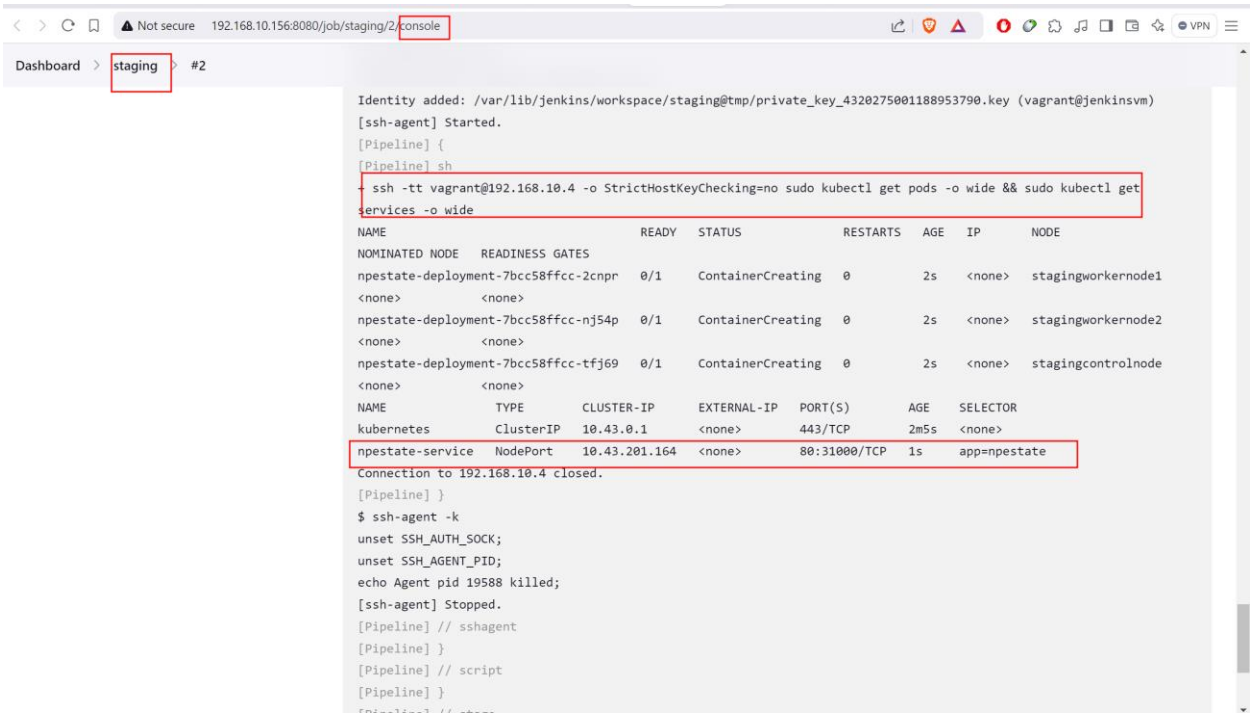

Not secure 192.168.10.156:8080/view/all/newJob
Jenkins
Search (CTRL+K)
Admin
Dashboard
All
New Item
New Item
Enter an item name
staging
Select an item type
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps 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
OK


Not secure 192.168.10.156:8080/job/staging/configure
Dashboard > staging > Configuration
Configure
General
Advanced Project Options
Pipeline
Build Triggers
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Quiet period
Trigger builds remotely (e.g., from scripts)
Advanced Project Options
Advanced
Pipeline
Definition
Pipeline script
Save
Apply

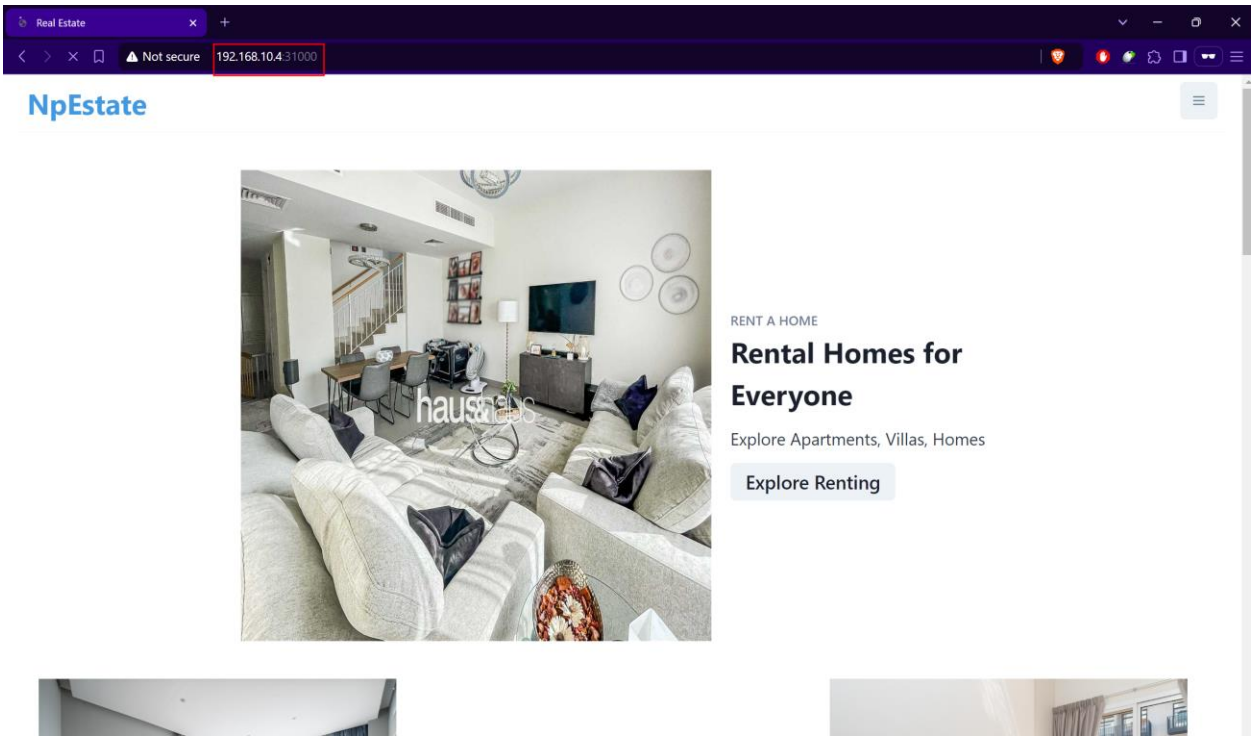

Not secure 192.168.10.156:8080/job/staging/configure
VPN
Dashboard > staging > Configuration
Configure
General
Advanced Project Options
Pipeline
Pipeline
Definition
Pipeline script from SCM
SCM
Git
Repositories
Repository URL
https://github.com/sanjeebnepal/DEVOPS-final-2.git
Please enter Git repository.
Credentials
- none -
Add
Save
Apply


Not secure 192.168.10.156:8080/job/staging/configure
Dashboard > staging > Configuration
Configure
General
Advanced Project Options
Pipeline
Branches to build
Branch Specifier (blank for 'any')
*/staging
Add Branch
Repository browser
(Auto)
Additional Behaviours
Add
Script Path
Jenkinsfile
Lightweight checkout
Save
Apply


Not secure 192.168.10.156:8080/job/staging/
Dashboard > staging >
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
GitHub Hook Log
Builds
Filter
Today
#2 3:55 AM
#1 3:50 AM
staging
Add description
Stage View
Declarative: Checkout SCM
Setup SSH Agent
Check and Install K3s on Control Node
Check and Install K3s on Worker Nodes
Deploy Application to Kubernetes
Verify Deployment
Average stage times:
(Average full run time: ~2min 48s)
1s
436ms
22s
58s
1s
731ms
#2
Dec 13
06:55
2 commits
703ms
258ms
43s
1min 57s
3s
1s
#1
Dec 13
06:50
No Changes
1s
614ms
962ms
failed
60ms
failed
59ms
failed
129ms
failed
Permalinks
Last build (#1), 5 min 34 sec ago
Last failed build (#1), 5 min 34 sec ago
Last unsuccessful build (#1), 5 min 34 sec ago
Last completed build (#1), 5 min 34 sec ago



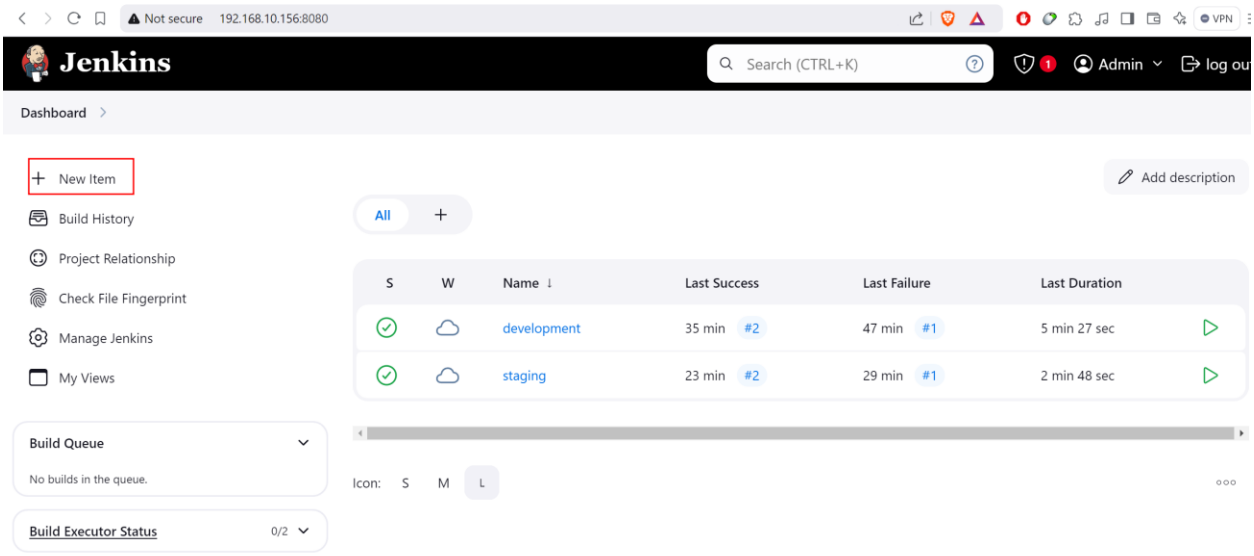
After the successful completion of the pipeline, we can take IP of any control or worker node and access the web app using the port 31000 mentioned in our service.yaml configuration.

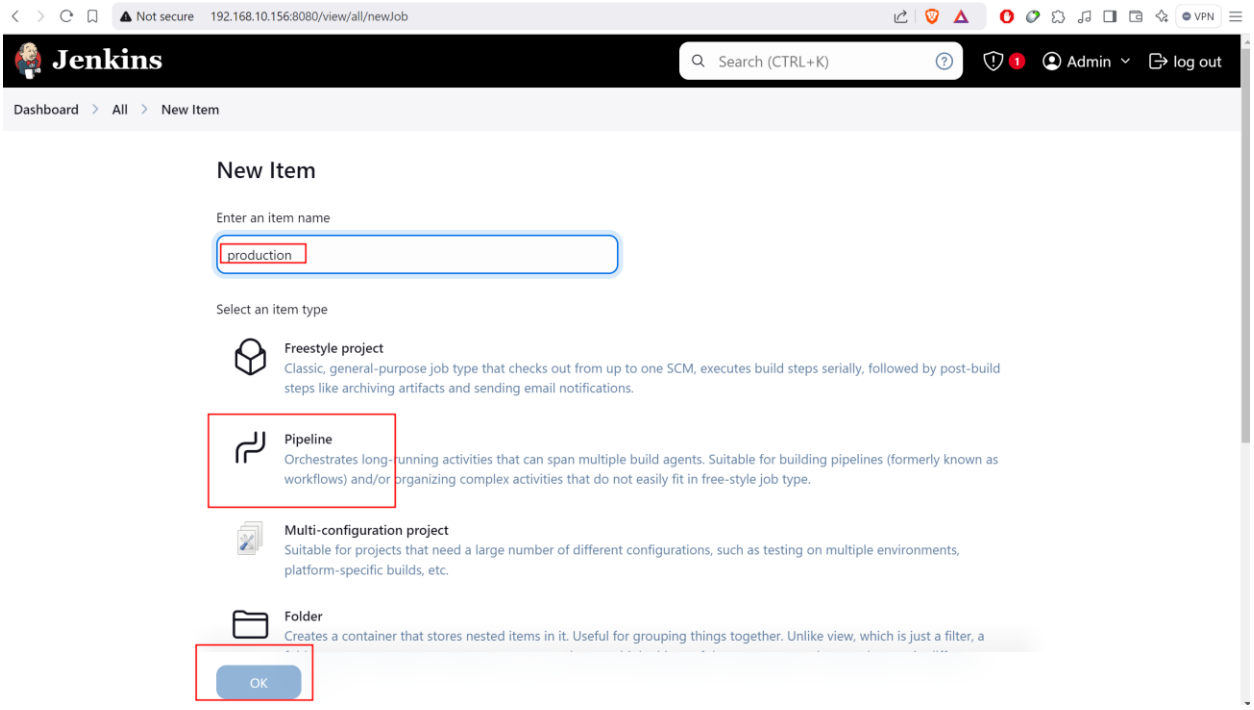


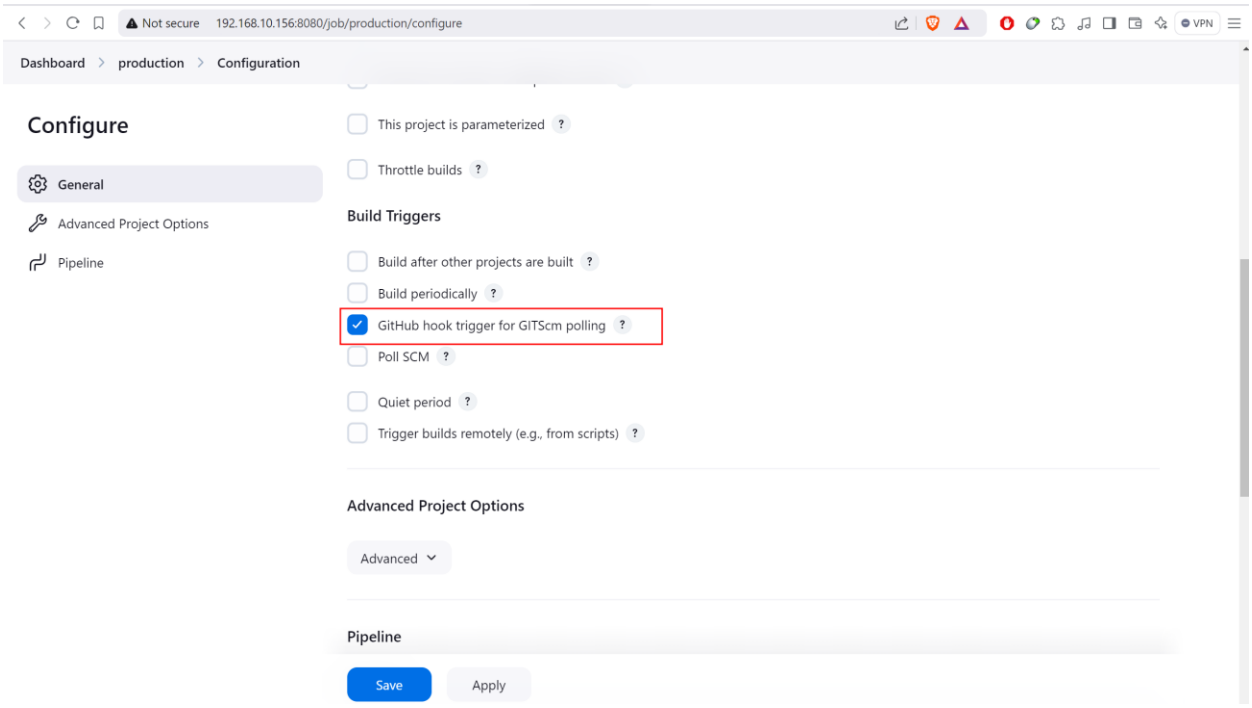
Production Pipeline (Production Environment)

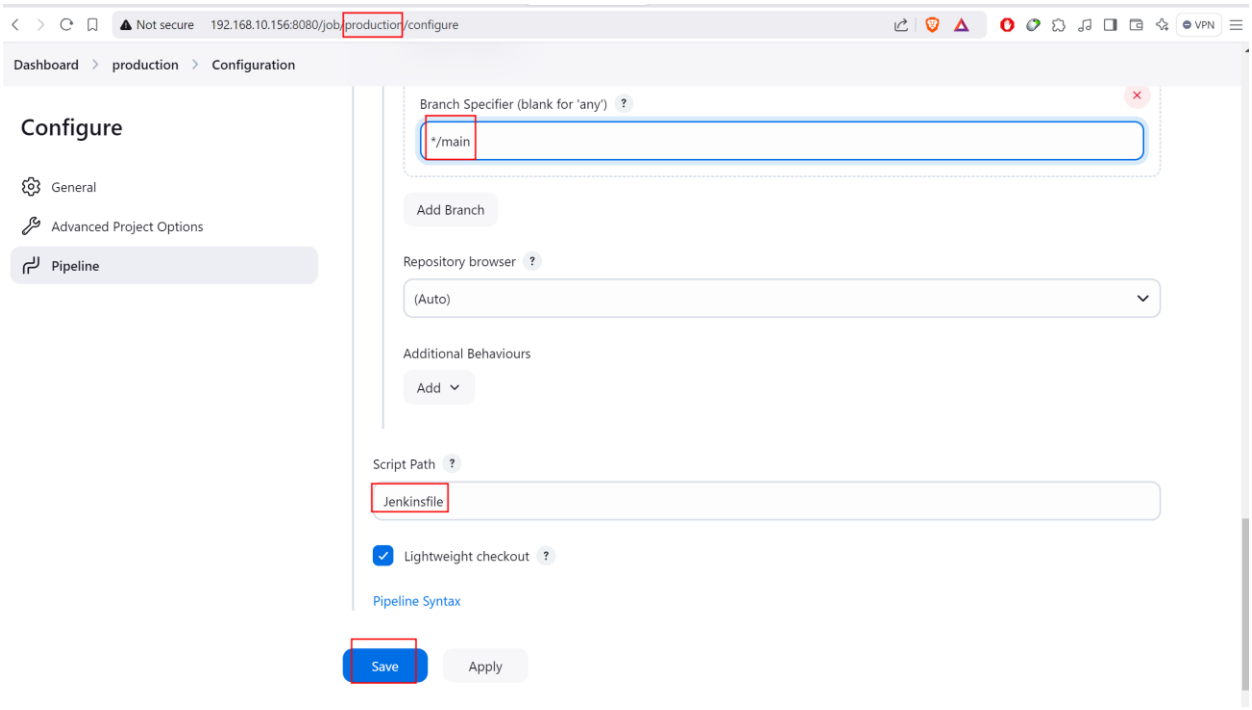
Finally, after all the testing of code and images in development and staging environments, we create a production environment like the staging with three virtual machines and deploy the code.

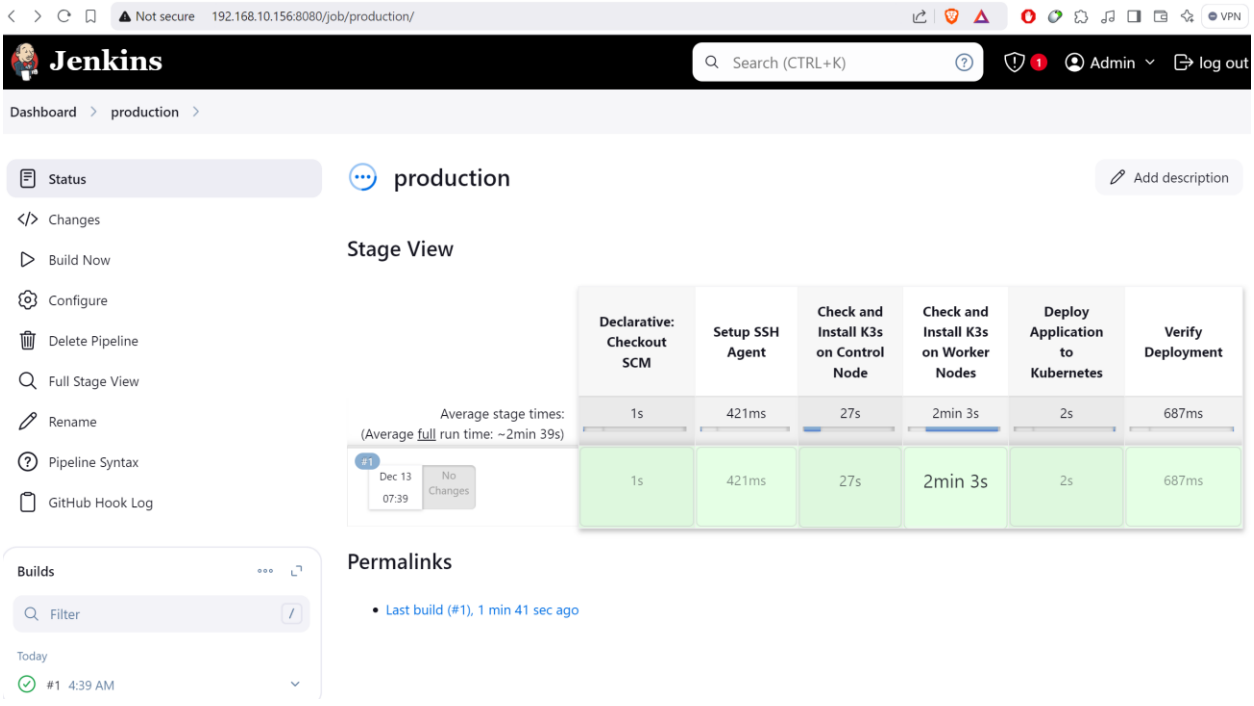
So, the setup is the same with the private SSH key saved in Jenkins and public keys in each of the worker and control nodes. Then, we create the pipeline and see the result of deployment.

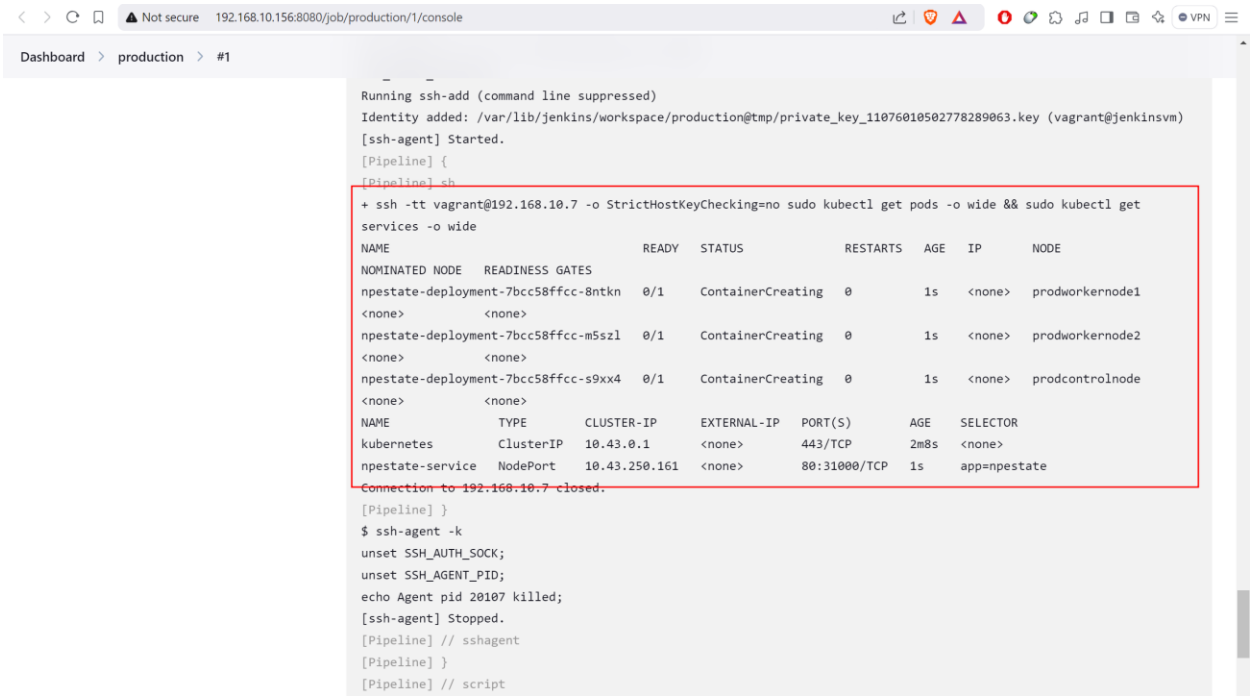

Not secure 192.168.10.156:8080
Jenkins
Search (CTRL+K)
Admin
log out
Dashboard
New Item
Build History
Project Relationship
Check File Fingerprint
Manage Jenkins
My Views
Add description
All
S
W
Name
Last Success
Last Failure
Last Duration
development
35 min #2
47 min #1
5 min 27 sec
staging
23 min #2
29 min #1
2 min 48 sec
Build Queue
No builds in the queue.
Build Executor Status
0/2
Icon: S M L

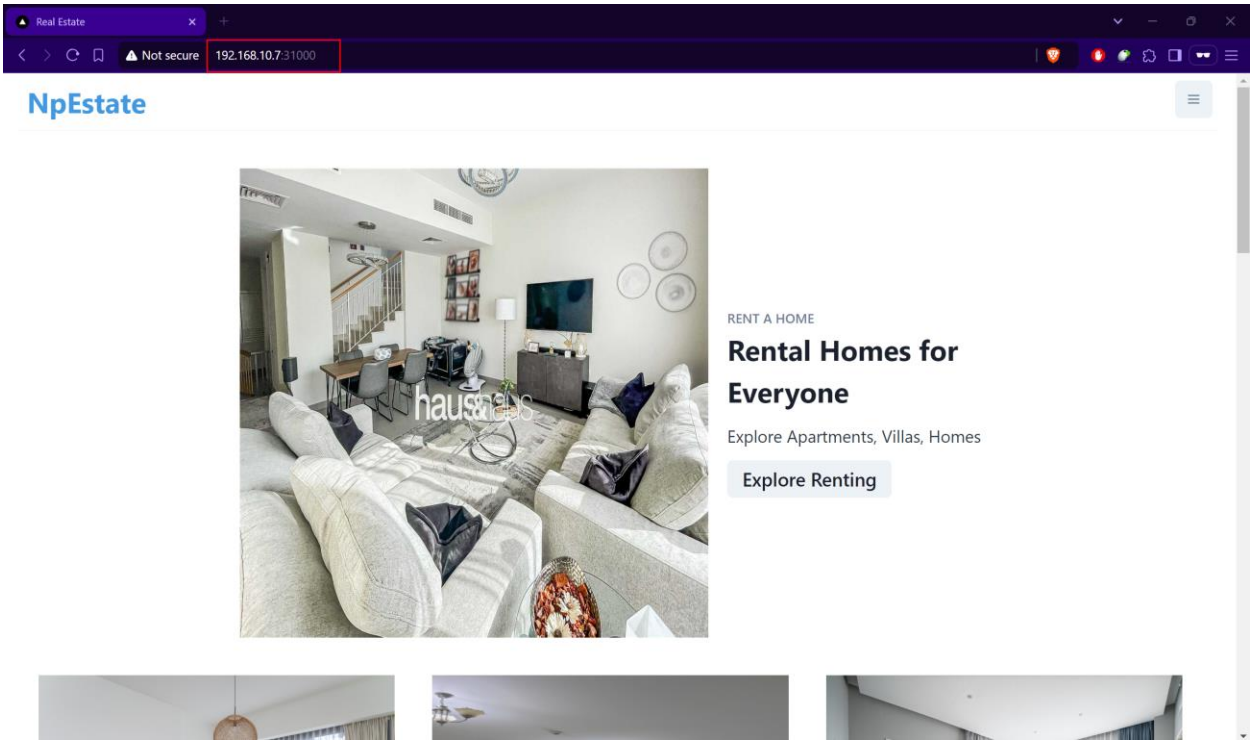

Not secure 192.168.10.156:8080/view/all/newJob
Jenkins
Search (CTRL+K)
Admin
log out
Dashboard
All
New Item
New Item
Enter an item name
production
Select an item type
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build 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
OK


Not secure 192.168.10.156:8080/job/production/configure
Dashboard > production > Configuration
Configure
General
Advanced Project Options
Pipeline
This project is parameterized
Throttle builds
Build Triggers
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Quiet period
Trigger builds remotely (e.g., from scripts)
Advanced Project Options
Advanced
Pipeline
Save
Apply

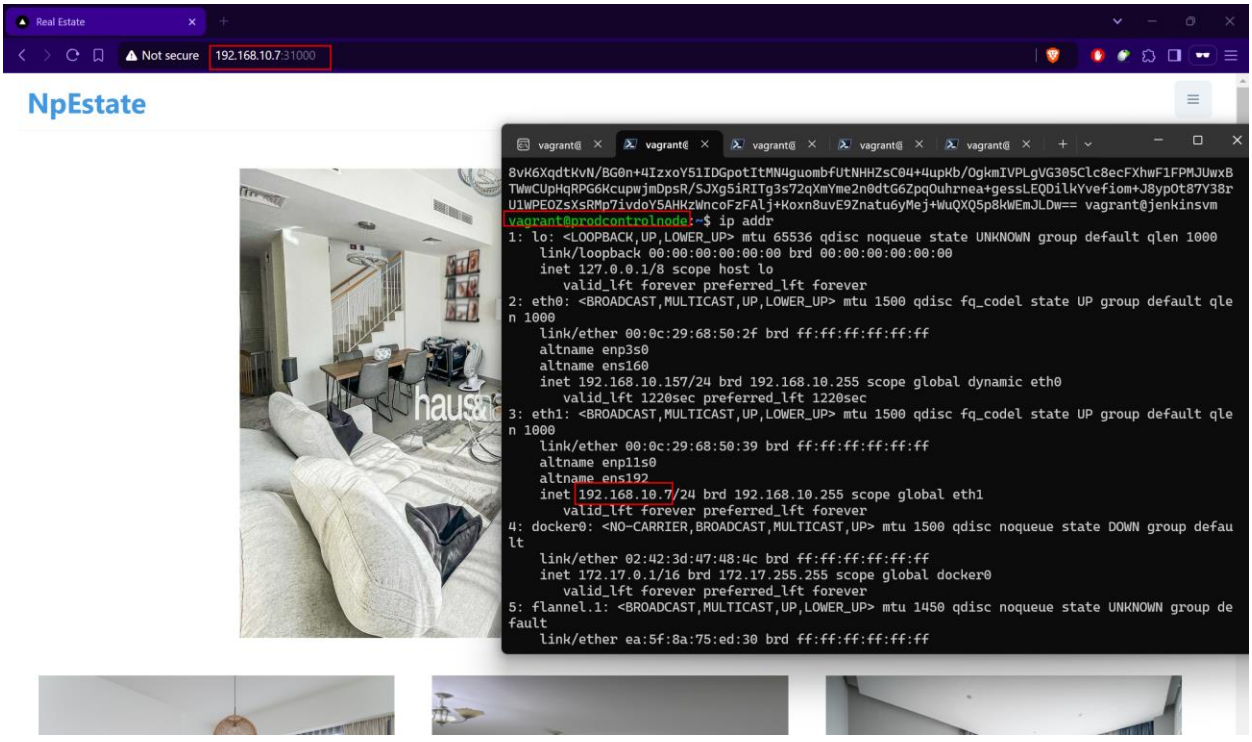

Not secure 192.168.10.156:8080/job/production/configure
Dashboard > production > Configuration
Configure
General
Advanced Project Options
Pipeline
Branch Specifier (blank for 'any')
*/main
Add Branch
Repository browser
(Auto)
Additional Behaviours
Add
Script Path
Jenkinsfile
Lightweight checkout
Pipeline Syntax
Save
Apply


Not secure 192.168.10.156:8080/job/production/
Jenkins
Search (CTRL+K)
Admin
log out
Dashboard
production
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
GitHub Hook Log
production
Add description
Stage View
Declarative: Checkout SCM
Setup SSH Agent
Check and Install K3s on Control Node
Check and Install K3s on Worker Nodes
Deploy Application to Kubernetes
Verify Deployment
Average stage times:
(Average full run time: ~2min 39s)
1s
421ms
27s
2min 3s
2s
687ms
#1
Dec 13
07:39
No Changes
Permalinks
Last build (#1), 1 min 41 sec ago
Builds
Filter
Today
#1 4:39 AM


Not secure 192.168.10.156:8080/job/production/1/console
Dashboard
production
#1
Running ssh-add (command line suppressed)
Identity added: /var/lib/jenkins/workspace/production@tmp/private_key_11076010502778289063.key (vagrant@jenkinsvm)
[ssh-agent] Started.
[Pipeline] {
[Pipeline] sh
+ ssh -tt vagrant@192.168.10.7 -o StrictHostKeyChecking=no sudo kubectl get pods -o wide && sudo kubectl get services -o wide
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES
npestate-deployment-7bcc58ffcc-8ntkn 0/1 ContainerCreating 0 1s <none> prodworkernode1 <none> <none>
npestate-deployment-7bcc58ffcc-m5szl 0/1 ContainerCreating 0 1s <none> prodworkernode2 <none> <none>
npestate-deployment-7bcc58ffcc-s9xx4 0/1 ContainerCreating 0 1s <none> prodcontrolnode <none> <none>
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR
kubernetes ClusterIP 10.43.0.1 <none> 443/TCP 2m8s <none>
npestate-service NodePort 10.43.250.161 <none> 80:31000/TCP 1s app=npestate
Connection to 192.168.10.7 closed.
[Pipeline] }
$ ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 20107 killed;
[ssh-agent] Stopped.
[Pipeline] // sshagent
[Pipeline] }
[Pipeline] // script



We can see that the IP of the control node in the production environment lets us access the web application on port 31000. So, the deployment was successful.



Conclusion

The project was a great way to build the foundation for CI/CD implementation and DevOps. We got to learn about many basic and very important concepts, advantages, and complications of the CI/CD implementation. Beginning from Vagrant, Git and GitHub to Jenkins, Docker and Kubernetes, we were able to create a complete working CI/CD implementation with proper testing of code and deployment in different environments.