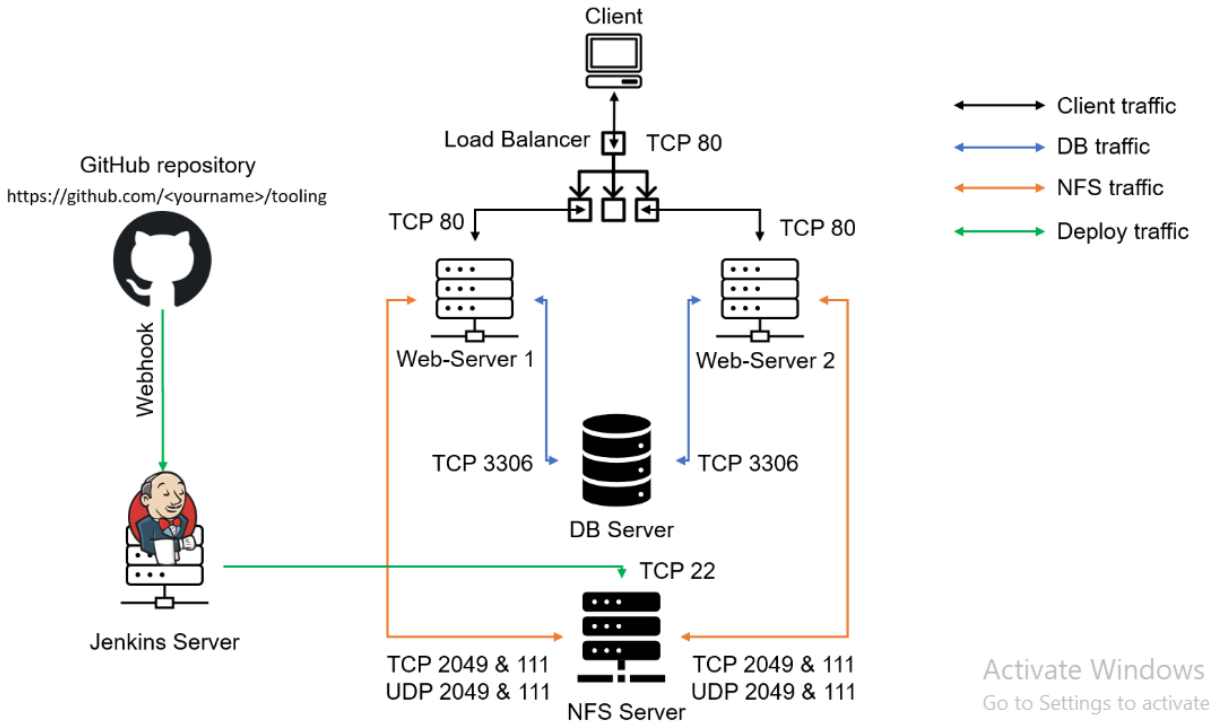
PROJECT 9

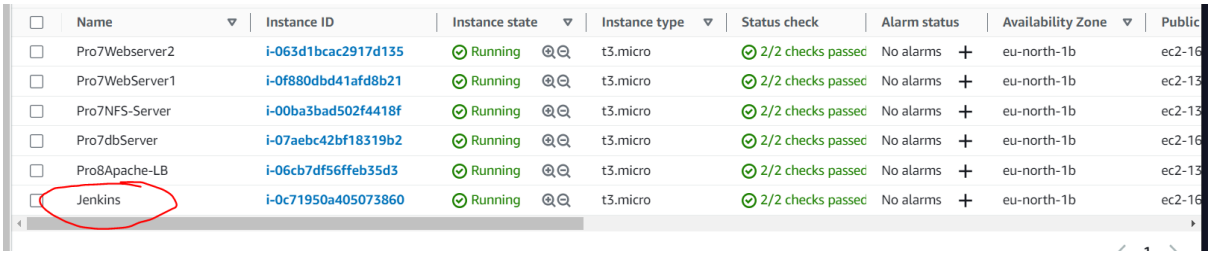


Continous Integration Pipeline For Tooling Website

INSTALL AND CONFIGURE JENKINS SERVER

Step1: Install Jenkins server

- e an AWS EC2 server based on Ubuntu Server 20.04 LTS and name it "Jenkins"



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public
Pro7Webserver2	i-063d1bcac2917d135	Running	t3.micro	2/2 checks passed	No alarms	eu-north-1b	ec2-16
Pro7WebServer1	i-0f880dbd41afd8b21	Running	t3.micro	2/2 checks passed	No alarms	eu-north-1b	ec2-13
Pro7NFS-Server	i-00ba3bad502f4418f	Running	t3.micro	2/2 checks passed	No alarms	eu-north-1b	ec2-13
Pro7dbServer	i-07aebc42bf18319b2	Running	t3.micro	2/2 checks passed	No alarms	eu-north-1b	ec2-16
Pro8Apache-LB	i-06cb7df56ffeb35d3	Running	t3.micro	2/2 checks passed	No alarms	eu-north-1b	ec2-13
Jenkins	i-0c71950a405073860	Running	t3.micro	2/2 checks passed	No alarms	eu-north-1b	ec2-16

- Install JDK (since Jenkins is a Java-based application)

```
sudo apt update
sudo apt install default-jdk-headless
```

```
ubuntu@ip-172-31-38-178:~$ sudo apt install default-jdk-headless
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java default-jre-headless fontconfig-config fonts-dejavu-core java-common libavahi-client3 libavahi-common-data libavahi-common3 libcups2
  libfontconfig1 libgraphite2-3 libharfbuzz0b libjpeg-turbo8 libjpeg8 liblcms2-2 libpcsclite1 openjdk-11-jdk-headless openjdk-11-jre-headless
Suggested packages:
  default-jre cups-common liblcms2-utils pcscd openjdk-11-demo openjdk-11-source libnss-mdns fonts-dejavu-extra fonts-ipafont-gothic fonts-ipafont-mincho
  fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  ca-certificates-java default-jdk-headless default-jre-headless fontconfig-config fonts-dejavu-core java-common libavahi-client3 libavahi-common-data
  libavahi-common3 libcups2 libfontconfig1 libgraphite2-3 libharfbuzz0b libjpeg-turbo8 libjpeg8 liblcms2-2 libpcsclite1 openjdk-11-jdk-headless
  openjdk-11-jre-headless
```

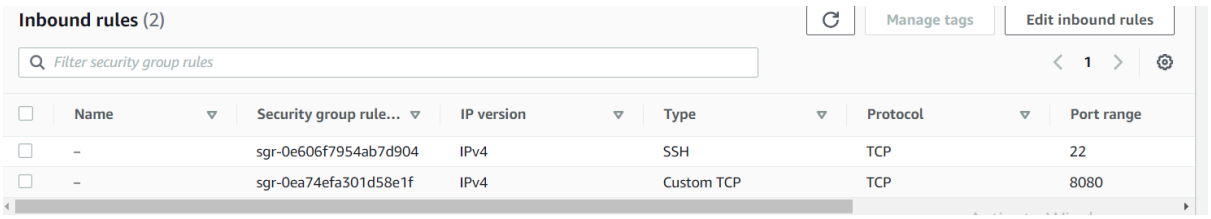
- Install Jenkins

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
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 -y jenkins
```

```
ubuntu@ip-172-31-38-178:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2023-07-17 17:54:14 UTC; 49s ago
   Main PID: 5289 (java)
      Tasks: 42 (limit: 1111)
     Memory: 304.6M
     CGroup: /system.slice/jenkins.service
             └─5289 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Jul 17 17:53:34 ip-172-31-38-178 jenkins[5289]: 5469278a32464534892d30c853336790
Jul 17 17:53:34 ip-172-31-38-178 jenkins[5289]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jul 17 17:53:34 ip-172-31-38-178 jenkins[5289]: ************************************************************
Jul 17 17:53:34 ip-172-31-38-178 jenkins[5289]: ************************************************************
Jul 17 17:53:34 ip-172-31-38-178 jenkins[5289]: ************************************************************
Jul 17 17:54:14 ip-172-31-38-178 jenkins[5289]: 2023-07-17 17:54:14.914+0000 [id=30]        INFO        jenkins.InitReactorRunner$1#onAttained: Completed initializati
Jul 17 17:54:14 ip-172-31-38-178 jenkins[5289]: 2023-07-17 17:54:14.935+0000 [id=22]        INFO        hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and ru
Jul 17 17:54:14 ip-172-31-38-178 systemd[1]: Started Jenkins Continuous Integration Server.
Jul 17 17:54:15 ip-172-31-38-178 jenkins[5289]: 2023-07-17 17:54:15.331+0000 [id=46]        INFO        h.m.DownloadService$Downloadable#load: Obtained the updated da
Jul 17 17:54:15 ip-172-31-38-178 jenkins[5289]: 2023-07-17 17:54:15.331+0000 [id=46]        INFO        hudson.util.Retrier#start: Performed the action check updates
```


- By default Jenkins server uses TCP port 8080 – open it by creating a new Inbound Rule in your EC2 Security Group



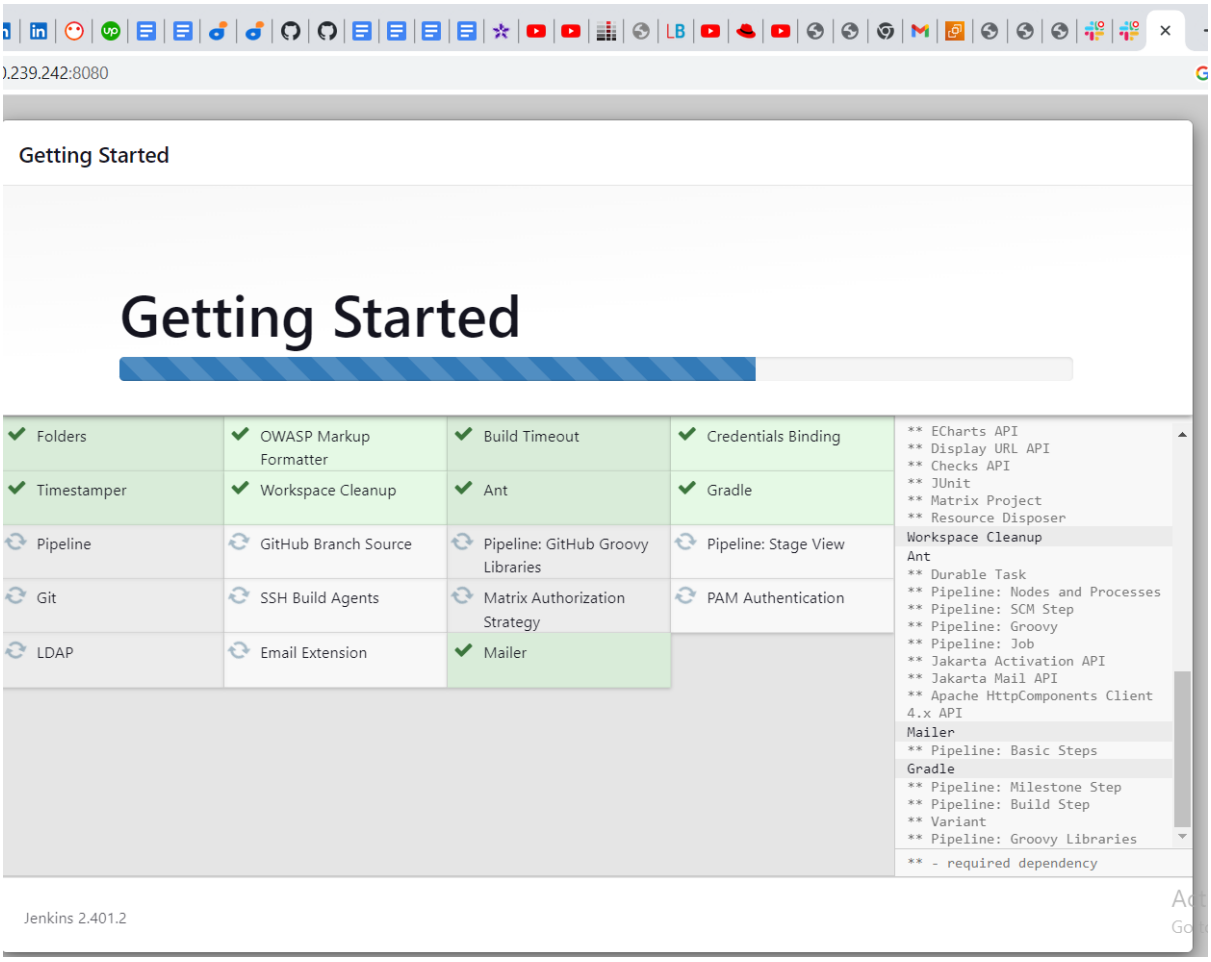
- Initial Jenkins setup.

From your browser access

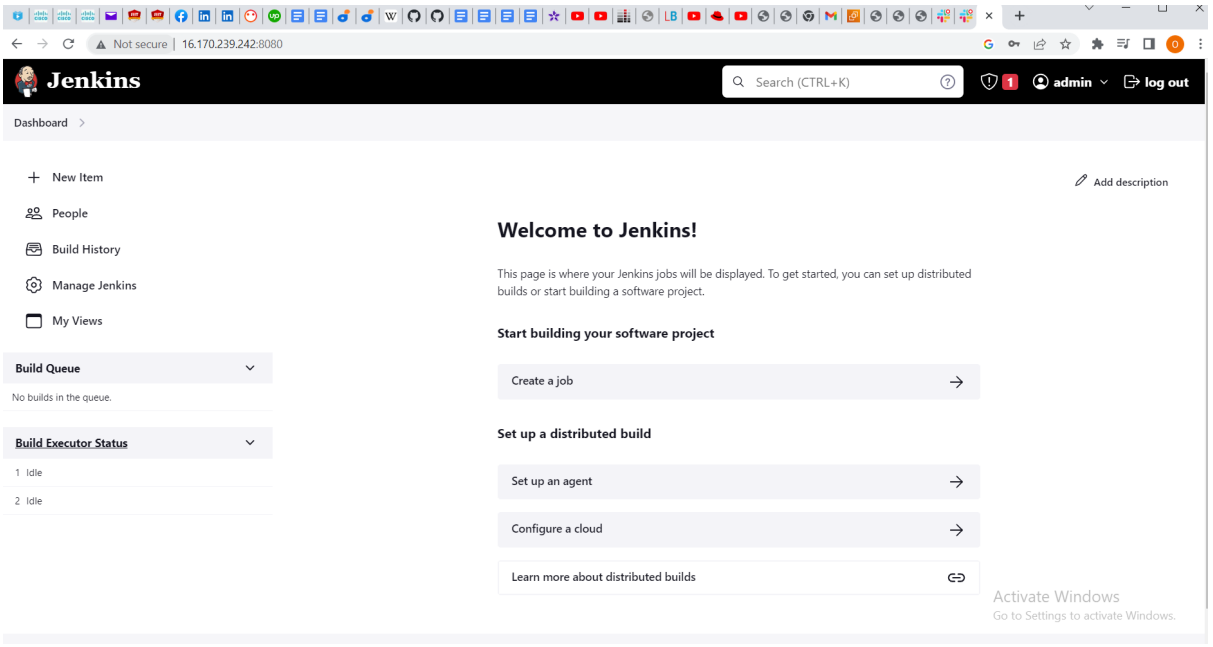
`http://<Jenkins-Server-Public-IP-Address-or-Public-DNS-Name>:8080`

Retrieve it from your server: sudo cat /var/lib/jenkins/secrets/initialAdminPassword

```
ubuntu@ip-172-31-38-178:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
5469278a32464534892d30c853336790
ubuntu@ip-172-31-38-178:~$
```



Click on get started;

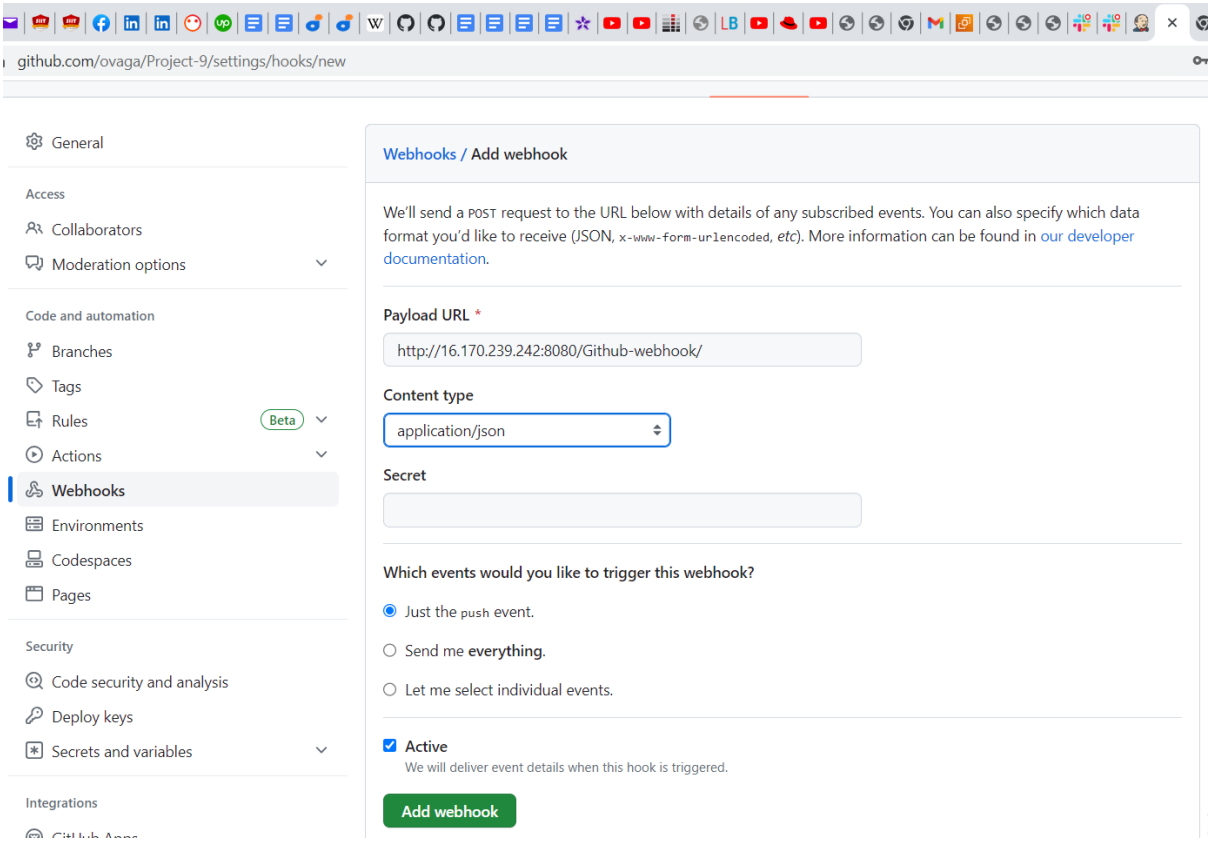


- Configure Jenkins to retrieve source codes from GitHub using Webhooks

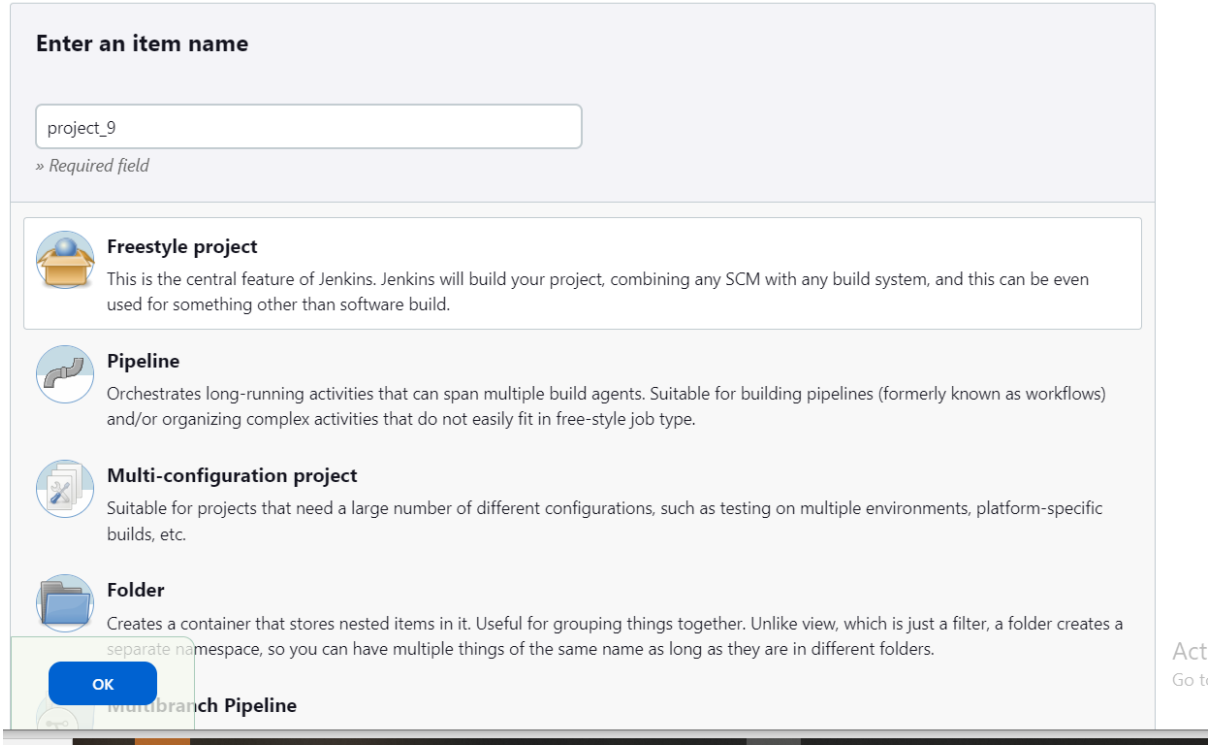
NB: We will configure a simple Jenkins job/project (these two terms can be used interchangeably). This job will be triggered by GitHub webhooks and will execute a 'build' task to retrieve codes from GitHub and store it locally on Jenkins server.

1. Enable webhooks in your GitHub repository settings

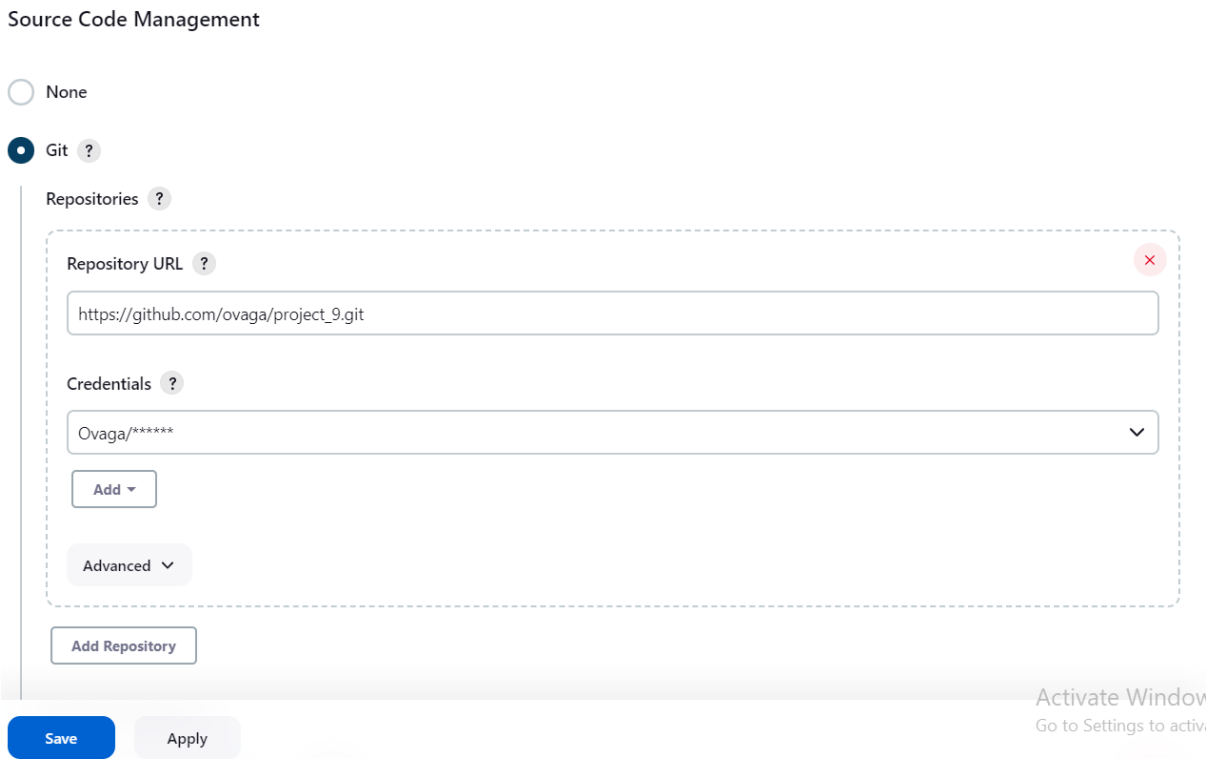
To create webhook, go to the settings tab on the github repo and click on webhooks. Webhook should look like this
`<public_ip_of_jenkins_server>:8080/github-webhook/`



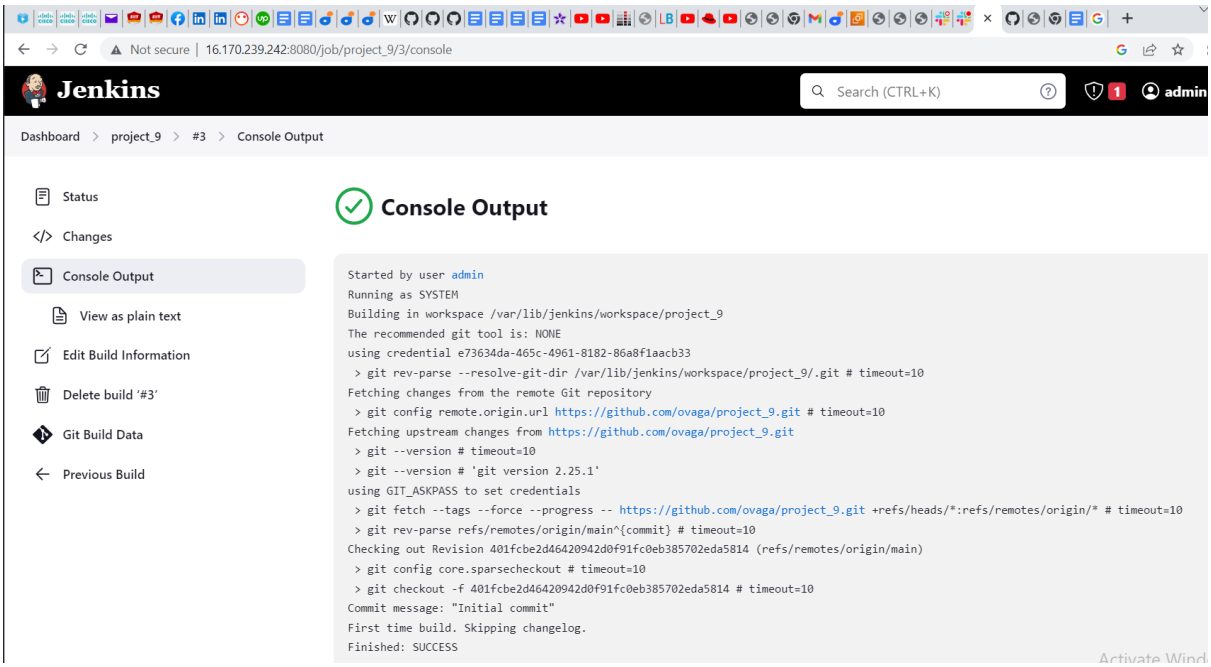
Go to Jenkins web console, click “New Item” and create a “Freestyle project”



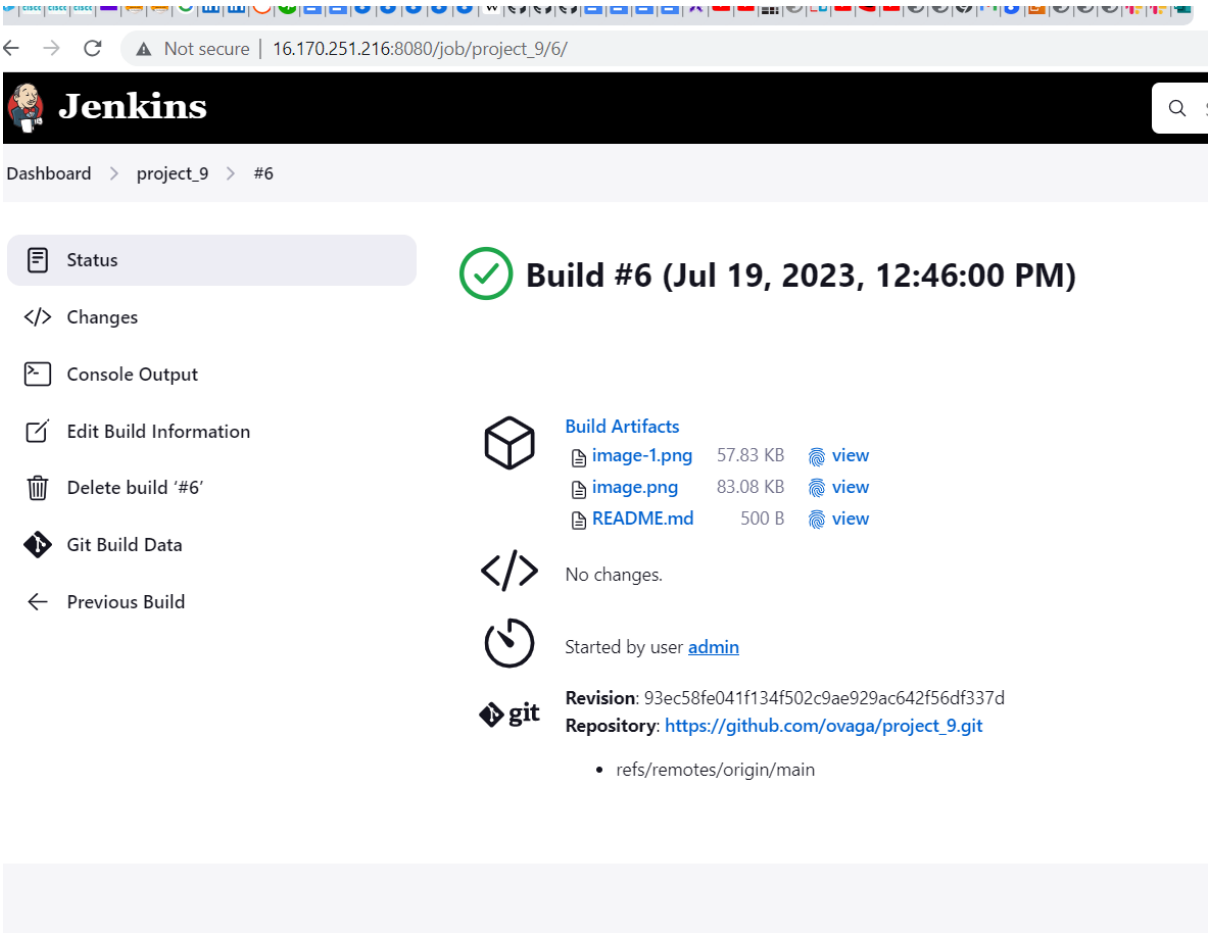
To connect your GitHub repository, you will need to provide its URL, you can copy from the repository itself. Provide the credentials (user/password) so Jenkins could access files in the repository.



Save the configuration and try to run the build. For now we can only do it manually.



NB: The build is successful as shown in the output. However, it runs only when we trigger it manually. Let us fix it

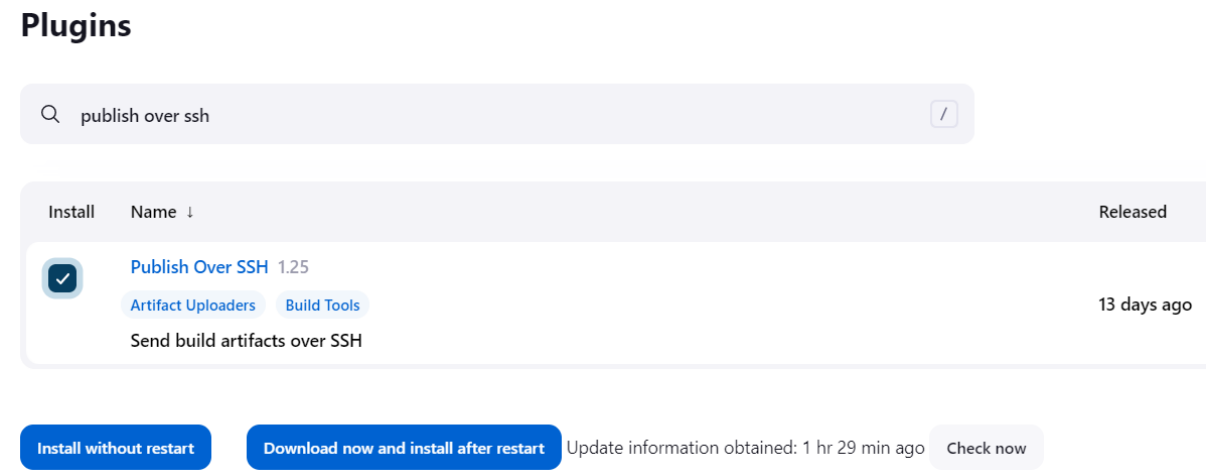


Configuring Jenkins To Copy Files(Artifact) to NFS Server

To achieve this, we install the `Publish Via SSH` pluging on Jenkins. The plugin allows one to send newly created packages to a remote server and install them, start and stop services that the build may depend on and many other use cases.

On main dashboard select "Manage Jenkins" and choose "Manage Plugins" menu item.

On "Available" tab search for "Publish Over SSH" plugin and install it



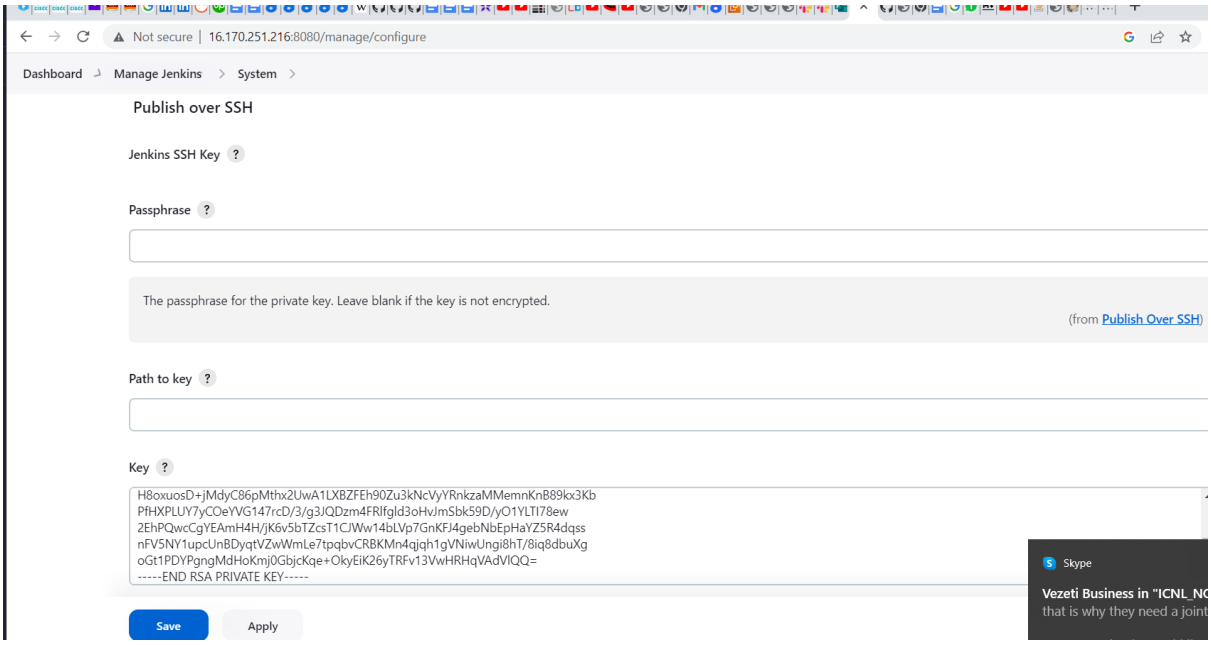
Configure the job to copy artifacts over to NFS server. On main dashboard select "Manage Jenkins" and choose "Configure System" menu item.

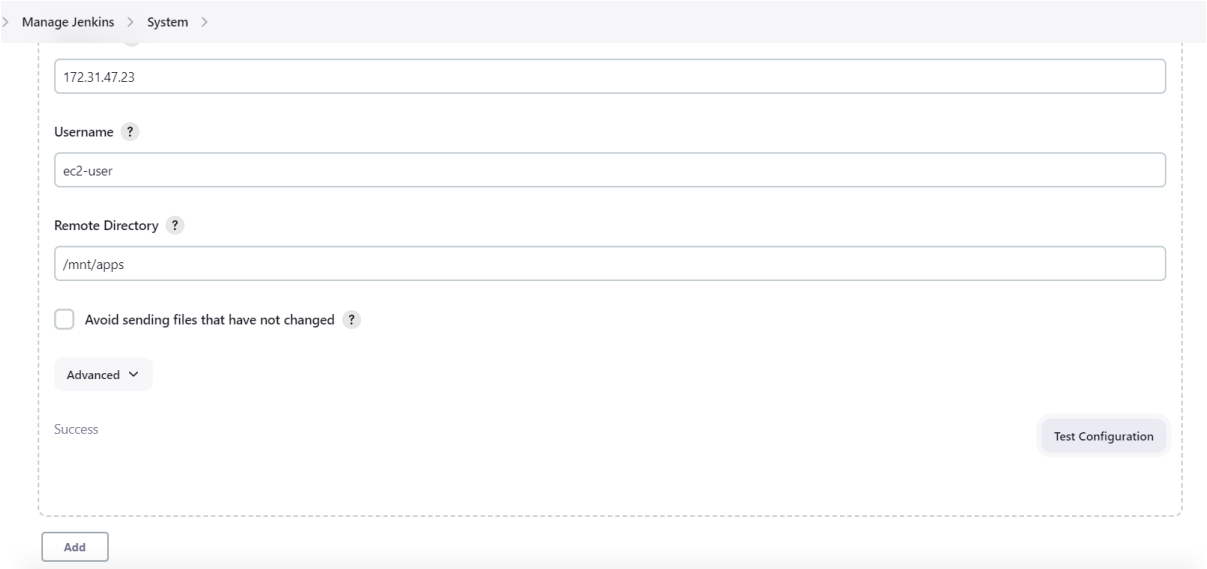
Scroll down to Publish over SSH plugin configuration section and configure it to be able to connect to the NFS server:

Provide a private key (content of .pem file that you use to connect to NFS server via SSH/Putty)

Hostname – can be private IP address of NFS server
Username – ec2-user (since NFS server is based on EC2 with RHEL 8)
Remote directory – /mnt/apps since our Web Servers use it as a mointing point to retrieve files from the NFS server

Test the configuration and make sure the connection returns Success. Remember, that TCP port 22 on NFS server must be open to receive SSH connections.

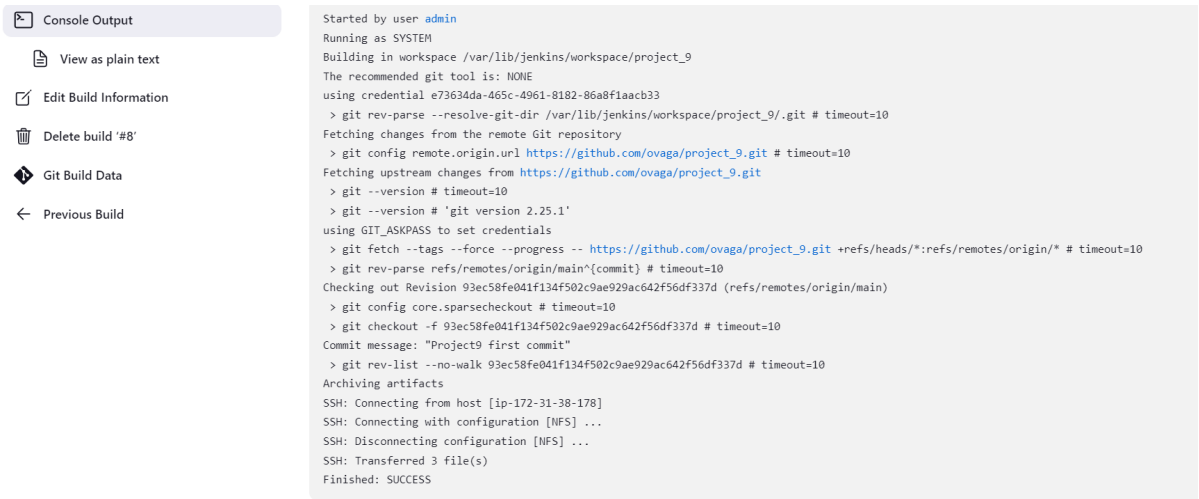




Webhook will trigger a new job and in the “Console Output” of the job you will find something like this:

```
SSH: Transferred 25 file(s)
```

```
Finished: SUCCESS
```



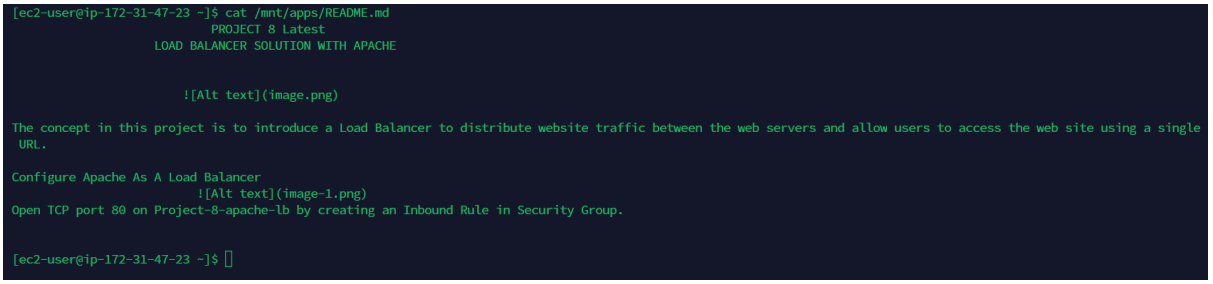
o make sure that the files in

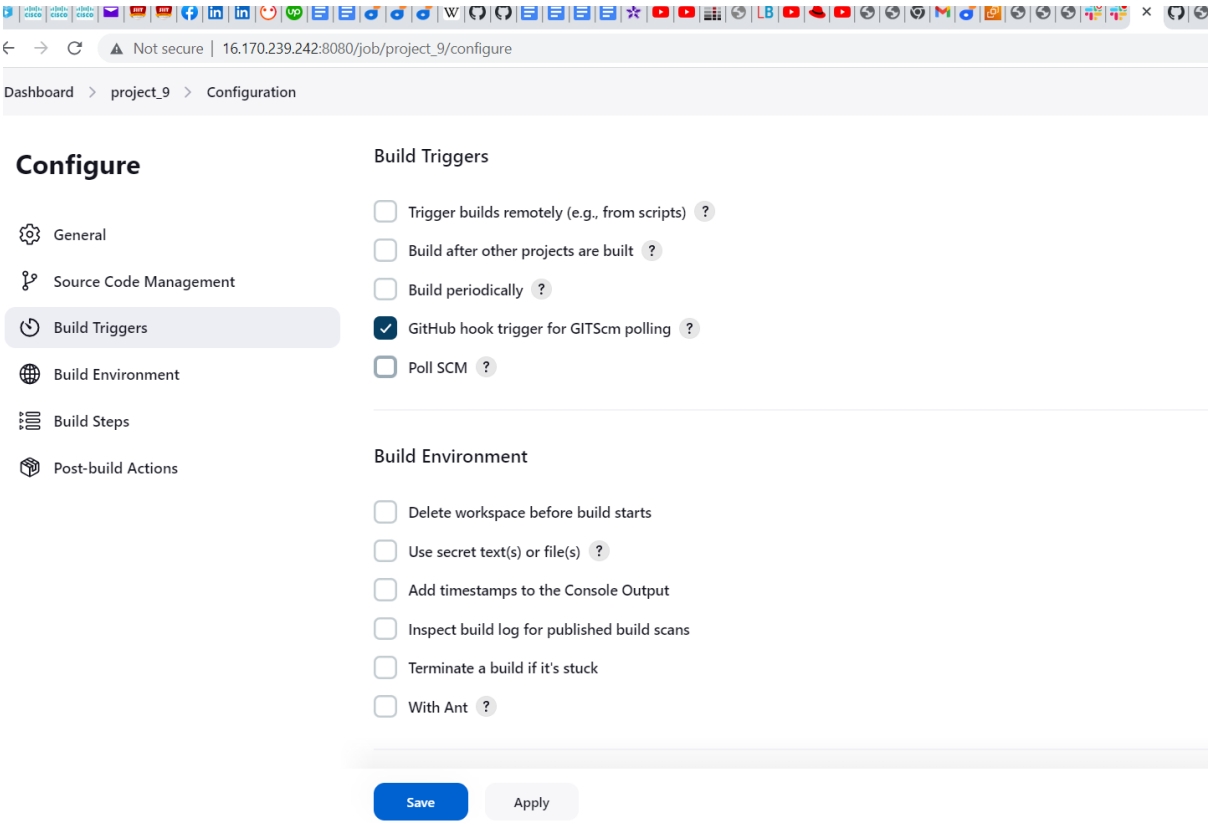
`/mnt/apps`

have been updated – connect via SSH/Putty to your NFS server and check README.MD file

`cat /mnt/apps/README.md`

If you see the changes you had previously made in your GitHub – the job works as expected





- Configure “Post-build Actions” to archive all the files – files resulted from a build are called “artifacts”