CONTINUOUS INTEGRATION PIPELINE FOR TOOLING WEBSITE

A DevOps team utilizes various tooling solutions in order to help the team carry out their day-to day activities in managing, developing, testing, deploying and monitoring different projects. This project also majors on how Jenkins contribute to the DevOps process by continually integrating developer codes in a timely manner and shows how it connects with the webserver to deliver a deployable software products.

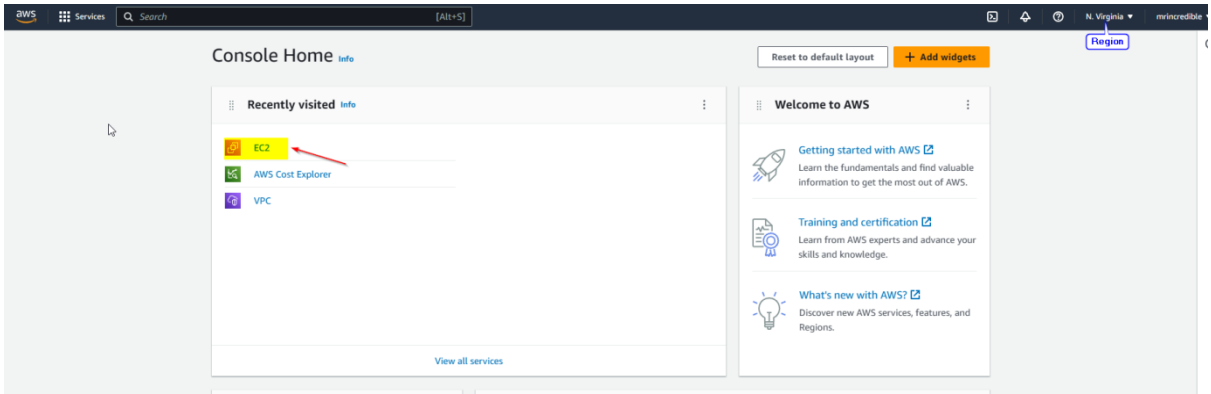
In this project we would be implementing a DevOps solution that consist of the following

Pre-requisite for the projects is the following.

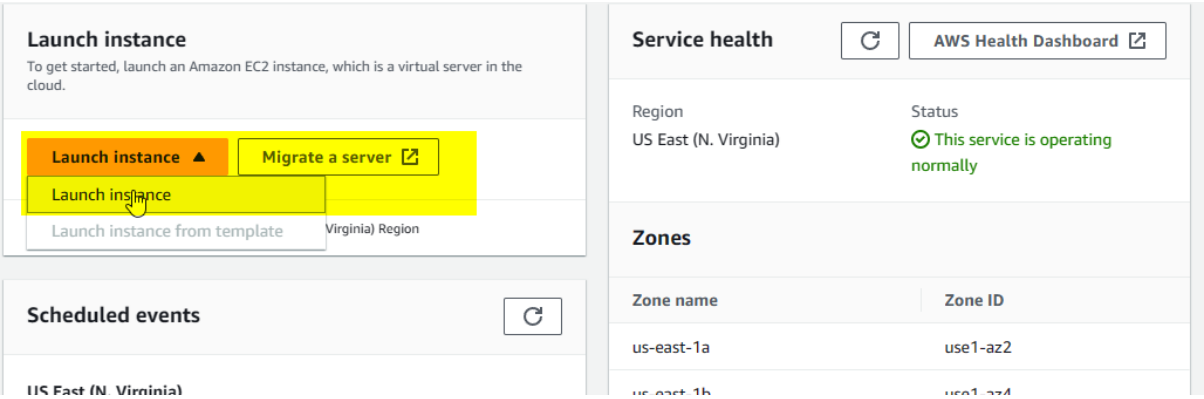
1) Fundamental Knowledge of Installing and downloading software
2) Basic Understanding of Linux Commands
3) AWS account login with EC2 instances
4) Webserver Linux: Ubuntu 20.4 LTS
5) NFS Server: Red Hat Enterprise Linux 9
6) Code Repository: GitHub
7) Internet connection

IMPLEMENTATION STEPS:

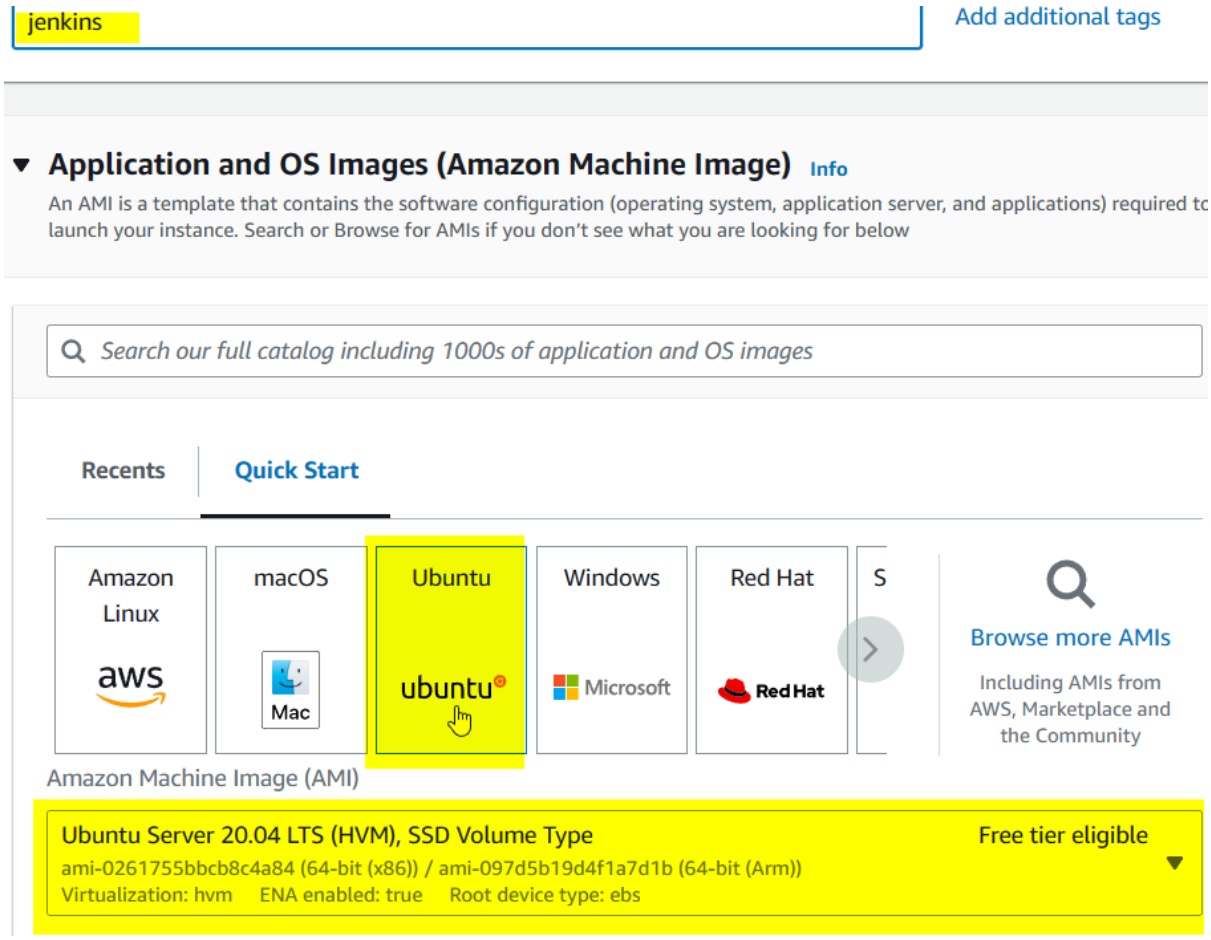
i) Ensure you login with your details to your AWS console via the https://aws.amazon.com
ii) Click on the EC2 link to create instances.



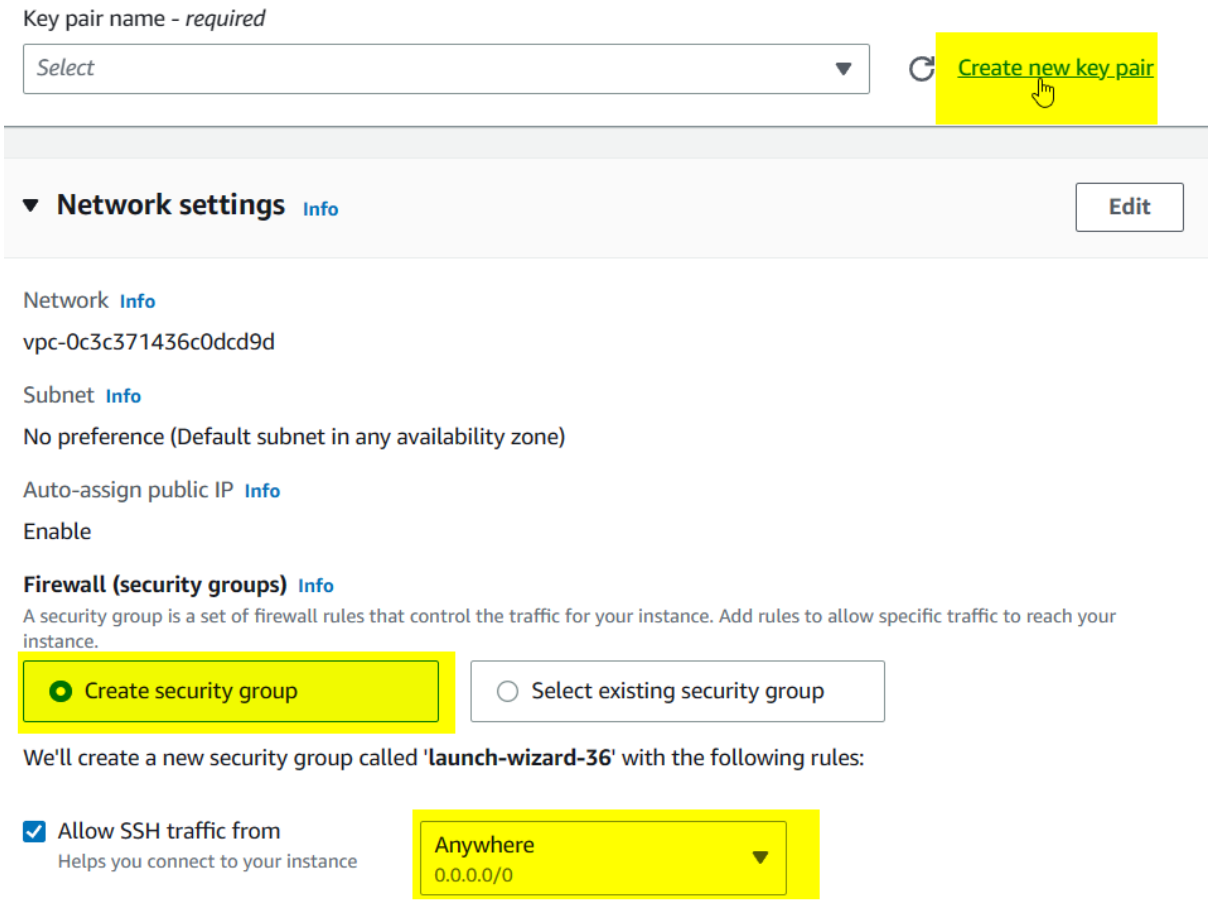
iii)Click on launch instance dropdown button and select launch



Select Ubuntu from the quick start option and note that amazon machine image selection varies from user to user .Select Ubuntu 20.4 LTS SSD Volume type .



Click on the “Create new key pair” link and ensure the Checkbox remains unchanged on the “Create security group”.



Key pair name

Key pairs allow you to connect to your instance securely.

jenkins

The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA
RSA encrypted private and public key pair

ED25519
ED25519 encrypted private and public key pair

Private key file format

.pem
For use with OpenSSH

.ppk
For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

Cancel

Create key pair

Once new key pair is created we select 1 and launch instance

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Configure storage Info

Advanced

1x 8 GiB gp2 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

0 x File systems

Edit

Number of instances Info

1

Software Image (AMI)

Canonical, Ubuntu, 20.04 LTS, ...read more

ami-0261755bbcb8c4a84

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

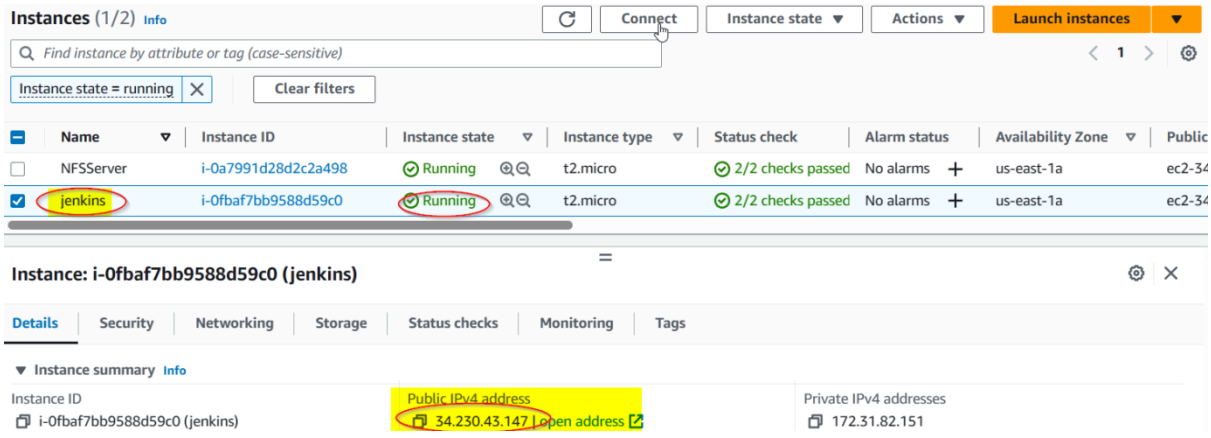
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier

Cancel

Launch instance

Click to connect to ssh



JENKINS SERVER CONFIGURATION

To configure Jenkins, we can navigate to the documentation on their website and look at it to understand the platform of download that suits you .In this case we are using the Ubuntu Debian .

Open git bash on visual studio code or whichever console is convenient to use. We are using git bash here with Visual Studio Code

We rename the ip address as webserver as seen below and perform an update check and installing the default headless jdk as shown below.

```
oshor@Oshority MINGW64 ~/Downloads (master)
$ ssh -i "jenkins.pem" ubuntu@ec2-34-230-43-147.compute-1.amazonaws.com
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1036-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
```

```
ubuntu@ip-172-31-82-151:~$ sudo hostname jenkins-server
ubuntu@ip-172-31-82-151:~$ bash
ubuntu@jenkins-server:~$ sudo apt update
sudo apt install default-jdk-headless
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
```

We then proceed to install Jenkins and to ensure easy installation add the key and perform the actions as seen below .

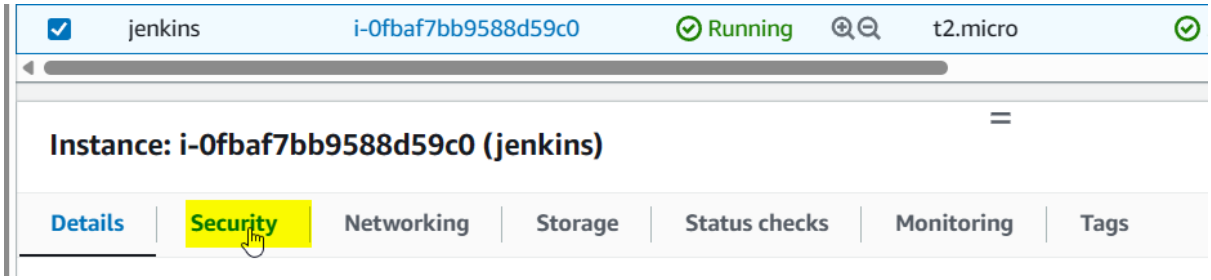
```
ubuntu@jenkins-server:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:5 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:7 http://security.ubuntu.com/ubuntu focal-security InRelease
Get:8 https://pkg.jenkins.io/debian-stable binary/ Packages [24.9 kB]
Fetched 27.8 kB in 1s (36.5 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
76 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@jenkins-server:~$ sudo apt-get install jenkins -y
Reading package lists... Done
Building dependency tree
```

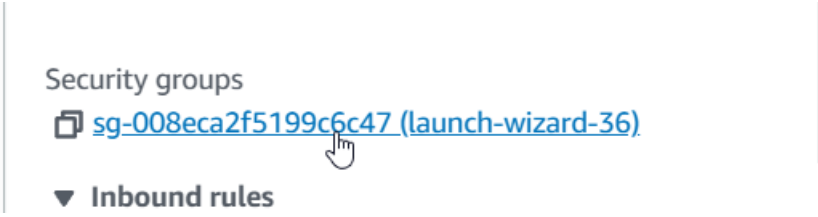
```
ubuntu@jenkins-server:~$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-ke
y add
OK
ubuntu@jenkins-server:~$ udo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \
>     /etc/apt/sources.list.d/jenkins.list'

Command 'udo' not found, but can be installed with:

sudo apt install udo

ubuntu@jenkins-server:~$ sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \
    /etc/apt/sources.list.d/jenkins.list'
ubuntu@jenkins-server:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
```

Once installed ,we can confirm if Jenkins was successfully installed .

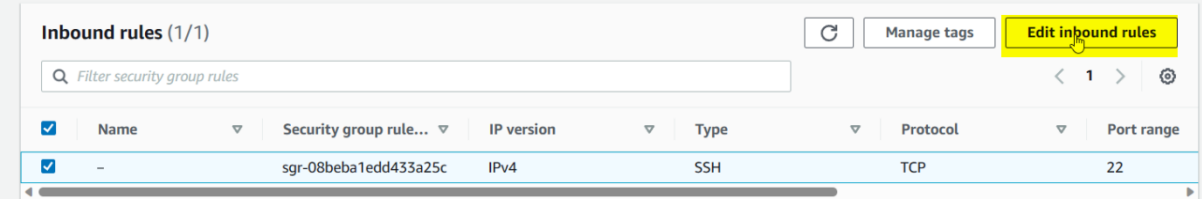
```
ubuntu@jenkins-server:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-06-20 02:57:24 UTC; 3min 26s ago
   Main PID: 4573 (java)
      Tasks: 36 (limit: 1141)
     Memory: 301.5M
     CGroup: /system.slice/jenkins.service
             └─4573 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=
```

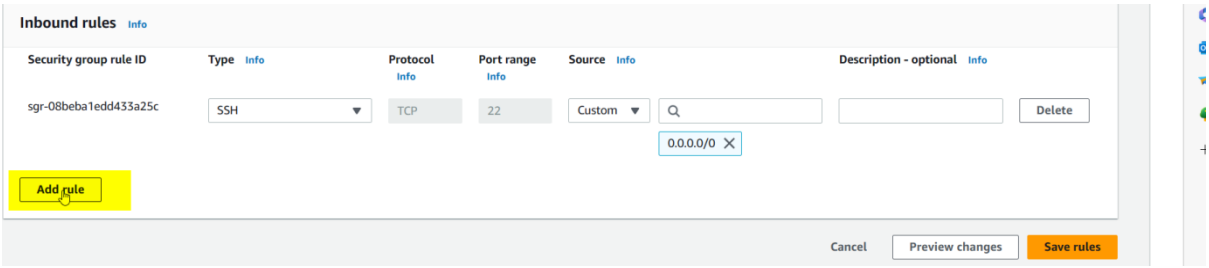
Please note that Jenkins runs on port 8080. Hence we would need to add the security group in the Jenkins server to ensure it successful launched on the web page



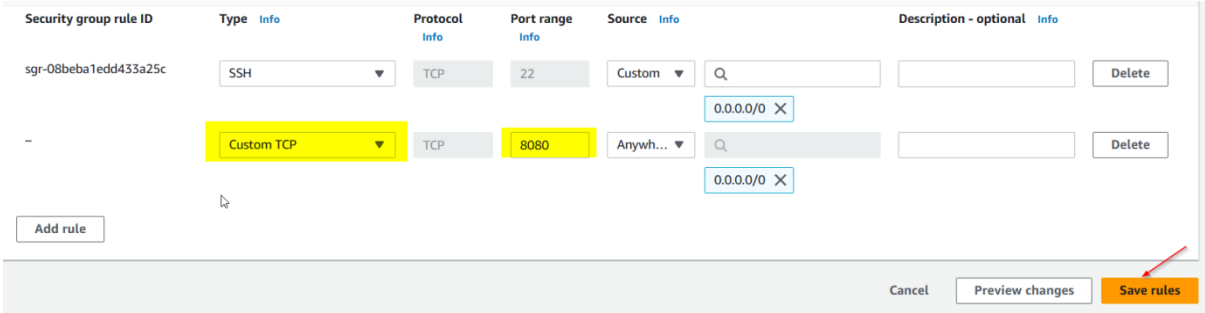


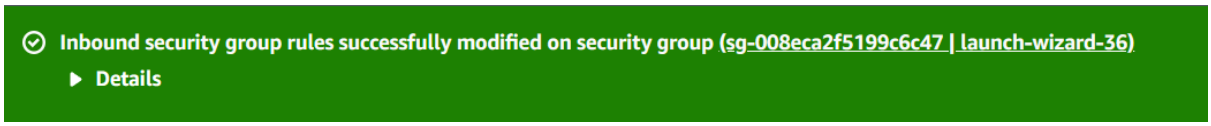
Edit the inbound rules and add the rule of port :8080 for Jenkins





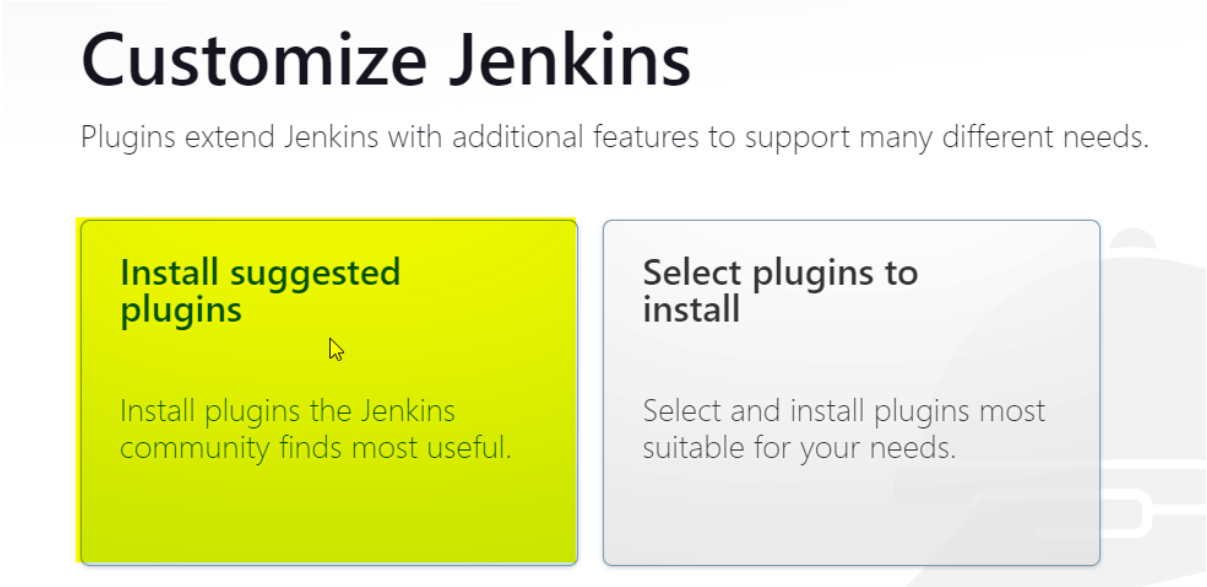
Save the rules

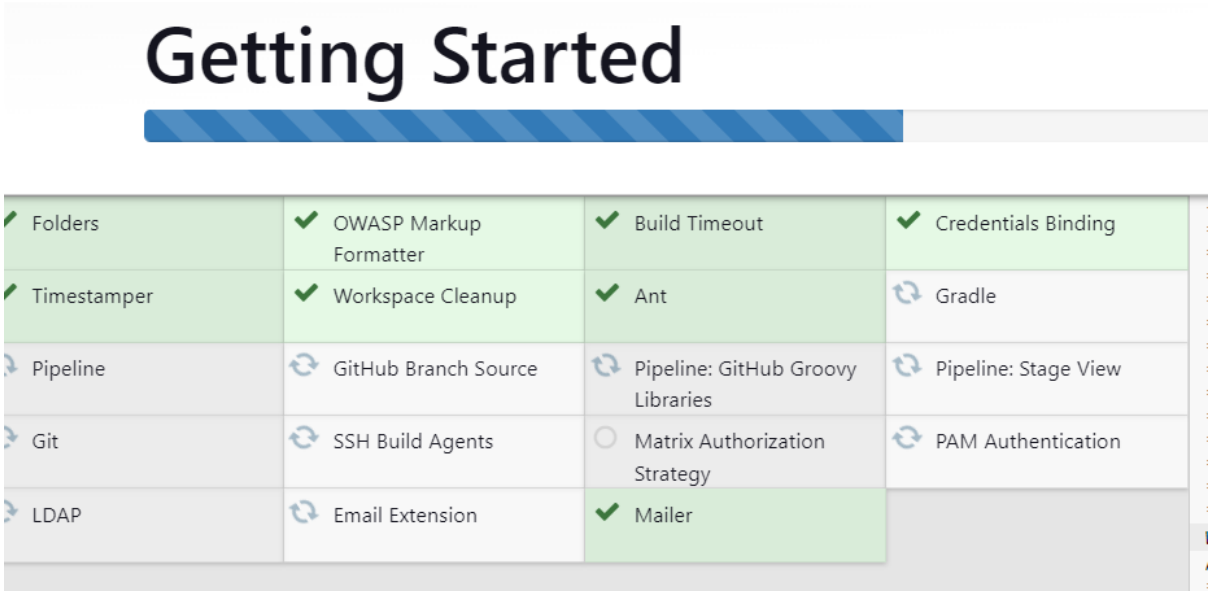




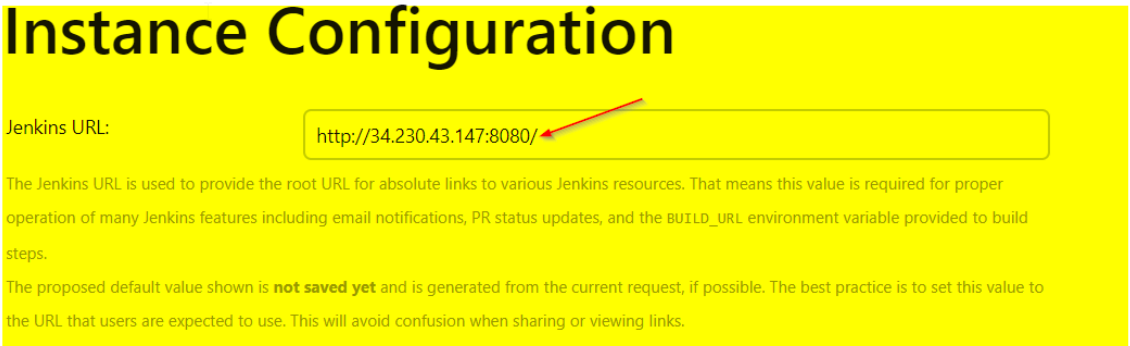
Proceed to launch the public ip address and your Jenkins would be displayed.

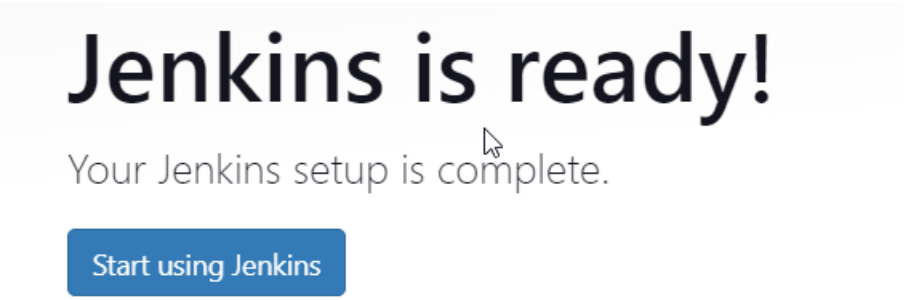
You can retrieve your admin password in your terminal and paste here to access as the admin. Once done you should see this page and click on Installed suggested plugins and have all suggested plugins installed as shown below.





One you fill in your details and have it successful, you get this page and Jenkins is ready to get used.

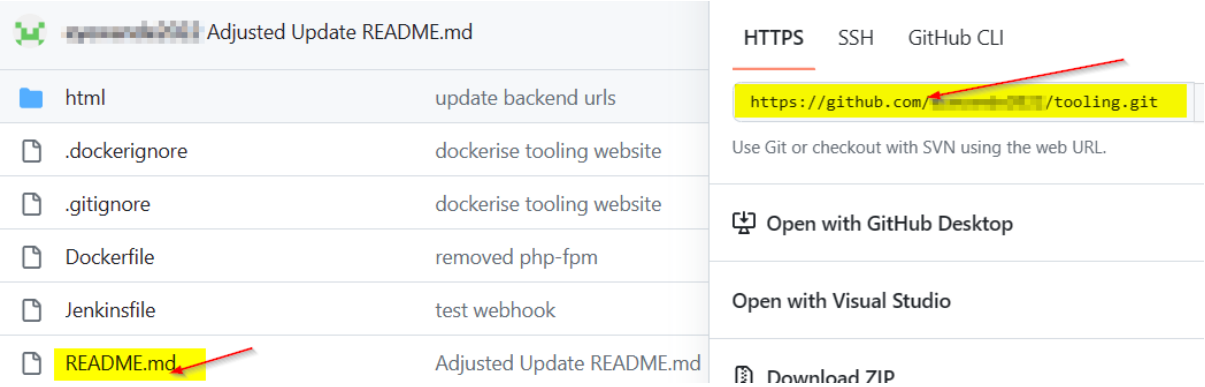




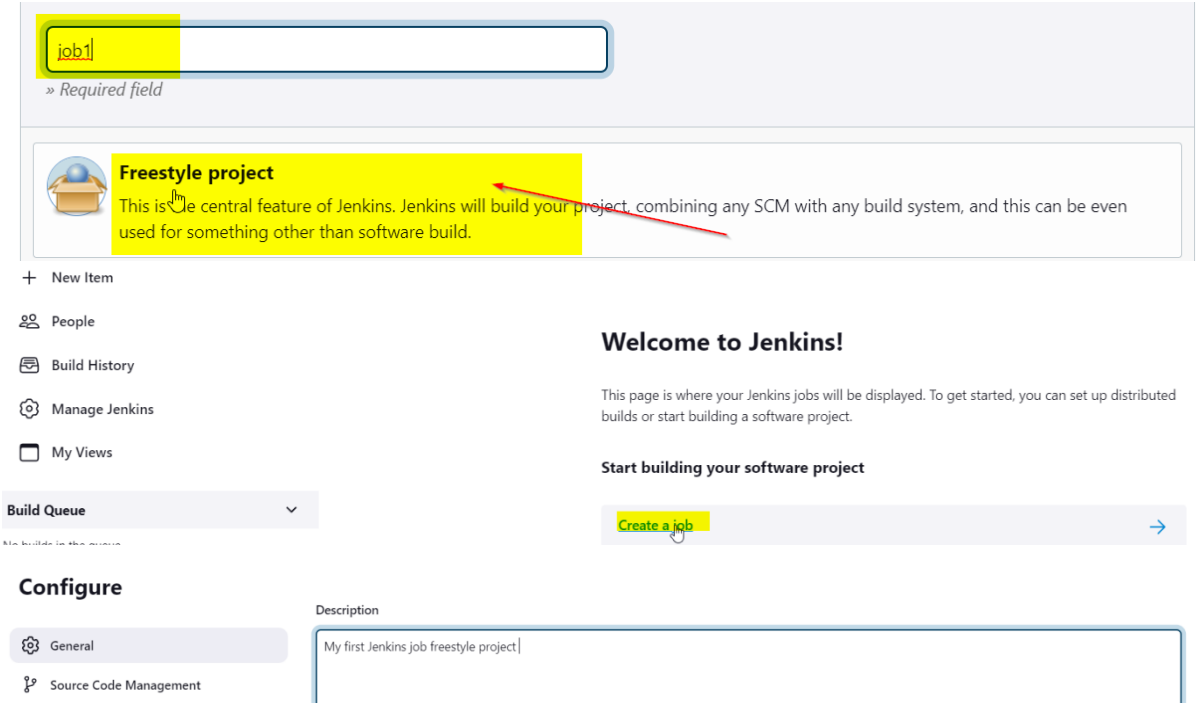
JENKINS CONFIGURATION WITH GITHUB USING WEBHOOKS

Next steps would be to configure Jenkins to retrieve source codes from GitHub using Webhooks.

Firstly we create our first job on Jenkins and connect to a repository chosen to perform this actions .We are using a README.md file to perform this as well as the URL copied

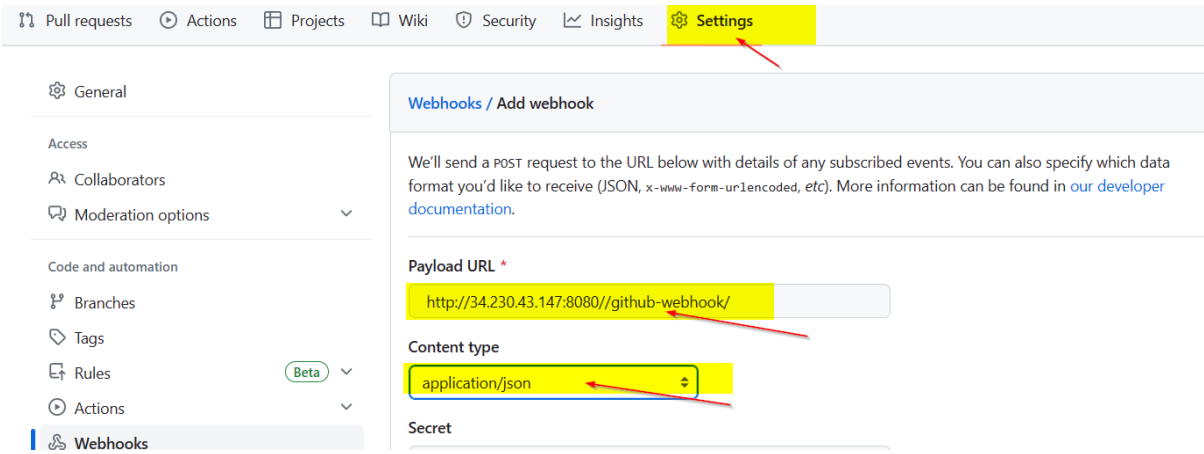


Creating a job on Jenkins. You create a name and select “free project” and click ok

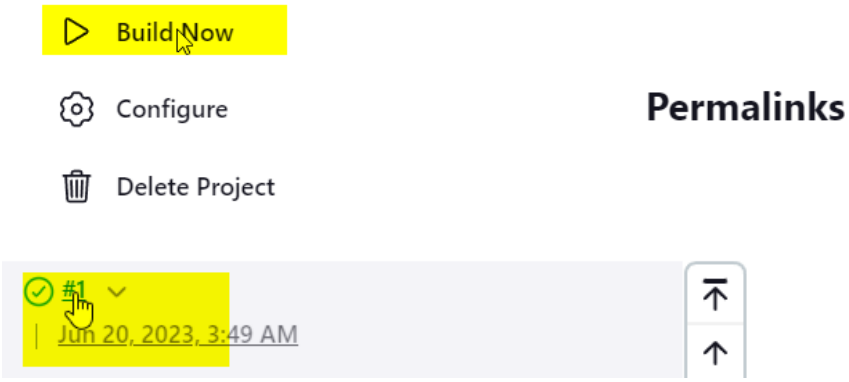


You describe the job. Navigate to Source code management and click git and fill in the repository URL . Click on “Apply” and “Save”

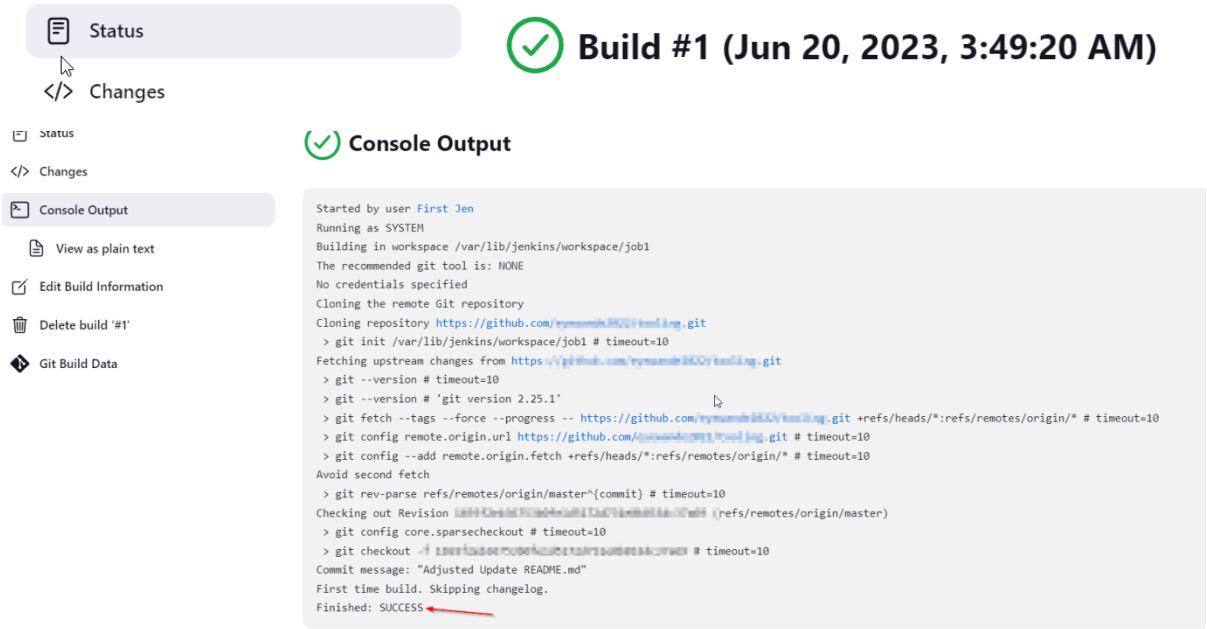
Navigate to your GitHub account and click on settings and add your webhook as shown below .



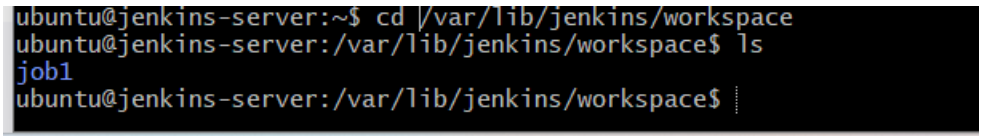
Once successfully added. We trigger the “Build Now” button and confirm if the build was successful



Navigate through the status and console output which gives you the details of the job and also states if it was a success as shown below

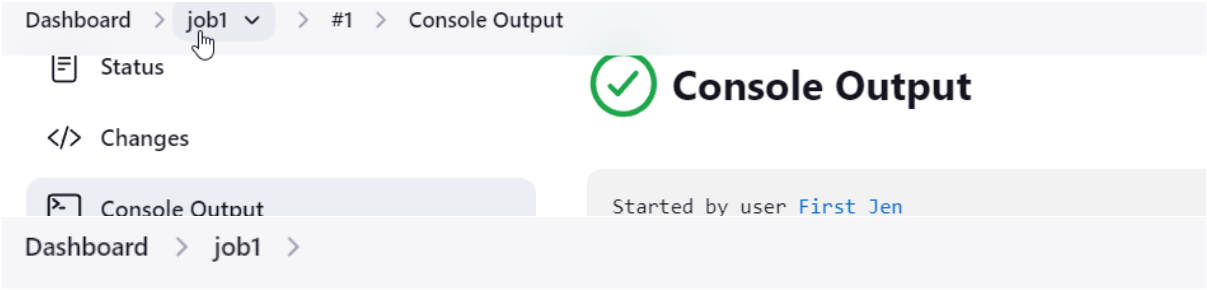


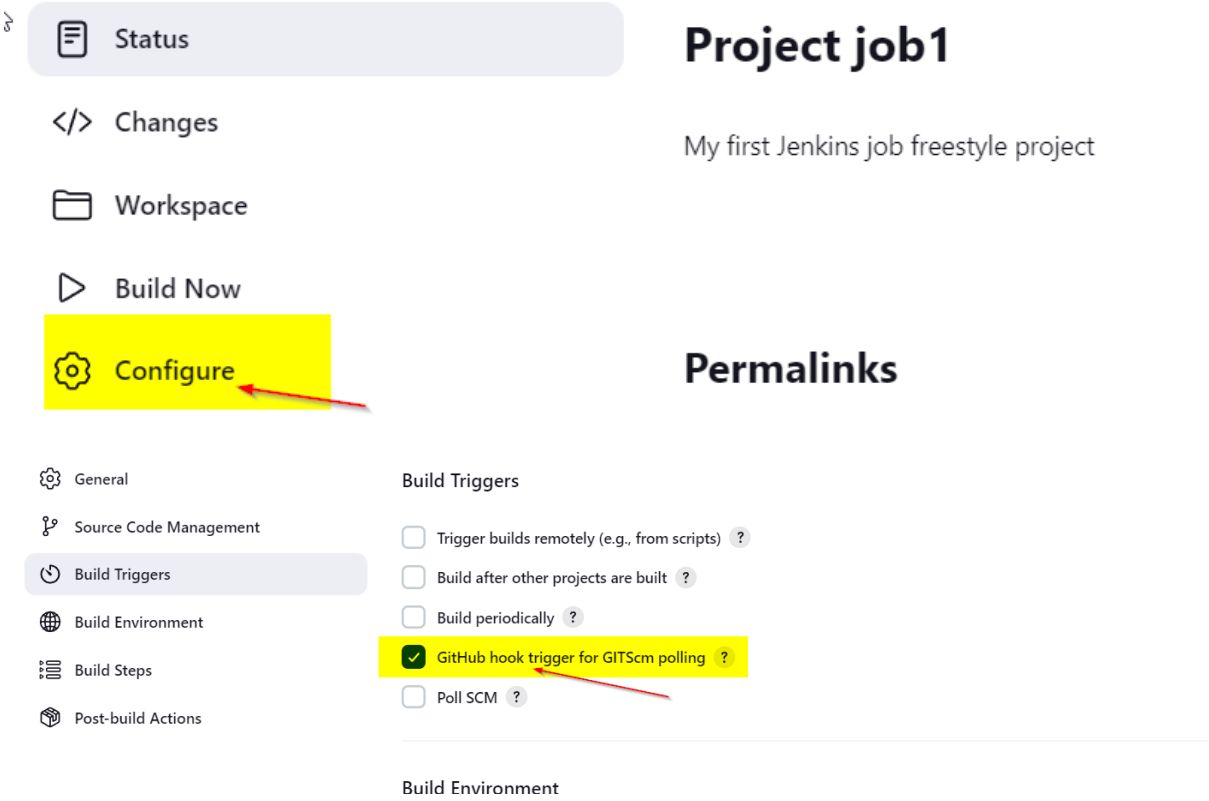
The console output illustrated how Jenkins created a workspace file with the help of the user admin called “job1”. If we navigate to this (/var/lib/jenkins/workspace) directory on our local machine and view the file content you would see the job1 created as shown below



Please note that this is a manual trigger hence it doesn't run anything. To begin running ,we need to add 2 configurations in our project

To get this done we need to navigate to the "configure" and navigate to "Build Trigger" and select the GitHub webhook as shown below .

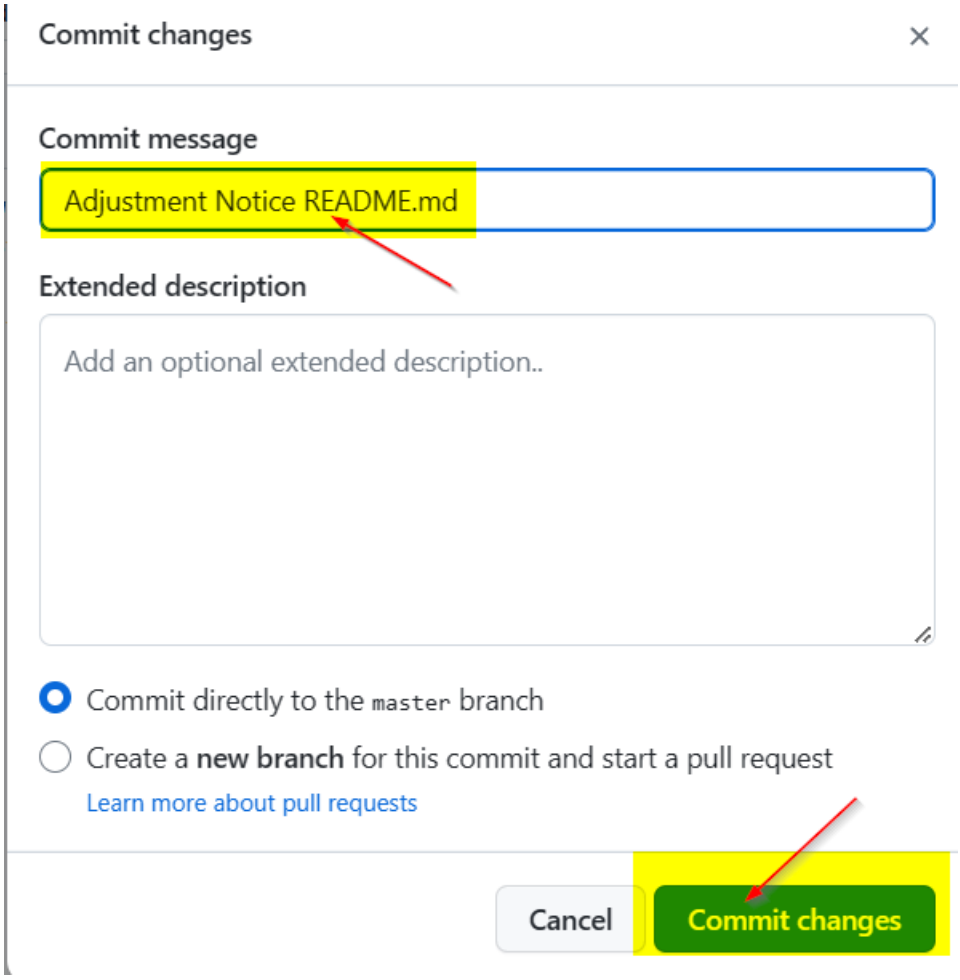




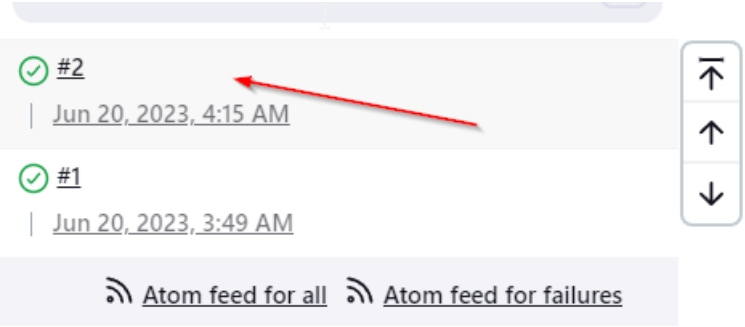
Apply and Save. Then proceed to try to automate using your Jenkins which is different from manually building it as we did previously .

Navigate to the README.md file on GitHub and do the editing and click on commit changes

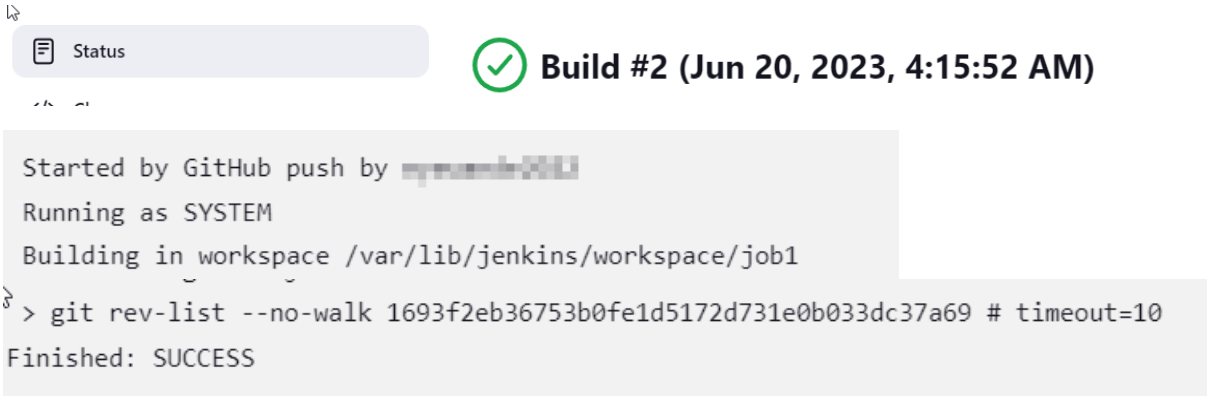
Ensure you also make changes on the commit message to always show clear difference between changes and commit on Jenkins.



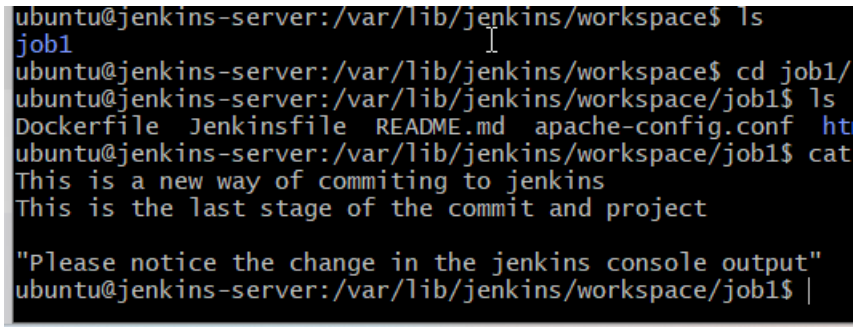
Once committed, Jenkins is triggered within seconds and hence we can see the second build



Click on the build2 to check the status and console output respectively.



On the terminal. Let us change directory to job1. It is evident that the README.md is located in the job1 folder and if we check its content, it is as expected and displayed below

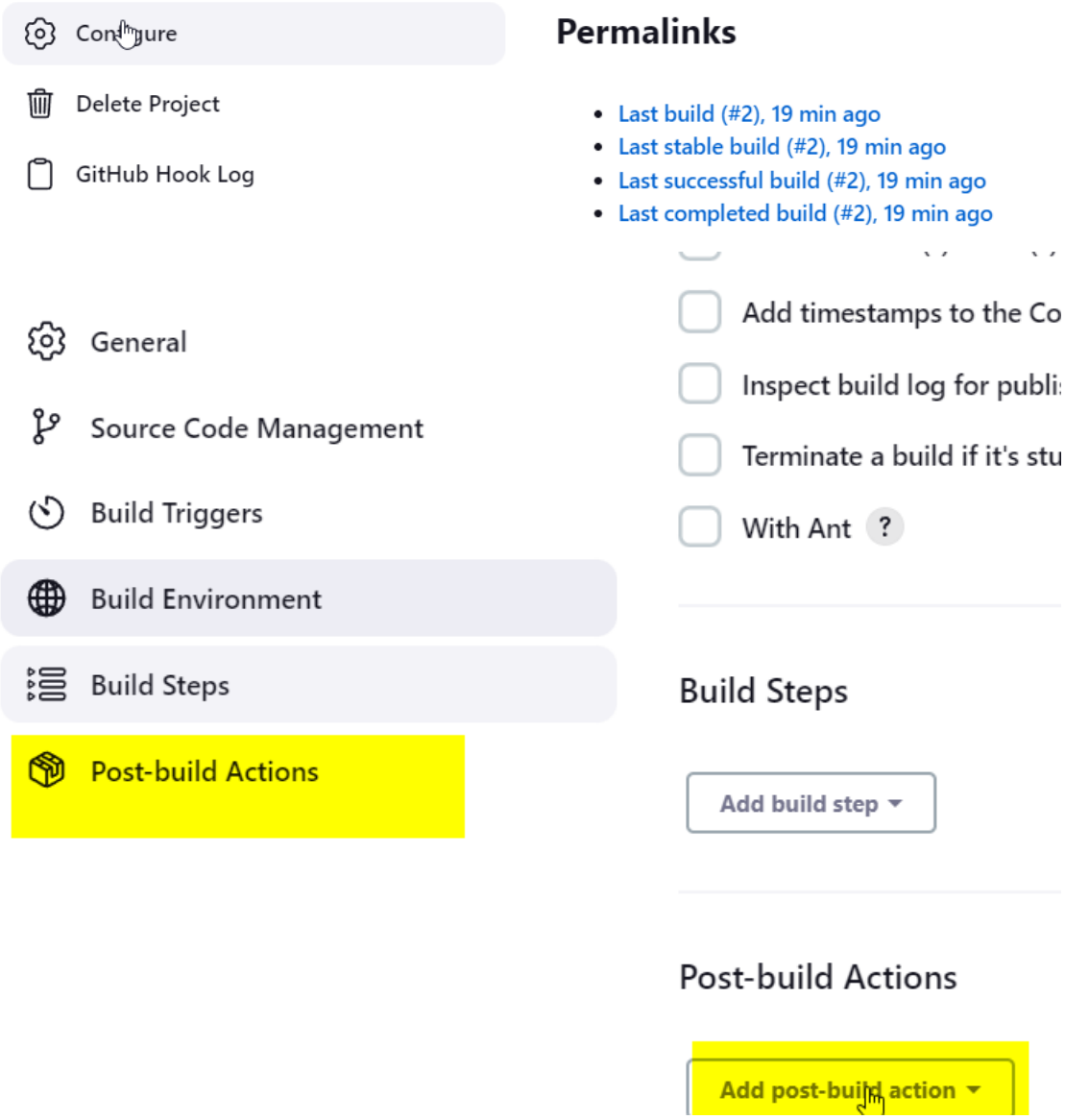


As a DevOps engineer, we need to be able to set up a Jenkins server that would help in monitoring the changes we have on version control management software like GitHub .This would make it easy for monitoring the source codes the developers load on the git repository

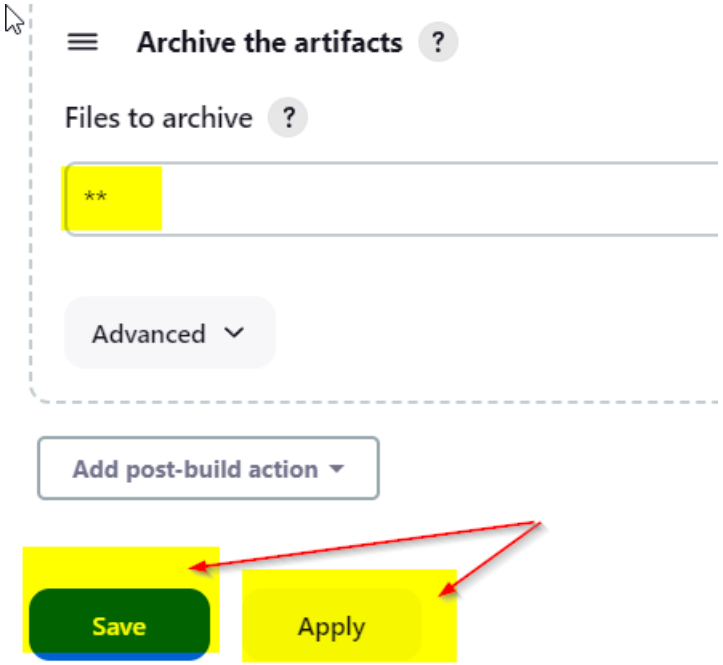
POST-BUILD ACTIONS CONFIGURATION

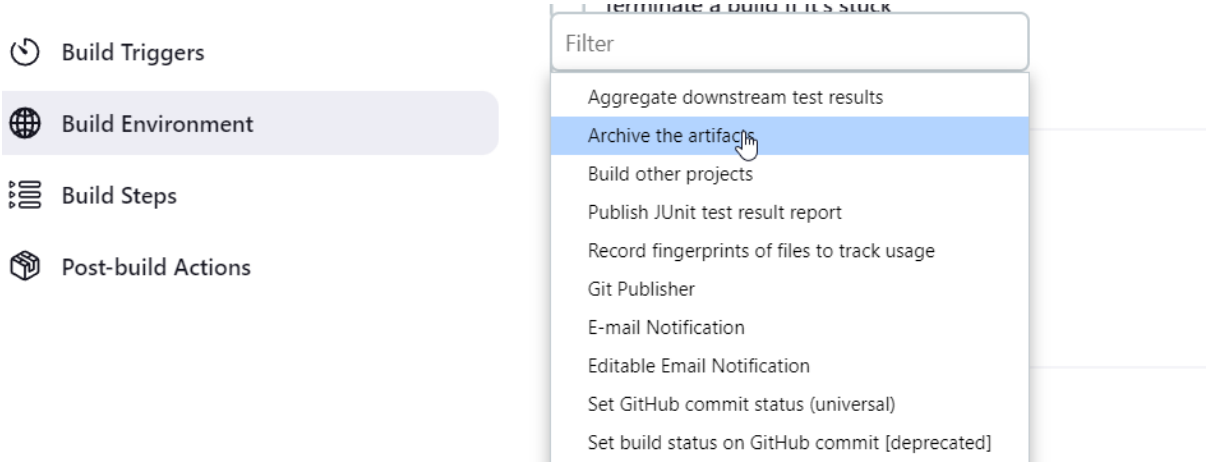
Whenever Jenkins triggers source codes, it packages the files and they are called artefacts.

Navigate back to “configure” section in Jenkins and click on it and locate the post-Build actions as seen below



Select “archive the artifacts”, Type ** to make sure Jenkins archives every build artifacts. Click on “ Apply” and “save”.





Next step is to find the Jenkins build on the terminal and change directory into job1 and check its content, locate builds folder and access it

```
ubuntu@jenkins-server:/var/lib/jenkins/workspace/job1$ cd /var/lib/jenkins/jobs/job1
ubuntu@jenkins-server:/var/lib/jenkins/jobs/job1$ ls
builds  config.xml  github-polling.log  nextBuildNumber
ubuntu@jenkins-server:/var/lib/jenkins/jobs/job1$ cd builds/
ubuntu@jenkins-server:/var/lib/jenkins/jobs/job1/builds$ ls
1  2  3  legacyIds  permalinks
ubuntu@jenkins-server:/var/lib/jenkins/jobs/job1/builds$ ls -la
total 24
drwxr-xr-x 5 jenkins jenkins 4096 Jun 20 04:41 .
drwxr-xr-x 3 jenkins jenkins 4096 Jun 20 04:41 ..
drwxr-xr-x 2 jenkins jenkins 4096 Jun 20 03:49 1
drwxr-xr-x 2 jenkins jenkins 4096 Jun 20 04:15 2
drwxr-xr-x 3 jenkins jenkins 4096 Jun 20 04:41 3
-rw-r--r-- 1 jenkins jenkins    0 Jun 20 03:37 legacyIds
-rw-r--r-- 1 jenkins jenkins  126 Jun 20 04:41 permalinks
ubuntu@jenkins-server:/var/lib/jenkins/jobs/job1/builds$
```

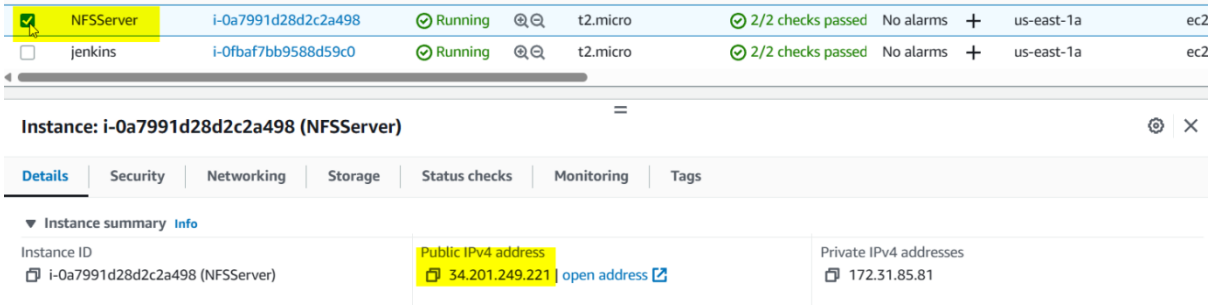
Navigating to “build 3” folders, we can see the archived files we shown below

```
1  2  3  legacyIds  permalinks
ubuntu@jenkins-server:/var/lib/jenkins/jobs/job1/builds$ cd 3/
ubuntu@jenkins-server:/var/lib/jenkins/jobs/job1/builds/3$ ls
archive  build.xml  changelog.xml  log  polling.log
ubuntu@jenkins-server:/var/lib/jenkins/jobs/job1/builds/3$ cd archive
ubuntu@jenkins-server:/var/lib/jenkins/jobs/job1/builds/3/archive$ ls
Dockerfile  Jenkinsfile  README.md  apache-config.conf  html  start-apache  tooling-db.sql
ubuntu@jenkins-server:/var/lib/jenkins/jobs/job1/builds/3/archive$
```

CONFIGURE JENKINS TO COPY FILES TO THE NFS SERVER VIA SSH

We have the artifacts saved in our Jenkins server ,we have to copy them into our NFS server and have to save it to our /mnt/apps directory

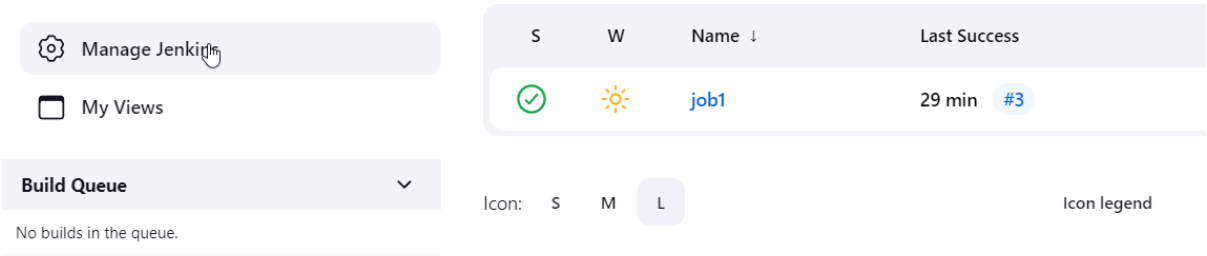
Navigate to the nfs server created and



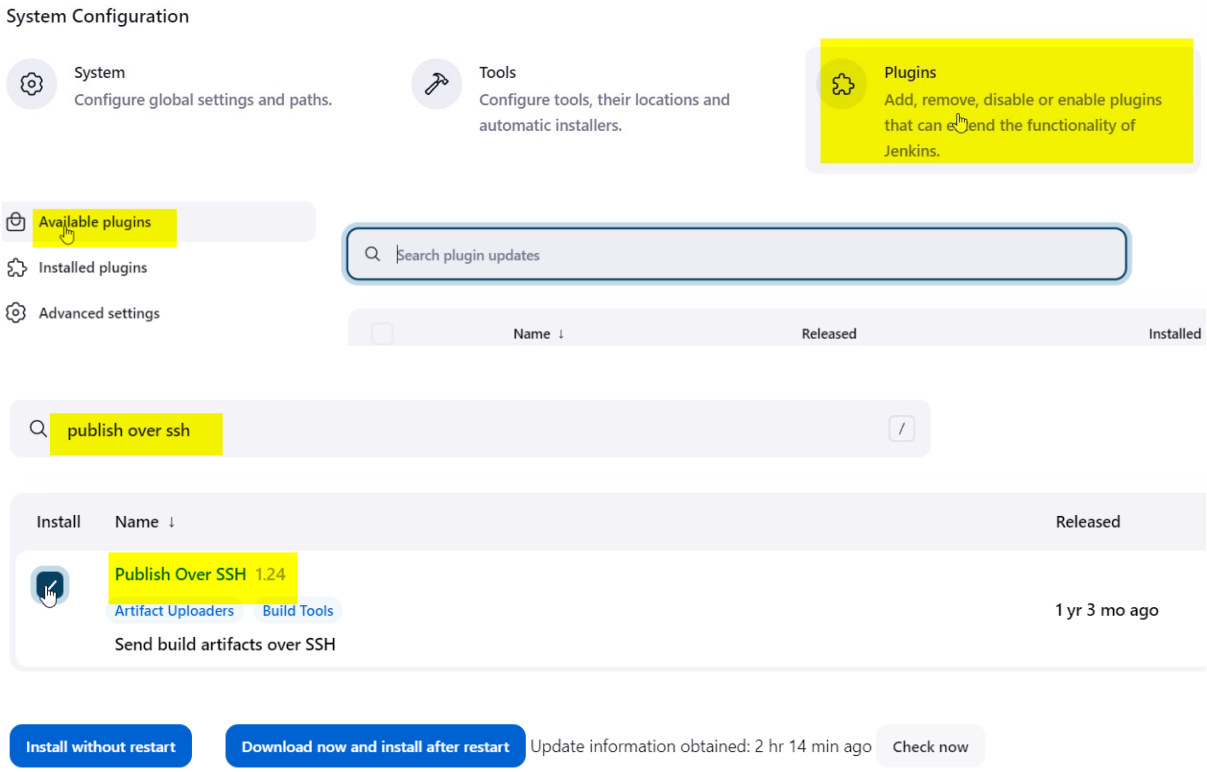
Check the content of /mnt/apps directory.

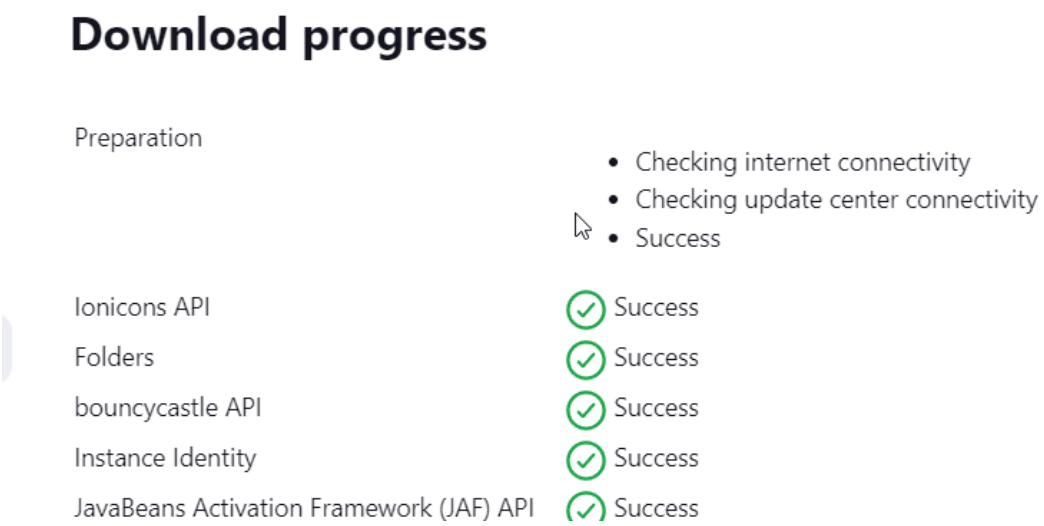
```
[ec2-user@nfs-server ~]$ ls /mnt
apps  logs  opt
[ec2-user@nfs-server ~]$ ls /mnt/apps
html
[ec2-user@nfs-server ~]$ cd /mnt/apps
```

We should install the Publish Over SSH plugin on Jenkins. Navigate to “Manage Jenkins”.

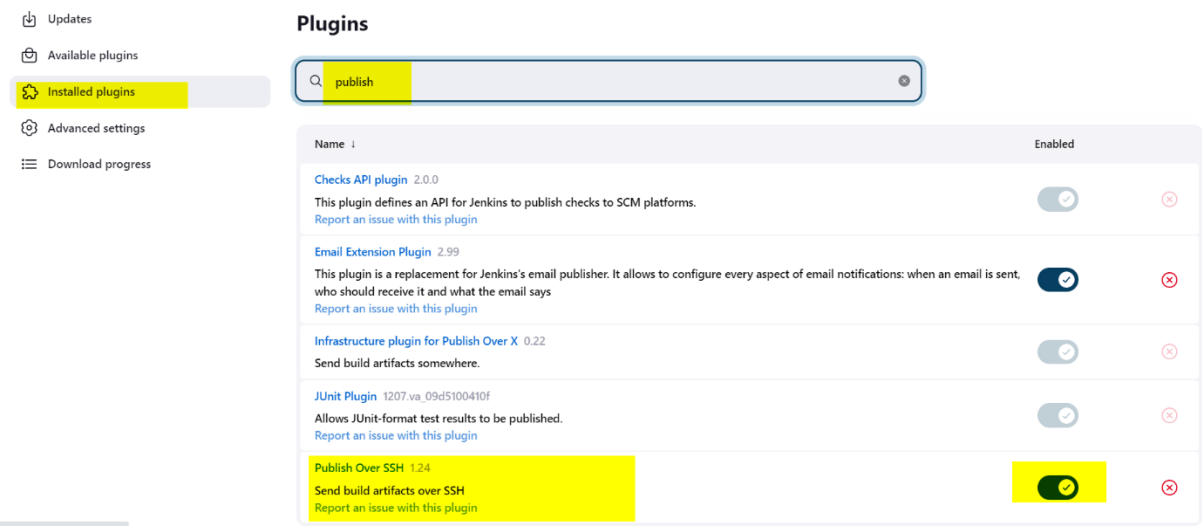


Select Plugins, Click on available plugins and type on the search box “Publish Over SSH” and make sure you select and install without restart

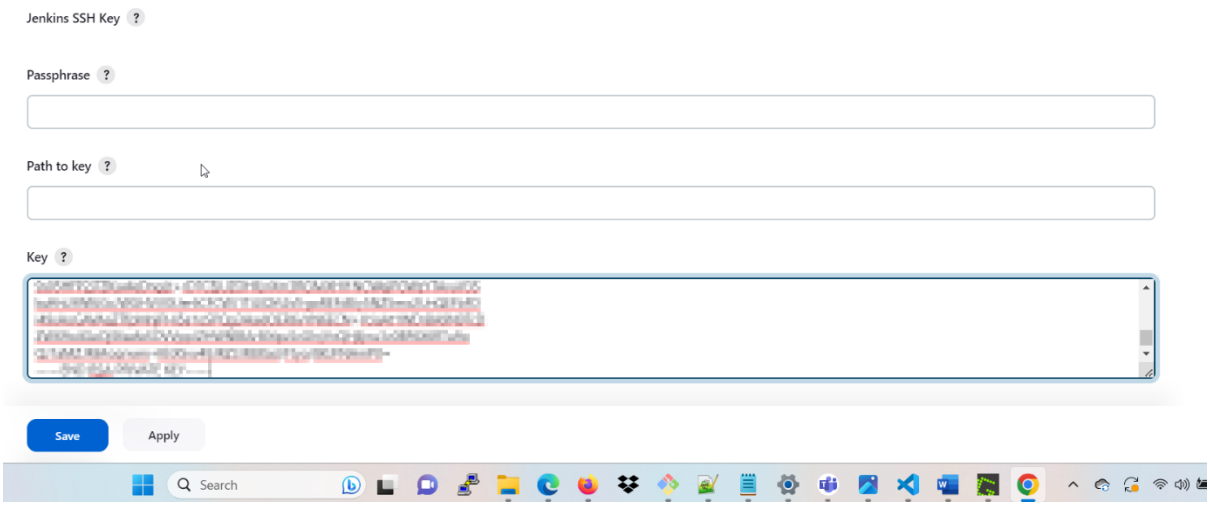




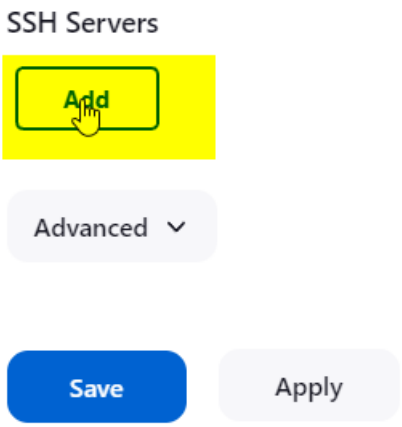
Successfully installed. Check Installed plugin and it can be seen it is correctly installed.

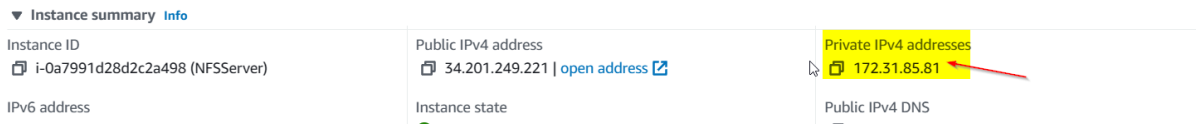


Navigate back to Manage Jenkins and click on Systems and navigate to Publish over SSH. Provide your private key

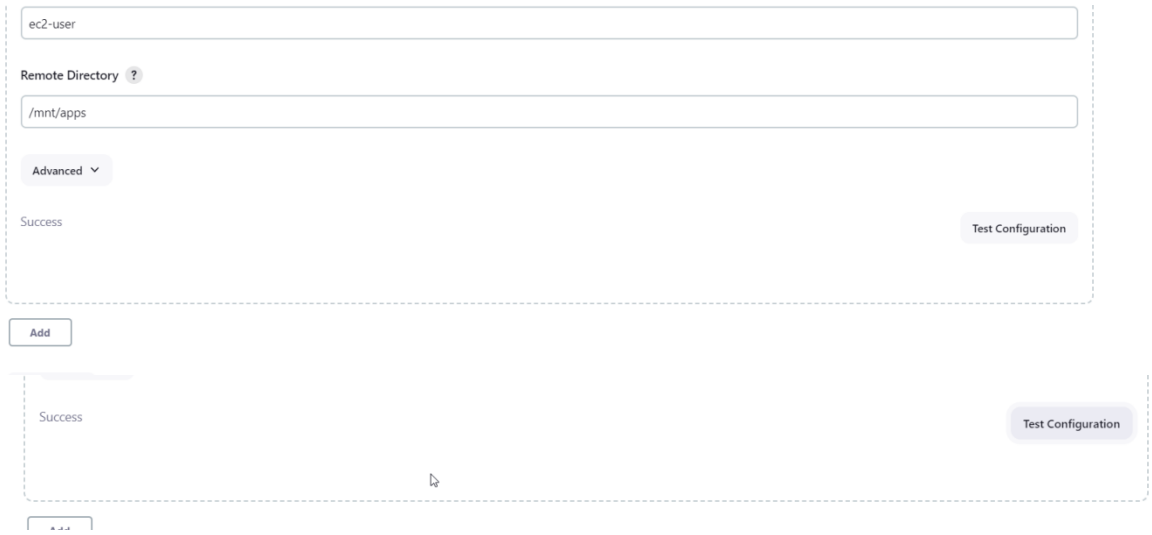


Click on Add SSH and fill in the server name, private ip address ,terminal username and remote directory





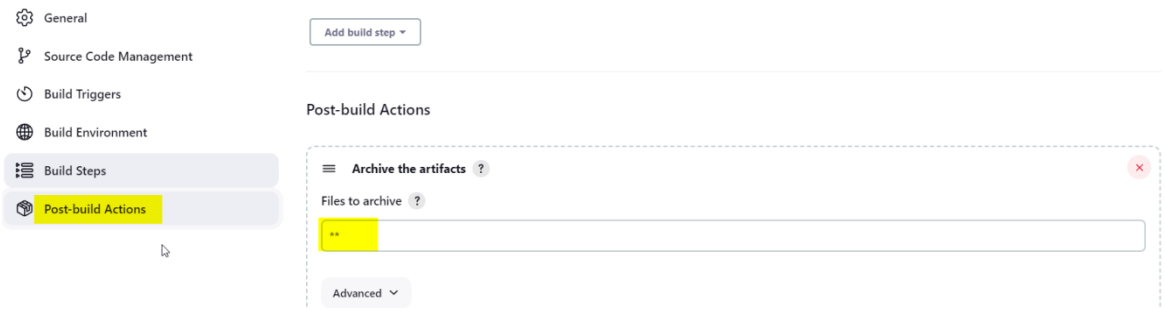
Then we proceed to test configuration and it shows success



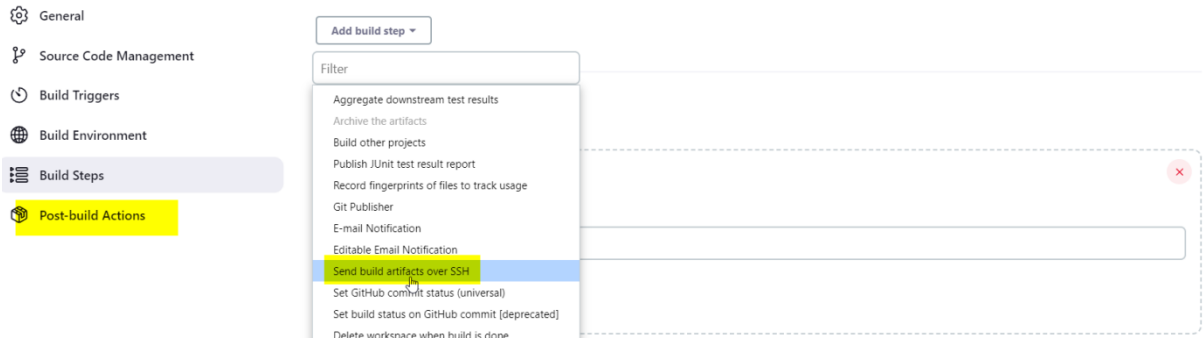
Click on “Apply” and ”Save ” and ensure that port :22 is open in the nfs server .

We are to configure it to send all files produced by the build into our previously defined remote directory .

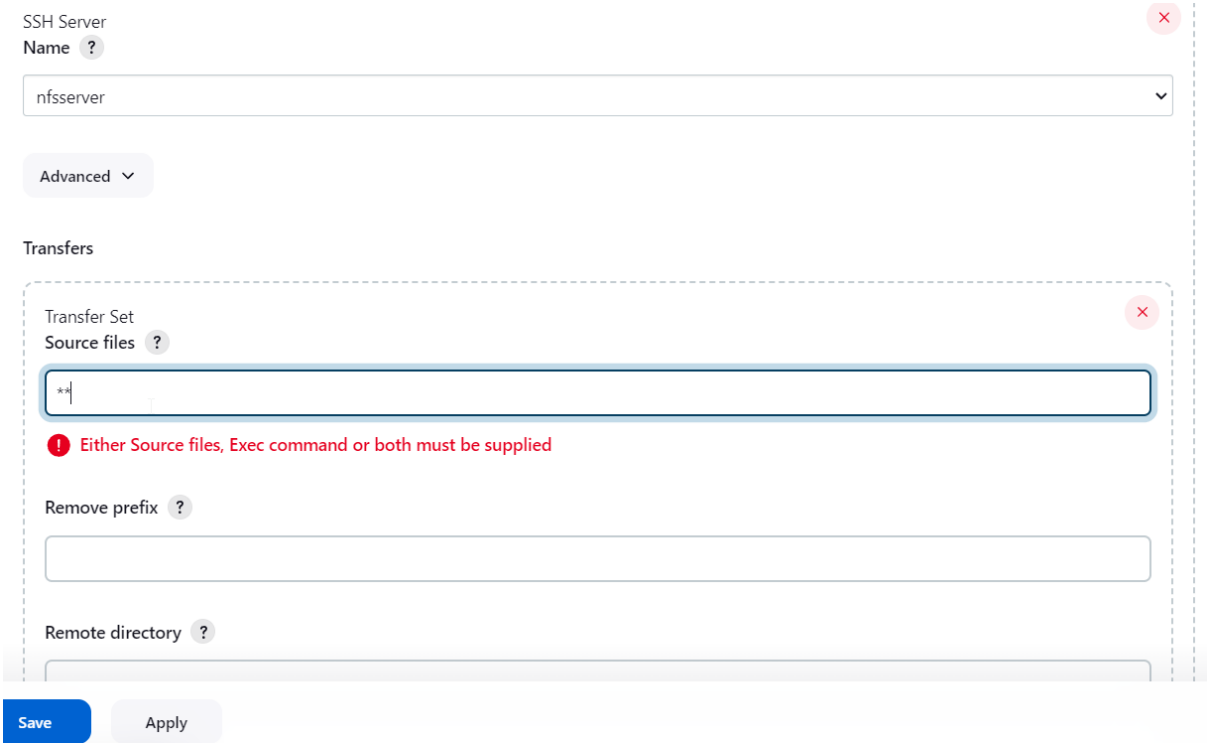
Navigate back to the Jenkins dashboard, go to configure and navigate to the post-build actions



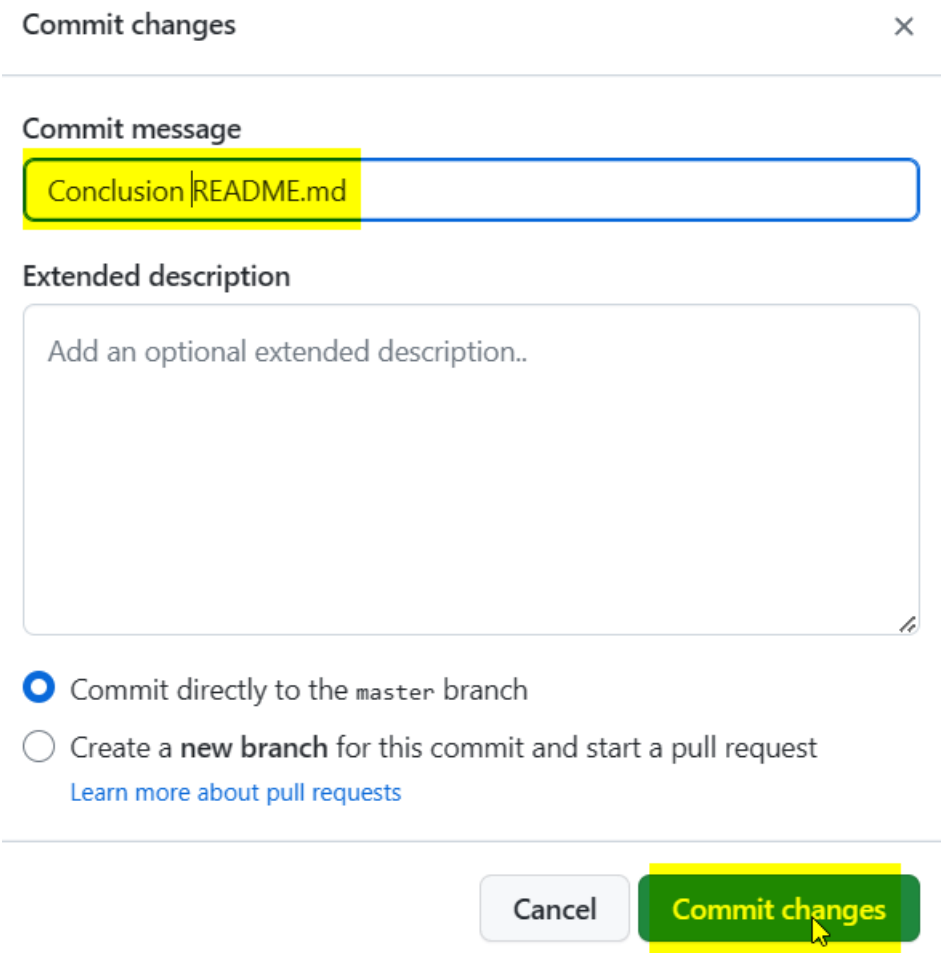
Add another post build action. “send build actions over SSH

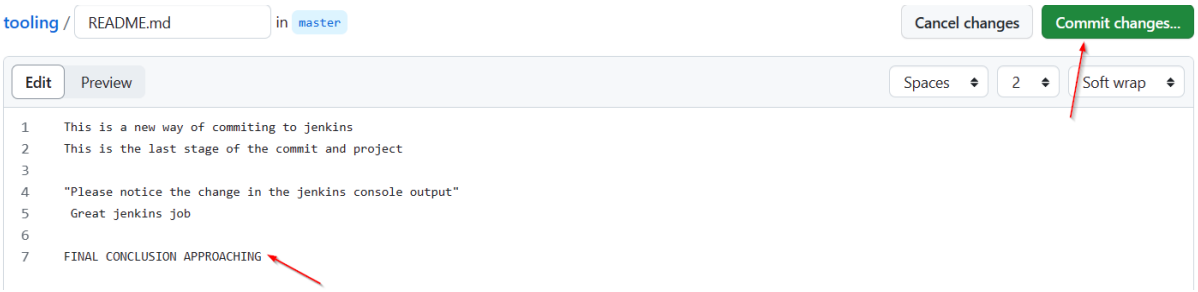


Type in ** in the source file edit box and “Apply” and “Save “

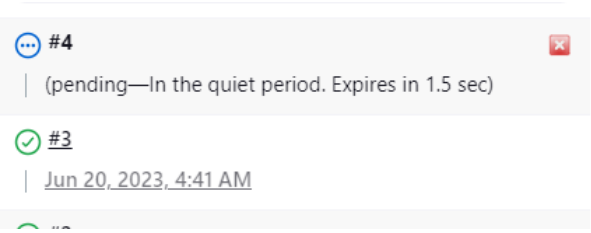


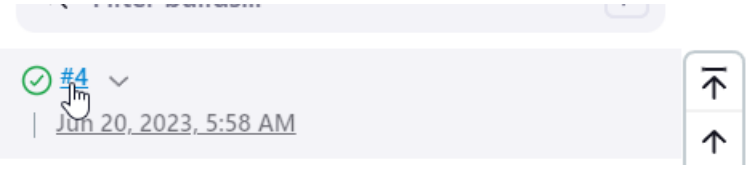
Now Jenkins is listening now. We navigate back to git hub and make a change.



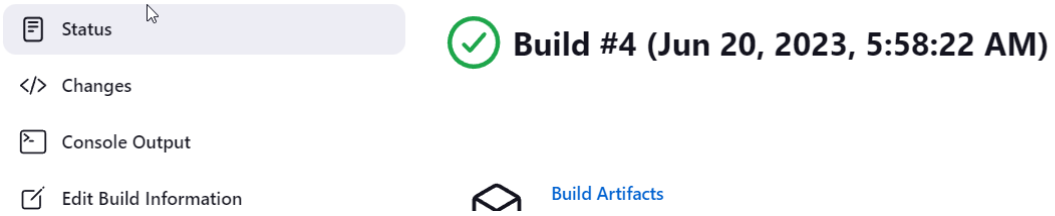


Once changes is committed and we check on Jenkins its pending and then successful .





Clicking the link, we can check the status as successful.



Navigate back to the terminal and check the files in /mnt/apps as shown below, we can see all the files are there .

```
[ec2-user@ip-172-31-85-81 apps]$ sudo hostname nfs-server
[ec2-user@ip-172-31-85-81 apps]$ bash
[ec2-user@nfs-server apps]$ ls
html  mnt
[ec2-user@nfs-server apps]$ ls -la /mnt/apps
total 36
drwxr-xr-x. 4 ec2-user ec2-user  172 Jun 20 05:58 .
drwxr-xr-x. 5 root     root       41 Jun 12 16:56 ..
-rw-r--r--. 1 ec2-user ec2-user  332 Jun 20 05:58 apache-config.conf
-rw-r--r--. 1 ec2-user ec2-user  313 Jun 20 05:58 Dockerfile
-rw-r--r--. 1 ec2-user ec2-user   47 Jun 20 05:58 .dockerignore
drwxr-xr-x. 3 ec2-user ec2-user 4096 Jun 15 21:49 html
-rw-r--r--. 1 ec2-user ec2-user 4202 Jun 20 05:58 Jenkinsfile
drwxr-xr-x. 3 ec2-user ec2-user   18 Jun 19 20:25 mnt
-rw-r--r--. 1 ec2-user ec2-user  201 Jun 20 05:58 README.md
-rw-r--r--. 1 ec2-user ec2-user  163 Jun 20 05:58 start-apache
-rw-r--r--. 1 ec2-user ec2-user 1674 Jun 20 05:58 tooling-db.sql
[ec2-user@nfs-server apps]$
```

We have been able to move the files from Jenkins artifacts to the NFS server.

```
-rw-r--r--. 1 ec2-user ec2-user 1674 Jun 20 05:58 tooling-db.sql
[ec2-user@nfs-server apps]$ ls
apache-config.conf  Dockerfile  html  Jenkinsfile  mnt  README.md  start-apache  tooling-db.sql
[ec2-user@nfs-server apps]$
```

This is exactly what we want to achieve.

Checking the README.md file we can see the files on there.

```
[ec2-user@nfs-server apps]$ cat README.md
This is a new way of commiting to jenkins
This is the last stage of the commit and project

"Please notice the change in the jenkins console output"
 Great jenkins job

FINAL CONCLUSION APPROACHING
[ec2-user@nfs-server apps]$
```

We have been able to successfully implement the Jenkins server.