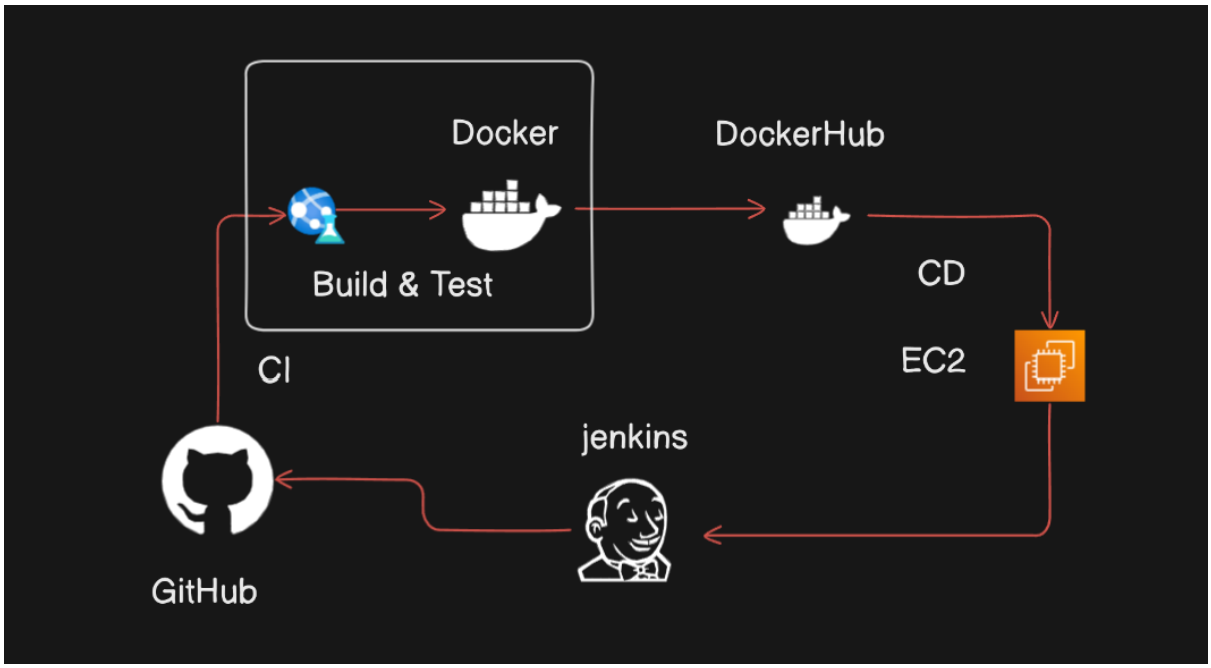
Jenkins Declarative Pipeline with Docker

django-notes-app-CICD Project

Description:

This project involves setting up a continuous integration and continuous deployment (CI/CD) pipeline using GitHub for a Django application. The application will be containerized using docker and deployed to an AWS EC2 instance



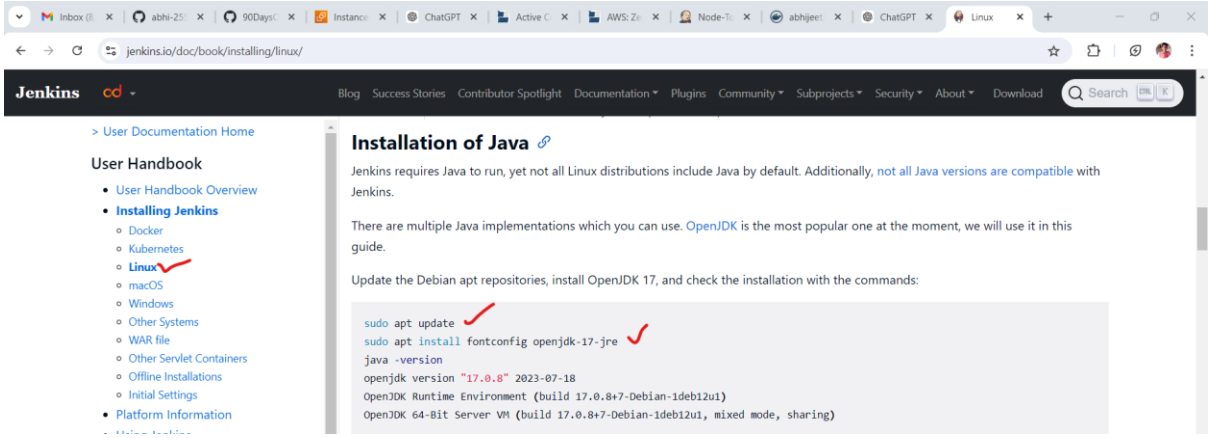
- **Code will be on GitHub repository**
- **This code will be tested and for that there should be a virtual environment for which docker is required which contains os, code, tests etc.**
- **After testing we have to push the container to dockerHub registry so that it should run on any machine by pulling the image on ec2 or k8's cluster**
- **To automate the entire process we use Jenkins as CI-CD automation tool**
- **GitHub url for ref: https://github.com/abhi-255/django-notes-app.git**

Steps:

1) **Launch ec2 instance.**
   This is our Jenkins master so we need to install Jenkins on this server

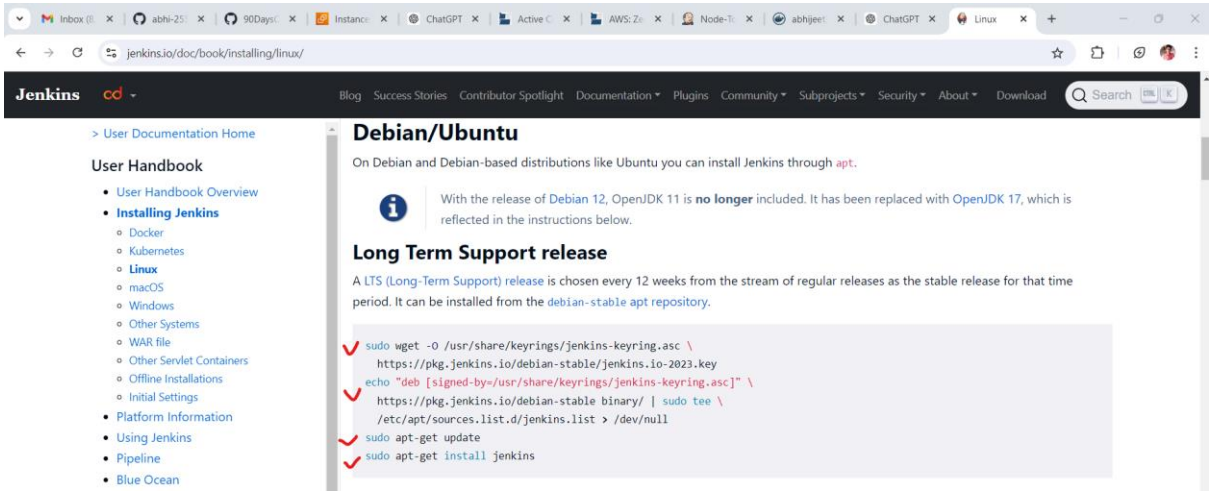
2) **Jenkins Installation:**

First need to Install java
Search for install Jenkins on ubuntu on google
Follow the commands marked in the image to install java:



sudo apt update
sudo apt install fontconfig openjdk-17-jre
java -version

3) Install Jenkins follow the commands mentioned in the image: Use Long Term Support release



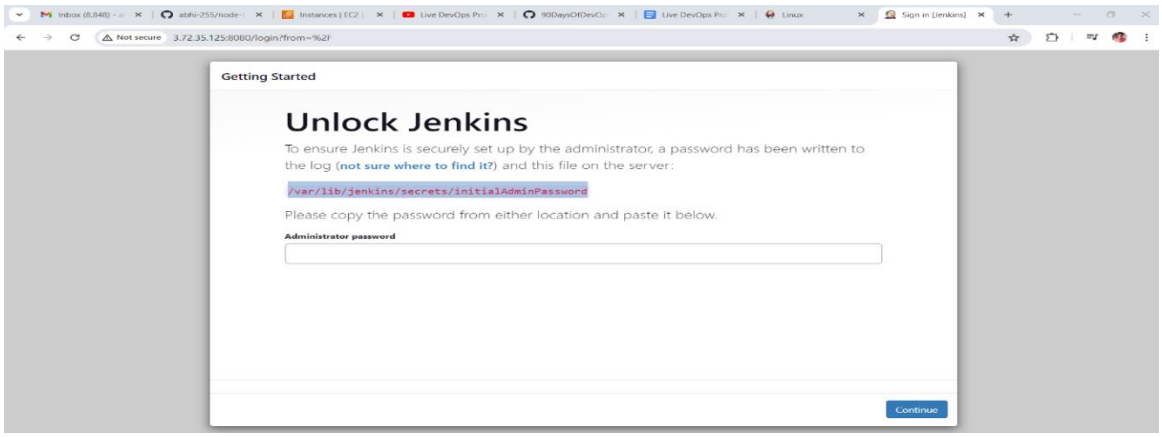
Run the following commands:

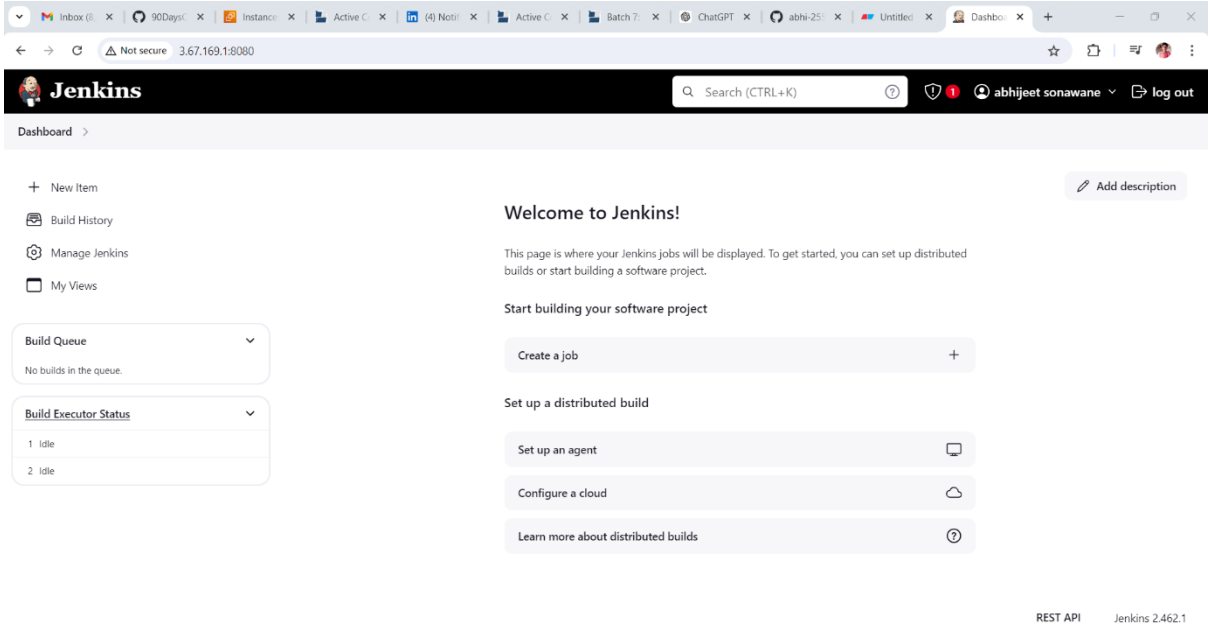
a) sudo systemctl status Jenkins

4) Jenkins setup:

a) To access Jenkins dashboard we need to add port 8080 into security group.

b) Serch the ip address of ec2:8080 and login to Jenkins dashboard
Following command will give you the paasword, enter it and login to dashboard
sudo cat /var/lib/Jenkins/initialAdminpassword





Jenkins master setup completed.

Now we have create a agent or worker node where we run our pipeline.

5) **Agent Setup:**
   Launch ec2 instance.
   This is our worker node

6) **Now we have to create connection between master and worker node.**
   Master is accessing node so private key should be on master and public key should go to worker node

   On Master:
   - **ssh-keygen**
   - **cd .ssh/**
   - **ls**
   - **cat id_ed25519.pub >>This public key we have to add in worker node**

```
ubuntu@ip-172-31-18-175:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ed25519
Your public key has been saved in /home/ubuntu/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:bmJGZPXHgY8CdzwXdYhey7KOolS2l9cjX0JnND2Vu5w ubuntu@ip-172-31-18-175
The key's randomart image is:
+--[ED25519 256]--+
|        .. .o+..+|
|      ....=.o.ooo|
|      oo ..*oo ++|
|     o  . ..+ ooo|
|      . S.   oo =|
|     . + . ..o E |
|      = + ooo + .|
|     + o.....o + |
|       .. .    . |
+----[SHA256]-----+
ubuntu@ip-172-31-18-175:~$ cd .ssh/
ubuntu@ip-172-31-18-175:~/.ssh$ ls
authorized_keys  id_ed25519  id_ed25519.pub
ubuntu@ip-172-31-18-175:~/.ssh$ cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPW5jdOO2mOnYEkwhDl13r6CAp/7yijLLKk2utE2zj/8 ubuntu@ip-172-31-18-175
ubuntu@ip-172-31-18-175:~/.ssh$
```

Now go to Worker Node:

Follow the commands:

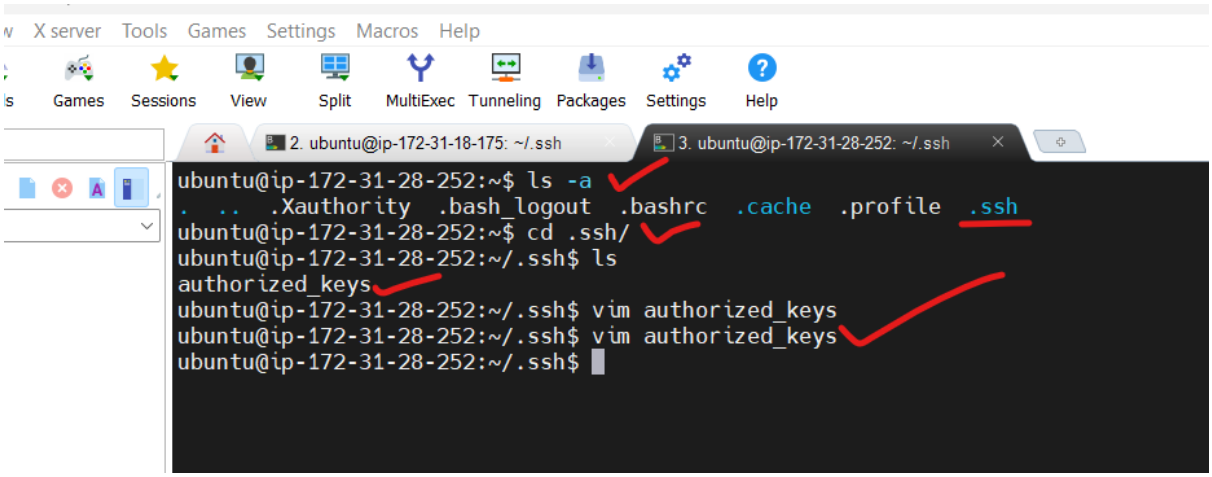
cd

ls -a

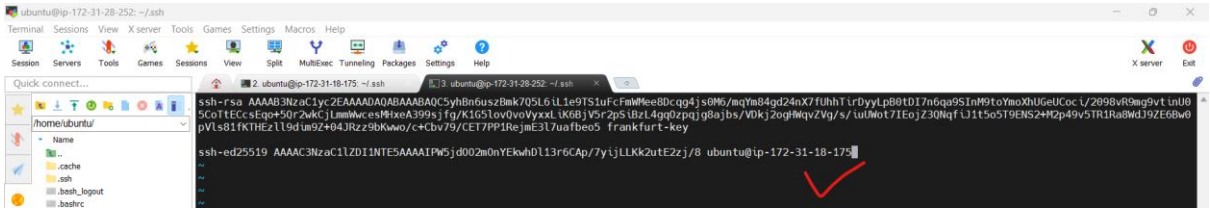
cd .ssh

ls

vim authorized_keys



Add the public key of master here:

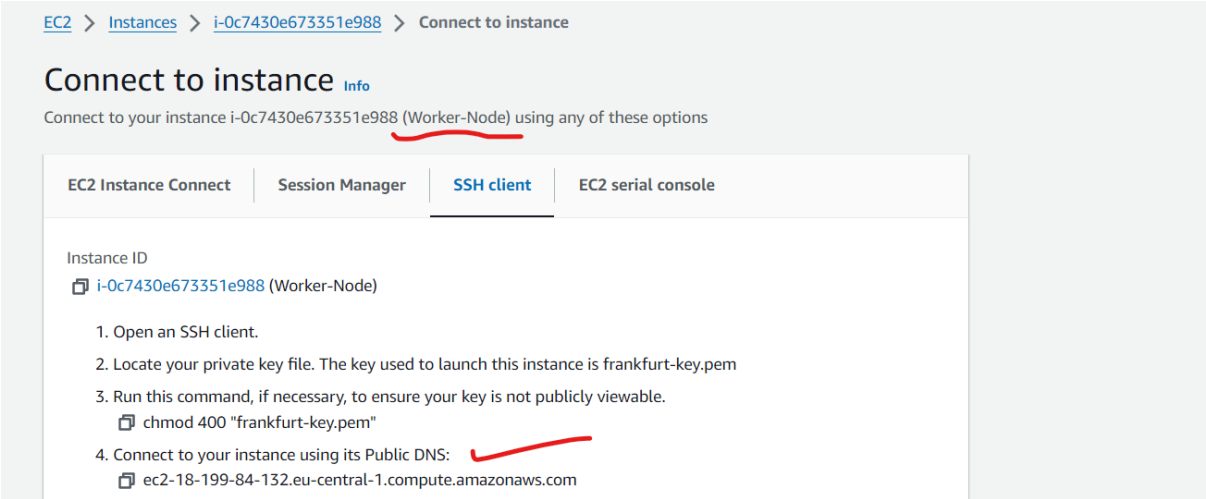


Check master can access node or not using:

ssh -i id_ed25519 ubuntu@ec2-18-199-84-132.eu-central-1.compute.amazonaws.com

id_ed25519 this is master's private key

ec2-18-199-84-132.eu-central-1.compute.amazonaws.com this is worker node DNS



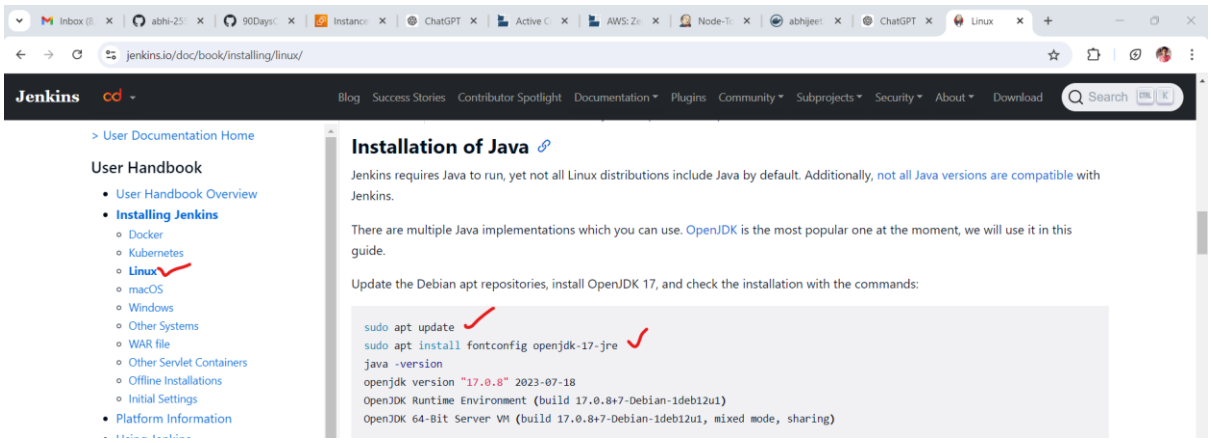
```
ubuntu@ip-172-31-18-175:~/.ssh$ cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPW5jd002m0nYEkwhDl13r6CAp/7yijLLKk2utE2zj/8 ubuntu@ip-172-31-18-175
ubuntu@ip-172-31-18-175:~/.ssh$
ubuntu@ip-172-31-18-175:~/.ssh$
ubuntu@ip-172-31-18-175:~/.ssh$
ubuntu@ip-172-31-18-175:~/.ssh$ ssh -i id_ed25519 ubuntu@ec2-18-199-84-132.eu-central-1.compute.amazonaws.com
The authenticity of host 'ec2-18-199-84-132.eu-central-1.compute.amazonaws.com (172.31.28.252)' can't be established.
ED25519 key fingerprint is SHA256:00Y58iFbkThwt2FuwG2oI11V5klFzKbgukm2LMruHXY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-199-84-132.eu-central-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1009-aws x86_64)

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

 System information as of Wed Aug  7 18:38:41 UTC 2024

  System load:  0.0               Processes:             109
  Usage of /:   22.9% of 6.71GB   Users logged in:       1
  Memory usage: 21%               IPv4 address for enX0: 172.31.28.252
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Aug  7 18:30:42 2024 from 103.171.186.75
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-28-252:~$
ubuntu@ip-172-31-28-252:~$
```

Node DNS

exit

7) On Worker node:

We have to Install java
Search for install Jenkins on ubuntu on google
Follow the commands marked in the image to install java:

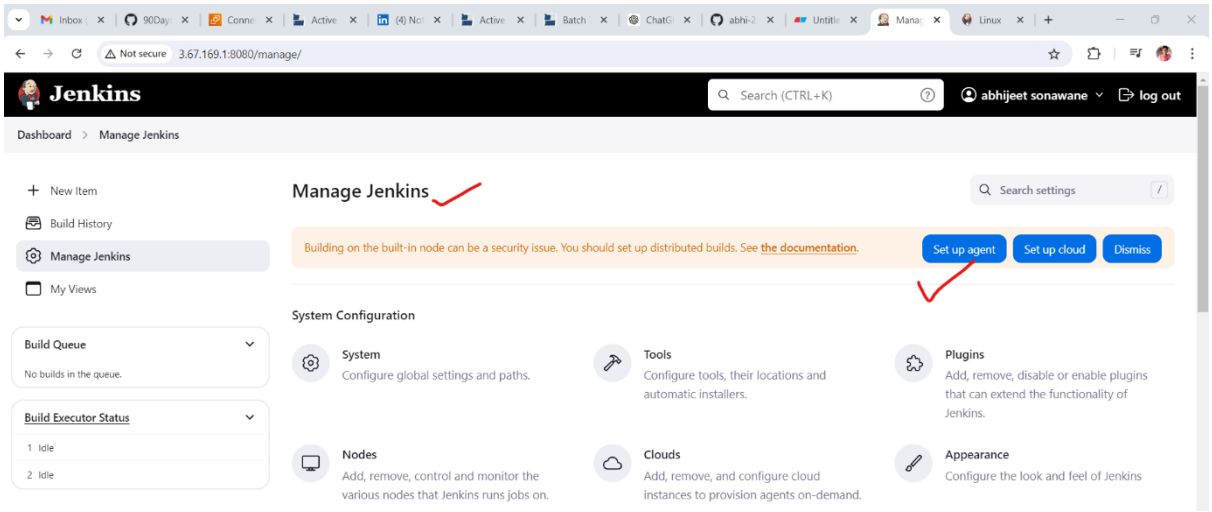


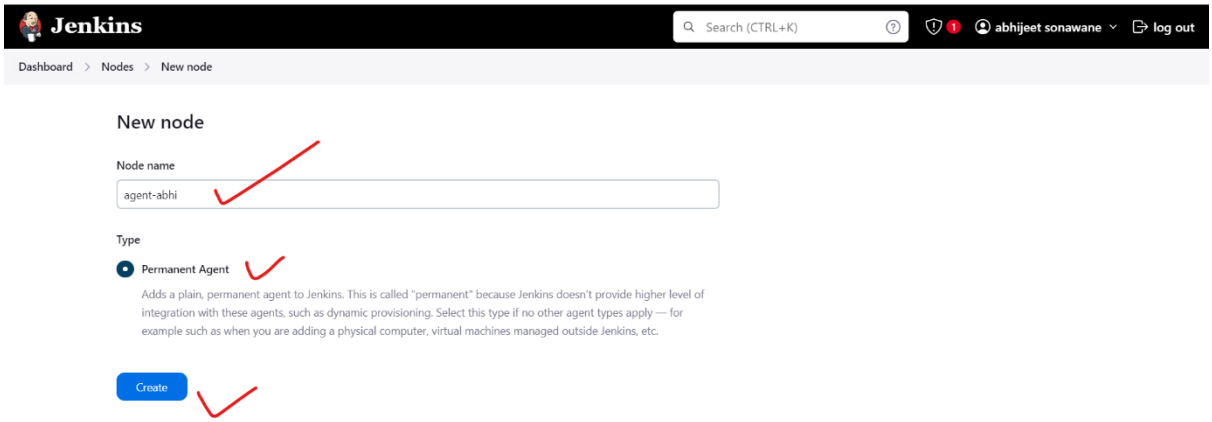
sudo apt update
sudo apt install fontconfig openjdk-17-jre
java -version

Till here server to server connection is done.
Now we have do server to Jenkins connection. For that the private key of master need to add in Jenkins
To achieve this we have to create agent on Jenkins dashboard
Go to Jenkins dashboard

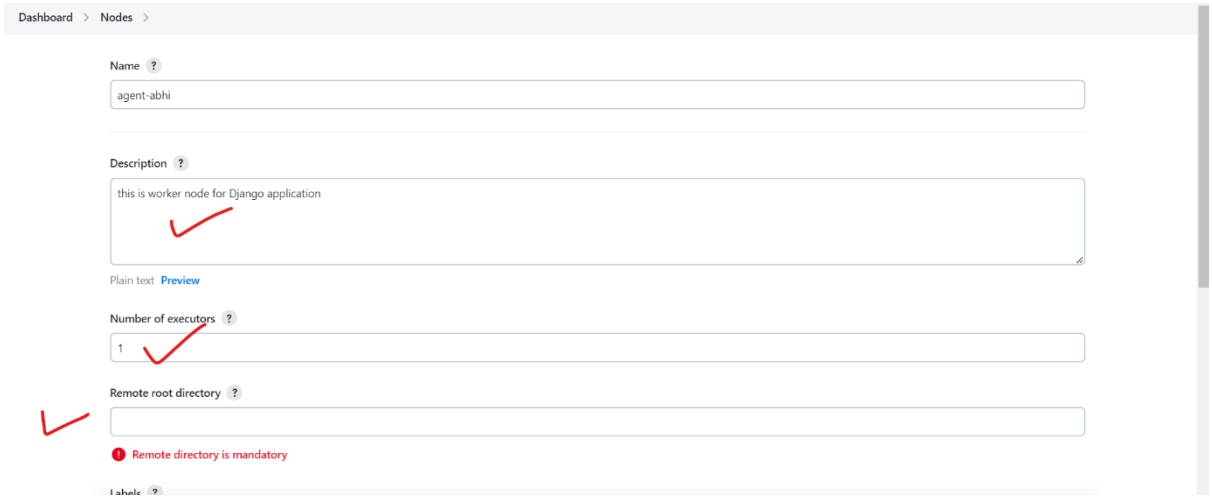
Follo the path:

Manage Jenkins >> set up agent





Provide name and select type as permanent then create, you will redirect to following:



For remote root directory path:

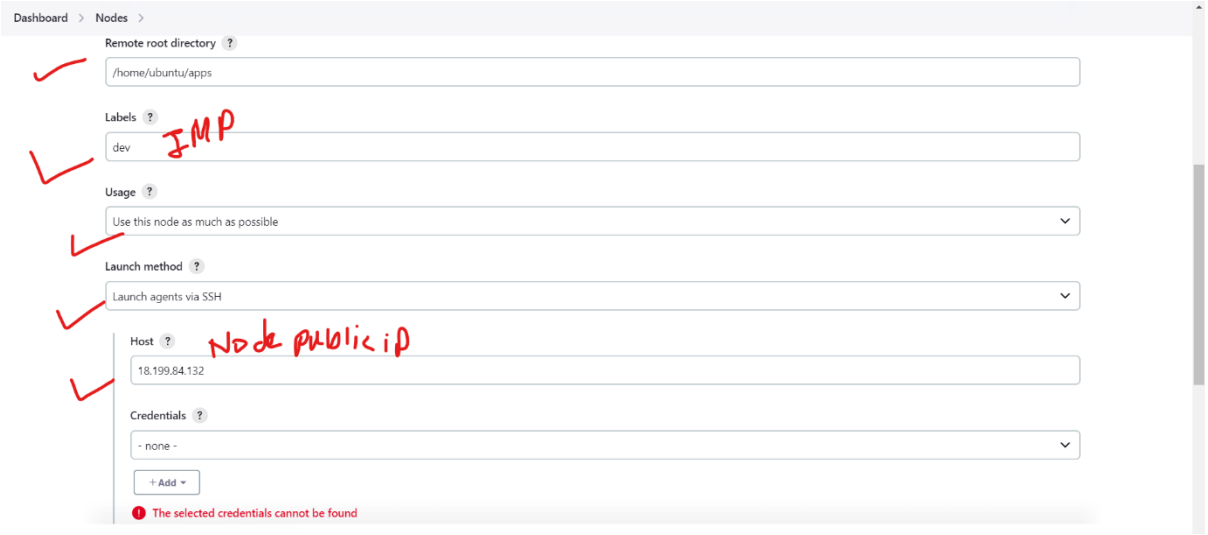
We need to create a directory on worker node.

mkdir apps

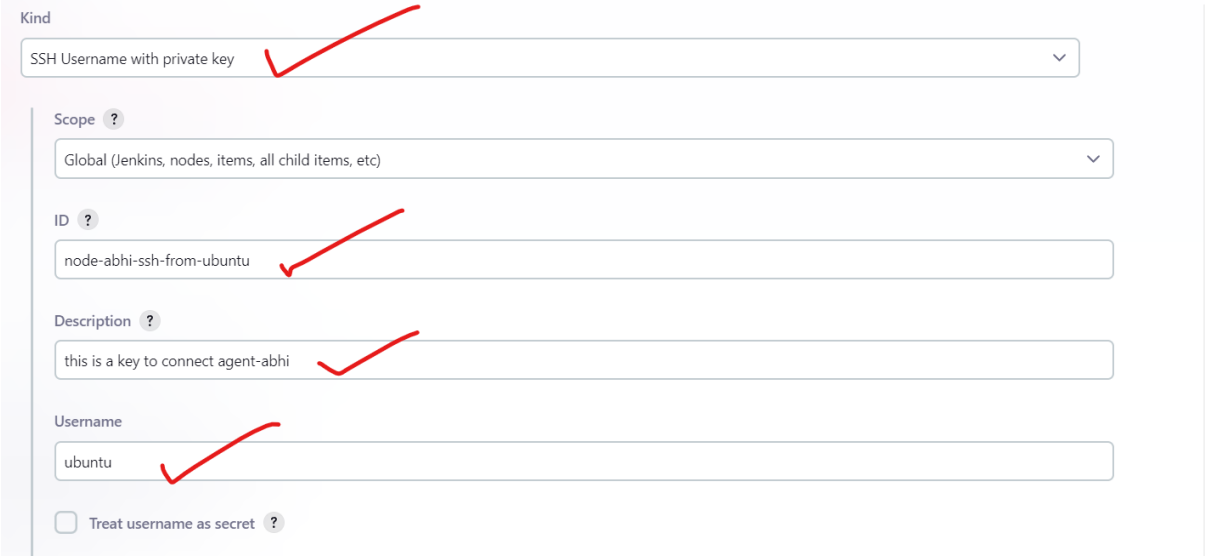
cd apps

pwd

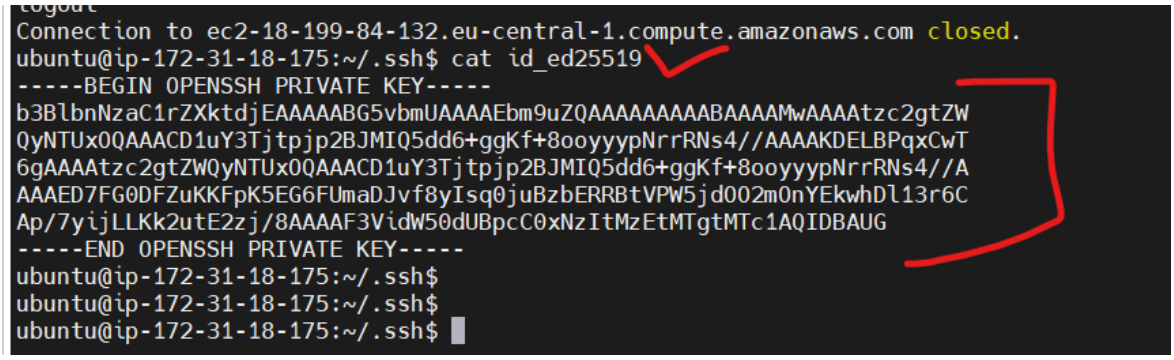
/home/ubuntu/apps -----this path we have to add in above section



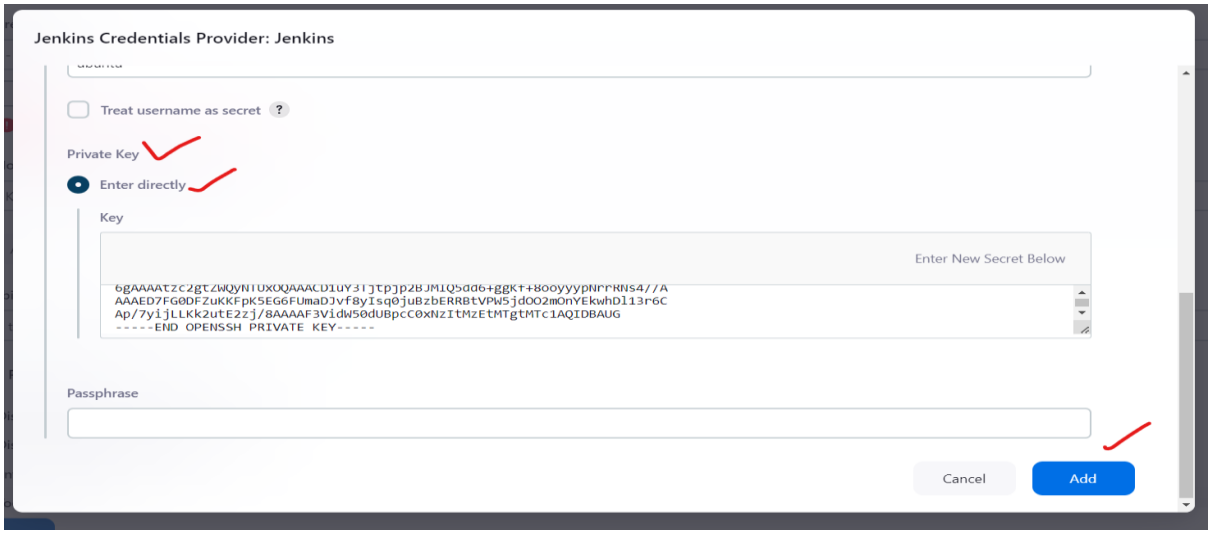
Add credentials so that Jenkins can connect to this server:



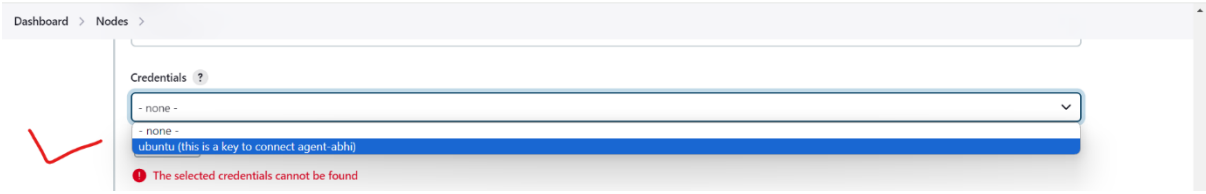
Now we have to add private key of master here.
Go to master node and run> cat id_ed25519

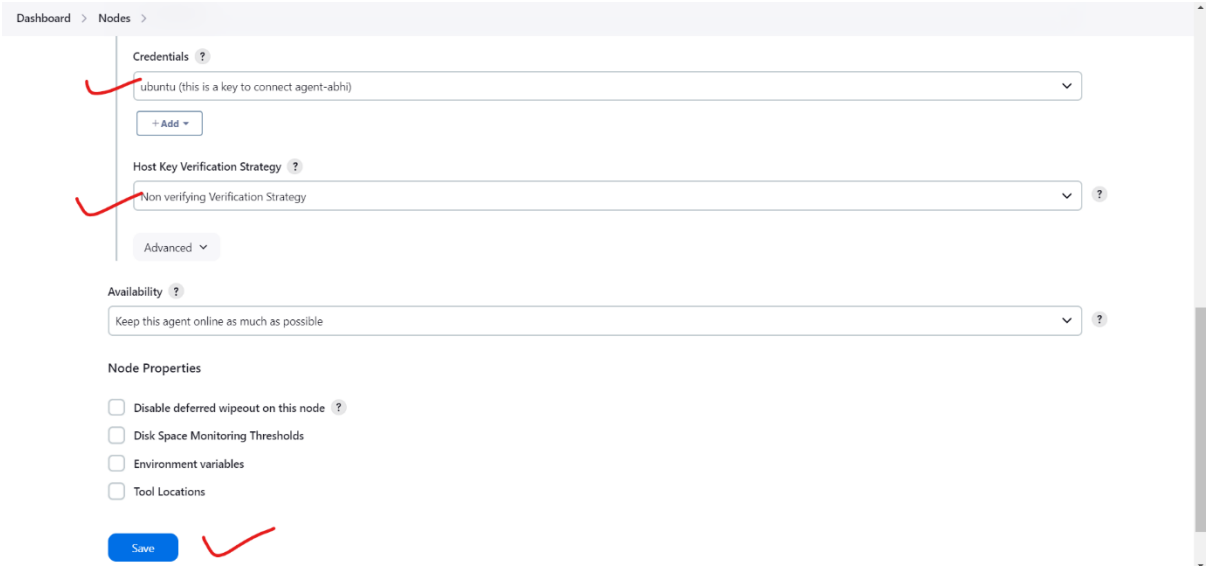


Copy the above private key and paste in node configurations:

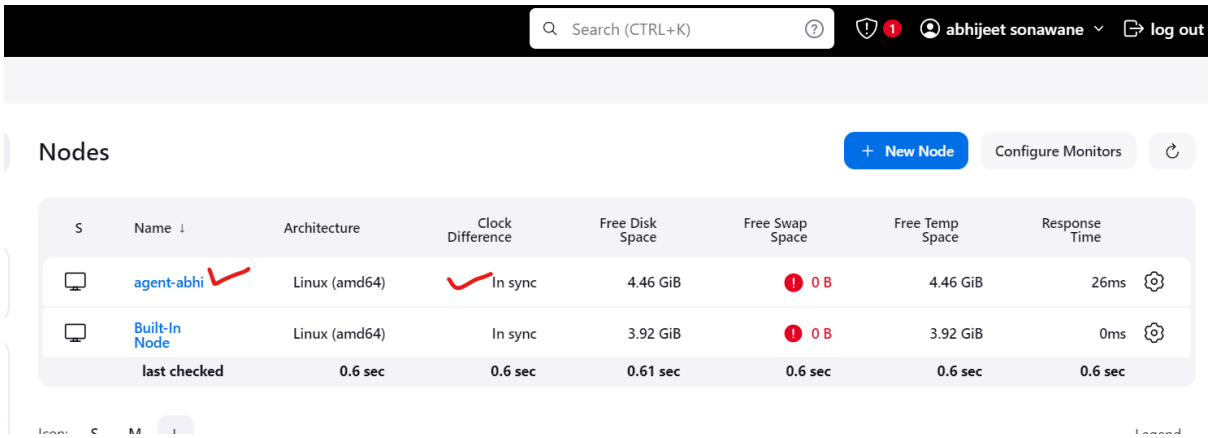


Select the credentials now:





Refresh and check:



8) **On Worker node:**

We have to Install docker and docker compose:

a) **sudo apt-get install docker.io**
b) **sudo apt-get install docker-compose-v2**
c) **docker ps**

after running docker ps command there will be error as permission denied so we need to add user into docker group using following command:

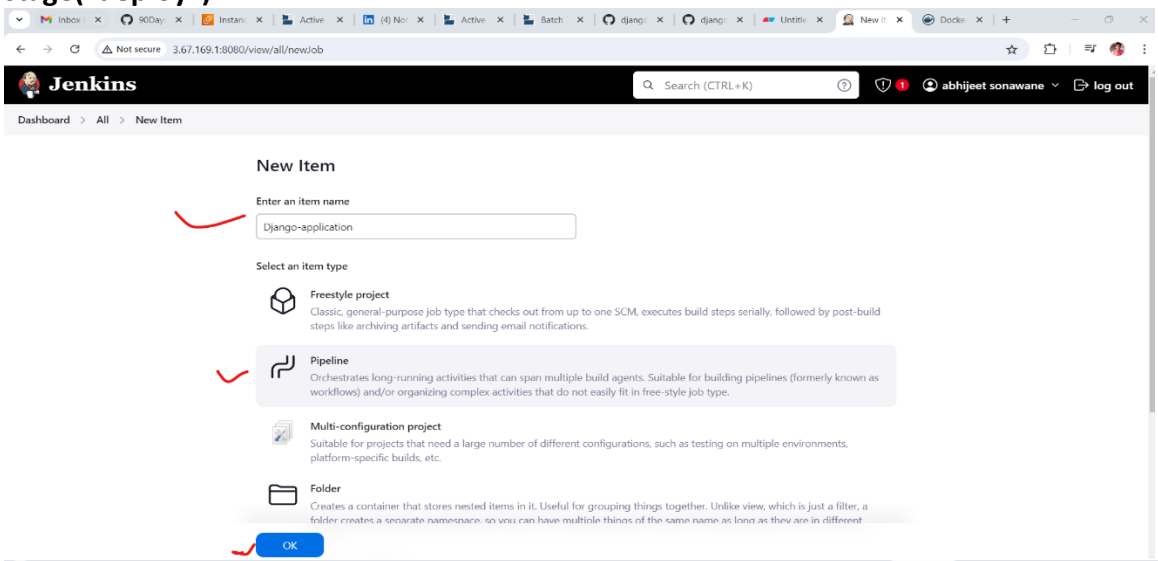
- **sudo usermod -aG docker $USER**
- **sudo chown ubuntu /var/run/docker.sock**

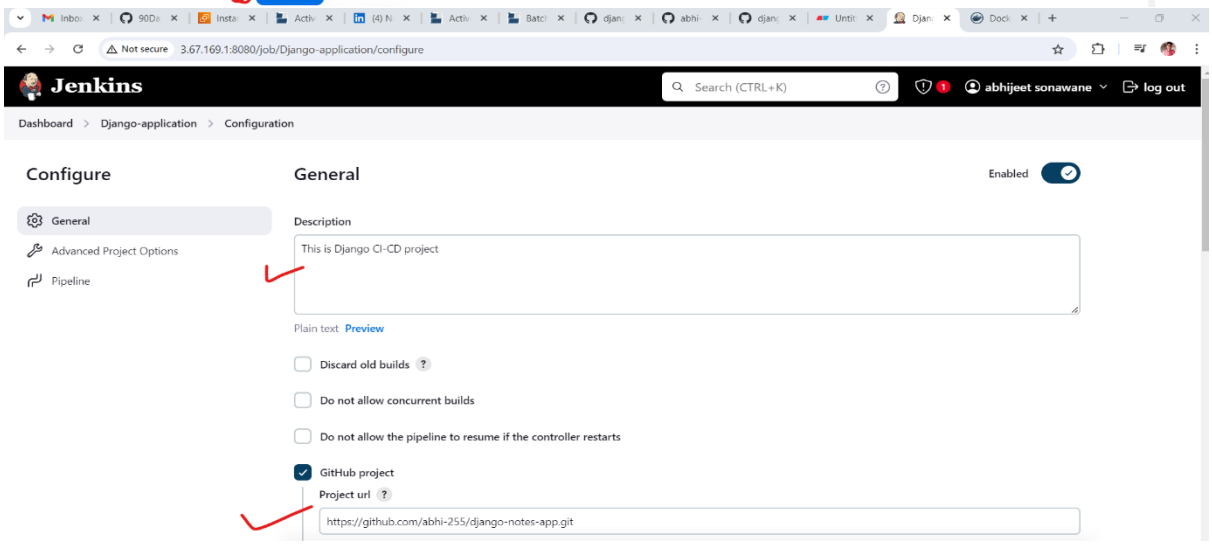
sudo reboot and connect to server after few minutes

Now check docker ps command, there will be no error.

9) **Create a job as pipeline**

Stages will be:
stage("clone code")
stage("build and test")
stage("push image to dockerhub")
stage("deploy")

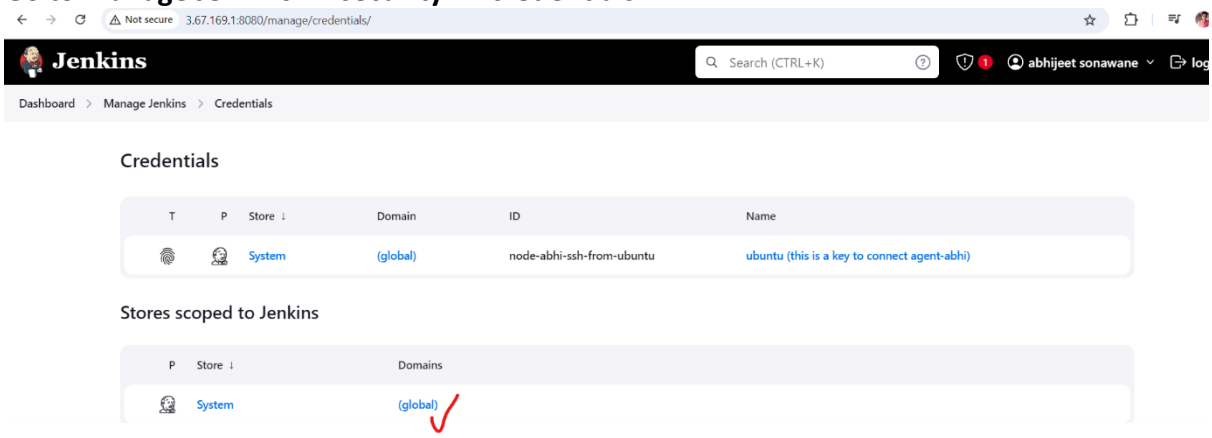




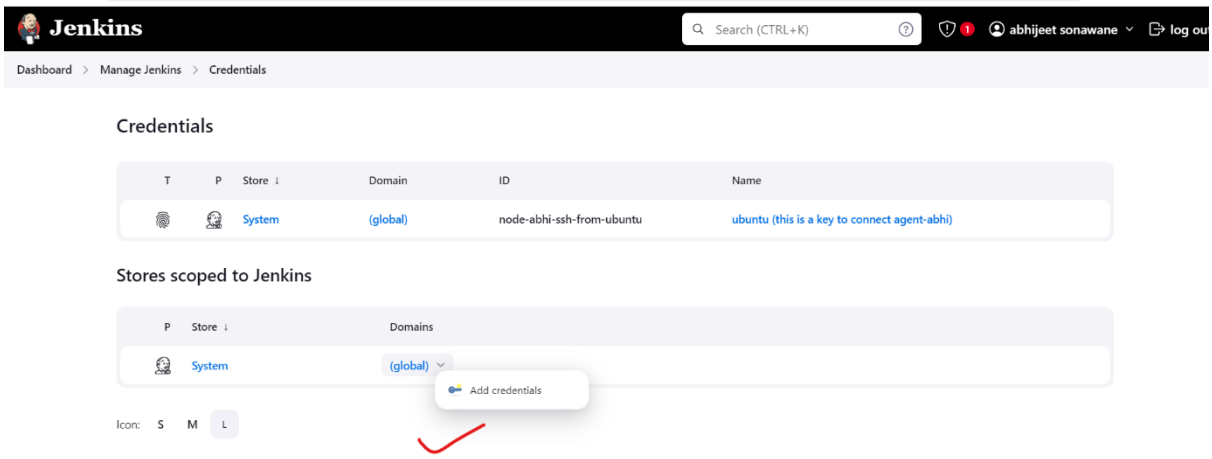
Save the job

As we are building a image and pushing it to dockerhub we have to add docker hub credentials also.

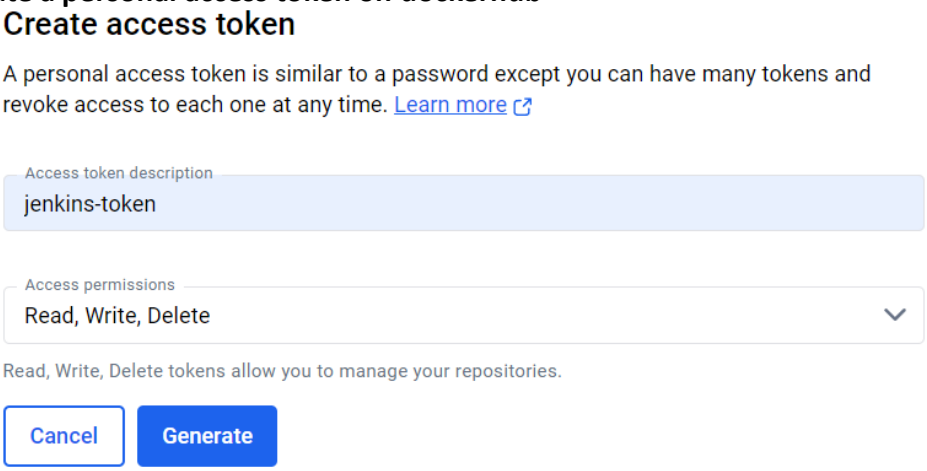
Go to manage Jenkins >> security >> credentials



Click on global option and then add credentials



Generate a personal access token on dockerhub



Access token description
jenkins-token

Access permissions
Read, Write, Delete

To use the access token from your Docker CLI client:

1. Run

```
$ docker login -u abhijeet311
```

Copy

2. At the password prompt, enter the personal access token.

```
dckr_pat_[redacted]fg
```

Copy

Back to access tokens

Copy the token

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

Username ?
abhijeet311

Treat username as secret ?

Password ?
••••••••••••••••••••••••••••••••••

ID ?
dockercred

Description ?
credentials for dockerhub

Create

Use the docker hub username,
Password= personal access token
Id= dockercred (Remember your id)
Click on create

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

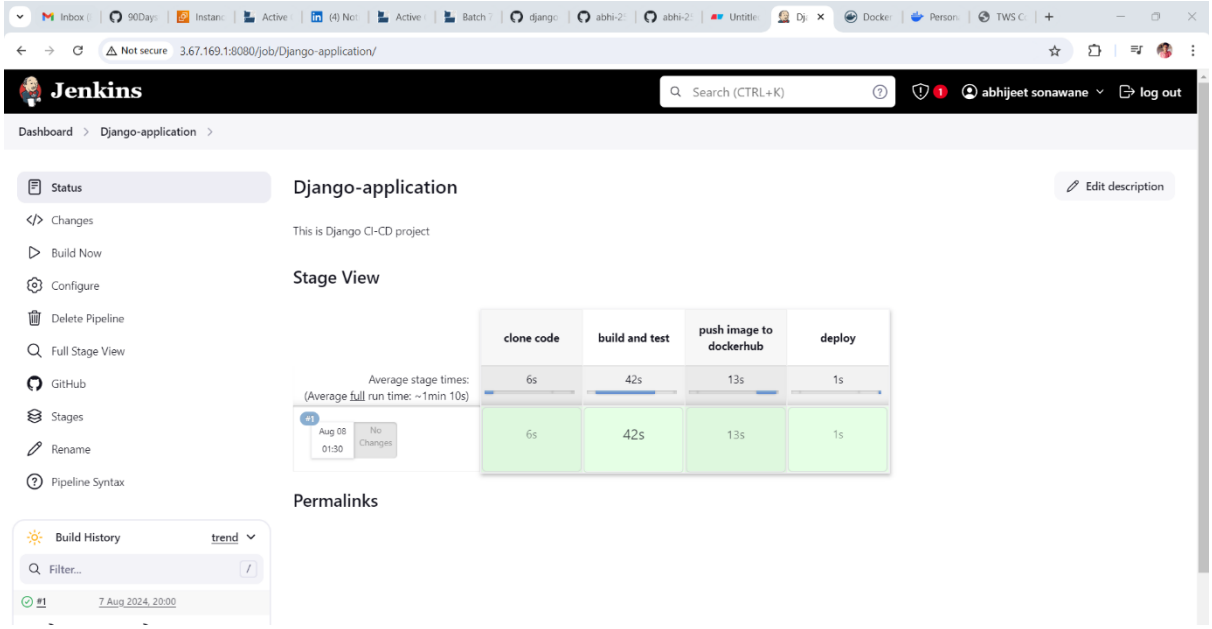
ID	Name	Kind	Description
node-abhi-ssh-from-ubuntu	ubuntu (this is a key to connect agent-abhi)	SSH Username with private key	this is a key to connect agent-abhi
dockercred	abhijeet311/****** (credentials for dockerhub)	Username with password	credentials for dockerhub

Icon: S M L

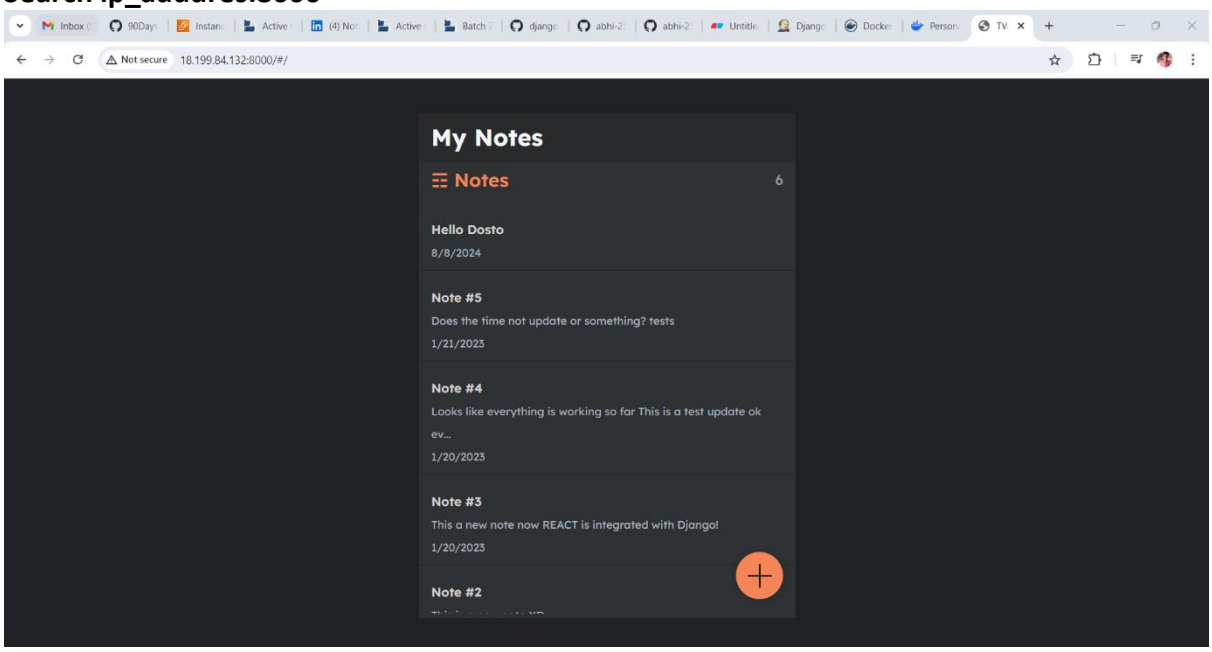
Install stage view plugin for pipeline

Now configure the job again using pipeline attached in this document

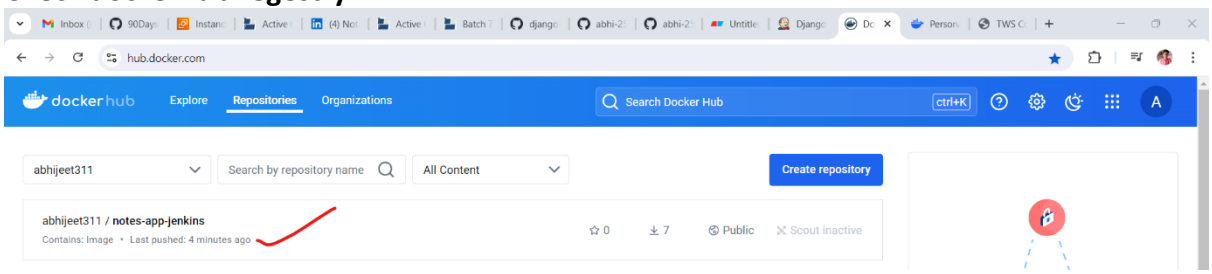
You will get the output:



Search ip_adddres:8000



Check dockerhub regestry:



Not secure 3.67.169.1:8080/job/Django-application/configure

Dashboard > Django-application > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

```
pipeline {
    agent{
        node{
            label "dev"
        }
    }
    stages {
        stage("clone code"){
            steps{
                git url: "https://github.com/abhi-255/django-notes-app.git", branch: "main"
            }
        }
        stage("build and test"){
            steps{
                sh "whoami"
                sh "docker build -t notes-app-jenkins:latest ."
            }
        }
        stage("push image to dockerhub"){
            steps{
                withCredentials(
                    [usernamePassword(
                        credentialsId:"dockercred",
                        passwordVariable:"dockerHubPass",
                        usernameVariable:"dockerHubUser"
                        )
                    ]
                ){
                sh "docker image tag notes-app-jenkins:latest ${env.dockerHubUser}/notes-app-jenkins:latest"
                sh "docker login -u ${env.dockerHubUser} -p ${env.dockerHubPass}"
                sh "docker push ${env.dockerHubUser}/notes-app-jenkins:latest"

                }
            }
        }

        stage("deploy"){
            steps{
                sh "docker compose up -d"

            }
        }
    }
}
```