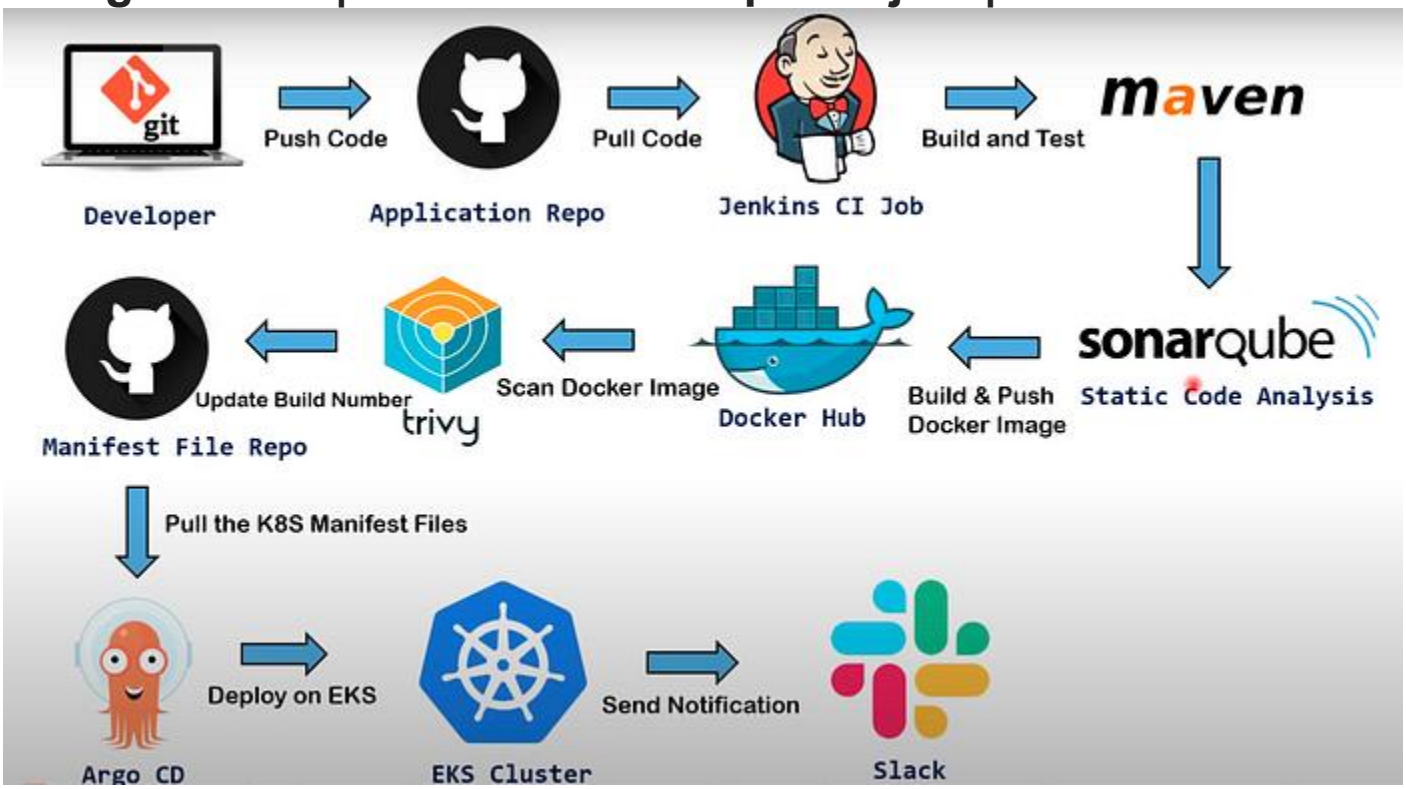
Deploy Register website

Real Time DevOps Project | Deploy to Kubernetes Using Jenkins | End to End DevOps Project | CICD



Step 1: Server 1- Jenkins-Master

Make one server.

Change the hostname IP address to name.

```
sudo nano /etc/hostname
```

Once we update the name, we will reboot our system.

```
sudo init 6
```

Install Jenkins on your server, and as you know, Jenkins can not run without JAVA. So, JAVA installation is a must.

For installation, read my DevOps tools installation from the below link.

Jenkins Installation

Now you have to uncomment two lines in sshd_config, So write a command

```
sudo nano /etc/ssh/sshd_config
```

```
#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
AuthorizedKeysFile      .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none
```

Reload service

```
sudo service sshd reload
```

Generate SSH key

```
ssh-keygen
```

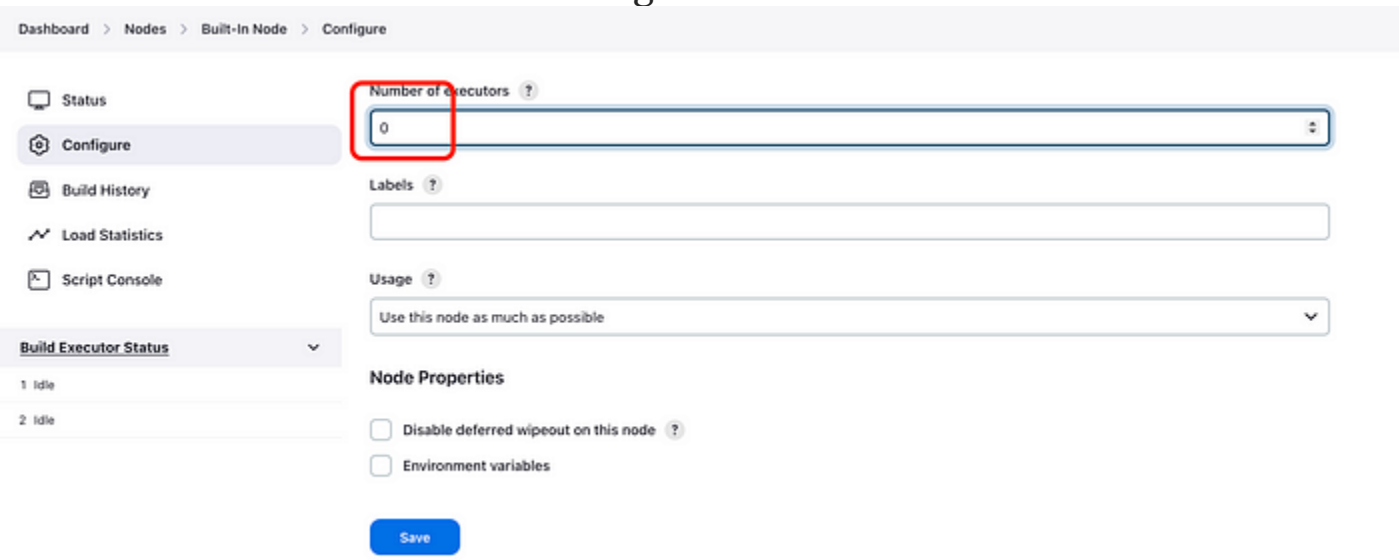
Copy the ssh pub key

```
ubuntu@Jenkins-Master:~$ pwd
/home/ubuntu
ubuntu@Jenkins-Master:~$ cd .ssh/
ubuntu@Jenkins-Master:~/.ssh$ ls
authorized_keys  id_rsa  id_rsa.pub
ubuntu@Jenkins-Master:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDRd4dZzufcklSQRtsrRgbWTMJe1cB4aVdUJ2PptH4ILrqHtS+r5FSD1q5vawdI
CMVk7uJEQZ/7Yi7LrIk3ts089mBuxmRMY+PoXfOnnSYkIinRR3cGd52aZoY7TBQs1Zt8Ilry7BgItsA54E/7ycYxA0nZyCcnQmSH
YdpcY2WQCq8FweJaOwWNhev2ypcuAFpIJQQmzTj7rs7F+JNEJ45L3dTvTvwNyv4GDfoVhIIN1GNxTVbIovwvDGn9aBvXUTfrOcQS
P+zt9Fk6RwQs96OBnUqcxN3Ds9/+hcwToSWSjFpXdrWhM1S1VTQueCNXNbYA6khKA7d8d6c0y/pTiiPsaDjCG3xcfBk6fWqQqQGC
Exj/csr1goCBRpuFtqTKugqqyBnEWIiNebNh/Dj76Eil/YBZboE5JVVNFXI4be8u9lAHldbNglGUtbqnbxZ6/gidF2EinmgyAt2N
iyB3ETv3HpTENEo/EYpw9Y/ktUmRQJPHUtHlZ0Bfc= ubuntu@Jenkins-Master
ubuntu@Jenkins-Master:~/.ssh$
```

Open Jenkins with server IP :8080

In Jenkins, open Dashboard > Manage Jenkins > Nodes

Under Node > Built in Node > Configure



Save and go to Dashboard again.

Dashboard > Manage Jenkins > Nodes > New node

Dashboard > Manage Jenkins > Nodes > New node

New node

Node name

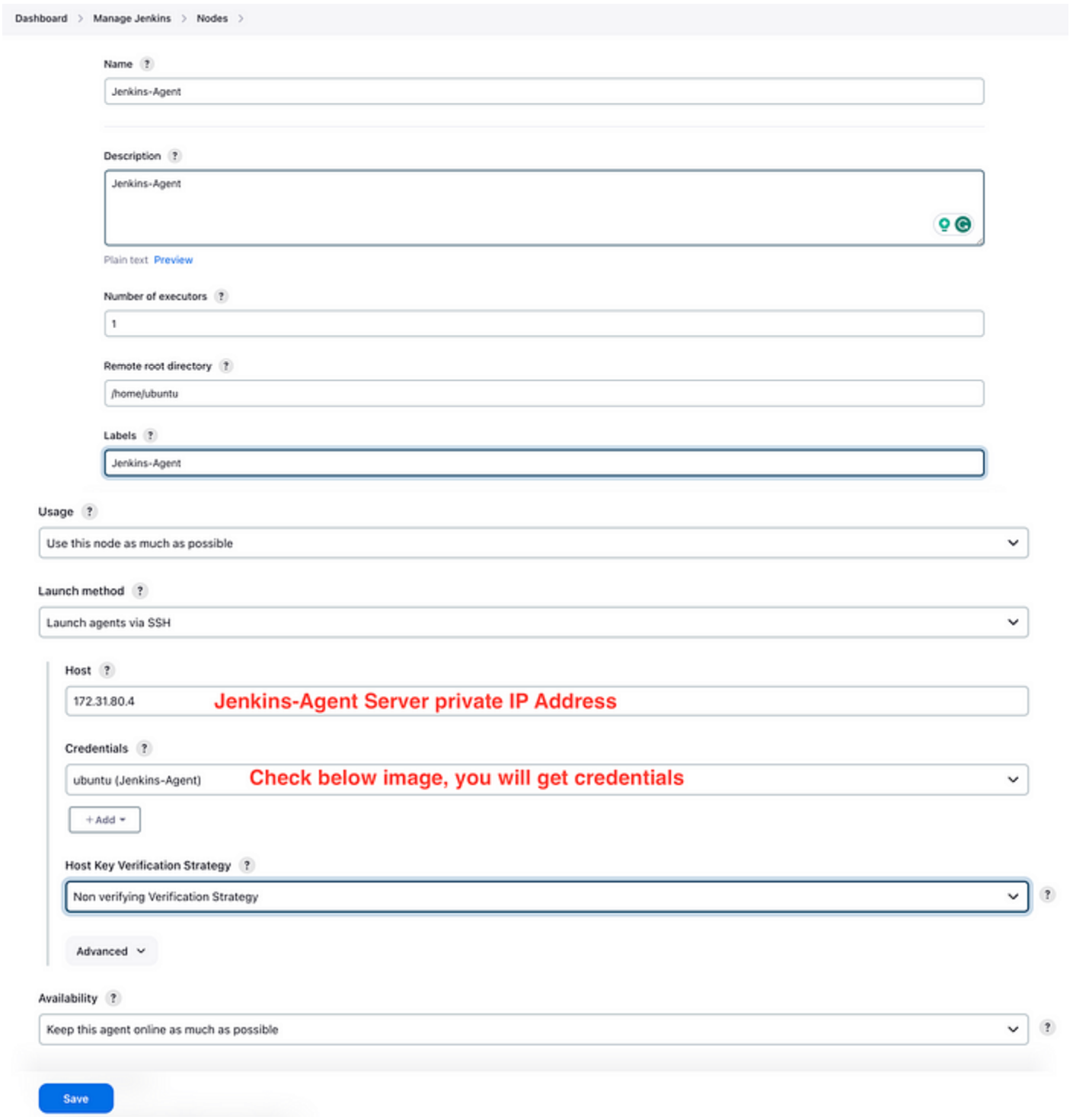
Jenkins-Agent

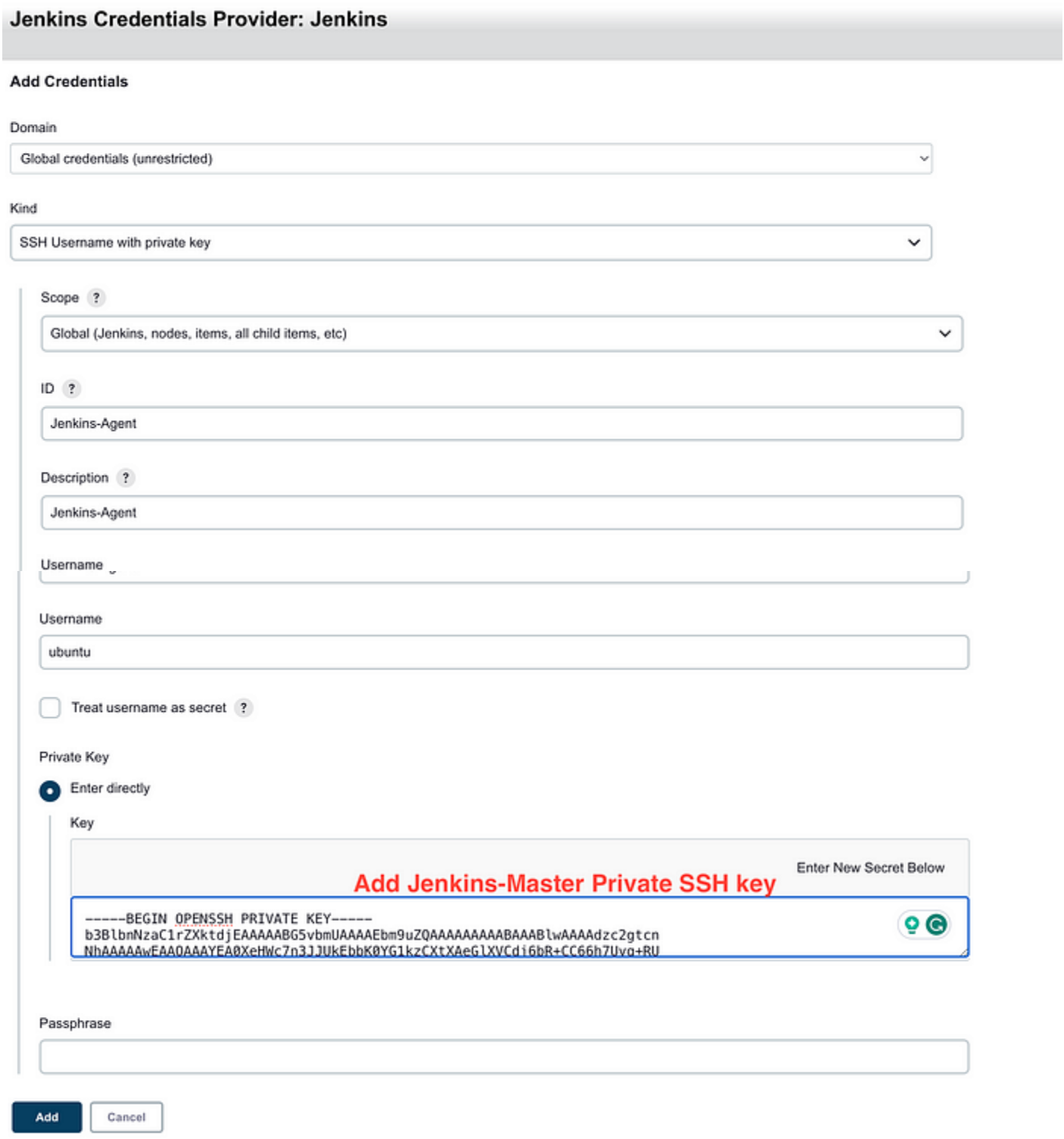
Type

Permanent Agent

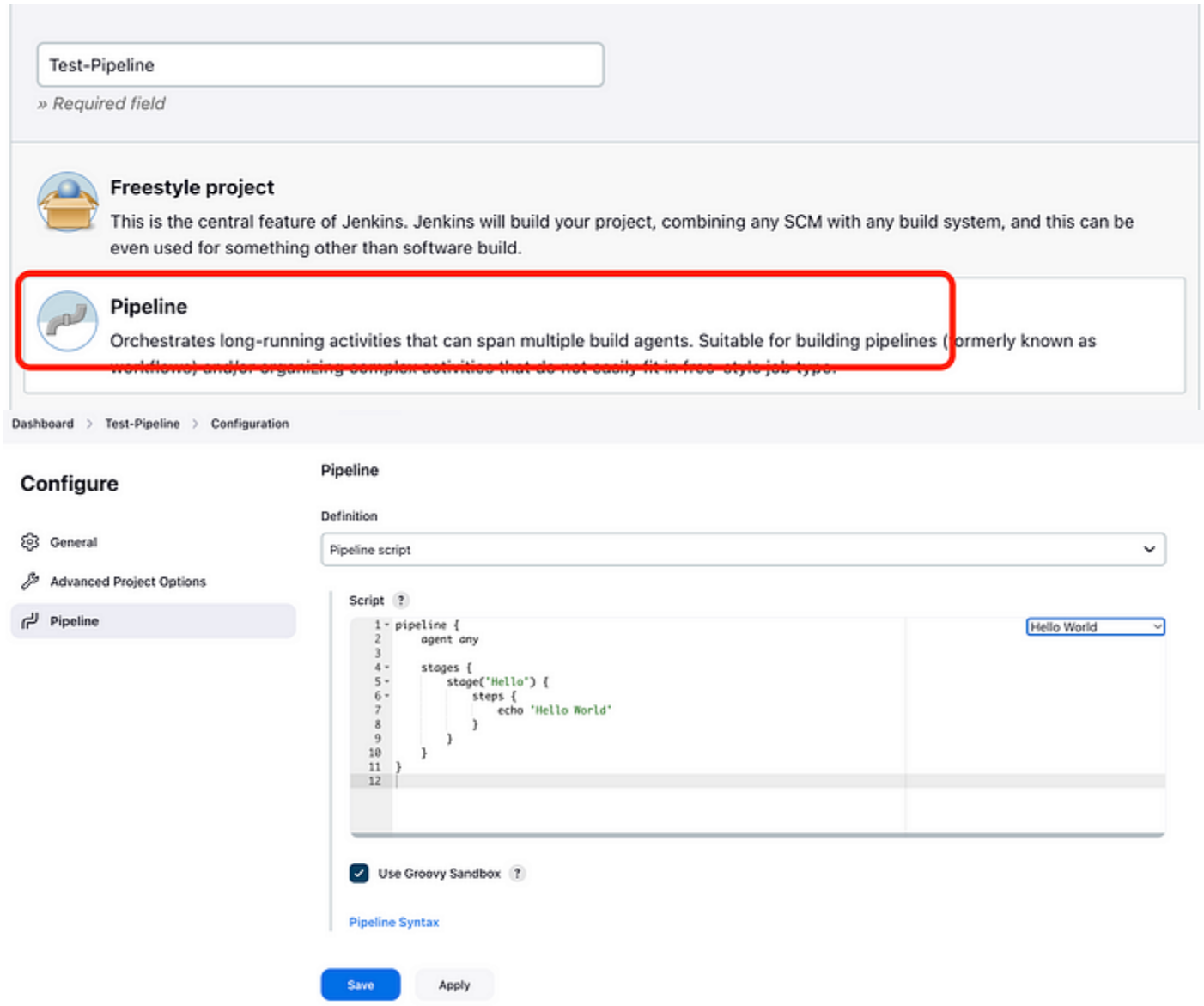
Adds a plain, permanent agent to Jenkins. This is called "permanen
higher level of integration with these agents, such as dynamic prov
types apply — for example such as when you are adding a physical
outside Jenkins, etc.

Create


Dashboard > Manage Jenkins > Nodes >
Name
Jenkins-Agent
Description
Jenkins-Agent
Plain text Preview
Number of executors
1
Remote root directory
/home/ubuntu
Labels
Jenkins-Agent
Usage
Use this node as much as possible
Launch method
Launch agents via SSH
Host
172.31.80.4
Jenkins-Agent Server private IP Address
Credentials
ubuntu (Jenkins-Agent)
Check below image, you will get credentials
+ Add
Host Key Verification Strategy
Non verifying Verification Strategy
Advanced
Availability
Keep this agent online as much as possible
Save



Start new pipeline



Test Jenkins

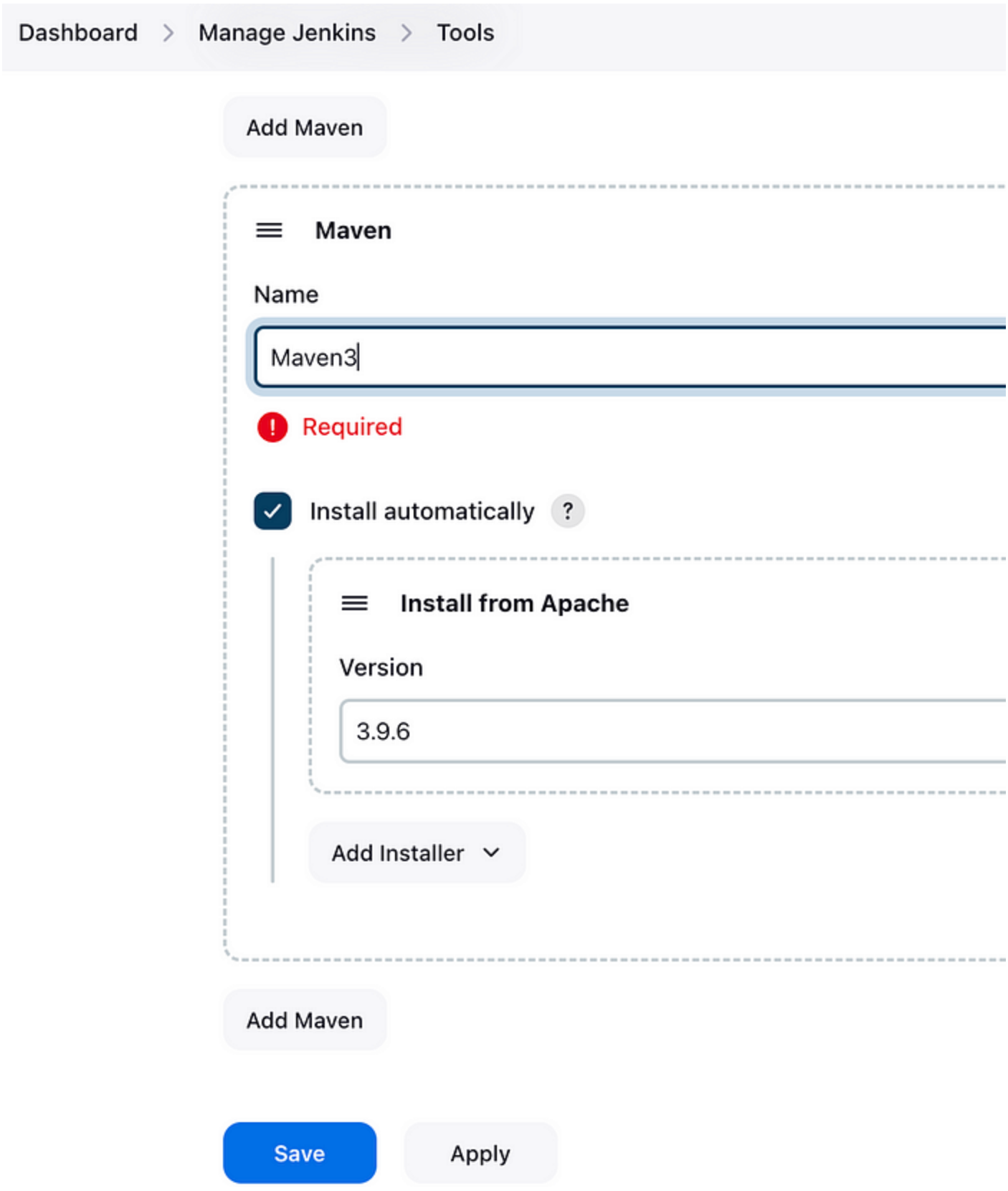
Build now.

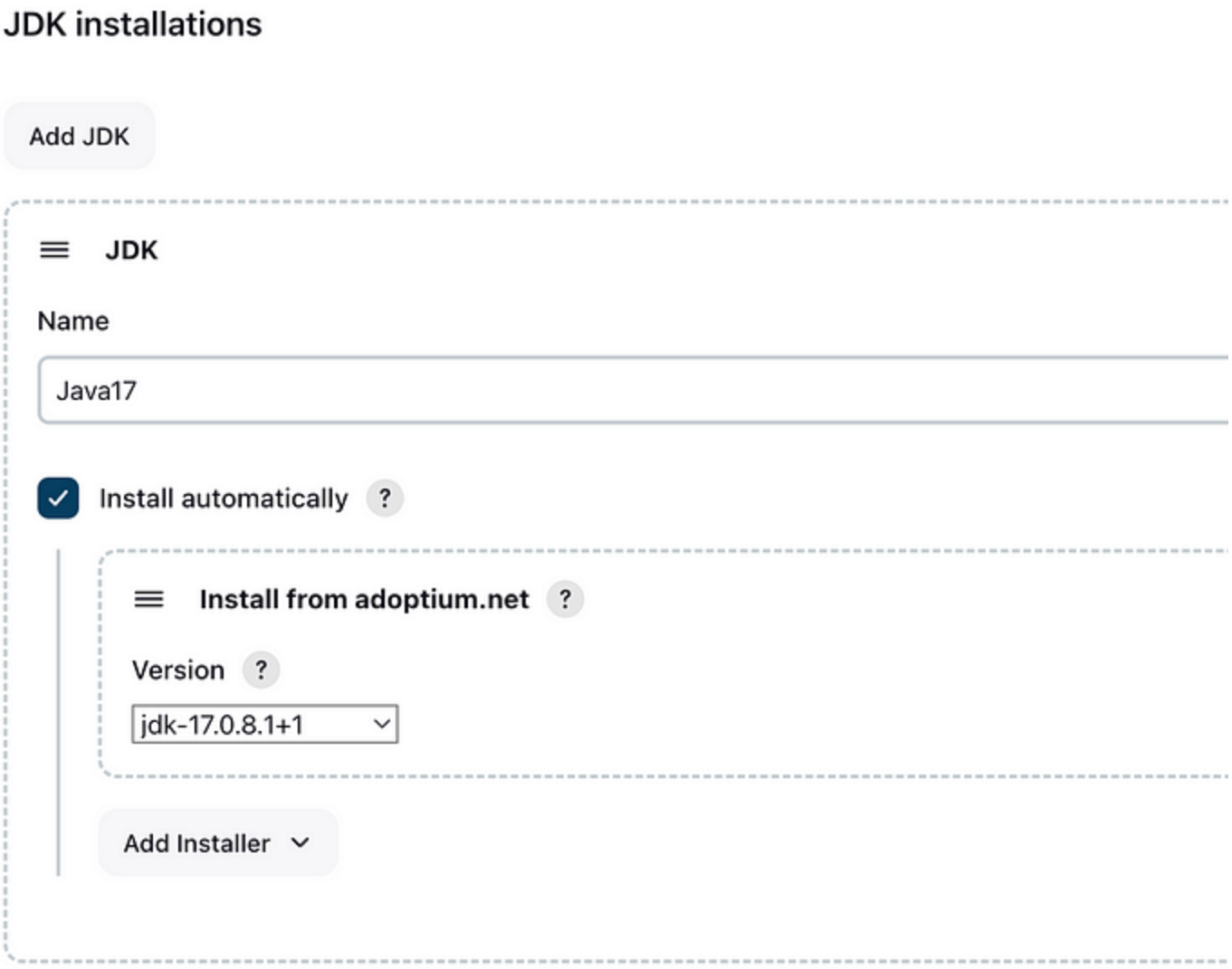
Dashboard > Manage Jenkins > Plugins

-> Maven Integration in Jenkins.

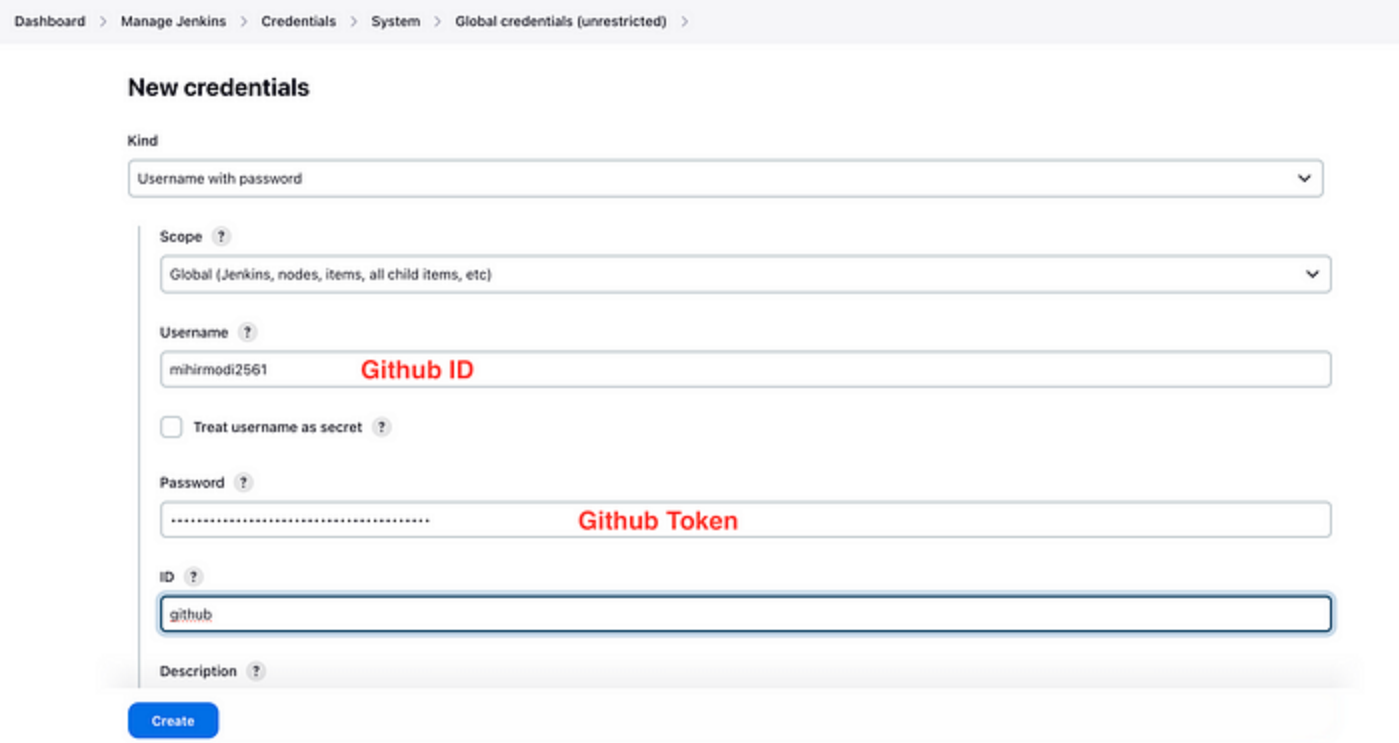
→ Pipeline Maven Integration

→ Eclipse Temurin Installer


Dashboard
Manage Jenkins
Tools
Add Maven
Maven
Name
Maven3
Required
Install automatically
Install from Apache
Version
3.9.6
Add Installer
Add Maven
Save
Apply



Save and start add-in (git-hub)



Step 2: Server 2- Jenkins-Agent

Like Jenkins-Master, we must make another Jenkins-Agent server and change the **hostname**.

Install JAVA from DevOps tools installation → **JAVA INSTALLATION**.

Install Docker from DevOps tools installation → **Docker INSTALLATION**.

Give a rights docker to **Jenkins-Agent user.**

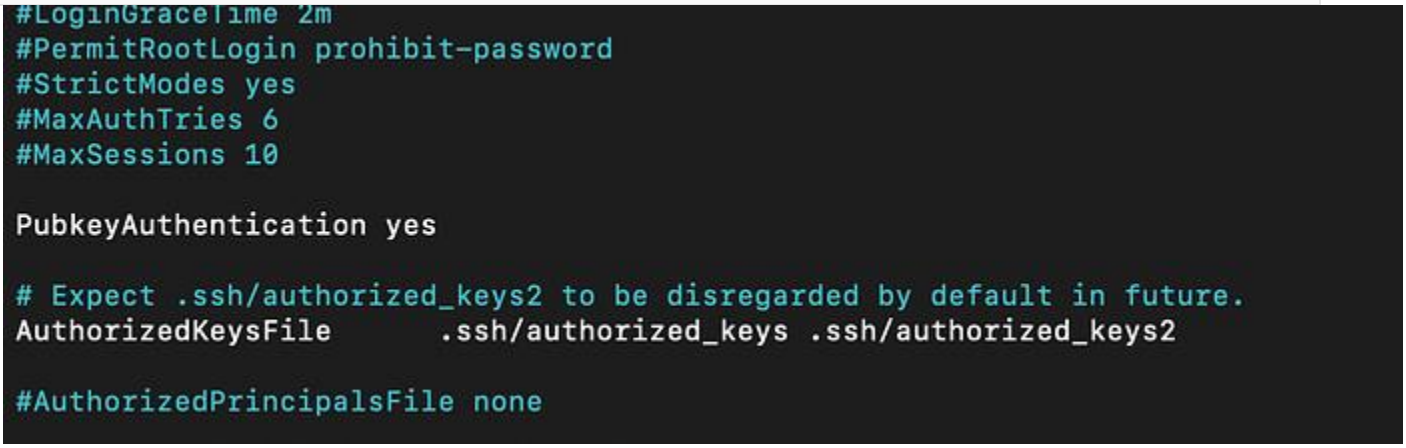
```
sudo usermod -aG docker $USER
```

Restart server:

```
sudo init 6
```

Now you have to uncomment two lines in sshd_config, So write a command

```
sudo nano /etc/ssh/sshd_config
```



Reload service

```
sudo service sshd reload
```

Paste the ssh pub key.

Step: Create a Jenkins File in the GitHub repo.

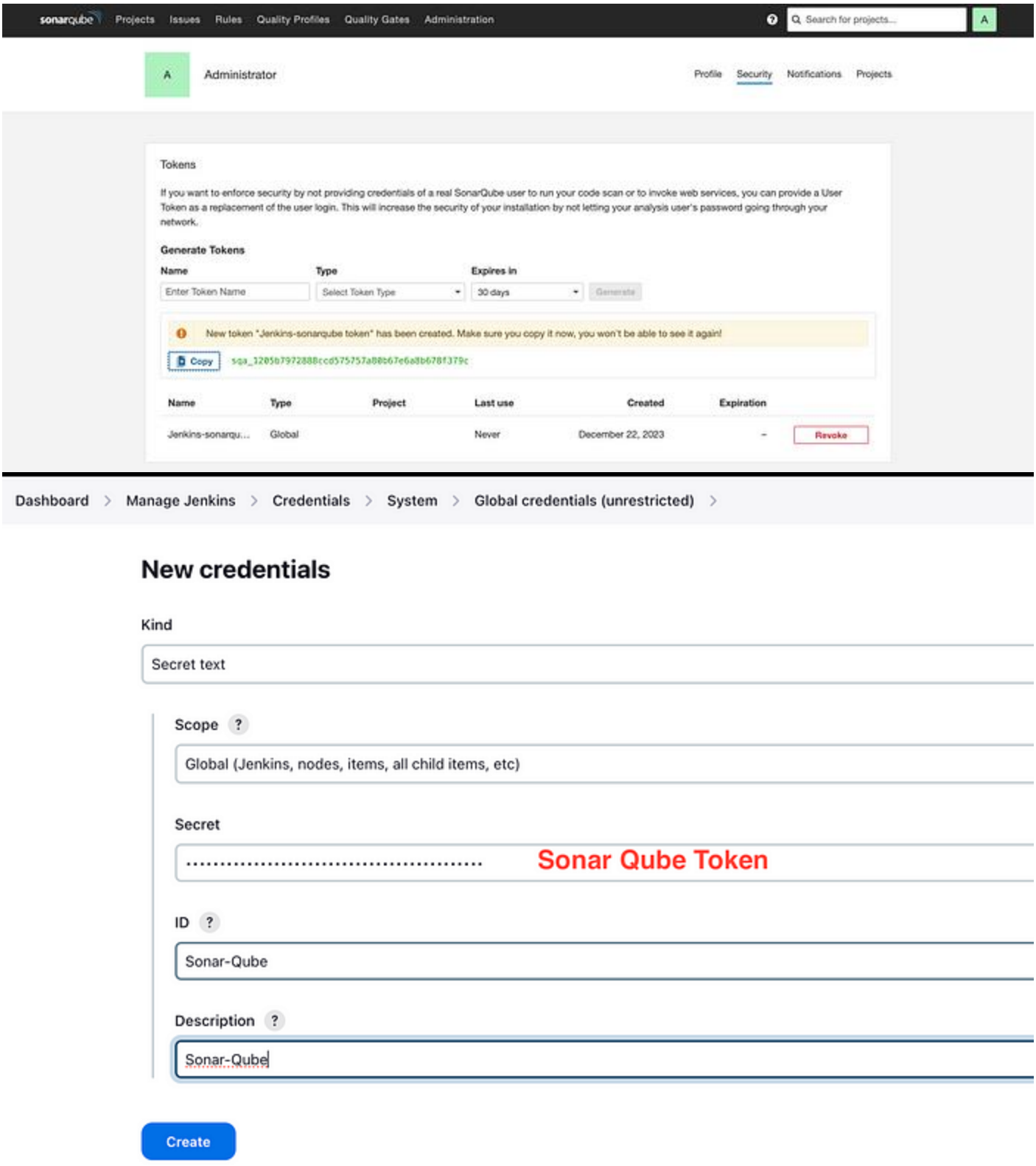
Github link:- https://github.com/mihirmodi2561/register-app

Step 4: **New Server SONAR QUBE**

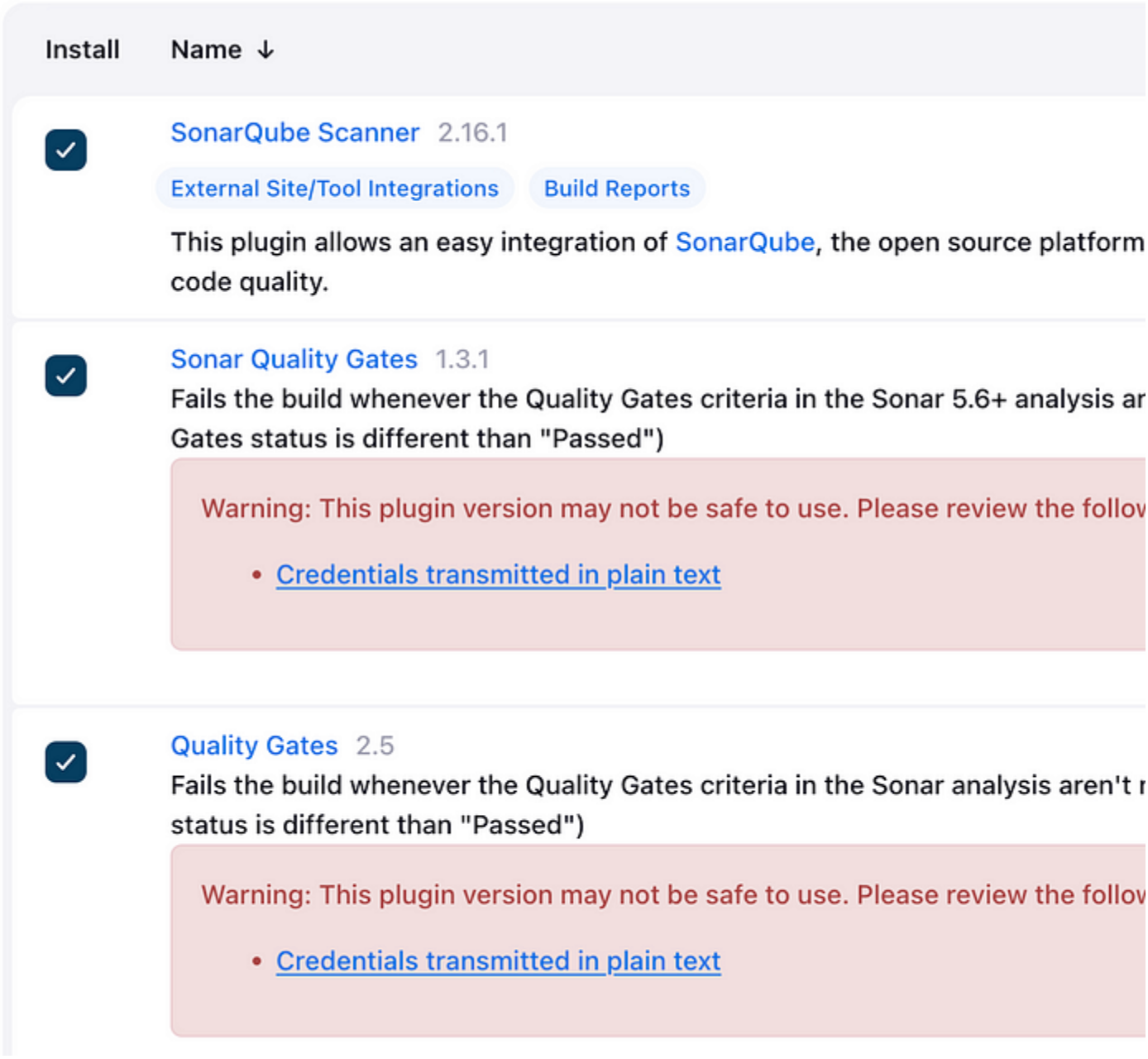
SONAR QUBE INSTALLTION

Once you have installed Sonar Qube, start integrating with Jenkins.

My Account >Security > Generate Token



Add This 3 Plugin in Jenkins



Once you installed, Now you have to set up in Jenkins system.

Manage Jenkins > system

Name

sonar-qube-server

Server URL

Default is http://localhost:9000

172.31.92.63:9000

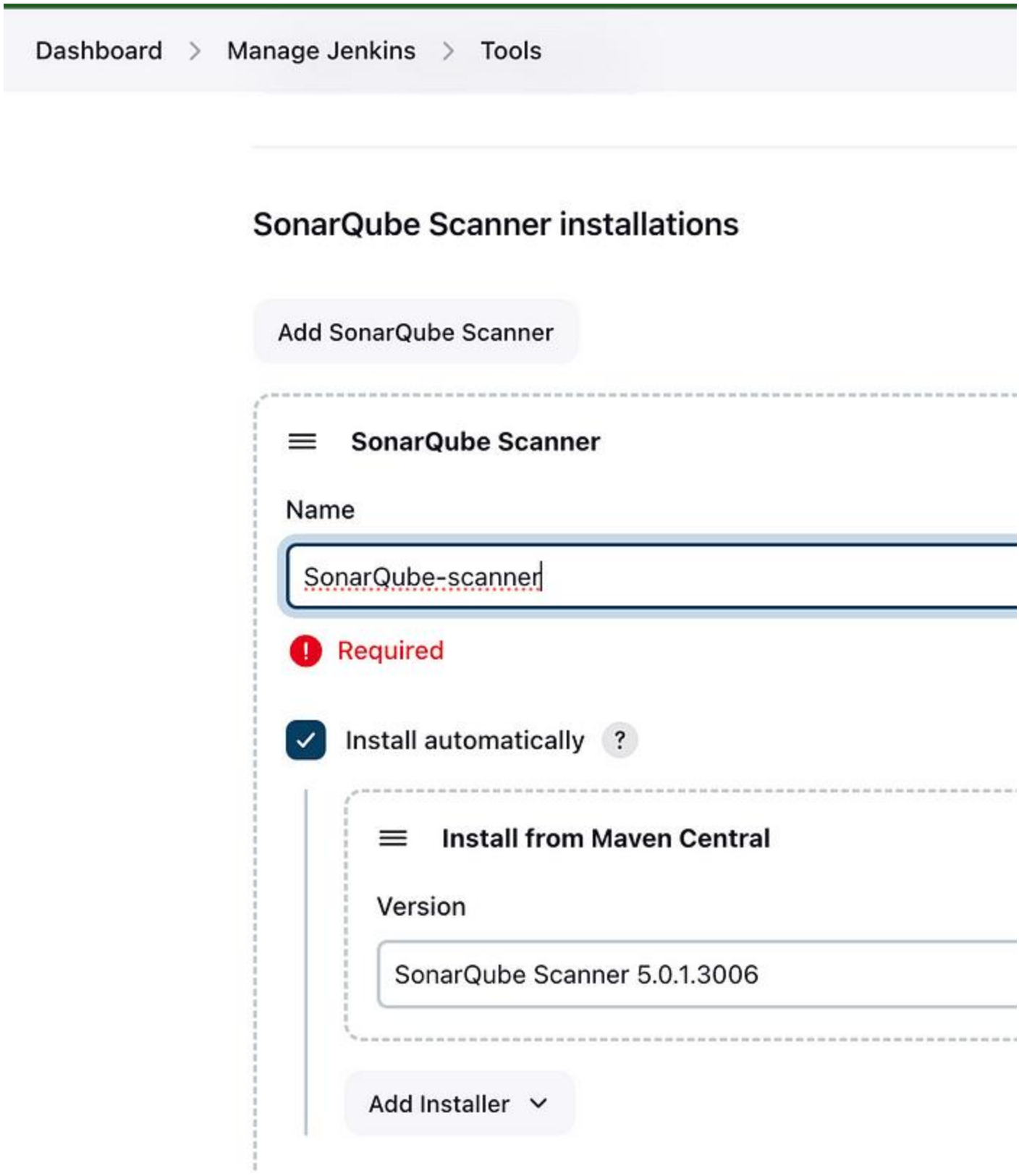
Server authentication token

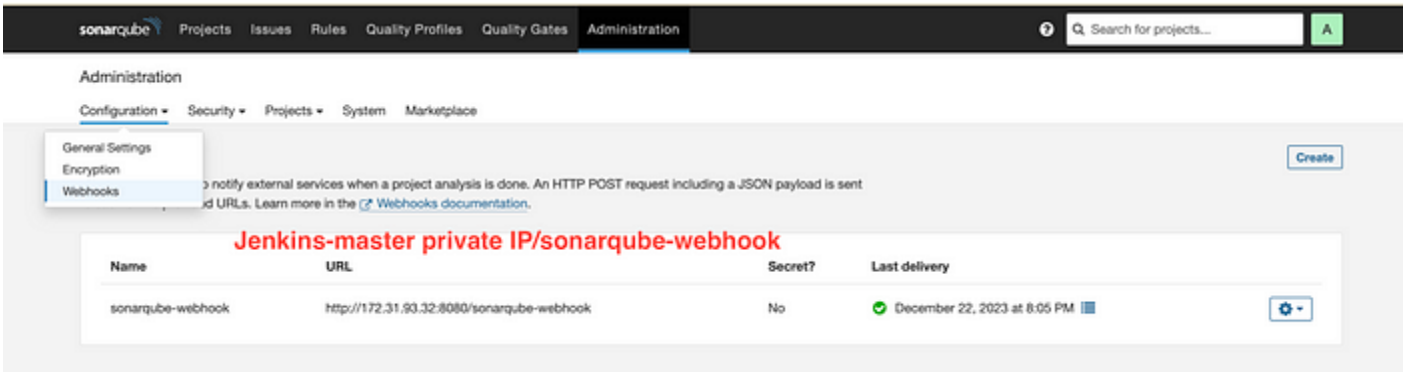
SonarQube authentication token. Mandatory when anonymous access is disabled.

Sonar-Qube

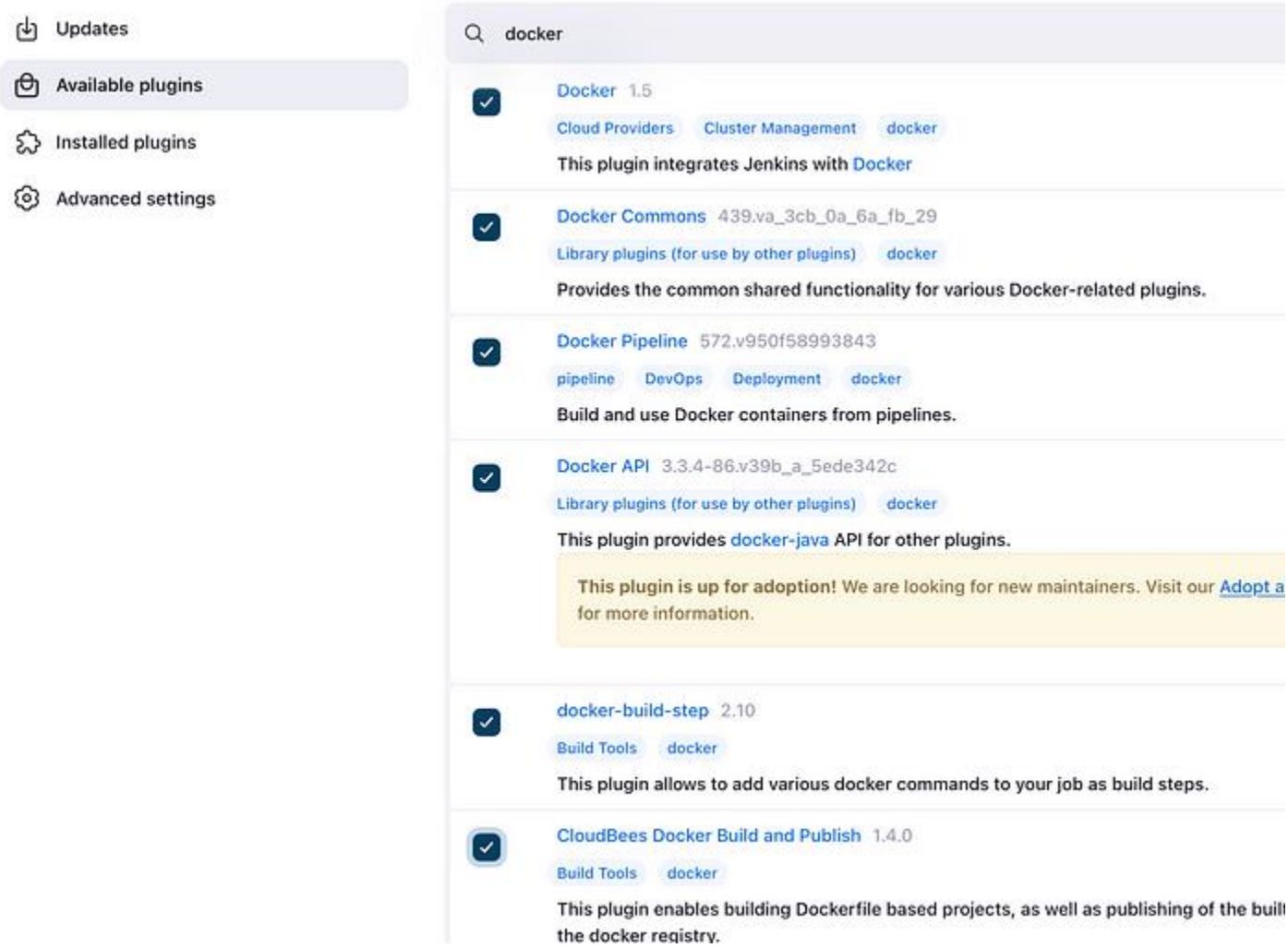
+ Add

Advanced


Dashboard
Manage Jenkins
Tools
SonarQube Scanner installations
Add SonarQube Scanner
SonarQube Scanner
Name
SonarQube-scanner
Required
Install automatically
?
Install from Maven Central
Version
SonarQube Scanner 5.0.1.3006
Add Installer



Add Plugins for Docker



Run Jenkins script

```
pipeline{
    agent { label 'Jenkins-Agent'}
    tools{
        jdk 'Java17'
        maven 'Maven3'
    }
    environment{
            APP_NAME = "register-app-ci"
            RELEASE = "1.0.0"
            DOCKER_USER = "mihirmodi2561"
            DOCKER_PASS = 'DockerHub'
            IMAGE_NAME = "${DOCKER_USER}" + "/" + "${APP_NAME}"
            IMAGE_TAG = "${RELEASE}-${BUILD_NUMBER}"
    }
    stages{
        stage("Cleanup Worksapce"){
            steps{
                cleanWs()
            }
        }
        stage("Check out from SCM"){
            steps{
                git credentialsId: 'github', url:
'https://github.com/mihirmodi2561/Register-app-ci-cd'
            }
        }
        stage("Build Application"){
            steps{
                sh "mvn clean package"
            }
        }
        stage("Test Application"){
            steps{
                sh "mvn test"
            }
        }
        stage("SonarQube Analysis"){
            steps{
                script{
                    withSonarQubeEnv(credentialsId: 'Sonar-Qube'){
                    sh "mvn sonar:sonar"
                    }
                }
            }
        }
        stage("Quality check"){
            steps{
                script{
                    waitForQualityGate abortPipeline: false, credentialsId:
'Sonar-Qube'
                }
            }
```

```
        }
        stage("Build & Push Docker Image"){
            steps{
                script{
                    docker.withRegistry('',DOCKER_PASS){
                        docker_image = docker.build "${IMAGE_NAME}"
                    }


                    docker.withRegistry('',DOCKER_PASS){
                        docker_image.push("${IMAGE_TAG}")
                        docker_image.push('latest')
                    }
                }
            }
        }
        stage("Trivy sacan"){
            steps{
                script{
                    sh(' docker run -v
/var/run/docker.sock:/var/run/docker.sock aquasec/trivy image
mihirmodi2561/register-app-ci:latest --no-progress --scanner vuln --exit-code
0 --severity HIGH, CRITICAL  --format table')
                }
            }
        }
        stage('Clean Artifacts'){
            steps{
                script{
                    sh "docker rmi ${IMAGE_NAME}:${IMAGE_TAG}"
                    sh "docker rmi ${IMAGE_NAME}:latest"
                }
            }
        }
    }
}
```

Setup Bootstrap EKS Server.

Install AWS Cli on the above EC2

```
sudo su
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
apt install unzip,   $ unzip awscliv2.zip
sudo ./aws/install
```

```
pip3 install --user awscli
sudo ln -s $HOME/.local/bin/aws /usr/bin/aws
aws --version
```

Installing kubectl

```
sudo su
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.1/2023-04-
19/bin/linux/amd64/kubectl
ll , $ chmod +x ./kubectl  //Gave executable permisions
mv kubectl /bin   //Because all our executable files are in /bin
kubectl version --output=yaml
```

Installing eksctl

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname
-s)_amd64.tar.gz" | tar xz -C /tmp
cd /tmp
$ ll
$ sudo mv /tmp/eksctl /bin
$ eksctl version
```

Setup Kubernetes using eksctl

```
eksctl create cluster --name virtualtechbox-cluster \
--region ap-south-1 \
--node-type t2.small \
--nodes 3 \
```

check the node from below.

```
kubectl get nodes
```

Add IAM role to EKS-bootstrap

Add Permission of AdministratorAcess
after creating the IAM Role, add the IAM role to the EKS-bootstrap server.

ArgoCD Installation on EKS Cluster and Add EKS Cluster to ArgoCD

First, create a namespace

```
kubectl create namespace argocd
```

Next, let's apply the yaml configuration files for ArgoCd

```
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```

Now, we can view the pods created in the ArgoCD namespace.

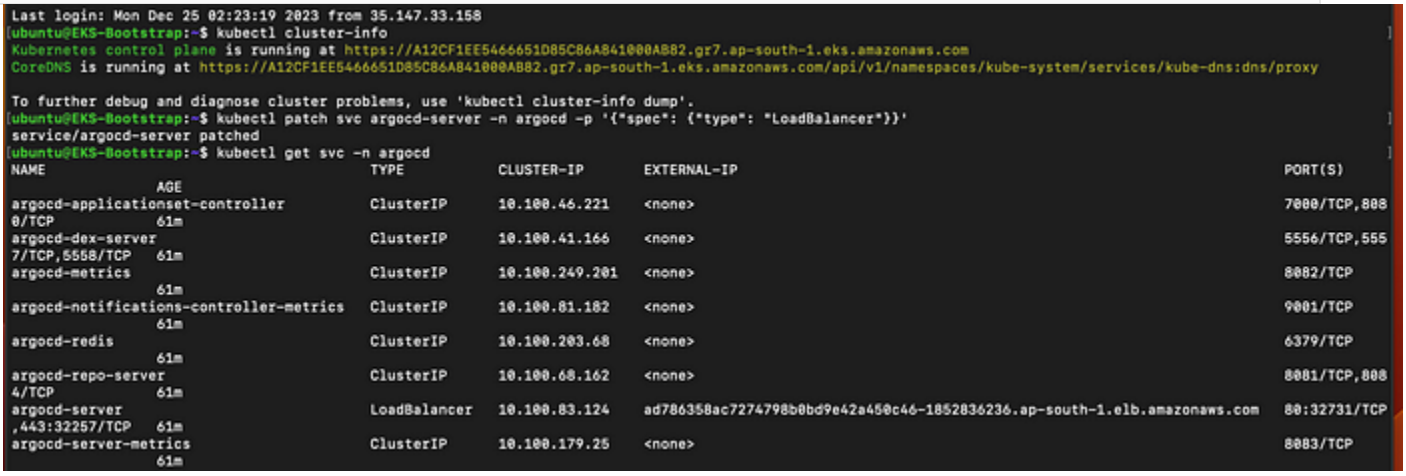
```
kubectl get pods -n argocd
```

To interact with the API Server we need to deploy the CLI:

```
curl --silent --location -o /usr/local/bin/argocd
https://github.com/argoproj/argo-cd/releases/download/v2.4.7/argocd-linux-
amd64
chmod +x /usr/local/bin/argocd
```

Expose argocd-server

```
kubectl patch svc argocd-server -n argocd -p '{"spec": {"type":
"LoadBalancer"}}'
```



Wait about 2 minutes for the LoadBalancer creation

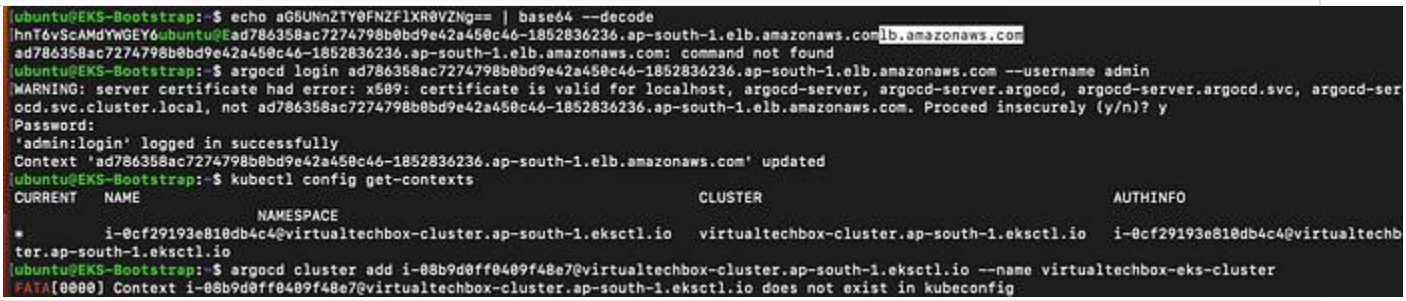
```
kubectl get svc -n argocd
```

Get a password and decode it.

```
kubectl get secret argocd-initial-admin-secret -n argocd -o yaml
echo WXVpLUg2LWxoWjRkSHFmSA== | base64 --decode
```

login to ArgoCD from CLI

```
argocd login ad786358ac7274798b0bd9e42a450c46-1852836236.ap-south-1.elb.amazonaws.com --username admin
```



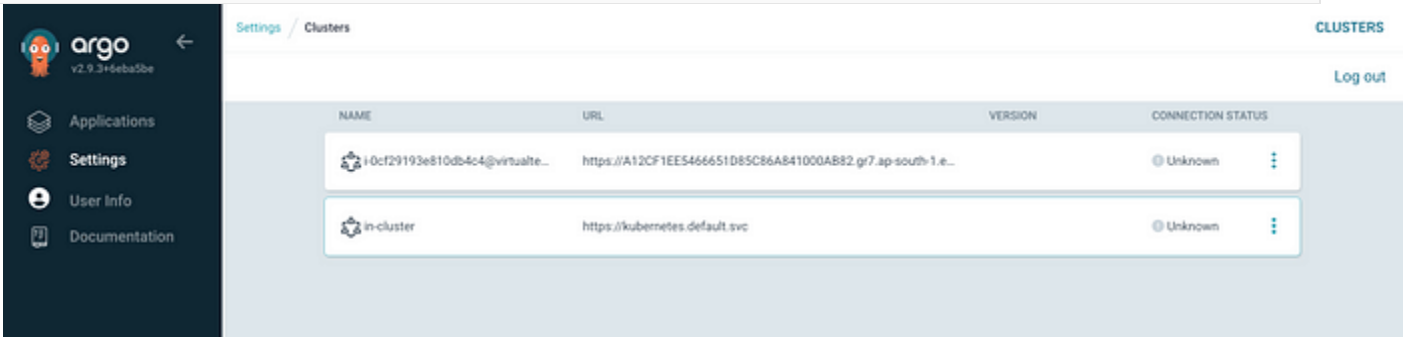

```
argocd cluster list
```

Below command will show the EKS cluster

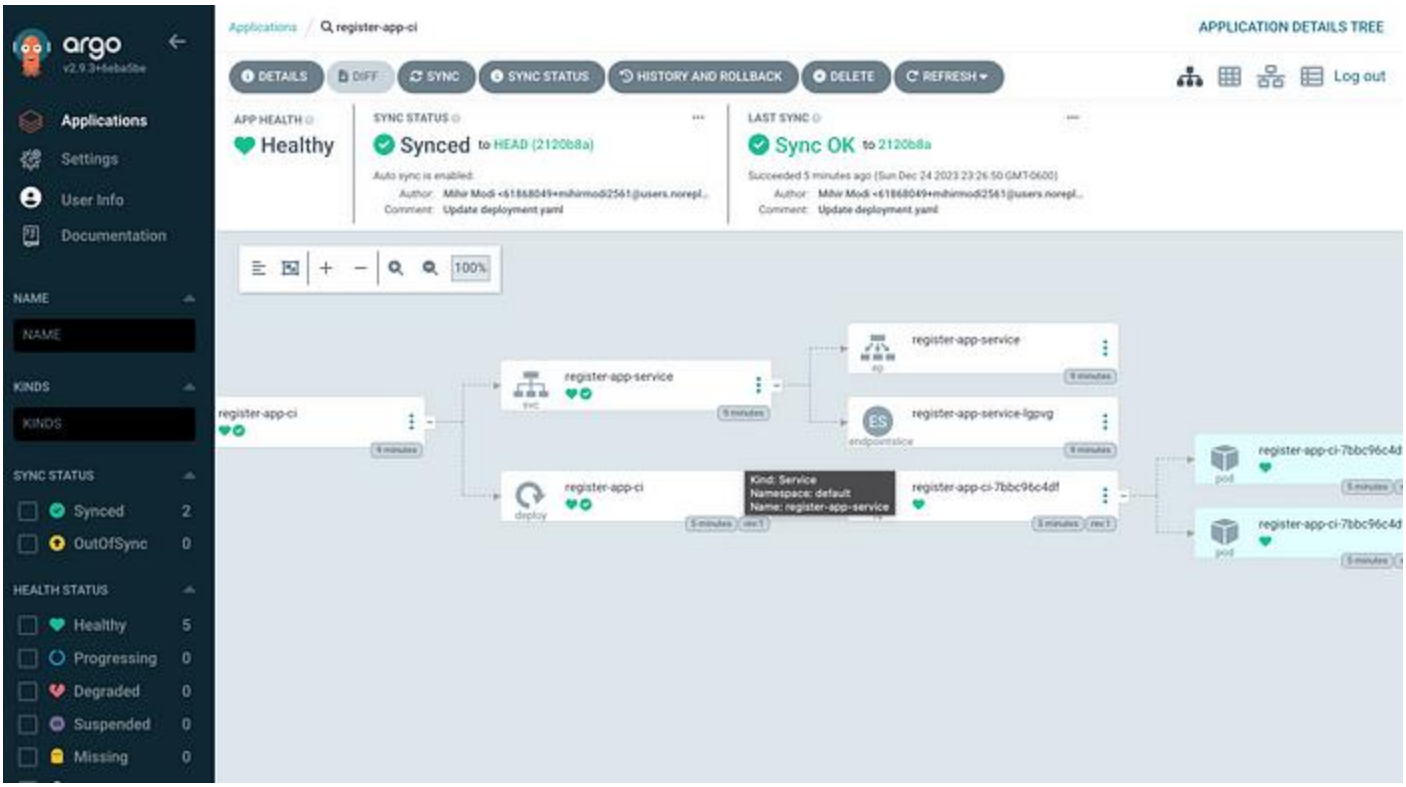
```
kubectl config get-contexts
```

Add the above EKS cluster to ArgoCD with the below command

```
argocd cluster add i-0cf29193e810db4c4@virtualtechbox-cluster.ap-south-
1.eksctl.io
kubectl get svc
```



argo → setting → repositories → connect repo with github repo



argo application healthy.

Website launched sueccsfully.

Thank you..