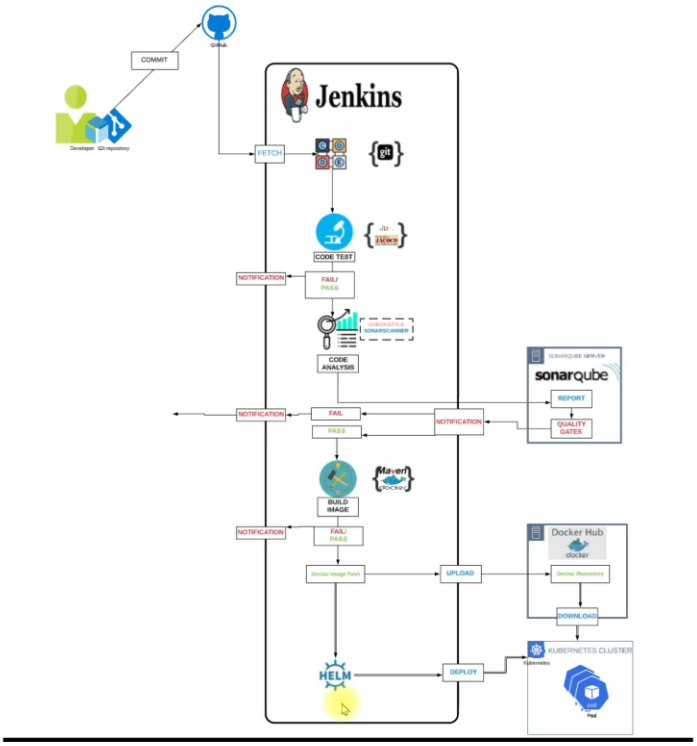
CICD with docker and Kubernetes



This is a CICD pipeline project, when a Developer make a code change and commit to git hub, Jenkins Pipeline will get triggered (Jenkins poll for GitHub changes every minute) and pull the application source code, Docker file, helm charts from GitHub. Jenkins will perform Code Unit test, Code analysis using sonar Qube and the report will be generated. Quality Gates is configured in Sonar qube which helps to limit the pipeline flow based on the number of bugs.

If Quality gate check is passed, docker image will be built and pushed to docker hub. Following the pipeline, Helm charts will be called, It will deploy kubernetes cluster which in turn pull images from Dockerhub, launches the PODS and host the application.

FLOW OF EXECUTION

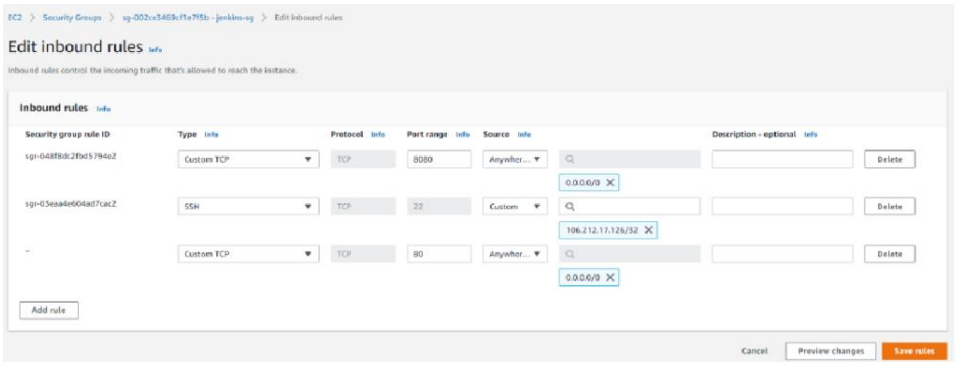
1. Continuous Integration Setup
   a. Jenkins, Sonarqube & Nexus (Continuous Integration Project)
2. Dockerhub account (Containerization Project)
3. Store Dockerhub credentials in Jenkins
4. Setup Docker Engine in Jenkins
5. Install Plugins in Jenkins
   a. Docker-pipeline
   b. Docker
   c. Pipeline utility
6. Create Kubernetes Cluster with Kops
7. Intall Helm in Kops VM
8. Create Helm Charts
9. Test Charts in K8s Cluster in test namespace..

FLOW OF EXECUTION

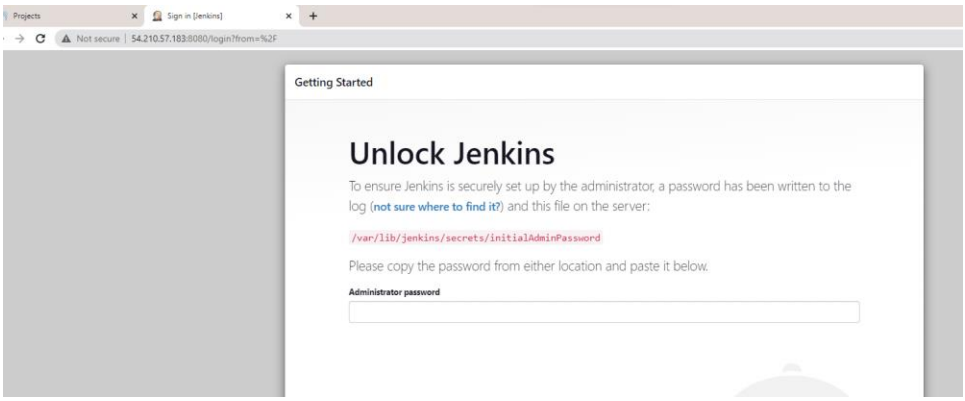
10. Add Kops VM as Jenkins Slave
11. Create Pipeline code [Declarative]
12. Update Git Reposiroty with
    a. Helm Charts
    b. Dockerfile
    c. Jenkinsfile (Pipeline code)
13. Create Jenkins job for Pipeline
14. Run & Test the job.

1. Launch the Jenkins server, ubuntu t2 small and refer user data from repo Jenkins_CI_Pipleine/jenkins-setup.sh at main · satzwebio/Jenkins_CI_Pipleine · GitHub

And update the sec group as below,



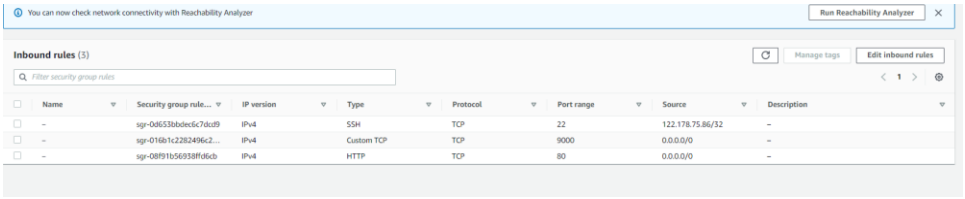
”



Get admin pwd from and login jenkins

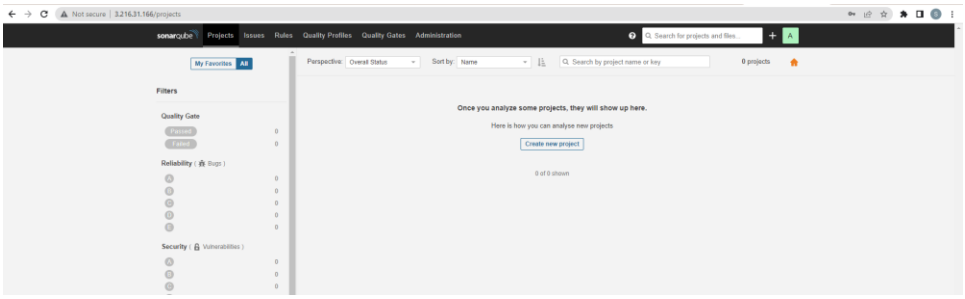
2. Launch sonar server, ubuntu t2 medium and refer user data from repo [Jenkins_CI_Pipleine/sonar-setup.sh at main · satzwebio/Jenkins_CI_Pipleine · GitHub]

Update the security groups as below,

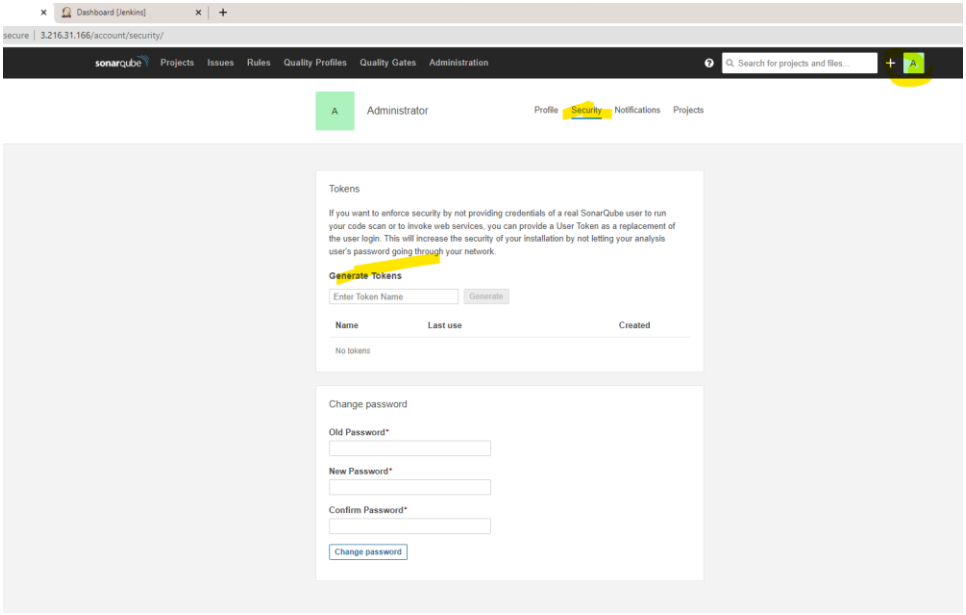


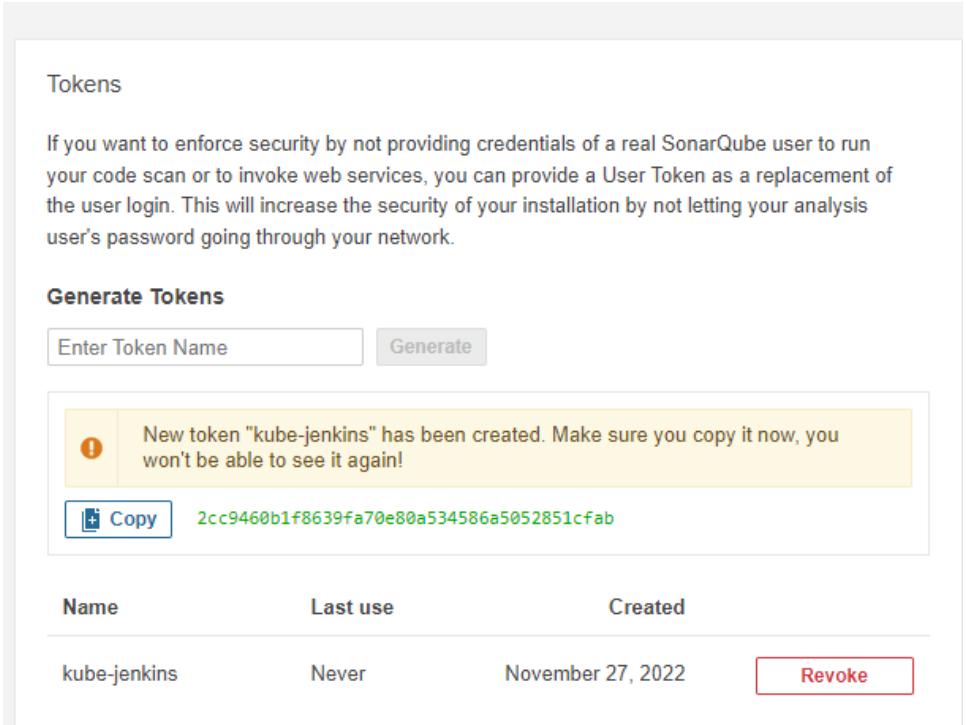
Launch instance, login ssh and check systemctl status sonarqube;

Check in port 80 or 9000, username is both admin



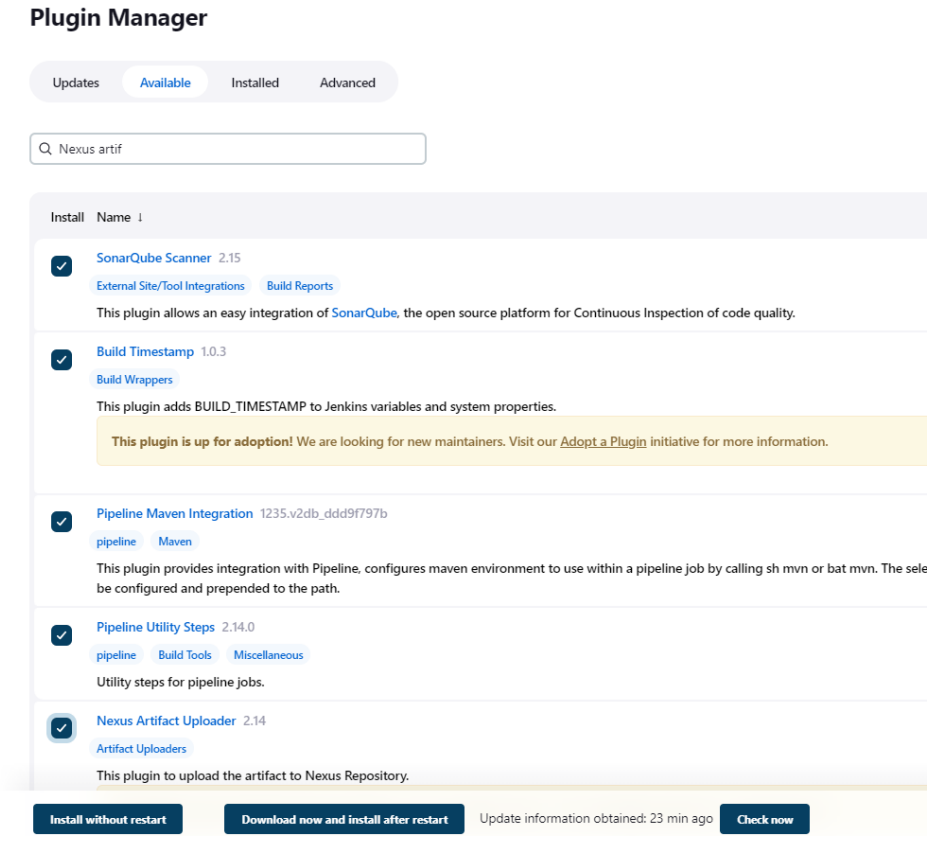
3. Generate new token in Sonar



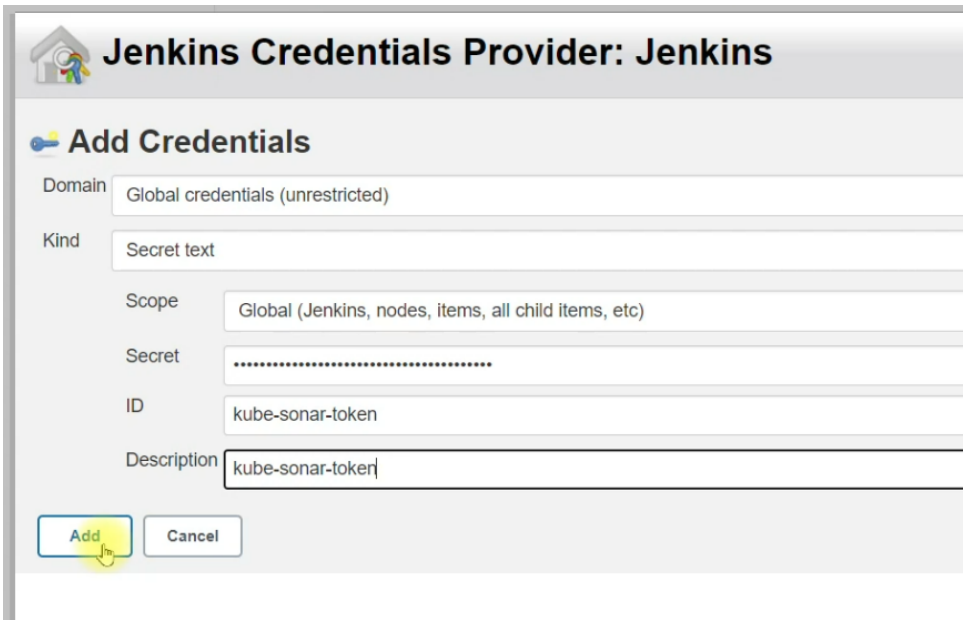


Token: 2cc9460b1f8639fa70e80a534586a5052851cfab

4. Install the below plugins on Jenkins,

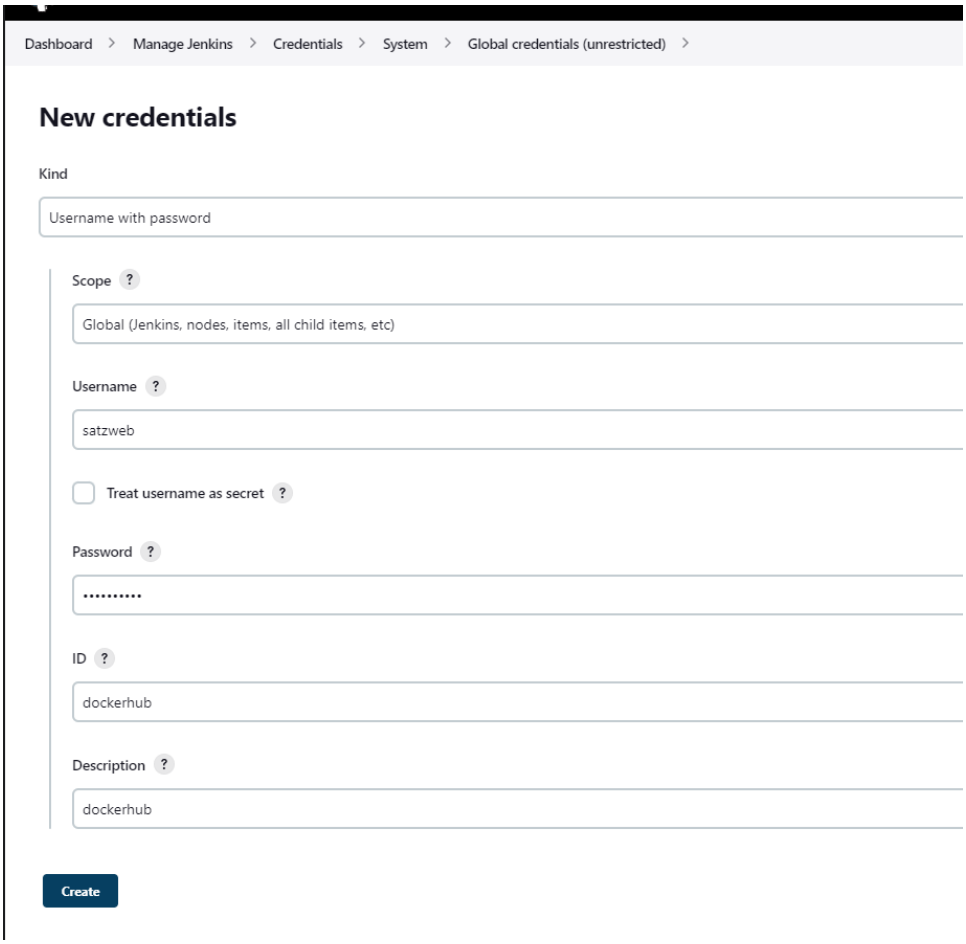


5. In Jenkins, configure sonar qube server, Mention name as sonar-pro, Enter Sonarqube private ip, add sonar qube authentication token.



6. Update All traffic from Sonar security group to Jenkins and similarly allow all traffic from Jenkins security group to Sonar.

7. Create new credentials in Jenkins, this is to add docker hub creds,



8. Install Docker engine on Jenkins, refer https://docs.docker.com/engine/install/ubuntu/

```
ubuntu@ip-172-31-53-191:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset>
     Active: active (running) since Sun 2022-11-27 10:01:52 UTC; 41s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 6758 (dockerd)
      Tasks: 9
     Memory: 28.9M
        CPU: 468ms
     CGroup: /system.slice/docker.service
             └─6758 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont>

Nov 27 10:01:51 ip-172-31-53-191 dockerd[6758]: time="2022-11-27T10:01:51.62785>
Nov 27 10:01:51 ip-172-31-53-191 dockerd[6758]: time="2022-11-27T10:01:51.62802>
Nov 27 10:01:51 ip-172-31-53-191 dockerd[6758]: time="2022-11-27T10:01:51.71630>
Nov 27 10:01:52 ip-172-31-53-191 dockerd[6758]: time="2022-11-27T10:01:52.17845>
Nov 27 10:01:52 ip-172-31-53-191 dockerd[6758]: time="2022-11-27T10:01:52.32802>
Nov 27 10:01:52 ip-172-31-53-191 dockerd[6758]: time="2022-11-27T10:01:52.39016>
Nov 27 10:01:52 ip-172-31-53-191 dockerd[6758]: time="2022-11-27T10:01:52.39034>
Nov 27 10:01:52 ip-172-31-53-191 systemd[1]: Started Docker Application Contain>
Nov 27 10:01:52 ip-172-31-53-191 dockerd[6758]: time="2022-11-27T10:01:52.42771>
```

9. As a root add Jenkins user to the Jenkins group, and then reboot the machine.

```
ubuntu@ip-172-31-53-191:~$ sudo -i
root@ip-172-31-53-191:~# su - jenkins
jenkins@ip-172-31-53-191:~$ docker images
Got permission denied while trying to connect to the Docker daemon socket at uni
x:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images/js
on": dial unix /var/run/docker.sock: connect: permission denied
jenkins@ip-172-31-53-191:~$ exit
logout
root@ip-172-31-53-191:~# usermod -aG docker jenkins
root@ip-172-31-53-191:~# id jenkins
uid=114(jenkins) gid=121(jenkins) groups=121(jenkins),999(docker)
root@ip-172-31-53-191:~# su - jenkins
jenkins@ip-172-31-53-191:~$ Docker images
Command 'Docker' not found, did you mean:
  command 'docker' from snap docker (20.10.17)
  command 'docker' from deb docker.io (20.10.12-0ubuntu4)
  command 'docker' from deb podman-docker (3.4.4+ds1-1ubuntu1)
See 'snap info <snapname>' for additional versions.
jenkins@ip-172-31-53-191:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
hello-world   latest    feb5d9fea6a5   14 months ago   13.3kB
jenkins@ip-172-31-53-191:~$
```

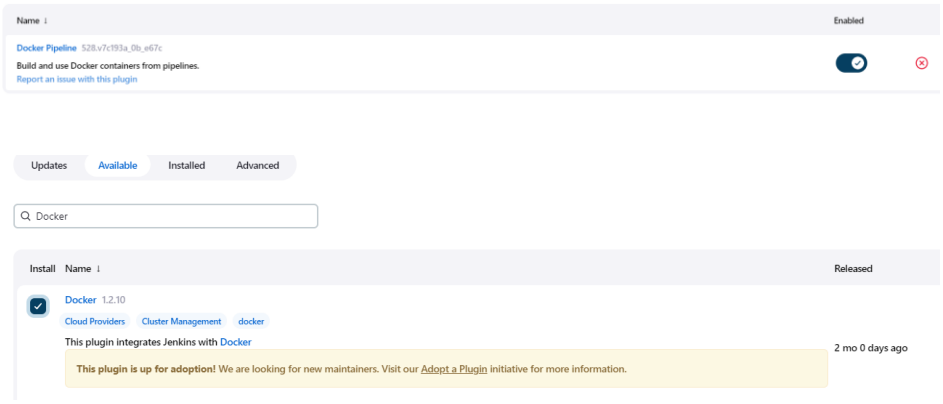
```
logout
root@ip-172-31-53-191:~# reboot
root@ip-172-31-53-191:~# Connection to 54.210.57.183 closed by remote host.
Connection to 54.210.57.183 closed.

satzw@LAPTOP-C4RG1671 MINGW64 ~/Downloads
$
```

10. Install Below plugin on Jenkins, and install without restart

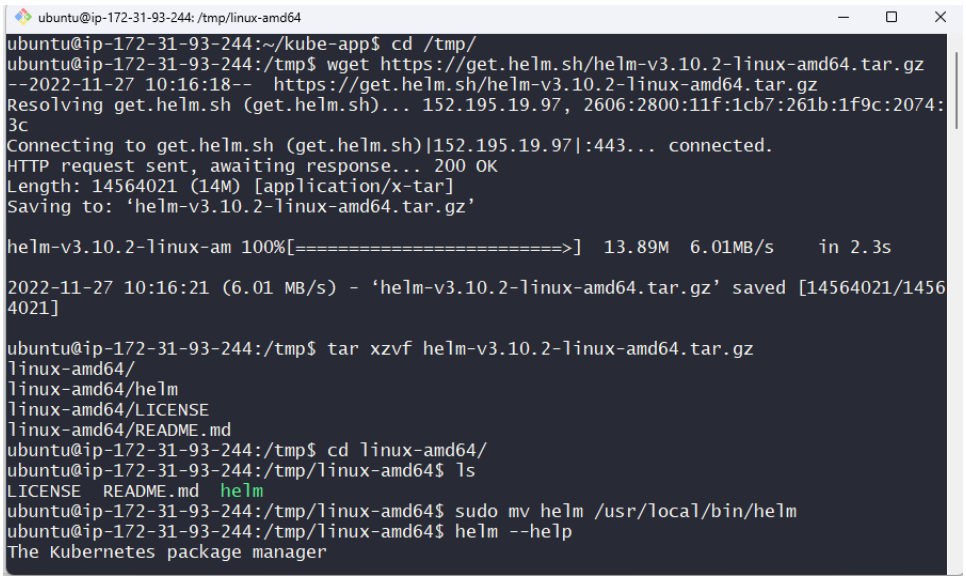
    Docker

    Docker Pipleine



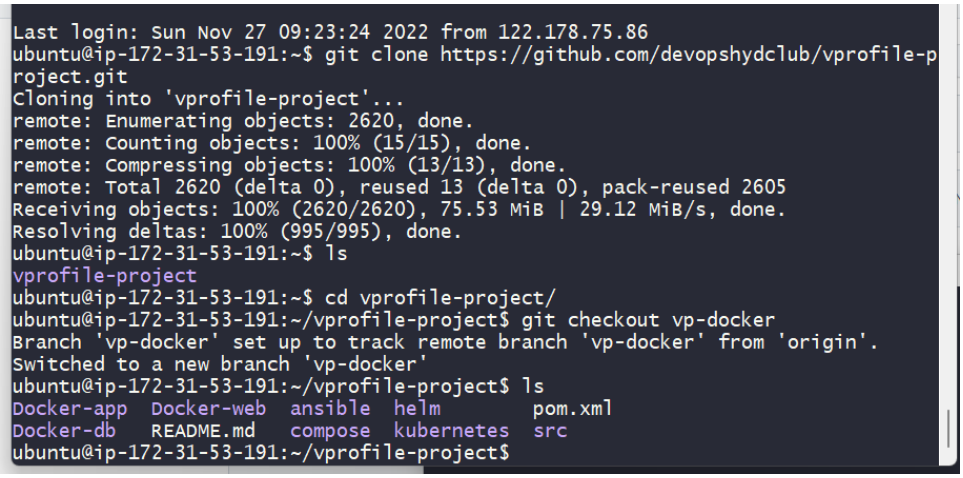
11. In KOPS VM download HELM

Get the coplink from https://github.com/helm/helm/releases. Check the os Linux amd64

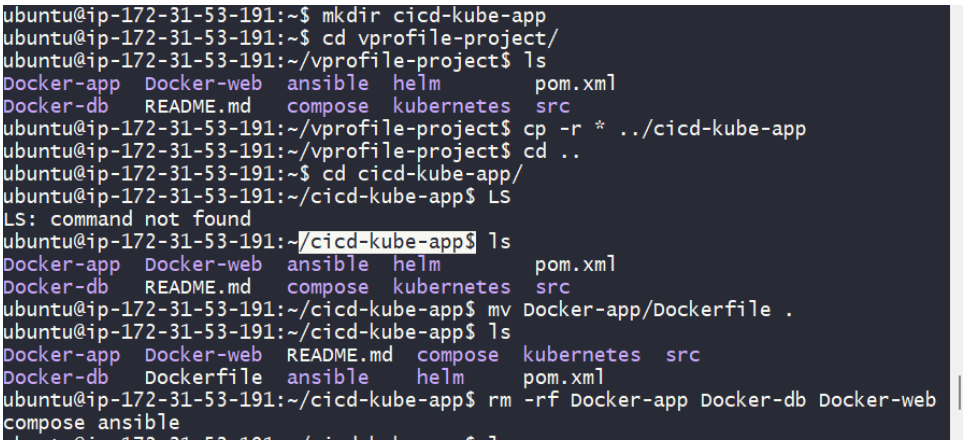


From here the steps are option. YOu can clone the https://github.com/imranvisualpath/cicd-kube-docker.git

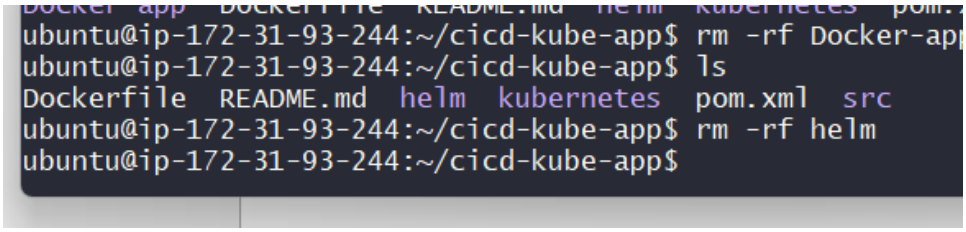
12. Get into KOPS Vm, and clone the repository



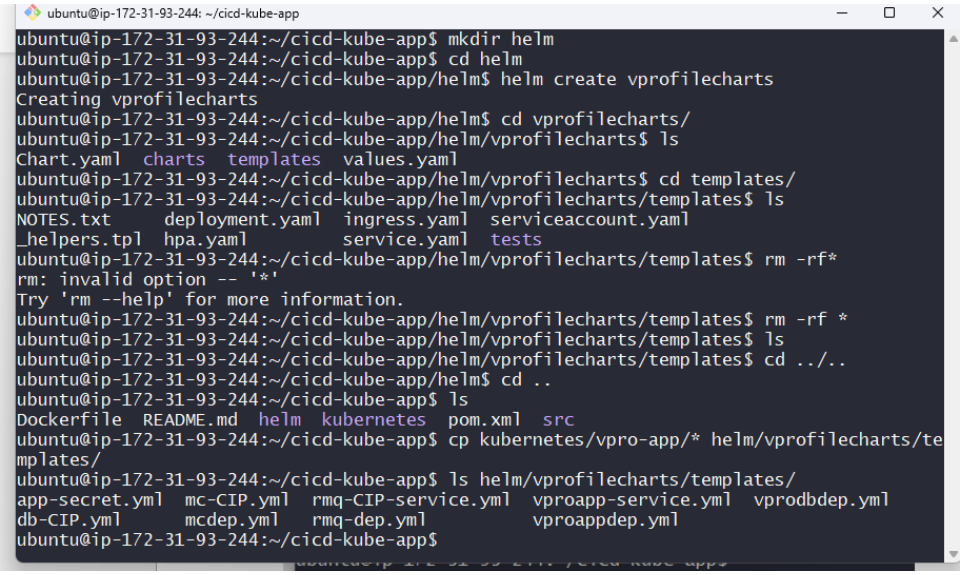
13. Copy all the files from vprofile to Cicd directory created on home, remove unwanted files.



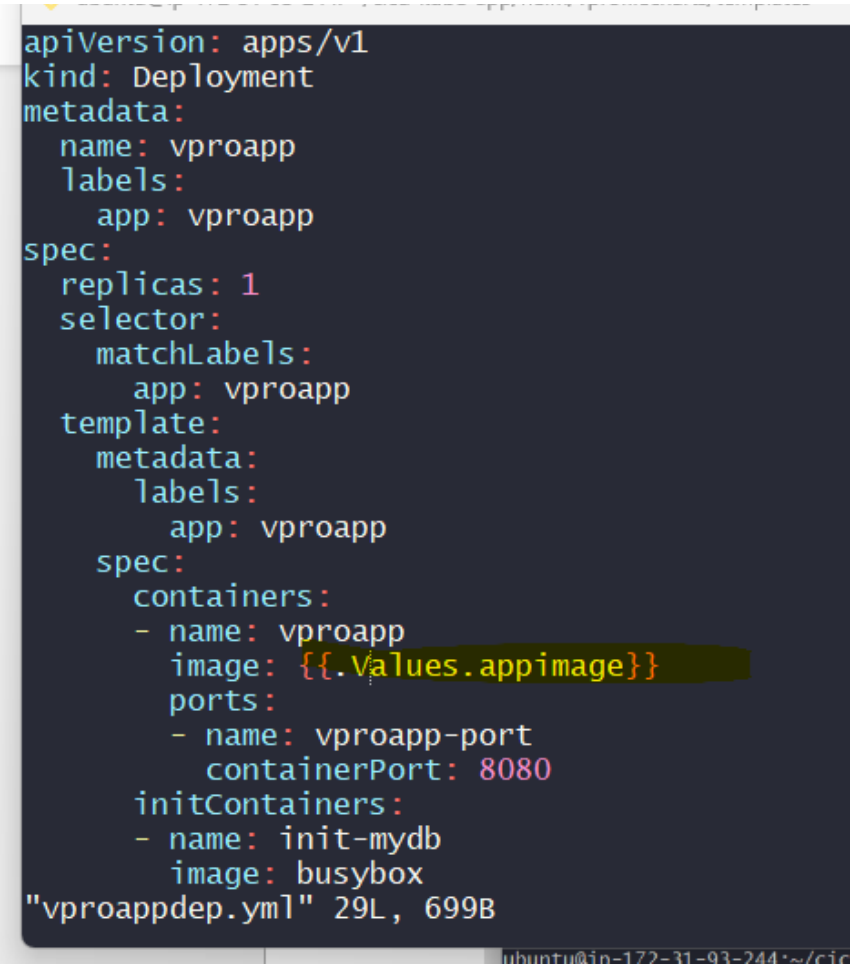
Remove Helm directoryas well.



14. Create a helm directory and create helm charts. Then move all the deployment, service files to helm/templates directory.



Vim Vproappdep.yml



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vproapp
  labels:
    app: vproapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: vproapp
  template:
    metadata:
      labels:
        app: vproapp
    spec:
      containers:
      - name: vproapp
        image: {{.Values.appimage}}
        ports:
        - name: vproapp-port
          containerPort: 8080
      initContainers:
      - name: init-mydb
        image: busybox
"vproappdep.yml" 29L, 699B
```

Just for testing we created a test namespace and launched the stacks. It creates deployment, service stc...

```
ubuntu@ip-172-31-93-244:~/cicd-kube-app$ helm install --namespace test vprofile-stack he
lm/vprofilecharts --set appimage=imranvisualpath/vproappdock:9
NAME: vprofile-stack
LAST DEPLOYED: Sun Nov 27 11:15:04 2022
NAMESPACE: test
STATUS: deployed
REVISION: 1
TEST SUITE: None
ubuntu@ip-172-31-93-244:~/cicd-kube-app$ helm list --namespace test
NAME            NAMESPACE       REVISION        UPDATED                                 S
TATUS   CHART                   APP VERSION
vprofile-stack  test            1               2022-11-27 11:15:04.286339778 +0000 UTCd
eployed vprofilecharts-0.1.0    1.16.0
ubuntu@ip-172-31-93-244:~/cicd-kube-app$ kubectl get all --namespace test
NAME                            READY   STATUS    RESTARTS   AGE
pod/vproapp-5d56b6469d-lkcsc    1/1     Running   0          42s
pod/vprodb-77668447fc-jvlpw     1/1     Running   0          42s
pod/vpromc-7db9bfbd6d-cthkd     1/1     Running   0          42s
pod/vpromq01-5bd75bf4bc-p6jf5   1/1     Running   0          42s

NAME                     TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        A
GE
service/vproapp-service   LoadBalancer   100.70.207.153   <pending>     80:31312/TCP   4
```

To uninstall the stacks,

```
ubuntu@ip-172-31-93-244:~/cicd-kube-app$ helm delete vprofile-stack --namespace test
release "vprofile-stack" uninstalled
ubuntu@ip-172-31-93-244:~/cicd-kube-app$
```

Create a prod namespace:

```
ubuntu@ip-172-31-93-244:~/cicd-kube-app$ kubectl create namespace prod
namespace/prod created
ubuntu@ip-172-31-93-244:~/cicd-kube-app$
```

Create jenkinsfile

Refer our repo : https://github.com/satzwebio/cicd-kube-docker.git

In KOPS VM, create jenkins-slave directory and install jdk

```
ubuntu@ip-172-31-32-194:~$ mkdir jenkins-slave
ubuntu@ip-172-31-32-194:~$ sudo apt install openjdk-8-jdk -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
openjdk-8-jdk is already the newest version (8u272-b10-0ubuntu1~20.04).
0 upgraded, 0 newly installed, 0 to remove and 59 not upgraded.
ubuntu@ip-172-31-32-194:~$
```

In addition perfrom below,

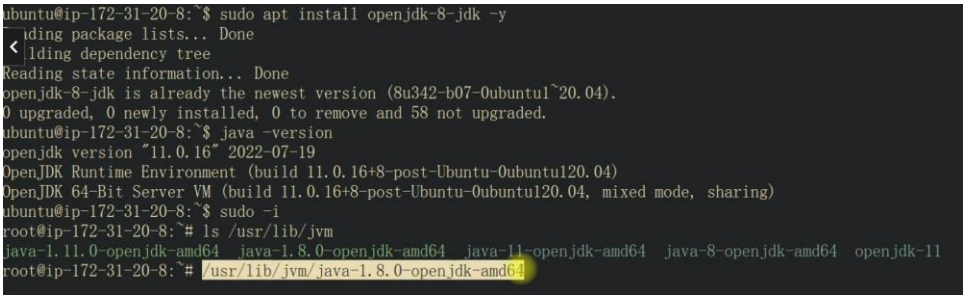
```
sudo apt remove openjdk-8-jdk -y

sudo apt purge openjdk-8-jdk -y
```

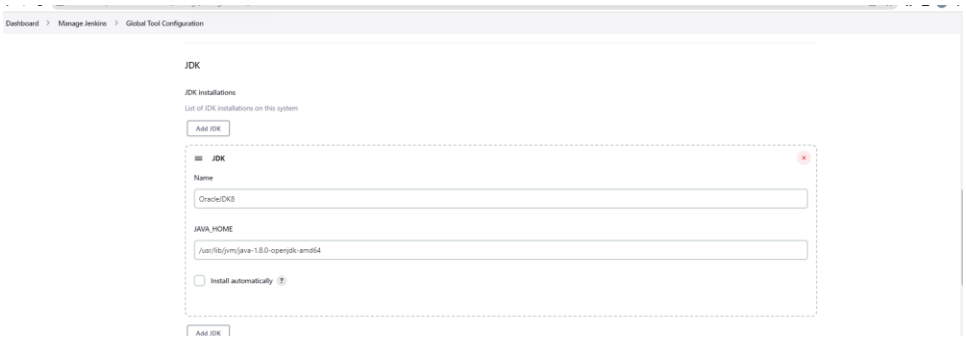
```
sudo apt update

sudo apt install openjdk-11-jdk -y
```

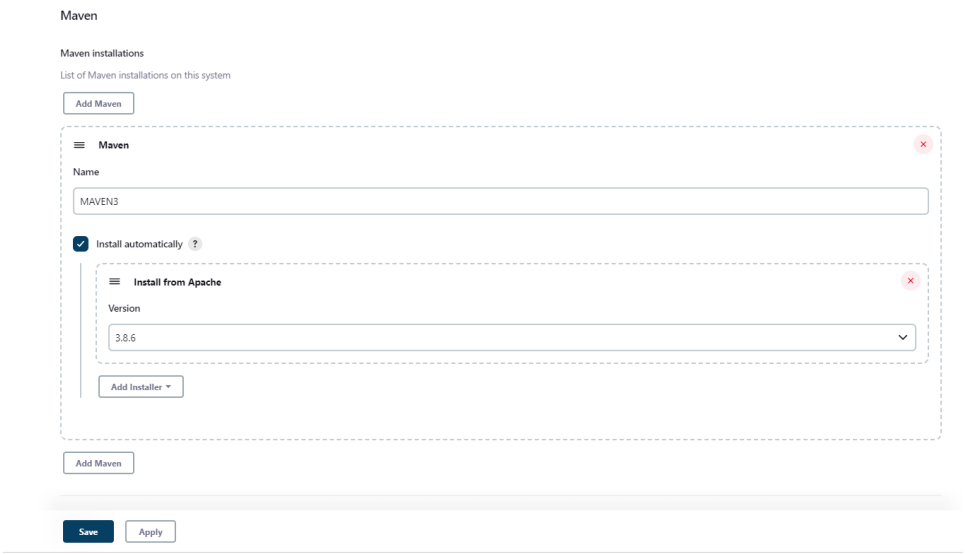
One more Java package need to be installed for Jenkins to access



And specify the path in Jenkins,



Then add MAVEN,



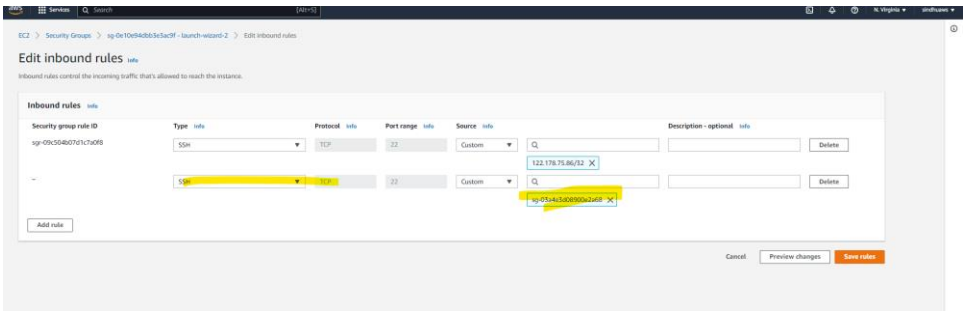
Create opt/jenkins-slave directory and give ubuntu permission.

So jenkins master will have an agent in this directory.

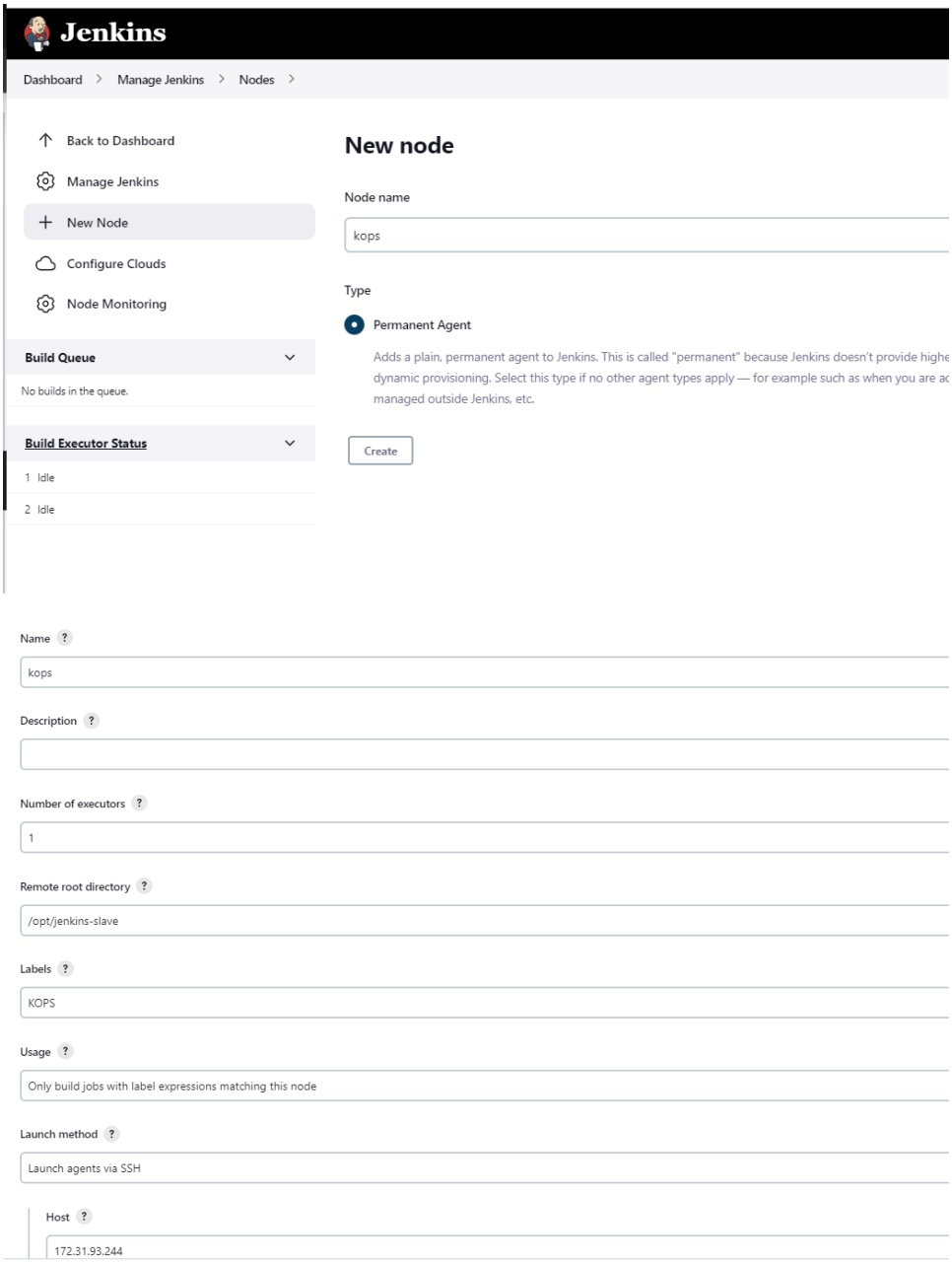
Jenkins will ssh into this VM with ubuntu user, so ubuntu should own this directory

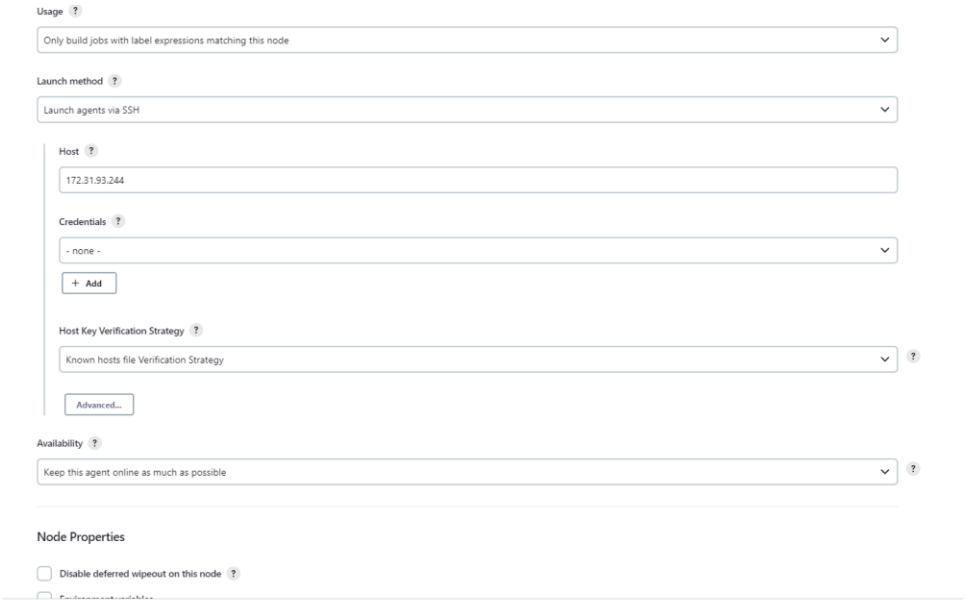
```
ubuntu@ip-172-31-93-244:~$ sudo mkdir /opt/jenkins-slave
ubuntu@ip-172-31-93-244:~$ sudo chown ubuntu.ubuntu /opt/jenkins-slave -R
```

Update security group, allow ssh from Jenkins to KOPS vm,



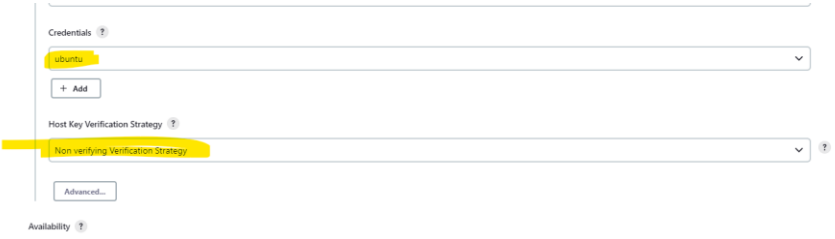
In Jenkins, create a new node



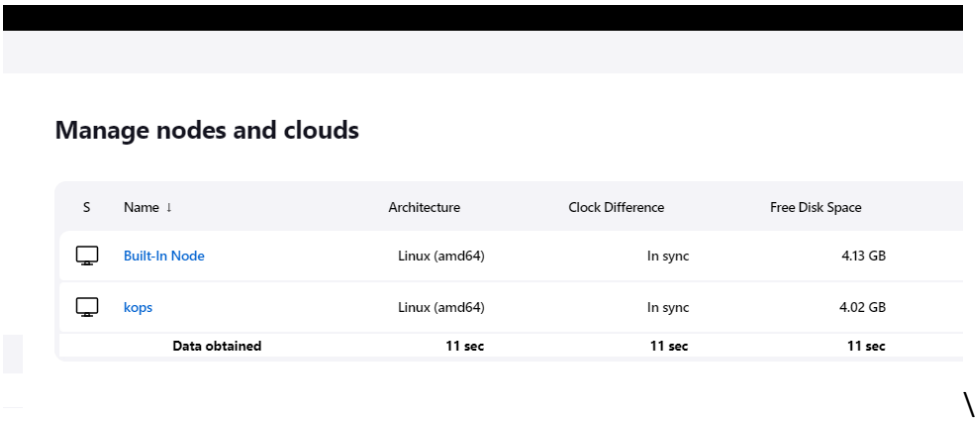


Add credentials, enter KOPS private key

Select the cred and host verification strategy

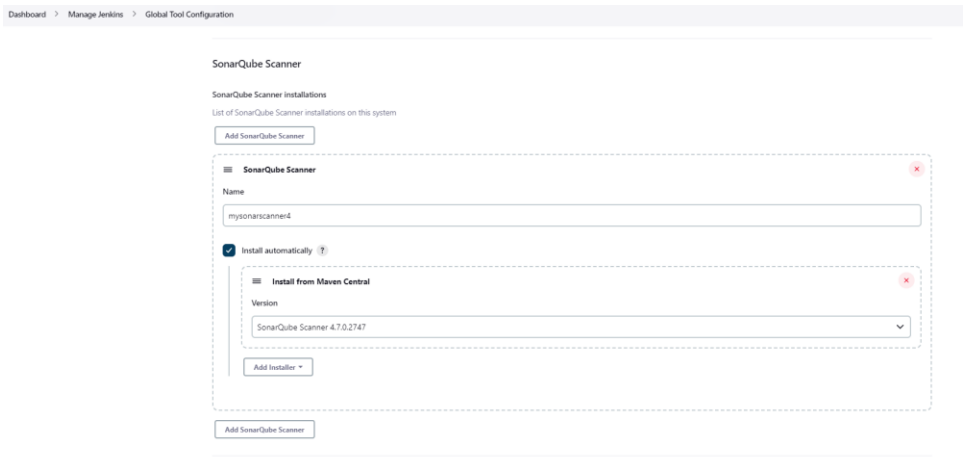


Agent should be connected successfully,

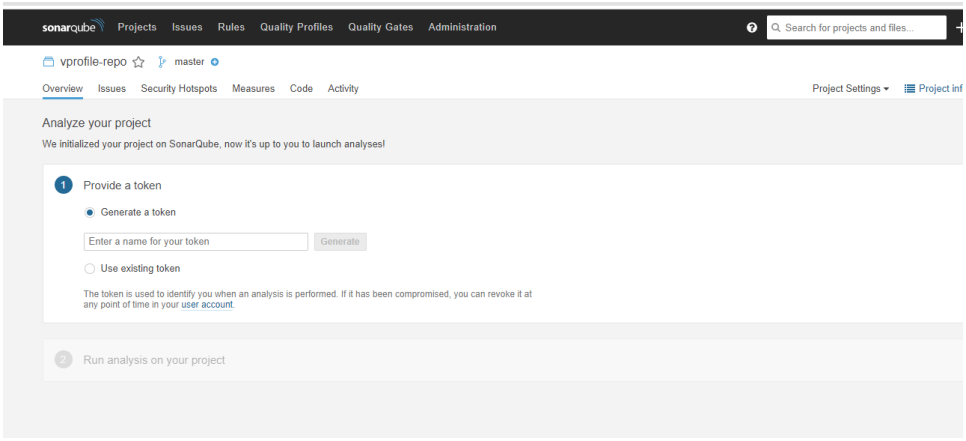


\

In Jenkins add mysonarscanner4 under sonar qube scanner, since we have specified this name in jenkins file,

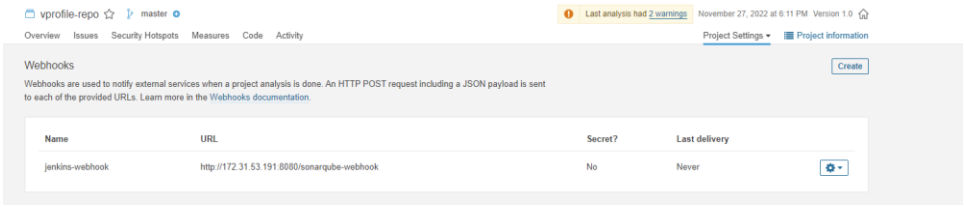


Create a new project in sonar qube,

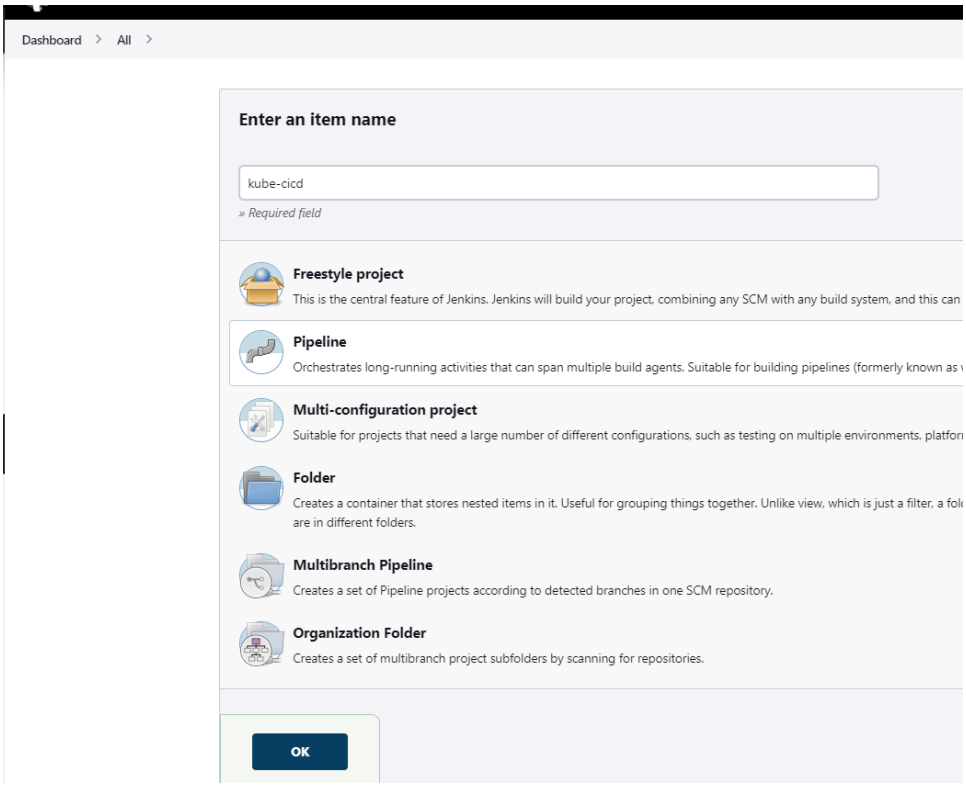


In project setting under webhook, create below..

Provide private ip of Jenkins.

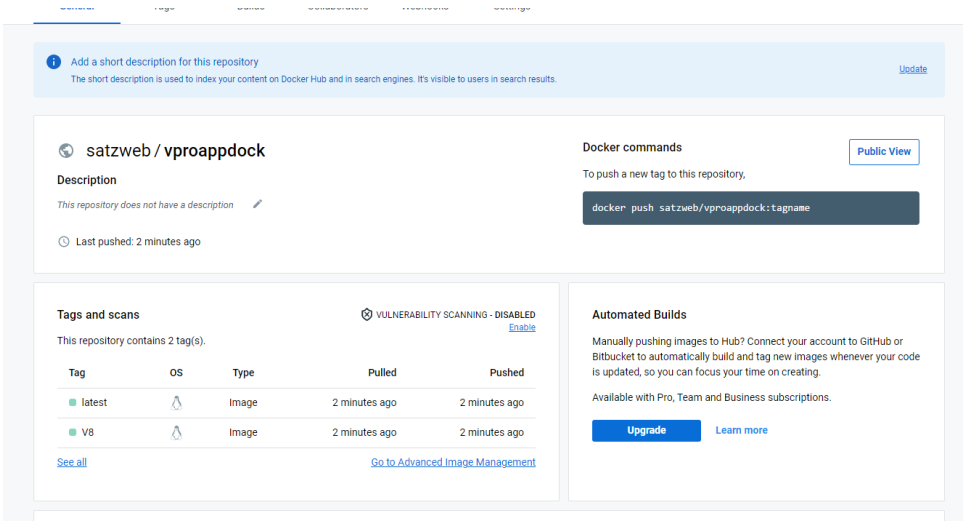


Create a pipeline project in jenkins,



Pipeline finished successfully.

In the docker hub, we can see the images,

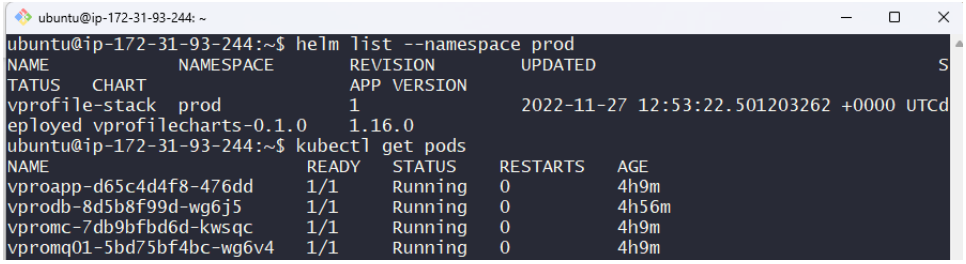


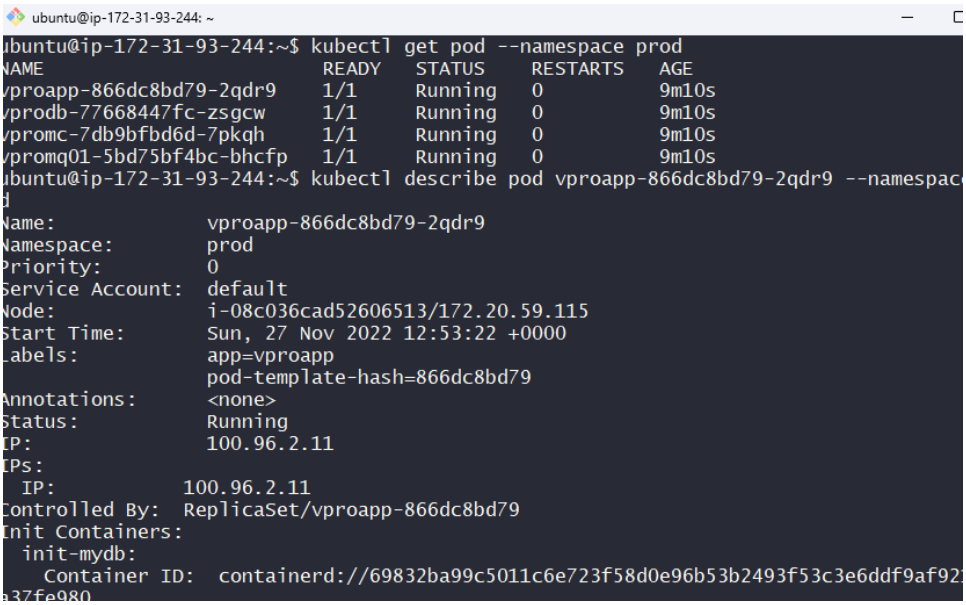
We can see the unused images are removed from the docker hub

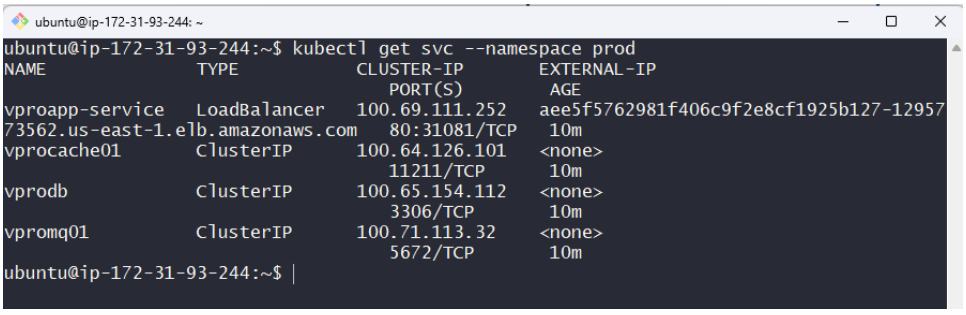
```
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Remove Unused docker image)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ docker rmi satzweb/vproappdock:V8
Untagged: satzweb/vproappdock:V8
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Kubernetes Deploy)
[Pipeline] node
Running on kops in /opt/jenkins-slave/workspace/kube-cicd
[Pipeline] {
[Pipeline] checkout
```

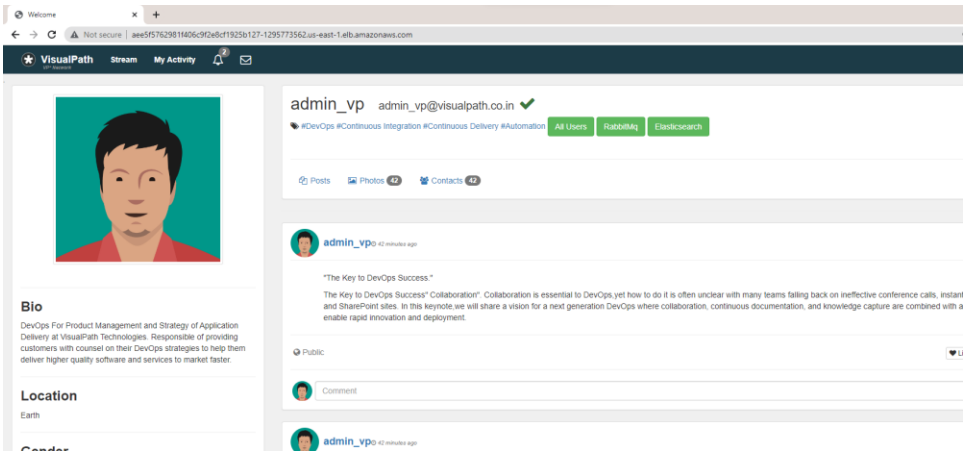
Can see the helm is installing the required stack,

```
[Pipeline] sh
+ helm upgrade --install --force vprofile-stack helm/vprofilecharts --set appimage=satzweb/vproappdock:V8 --namespace prod
Release "vprofile-stack" does not exist. Installing it now.
NAME: vprofile-stack
LAST DEPLOYED: Sun Nov 27 12:53:22 2022
NAMESPACE: prod
STATUS: deployed
REVISION: 1
TEST SUITE: None
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
```

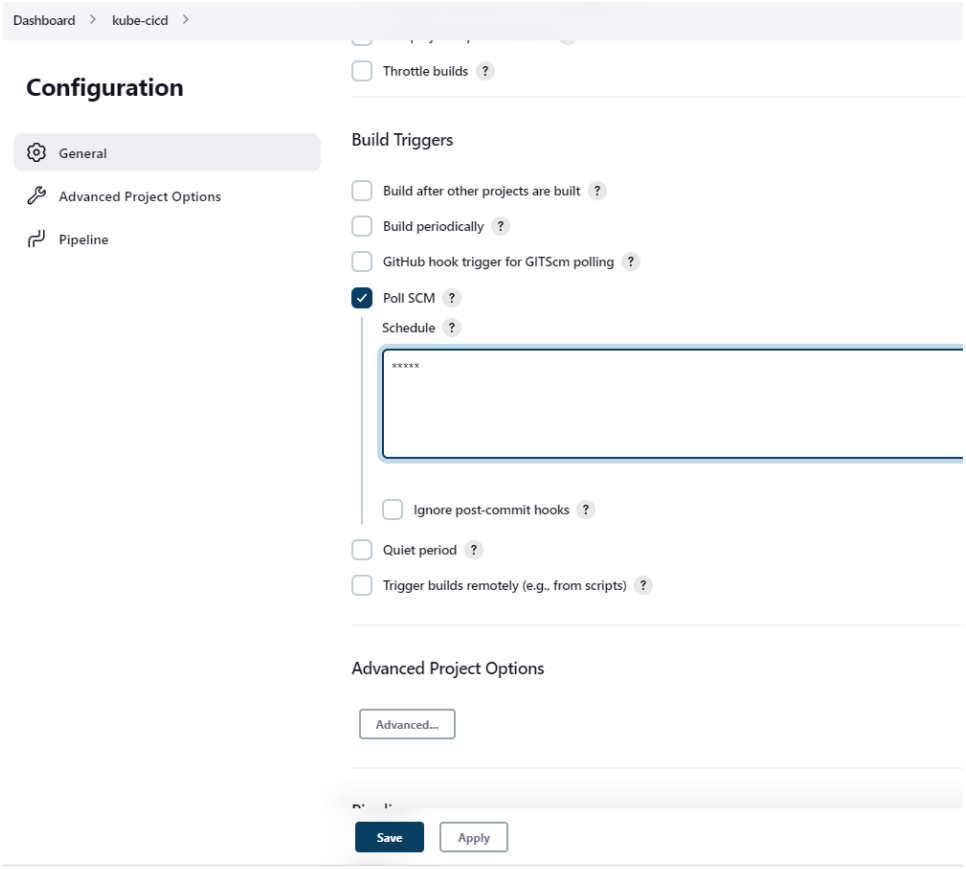

ubuntu@ip-172-31-93-244: ~
ubuntu@ip-172-31-93-244:~$ helm list --namespace prod
NAME NAMESPACE REVISION UPDATED S
TATUS CHART APP VERSION
vprofile-stack prod 1 2022-11-27 12:53:22.501203262 +0000 UTCd
eployed vprofilecharts-0.1.0 1.16.0
ubuntu@ip-172-31-93-244:~$ kubectl get pods
NAME READY STATUS RESTARTS AGE
vproapp-d65c4d4f8-476dd 1/1 Running 0 4h9m
vprodb-8d5b8f99d-wg6j5 1/1 Running 0 4h56m
vpromc-7db9bfbd6d-kwsqc 1/1 Running 0 4h9m
vpromq01-5bd75bf4bc-wg6v4 1/1 Running 0 4h9m


ubuntu@ip-172-31-93-244: ~
ubuntu@ip-172-31-93-244:~$ kubectl get pod --namespace prod
NAME READY STATUS RESTARTS AGE
vproapp-866dc8bd79-2qdr9 1/1 Running 0 9m10s
vprodb-77668447fc-zsgcw 1/1 Running 0 9m10s
vpromc-7db9bfbd6d-7pkqh 1/1 Running 0 9m10s
vpromq01-5bd75bf4bc-bhcfp 1/1 Running 0 9m10s
ubuntu@ip-172-31-93-244:~$ kubectl describe pod vproapp-866dc8bd79-2qdr9 --namespac
e
Name: vproapp-866dc8bd79-2qdr9
Namespace: prod
Priority: 0
Service Account: default
Node: i-08c036cad52606513/172.20.59.115
Start Time: Sun, 27 Nov 2022 12:53:22 +0000
Labels: app=vproapp
pod-template-hash=866dc8bd79
Annotations: <none>
Status: Running
IP: 100.96.2.11
IPs:
IP: 100.96.2.11
Controlled By: ReplicaSet/vproapp-866dc8bd79
Init Containers:
init-mydb:
Container ID: containerd://69832ba99c5011c6e723f58d0e96b53b2493f53c3e6ddf9af92
a37fe980


ubuntu@ip-172-31-93-244: ~
ubuntu@ip-172-31-93-244:~$ kubectl get svc --namespace prod
NAME TYPE CLUSTER-IP EXTERNAL-IP
PORT(S) AGE
vproapp-service LoadBalancer 100.69.111.252 aee5f5762981f406c9f2e8cf1925b127-12957
73562.us-east-1.elb.amazonaws.com 80:31081/TCP 10m
vprocache01 ClusterIP 100.64.126.101 <none>
11211/TCP 10m
vprodb ClusterIP 100.65.154.112 <none>
3306/TCP 10m
vpromq01 ClusterIP 100.71.113.32 <none>
5672/TCP 10m
ubuntu@ip-172-31-93-244:~$



Instead of Build manually in jenkins, we can update it to Poll SCM, so whenever a change in git repo, a new build will get triggered and deploy the stack to kuberntes.




ubuntu@ip-172-31-93-244: ~
ubuntu@ip-172-31-93-244:~$ kops delete cluster --name kops.satzwebio.com --state=s3://satz-kops-bucket --yes
W1127 13:11:06.361863   10850 aws.go:2250] (new) cluster tag not found on volume:vol-039fe83f2eed5ccf1
TYPE   NAME   ID
autoscaling-config   master-us-east-1a.masters.kops.satzwebio.com   lt-02d9f92858d24cbf8
autoscaling-config   nodes-us-east-1a.kops.satzwebio.com   lt-04a792daf8cd40a41
autoscaling-group   master-us-east-1a.masters.kops.satzwebio.com   master-us-east-1a.masters.kops.satzwebio.com
autoscaling-group   nodes-us-east-1a.kops.satzwebio.com   nodes-us-east-1a.kops.satzwebio.com
dhcp-options   kops.satzwebio.com   dopt-05513f795f4373891