Step 1:

Build nexus application to store our build image.

We are creating nexus app using Kubernetes cluster.

First we need to setup Kubernetes cluster. We are using terraform code build EKS cluster.

<https://github.com/ksnithya/nexus.git>

Git clone https://github.com/ksnithya/nexus.git

cd nexus

terraform init

terraform plan

terraform apply

Step 2:

We are creating “nexus app” using eks cluster.

Under same repo code available under “eks1” directory.

cd eks1

kubectl create -f .

[ec2-user@ip-172-31-42-60 eks1]$ kubectl get deploy

NAME READY UP-TO-DATE AVAILABLE AGE

nexus 1/1 1 1 5m19s

[ec2-user@ip-172-31-42-60 eks1]$ kubectl get pod

NAME READY STATUS RESTARTS AGE

nexus-6f9c695488-f7ctp 1/1 Running 0 5m27s

[ec2-user@ip-172-31-42-60 eks1]$ kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 172.20.0.1 <none> 443/TCP 82m

nexus-load LoadBalancer 172.20.46.174 a4511f8cac2974ea4996b8455e65d17d-2136990638.ap-south-1.elb.amazonaws.com 8081:32229/TCP 57m

nexus-repo LoadBalancer 172.20.231.31 a9be02f4ce16a49139d04fb31af92eff-688228379.ap-south-1.elb.amazonaws.com 8082:30830/TCP 50m

[ec2-user@ip-172-31-42-60 eks1]$ kubectl get pvc

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

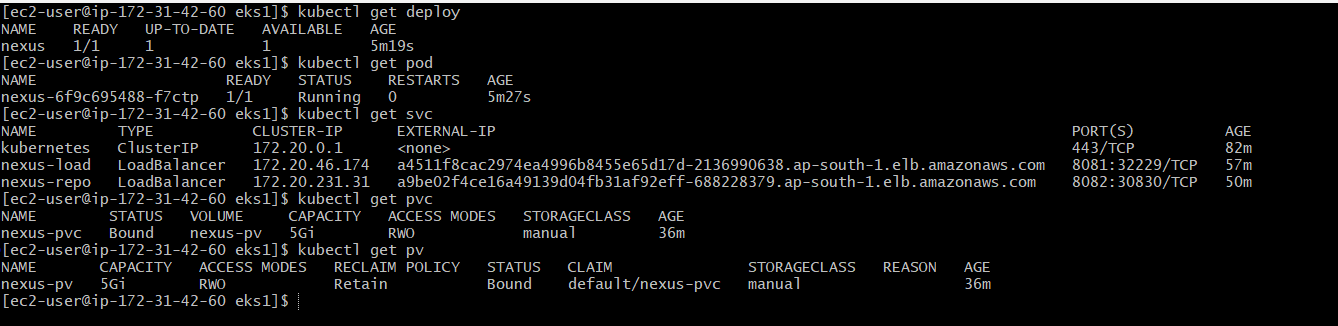
nexus-pvc Bound nexus-pv 5Gi RWO manual 36m

[ec2-user@ip-172-31-42-60 eks1]$ kubectl get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE

nexus-pv 5Gi RWO Retain Bound default/nexus-pvc manual 36m

[ec2-user@ip-172-31-42-60 eks1]$



Now we can connect to nexus app using belo link.

http://<DNS name of svc>:8081

For nexus3 user name is admin and password will be available under “cat /nexus-data/admin.password”. For nexus2 user name is ‘admin’ and password is ‘admin123’.

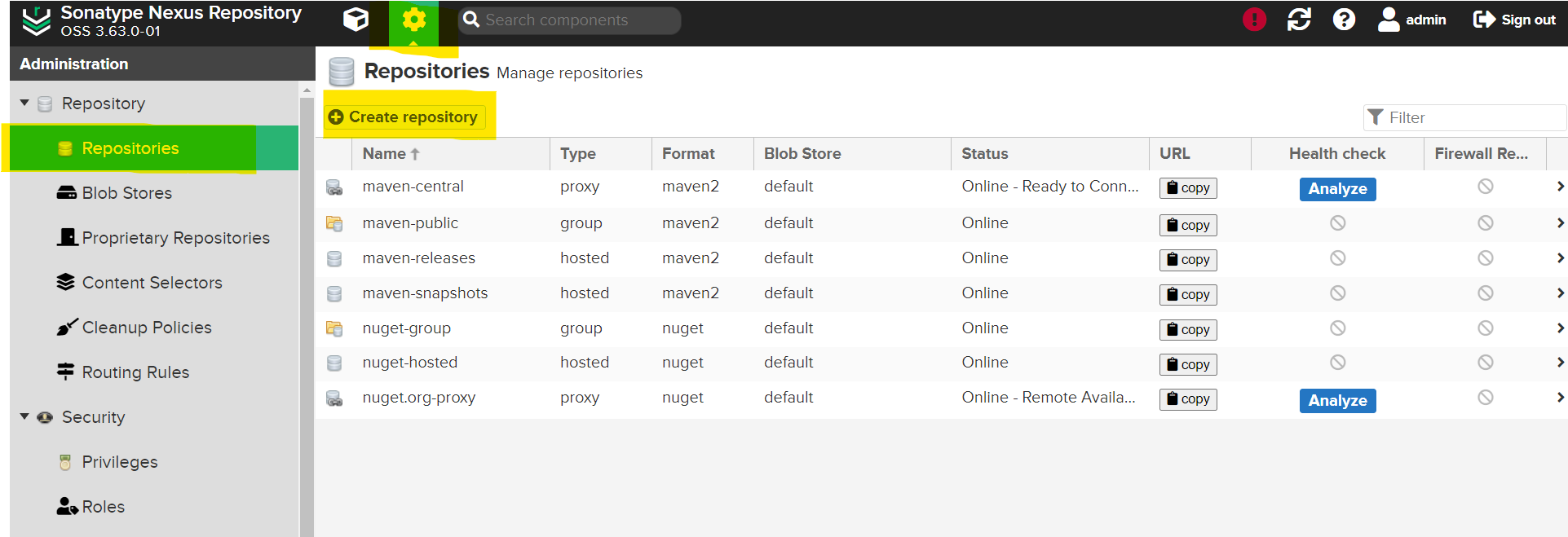
A screenshot of a computer

Description automatically generated

Step 3:

Now we need to enable docker host repository to push our docker image.

Click on “Server admin and configuration” -> repository -> create repository



Now click on “Docker Hosted”

A white and grey line

Description automatically generated with medium confidence

Give repository name and enable http connection. You can give any port number.

A screenshot of a computer

Description automatically generated

Step 4:

If we want to connect to this repository using http port we need to add that service to our deployment.

Create below service.

[ec2-user@ip-172-31-42-60 eks]$ cat service1.yml

apiVersion: v1

kind: Service

metadata:

name: nexus-repo

selector:

app: nexus-server

type: LoadBalancer

ports:

- port: 8082

targetPort: 8082

nodePort: 32001

[ec2-user@ip-172-31-42-60 eks]$ kubectl create -f service1.yml

[ec2-user@ip-172-31-42-60 eks1]$ kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 172.20.0.1 <none> 443/TCP 82m

nexus-load LoadBalancer 172.20.46.174 a4511f8cac2974ea4996b8455e65d17d-2136990638.ap-south-1.elb.amazonaws.com 8081:32229/TCP 57m

nexus-repo LoadBalancer 172.20.231.31 a9be02f4ce16a49139d04fb31af92eff-688228379.ap-south-1.elb.amazonaws.com 8082:30830/TCP 50m

Step 5:

We need to login to our nexus repo from our docker server.

Ex: docker login <dns name of service of repo service>:8082

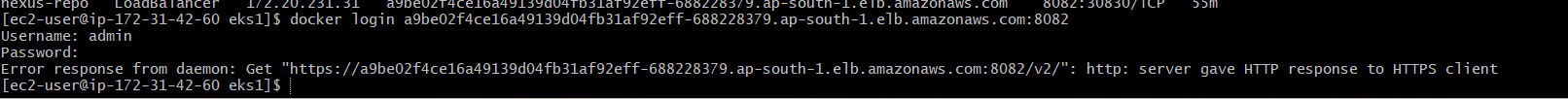
[ec2-user@ip-172-31-42-60 eks1]$ docker login a9be02f4ce16a49139d04fb31af92eff-688228379.ap-south-1.elb.amazonaws.com:8082

Username: admin

Password:

Error response from daemon: Get "https://a9be02f4ce16a49139d04fb31af92eff-688228379.ap-south-1.elb.amazonaws.com:8082/v2/": http: server gave HTTP response to HTTPS client

[ec2-user@ip-172-31-42-60 eks1]$



We will get above error. To fix this error we need to add our nexus registry details to docker deamon file.

Adding

{ "insecure-registries":["host:port"] } – [<ip of workednode cluster server>:port number of new service>]

Ex: { "insecure-registries":["13.201.134.92:32001", "13.233.38.79:32001"] }

to

/etc/docker/daemon.json

Then to below file

/etc/default/docker

and put the line

DOCKER\_OPTS="--config-file=/etc/docker/daemon.json"

in it and then restarted the docker daemon with

sudo systemctl stop docker and sudo systemctl start docker.

Now try to login it will work.

A computer screen with text

Description automatically generated

Step 6:

Now we can move our docker image to our nexus app.

A computer screen with many white text

Description automatically generated

[ec2-user@ip-172-31-42-60 jenkins]$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

13.201.134.92:32001/my-resume v1 403b5ff47bef 4 days ago 514MB

my-resume v1 403b5ff47bef 4 days ago 514MB

<none> <none> 2ed9cf5fdaad 4 days ago 514MB

gitlab/gitlab-ce latest 140c322810df 7 days ago 2.87GB

postgres latest 391a00ec7cac 13 days ago 425MB

sonarqube latest dee4f32d6f90 2 weeks ago 725MB

**Need to tag image with our repo details. <repo ip>:<port number>/<repo name>/<image-name>:<tag>**

[ec2-user@ip-172-31-42-60 jenkins]$ docker tag my-resume:v1 a9be02f4ce16a49139d04fb31af92eff-688228379.ap-south-1.elb.amazonaws.com:8082/nx-docker/nithya-image:v1

[ec2-user@ip-172-31-42-60 jenkins]$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

13.201.134.92:32001/my-resume v1 403b5ff47bef 4 days ago 514MB

13.201.134.92:32001/nx-docker/my-resume v1 403b5ff47bef 4 days ago 514MB

my-resume v1 403b5ff47bef 4 days ago 514MB

<none> <none> 2ed9cf5fdaad 4 days ago 514MB

gitlab/gitlab-ce latest 140c322810df 7 days ago 2.87GB

postgres latest 391a00ec7cac 13 days ago 425MB

sonarqube latest dee4f32d6f90 2 weeks ago 725MB

[ec2-user@ip-172-31-42-60 jenkins]$ docker push a9be02f4ce16a49139d04fb31af92eff-688228379.ap-south-1.elb.amazonaws.com:8082/nx-docker/nithya-image:v1

The push refers to repository [13.201.134.92:32001/nx-docker/my-resume]

d2866bdaffb9: Pushed

18701473167d: Pushed

8b80f6ba0e97: Pushed

63e923e36ca6: Pushed

5f70bf18a086: Pushed

74ddd0ec08fa: Pushed

v1: digest: sha256:7602f2c4cebdd6a3415552403d840ec8b0fd6092eb99650d34dad7664bb65d12 size: 1571

Step 7:

Now we can got to our nexus app and see our repo we can see our pushed image.

Click on Browse -> our repo name

A screenshot of a computer

Description automatically generated

We can see our image.

A screenshot of a computer

Description automatically generated

[ec2-user@ip-172-31-42-60 eks1]$ docker login http://a985c9e04c0324791927c67053ee2579-34773242.ap-south-1.elb.amazonaws.com:8081/repository/nx-docker/

Username: admin

Password:

Error response from daemon: login attempt to http://a985c9e04c0324791927c67053ee2579-34773242.ap-south-1.elb.amazonaws.com:8081/v2/ failed with status: 404 Not Found

[ec2-user@ip-172-31-42-60 eks1]$

[ec2-user@ip-172-31-42-60 eks1]$ docker login http://a985c9e04c0324791927c67053ee2579-34773242.ap-south-1.elb.amazonaws.com:8082/repository/nx-docker/

Username: admin

Password:

Error response from daemon: Get "http://a985c9e04c0324791927c67053ee2579-34773242.ap-south-1.elb.amazonaws.com:8082/v2/": net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)

[ec2-user@ip-172-31-42-60 eks1]$