Steps to create ECR, how to build docker image and push to ECR, How to build EKS cluster and deploy applications, Also how to build same application using Helm chart and Jenkins pipeline.

Step 1:

Create EC2 instance.

Then install docker, aws cli, git

Follow steps in below link to install docker in Amazon Linux.

https://www.cyberciti.biz/faq/how-to-install-docker-on-amazon-linux-2/

To install git:

Sudo yum install git -y

To install aws cli:

Follow below link:

https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

To fix the path issue:

```
[ec2-user@ip-172-31-42-60 ~]$ aw --version
-bash: aw: command not found
```

https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-troubleshooting.html#tshoot-install-not-found

Step 2:

Now clone the repo where we have our dockerfile.

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

```
[ec2-user@ip-172-31-42-60 docker]$ git clone
https://github.com/ksnithya/jenkins.git
Cloning into 'jenkins'...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (109/109), done.
remote: Total 111 (delta 39), reused 3 (delta 0), pack-reused
0
Receiving objects: 100% (111/111), 25.16 KiB | 5.03 MiB/s,
done.
Resolving deltas: 100% (39/39), done.

[ec2-user@ip-172-31-42-60 docker]$ ls -l
total 0
drwxr-xr-x. 4 ec2-user ec2-user 120 Dec 16 11:57 jenkins

[ec2-user@ip-172-31-42-60 docker]$ cd jenkins/
[ec2-user@ip-172-31-42-60 jenkins]$ ls -l
```

```
total 16
-rw-r--r--. 1 ec2-user ec2-user  129 Dec 16 11:57 Dockerfile
-rw-r--r--. 1 ec2-user ec2-user 2628 Dec 16 11:57 Jenkinsfile
-rw-r--r--. 1 ec2-user ec2-user 2737 Dec 16 11:57 index.html
-rw-r--r--. 1 ec2-user ec2-user   55 Dec 16 11:57 sonar-
project.properties
[ec2-user@ip-172-31-42-60 jenkins]$
```

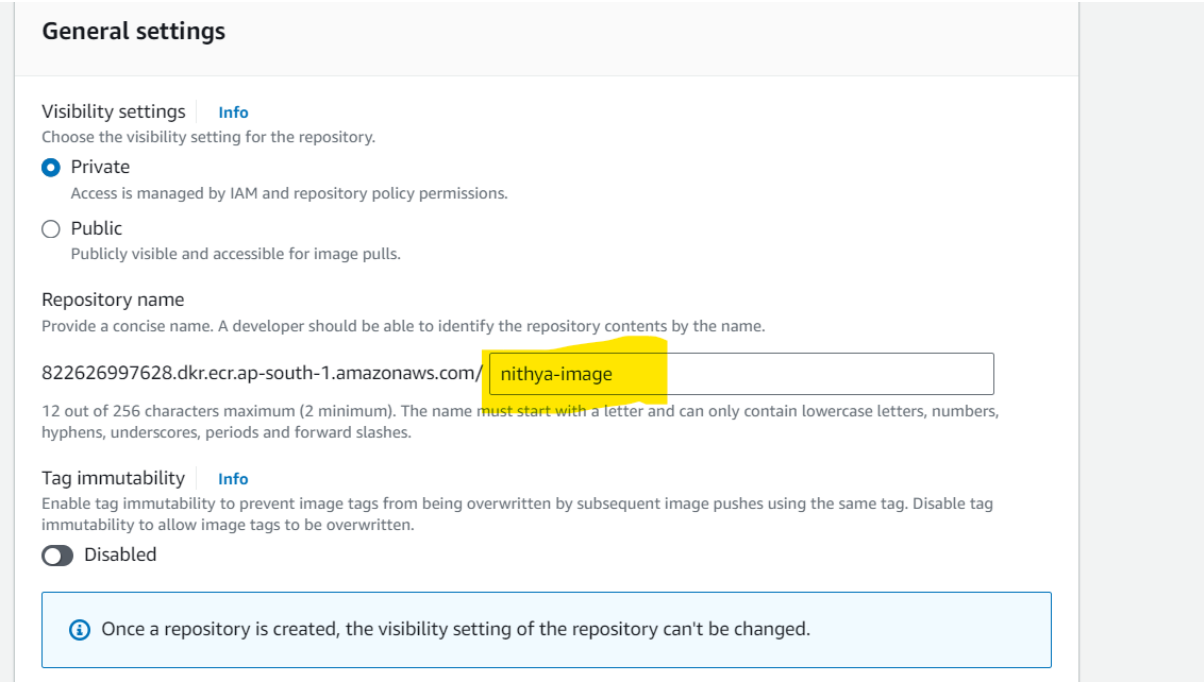
```
[ec2-user@ip-172-31-42-60 jenkins]$ docker build -t
nithya-image .
[ec2-user@ip-172-31-42-60 jenkins]$ docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
nithya-image   latest    626bd6616bf4   15 seconds ago   514MB
postgres       latest    391a00ec7cac   8 days ago       425MB
sonarqube      latest    dee4f32d6f90   2 weeks ago      725MB
```

Step 3:

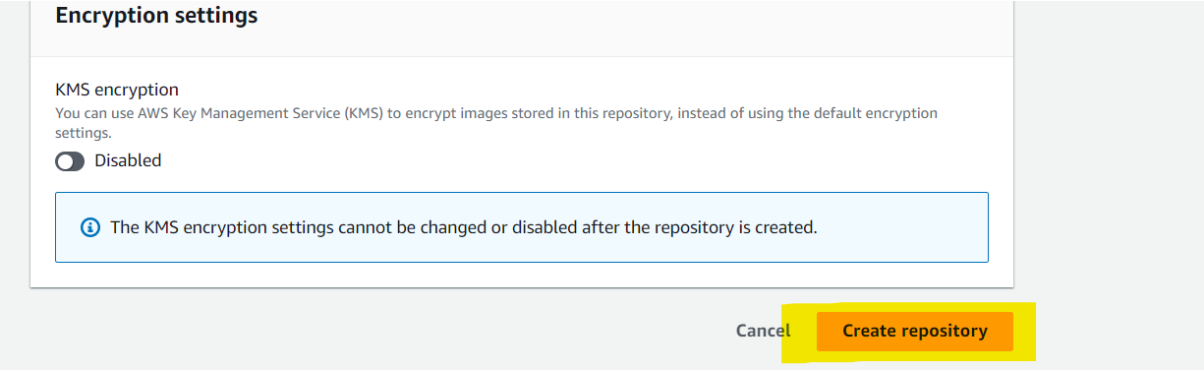
Create ECR.

Go inside ECR home page. Click on “Get started”

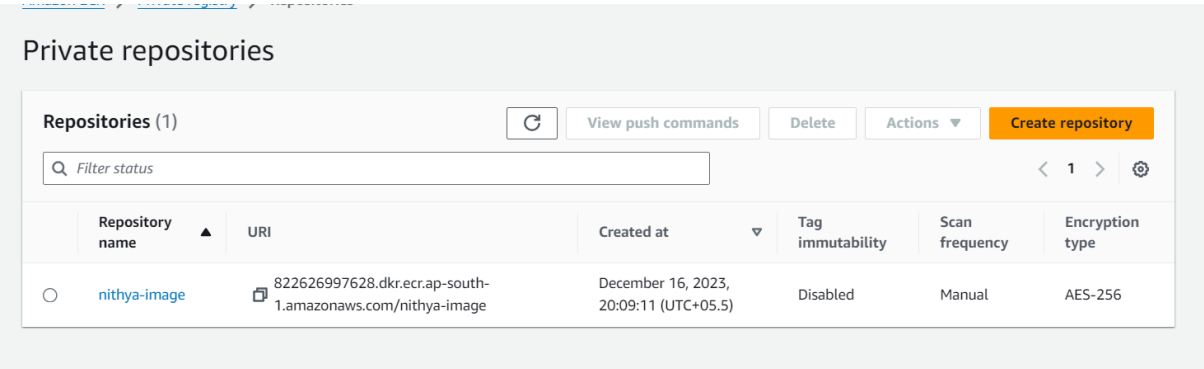
Give repository name.



Then click on “Create Repository”



Now repository will be created.



Step 4:

Login to ECR to pull images to repository

First we need to configure aws credentials to EC2 instance.

```
[ec2-user@ip-172-31-42-60 jenkins]$ aws configure
```

```
AWS Access Key ID [None]: xxxxxxxxxxxxxxxx
AWS Secret Access Key [None]: xxxxxxxxxxxxxxxxxxxxxxxx
Default region name [None]:
Default output format [None]:
```

Now we can login to ECR.

aws ecr get-login-password --region <your-region> | docker login --username AWS --password-stdin <your-aws-account-id>.dkr.ecr.<your-region>.amazonaws.com

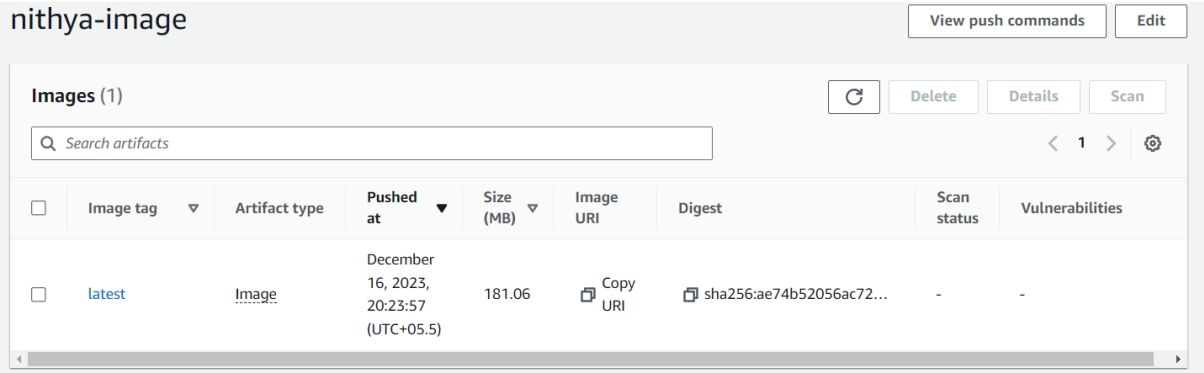
aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin xxxxxx.dkr.ecr.ap-south-1.amazonaws.com

Step 5:

Now we can move our build image to our ECR.

```
[ec2-user@ip-172-31-42-60 .aws]$ docker images
REPOSITORY      TAG         IMAGE ID        CREATED           SIZE
nithya-image    latest      626bd6616bf4    12 minutes ago    514MB
postgres        latest      391a00ec7cac    8 days ago        425MB
sonarqube       latest      dee4f32d6f90    2 weeks ago       725MB


docker tag nithya-image:latest 822626997628.dkr.ecr.ap-south-1.amazonaws.com/nithya-resume:latest
[ec2-user@ip-172-31-42-60 .aws]$ docker tag nithya-image:latest 822626997628.dkr.ecr.ap-south-1.amazonaws.com/nithya-resume:latest
[ec2-user@ip-172-31-42-60 .aws]$ docker push 822626997628.dkr.ecr.ap-south-1.amazonaws.com/nithya-iamge:latest
The push refers to repository [822626997628.dkr.ecr.ap-south-1.amazonaws.com/nithya-resume]
9def9ac4dd5e: Pushed
18701473167d: Pushed
8b80f6ba0e97: Pushed
63e923e36ca6: Pushed
5f70bf18a086: Pushed
74ddd0ec08fa: Pushed
latest: digest: sha256:ae74b52056ac725588f82c01cbf8496b82b39de2f2af00703269818 3005f3f1d size: 1571

[ec2-user@ip-172-31-42-60 .aws]$ docker images
REPOSITORY
TAG         IMAGE ID        CREATED           SIZE
822626997628.dkr.ecr.ap-south-1.amazonaws.com/nithya-image
latest      626bd6616bf4    14 minutes ago    514MB
nithya-image
latest      626bd6616bf4    14 minutes ago    514MB
postgres
latest      391a00ec7cac    8 days ago        425MB
sonarqube
latest      dee4f32d6f90    2 weeks ago       725MB
```

Step 6:
Now we can see our image in ECR.



Step 7:

Now we can build container using ECS or EKS. Now we are going to see how to create EKS using terraform.

Installation procedure:
https://developer.hashicorp.com/terraform/install

Follow below steps to install terraform on AmazonLinux.

```
sudo yum install -y yum-utils shadow-utils
sudo yum-config-manager --add-repo
https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
sudo yum -y install terraform
```

Step 8:

EKS terraform code available in below link.
https://github.com/ksnithya/eks-helm.git

We can clone to our system.

```
[ec2-user@ip-172-31-42-60 terraform]$ git clone
https://github.com/ksnithya/eks-helm.git
[ec2-user@ip-172-31-42-60 terraform]$ ls -l
total 0
drwxr-xr-x. 4 ec2-user ec2-user 150 Dec 16 15:07 eks-helm
[ec2-user@ip-172-31-42-60 terraform]$ cd eks-helm/
[ec2-user@ip-172-31-42-60 eks-helm]$ ls -l
total 24
-rw-r--r--. 1 ec2-user ec2-user  268 Dec 16 15:07 README.md
-rw-r--r--. 1 ec2-user ec2-user 2153 Dec 16 15:07 eks-
cluster.tf
-rw-r--r--. 1 ec2-user ec2-user 1062 Dec 16 15:07 eks-out.tf
-rw-r--r--. 1 ec2-user ec2-user 2294 Dec 16 15:07 eks-vpc.tf
-rw-r--r--. 1 ec2-user ec2-user 3551 Dec 16 15:07 eks-worker-
nodes.tf
drwxr-xr-x. 4 ec2-user ec2-user  155 Dec 16 15:07 helm
-rw-r--r--. 1 ec2-user ec2-user  220 Dec 16 15:07 version.tf

[ec2-user@ip-172-31-42-60 eks-helm]$ terraform init
```

```
[ec2-user@ip-172-31-42-60 eks-helm]$ terraform plan

[ec2-user@ip-172-31-42-60 eks-helm]$ terraform apply
```

Step 8:

We need to install kubectl to access the eks cluster.

Follow below link for installation.
https://medium.com/@saeidlaalkaei/installing-kubectl-on-amazon-linux-2-machine-fc82a3e6b7c8

```
[ec2-user@ip-172-31-42-60 ~]$ sudo curl -LO
"https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
[ec2-user@ip-172-31-42-60 ~]$ ls -l
total 107028
drwxr-xr-x. 3 ec2-user ec2-user       78 Dec 15 17:47 aws
-rw-r--r--. 1 ec2-user ec2-user 59888445 Dec 16 11:49
awscliv2.zip
drwxr-xr-x. 3 ec2-user ec2-user       21 Dec 16 11:57 docker
-rw-r--r--. 1 root     root     49704960 Dec 16 15:22 kubectl
drwxr-xr-x. 3 ec2-user ec2-user       22 Dec 16 15:07
terraform
[ec2-user@ip-172-31-42-60 ~]$ sudo chmod +x kubectl
[ec2-user@ip-172-31-42-60 ~]$ sudo mv kubectl /usr/local/bin/
[ec2-user@ip-172-31-42-60 ~]$ kubectl version
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Now we need to add our cluster to kubeconfig file to access from our EC2 instance.

```
[ec2-user@ip-172-31-42-60 ~]$ aws eks --region ap-south-1
update-kubeconfig --name eks_cluster_demo
Added new context arn:aws:eks:ap-south-
1:822626997628:cluster/eks_cluster_demo to /home/ec2-
user/.kube/config
[ec2-user@ip-172-31-42-60 ~]$ [ec2-user@ip-172-31-42-60 ~]$
aws eks --region ap-south-1 update-kubeconfig --name
eks_cluster_demo
Added new context arn:aws:eks:ap-south-
1:822626997628:cluster/eks_cluster_demo to /home/ec2-
user/.kube/config
[ec2-user@ip-172-31-42-60 ~]$ kubectl get nodes
NAME                                           STATUS   ROLES
AGE   VERSION
ip-10-0-0-112.ap-south-1.compute.internal   Ready    <none>
10m   v1.28.3-eks-e71965b
ip-10-0-1-34.ap-south-1.compute.internal    Ready    <none>
10m   v1.28.3-eks-e71965b

NAME                                           STATUS   ROLES
AGE   VERSION
ip-10-0-0-112.ap-south-1.compute.internal   Ready    <none>
10m   v1.28.3-eks-e71965b
```

```
ip-10-0-1-34.ap-south-1.compute.internal    Ready    <none>
10m   v1.28.3-eks-e71965b


[ec2-user@ip-172-31-42-60 eks]$ kubectl get nodes
NAME                                        STATUS   ROLES
AGE   VERSION
ip-10-0-0-112.ap-south-1.compute.internal   Ready    <none>
15m   v1.28.3-eks-e71965b
ip-10-0-1-34.ap-south-1.compute.internal    Ready    <none>
15m   v1.28.3-eks-e71965b
```

Step 9:

Now we can create deployment and service to connect outside.

Code available below repo:
https://github.com/ksnithya/terraform-resume.git

```
mkdir terraform
cd terraform
git clone https://github.com/ksnithya/terraform-resume.git
[ec2-user@ip-172-31-42-60 eks]$ ls -l
total 12
-rw-r--r--. 1 ec2-user ec2-user 606 Dec 16 15:30 deploy.yml
-rw-r--r--. 1 ec2-user ec2-user 326 Dec 16 16:56 hpa.yml
-rw-r--r--. 1 ec2-user ec2-user 283 Dec 16 15:30 service.yml
[ec2-user@ip-172-31-42-60 eks]$
[ec2-user@ip-172-31-42-60 eks]$ kubectl create -f .

[ec2-user@ip-172-31-42-60 eks]$ kubectl get deploy,svc
NAME                              READY   UP-TO-DATE   AVAILABLE
AGE
deployment.apps/nithya-resume     2/2     2            2
81s

NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP
PORT(S)          AGE
service/kubernetes    ClusterIP   172.20.0.1      <none>
443/TCP          88m
service/nithya-svc    NodePort    172.20.142.43   <none>
8080:30000/TCP   71m
```

Step 10:

We can build same application using Helm chart.

First we install Helm in server.

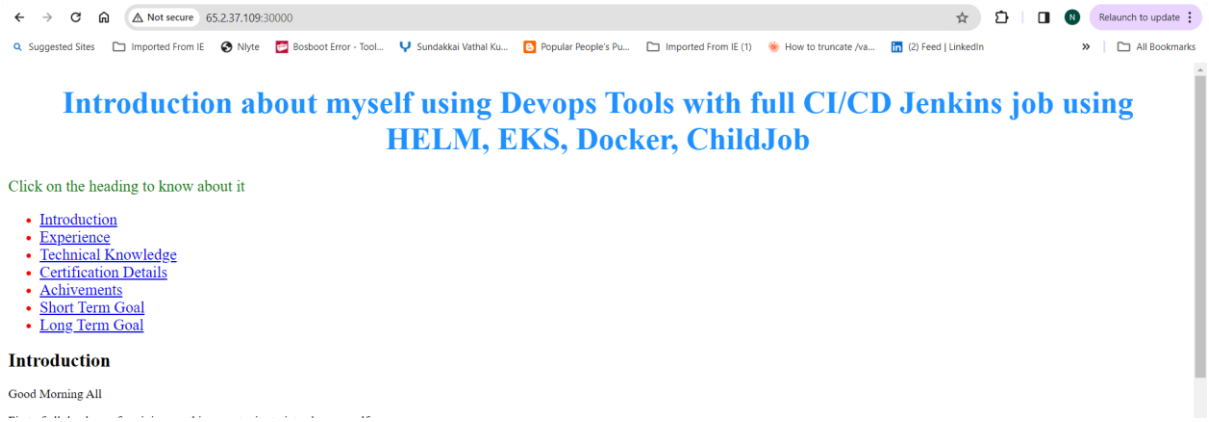
Follow below link for helm installation procedure.
https://helm.sh/docs/intro/install/

helm code available in below repositoty.
https://github.com/ksnithya/eks-helm.git

```
[ec2-user@ip-172-31-42-60 terraform]$ git clone
https://github.com/ksnithya/eks-helm.git
[ec2-user@ip-172-31-42-60 terraform]$ ls -l
total 0
drwxr-xr-x. 4 ec2-user ec2-user 150 Dec 16 15:07 eks-helm
[ec2-user@ip-172-31-42-60 terraform]$ cd eks-helm/
[ec2-user@ip-172-31-42-60 eks-helm]$ ls -l
total 24
-rw-r--r--. 1 ec2-user ec2-user  268 Dec 16 15:07 README.md
-rw-r--r--. 1 ec2-user ec2-user 2153 Dec 16 15:07 eks-
cluster.tf
-rw-r--r--. 1 ec2-user ec2-user 1062 Dec 16 15:07 eks-out.tf
-rw-r--r--. 1 ec2-user ec2-user 2294 Dec 16 15:07 eks-vpc.tf
-rw-r--r--. 1 ec2-user ec2-user 3551 Dec 16 15:07 eks-worker-
nodes.tf
drwxr-xr-x. 4 ec2-user ec2-user  155 Dec 16 15:07 helm
-rw-r--r--. 1 ec2-user ec2-user  220 Dec 16 15:07 version.tf

[ec2-user@ip-172-31-42-60 terraform]$ cd helm
[ec2-user@ip-172-31-42-60 helm]$ ls -l
total 8
-rw-r--r--. 1 ec2-user ec2-user  92 Dec 16 15:07 main.tf
drwxr-xr-x. 3 ec2-user ec2-user  79 Dec 16 15:07 nithya
-rw-r--r--. 1 ec2-user ec2-user 230 Dec 16 15:07 version.tf
[ec2-user@ip-172-31-42-60 helm]$
Note:
We can create helm chart using below command.
helm create <repo name>

[ec2-user@ip-172-31-42-60 helm]$ cd nithya/
[ec2-user@ip-172-31-42-60 nithya]$ ls -l
total 8
-rw-r--r--. 1 ec2-user ec2-user 1142 Dec 16 15:07 Chart.yaml
drwxr-xr-x. 3 ec2-user ec2-user  135 Dec 16 15:07 templates
-rw-r--r--. 1 ec2-user ec2-user 1891 Dec 16 15:07 values.yaml
[ec2-user@ip-172-31-42-60 nithya]$ helm install nithya .
NAME: nithya
LAST DEPLOYED: Sun Dec 17 07:24:26 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace default -o
jsonpath="{.spec.ports[0].nodePort}" services nithya)
  export NODE_IP=$(kubectl get nodes --namespace default -o
jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT

[ec2-user@ip-172-31-42-60 nithya]$ kubectl get pod
NAME                              READY   STATUS    RESTARTS
AGE
nithya-resume-79bb46854c-gxcvs    1/1     Running   0
14s
nithya-resume-79bb46854c-pqvtk    1/1     Running   0
14s
[ec2-user@ip-172-31-42-60 nithya]$ kubectl get svc
```

```
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)          AGE
kubernetes    ClusterIP   172.20.0.1       <none>
443/TCP          7m23s
nithya-svc    NodePort    172.20.215.240   <none>
8080:30000/TCP   23s
```

We can access the application using <public ip of workernode>:30000

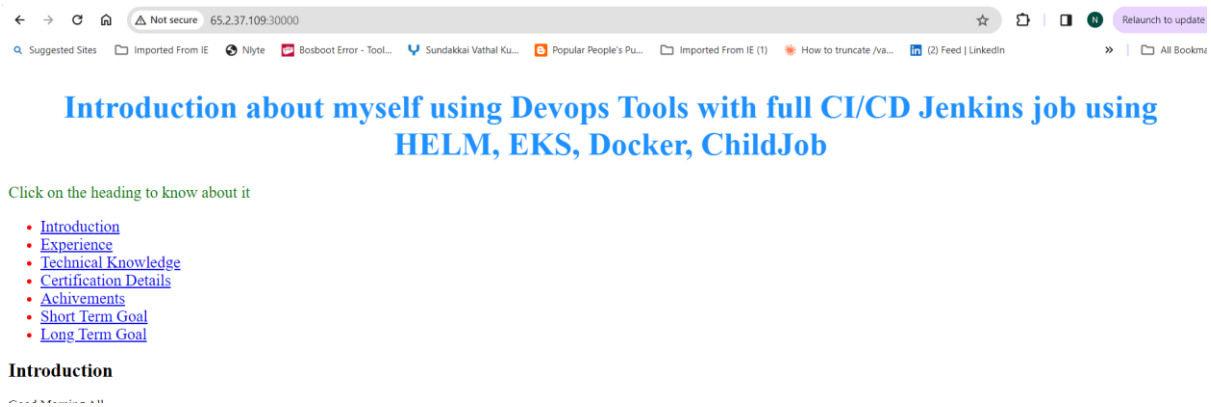


Step 11:

We can also deploy the same application using terraform.

```
[ec2-user@ip-172-31-42-60 terraform]$ git clone
https://github.com/ksnithya/eks-helm.git
[ec2-user@ip-172-31-42-60 terraform]$ ls -l
total 0
drwxr-xr-x. 4 ec2-user ec2-user 150 Dec 16 15:07 eks-helm
[ec2-user@ip-172-31-42-60 terraform]$ cd eks-helm/
[ec2-user@ip-172-31-42-60 eks-helm]$ ls -l
total 24
-rw-r--r--. 1 ec2-user ec2-user  268 Dec 16 15:07 README.md
-rw-r--r--. 1 ec2-user ec2-user 2153 Dec 16 15:07 eks-
cluster.tf
-rw-r--r--. 1 ec2-user ec2-user 1062 Dec 16 15:07 eks-out.tf
-rw-r--r--. 1 ec2-user ec2-user 2294 Dec 16 15:07 eks-vpc.tf
-rw-r--r--. 1 ec2-user ec2-user 3551 Dec 16 15:07 eks-worker-
nodes.tf
drwxr-xr-x. 4 ec2-user ec2-user  155 Dec 16 15:07 helm
-rw-r--r--. 1 ec2-user ec2-user  220 Dec 16 15:07 version.tf

[ec2-user@ip-172-31-42-60 terraform]$ cd helm
[ec2-user@ip-172-31-42-60 helm]$ ls -l
total 8
-rw-r--r--. 1 ec2-user ec2-user  92 Dec 16 15:07 main.tf
drwxr-xr-x. 3 ec2-user ec2-user  79 Dec 16 15:07 nithya
-rw-r--r--. 1 ec2-user ec2-user 230 Dec 16 15:07 version.tf
[ec2-user@ip-172-31-42-60 helm]$
[ec2-user@ip-172-31-42-60 helm]$ terraform init

Initializing the backend...
```

```
Initializing provider plugins...
- Finding latest version of hashicorp/helm...
- Finding latest version of hashicorp/kubernetes...
- Installing hashicorp/kubernetes v2.24.0...
- Installed hashicorp/kubernetes v2.24.0 (signed by HashiCorp)
- Installing hashicorp/helm v2.12.1...
- Installed hashicorp/helm v2.12.1 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to
record the provider
selections it made above. Include this file in your version
control repository
so that Terraform can guarantee to make the same selections by
default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running
"terraform plan" to see
any changes that are required for your infrastructure. All
Terraform commands
should now work.

If you ever set or change modules or backend configuration for
Terraform,
rerun this command to reinitialize your working directory. If
you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-42-60 helm]$ terraform plan

[ec2-user@ip-172-31-42-60 helm]$ terraform apply
[ec2-user@ip-172-31-42-60 helm]$ kubectl get pods
NAME                              READY   STATUS    RESTARTS
AGE
nithya-resume-79bb46854c-frvcm    1/1     Running   0
14s
nithya-resume-79bb46854c-vv4c4    1/1     Running   0
14s
[ec2-user@ip-172-31-42-60 helm]$ kubectl get deploy
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
nithya-resume   2/2     2            2           20s
[ec2-user@ip-172-31-42-60 helm]$ kubectl get svc
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)          AGE
kubernetes   ClusterIP   172.20.0.1       <none>
443/TCP          23m
nithya-svc   NodePort    172.20.184.143   <none>
8080:30000/TCP   27s
[ec2-user@ip-172-31-42-60 helm]$
```



Step 12:

How to do above all task completely using Jenkins CI/CD pipeline.

First we need to install Jenkins.
https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/

To make Jenkins id to access aws and Kubernetes follow below steps.

Su - Jenkins
aws configure
aws eks --region ap-south-1 update-kubeconfig --name eks_cluster_demo

First we create job for helm.

New Item -> <give job name> -> pipeline -> ok

Give below code.

```
pipeline {
    agent any
        stages {
            stage('Download Repo') {
                steps {
                    // Get some code from a GitHub repository
                    git 'https://github.com/ksnithya/helm-
nithya.git'
                }
            }
            stage('check chart exist'){
                steps{
                    script{
                        def x = sh(script: "helm status nithya
| wc -l", returnStdout: true)
                        if (x.toInteger() > 0){
                            sh '''cd nithya
                            helm uninstall nithya
                            helm install nithya .
                            '''
```

```
                        } else {
                            sh '''cd nithya
                            helm install nithya .
                            '''
                        }
                    }
                }
            }
        }
}
```

Then we create main job and add helm job into it.

New Item -> <give job name> -> pipeline -> ok

Use below code.

```
pipeline {
    agent any
       stages {
        stage('Download Repo') {
            steps {
                // Get some code from a GitHub repository
                git 'https://github.com/ksnithya/jenkins.git'
            }
        }
        stage('Image Delete') {
            steps {
                script {
                    def x = sh(script: "docker images | grep
nithya-image | wc -l", returnStdout: true)
                    if (x.toInteger() > 0){
                        sh '''
                        docker rmi -f 822626997628.dkr.ecr.ap-
south-1.amazonaws.com/nithya-image:latest
                        '''
                    }
                }
            }
        }
        stage('Image Build') {
            steps {
                // img = sh("docker inspect nithya:v1 >
/dev/null 2>&1 && echo yes || echo no")
                // sh "echo 'output: ${img}'"
                sh 'docker build -t 822626997628.dkr.ecr.ap-
south-1.amazonaws.com/nithya-image:latest .'
            }
        }
        stage('Image Pull') {
            steps {
                script {
                    sh 'docker push 822626997628.dkr.ecr.ap-
south-1.amazonaws.com/nithya-image:latest'
                }
            }
        }
        stage('triggerHelmJob') {
```

```
                steps {
                    build job: "helm-nithya", wait: true
                }
            }
        }
}
```