**Create Build and Deploy Pipeline for K8s application deployment.**

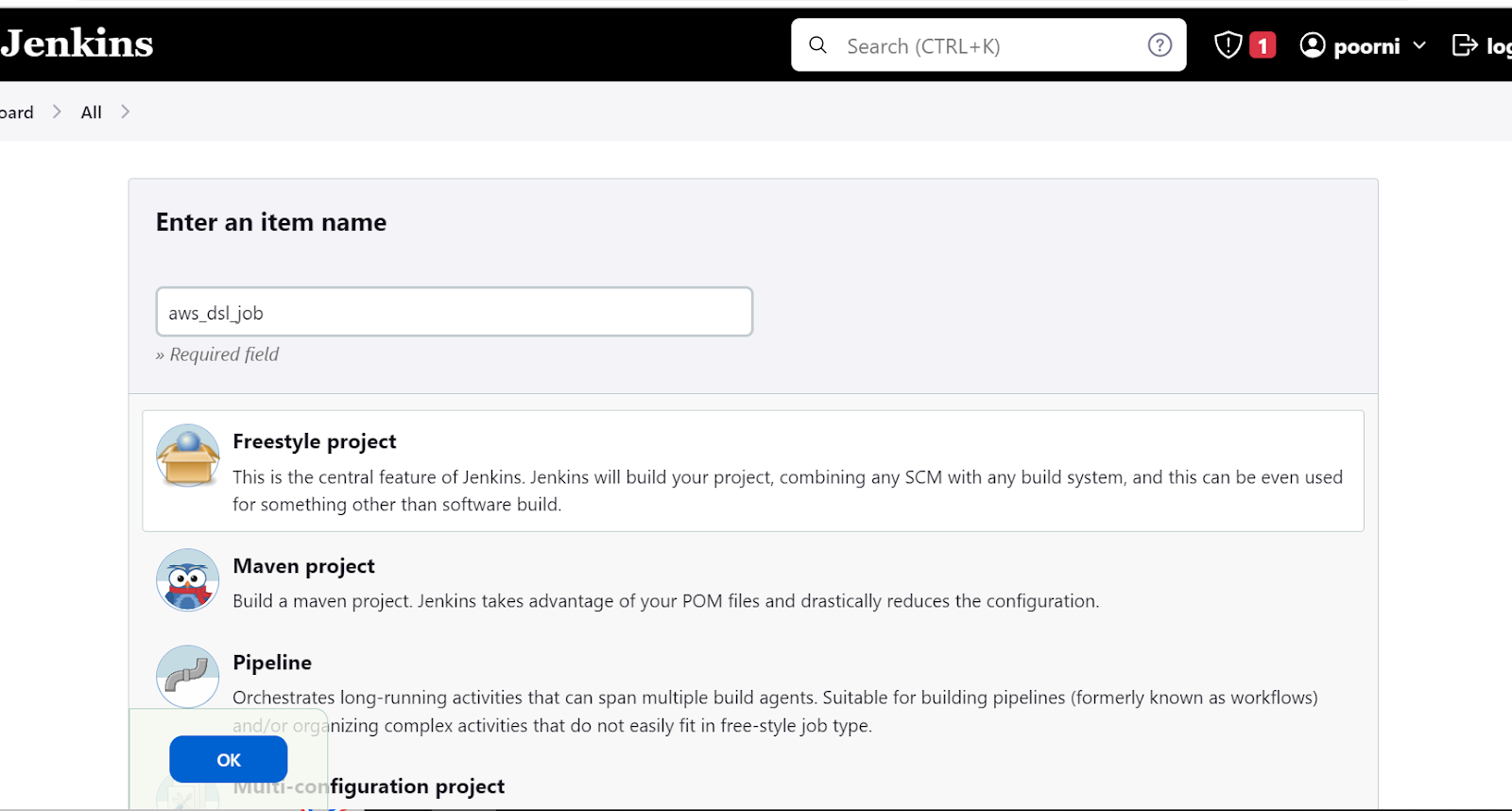
1.Create Job DSL code to create build and deploy pipeline:

* Installed job dsl plugin in jenkins server
* Add aws secret key ,accesskey in jenkins credential
* Install docker and add docker as group user

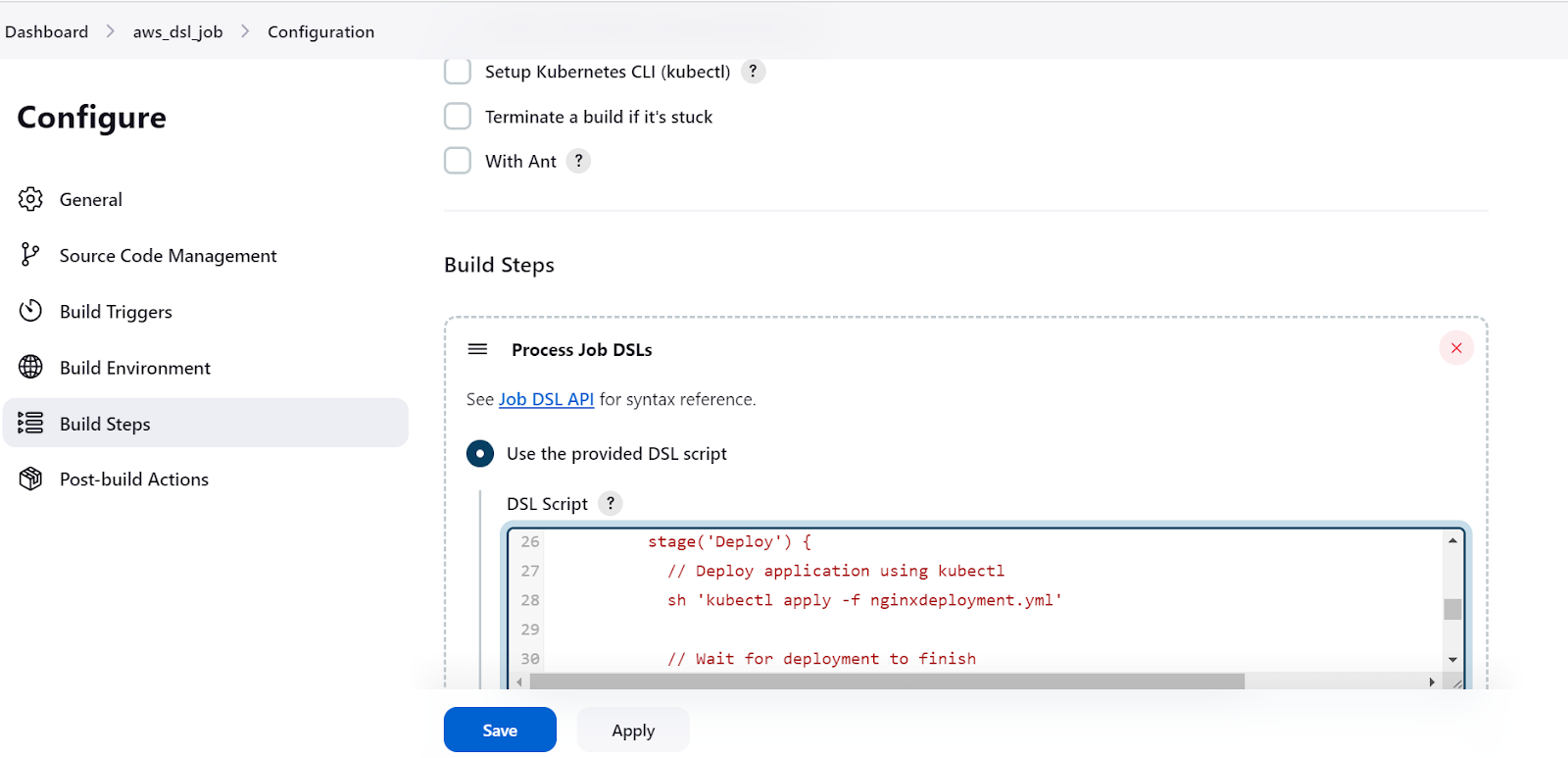
**cmd:**

1. **usermod -aG docker user\_name**
2. **usermod -aG docker $USER**
3. **sudo groupadd docker**
4. **sudo systemctl restart docker**

2.Created  a freestyle job to Run Job DSL code to create the pipeline jobs.



**In Built steps select DSL script**



* Write a docker file to build image and the docker build commands are executed in the BuildPipeline

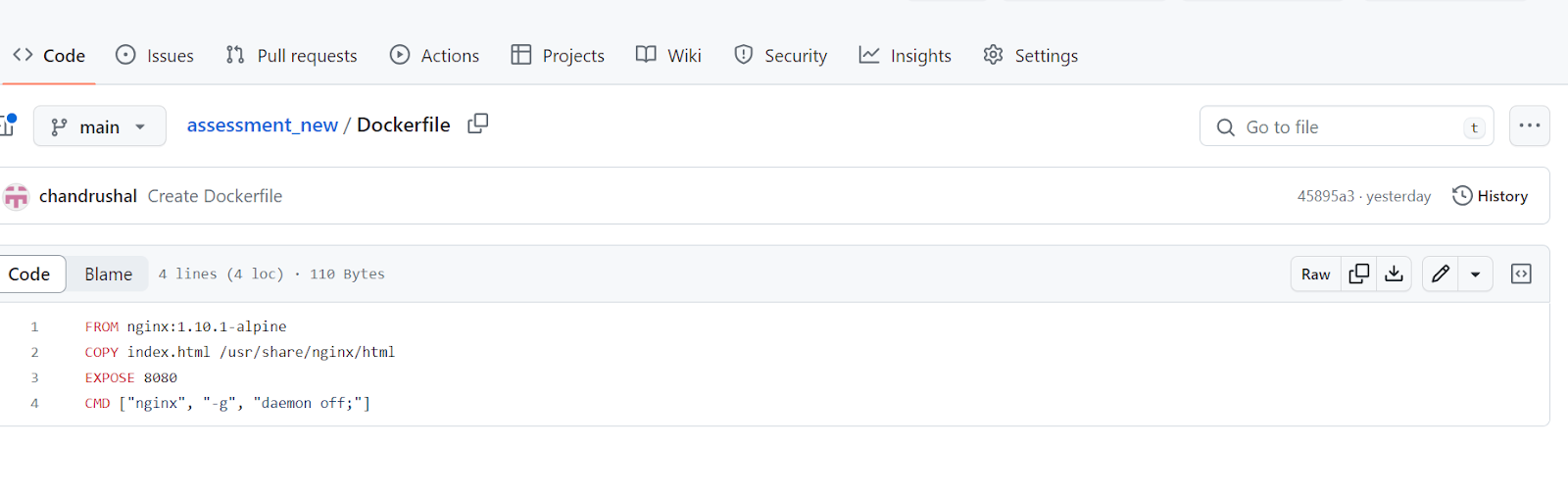
**Docker file code**

FROM nginx:1.10.1-alpine

COPY index.html /usr/share/nginx/html

EXPOSE 8080

CMD ["nginx", "-g", "daemon off;"]



**Code to build :**

pipelineJob('BuildPipeline') {

  definition {

    cps {

      script("""

        node {

          stage('Build') {

            // Checkout source code from Git repository

           sh 'https://github.com/chandrushal/assessment\_new.git'

            // Build Docker image using Dockerfile

            sh 'docker build -t dsl .'

            // Push Docker image to AWS ECR

            sh 'docker push docker tag dsl:latest 354517960452.dkr.ecr.ap-south-1.amazonaws.com/dsl:latest/nginx'.

          }

        }

      """)

    }

  }

}

pipelineJob('DeployPipeline') {

  definition {

    cps {

      script("""

        node {

          stage('Deploy') {

            // Deploy application using kubectl

            sh 'kubectl apply -f nginxdeployment.yml'

            // Wait for deployment to finish

            sh 'kubectl rollout status deployment/nginxdeployment.yml'

            // Expose service using LoadBalancer

            sh 'kubectl expose deployment nginxdeployment.yml --type=LoadBalancer --port=80 --target-port=80'

            // Get LoadBalancer IP address

            sh 'kubectl get svc nginxdeployment.yml -o=jsonpath="{.status.loadBalancer.ingress[0].ip}" > lb-ip.txt'

          }

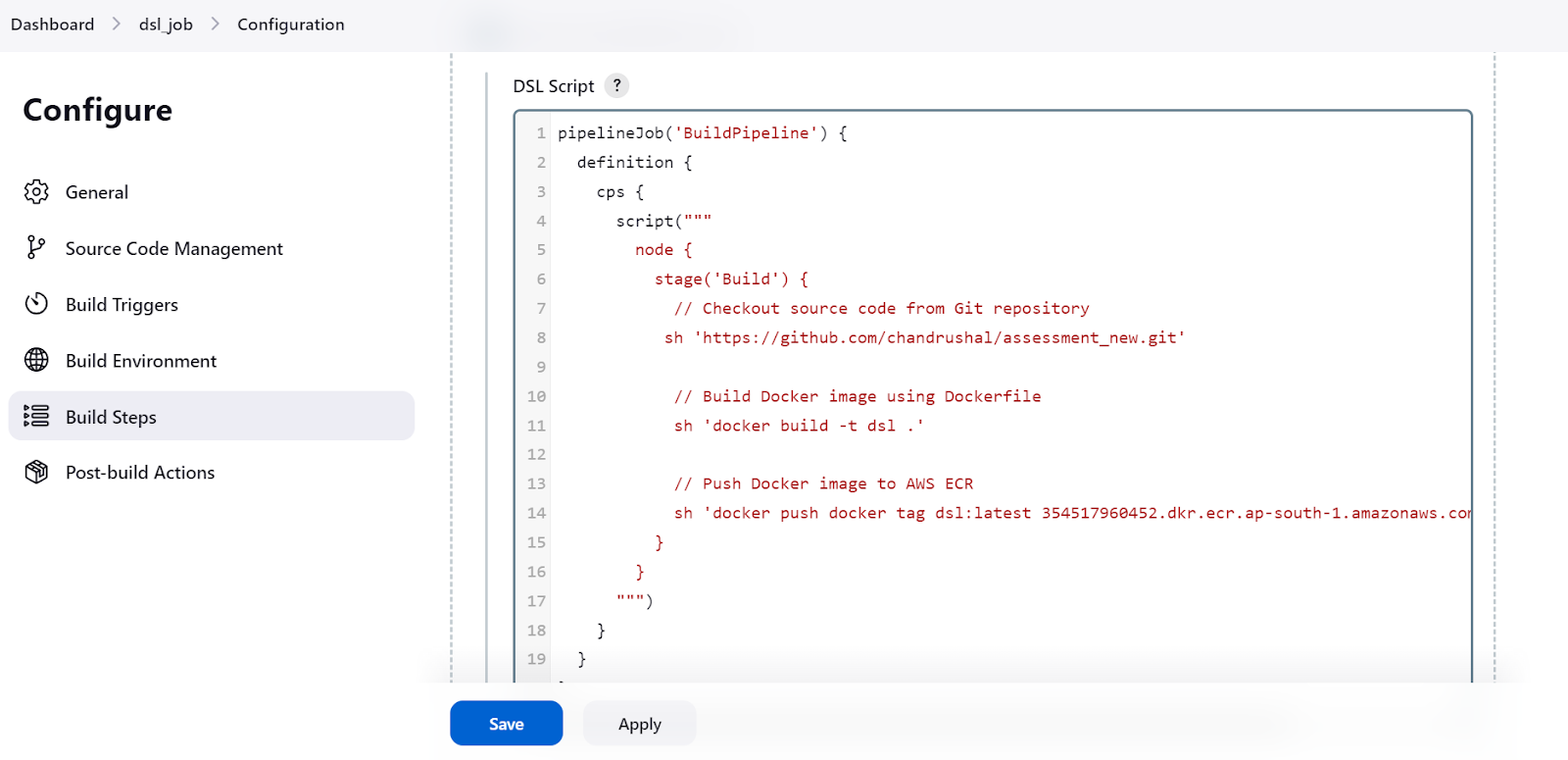
        }

      """)

    }

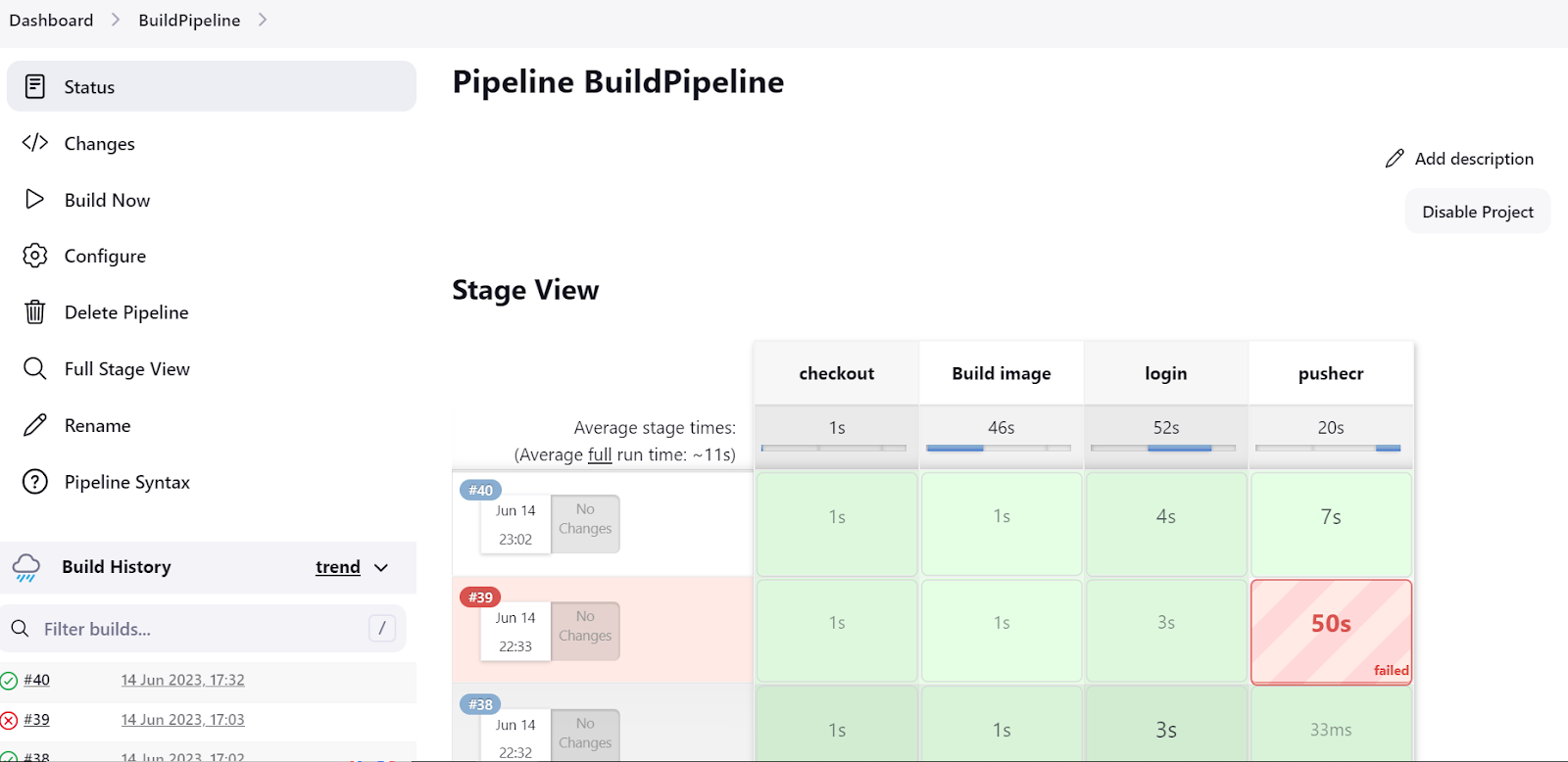
  }

}

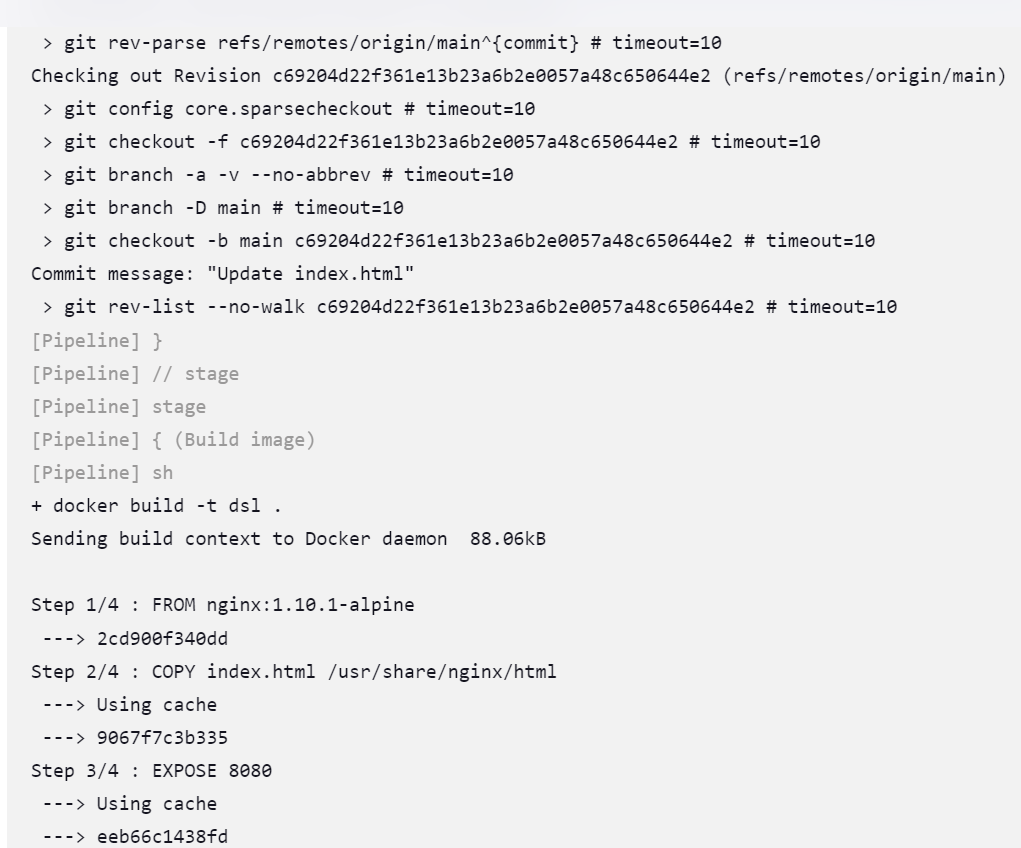


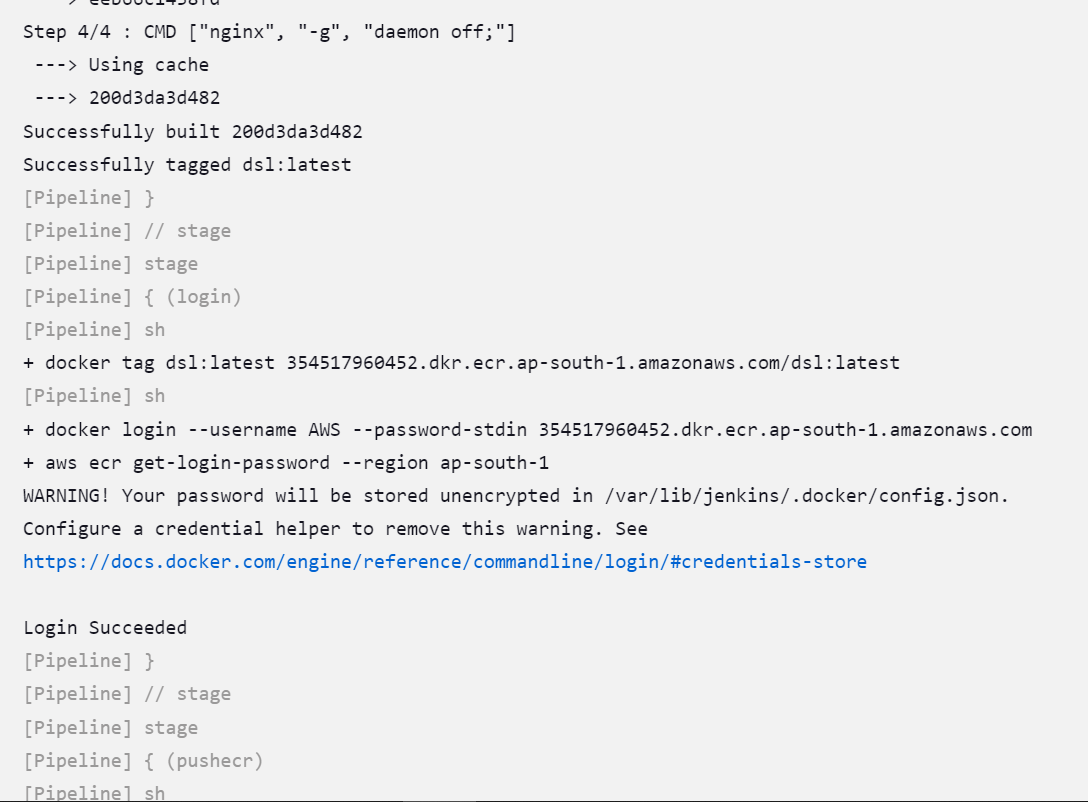


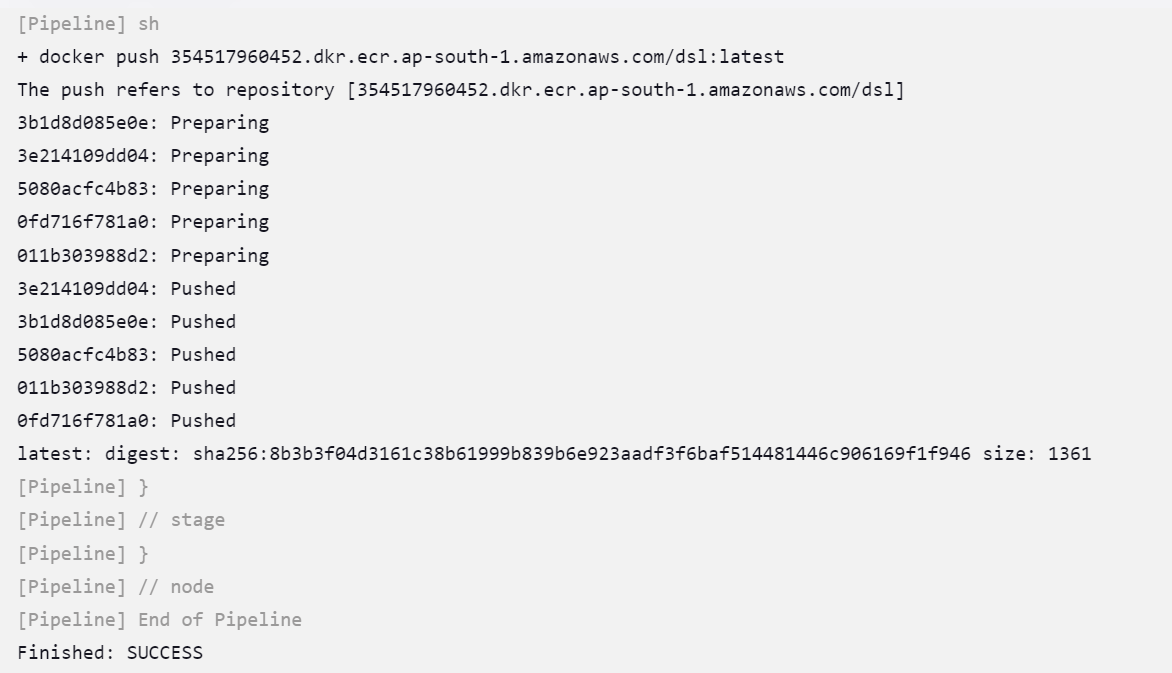
**Pipeline StageView:**







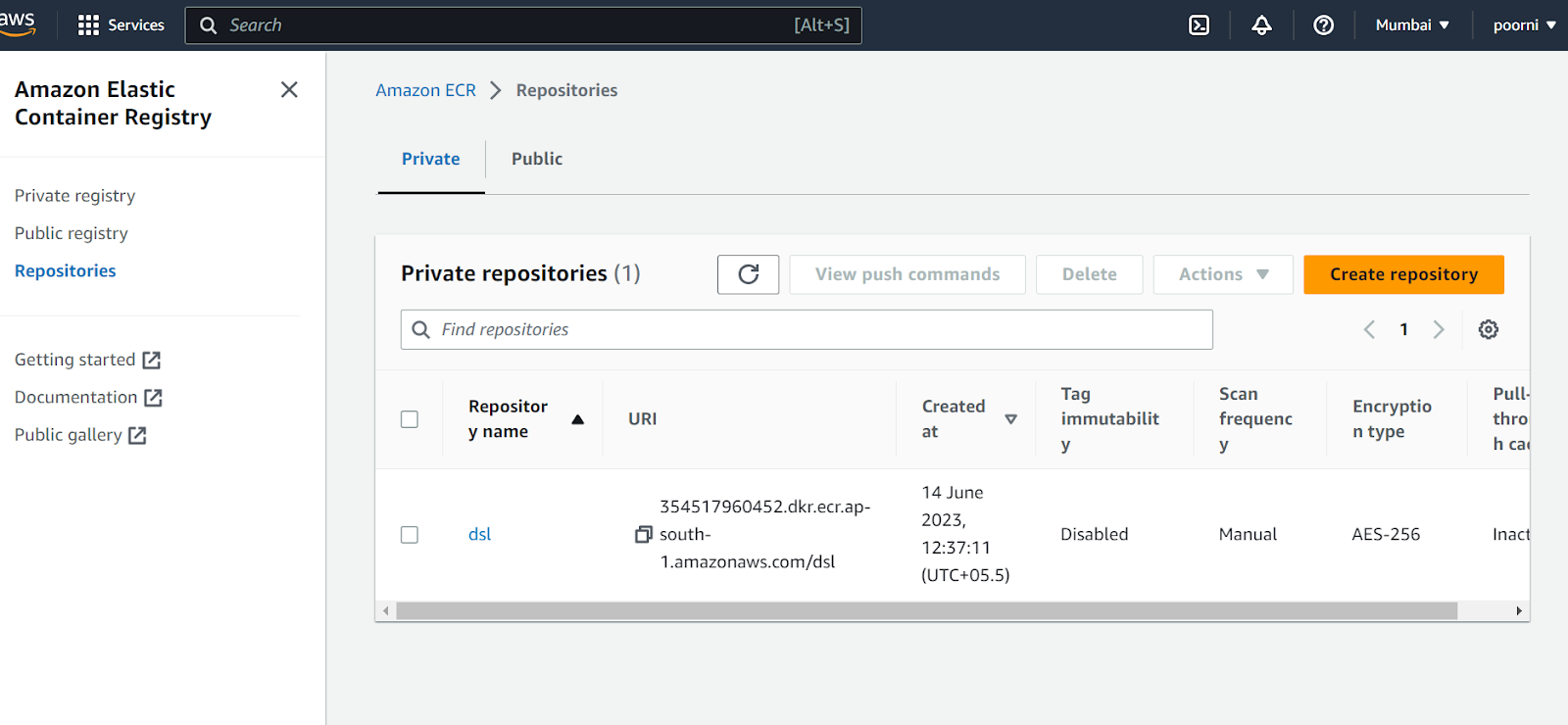




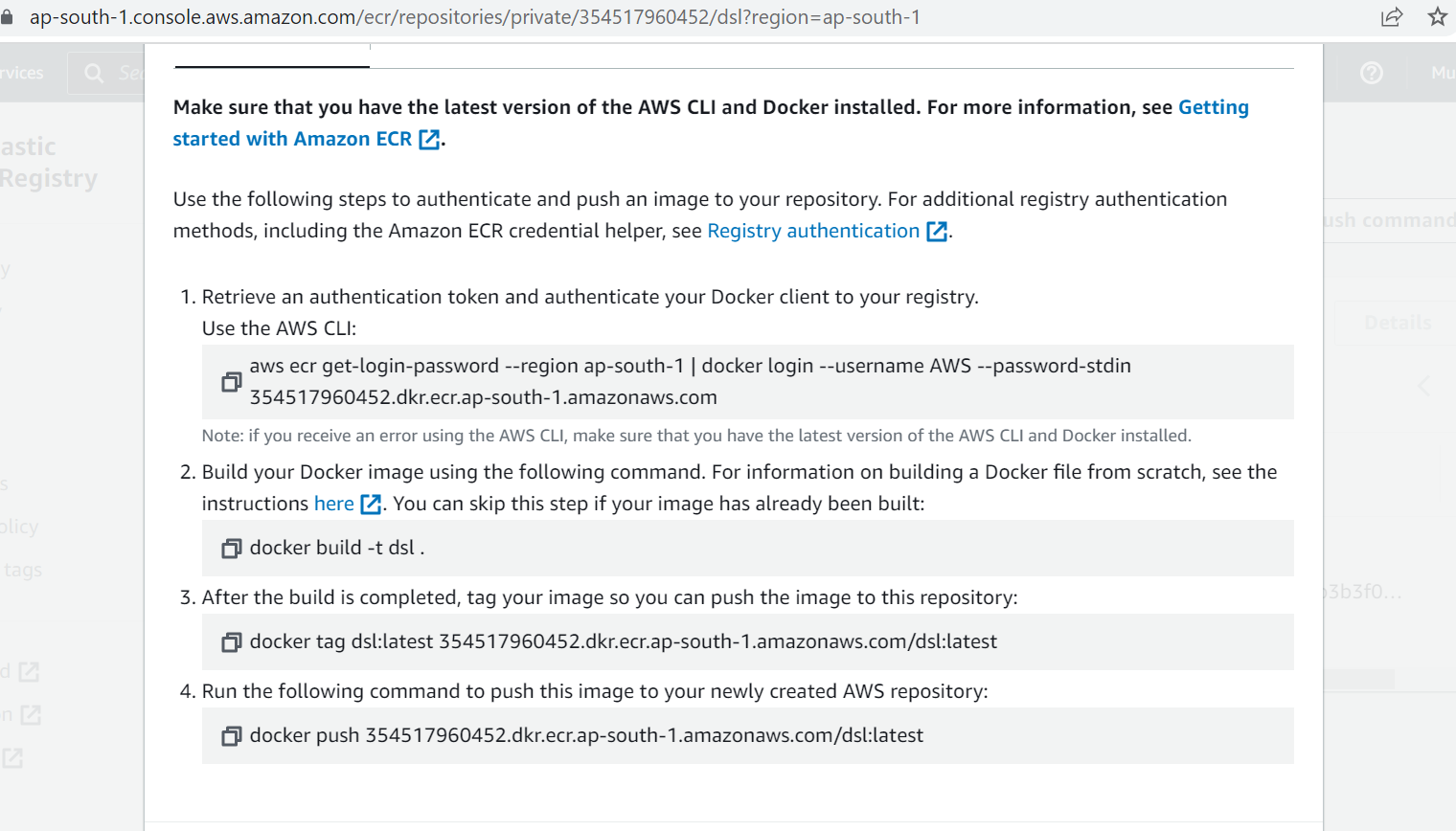
2. Build pipeline is image pipeline to build dockerfile and push image to aws elastic container registry(ECR).

* Dockerfile for nginx image is stored in git repository
* To push the docker image to ECR we have to create IAM role which has AmazonECRcontainerfullacces
* Attach that role with the EC2 instance

**ECR REPOSITORY:**



**Push commands for repository**



* authentication token and authenticate your Docker client to your registry. Use the AWS CLI:

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 354517960452.dkr.ecr.ap-south-1.amazonaws.com

* Build your Docker image using the following command. For information on building a Docker file from scratch

docker build -t dsl .

* After the build is completed, tag your image so you can push the image to this repository:

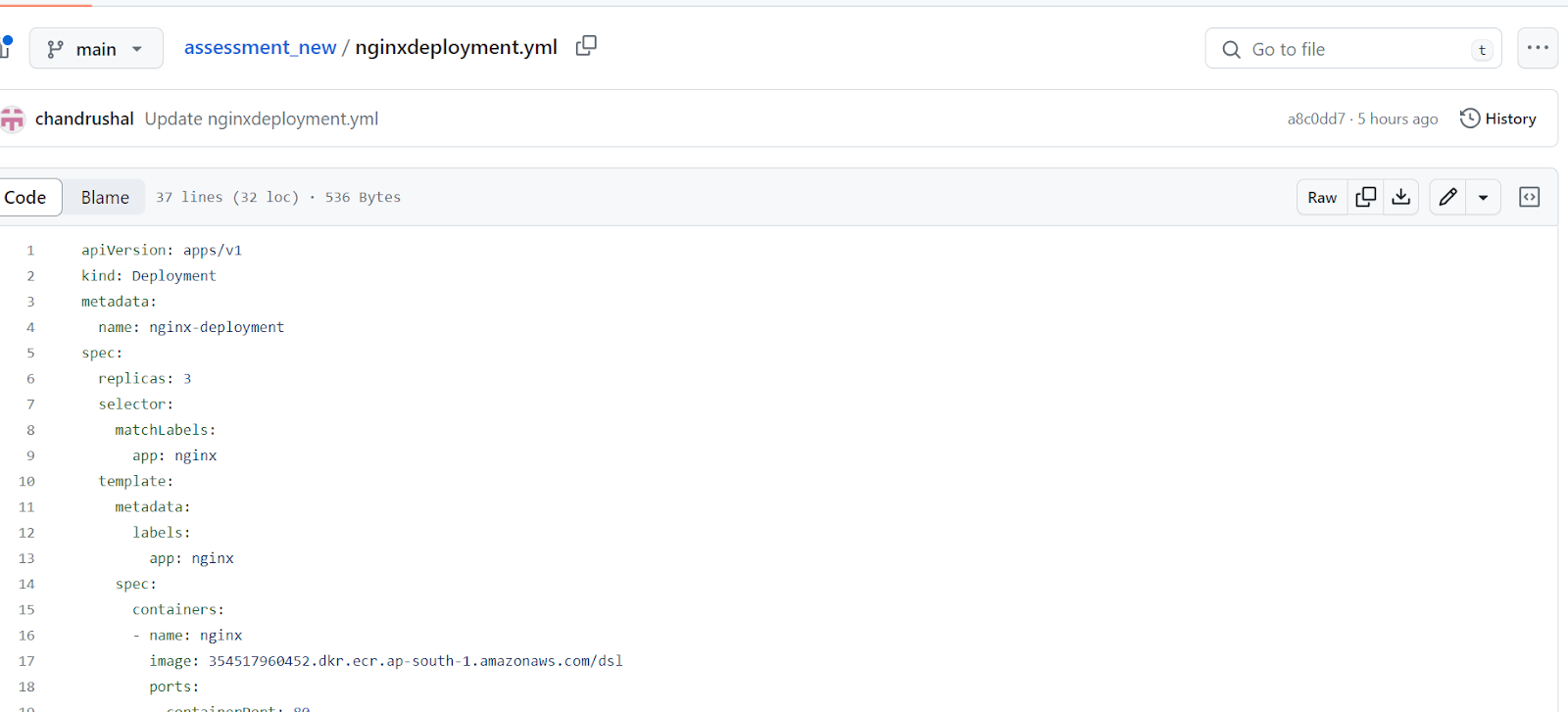
docker tag dsl:latest 354517960452.dkr.ecr.ap-south-1.amazonaws.com/dsl:latest

* Run the following command to push this image to your newly created AWS repository

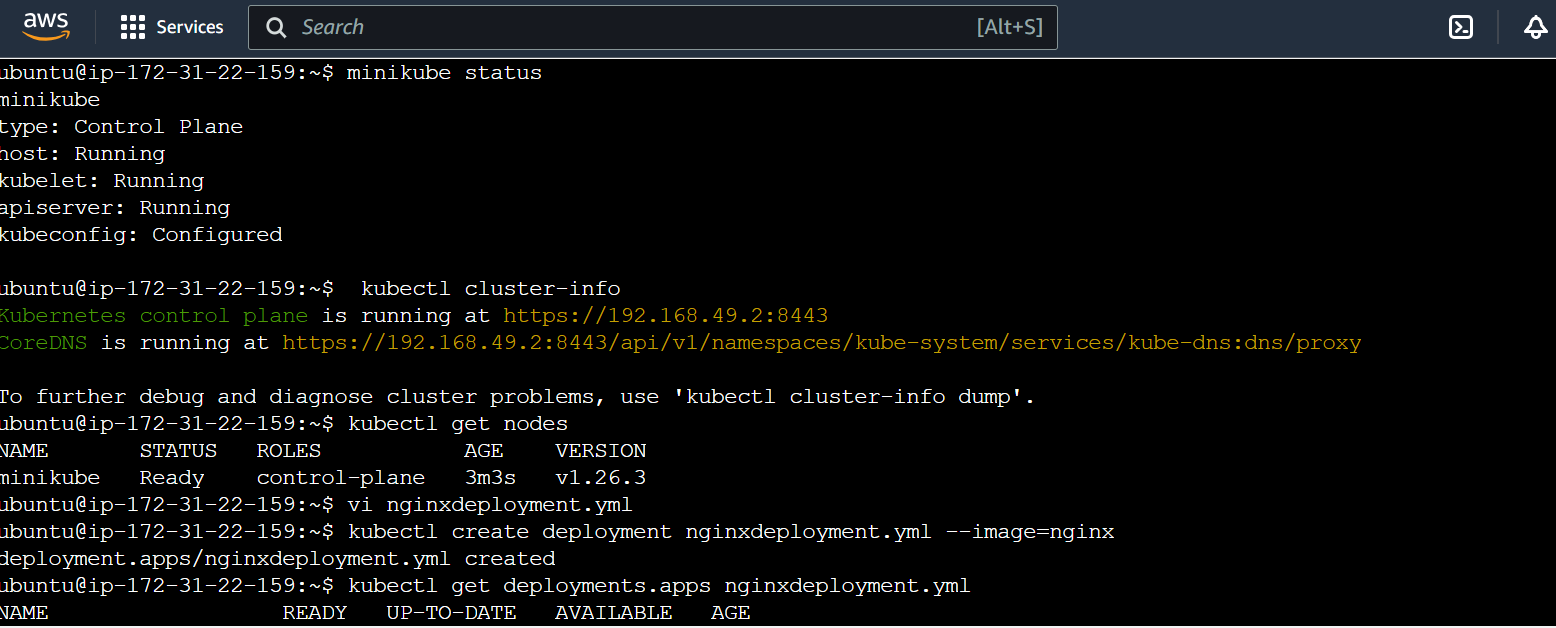
docker push 354517960452.dkr.ecr.ap-south-1.amazonaws.com/dsl:latest

3..deploy the docker nginx image from the ecr

In that deployment file specify the ecr repository url to pull the image from the ecr and save in the git repo



**Minikube Installed and cluster creation**



**Code for deployment.yml**

Deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

  name: nginx-deployment

  labels:

    app: nginx

spec:

  replicas: 3

  selector:

    matchLabels:

      app: nginx

  template:

    metadata:

      labels:

        app: nginx

    spec:

      containers:

      - name: nginx

        image: 354517960452.dkr.ecr.ap-south-1.amazonaws.com/dsl

        ports:

        - containerPort: 80

Once the nginx image is deployed it should get hit with th ipadderss and we can see the desired web page in the browser.