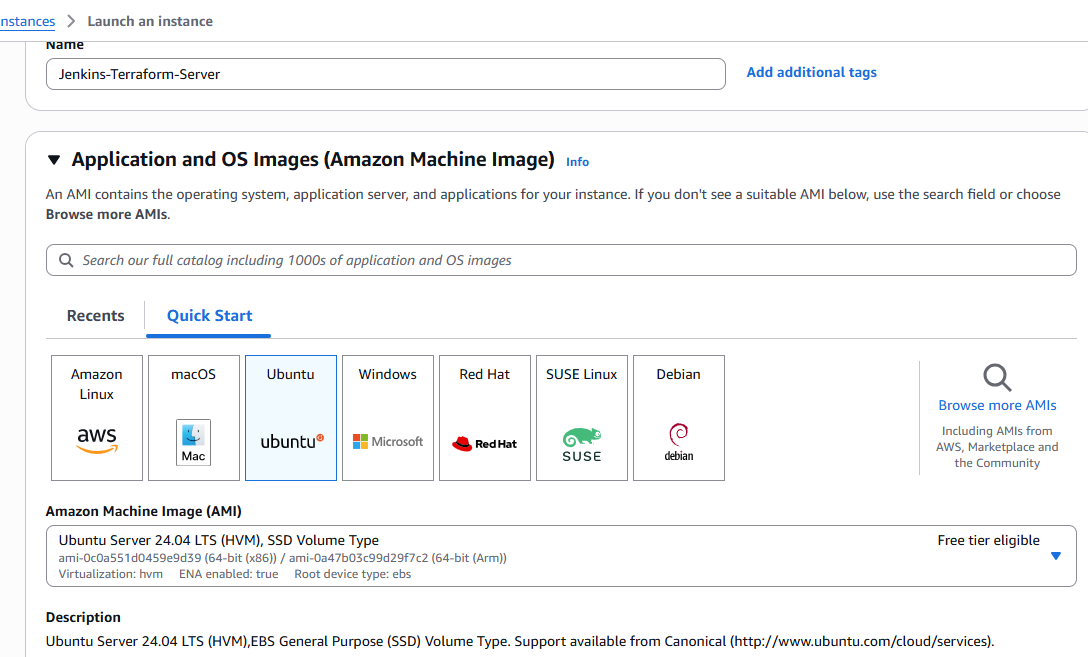
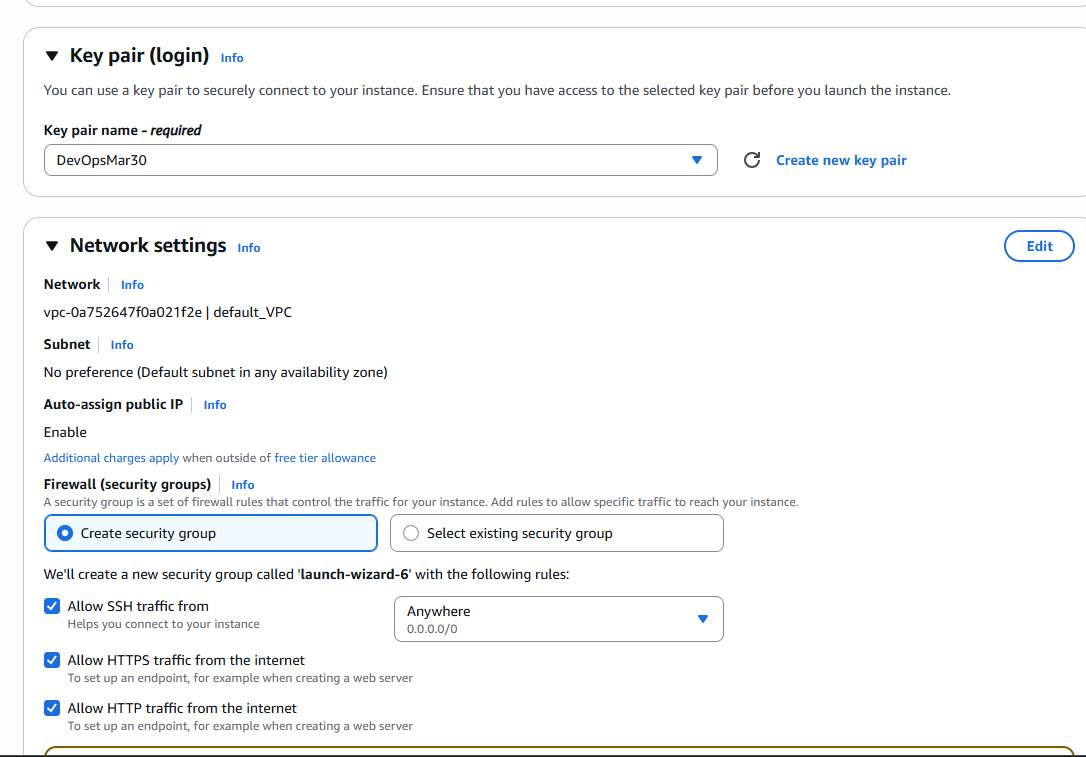
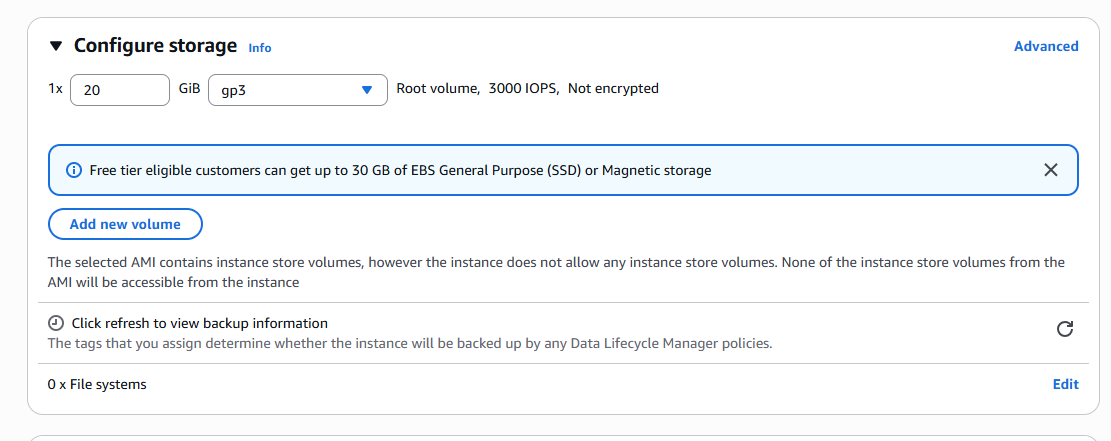
Jenkins Pipeline part 0

Jenkins CI/CD with React and SpringBoot

1. Create a Jenkins server with AWS CLI, Kubectl, Terraform, Jenkins, Java installed
2. Create a new Jenkins VM







===================================================================================

Jenkins CICD react and Spring boot

===================================================================================

1 --> Create a Jenkins server with aws cli, kubectl, terraform, Jenkins, java installed

#!/bin/bash

set -euxo pipefail

# Java installation for Jenkins

# Java installation for Jenkins

sudo apt update -y

sudo apt install -y openjdk-17-jdk # Change jre to jdk

sudo mkdir -p /etc/apt/keyrings

sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" \

| sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt update

sudo apt install -y jenkins

sudo systemctl enable jenkins

sudo systemctl start jenkins

# Terraform installation

sudo apt-get install -y gnupg software-properties-common curl

curl -fsSL https://apt.releases.hashicorp.com/gpg | gpg --dearmor | \

sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \

https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | \

sudo tee /etc/apt/sources.list.d/hashicorp.list > /dev/null

sudo apt-get update -y

sudo apt-get install -y terraform

# kubectl installation

KUBECTL\_VERSION=$(curl -L -s https://dl.k8s.io/release/stable.txt)

curl -LO "https://dl.k8s.io/release/${KUBECTL\_VERSION}/bin/linux/amd64/kubectl"

curl -LO "https://dl.k8s.io/release/${KUBECTL\_VERSION}/bin/linux/amd64/kubectl.sha256"

echo "$(cat kubectl.sha256) kubectl" | sha256sum --check

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

# AWS CLI installation

sudo apt install -y unzip

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

# Clean up

rm -rf aws awscliv2.zip kubectl kubectl.sha256

# Final log of versions

echo "✅ Installed versions:"

java -version

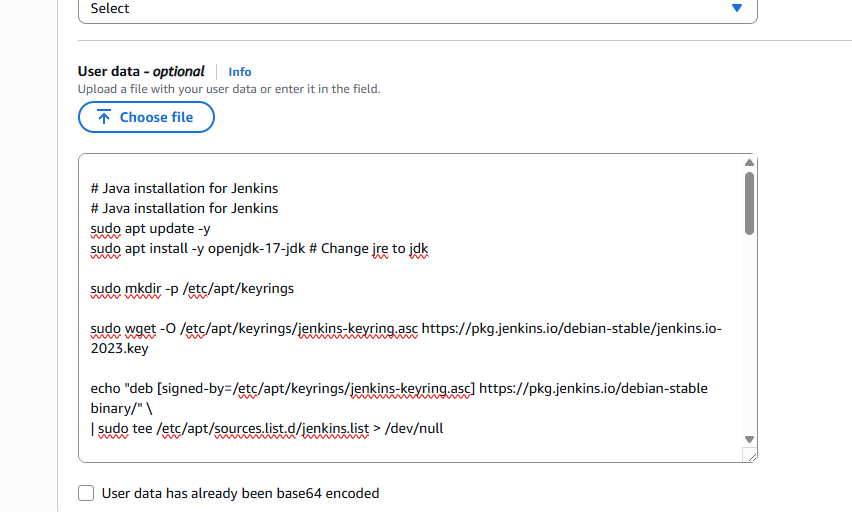
jenkins --version || echo "Jenkins installed"

terraform -version

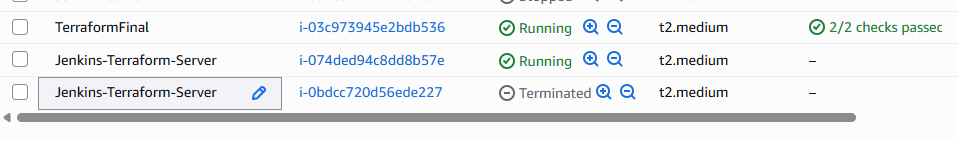
kubectl version --client

aws --version

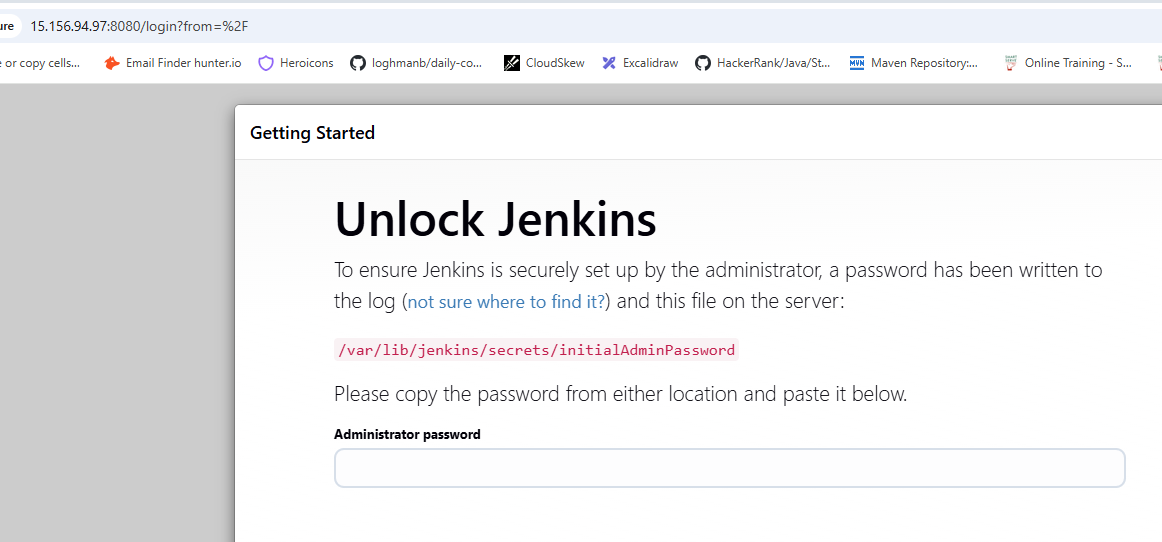
Port# 8080 for Jenkins



Copy paste

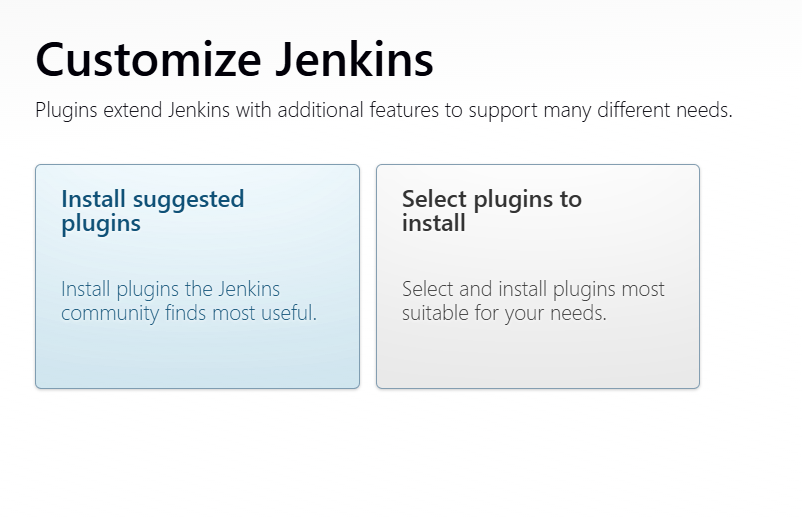


<http://15.156.94.97:8080/login?from=%2F>



ubuntu@ip-172-31-8-68:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Copy password and save in a file

Ins

Install suggested plugins

Fillout the login credentials page

admin, \*\*\*\*\*\*

2 --> Manage Jenkins --> Credentials --> Secret Text --> AWS\_ACCESS\_KEY\_ID and AWS\_SECRET\_KEY

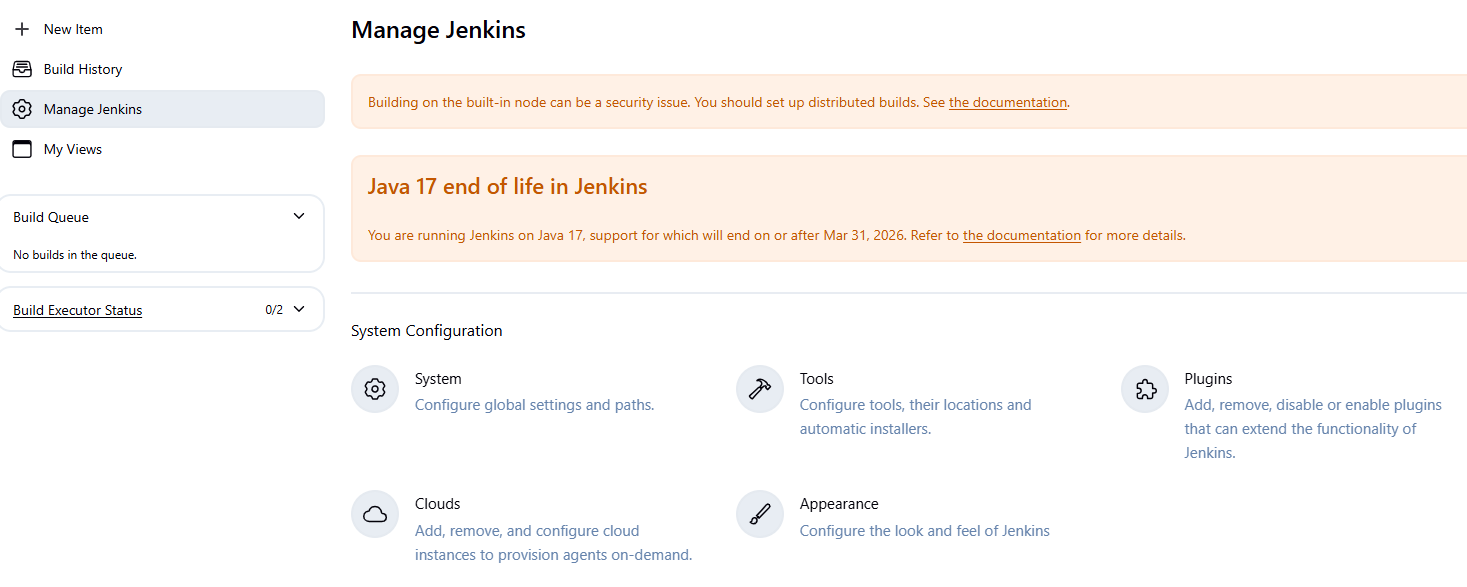
Get AWS\_ACCESS\_KEY from AWS account

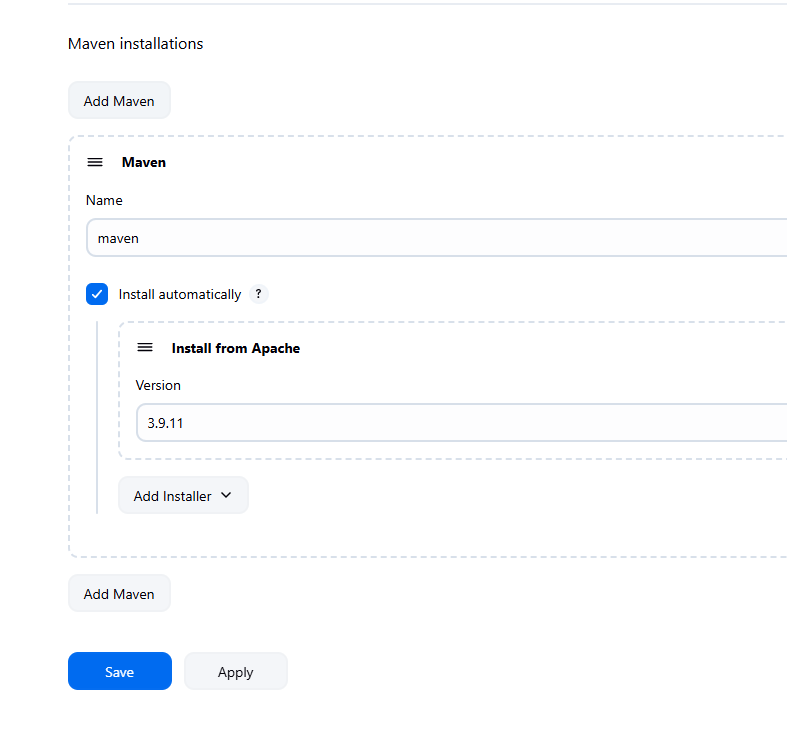
UserName and Password of DockerHub in credentials section

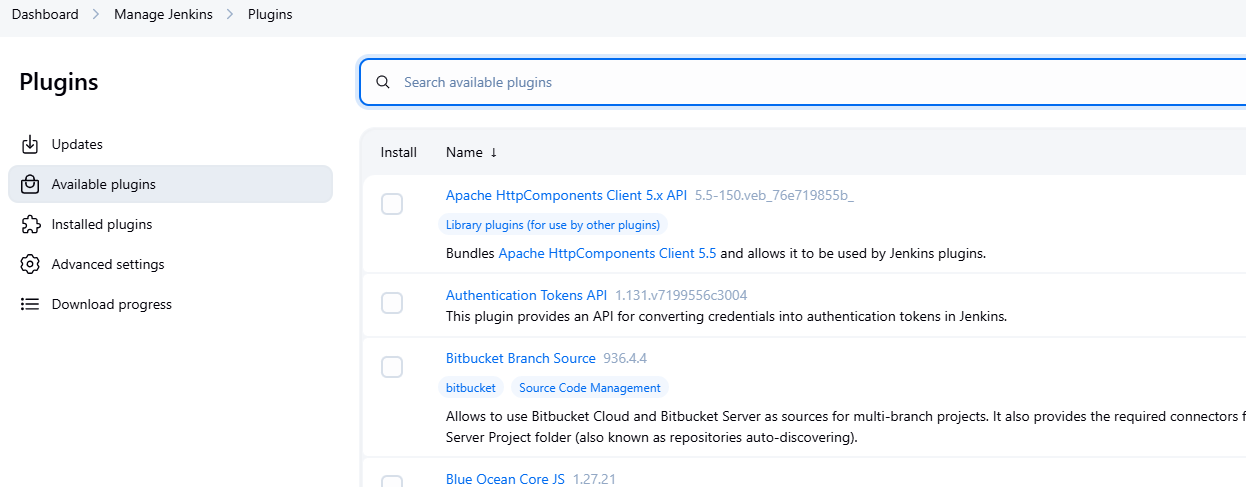
Terraform scripts

<https://github.com/SaiGit-source/DevopsWithAWS_Course/tree/main/Jenkins/Pipeline/Terraform>

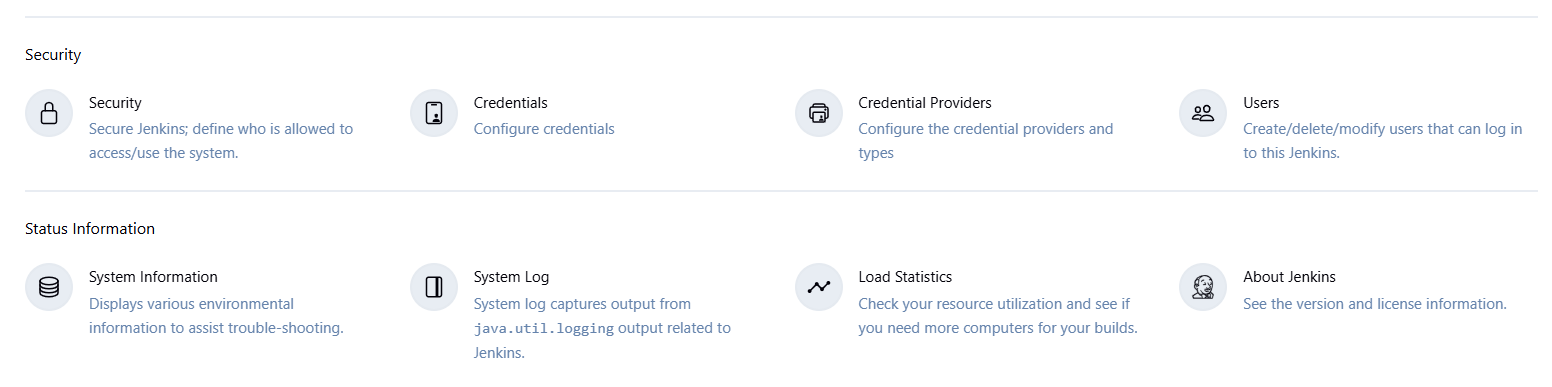
3 --> Jenkins Pipeline to create EKS cluster with Terraform scripts



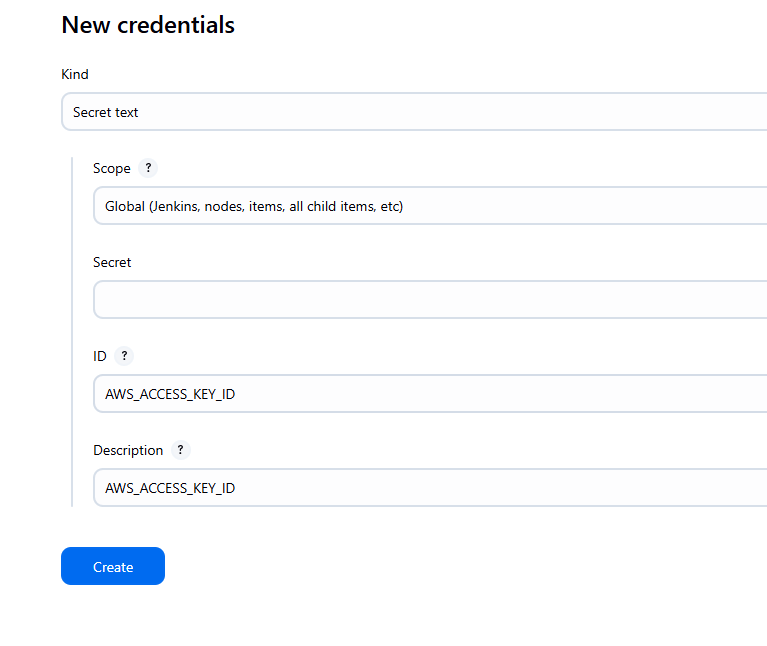




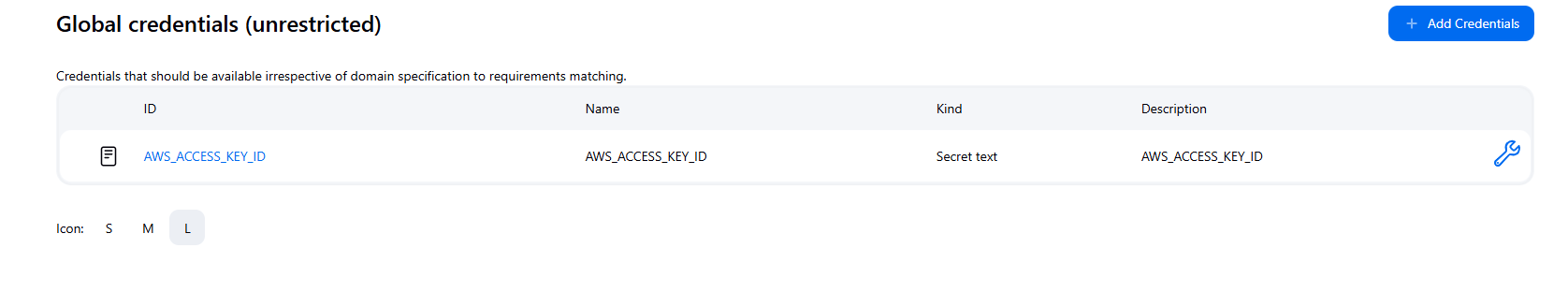
We are writing a Terraform script in order to start our EKS cluster. This Jenkins pipeline will clone the Terraform script. This pipeline is responsible to take the Terraform script (Plan, Execute) the Terraform script.



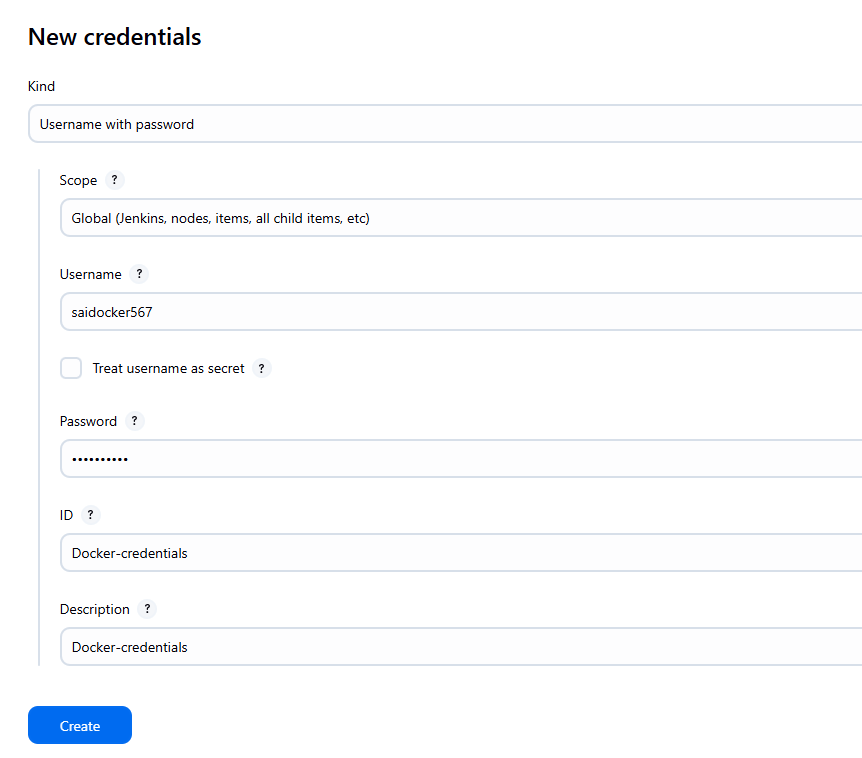
Click Credentials --> click on (global)

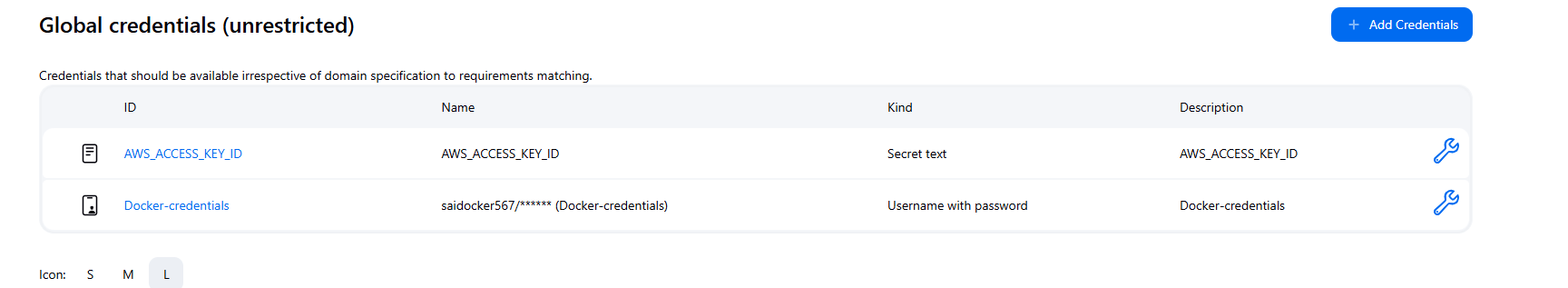


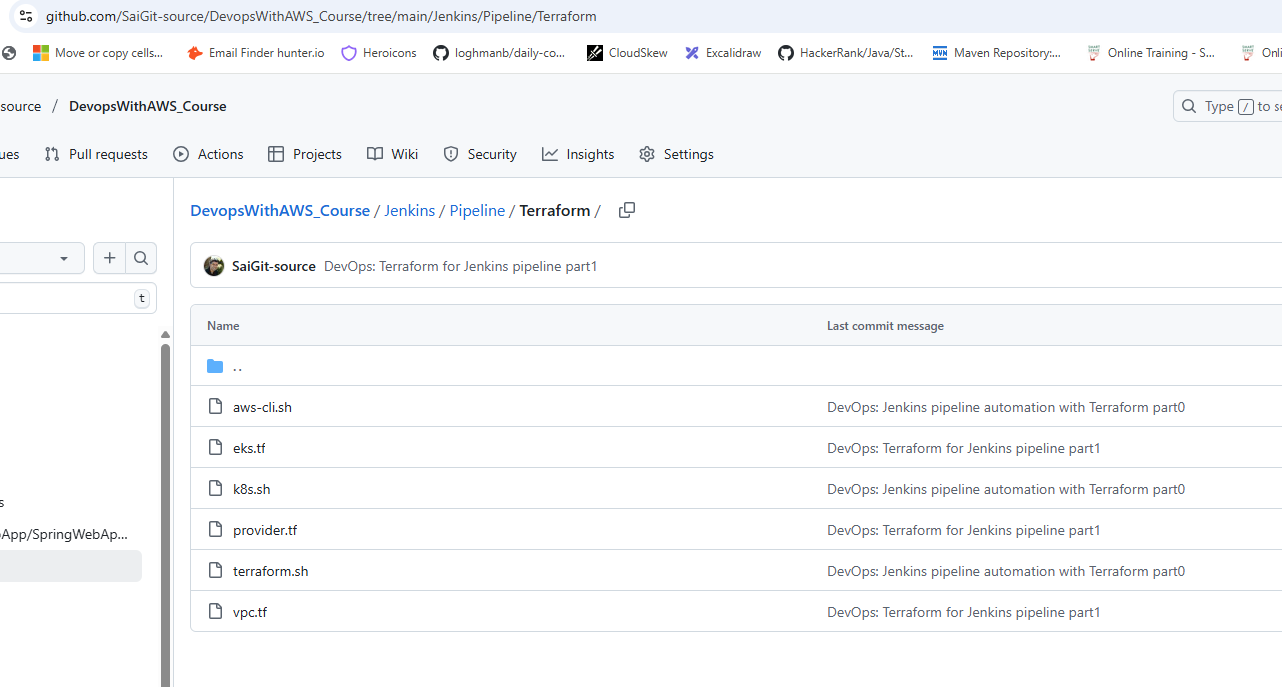
Secret is the password created in Jenkins at the start



Click Add Credentials for Docker



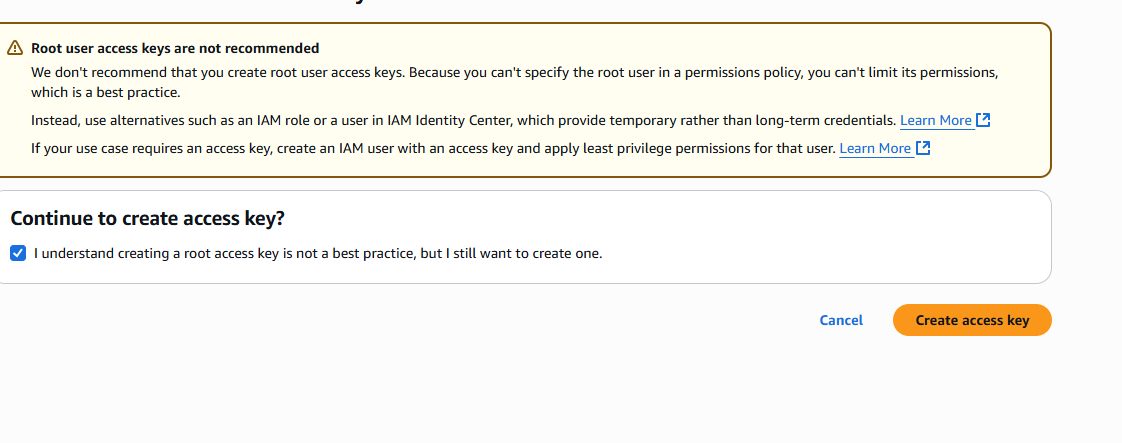


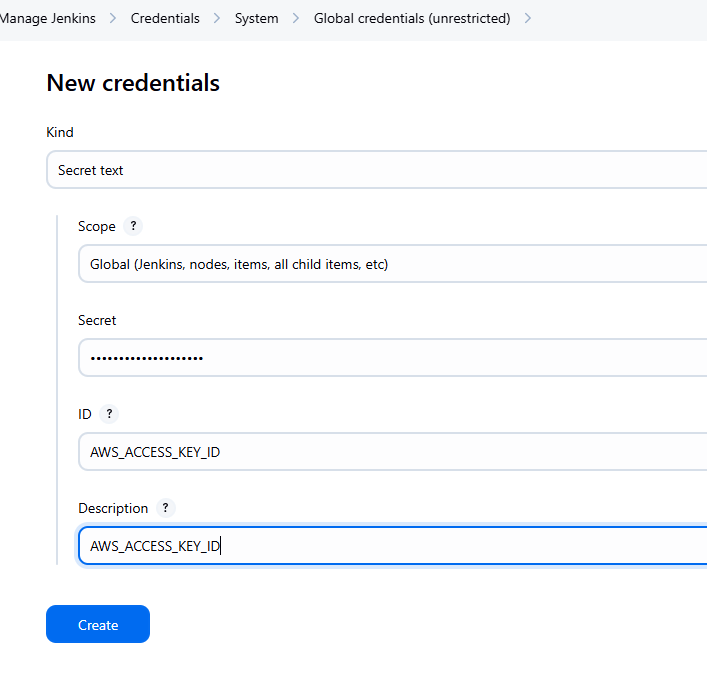


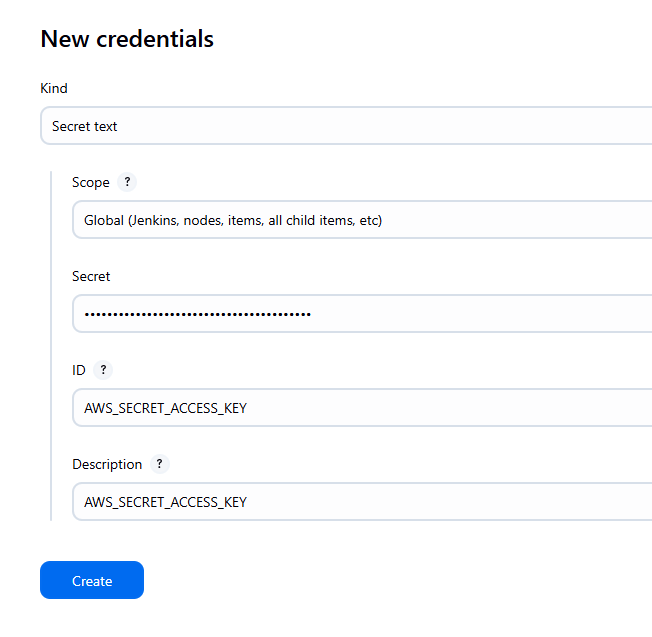
Terraform files are on Git

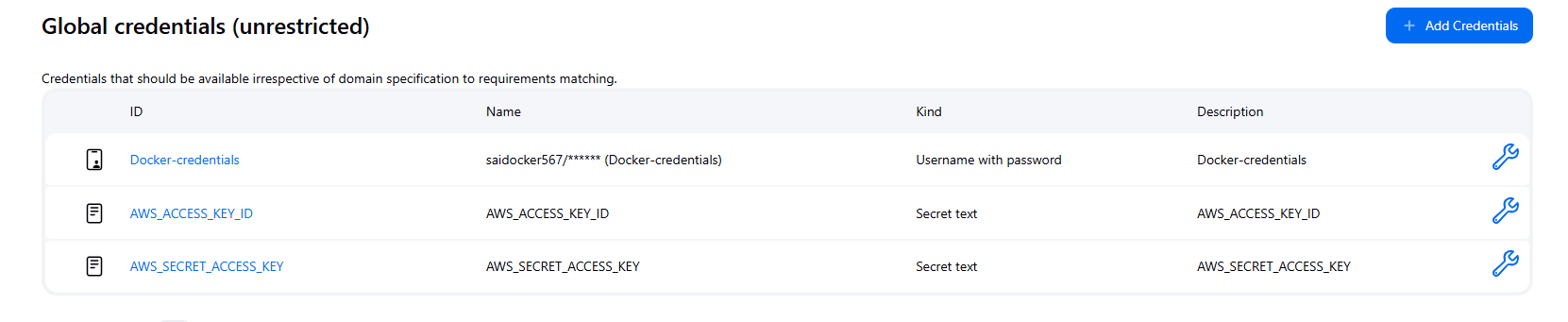
<https://github.com/SaiGit-source/DevopsWithAWS_Course/tree/main/Jenkins/Pipeline/Terraform>

Create access key









3 ==> Jenkins Pipeline to create eks cluster with terraform scripts

==============================================================================

pipeline {

agent any

environment {

AWS\_ACCESS\_KEY\_ID = credentials('AWS\_ACCESS\_KEY\_ID')

AWS\_SECRET\_ACCESS\_KEY = credentials('AWS\_SECRET\_ACCESS\_KEY')

AWS\_DEFAULT\_REGION = 'ca-central-1'

}

stages{

stage('Checkout SCM'){

steps{

script{

git branch: 'main', url: 'https://github.com/Haider7214/terraform2.git'

}

}

}

stage('Initializing Terraform'){

steps{

script{

dir('terraform'){

sh 'terraform init -upgrade'

}

}

}

}

stage('Validating Terraform'){

steps{

script{

dir('terraform'){

sh 'terraform validate'

}

}

}

}

stage('Previewing the infrastructure to be created'){

steps{

script{

dir('terraform'){

sh 'terraform plan'

}

}

}

}

stage('Create/Destroy an EKS cluster'){

steps{

script{

dir('terraform'){

sh 'terraform apply --auto-approve'

}

}

}

}

}

}

Install node in Jenkins server ( to run react ap in pipeline)

curl -fsSL https://deb.nodesource.com/setup\_18.x | sudo -E bash -

sudo apt-get install -y nodejs

node -v

npm -v

Install Docker in Jenkins server

sudo apt-get update

sudo apt-get upgrade -y

sudo apt-get install \

ca-certificates \

curl \

gnupg \

lsb-release -y

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

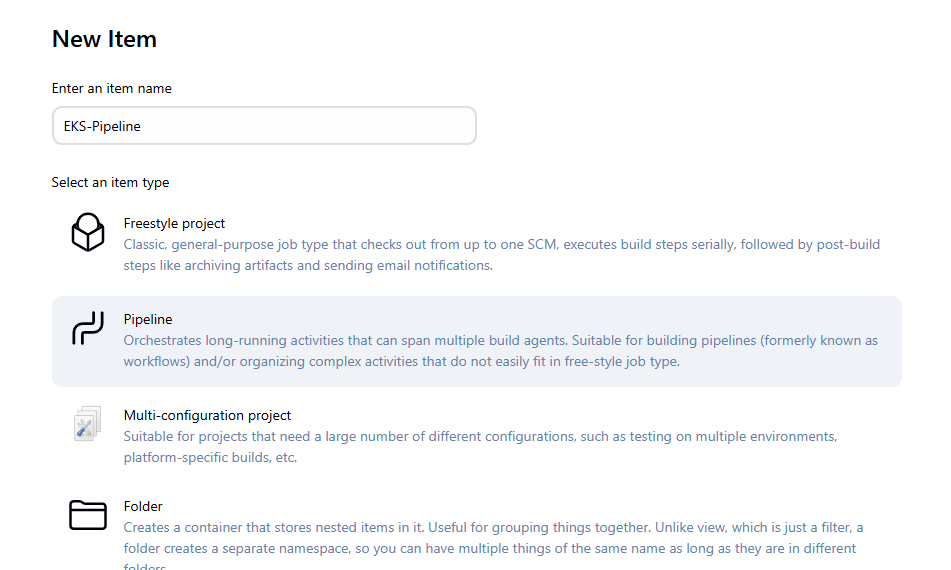
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

sudo usermod -aG docker jenkins

sudo systemctl restart jenkins

docker --version

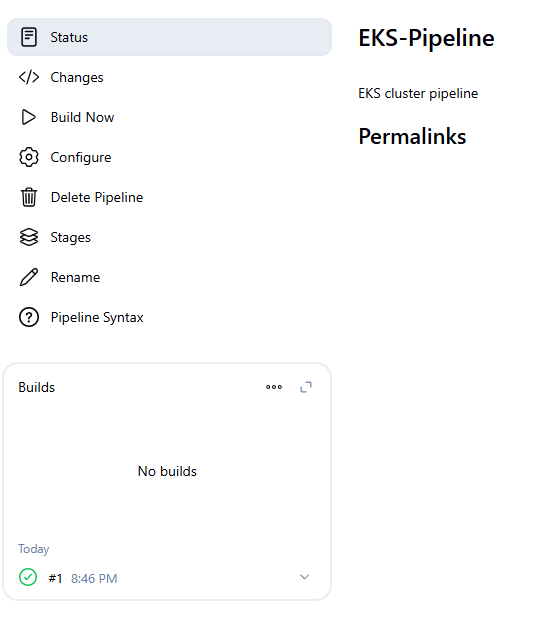




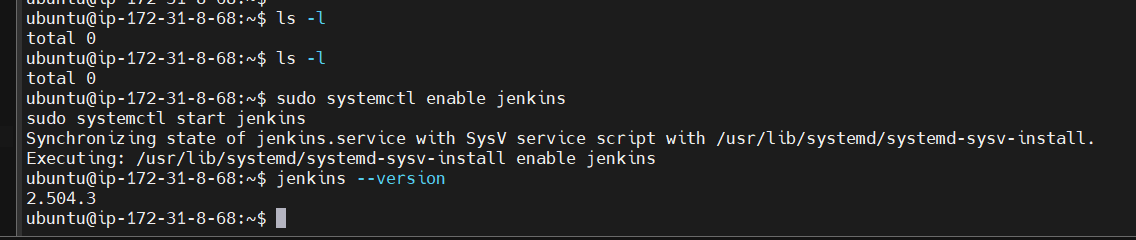
36:36

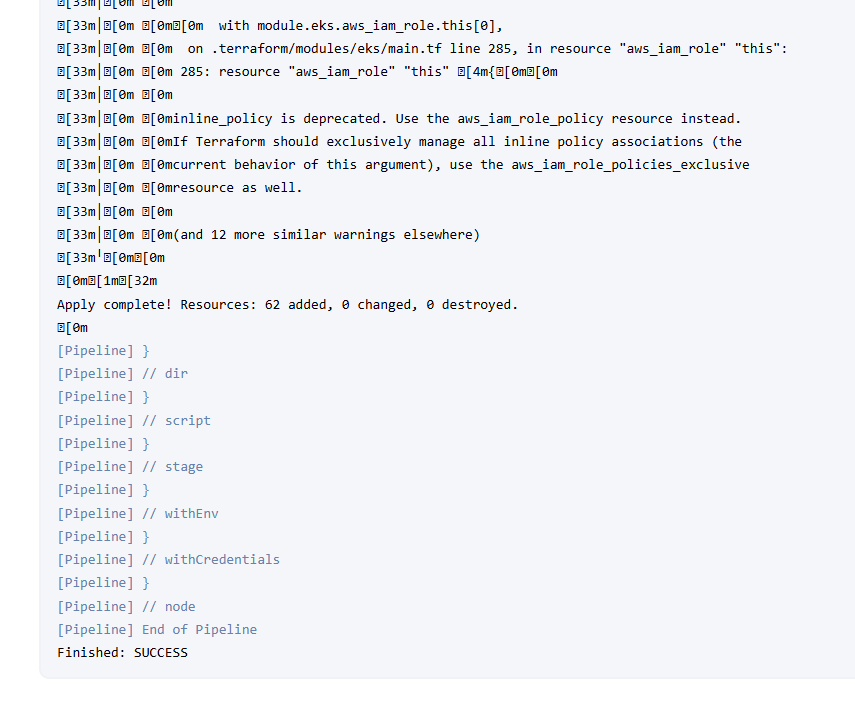
Apply and Save

Build Now



Restarting Jenkins server





Install node in Jenkins server ( to run react app in pipeline)

curl -fsSL https://deb.nodesource.com/setup\_18.x | sudo -E bash -

sudo apt-get install -y nodejs

node -v

npm -v

Install Docker in Jenkins server

sudo apt-get update

sudo apt-get upgrade -y

sudo apt-get install \

ca-certificates \

curl \

gnupg \

lsb-release -y

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

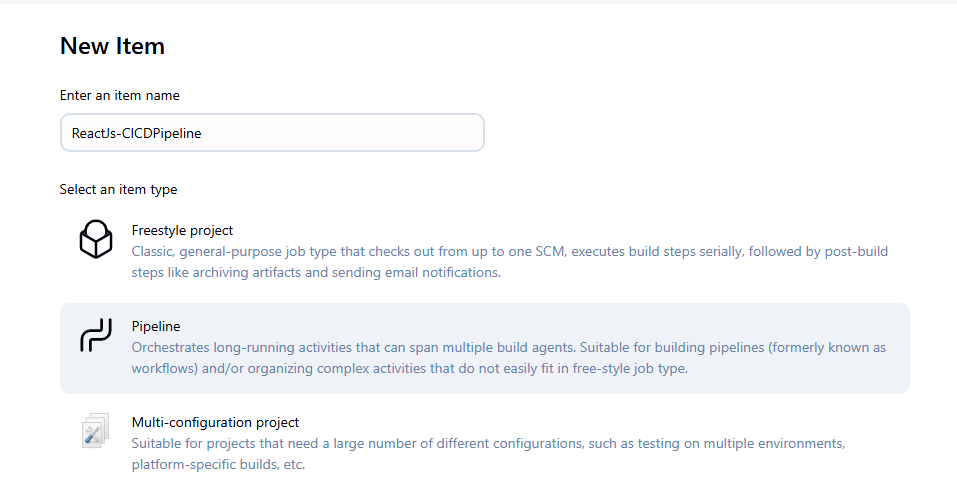
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

sudo usermod -aG docker jenkins

sudo systemctl restart jenkins

docker --version



React Js CICD Pipeline ==> Jenkins

====================================================================

pipeline {

agent any

environment {

REGISTRY = 'Docker-credentials' // Docker Hub credentials ID in Jenkins

KUBECONFIG = '/var/lib/jenkins/.kube/config' // Path to kubeconfig on Jenkins server

}

stages {

stage('Checkout') {

steps {

git branch: 'main', url: 'https://github.com/Gaurav560/student-management-system-frontend.git'

}

}

stage('Create Dockerfile and K8s Manifest') {

steps {

script {

// ✅ Fixed Dockerfile (changed /app/build to /app/dist)

writeFile file: 'Dockerfile', text: '''

FROM node:18-alpine AS build

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

FROM nginx:alpine

COPY --from=build /app/dist /usr/share/nginx/html

EXPOSE 80

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

'''

// Kubernetes Deployment YAML

writeFile file: 'react-deployment.yaml', text: """

apiVersion: apps/v1

kind: Deployment

metadata:

name: react-deploy

spec:

replicas: 2

selector:

matchLabels:

app: react-app

template:

metadata:

labels:

app: react-app

spec:

containers:

- name: react-container

image: haidertelusko/react-app:${env.BUILD\_NUMBER}

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: react-service

spec:

selector:

app: react-app

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

"""

}

}

}

stage('Build React App') {

steps {

sh 'npm install'

sh 'npm run build'

}

}

stage('Docker Build & Push') {

steps {

script {

def imageTag = "haidertelusko/react-app:${env.BUILD\_NUMBER}"

docker.build(imageTag, '.')

docker.withRegistry('', REGISTRY) {

docker.image(imageTag).push()

docker.image(imageTag).push('latest')

}

}

}

}

stage('Deploy to Kubernetes') {

steps {

sh """

export KUBECONFIG=${env.KUBECONFIG}

kubectl apply -f react-deployment.yaml

kubectl set image deployment/react-deploy react-container=haidertelusko/react-app:${env.BUILD\_NUMBER}

kubectl rollout status deployment/react-deploy

"""

}

}

}

}



Kubernetes part is removed

KUBECONFIG = '/var/lib/jenkins/.kube/config' // Path to kubeconfig on Jenkins server

Remove Kubernetes writeFile part also

pipeline {

agent any

environment {

REGISTRY = 'Docker-credentials' // Docker Hub credentials ID in Jenkins

}

stages {

stage('Checkout') {

steps {

git branch: 'main', url: 'https://github.com/Gaurav560/student-management-system-frontend.git'

}

}

stage('Create Dockerfile and K8s Manifest') {

steps {

script {

// ✅ Fixed Dockerfile (changed /app/build to /app/dist)

writeFile file: 'Dockerfile', text: '''

FROM node:18-alpine AS build

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

FROM nginx:alpine

COPY --from=build /app/dist /usr/share/nginx/html

EXPOSE 80

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

'''

}

}

}

stage('Build React App') {

steps {

sh 'npm install'

sh 'npm run build'

}

}

stage('Docker Build & Push') {

steps {

script {

def imageTag = "saidocker567/react-app:${env.BUILD\_NUMBER}"

docker.build(imageTag, '.')

docker.withRegistry('', REGISTRY) {

docker.image(imageTag).push()

docker.image(imageTag).push('latest')

}

}

}

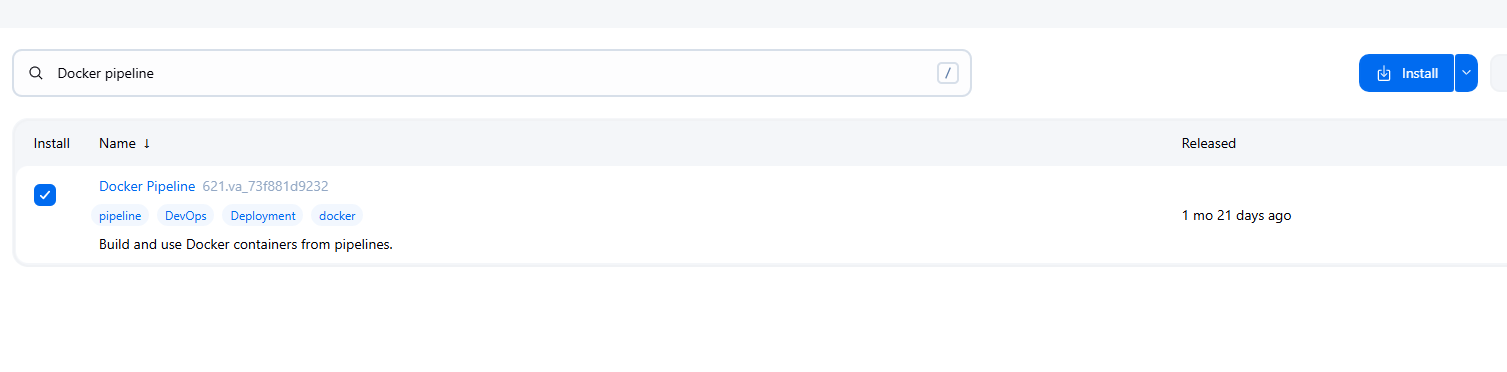
}

}

}

We need to install Docker Pipeline plugin, we need it for a particular function

Manage Jenkins --> Plugins



pipeline {

agent any

environment {

REGISTRY = 'Docker-credentials' // Docker Hub credentials ID in Jenkins

}

stages {

stage('Checkout') {

steps {

git branch: 'main', url: 'https://github.com/Gaurav560/student-management-system-frontend.git'

}

}

stage('Create Dockerfile and K8s Manifest') {

steps {

script {

// ✅ Fixed Dockerfile (changed /app/build to /app/dist)

writeFile file: 'Dockerfile', text: '''

FROM node:18-alpine AS build

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

FROM nginx:alpine

COPY --from=build /app/dist /usr/share/nginx/html

EXPOSE 80

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

'''

}

}

}

stage('Docker Build & Push') {

steps {

script {

def imageTag = "saidocker567/react-app:${env.BUILD\_NUMBER}"

docker.build(imageTag, '.')

docker.withRegistry('', REGISTRY) {

docker.image(imageTag).push()

docker.image(imageTag).push('latest')

}

}

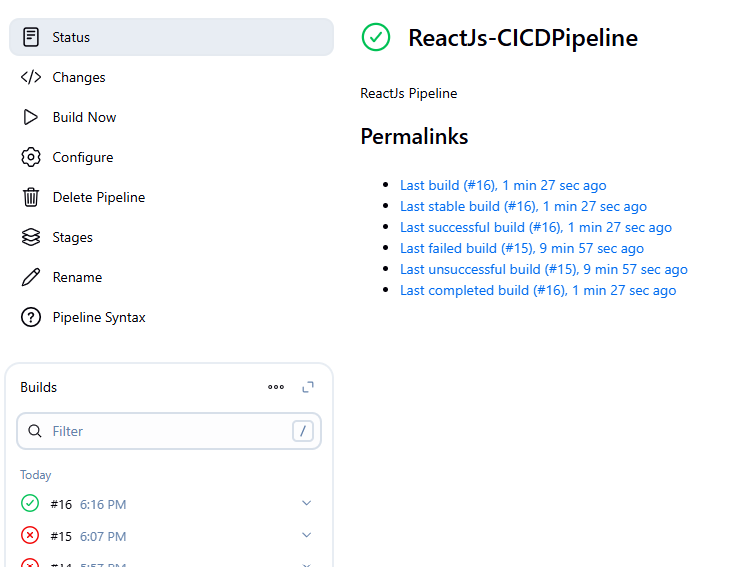
}

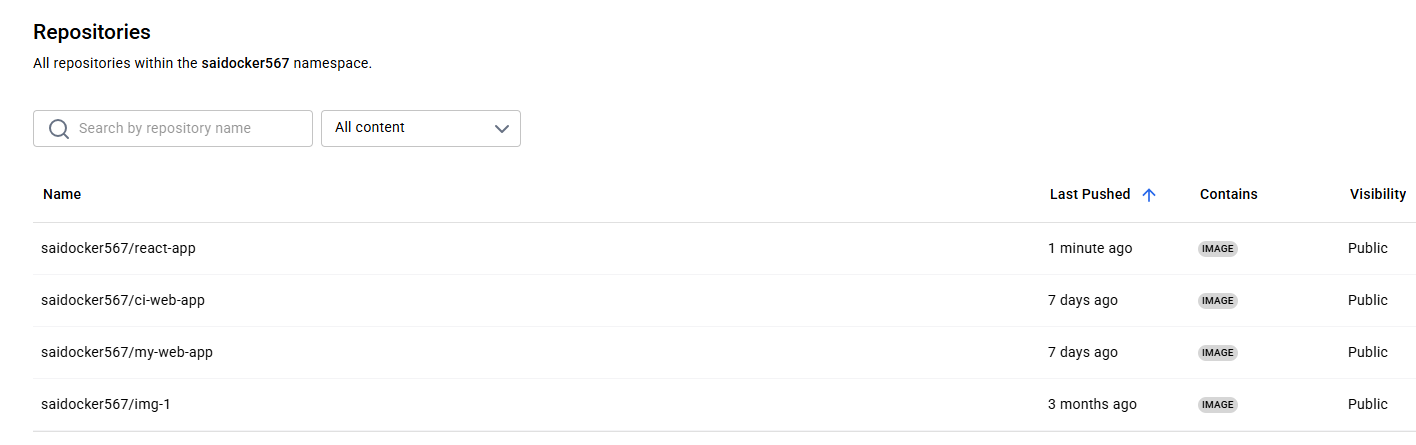
}

}

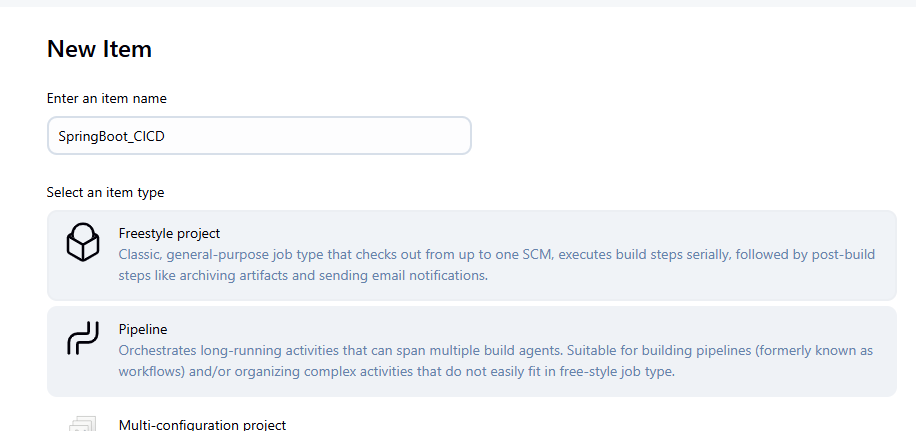
}

Restarted Jenkins server and it worked





Write a Pipeline for SpringBoot application. If SpringBoot application is deployed, we have to make changes to React application, we got to re-run the pipeline



CICD Jenkins for backend spring boot app

=========================================================================

pipeline {

agent any

environment {

REGISTRY = 'Docker-credentials' // Docker Hub credentials ID in Jenkins

KUBECONFIG = '/var/lib/jenkins/.kube/config' // Path to kubeconfig on Jenkins server

}

stages {

stage('Checkout') {

steps {

git branch: 'main', url: 'https://github.com/Gaurav560/SMProject.git'

}

}

stage('Create Dockerfile and K8s Manifest') {

steps {

script {

// ✅ Dockerfile for Spring Boot inside subdirectory

writeFile file: 'Dockerfile', text: '''

FROM openjdk:17-alpine

VOLUME /tmp

COPY BackEndSMProject/target/\*.jar app.jar

ENTRYPOINT ["java","-jar","/app.jar"]

'''

// ✅ Kubernetes Deployment and Service YAML

writeFile file: 'spring-deployment.yaml', text: """

apiVersion: apps/v1

kind: Deployment

metadata:

name: springboot-deploy

spec:

replicas: 2

selector:

matchLabels:

app: springboot-app

template:

metadata:

labels:

app: springboot-app

spec:

containers:

- name: springboot-container

image: saidocker567/springboot-app:${env.BUILD\_NUMBER}

ports:

- containerPort: 8080

env:

- name: SPRING\_PROFILES\_ACTIVE

value: "prod"

---

apiVersion: v1

kind: Service

metadata:

name: springboot-service

spec:

selector:

app: springboot-app

ports:

- protocol: TCP

port: 8080

targetPort: 8080

type: LoadBalancer

"""

}

}

}

stage('Build Spring Boot App') {

steps {

// Fix permissions then build inside subdirectory

sh '''

cd BackEndSMProject

chmod +x mvnw

./mvnw clean package -DskipTests

'''

}

}

stage('Docker Build & Push') {

steps {

script {

def imageTag = "saidocker567/springboot-app:${env.BUILD\_NUMBER}"

docker.build(imageTag, '.')

docker.withRegistry('', REGISTRY) {

docker.image(imageTag).push()

docker.image(imageTag).push('latest')

}

}

}

}

stage('Deploy to Kubernetes') {

steps {

sh """

export KUBECONFIG=${env.KUBECONFIG}

kubectl apply -f spring-deployment.yaml

kubectl set image deployment/springboot-deploy springboot-container=saidocker567/springboot-app:${env.BUILD\_NUMBER}

kubectl rollout status deployment/springboot-deploy

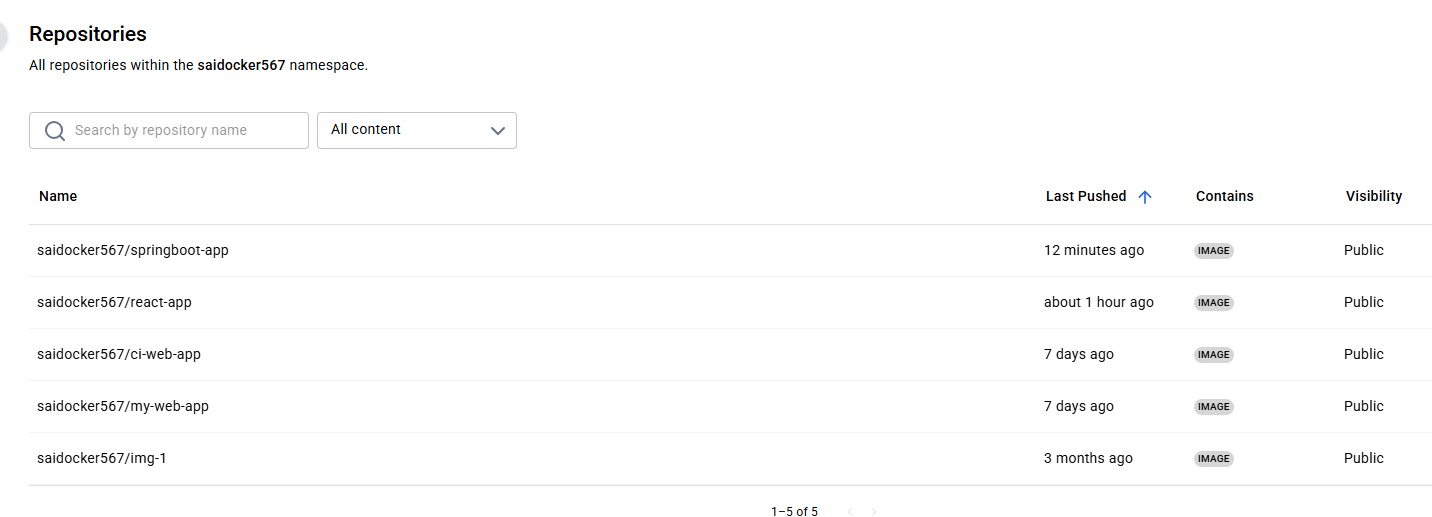
"""

}

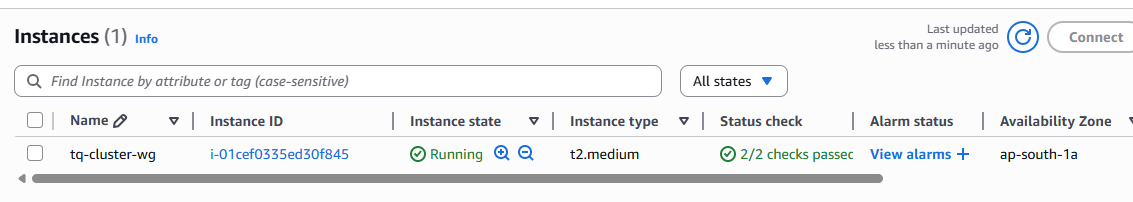
}

}

}



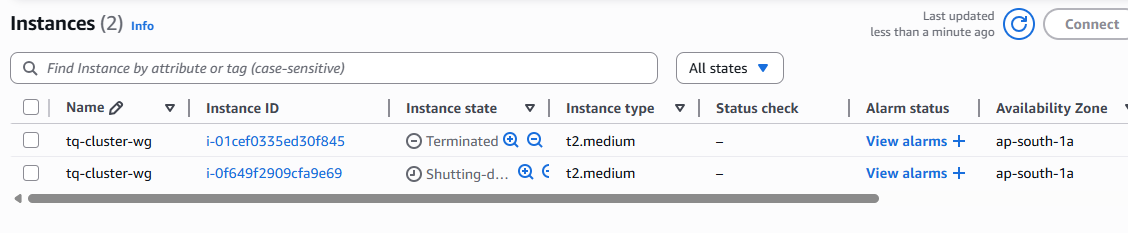
Problem is EKS-pipeline created all resources in ap-south-1 region, we want in ca-central-region



Using same pipeline to destory resources in EKS-pipeline



We can see it is being terminated in EKS-pipeline



Updating git repo in EKS-pipeline

pipeline {

agent any

environment {

AWS\_ACCESS\_KEY\_ID = credentials('AWS\_ACCESS\_KEY\_ID')

AWS\_SECRET\_ACCESS\_KEY = credentials('AWS\_SECRET\_ACCESS\_KEY')

AWS\_DEFAULT\_REGION = 'ca-central-1'

}

stages{

stage('Checkout SCM'){

steps{

script{

git branch: 'main', url: 'https://github.com/SaiGit-source/Terraform2.git'

}

}

}

stage('Initializing Terraform'){

steps{

script{

dir('terraform'){

sh 'terraform init -upgrade'

}

}

}

}

stage('Validating Terraform'){

steps{

script{

dir('terraform'){

sh 'terraform validate'

}

}

}

}

stage('Previewing the infrastructure to be created'){

steps{

script{

dir('terraform'){

sh 'terraform plan'

}

}

}

}

stage('Create/Destroy an EKS cluster'){

steps{

script{

dir('terraform'){

sh 'terraform apply --auto-approve'

}

}

}

}

}

}

No matter what you do, it creates clusters and instances in ap-south-1 region only, the problem is we have our EKS proxy server in ca-central-1. So lets try with that

pipeline {

agent any

environment {

AWS\_ACCESS\_KEY\_ID = credentials('AWS\_ACCESS\_KEY\_ID')

AWS\_SECRET\_ACCESS\_KEY = credentials('AWS\_SECRET\_ACCESS\_KEY')

AWS\_DEFAULT\_REGION = 'ap-south-1'

}

stages{

stage('Checkout SCM'){

steps{

script{

git branch: 'main', url: 'https://github.com/Haider7214/terraform2.git'

}

}

}

stage('Initializing Terraform'){

steps{

script{

dir('terraform'){

sh 'terraform init -upgrade'

}

}

}

}

stage('Validating Terraform'){

steps{

script{

dir('terraform'){

sh 'terraform validate'

}

}

}

}

stage('Previewing the infrastructure to be created'){

steps{

script{

dir('terraform'){

sh 'terraform plan'

}

}

}

}

stage('Create/Destroy an EKS cluster'){

steps{

script{

dir('terraform'){

sh 'terraform apply --auto-approve'

}

}

}

}

}

}

1:23:30

Update EKS Cluster Config File in Jenkins Server

=========================================================

Execute the below command in EKS Management Machine and Copy config file data

$ cat .kube/config

Connect to Jenkins server execute the following command to add config file into Jenkins server

$ cd /var/lib/jenkins

$ sudo mkdir .kube

$ sudo vi .kube/config

( paste config file data copied from eks host machine )

Check eks nodes

$ kubectl get nodes

$ cd ~

$ ls -la

$ sudo vi .kube/config

( paste config file data copied from eks host machine )

$ kubectl get nodes

( Follow live class instructions)

===================================================================

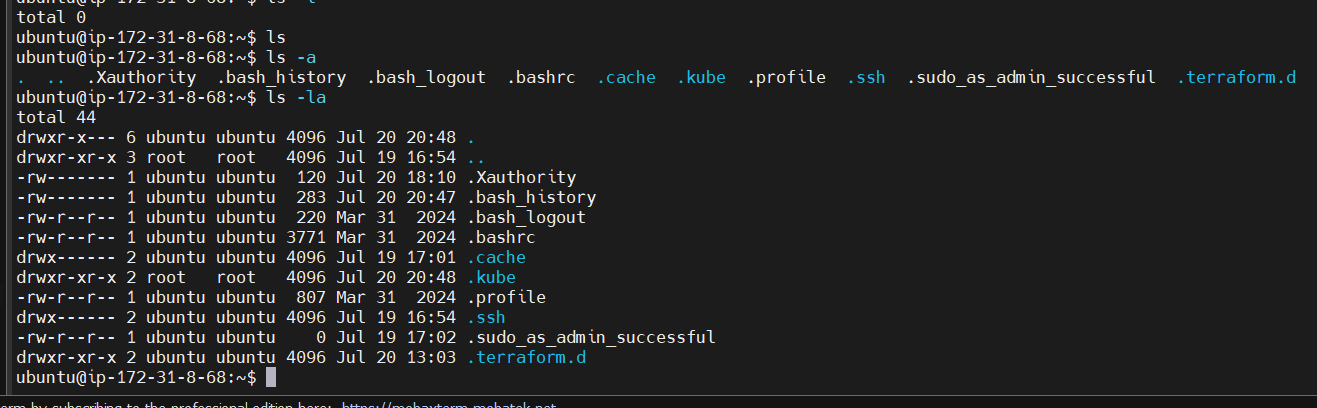
Once CICD React is success : kubectl get svc react-service

=================================================================

github repo https://github.com/teluskoOrg/DemoCiCd.git

demo for aws services for CICD

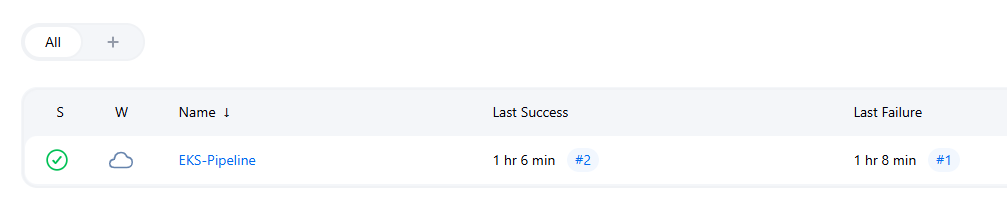
In the Jenkins-server



1:24:37

I ran the Terraform scripts (EKS-pipeline) from the TerraformFinal machine

Then the following commands to generate config file

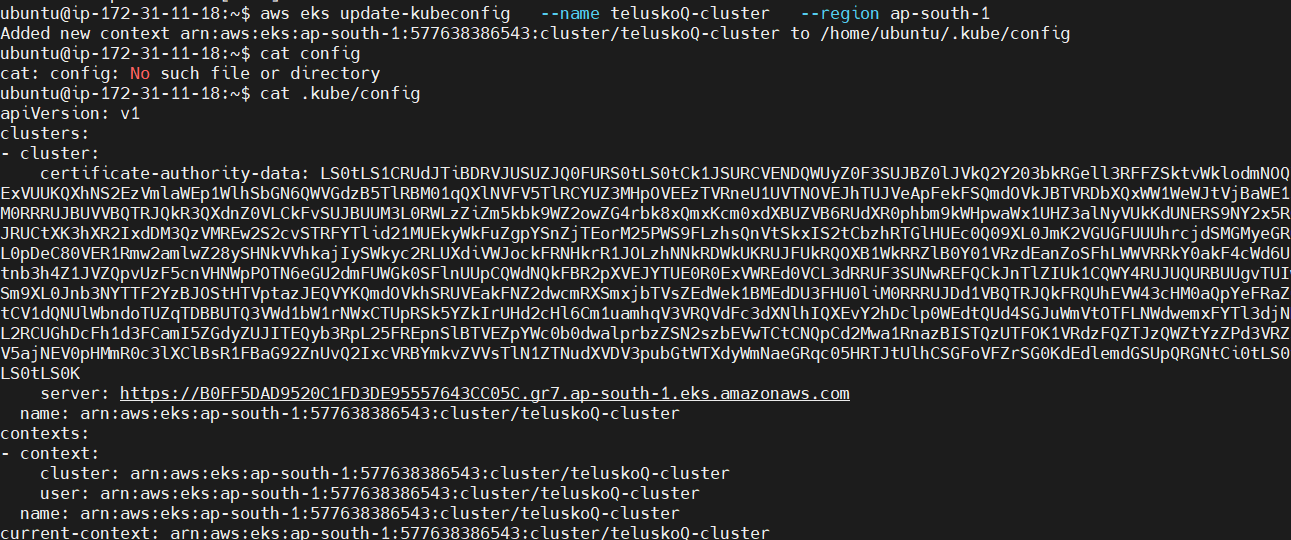


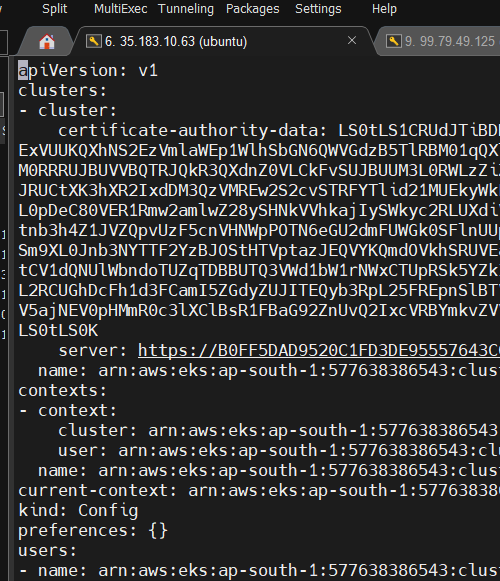
Generate config file

ubuntu@ip-172-31-11-18:~$ aws eks update-kubeconfig --name teluskoQ-cluster --region ap-south-1

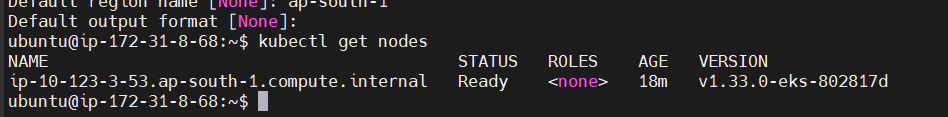
Added new context arn:aws:eks:ap-south-1:577638386543:cluster/teluskoQ-cluster to /home/ubuntu/.kube/config

Now we copy this info into the other machine. Jenkins-Server





Now we get nodes on the Jenkins-Server machine as well, even though the cluster is created on a different region. If required, run aws configure



We create the config file in both the places

ubuntu@ip-172-31-8-68:~$ cd /var/lib/jenkins

ubuntu@ip-172-31-8-68:/var/lib/jenkins$ sudo mkdir .kube

ubuntu@ip-172-31-8-68:/var/lib/jenkins$ sudo vi config

ubuntu@ip-172-31-8-68:/var/lib/jenkins$

ubuntu@ip-172-31-8-68:/var/lib/jenkins/.kube$ sudo vi config

ubuntu@ip-172-31-8-68:/var/lib/jenkins/.kube$ ls -la

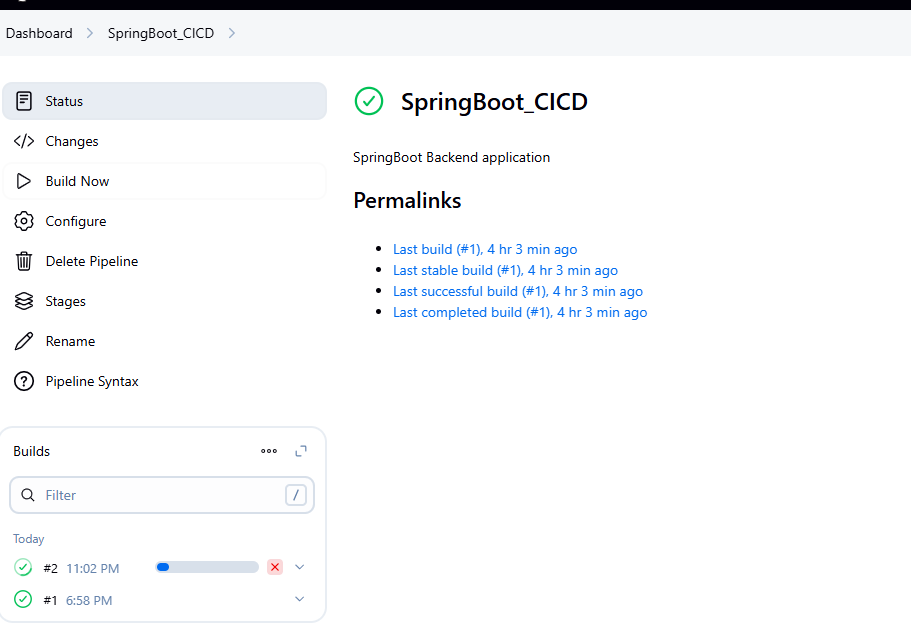
total 12

drwxr-xr-x 2 root root 4096 Jul 20 22:53 .

drwxr-xr-x 20 jenkins jenkins 4096 Jul 20 22:51 ..

-rw-r--r-- 1 root root 2367 Jul 20 22:53 config

I got to the other Server: Jenkins-Server and run SpringBootCICD pipeline



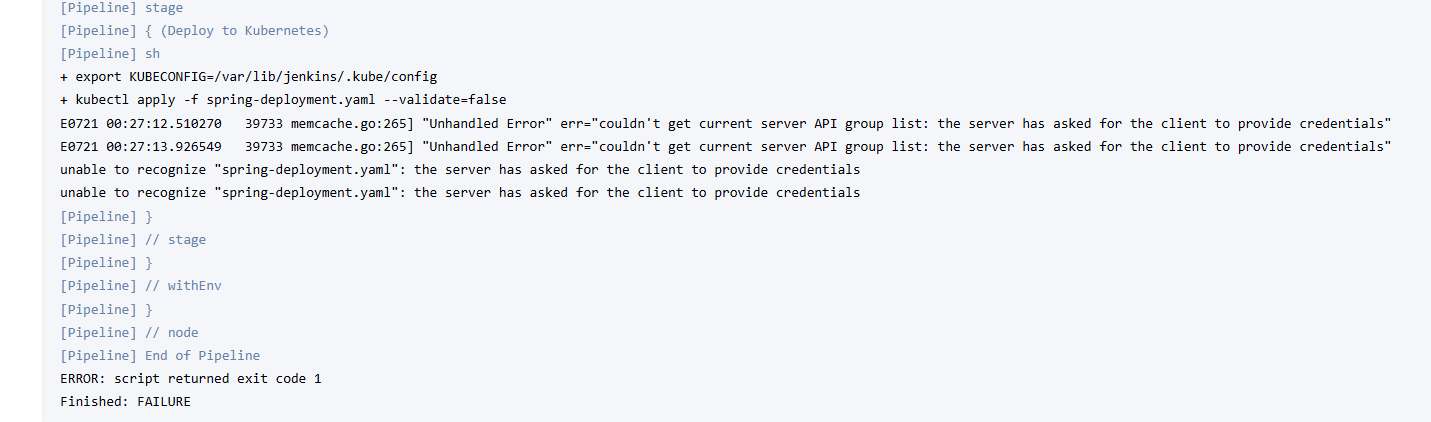
1:32:04

Now go back to ReactJs-CICDPipeline add the Kubernetes part in the Configuration

Trying

sudo chown -R jenkins:jenkins /var/lib/jenkins/.kube

Anyways, now we are going to try SpringBootCICD



Facing some issue with kubectl apply, will fix tomorrow

Destroying EKS cluster and all resources