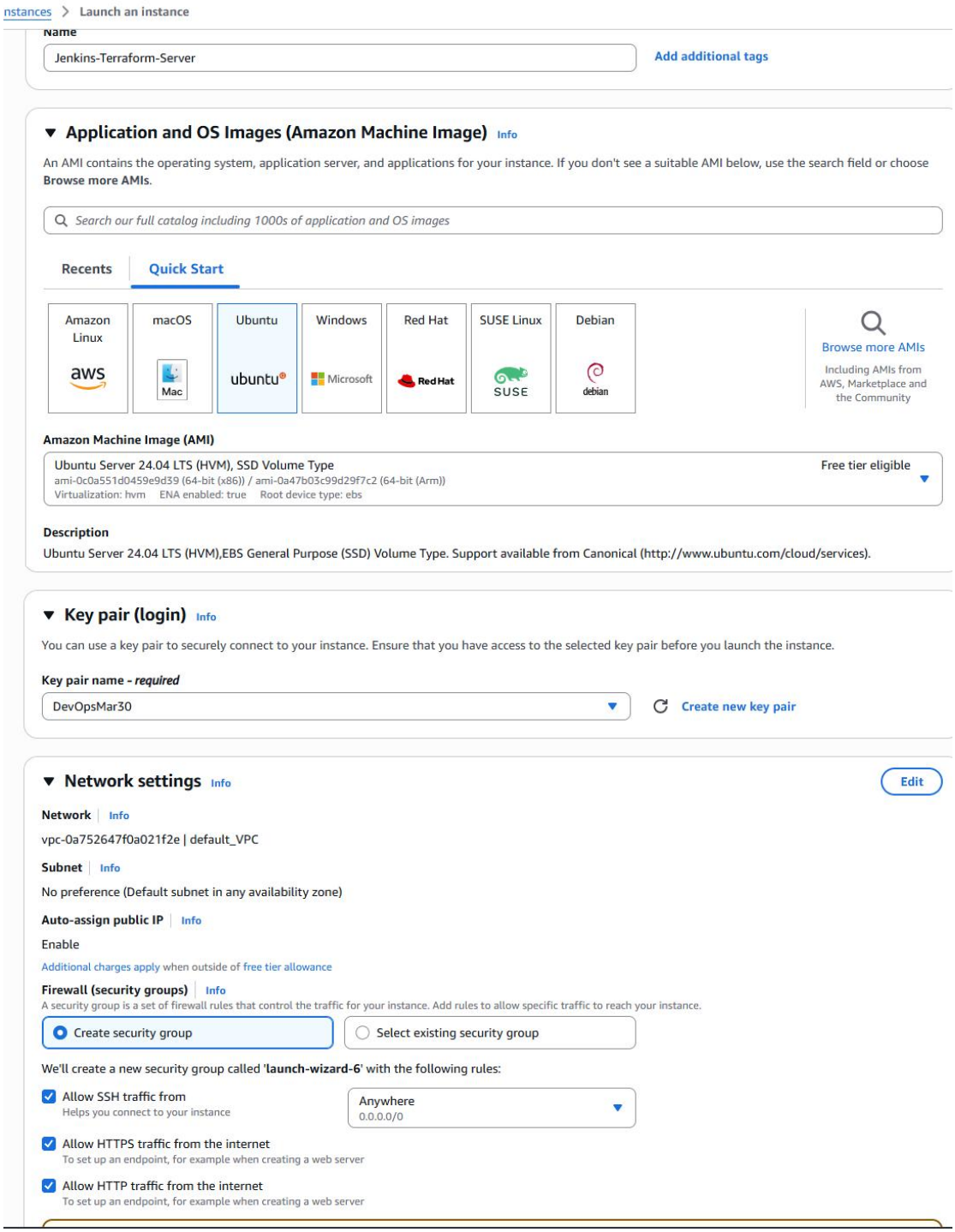
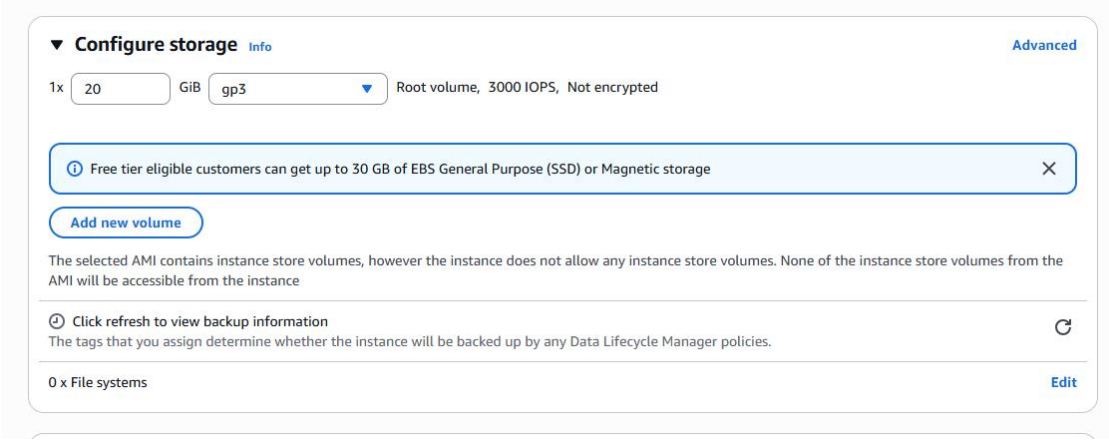
Jenkins Pipeline part 0

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





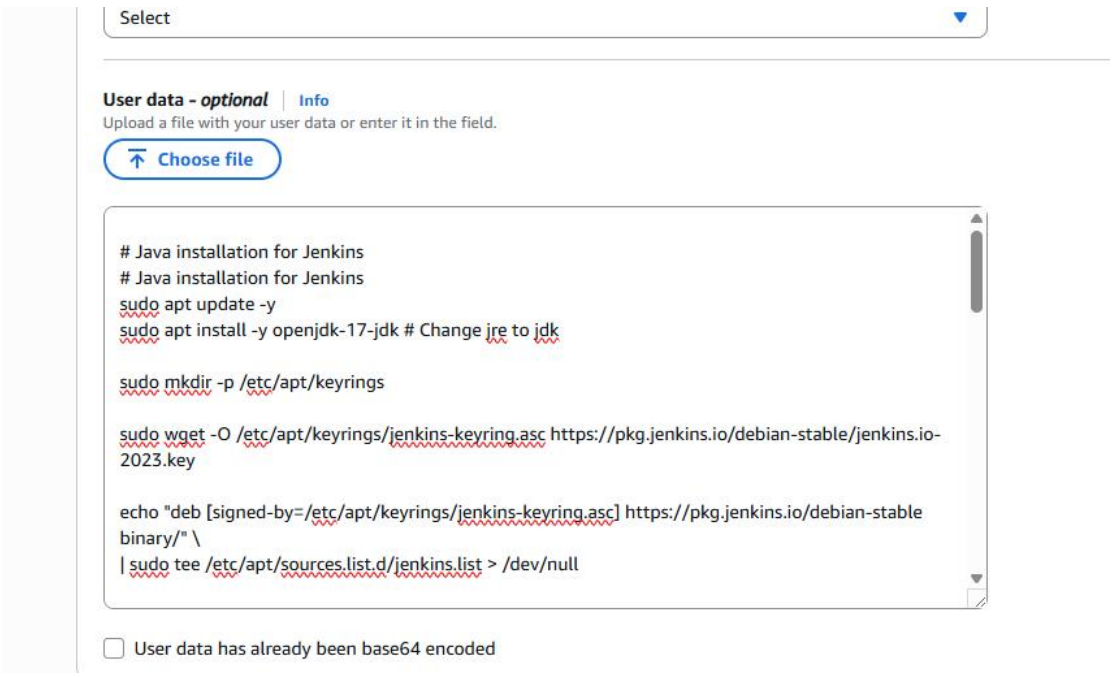
==========
Jenkins CICD react and Spring boot
==========

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

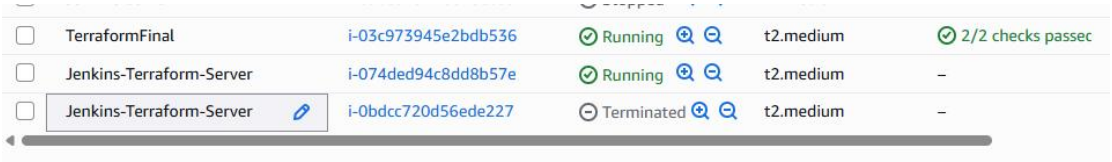
```
#!/bin/bash
set -euxo pipefail

# 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
```

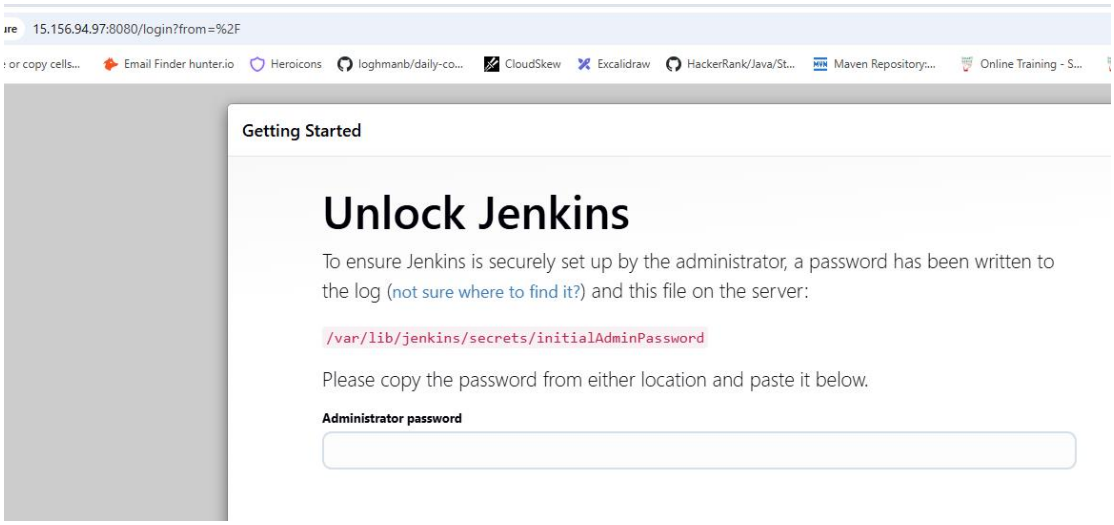
Port# 8080 for Jenkins



Copy paste



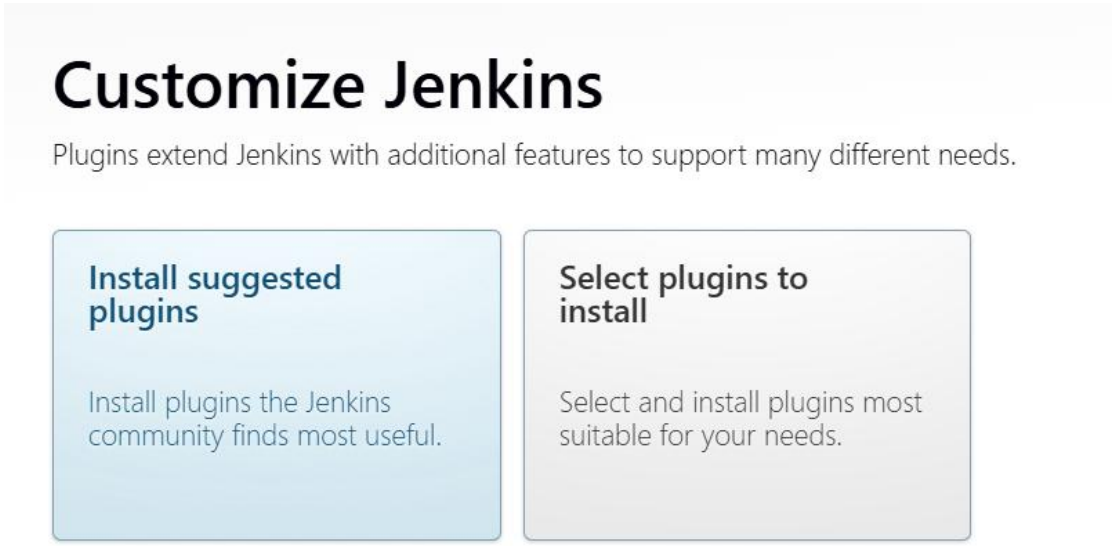
http://15.156.94.97:8080/login?from=%2F



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

Copy password and save in a file

Ins



Install suggested plugins

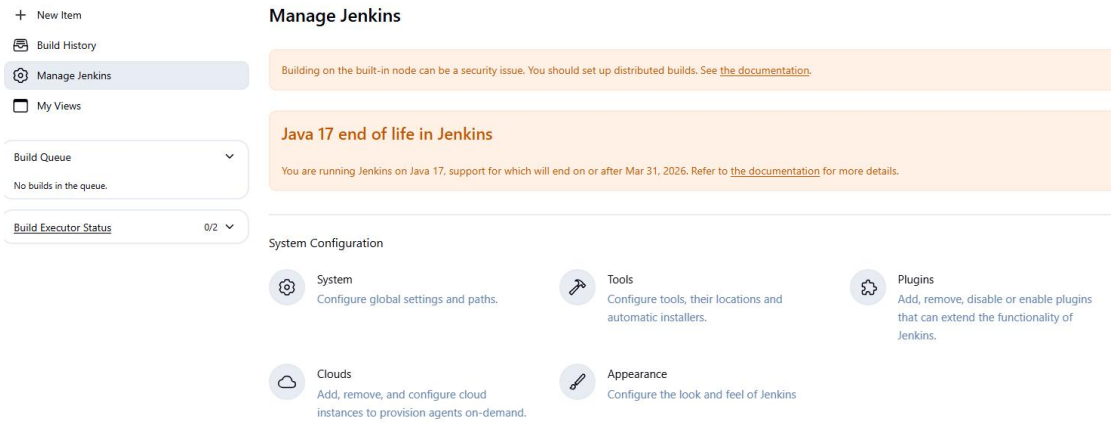
Fillout the login credentials page

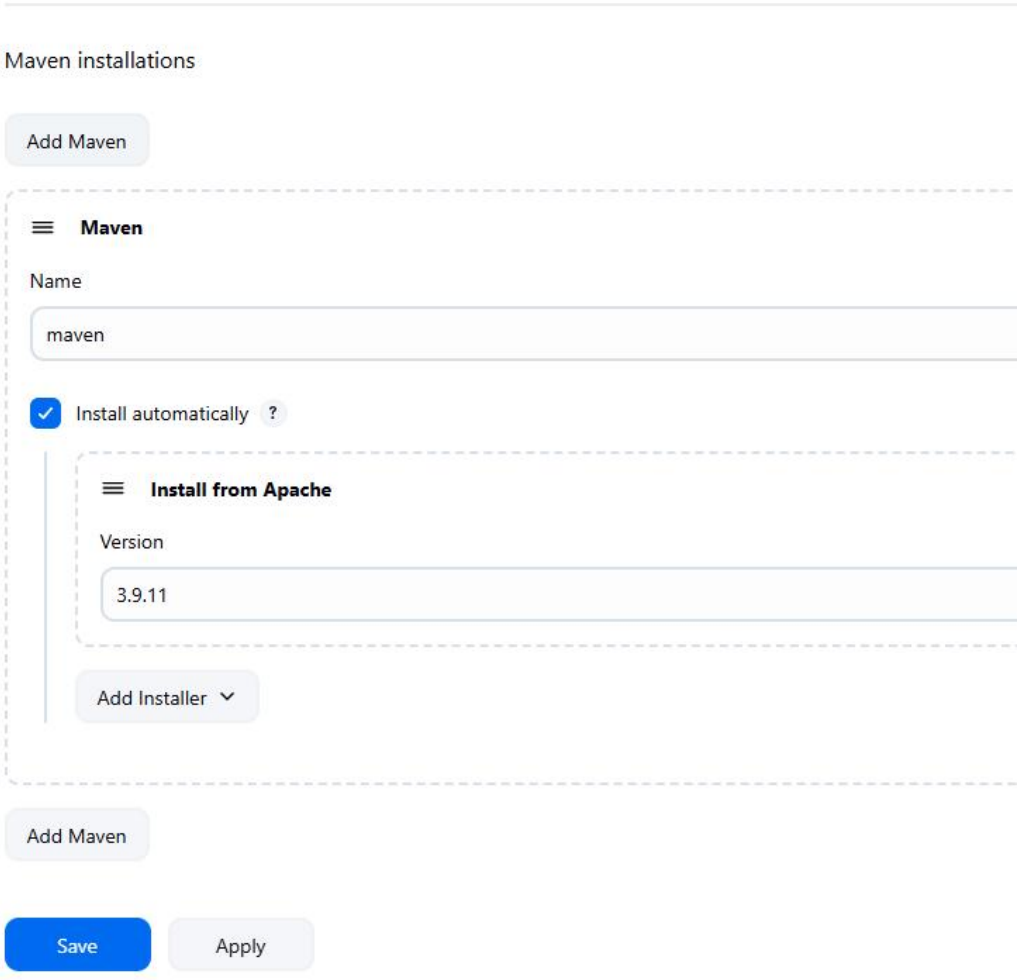
admin, ******

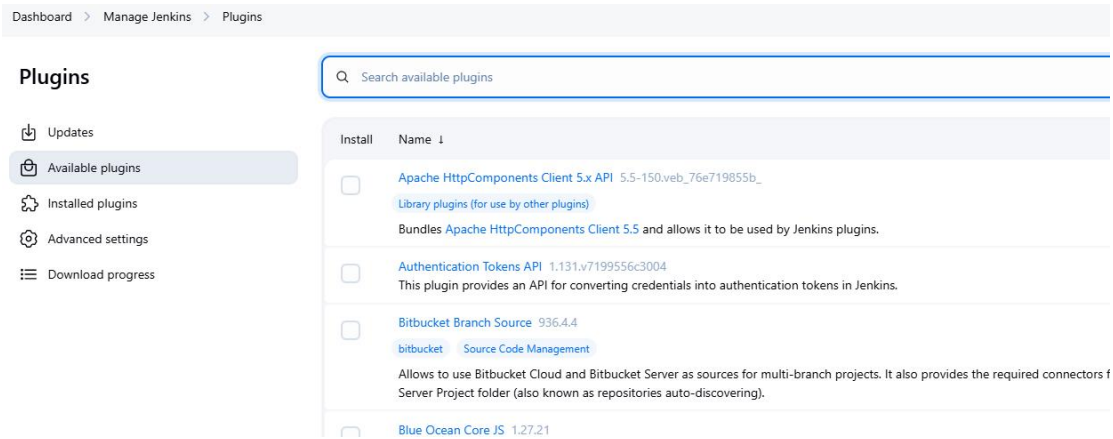
2 --> Manage Jenkins --> Credentials --> Secret Text --> AWS_ACCESS_KEY_ID and AWS_SECRET_KEY
Get AWS_ACCESS_KEY from AWS account
UserName and Password of DockerHub in credentials section

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

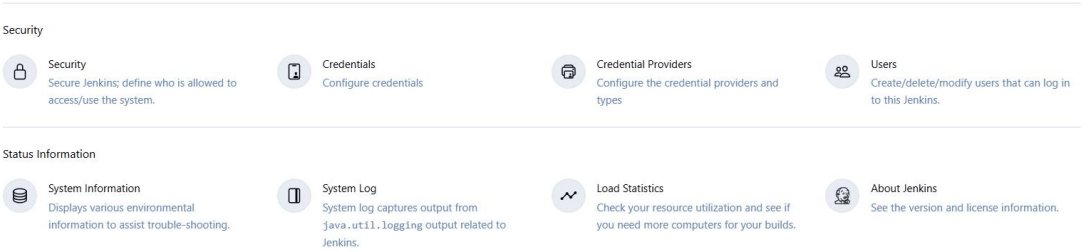
3 --> Jenkins Pipeline to create EKS cluster with Terraform scripts



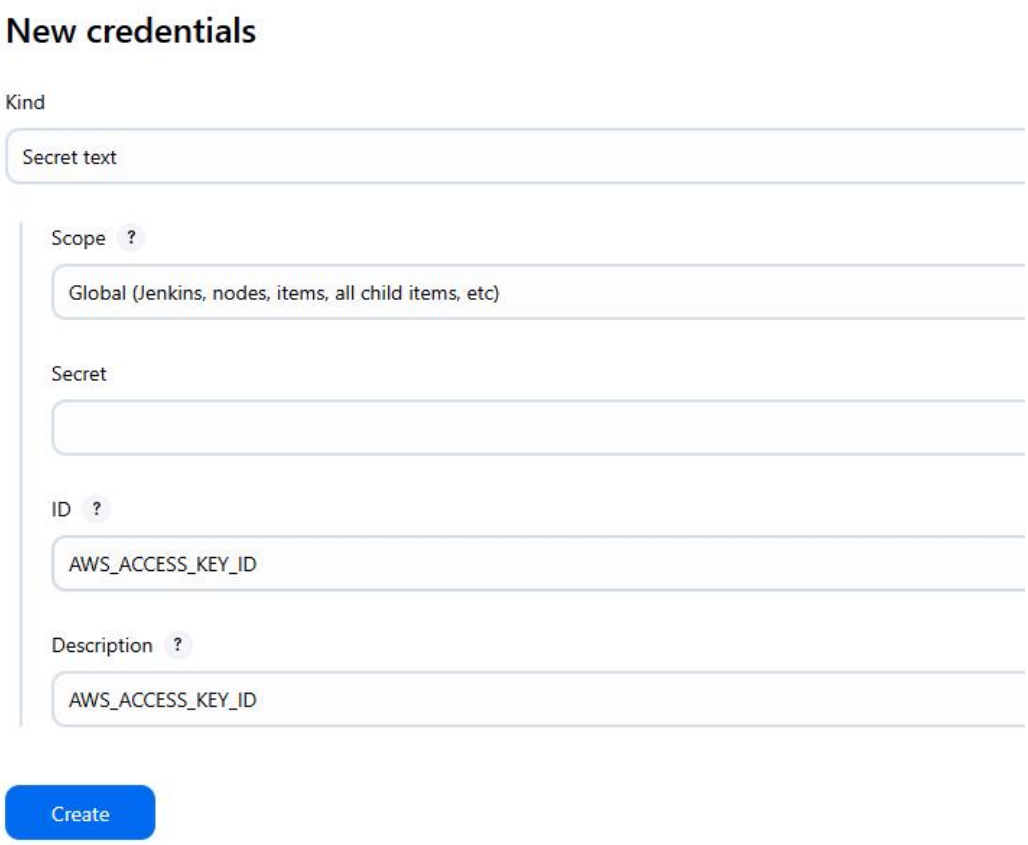




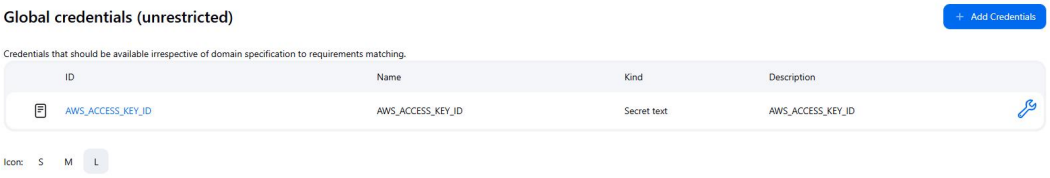
We are writing a Terraform script in order to start our EKS cluster. This Jenkins pipeline will clone the Terraform script. This pipeline is responsible to take the Terraform script (Plan, Execute) the Terraform script.



Click Credentials --> click on (global)



Secret is the password created in Jenkins at the start



Click Add Credentials for Docker

New credentials

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

saidocker567

Treat username as secret ?

Password ?

••••••••••

ID ?

Docker-credentials

Description ?

Docker-credentials

Create

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
AWS_ACCESS_KEY_ID	AWS_ACCESS_KEY_ID	Secret text	AWS_ACCESS_KEY_ID
Docker-credentials	saidocker567/****** (Docker-credentials)	Username with password	Docker-credentials

Icon: S M L

github.com/SaiGit-source/DevopsWithAWS_Course/tree/main/Jenkins/Pipeline/Terraform

Move or copy cells... | Email Finder hunter.io | Heroicons | loghmanb/daily-co... | CloudSkew | Excalidraw | HackerRank/Java/St... | Maven Repository:... | Online Training - S...

source / DevopsWithAWS_Course

Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

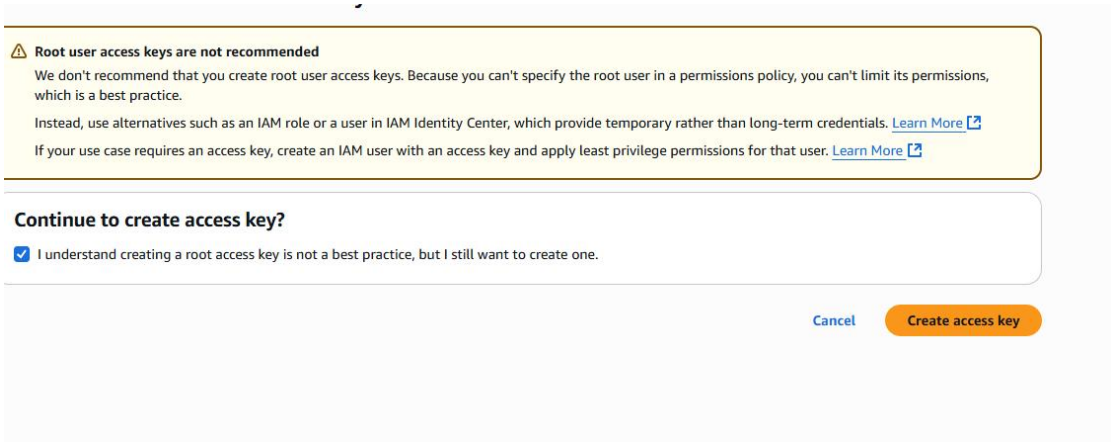
DevopsWithAWS_Course / Jenkins / Pipeline / Terraform /

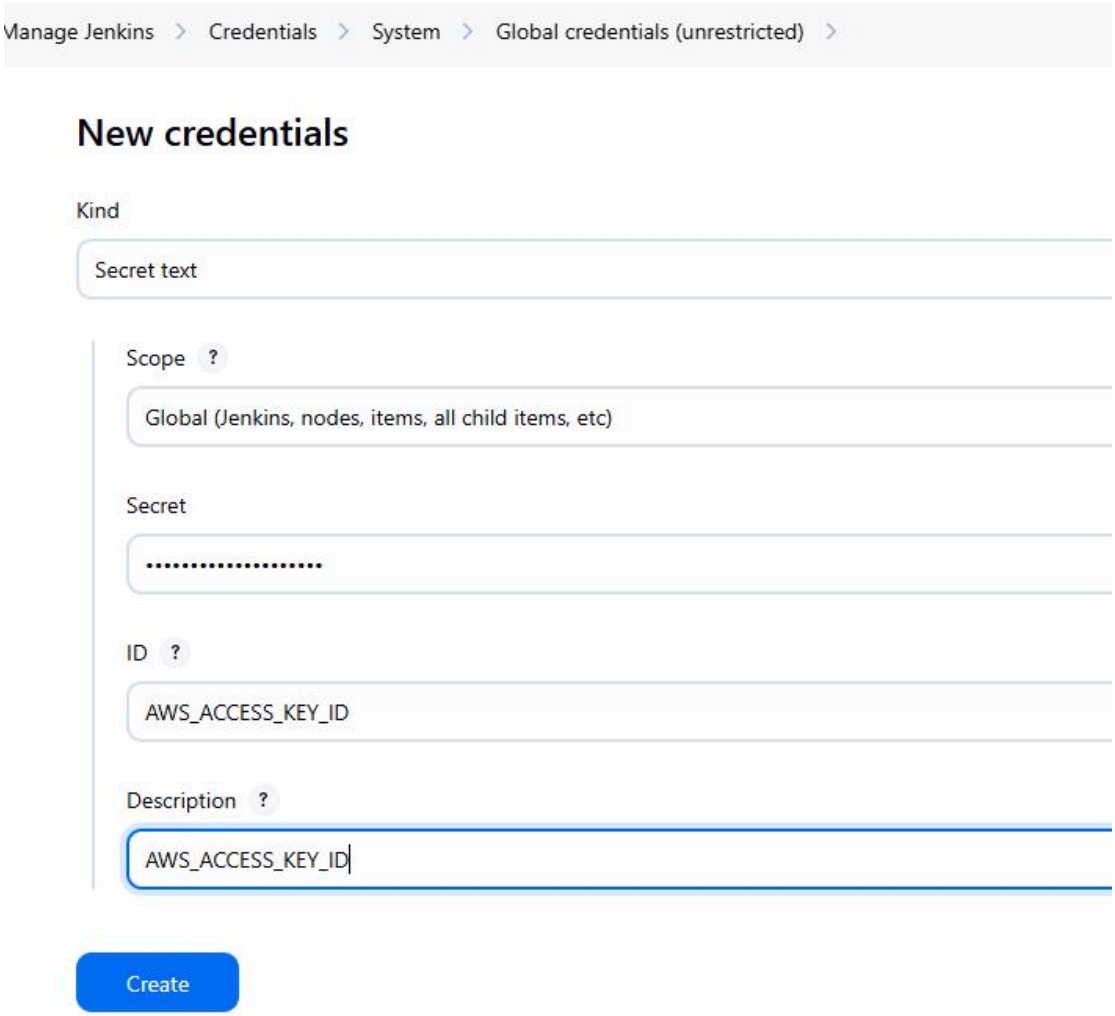
SaiGit-source DevOps: Terraform for Jenkins pipeline part1

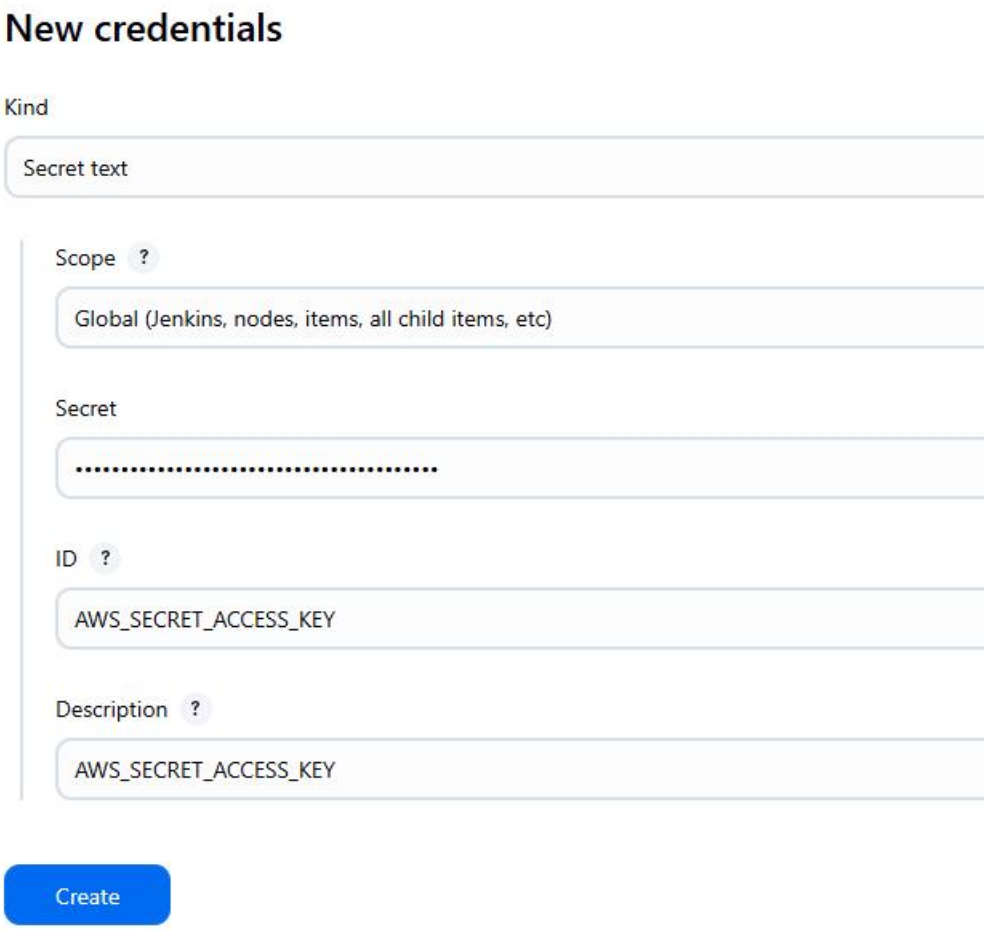
Name	Last commit message
..	
aws-cli.sh	DevOps: Jenkins pipeline automation with Terraform part0
eks.tf	DevOps: Terraform for Jenkins pipeline part1
k8s.sh	DevOps: Jenkins pipeline automation with Terraform part0
provider.tf	DevOps: Terraform for Jenkins pipeline part1
terraform.sh	DevOps: Jenkins pipeline automation with Terraform part0
vpc.tf	DevOps: Terraform for Jenkins pipeline part1

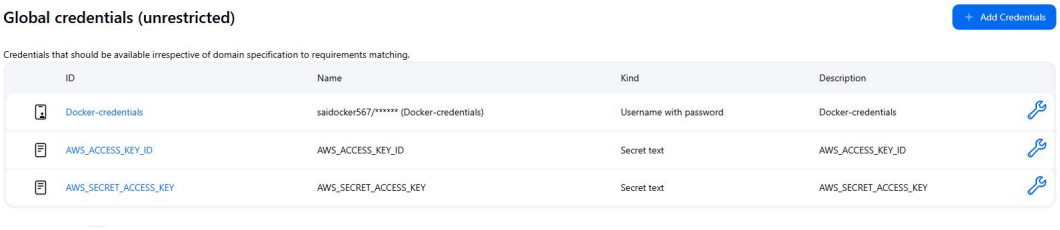
Terraform files are on Git
https://github.com/SaiGit-source/DevopsWithAWS_Course/tree/main/Jenkins/Pipeline/Terraform

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'
                }
            }
        }
    }
  }
}
```

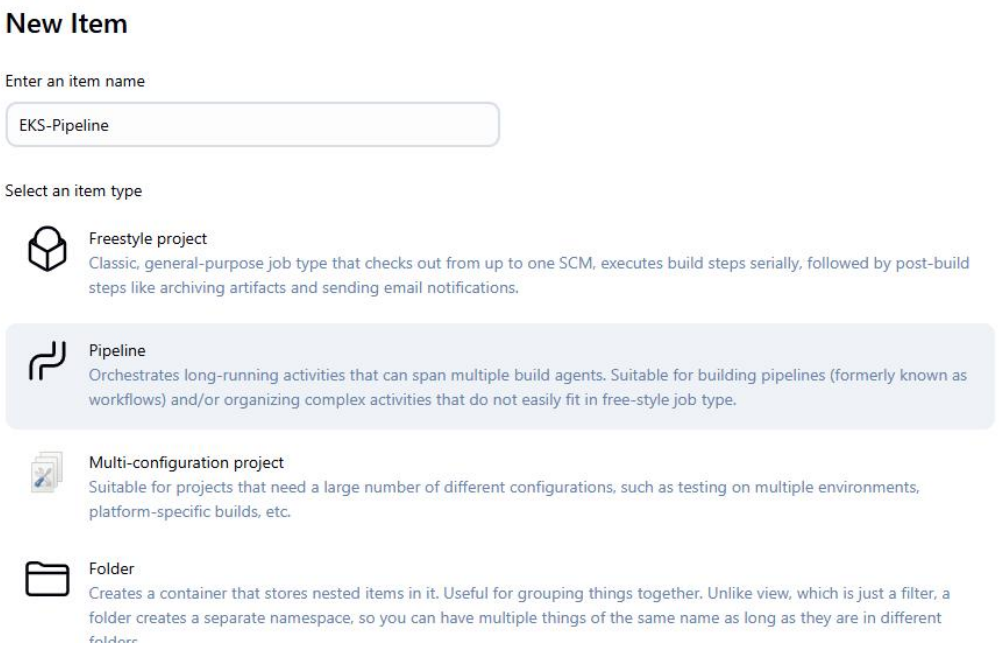
Install node in Jenkins server (to run react ap in pipeline)

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs
node -v
npm -v
```

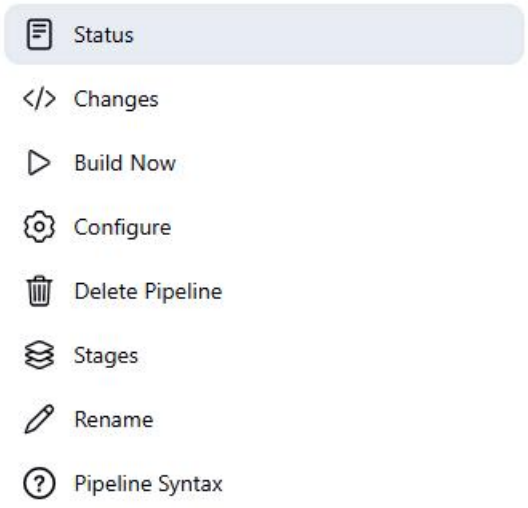
Install Docker in Jenkins server

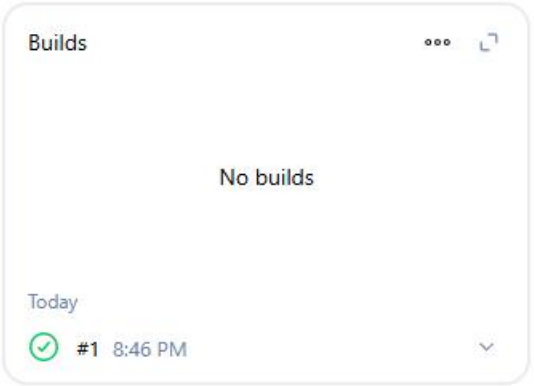
```
sudo apt-get update
sudo apt-get upgrade -y

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

```
ubuntu@ip-172-31-8-68:~$ ls -l
total 0
ubuntu@ip-172-31-8-68:~$ ls -l
total 0
ubuntu@ip-172-31-8-68:~$ sudo systemctl enable jenkins
sudo systemctl start jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
ubuntu@ip-172-31-8-68:~$ jenkins --version
2.504.3
ubuntu@ip-172-31-8-68:~$
```

```
[33m|[0m [0m[0m[0m  with module.eks.aws_iam_role.this[0],
[33m|[0m [0m  on .terraform/modules/eks/main.tf line 285, in resource "aws_iam_role" "this":
[33m|[0m [0m 285: resource "aws_iam_role" "this" [4m{[0m[0m
[33m|[0m [0m
[33m|[0m [0minline_policy is deprecated. Use the aws_iam_role_policy resource instead.
[33m|[0m [0mIf Terraform should exclusively manage all inline policy associations (the
[33m|[0m [0mcurrent behavior of this argument), use the aws_iam_role_policies_exclusive
[33m|[0m [0mresource as well.
[33m|[0m [0m
[33m|[0m [0m(and 12 more similar warnings elsewhere)
[33m'[0m[0m
[0m[1m[32m
Apply complete! Resources: 62 added, 0 changed, 0 destroyed.
[0m
[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

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

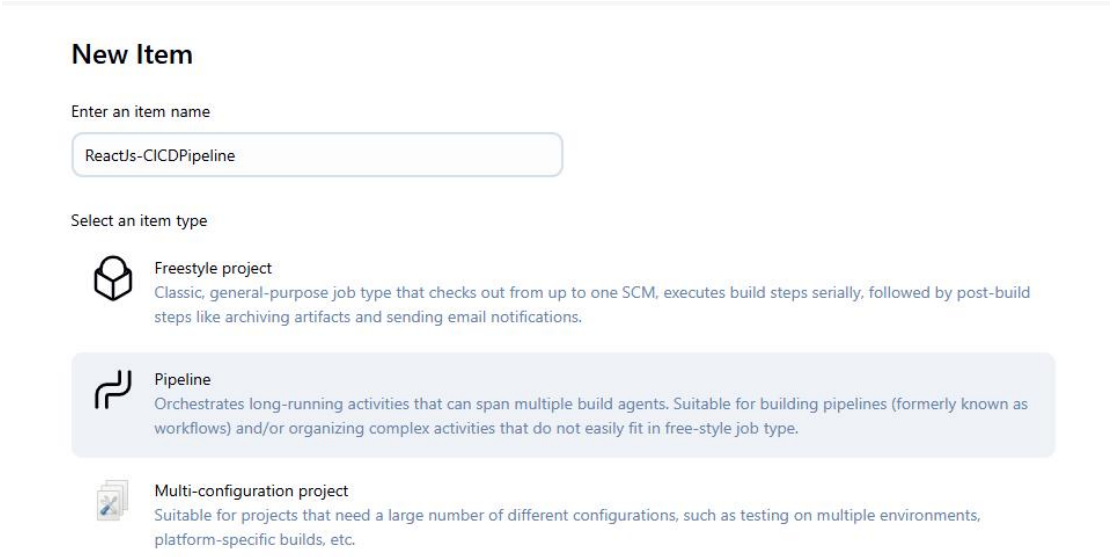
Install Docker in Jenkins server

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
"""
        }
      }
    }
```

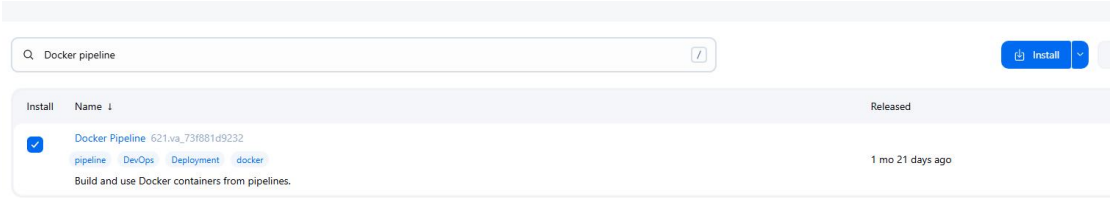
```
    stage('Build React App') {
      steps {
        sh 'npm install'
        sh 'npm run build'
      }
    }

    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')
     }
    }
   }
  }
```

```
  }
 }
```

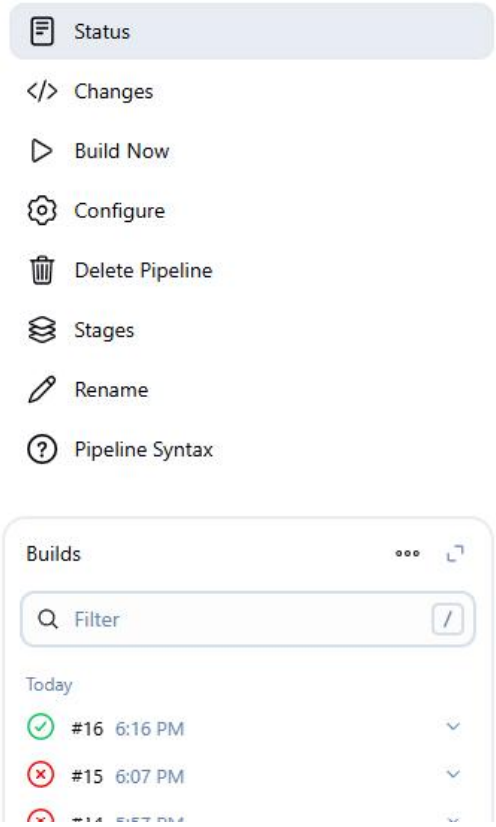
We need to install Docker Pipeline plugin, we need it for a particular function
Manage Jenkins --> Plugins

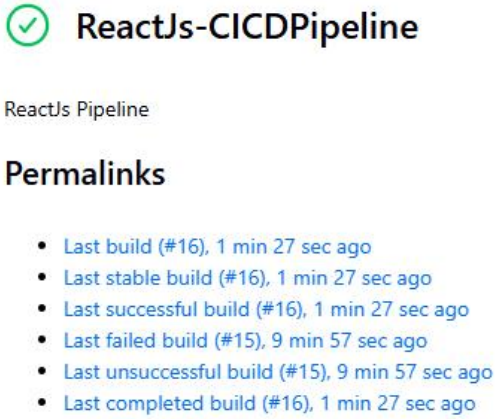


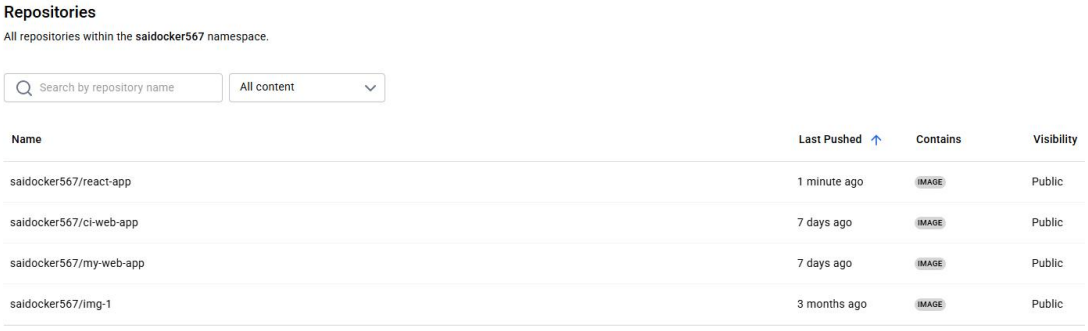

```
pipeline {
 agent any
 environment {
  REGISTRY = 'Docker-credentials'  // Docker Hub credentials ID in Jenkins
 }

 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')
      }
    }
  }
 }
}
}
```

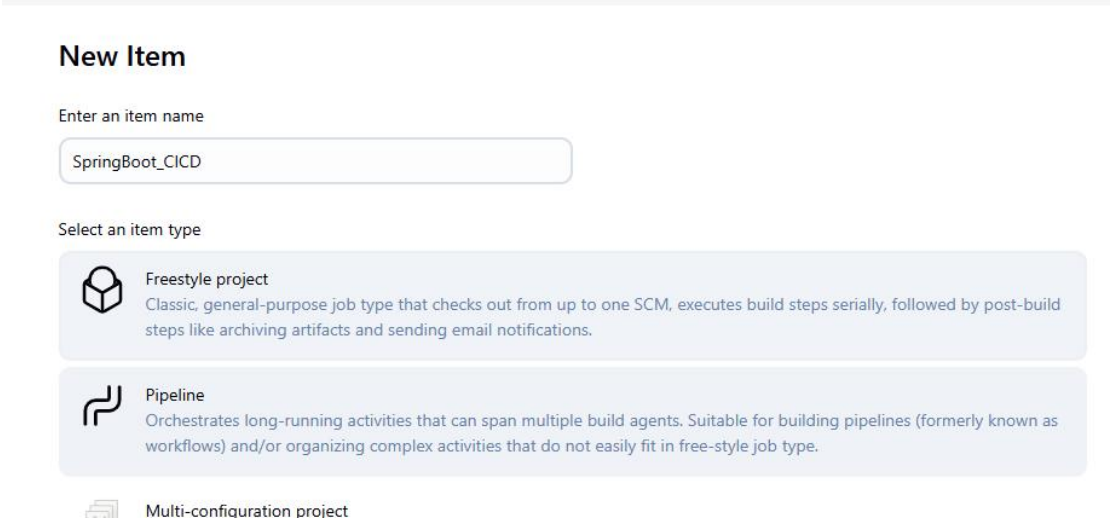
Restarted Jenkins server and it worked







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



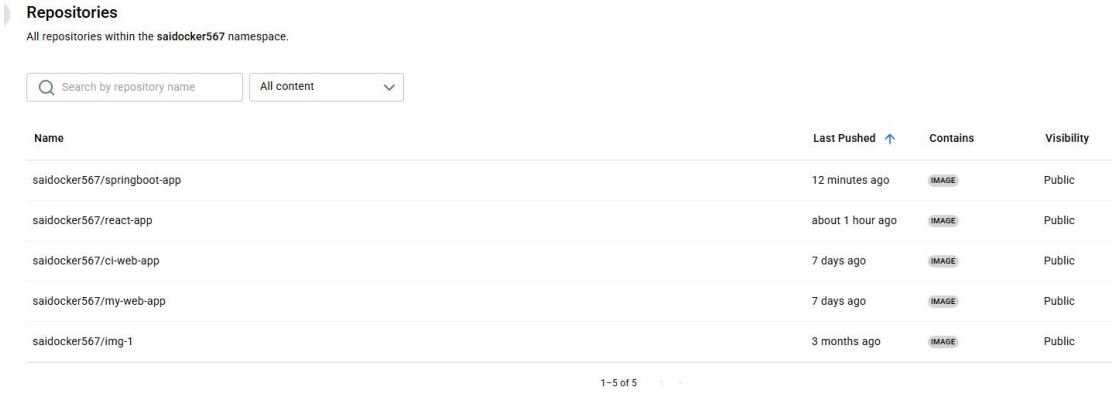
CICD Jenkins for backend SpringBoot app

==========

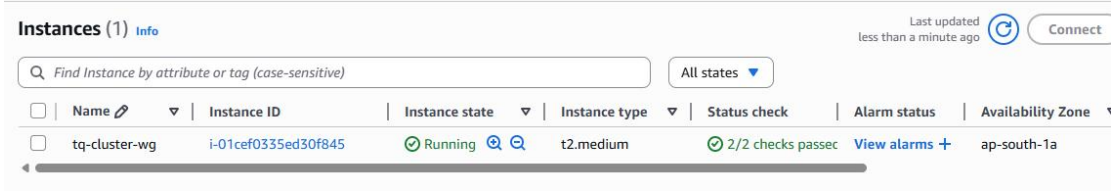
```
pipeline {
 agent any

 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')
          }
        }
      }
    }
```

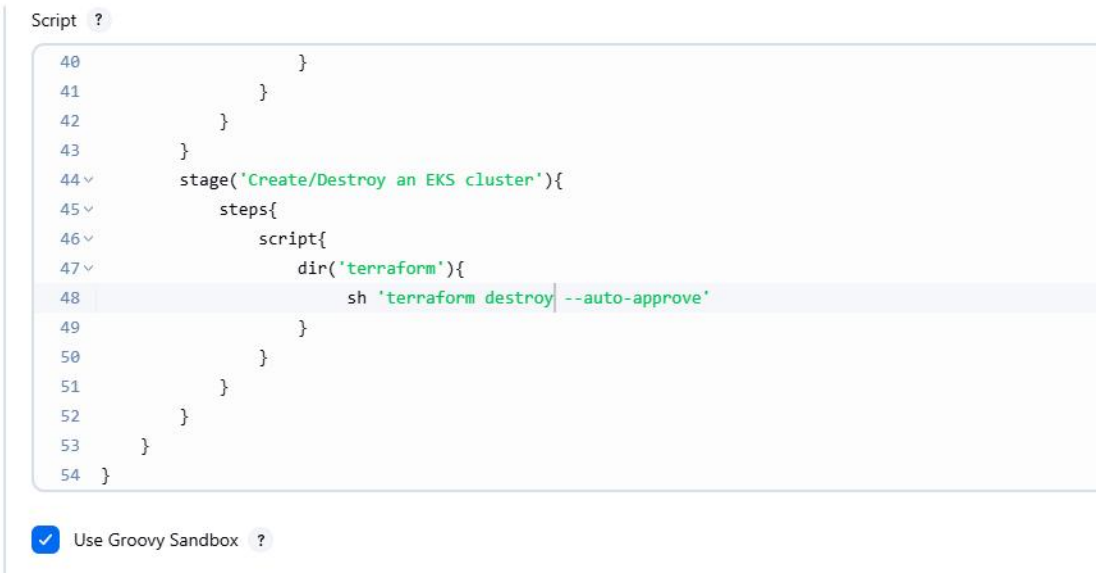
```
   stage('Deploy to Kubernetes') {
     steps {
       sh """
         export KUBECONFIG=${env.KUBECONFIG}
         kubectl apply -f spring-deployment.yaml
         kubectl set image deployment/springboot-deploy springboot-
container=saidocker567/springboot-app:${env.BUILD_NUMBER}
         kubectl rollout status deployment/springboot-deploy
       """
     }
   }
  }
}
```



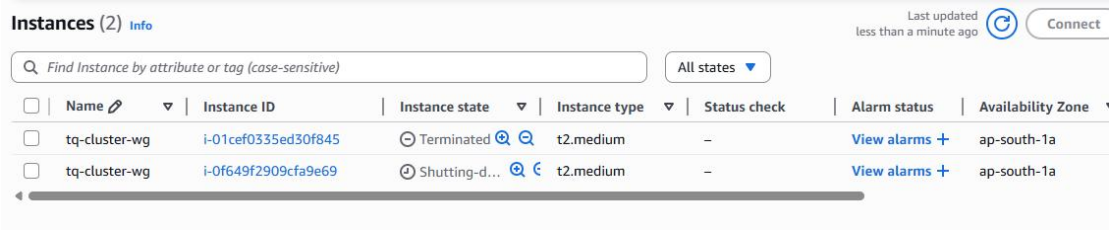
Problem is EKS-pipeline created all resources in ap-south-1 region, we want in ca-central-region



Using same pipeline to destroy 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

==========

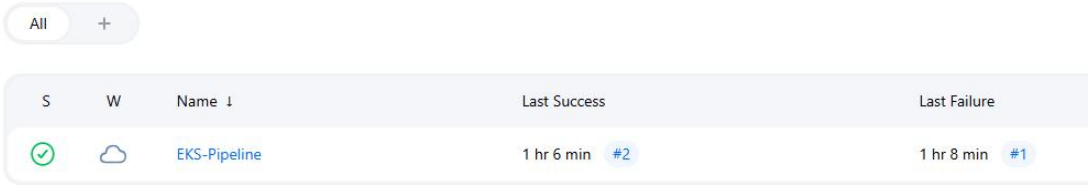
github repo https://github.com/teluskoOrg/DemoCiCd.git
demo for aws services for CICD

In the Jenkins-server

```
total 0
ubuntu@ip-172-31-8-68:~$ ls
ubuntu@ip-172-31-8-68:~$ ls -a
.  ..  .Xauthority  .bash_history  .bash_logout  .bashrc  .cache  .kube  .profile  .ssh  .sudo_as_admin_successful  .terraform.d
ubuntu@ip-172-31-8-68:~$ ls -la
total 44
drwxr-x--- 6 ubuntu ubuntu 4096 Jul 20 20:48 .
drwxr-xr-x 3 root   root   4096 Jul 19 16:54 ..
-rw------- 1 ubuntu ubuntu  120 Jul 20 18:10 .Xauthority
-rw------- 1 ubuntu ubuntu  283 Jul 20 20:47 .bash_history
-rw-r--r-- 1 ubuntu ubuntu  220 Mar 31  2024 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Mar 31  2024 .bashrc
drwx------ 2 ubuntu ubuntu 4096 Jul 19 17:01 .cache
drwxr-xr-x 2 root   root   4096 Jul 20 20:48 .kube
-rw-r--r-- 1 ubuntu ubuntu  807 Mar 31  2024 .profile
drwx------ 2 ubuntu ubuntu 4096 Jul 19 16:54 .ssh
-rw-r--r-- 1 ubuntu ubuntu    0 Jul 19 17:02 .sudo_as_admin_successful
drwxr-xr-x 2 ubuntu ubuntu 4096 Jul 20 13:03 .terraform.d
ubuntu@ip-172-31-8-68:~$
```

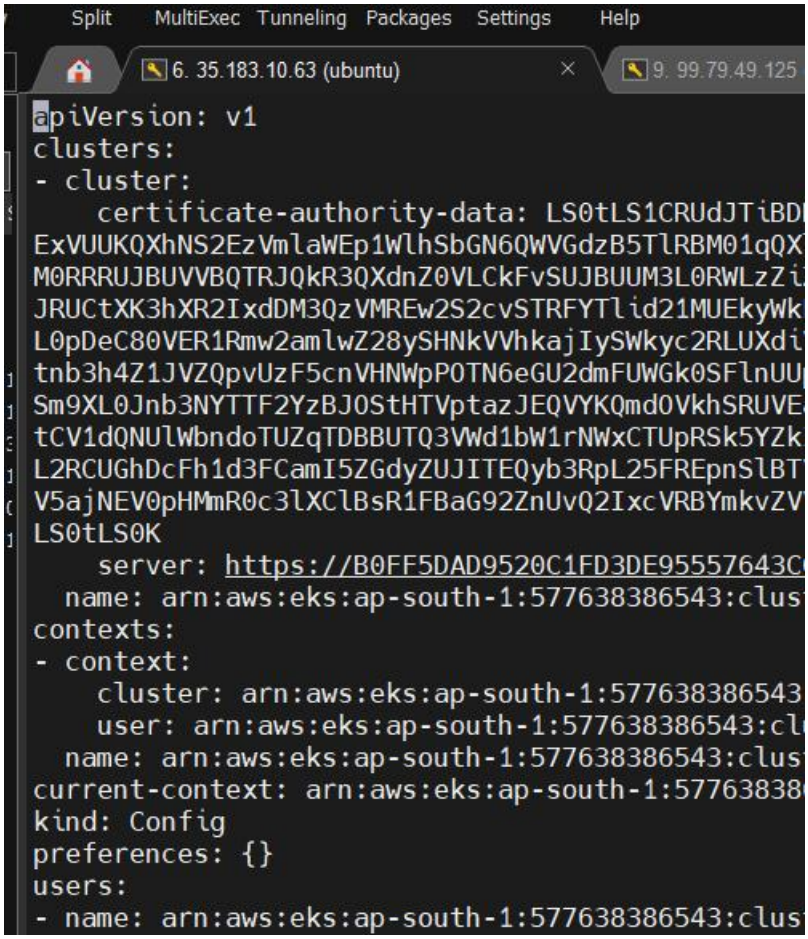
1:24:37

I ran the Terraform scripts (EKS-pipeline) from the TerraformFinal machine
Then the following commands to generate config file

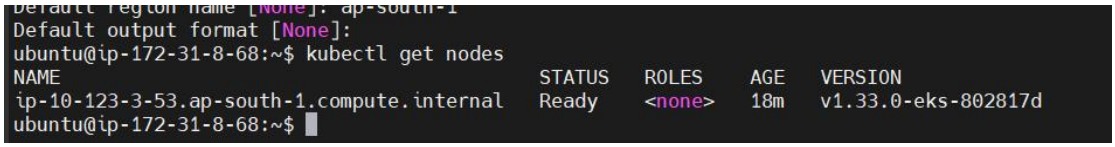


Generate config file
ubuntu@ip-172-31-11-18:~$ aws eks update-kubeconfig --name teluskoQ-cluster --region ap-south-1
Added new context arn:aws:eks:ap-south-1:577638386543:cluster/teluskoQ-cluster to /home/ubuntu/.kube/config
Now we copy this info into the other machine. Jenkins-Server

```
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
ubuntu@ip-172-31-11-18:~$ cat config
cat: config: No such file or directory
ubuntu@ip-172-31-11-18:~$ cat .kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJVkQ2Y203bkRGell3RFFZSktvWklodmNOQ
ExVUUKQXhNS2EzVmlaWEp1WlhSbGN6QWVGdzB5TlRBM01qQXlNVFV5TlRCYUZ3MHpOVEEzTVRneU1UVTNOVEJhTUJVeApFekFSQmdOVkJBTVRDbXQxWW1WeWJtVjBaWE1
M0RRRUJBUVVBQTRJQkR3QXdnZ0VLCkFvSUJBUUM3L0RWLzZiZm5kbk9WZ2owZG4rbk8xQmxKcm0xdXBUZVB6RUdXR0phbm9kWHpwaWx1UHZ3alNyVUkKdUNERS9NY2x5R
JRUCtXK3hXR2IxdDM3QzVMREw2S2cvSTRFYTlid21MUEkyWkFuZgpYSnZjTEorM25PWS9FLzhsQnVtSkxIS2tCbzhRTGlHUEc0Q09XL0JmK2VGUGFUUUhrcjdSMGMyeGR
L0pDeC80VER1Rmw2amlwZ28ySHNkVVhkajIySWkyc2RLUXdiVWJockFRNHkrR1JOLzhNNkRDWkUKRUJFUkRQOXB1WkRRZlB0Y01VRzdEanZoSFhLWWVRRkY0akF4cWd6U
tnb3h4Z1JVZQpvUzF5cnVHNWpPOTN6eGU2dmFUWGk0SFlnUUpCQWdNQkFBR2pXVEJYTUE0R0ExVWREd0VCL3dRRUF3SUNwREFQCkJnTlZIUk1CQWY4RUJUQURBUUgvTUI
Sm9XL0Jnb3NYTTF2YzBJOStHTVptazJEQVYKQmd0VkhSRUVEakFNZ2dwcmRXSmxjbTVsZEdWek1BMEdDU3FHU0liM0RRRUJBd1VBQTRJQkFRQUhEVW43cHM0aQpYeFRaZ
tCV1dQNUlWbndoTUZqTDBBUTQ3VWd1bW1rNWxCTUpRSk5YZkIrUHd2cHl6Cm1uamhqV3VRQVdFc3dXNlhIQXEvY2hDclp0WEdtQUd4SGJuWmVtOTFLNWdwemxFYTl3djN
L2RCUGhDcFh1d3FCamI5ZGdyZUJITEQyb3RpL25FREpnSlBTVEZpYWc0b0dwalprbzZSN2szbEVwTCtCNQpCd2Mwa1RnazBISTQzUTF0K1VRdzFQZTJzQWZtYzZPd3VRZ
V5ajNEV0pHMmR0c3lXClBsR1FBaG92ZnUvQ2IxcVRBYmkvZVVsTlN1ZTNudXVDV3pubGtWTXdyWmNaeGRqc05HRTJtUlhCSGFoVFZrSG0KdEdlemdGSUpQRGNtCi0tLS0
LS0tLS0K
    server: https://B0FF5DAD9520C1FD3DE95557643CC05C.gr7.ap-south-1.eks.amazonaws.com
  name: arn:aws:eks:ap-south-1:577638386543:cluster/teluskoQ-cluster
contexts:
- context:
    cluster: arn:aws:eks:ap-south-1:577638386543:cluster/teluskoQ-cluster
    user: arn:aws:eks:ap-south-1:577638386543:cluster/teluskoQ-cluster
  name: arn:aws:eks:ap-south-1:577638386543:cluster/teluskoQ-cluster
current-context: arn:aws:eks:ap-south-1:577638386543:cluster/teluskoQ-cluster
```



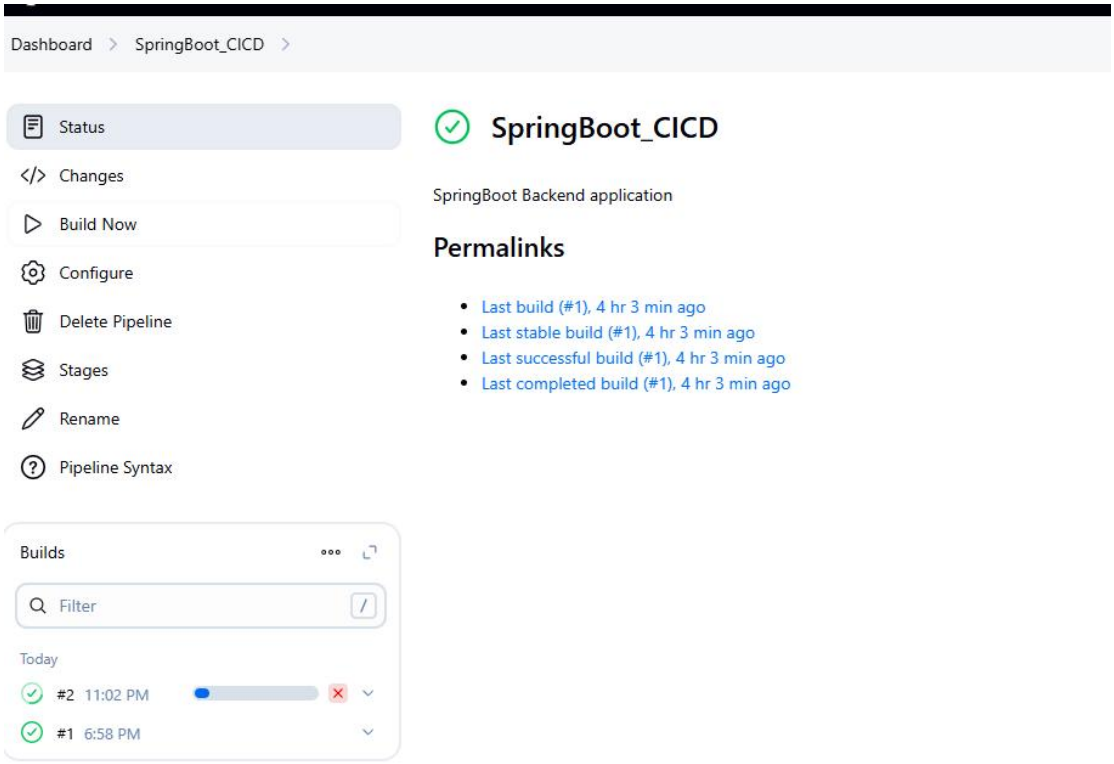
Now we get nodes on the Jenkins-Server machine as well, even though the cluster is created on a different region. If required, run aws configure



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

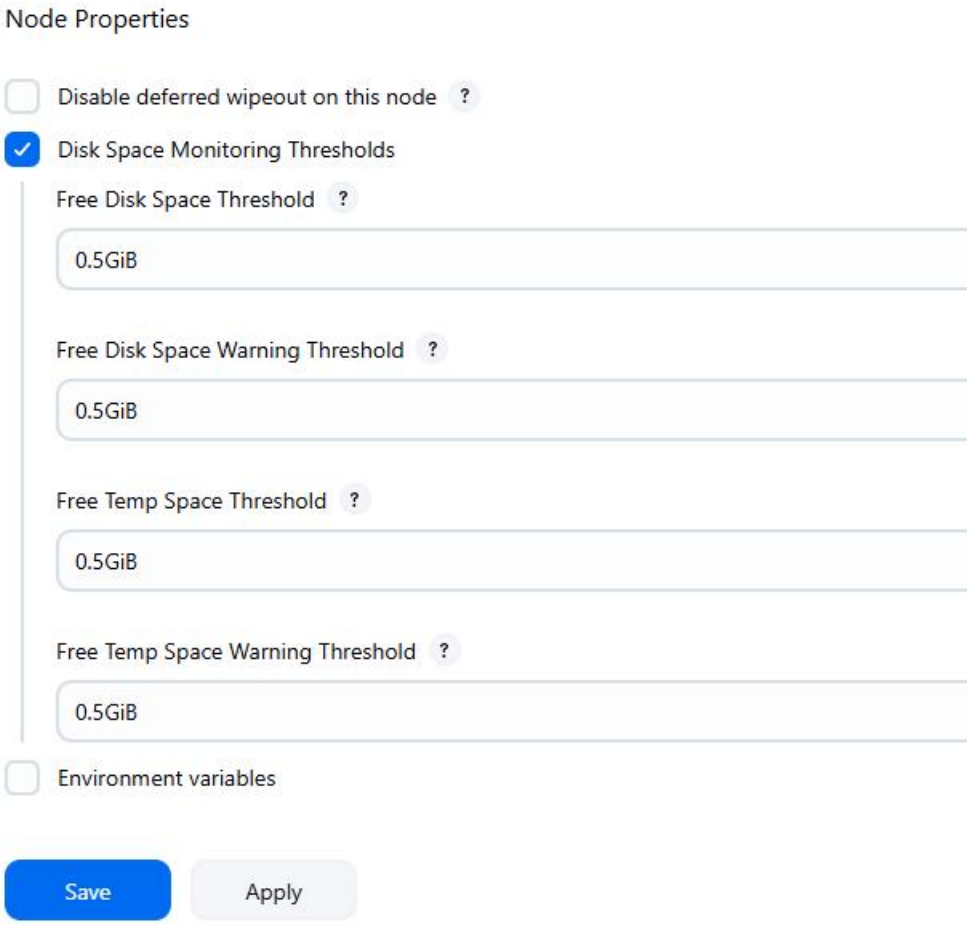
Anyways, now we are going to try SpringBootCICD

```
[Pipeline] stage
[Pipeline] { (Deploy to Kubernetes)
[Pipeline] sh
+ export KUBECONFIG=/var/lib/jenkins/.kube/config
+ kubectl apply -f spring-deployment.yaml --validate=false
E0721 00:27:12.510270   39733 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: the server has asked for the client to provide credentials"
E0721 00:27:13.926549   39733 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: the server has asked for the client to provide credentials"
unable to recognize "spring-deployment.yaml": the server has asked for the client to provide credentials
unable to recognize "spring-deployment.yaml": the server has asked for the client to provide credentials
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: script returned exit code 1
Finished: FAILURE
```

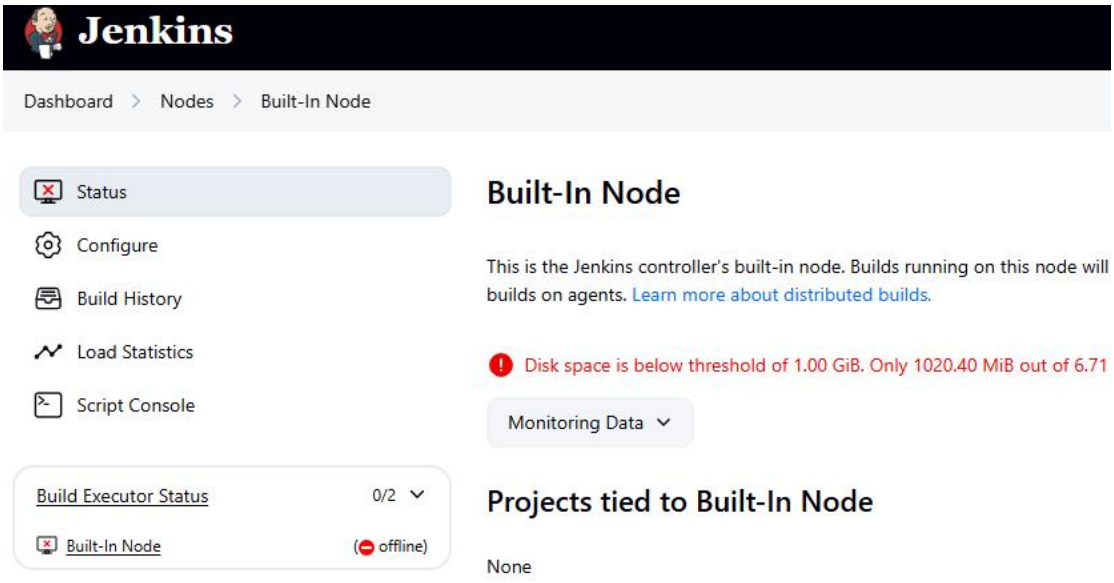
Facing some issue with kubectl apply, will fix tomorrow
Destroying EKS cluster and all resources

I start two VMs
Then ran sudo systemctl restart jenkins
Ran EKS-Pipeline in EKS-host VM (TerraformFinal)
Re-copied Config contents into Jenkins-VM

If the Pipeline gets stuck go to Build-In Node and adjust the thresholds

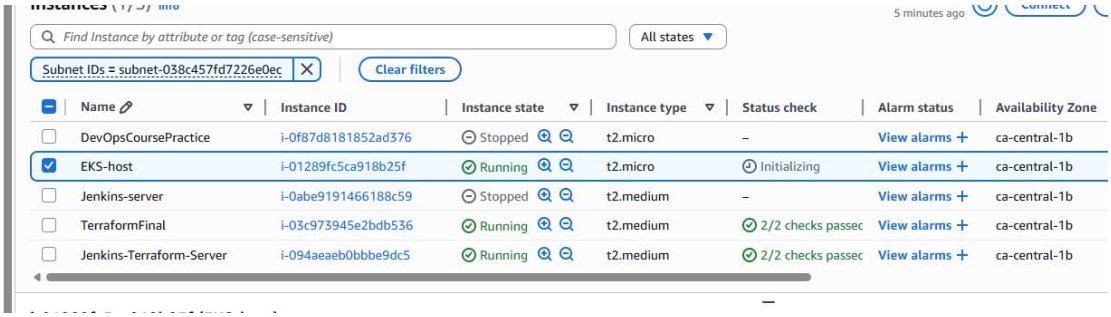


Now



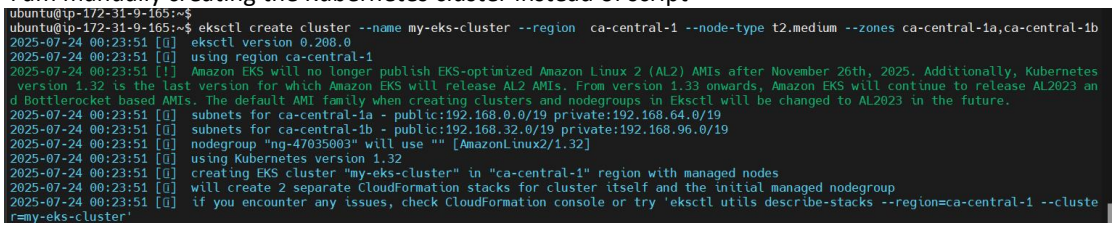
Now it will build

```
[0m[1mmodule.eks.aws_eks_cluster.this[0]: Creating...[0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[0]: Still creating... [00m10s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[1]: Still creating... [00m10s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[0]: Still creating... [00m20s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[1]: Still creating... [00m20s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[1]: Still creating... [00m30s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[0]: Still creating... [00m30s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[0]: Still creating... [00m40s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[1]: Still creating... [00m40s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[0]: Still creating... [00m50s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[1]: Still creating... [00m50s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[1]: Still creating... [01m00s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[0]: Still creating... [01m00s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[0]: Still creating... [01m10s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[1]: Still creating... [01m10s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[1]: Still creating... [01m20s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[0]: Still creating... [01m20s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[0]: Still creating... [01m30s elapsed][0m[0m
[0m[1mmodule.vpc.aws_nat_gateway.this[1]: Still creating... [01m30s elapsed][0m[0m
```



eksctl create cluster --name my-eks-cluster --region ca-central-1 --node-type t2.medium --zones ca-central-1a,ca-central-1b

I am manually creating the Kubernetes cluster instead of script



Copy Config and paste into new Jenkins-Server
Update in both locations

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

```
ubuntu@ip-172-31-9-165:~$
ubuntu@ip-172-31-9-165:~$ cat .kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJVXdjS
ExVUUKQXhNS2EzVmlaWEp1WlhSbGN6QWVGdzB5TlRBM01qUXdNREl6TXpWYUZ3MHpOVEEzTWpJd01ESTRNelZhTUJVeApFekFSQmd
M0RRRUJBUVVBQTRJQkR3QXdnZ0VLCkFvSUJBUUM4V1hEOEVWTU1lV1p4Ym9KR1pYb2I2VzZudGwvTHlJV09BM2loN1ZabWdhL0FEZ
MwNmg0VjNGcWgxNldnbVJiOXZNRW9KTXhUc2N2YXZRS09MaTdvVwpveHRJK29pSHFRSEc4b1AzVFc5RnIrNDZBbHhNTTg3VXlmZmN
aENXZFF1ejF4ZkVFaUw0eSt2R1hZM2ZjeUJPZTg1d0RyM25odGw4NS8zZFpjcXJ4WDVuejRlN1YKRTVjWFRzZ21VTmorNnNPUHc0U
p5ME90VU8vRgp6VE05aUFPclBHYW9iNnRRZkJVWFFybmNFZXF2QWdNQkFBR2pXVEJYTUE0R0ExVWREd0VCL3dRRUF3SUNwREFQCkJ
ekpqdVZuRUlnTkM2Q05xS0s4K3h1SjVUQVYKQmdOVkhSRUVEakFNZ2dwcmRXSmxjbTVsZEdWek1BMEdDU3FHU0liM0RRRUJDd1VBQ
9rM0N1MmpXYkQyS3ZGTjhjUWN6WmFqYXZtTHRlbVpJSXF6YWUzcVZ2NFptCm1xM0FUU0QvY21xR3R1OVlkYzZnQUZYd3pDRFlHSEF
NVd6cEhkZnV5cEhweTBhR09jQ3ZraGFpdll3QnVQSENIZFk3N3NNR0htbmlpdWFXeFg0OS9HbWxYQzY1MwpjWk9oOVJCQXExZUwrT
ExL2pYVDR1bXhDaEpxCnpmUUlCQVBlc0lBK3ZVVHgra0tZcmkvUW9oQ0VrUkwzS1VEcmtEYXQ3ckVvMDVhMm1hcVlhc3dMSzFoSDZ
LS0tLS0K
    server: https://D823BD57A25BECB641485BDEE0A025A4.gr7.ca-central-1.eks.amazonaws.com
  name: my-eks-cluster.ca-central-1.eksctl.io
contexts:
- context:
    cluster: my-eks-cluster.ca-central-1.eksctl.io
    user: i-01289fc5ca918b25f@my-eks-cluster.ca-central-1.eksctl.io
  name: i-01289fc5ca918b25f@my-eks-cluster.ca-central-1.eksctl.io
current-context: i-01289fc5ca918b25f@my-eks-cluster.ca-central-1.eksctl.io
kind: Config
```

I am using one of my old Jenkins server to run the pipeline

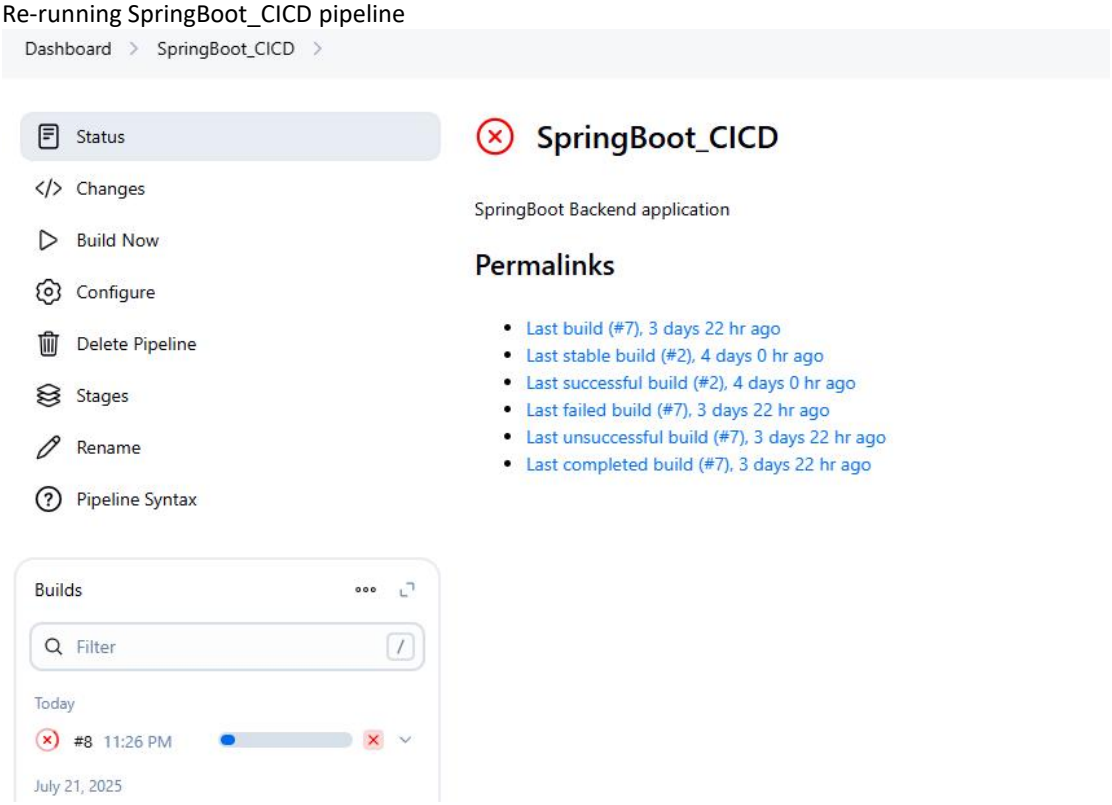
```
ubuntu@ip-172-31-11-116:~$
ubuntu@ip-172-31-11-116:~$ sudo vi .kube/config
ubuntu@ip-172-31-11-116:~$ kubectl get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
ip-192-168-20-71.ca-central-1.compute.internal   Ready    <none>   32m   v1.32.3-eks-473151a
ip-192-168-41-3.ca-central-1.compute.internal    Ready    <none>   32m   v1.32.3-eks-473151a
ubuntu@ip-172-31-11-116:~$
```

eksctl delete cluster --name my-eks-cluster --region ca-central-1
Trying to run pipeline to create EKS cluster from Jenkins-new machine itself
Generate config file
aws eks update-kubeconfig --name teluskoQ-cluster --region ap-south-1

```
Last login: Thu Jul 24 23:17:00 2025 from 136.56.236.206
ubuntu@ip-172-31-8-68:~$ 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
ubuntu@ip-172-31-8-68:~$ kubectl get nodes
NAME                                          STATUS   ROLES    AGE     VERSION
ip-10-123-3-147.ap-south-1.compute.internal   Ready    <none>   4m20s   v1.33.0-eks-802817d
ubuntu@ip-172-31-8-68:~$
```

```
[33m|[0m [0mresource as well.
[33m|[0m [0m
[33m|[0m [0m(and 12 more similar warnings elsewhere)
[33m'[0m[0m
[0m[1m[32m
Apply complete! Resources: 62 added, 0 changed, 0 destroyed.
[0m
[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Re-running SpringBoot_CICD pipeline



Failed

```
[Pipeline] sh
+ export KUBECONFIG=/.kube/config
+ kubectl get nodes
E0724 23:27:12.787326    5963 memcache.go:265] "Unhandled Error" err=<
        couldn't get current server API group list: <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fapi%3Ftimeout%3D32s'/><script id='redirect' data-redirect-url='/login?
from=%2Fapi%3Ftimeout%3D32s' src='/static/bf28ae9f/scripts/redirect.js'></script></head><body style='background-color:white; color:white;'>
        Authentication required
        <!--
        -->

        </body></html>
 >
E0724 23:27:12.789998    5963 memcache.go:265] "Unhandled Error" err=<
        couldn't get current server API group list: <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fapi%3Ftimeout%3D32s'/><script id='redirect' data-redirect-url='/login?
from=%2Fapi%3Ftimeout%3D32s' src='/static/bf28ae9f/scripts/redirect.js'></script></head><body style='background-color:white; color:white;'>
        Authentication required
        <!--
        -->
```

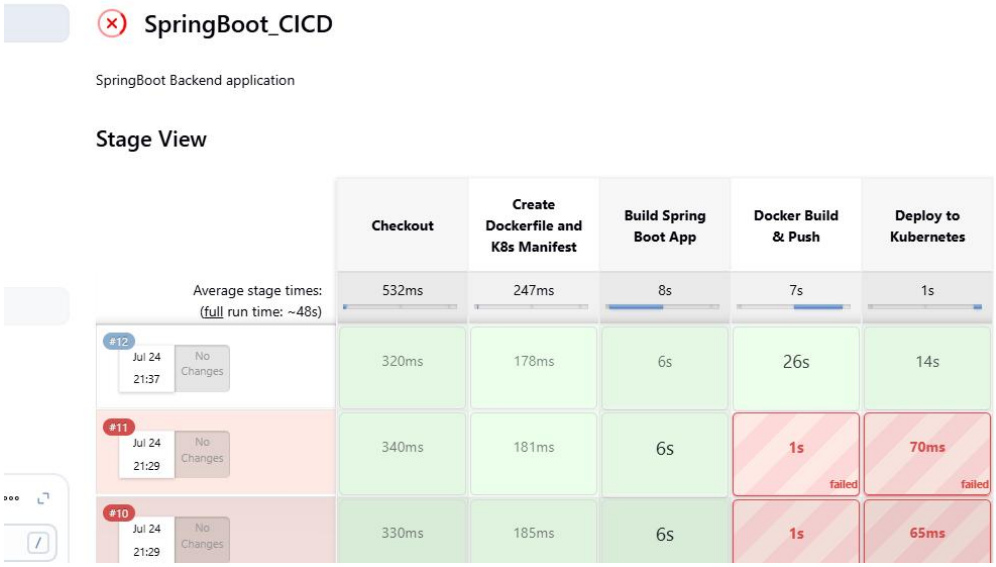
Manually created cluster on EKS-host

```
2025-07-24 23:38:36 [ℹ]  deploying stack "eksctl-my-eks-cluster-nodegroup-ng-f08deda1"
2025-07-24 23:38:36 [ℹ]  waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-ng-f08deda1"
2025-07-24 23:39:06 [ℹ]  waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-ng-f08deda1"
2025-07-24 23:39:50 [ℹ]  waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-ng-f08deda1"
2025-07-24 23:41:29 [ℹ]  waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-ng-f08deda1"
2025-07-24 23:41:29 [ℹ]  waiting for the control plane to become ready
2025-07-24 23:41:30 [✔]  saved kubeconfig as "/home/ubuntu/.kube/config"
2025-07-24 23:41:30 [ℹ]  no tasks
2025-07-24 23:41:30 [✔]  all EKS cluster resources for "my-eks-cluster" have been created
2025-07-24 23:41:30 [ℹ]  nodegroup "ng-f08deda1" has 2 node(s)
2025-07-24 23:41:30 [ℹ]  node "ip-192-168-28-217.ca-central-1.compute.internal" is ready
2025-07-24 23:41:30 [ℹ]  node "ip-192-168-52-17.ca-central-1.compute.internal" is ready
2025-07-24 23:41:30 [ℹ]  waiting for at least 2 node(s) to become ready in "ng-f08deda1"
2025-07-24 23:41:30 [ℹ]  nodegroup "ng-f08deda1" has 2 node(s)
2025-07-24 23:41:30 [ℹ]  node "ip-192-168-28-217.ca-central-1.compute.internal" is ready
2025-07-24 23:41:30 [ℹ]  node "ip-192-168-52-17.ca-central-1.compute.internal" is ready
2025-07-24 23:41:30 [✔]  created 1 managed nodegroup(s) in cluster "my-eks-cluster"
2025-07-24 23:41:32 [ℹ]  kubectl command should work with "/home/ubuntu/.kube/config", try 'kubectl get nodes'
2025-07-24 23:41:32 [✔]  EKS cluster "my-eks-cluster" in "ca-central-1" region is ready
ubuntu@ip-172-31-9-165:~$
```

```
ubuntu@ip-172-31-8-68:~$ kubectl get nodes
Error from server (Forbidden): nodes is forbidden: User "arn:aws:iam::577638386543:root" cannot list resource "nodes" in API group "" at the
cope
ubuntu@ip-172-31-8-68:~$
```

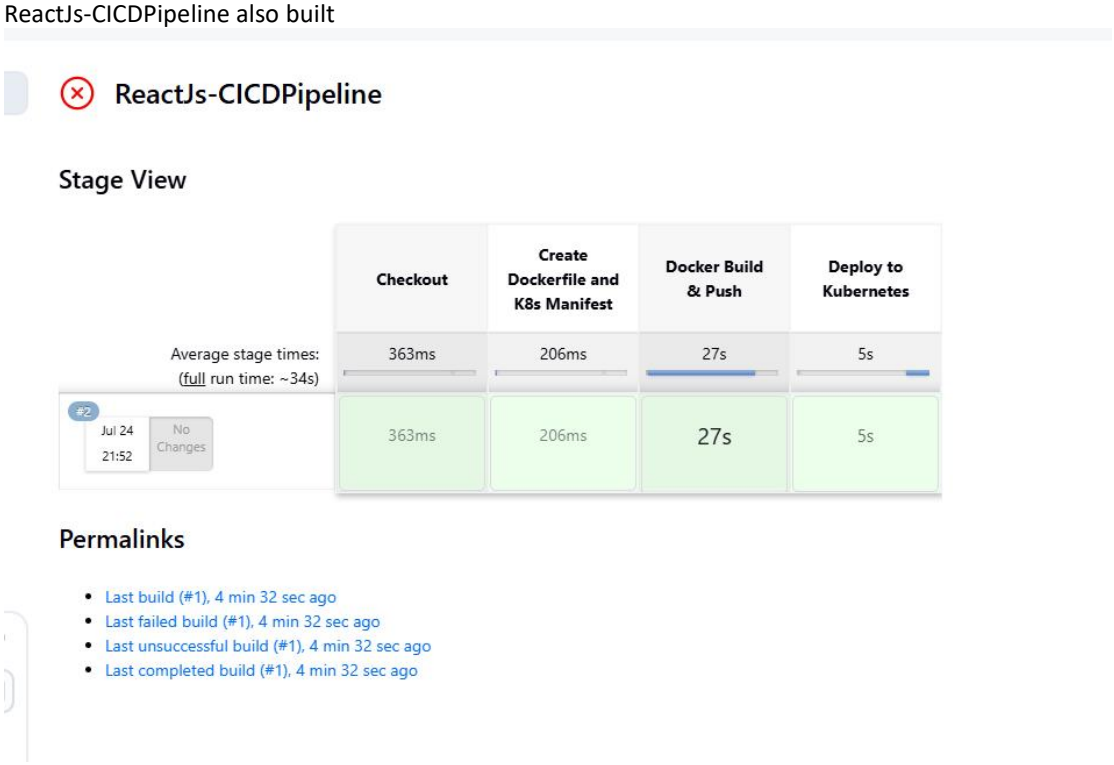
Trying Jenkins-server1
Next I install Docker Pipeline plugin

Finally built SpringBoot_CICD after multiple attempts

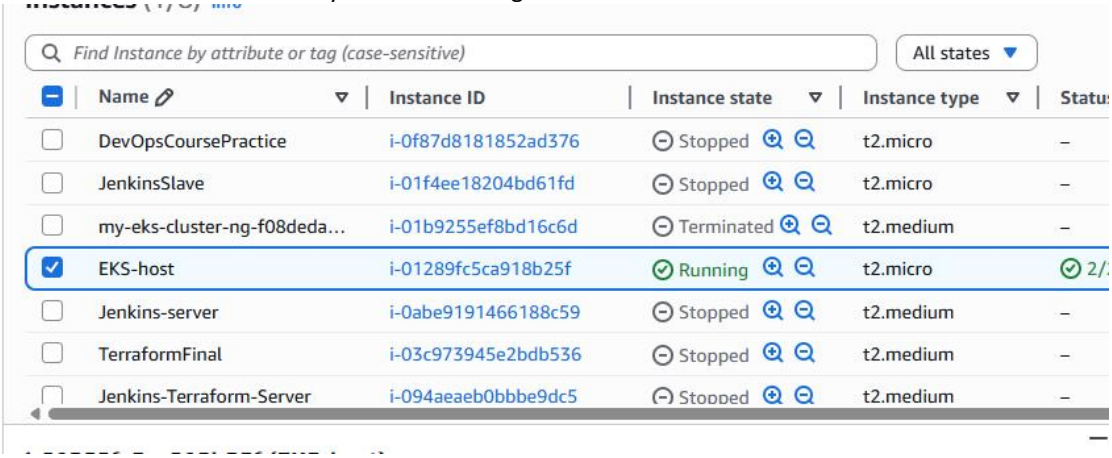



```
environment {
  REGISTRY = 'Sai_Docker_ID1'  // Docker Hub credentials ID in Jenkins
  KUBECONFIG = '/var/lib/jenkins/.kube/config'  // Path to kubeconfig on Jenkins server
}
```

ReactJs-CICDPipeline also built



EKS-Host
eksctl delete cluster --name my-eks-cluster --region ca-central-1



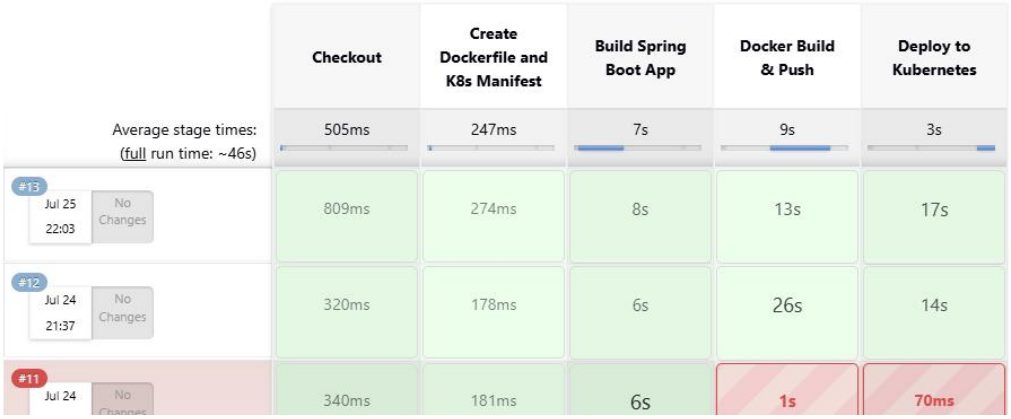
We are using EKS-host and Jenkins-server for now

I re-created the clusters in EKS-host then started Jenkins-server
I re-ran SpringBoot_CICD

SpringBoot_CICD

SpringBoot Backend application

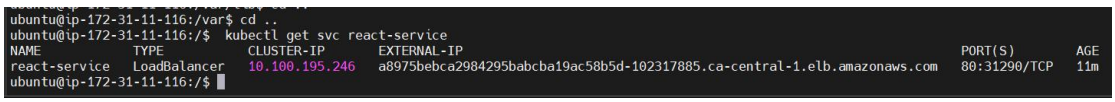
Stage View



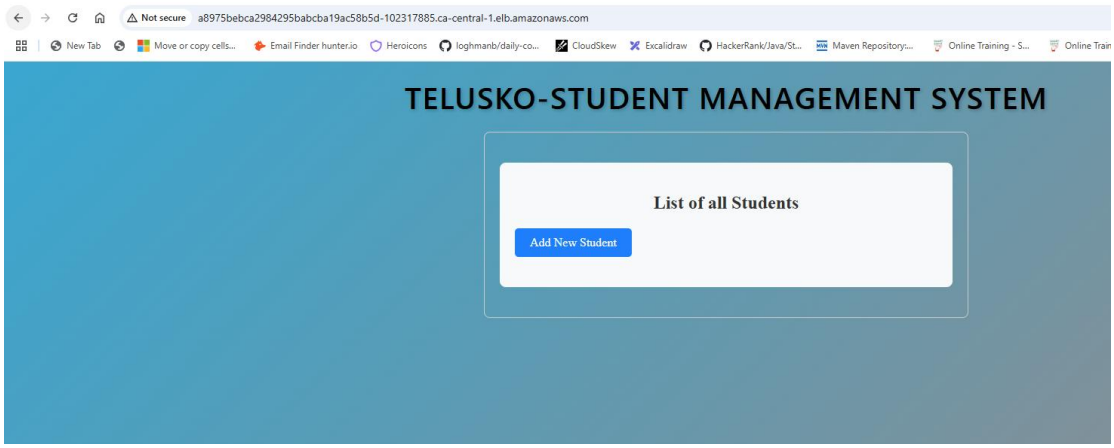
ReactJs-CICDPipeline

Stage View

Once CICD React is successful, kubectl get svc react-service



ubuntu@ip-172-31-11-116:/$ kubectl get svc react-service
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
react-service LoadBalancer 10.100.195.246 a8975bebca2984295babcba19ac58b5d-102317885.ca-central-1.elb.amazonaws.com 80:31290/TCP 11m



1:39

EKS-Host
eksctl delete cluster --name my-eks-cluster --region ca-central-1