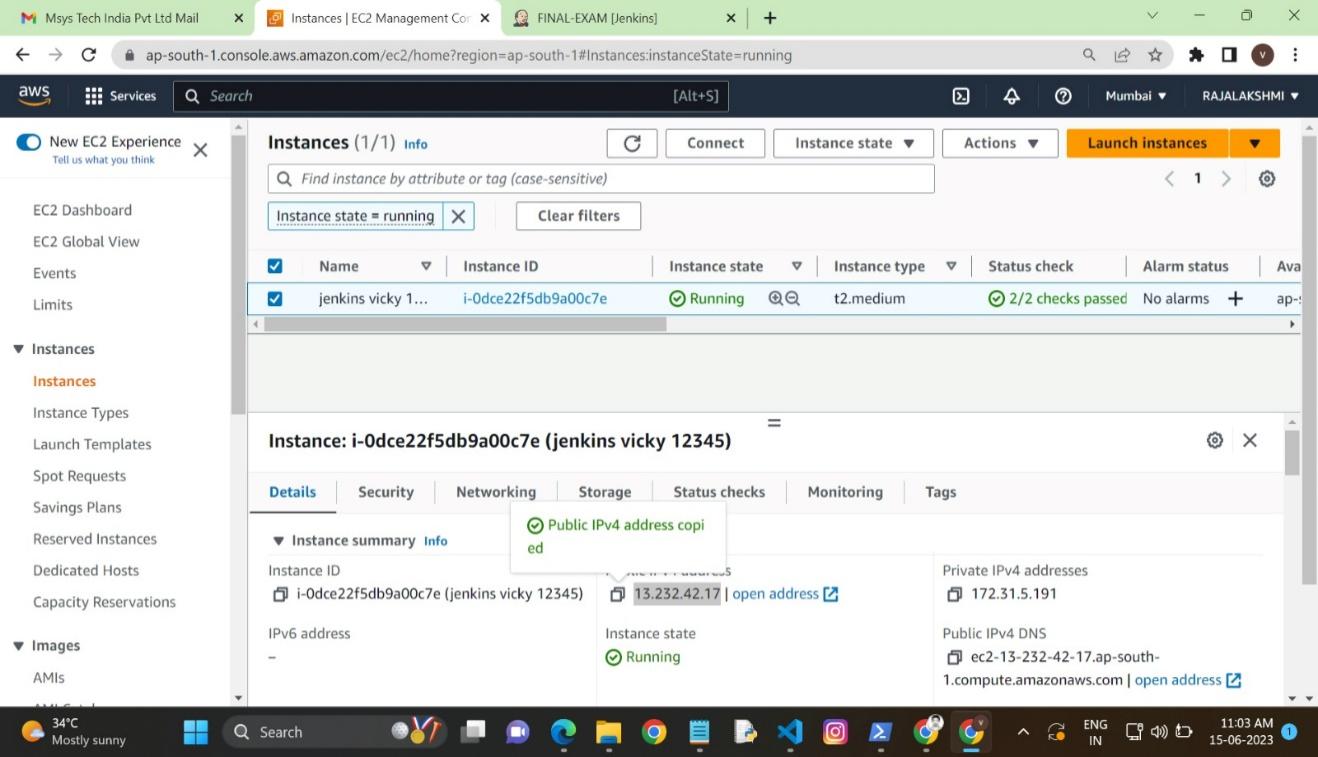
**Final Assessment**

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

1. **Create an EC2 instance in the AWS console:**



EC2 instance created

1. **Steps for installing Jenkins in the EC2 instance:**

sudo apt update -y

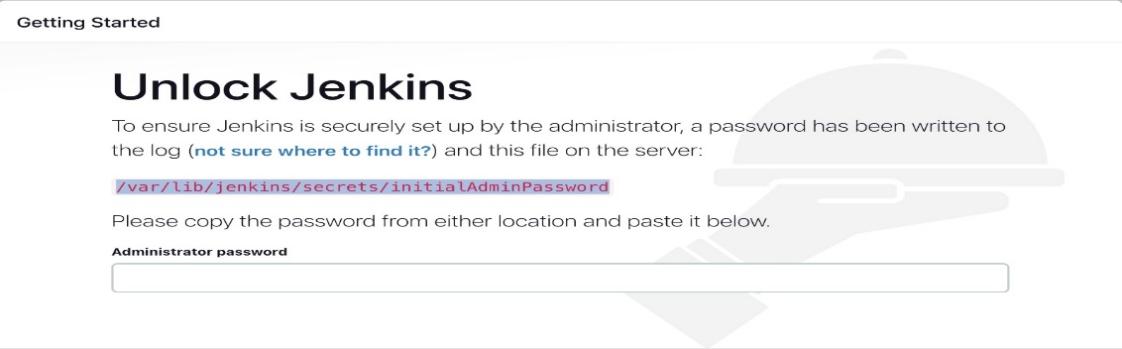
sudo apt install openjdk-8-jdk -y

#wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

#sudo apt-get update -y

#sudo apt-get install jenkins -y



Administrator password get with the help of cat cmd

A screenshot of a computer

Description automatically generated with medium confidence

Create a new Jenkins pipeline:

Jenkins, create a new pipeline job and configure it with the Git repository URL for the Java application.

Add a Jenkins file to the Git repository to define the pipeline stages.

1. **Create the Pipeline Job for the docker image build and pushing repositories:**

pipeline {

agent {

docker {

image 'abhishekf5/maven-abhishek-docker-agent:v1'

args '--user root -v /var/run/docker.sock:/var/run/docker.sock' // mount Docker socket to access the host's Docker daemon

}

}

stages {

stage('Checkout') {

steps {

sh 'echo passed'

//git branch: 'main', url: ' https://github.com/vignesh020919997/Jenkins-Zero-To-Hero’

}

}

stage('Build and Test') {

steps {

sh 'ls -ltr'

// build the project and create a JAR file

sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn clean package'

}

}

stage('Build and Push Docker Image') {

environment {

DOCKER\_IMAGE = " dockervignesh97/ultimate-cicd:${BUILD\_NUMBER}"

// DOCKERFILE\_LOCATION = "java-maven-sonar-argocd-helm-k8s/spring-boot-app/Dockerfile"

REGISTRY\_CREDENTIALS = credentials('docker-cred')

}

steps {

script {

sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && docker build -t ${DOCKER\_IMAGE} .'

def dockerImage = docker.image("${DOCKER\_IMAGE}")

docker.withRegistry('https://index.docker.io/v1/', "docker-cred") {

dockerImage.push()

}

}

}

}

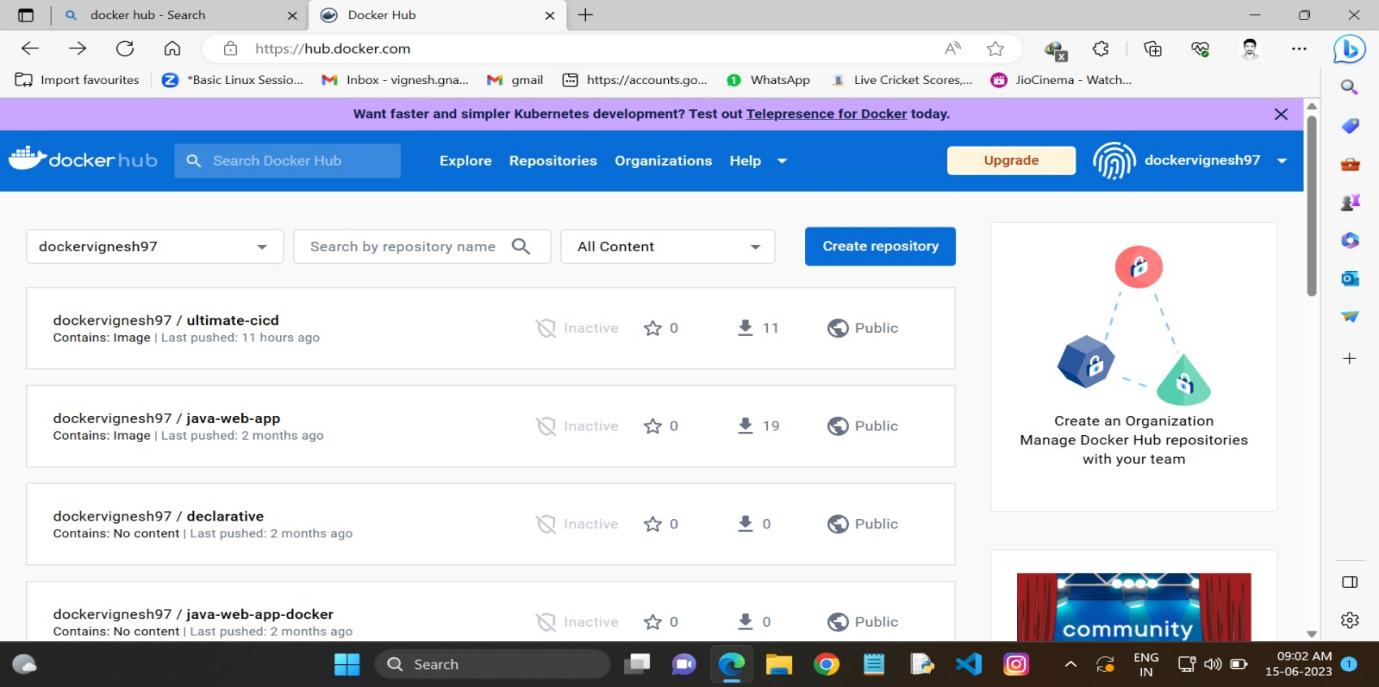


Image pushed in HUB

1. **Install Minikube in the local system:**
2. After installing kubectl
3. Then minikube addon enable ingress
4. Using the kubectl cmd to deploy the yaml file

A screenshot of a computer

Description automatically generated

Execute minikube command

A picture containing text, screenshot, software, multimedia software

Description automatically generated

Execute yaml file in K8

vi) Commands:

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

kubectl apply -f ingress.yaml

v)yaml file Deployment in k8

apiVersion: apps/v1

kind: Deployment

metadata:

name: sprint-boot-app

labels:

app: sprint-boot-app

spec:

replicas: 2

selector:

matchLabels:

app: sprint-boot-app

template:

metadata:

labels:

app: sprint-boot-app

spec:

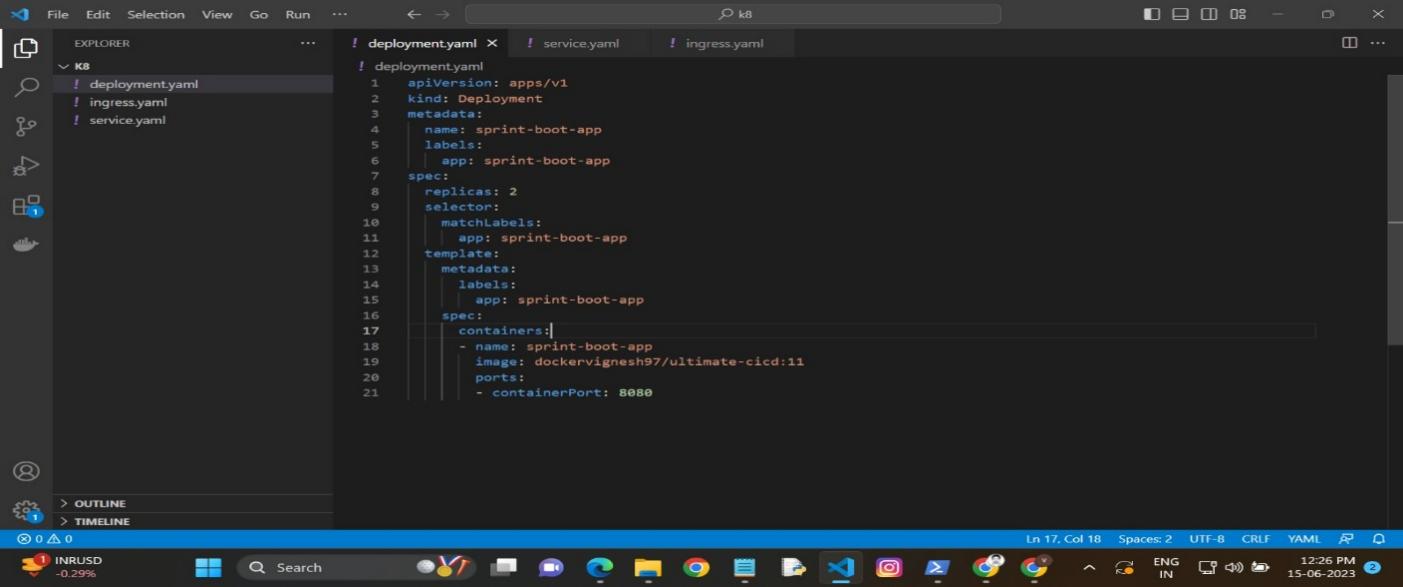
containers:

- name: sprint-boot-app

image: dockervignesh97/ultimate-cicd:11

ports:

- containerPort: 8080



Deployment yaml file

vi) yaml file Service in K8

apiVersion: v1

kind: Service

metadata:

name: spring-boot-app-service

spec:

type: NodePort

ports:

- name: http

port: 8080

targetPort: 8080

protocol: TCP

selector:

app: sprint-boot-app

A screenshot of a computer

Description automatically generated

Service. yaml file

1. yaml file ingress in k8

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: example-ingress

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /$1

spec:

rules:

- host: samraj.co

http:

paths:

- path: /

pathType: Prefix

backend:

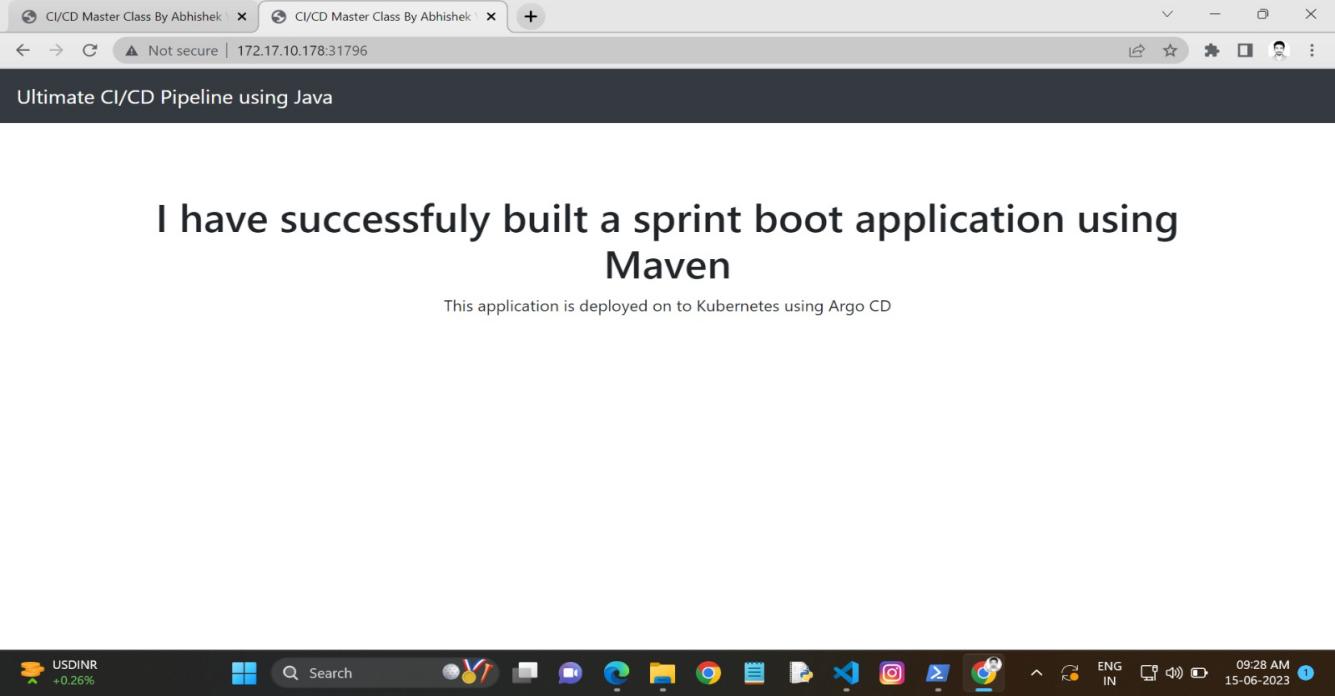
service:

name: spring-boot-app-service

port:

number: 8080

1. **After that I Hit the IP with port number:**



**It works** Run http://<ip\_address>:<port> IP: 172.17.10.178 , Port number: 31796

<http://172.17.10.178:31796> Finally application deploys the K8

1. **The code stuff in a git repository:**

**Git-repositorylink:**https://github.com/vignesh020919997/Jenkins-Zero-To-Hero