**Build Jar:**

**In development machine we built the jar with below command**

mvn package -DskipTests=true

since we have already tested the application before we skip the test for building a jar that will be used for deployment.

**EC2 VM(ubuntu) Creation and Remote-login**

we created the EC2(IaaS) virtual machine with the public-private key and used private key for ssh remote logged-in

We use putty for remote login and which requires public Ip of vm and private key.

**Installing Docker on VM:**

We use installation\_script for docker installation**.**

**Created Dockerfile**.

#base image is jre

FROM adoptopenjdk:16.0.1\_9-jre-hotspot-focal

#copy the jar to root directory of image and renamed the jar to foodapp.jar

COPY target/onlinefoodapp.jar /foodapp.jar

#command that will be executed when container will run

CMD java -jar /foodapp.jar

**Created s3 bucket and uploaded the jar**

Created s3 bucket onlinefoodapp and uploaded the jar

We fetched the jar in the vm using wget command that is below

**Created commit and cloned the repository in virtual machine**

Created commit for the docker file and cloned the repository in vm

git clone **https://github.com/rushichavan-github/kubernateFiles.git**

**Build and Push Docker-image to Docker Hub**

Moved the jar to the target directory and build the image

wget https://onlinefoodapp.s3.amazonaws.com/onlinefoodapp.jar

docker build -t foodapp:latest .

docker tag foodapp:latest public.ecr.aws/m3q9r3m4/ofds:latest

docker push public.ecr.aws/m3q9r3m4/ofds:latest

**Running Docker Container**

Below cmd pulls and runs the docker-container in daemon mode

Container name is app

Our application runs in 8080 port in container and we are mapping with 8080 port of the host(vm)

docker run -d -p 8085:8085 –-name app public.ecr.aws/m3q9r3m4/ofds:latest

**Kubernetes:**

**setting environment**

Installing awscli

awscli is command-client for managing aws services and resources

**aws configure:**

for configuring awscli(access\_key, region)

Installing kubectl from the installation\_script

Installing eksctl from the installation\_script

**Creating cluster with eksctl utility command and creating configuration for kubectl to connect with cluster:**

Below command will create cluster with name sprint2cluster in region us-east-1 nodegroup name is linux-nodes which contains 5 worker nodes and every worker node is of machine type t2.micro

eksctl create cluster --name sprint2cluster  --region us-east-1 --nodegroup-name linux-nodes --node-type t2.micro --nodes 5

creating configuration for kubectl autorization to the created cluseter

aws eks --region us-east-1 update-kubeconfig  --name sprint2cluster

**Creating deployment and service yaml:**

**deployment.yml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: fooddep

namespace: default

spec:

replicas: 1

selector:

matchLabels:

app: fooddep

template:

metadata:

labels:

app: fooddep

spec:

containers:

- image: public.ecr.aws/m3q9r3m4/ofds:latest

name: ofds

ports:

- containerPort: 8085

**Service.yml**

apiVersion: v1

kind: Service

metadata:

name: fooddep

spec:

ports:

- port: 8085

protocol: TCP

targetPort: 8085

selector:

app: fooddep

type: LoadBalancer

we created the commits for deployment and service yml

we pulled the commit in the vm

git pull origin master

**Creating deployment in cluster:**

**kubectl apply -f fdep.yml**

this will create deployment in the Kubernetes cluster with the name university and respective will be created and pods for deployment also created

**Creating service in cluster :**

**kubectl apply -f fser.yml**

since we have already created deployment in previous step our application is available in the internal-cluster network but we want it to be access from outside so we created service of LoadBalancer service

now we can access it from outside using external-ip