Day 8

Login to AWS account

Then create EC2 instance

Open all ports.

After installed all required software please check all version

docker --version

docker images

docker ps

kubectl version

minikube version

now we need to start minikube cluster using below command

minikube start --driver=docker

kubectl that kube control is a command line tool which help to communicate with

Kubernetes cluster environments.

kubectl cluster-info it is use to provide the cluster details.

minikube status it is use to check cluster details.

Application -🡪 Java, Python, Node JS or angular or html

To deploy this application we need to create the image of those application.

Using docker image we can run the application ie java or python or angular etc.

Once you run the image container will start and that application up.

In Kubernetes we can run container directly. If we want to run the container we need to create pods.

Pods is a collection of one or more than container. to run its containers. It is the smallest unit of a Kubernetes application.

In pods can container one container or more than one container.

In one machine we can run more than one pods. Those pods can communicate with each other using cluster ip address with help of some service.

Node1 Node2

Machine1 Machine2

Node

Pods

Container

Image

Application

Kubelet is command line tool responsible to manage the life the pods.

kubectl get pods it display all pods details

kubectl get nodes it display all nodes details.

In Kubernetes using kubectl we can use declarative or imperative manger we can pods as well as other resources.

To create the pods we need one image and image is responsible to run the application.

Using imperative manner we will create simple pods with pre defined image busybox.

Imperative manager use command to create the pods etc.

kubectl run testpod --image=busybox -it

this command is use to create the pods and run the pods for busybox image.

kubectl get pods

kubectl describe pods testpods

this command is use to find the pods details.

Now create the pods using declarative manner.

In declarative manner we need to create .yml or .yaml and write all instruction to create as well run the pods.

So first create the file with any name with extension .yml or .yaml

testpods.yml

apiVersion: v1

kind: Pod

metadata:

name: testpod

spec:

containers:

- name: my-test-container

image: busybox

Esc

:wq!

kubectl apply -f testpods.yml

Now will create pods with our own image to deploy the application in Kubernetes.

In EC2 instance

vi index.html

copy and paste the html code if you want change your name.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h2>Welcome to Simple web page created by Akash deploy using kubernetes</h2>

</body>

</html>

Esc

:wq!

Now we need to create Dockerfile

vi Dockerfile

FROM nginx

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

Esc

:wq!

Now we need to create the image

docker build -t my-web . -f Dockerfile

So now we will publish the our image in Docker hub account.

Before pull

docker tag my-web akashkale/myweb:234

docker login

docker hub account id and password

docker push akashkale/myweb:234

after image created successfully now we need to verify our image.

Vi pods.yml

apiVersion: v1

kind: Pod

metadata:

name: my-server

labels:

app: my-pod

spec:

containers:

- name: web-server-container

image: akashkale/myweb:234

esc

:wq!

Using the file please create pods

kubectly apply -f pods.yml

kubectl get pods

after created pods please verify the status of pods using command as

kubectl get pods

Pods always run inside a cluster environment.

So if you want to expose our application outside cluster we need to use service.

kubectl get service it will provide all service details.

Now create the service.yml file

Vi sevice.yml

apiVersion: v1

kind: Service

metadata:

name: simple-app-service

spec:

type: NodePort

selector:

app: my-pod

ports:

- port: 80

targetPort: 80

protocol: TCP

Esc

:wq!

now create the service

kubectl apply -f service.yml

after create service you can find service details using command as

kubectl get service

Kubernetes cluster environment can be minikube, kind, docker desktop, kubeadm

Minikube cluster so Minikube cluster provide unique ip address to access this service.

Minikube service my-servicename --url

Now we will get IP Address

Then using

CRUL command you can access this application within a cluster

CURL IpAddress

You can access this application within cluster environment not outside or no public ip address.

If we want to allow to access then we need run one of the command as of now.

kubectl port-forward --address 0.0.0.0 service/simple-app-service 8181:80

using EC2 instance public ip address you can access this application

<http://publicIdAddress:8181>

to delete the service we need to run the command as

kubectl delete service servicename

to delete the pods we need to run the command as

kubectl delete pods podsname

In Kubernetes always advisable to create the pods using Kubernetes deployments.

Kubernetes deployment provide the help to check the health of pods and restart the pod’ contains if any failure of the pods. It will to create more than one replica base upon our requirements.

Generally it is advisable to create pods we need to use deployment because deployment manager the creation and scaling of pods.

Create vi deployement.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: my-server

labels:

app: my-server

spec:

replicas: 5

selector:

matchLabels:

app: my-server

template:

metadata:

labels:

app: my-server

spec:

containers:

- name: web-server-container

image: akashkale/myweb:234

kubectl apply -f deployment.yml

to find the deployment details

kubectl get deployments

now create service file with name as

deploye.service.yml

apiVersion: v1

kind: Service

metadata:

name: simple-app-service

labels:

app: my-server

spec:

type: LoadBalancer

selector:

app: my-server

ports:

- port: 80

targetPort: 80

protocol: TCP