## Day 9

Kubernetes Cluster → Master Node

Worker Node 1

**Pods** 

Front end technologies → angular -→ create deployment.yml file to create the pods and expose the service

## Worker Node 2

Pods : backend side technologies and database container.

backend end technologies → spring boot/python/express js -→ create deployment.yml file to create the pods and expose the service

database --→ mongo db or mysql -→ create the deployment.yml file to create the pods and expose the service.

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.

Namespace: name space help use to combine more than one resource with unique name.

If we deploy any application without pods it deploy in default namespace.

Vi namespace.yml

apiVersion: v1

kind: Namespace

metadata:

name: dev

kubectl apply -f namespace.yml

Now we will create the pods using deployment file with namespace as dev.

## vi deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: my-server

labels:

app: my-server

namespace: dev

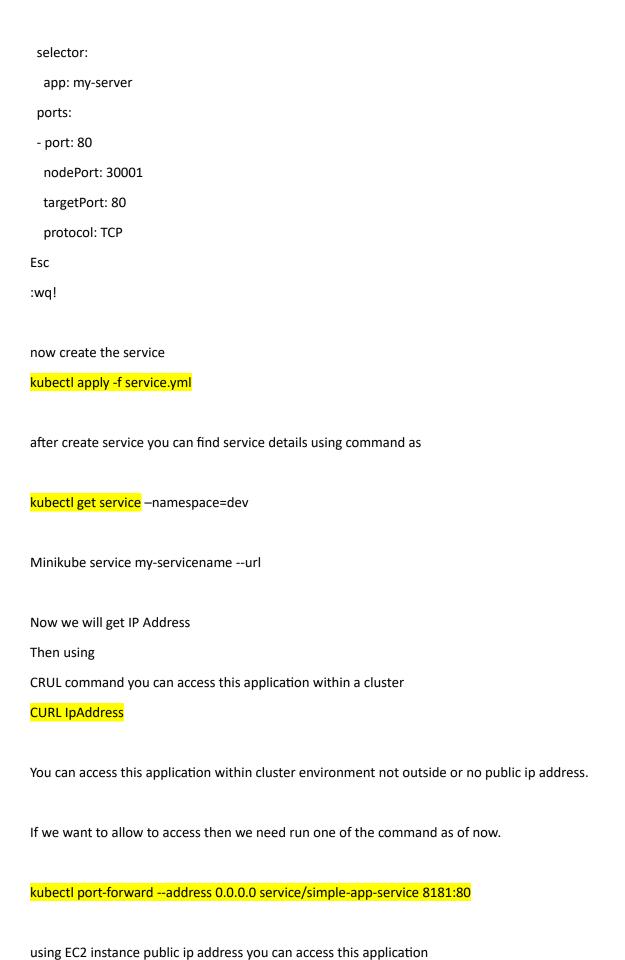
spec:

replicas: 3

selector:

matchLabels:

app: my-server	
template:	
metadata:	
labels:	
app: my-server	
spec:	
containers:	
- name: web-server-container	
image: akashkale/my-simple-kuberneties:1.0	
Esc	
:wq!	
kubectl apply -f deployement.yml	
kubectl get deploymentnamespace=dev	
kubectl get podsnamespace=dev	
if we want to access the container ie application inside a cluster or outside a cluster we need to expose pods with help of container.	
Vi sevice.yml	
apiVersion: v1	
kind: Service	
metadata:	
name: simple-app-service	
labels:	
app: my-server	
namespace: dev	
spec:	
type: LoadBalancer	



### http://publicIdAddress:8181

# minikube stop

## minikube delete

### **Ansible tools**

Server Machine -→ to deploy the application

Web application we need server.

Tomcat, web logic, jboss, Websphere, IIS, nginx, Apache etc.

Web Server or Application Server.

le java, python, express js, angular application, react js

**Database Server** 

MySQL, Mongo Db, Oracle, Db2 etc.

Node or Machine: if installed all required software.

We need to create more than one machine under one cluster environment.

10 machine --→ provide service for Java application

Java7 working and using tomcat 7.

Java8: we need to update java8 in all machine

Using ansible tool we connect all those machine and we can install all required software, database, server through ansible playbook.

In master node we run kubadm init master node

it will generate some token to join

In worker node1 using kubeadm join with token

In worker node1 using kubeadm join with token

10 host machine

4 - → java software java\_host

All four machine ip address.

 $3 \rightarrow$  python software

Python\_host

All three machine ip address

 $3 \rightarrow$  node js software

Node\_host

All three machine ip address

In local machine or VM machine

Create simple index.html page and create Dokerfile using nginx

then push this code in git hub account.

Vm machine

Then create Jenkin pipe line job which is responsible to pull the project

And re-build and push the image to docker hub account whenever we push the

New changes in git hub account.

We can view new changes in browser.

### EC2 instance

Then create Jenkin pipe line job which is responsible to pull the project

And re-build and push the image to docker hub account whenever we push the

New changes in git hub account.

We can view new changes in browser.