\*\*\*\*\*\*\*\*\*\* Diploy python-Django application using kubernetes \*\*\*\*\*\*\*\*\*\*\*\*\*

NOTE :- IF U R CREATING 2 NODES OR MOR THAN THAT THEN U HAVE O GO WITH PAY AS U GO SUPSCRIPTION

links for study:

https://www.hackerxone.com/2021/08/20/steps-to-install-kubectl-eksctl-on-ubuntu-20-04/

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.0.0/deploy/static/provider/cloud/deploy.yaml

https://youtu.be/3Uopc4AFjOY

https://youtu.be/3BnrXapY7zo

step 1 :- create kubernetes cluster on azure portal

a) go inside azure portal search kubernetes services open that and create cluster

b) fill the appripriate details

resource group name --- give ur rescrs name

cluster preset configuration --- keep it as standard

Kubernetes cluster name --- give any name what u want

resign ---

availibility zone --- none

kubernetes version --- keep default one

node size --- u can choose only recomended size u cant change that size

scale method --- autoscaling

node count range --- min=1 maximum=2(as per ur requirement)

Review + create

step 2 :- create kubernetes cluster in azure portal

https://www.youtube.com/watch?v=Zdcrv23WhWE&t=309s (reference video for creating cluster)

step 3 :- create vm in azure portal and access throgh putty

download azure cli on that vm using bellow commonds

Option 1: Install with one command

curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

Option 2: Step-by-step installation instructions

a) Get packages needed for the install process:

sudo apt-get update

sudo apt-get install ca-certificates curl apt-transport-https lsb-release gnupg

b) Download and install the Microsoft signing key:

curl -sL https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/microsoft.gpg > /dev/null

c) Add the Azure CLI software repository:

AZ\_REPO=$(lsb\_release -cs)

echo "deb [arch=amd64] https://packages.microsoft.com/repos/azure-cli/ $AZ\_REPO main" | sudo tee /etc/apt/sources.list.d/azure-cli.list

d) Update repository information and install the azure-cli package:

sudo apt-get update

sudo apt-get install azure-cli

e) Install specific version To view available versions with command:

apt-cache policy azure-cli

f) To install specific version:

sudo apt-get install azure-cli=2.0.31-1~bionic

az --version (check version of azure cli)

step 4 :- install kubectl

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

curl -LO "https://dl.k8s.io/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"

echo "$(cat kubectl.sha256) kubectl" | sha256sum --check

install kubectl cli

sudo az aks install-cli

step 4 :- login through azure cli to azure portal

az login

it will ask u to open link in brouser and past the code inside that browser do that after this u will loged in your azure portal

step 5 :- go inside ur cluster in azure portal click on overview click on connect copy all cmds one by one and past in the putty

below are cmds in connect of my cluster

az account set --subscription 67001b0c-d515-4666-a6f3-efc4f4c89712

az aks get-credentials --resource-group myResrcGrp --name myCluster

# List all deployments in all namespaces

kubectl get deployments --all-namespaces=true

step 6 :- check the nodes are getting or not

kubectl get nodes

if ur getting this type of result then ur correct

NAME STATUS ROLES AGE VERSION

aks-agentpool-12493530-vmss000000 Ready agent 25h v1.22.6

step 7 :- u have to create manifest files for database backend and frontend

How to write manifest file :--

object type : kind

object name : metadata

object components or status : spec

version using to create an object :apiVersion: v1

step 8 :- Create manifest file for database

sudo nano database.yml

( image: moshtab/database\_imageupdated:latest --> this is image in docker hub instead of this image u can use the image in ACR. ACR is the one feature of azure it is one service provided by azure like docker hub but docker hub is public and ACR is private )

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*database.yml\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

labels:

app: mysql

spec:

replicas: 1

selector:

matchLabels:

app: mysql

template:

metadata:

labels:

app: mysql

spec:

containers:

- name: mysql

image: moshtab/database\_imageupdated:latest

resources:

limits:

cpu: 500m

requests:

cpu: 200m

---

kind: Service

apiVersion: v1

metadata:

name: database-service

spec:

selector:

app: mysql

type: ClusterIP

ports:

- name: mysql

port: 3306

targetPort: 3306

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

kubectl apply -f database.yml --> cmd to apply manifest file

kubectl get pods --> cmd for getting created pods

step 9 :- create deployement and service set for backend

( #Note:- The backend Image should contain the host name as the name of database service)

sundo nano backend.yml

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*backend.yml\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

apiVersion: apps/v1

kind: Deployment

metadata:

name: backend

labels:

app: backend

spec:

replicas: 1

selector:

matchLabels:

app: backend

strategy:

type: Recreate

template:

metadata:

labels:

app: backend

spec:

containers:

- name: backend

image: moshtab/backend\_imageupdated:latest

imagePullPolicy: Always

ports:

- name: tcp

containerPort: 8000

resources:

limits:

cpu: 500m

requests:

cpu: 200m

---

kind: Service

apiVersion: v1

metadata:

name: backend

spec:

selector:

app: backend

type: ClusterIP

ports:

- name: backend

port: 8000

targetPort: 8000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

kubectl apply -f backend.yml

kubectl get pods

kubectl describe pod podName

#To enter into a backend pod and check whether the application is running or not

kubectl exec -it NameOfPod -- /bin/bash (NameOfPod will get from: kubectl get pods)

curl http://localhost:8000

#If it shows connection refused to connect 8000 port then do (ps -elf)

python manage.py runserver

#Then u can see the error like it is not connecting to your database service, so change the database service name as host name in settings.py

then apply it again for database service

kubectl apply -f database.yml

step 10 :-create deployement,service and configMap set for frontend

( #Note:- In frontend image it should have the proxypass as the backend service name)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*frontend.yml\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

apiVersion: apps/v1

kind: Deployment

metadata:

name: frontend

labels:

app: frontend

spec:

replicas: 1

selector:

matchLabels:

app: frontend

strategy:

type: Recreate

template:

metadata:

labels:

app: frontend

spec:

containers:

- name: frontend

image: moshtab/frontend\_imageupdated:latest

imagePullPolicy: Always

volumeMounts:

- name: chatapp

mountPath: /etc/nginx/conf.d/

readOnly: true

ports:

- containerPort: 80

resources:

limits:

cpu: 500m

requests:

cpu: 200m

volumes:

- name: chatapp

configMap:

name: configmap

---

kind: Service

apiVersion: v1

metadata:

name: frontend

spec:

selector:

app: frontend

type: ClusterIP

ports:

- name: frontend

port: 80

targetPort: 80

---

kind: ConfigMap

apiVersion: v1

metadata:

name: configmap

data:

nginxconf.conf: |

server {

listen 80 ;

server\_name \_default;

root /new\_chatapp/fundoo;

location / {

proxy\_pass http://backend:8000;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

kubectl apply -f frontend.yml

kubectl get pods

#To enter into a frontend pod and check whether the application is running or not

kubectl exec -it NameOfPod -- /bin/bash (NameOfPod will get from: kubectl get pods)

curl http://localhost

#If it shows connection refused to connect 80 port then do

#Check your nginx config syntax:

nginx -t

#If you get an error, you would need to fix that problem like your having incorrect proxypass in nginx configuration in sites-available folder and then you could restart nginx:

apt install systemctl

systemctl restart nginx

systemctl status nginx

#For trubleshooting nginx server link: https://www.digitalocean.com/community/questions/how-to-troubleshoot-common-nginx-issues-on-linux-server

#If you get Syntax OK when running nginx -t then your confiruation is correct, so I would recommend checking your error logs:

tail -f /var/log/nginx/error.log

step 11 :- Create a domain from freenom

Link: <https://my.freenom.com/clientarea.php?action=domains>

watch this video for creation of domain

https://youtu.be/3Uopc4AFjOY

step 11 :- Route 53

(NOTE :-Route 53 is the service provided by the AWS but in azure we don’t have that service instead of route 53 azure provides private DNS zone service you can try this one also or you can use the domain of other person which it created by aws)

After creation of domain configure it by creating a hosted zone in route 53 and copy four servers from route 53 to

freenom (Manage Domain-Management tools-Nameservers-Use Custom name servers-change Nameservers)

step 12 :-Now install Ingress-Controller

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.0.0/deploy/static/provider/cloud/deploy.yaml

step 13 :- Create ingress to access your application from outside world

sudo nano ingress.yml

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ingress.yml\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress

annotations:

kubernetes.io/ingress.class: nginx

spec:

rules:

- host: www.moshtab.tk

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: frontend

port:

number: 80

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

kubectl apply -f ingress.yml

step 14 :-Now go to route 53 and create a record in that click on alias,select Application Load Balancer, select DNS of LB.

Now check by

curl www.moshtab.tk

or check on browser

www.moshtab.tk

step 15 :-For AutoScaling Install Metric Server

link: <https://docs.aws.amazon.com/eks/latest/userguide/metrics-server.html>

kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

kubectl get deployment metrics-server -n kube-system

step 16 :-For horizontal pod autoscaling do

kubectl autoscale deployment deploymentName --cpu-percent=50 --min=1 --max=10

step 17 :-Now go to duplicate session and increase the load by

kubectl run -i --tty load-generator --rm --image=busybox --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"

step 18 :-Now come to Main session and check whether the pods are increasing or not by

kubectl get hpa