Demonstration on Node Port service:

Step 1: Create a YAML file for Nginx Deployment

To start, connect to the master ec2 instance via SSH as the root user.

You need to create a YAML file for Deploying the web application which will be further exposed using NodePort service.

Name of the YAML file –nginx-deploy.yaml

The content of the YAML files is shown below

cat > nginx-deploy.yaml << EOF

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx-app

spec:

replicas: 1

selector:

matchLabels:

app: nginx-app

template:

metadata:

labels:

app: nginx-app

spec:

containers:

- name: nginx-container

image: nginx:1.7.9

ports:

- containerPort: 80

EOF

In the above manifest file,the name of the Deployment is ‘nginx-deployment’ as highlighted under name section.

The Deployments is creating one replica of nginx-container image 1.7.9 as highlighted in the replicas and containers section respectively.

The Label of the Pod is ‘nginx-app’ as highlighted in the labels section, the same pod label is to be used during the creation of the NodePort Service.

To deploy the above nginx deployment you need to give the command –

kubectl create -f nginx-deploy.yaml

To view the Deployment and the Pod created by this deployment you can give the below commands –

kubectl get deploy

kubectl get pods

Step 2:

Create a YAML file for the NodePort Service

You need to create a YAML file for creating the NodePort service.

Name of the YAML file –nginx-service-sample.yaml

The content of the YAML file is shown below –

cat > nginx-service-sample.yaml << EOF

apiVersion: v1

kind: Service

metadata:

name: nodeport-service

labels:

app: test-nodeport

spec:

selector:

app: nginx-app

type: NodePort

ports:

- nodePort: 31111

port: 80

targetPort: 80

EOF

Or

cat > nginx-service-sample.yaml << EOF

apiVersion: v1

kind: Service

metadata:

name: loadbalancer-service

labels:

app: test-nodeport

spec:

selector:

app: nginx-app

type: LoadBalancer

ports:

- protocol: TCP

port: 80

targetPort: 80

EOF

In the above manifest file, the name of the NodePort service is ‘nodeport-service’ as mentioned in the name section of metadata highlighted.

In the selector section the same Pod label is mentioned which was given while deploying the nginx deployment in the previous step which is ‘nginx-app’.

The Nginx application is being exposed on the port 31111 as mentioned in the node Port section as highlighted.

Kindly ensure the TCP port 31111 is open in the security group of the instances.

To deploy the above NodePort service you need to give the command –

kubectl create -f nginx-service-sample.yaml

To view the Deployment and the Pod created by this deployment you can give the below commands–

kubectl get svc

Step4:

Creating a sample HTMLpage to test the NodePort service

To create a webpage in the Pod deployed, you need to get the name of Pod by giving the command –

kubectl get pods

To go inside the Pod to create the webpage, the below command can be given -kubectl exec [POD-NAME(from previous command)] -it -- /bin/sh

Once inside the Pod the below shown webpage can be created by giving the command shown below

cat <<EOF > /usr/share/nginx/html/test.html

<!DOCTYPE html>

<html>

<head>

<title>Testing..</title>

</head>

<body>

<h1 style="color:rgb(90,70,250);">Hello, replica Service...!</h1>

<h2>Congratulations, you are great :-) </h2>

</body>

</html>

EOF

type exit

The above command creates a html file test.html

Step 5:

Test using Node IP (external IP)

To access the Webpage created in the previous step from the local browser we need to give the URL as shown below –

Kubectl get pods -o wide ----to know the particular worker where the pod is running

http://<Public\_IP\_of\_worker\_instance\_where\_pod\_is\_deplyed>:<nodePort>/<html\_file\_name.html>

In this case the URL will be –http://<Public\_IP\_of\_worker\_instance\_where\_pod\_is\_deplyed>:31111/test.html

Kubectl get pods -o wide ----to know the particular worker where the pod is running

The Public IP of the worker instance where the Pod is deployed can be found out from the AWS EC2 management console.

Step 6:

Cleaning up the resources

The deployments and the service created can be deleted by giving the below commands –

kubectl delete -f <filename/path of deployment manifest file>

kubectl delete -f <filename/path of service manifest file>