Setup kubernetes cluster on AWS

1. Launch a t2.micro instance and install kubectl on that instance
   1. curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl
   2. chmod +x ./kubectl
   3. sudo mv ./kubectl /usr/local/bin/kubectl
2. Install kops
   1. wget https://github.com/kubernetes/kops/releases/download/1.8.1/kops-linux-amd64
   2. chmod +x kops-linux-amd64
   3. sudo mv kops-linux-amd64 /usr/local/bin/kops
3. Create cluster in kubernetes
   1. export KOPS\_STATE\_STORE=k8scluster.k8slab.xyz(BUCKET NAME
   2. export NAME=cluster.k8slab.xyz
   3. kops create cluster $NAME --cloud aws --zones "us-east-1a" --master-zones "us-east-1a" --networking weave --topology public --node-count 2 --node-size t2.small --kubernetes-version 1.8.10 --master-size t2.medium --vpc vpc-01390efaca0ca32c6 --state s3://k8scluster.k8slab.xyz --yes

(The above command will 2 node cluster all are in public subnet with default vpc

* 1. kops update cluster useast1.k8s.appychip.vpc --yes (optional)
  2. Check the availability of cluster:
     1. kops validate cluster --state s3://k8scluster.k8slab.xyz
  3. To check nodes:
     1. kubectl get nodes
  4. To update the nodes use the following file:(optional)
     1. kops edit ig nodes --state s3://k8scluster.k8slab.xyz
     2. kops rolling-update cluster --name cluster.k8slab.xyz --state s3://k8scluster.k8slab.xyz --yes

* 1. To get cluster info, which includes url of kubernetes master and kuberDNS use the following command:
     1. kubectl cluster-info
     2. To get credentials,use
        1. kubectl config view

(it will return username and password)

1. Updated the nodes(Optional)
   1. kops edit ig nodes --state s3://k8scluster.k8slab.xyz
   2. Change the instance size or type
   3. kops update cluster --name $NAME --state s3://k8scluster.k8slab.xyz --yes
   4. Incase if it it ask your to rolling update the cluster then run the following command:

kops rolling-update cluster --name $NAME --state s3://k8scluster.k8slab.xyz --yes

1. Create deployment and service. Your can create deployment and service in two way, either you can directly run the command or create your own yaml file for service and deployments

* **First Approach: Run the below commands to create service and deployment**
  1. Let’s go through first approach
     1. Deploy python Container
     2. Create a python deployment

( use nginx image directly or use the image that have created in previous session)

* + 1. kubectl run nginx-deployment --image=nginx --replicas=2 --port=80

Or

kubectl run python-deployment --image=aashishch65/demo --replicas=2 --port=5000

* + 1. Once it is deployed check the deployment and pods

Check the pods: kubectl get pods

Check the deployment: kubectl get deployments

Expose the deployment as service. This will create an ELB in front of those 2 containers and allow us to publicly access them:

* 1. kubectl expose deployment nginx-deployment--port=80 --type=LoadBalancer

Or

Kubectl expose deployment python-deployment --port=5000 --type=LoadBalancer

* 1. To access the external url just type the command
     1. kubectl get services
* **Second Approach: Create python-deployment.yaml and python-service.yaml**

1. Create **python-deployment.yaml** file

apiVersion: apps/v1beta1

kind: Deployment #deployment name

metadata:

name: python-deployment1

labels:

app: python

spec:

replicas: 3 # creates three replicated Pods

selector: # it defines how the Deployment finds which Pods to manage. In this case, we simply select on one label defined in the Pod template

matchLabels:

app: python

template:

metadata:

labels:

app: python

spec: # indicates that the Pods run one container, python

containers:

- name: python

image: aashishch65/demo:latest

ports:

- containerPort: 5000

**Command: kubectl create -f python-deployment.yaml**

1. Create **python-service.yaml** file

**apiVersion: v1**

**kind: Service**

**metadata:**

**labels:**

**name: pythonservice**

**name: pythonservice**

**spec:**

**ports:**

**- port: 5000**

**selector:**

**app: python**

**type: LoadBalancer**

**Command: kubectl create -f python-service.yaml**

1. Roll out and Roll back update
   1. If you want to Roll out some new changes for example want to update the docker image, then use the following commands:
      1. **kubectl edit deployment/python-deployment1** (change the name of docker image)

Or

Directly update the image

**kubectl set image deployment/python-deployment1 python=aashishch65/demo:latest**

* + 1. Update the rollout

**kubectl rollout status deployment python-deployment1**

* 1. If you want to Roll back some change to the previous images use following commands:
     1. **Get the details of rollout version and then rollback to specific version**

**kubectl rollout history deployment/python-deployment1**

**(It will return rollout version)**

**kubectl rollout history deployment/python-deployment1 --revision=4**

**(It will return the details of specific rollout version)**

**Or**

**Now Rollout to some specific version**

**kubectl rollout undo deployment/python-deployment1 --to-revision=4**

1. Create replication-controller and service. (optional)

python-replication.yaml

apiVersion: v1

kind: ReplicationController

metadata:

name: python

spec:

replicas: 2

selector:

app: python

template:

metadata:

name: python

labels:

app: python

spec:

containers:

- name: demo

image: aashishch65/demo:latest

ports:

- containerPort: 5000

Command: kubectl create -f python-replication.yaml

Python-service.yaml

apiVersion: v1

kind: Service

metadata:

labels:

name: pythonservice

name: pythonservice

spec:

ports:

# The port that this service should serve on.

- port: 5000

# Label keys and values that must match in order to receive traffic for this service.

selector:

app: python

type: LoadBalancer

Command: kubectl create -f python-service.yaml

* kubectl get services (Copy the elb url in browser with :5000)

1. Delete the resources
   1. Delete the service
      1. kubectl delete service pythonservice
   2. Delete the replication controller
      1. kubectl delete rc python
   3. Delete the deployment
      1. Kubectl delete deployment python-deployment
2. To delete the cluster
   1. kops delete cluster --name=cluster.k8slab.xyz --state s3://k8scluster.k8slab.xyz --yes

References:

Kops Lifecycle: <https://github.com/kubernetes/kops/blob/master/docs/boot-sequence.md>