# Kubernetes Lab

# Connect to Kubernetes Master Node using the Public IP

# Open an SSH client.

# Locate your private key file (keypair.pem)

# # chmod 400 keypair.pem

# Now you will be able to SSH using your Public DNS/Public IP

# ssh -i "keypair.pem" [ec2-user@ec2-xx-xx-xx-xx.ap-south-1.compute.amazonaws.com](mailto:ec2-user@ec2-xx-xx-xx-xx.ap-south-1.compute.amazonaws.com)

# NB:- If SSH connectivity does not work, see changing the chmod to 600 (chmod 600 singaporekeypair.pem)

# [ec2-user@ip-xx-xx-xx-xx ~]$ sudo su

# # yum install -y git [Only If GIT is not installed in your EC2 instance]

# Steps to Check out from git repository

# # mkdir gitrepo

# # cd gitrepo

# # git init

# # git config --global user.email "mymail@email.com"

# # git config --global user.name " git user name"

# # git clone https://github.com/nevin-cleetus/k8.git

# Ensure

# 1. Internet is working

# 2. Nobody should be connected to vpn or any other proxy.

# 3. Disable firewall if enabled.

# 

# 

# Lab Exercise

# POD

# Verify the Kubernetes Master and Worker Node(s) are in running state

# [ec2-user@ip-xx-xx-xx-xx ~]$ sudo su

[root@ip- kubernetes] kubectl get nodes

Confirm both Master and Worker node is in Ready state.

NAME STATUS ROLES AGE VERSION

ip- xxxxxxx Ready master 15m v1.18.0

ip-xxxxxxx Ready <none> 10m v1.18.0

[root@ip- kubernetes] cd gitrepo/k8/day1/yaml/pod/

**Lab 1:- Nginx POD**

1. Create the nginx pod using the nginx-pod.yaml file

[root@ip- kubernetes] kubectl create -f nginx-pod.yaml

Expected Output: pod/nginx created

1. Check the status of the POD

[root@ip- kubernetes] kubectl get pods

Expected Output

NAME READY STATUS RESTARTS AGE

nginx 1/1 Running 0 10s

1. To get more details of the POD, we can use the ‘get pods’ command with ‘-o wide’ option. Use the IP to confirm the Nginx is running on port 80.

[root@ip- kubernetes] kubectl get pods -o wide

Expected Output

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

nginx 1/1 Running 0 6m19s 192.168.188.26 ip-172-31-26-127.ap-southeast-1.compute.internal <none> <none>

[root@ip- kubernetes] curl 192.168.188.26:80

Expected Output : - Nginx landing page with the message ‘Thank you for using nginx.’

1. To connect to the running container

[root@ip- kubernetes] kubectl exec -it nginx -- /bin/sh

#ls

#exit

1. To see the details of the POD created.

[root@ip- kubernetes] kubectl describe pod nginx

Expected Output: - Details of the pod including IP, Resource Details, Status etc

1. To see the POD logs

[root@ip- kubernetes] kubectl logs nginx

Expected Output: - log output on the console

1. To get the yaml representation of the running pod

[root@ip- kubernetes] kubectl get pod nginx -o yaml

Expected Output: YAML representation of the POD

# Lab Exercise 2

# Replication Controller

# Verify the Kubernetes Master and Worker Node(s) are in running state

# 

[root@ip- kubernetes] cd gitrepo/k8/day1/yaml/replication\_controller

1. Create multiple instances of nginx pod using the ng-replication-controller.yaml file

[root@ip- kubernetes] kubectl create -f ng-replication-controller.yaml

Expected Output: replicationcontroller/nginx created

1. Check the status of the POD

[root@ip- kubernetes] kubectl get rc

[root@ip- kubernetes] kubectl get pods

Expected Output

Specified number of nginx pod instances should get created.

# Lab Exercise 3

# ReplicaSet

# Verify the Kubernetes Master and Worker Node(s) are in running state

# 

[root@ip- kubernetes] cd gitrepo/k8/day1/yaml/replicationset

1. Create multiple instances of nginx pod using the nginx-replicaset.yaml file

[root@ip- kubernetes] kubectl create -f nginx-replicaset.yaml

Expected Output: replicaset/nginx created

1. Check the status of the POD

[root@ip- kubernetes] kubectl get rs

[root@ip- kubernetes] kubectl get pods

Expected Output

Specified number of nginx pod instances should get created.

# Lab Exercise 4

# Deployment

# Verify the Kubernetes Master and Worker Node(s) are in running state

# 

[root@ip- kubernetes] cd gitrepo/k8/day1/yaml/deployment

1. Create multiple instances of nginx pod using the nginx-deployment.yaml file

[root@ip- kubernetes] kubectl create -f nginx-deployment.yaml

Expected Output: deployment.apps/nginx-deployment created

1. Check the status of the POD

[root@ip- kubernetes] kubectl get deployment

Expected Output

NAME DESIRED CURRENT READY AGE

nginx-deployment-6bbb7cb484 2 2 2 100m

[root@ip- kubernetes] kubectl get rs

[root@ip- kubernetes] kubectl get pods

Specified number of nginx pod instances should get created.

NAME READY STATUS RESTARTS AGE

nginx-deployment-6bbb7cb484-572tx 1/1 Running 0 102m

nginx-deployment-6bbb7cb484-j58f4 1/1 Running 0 102m

# Modify the deployment to use different image

[root@ip- kubernetes] kubectl set image deployment nginx-deployment nginx=nginx:1.15 --record

deployment.apps/nginx-deployment image updated

[root@ip- kubernetes] kubectl get rs

A new replication set will be created. Run the above step few times to see the behavior

NAME DESIRED CURRENT READY AGE

nginx-deployment-6bbb7cb484 1 1 1 110m

nginx-deployment-7964d8b76c 2 2 1 9s

[root@ip-172-31-21-212 replicaset]# kubectl get rs

NAME DESIRED CURRENT READY AGE

nginx-deployment-6bbb7cb484 0 0 0 110m

nginx-deployment-7964d8b76c 2 2 2 13s

[root@ip-172-31-21-212 replicaset]# kubectl get rs

NAME DESIRED CURRENT READY AGE

nginx-deployment-6bbb7cb484 0 0 0 110m

nginx-deployment-7964d8b76c 2 2 2 14s

[root@ip-172-31-21-212 replicaset]# kubectl rollout history deployment nginx-deployment

Expected Output

deployment.apps/nginx-deployment

REVISION CHANGE-CAUSE

1 <none>

2 kubectl set image deployment nginx-deployment nginx=nginx:1.15 --record=true

**Rollback changes made on the deployment**

[root@ip- kubernetes] kubectl set image deployment nginx-deployment nginx=nginx:1.16 -- record

deployment.apps/nginx-deployment image updated

[root@ip-replicaset]# kubectl rollout history deployment nginx-deployment --revision=1

deployment.apps/nginx-deployment with revision #1

Pod Template:

Labels: pod-template-hash=6bbb7cb484

software=light-reverse-proxy

Containers:

nginx:

Image: nginx

Port: 80/TCP

Host Port: 0/TCP

Environment: <none>

Mounts: <none>

Volumes: <none>

[root@ip-replicaset]# kubectl rollout undo deployment nginx-deployment --to-revision=1

**Update the No of replica using scale**

[root@ip-replicaset]# kubectl scale --replicas=4 deployment nginx-deployment

[root@ip-replicaset]# kubectl get rs

**Pause and Resume deployment**

Allow us to pause the deployment and apply the changes without a rollout. Once the resume is executed only the updated changes will be rolled out.

[root@ip-replicaset]# kubectl rollout pause deployment nginx-deployment

[root@ip-replicaset]# kubectl set image deployment nginx-deployment nginx=nginx:1.16 --record

root@ip-replicaset]#kubectl get rs

[root@ip-replicaset]# kubectl rollout resume deployment nginx-deployment

Now check the status to confirm the new deployment is rolled out.

root@ip-replicaset]#kubectl get rs

**Deployment Strategy**

**Strategy - Recreate**

[root@ip-replicaset]# kubectl apply -f nginx-deployment-recreate.yaml

[root@ip-replicaset]#kubectl get rs

**Strategy - RollingUpdate**

[root@ip-replicaset]# kubectl apply -f nginx-deployment-rolling.yaml

[root@ip-replicaset]#kubectl get rs

# Lab Exercise 5

# Service

# Verify the Kubernetes Master and Worker Node(s) are in running state

# 

[root@ip- kubernetes] cd gitrepo/k8/day1/yaml/service

1. Create the nginx pod using the nginx-pod.yaml file

[root@ip- kubernetes] kubectl create -f nginx-pod.yaml

Expected Output: pod/nginx created

1. Check the status of the POD

[root@ip- kubernetes] kubectl get pods

Expected Output

NAME READY STATUS RESTARTS AGE

nginx 1/1 Running 0 10s

# NodePort Service

[root@ip- kubernetes] cd gitrepo/k8/day1/yaml/service

[root@ip- kubernetes] kubectl create -f nginx-nodeport.yaml

service/nginx-nodeport-service created

[root@ip- kubernetes] kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 3d14h

nginx-nodeport-service NodePort 10.107.191.170 <none> 80:30080/TCP 14s

Now you will be able to access the Nginx page from outside the network.

Go to browser and try to access http://<any of the Public IP in your cluster>:30080/

**ClusterIP Service**

[root@ip- kubernetes] cd gitrepo/k8/day1/yaml/service

[root@ip- kubernetes] kubectl create -f nginx-clusterip.yaml

service/nginx-clusterip created

[root@ip- kubernetes] kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 3d14h

nginx-clusterip-service ClusterIP 10.105.213.188 <none> 80/TCP 82s

The access to the service will be available from any of the nodes in the cluster. It will not be available outside the cluster.

[root@ip- kubernetes] curl 10.105.213.188:80

Expected output: Will display the landing page of Nginx server

Go to browser and try to access http://<any of the Public IP in your cluster>:30080/

The page should not be accessible

**LoadBalancer Service**

[root@ip- kubernetes] cd gitrepo/k8/day1/yaml/service

[root@ip- kubernetes] kubectl create -f nginx-loadbalancer.yaml

service/nginx-clusterip created

[root@ip- kubernetes] kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 3d14h

nginx-loadbalancer-service LoadBalancer 10.106.98.219 <pending> 80:30610/TCP 42s

The access to the service will be available from any of the nodes in the cluster. It will not be available outside the cluster.

[root@ip- kubernetes] curl http://<Public IP>: 30610

Expected output: Will display the landing page of Nginx server