# Kubernetes Lab

# Connect to Kubernetes Master Node using the Public IP

# Open an SSH client.

# Locate your private key file (singaporekeypair.pem)

# # chmod 400 singaporekeypair.pem

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

# ssh -i "singaporekeypair.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

# # git init

# # git config --global user.email "nevin.cleetus@gmail.com"

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

# # mkdir gitrepo

# # cd gitrepo

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

# Ensure

# 1. Internet is working

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

# 3. Disable firewall if enabled.

# 

# 

# 

# Lab Exercise 1

# DaemonSet

# 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/kubernetes/day\_02\_kubernetes/yaml/daemonset

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

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

Expected Output: daemonset.apps/nginx-daemonset created

1. Check the status of the daemonset and pods

[root@ip- kubernetes] kubectl get ds

NAME DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE

nginx-daemonset 2 2 2 2 2 <none> 11m

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

Expected Output

Nginx Pod will created and started in each Node.

nginx-daemonset-8pn24 1/1 Running 0 23s

nginx-daemonset-zrmst 1/1 Running 0 23s

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 pod is created in all nodes.

[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.’

# Daemonset using NodeSelector

# Delete the daemonset created in the above step

# [root@ip- kubernetes] kubectl delete daemonset nginx-daemonset

[root@ip- kubernetes] kubectl create -f nginx-daemonset-with-nodeselector.yaml

[root@ip- kubernetes] kubectl get daemonset

Expected Result: - nginx-daemonset should be returned

[root@ip- kubernetes] kubectl get pods

Expected Result :- No records as there is no node configured with the selector gpu=true

Let’s update the node with the label and check. Replace the highlighted section with your public IP address of the worker node.

[root@ip- kubernetes] kubectl label node <node name> gpu=true

Now execute the get pods to confirm the POD is running in one node where we set the label.

[root@ip- kubernetes] kubectl get pods

nginx-daemonset-465jv 1/1 Running 0 27s 192.168.24.70 ip-172-31-31-122.ap-southeast-1.compute.internal <none> <none>

# Lab Exercise 2

# Volume

# emptyDir

# 

[root@ip- kubernetes] cd gitrepo/kubernetes/day\_02\_kubernetes/yaml/volumes

1. Create the Pod with emptyDir volume using the emptyDir.yaml file

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

Expected Output: pod/emptydir-pod created

1. Check the status of the POD

[root@ip- kubernetes] kubectl get pods

1. [root@ip- kubernetes] kubectl exec -it <pod name> /bin/sh

Go to the mount location and write a file under /cache directory.

#cd /cache

#touch 1.txt

1. Delete the POD and re-create it, the data will be gone.

[root@ip- kubernetes] kubectl delete pod emptydir-pod

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

[root@ip- kubernetes] kubectl exec -it <pod name> /bin/sh

#cd /cache

# ls

Expected Output : - Empty Directory

# hostPath

1. [root@ip- kubernetes] cd gitrepo/kubernetes/day\_02\_kubernetes/yaml/volumes
2. Create the Pod with hostPath volume using the hostPath.yaml file

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

Expected Output: "nginx-hostpath-pod" deleted

1. [root@ip- kubernetes] kubectl exec -it <pod name> /bin/sh

Go to the mount location and write a file under /nginx-hostpath directory.

1. #cd /nginx-hostpath
2. #touch 1.txt

#exit and delete the POD

[root@ip- kubernetes] kubectl delete pod nginx-hostpath-pod

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

Expected Output: "nginx-hostpath-pod" deleted

1. [root@ip- kubernetes] kubectl exec -it <pod name> /bin/sh

Go to the mount location and write a file under /nginx-hostpath directory.

The file created during the first step 1.txt should be present.

# Lab Exercise 3

# Secrets

[root@ip- kubernetes] cd gitrepo/kubernetes/day\_02\_kubernetes/yaml/secrets

Create a secret

1. [root@ip- kubernetes] kubectl create -f busybox-secret.yaml

secret/busybox-secret configured

1. [root@ip- kubernetes] kubectl get secrets

Expected result : - Display the busybox-secret details

1. [root@ip- kubernetes] kubectl describe secrets busybox-secret

Expected result : - Display the busybox-secret data

Create a POD and consume the secret using environment variable

[root@ip- kubernetes] kubectl create -f busybox-secret-by-env.yaml

Expected result : - POD busybox-by-env created

[root@ip- kubernetes] kubectl exec -it <pod-name> /bin/sh

# echo $username

Expected output : - admin

Create a POD and consume the secret from volume path

[root@ip- kubernetes] kubectl create -f busybox-secret-vol.yaml

[root@ip- kubernetes] kubectl exec -it <pod-name> /bin/sh

#ls /mydata

#cat username

# Lab Exercise 4

# ConfigMap

[root@ip- kubernetes] cd gitrepo/kubernetes/day\_02\_kubernetes/yaml/configmap

Create a configMap

1. [root@ip- kubernetes] kubectl create configmap example-literal-config --from-literal=maxmemory=100MB

configmap/example-redis-config created

1. [root@ip- kubernetes] kubectl get configmap

Expected result : - Display the configmap/example-redis-config details

1. [root@ip- kubernetes] kubectl describe configmap example-redis-config

Expected result : - Display the example-redis-config data

Create a POD and consume the configMap using environment variable

[root@ip- kubernetes] kubectl create -f configMap-env.yaml

Expected result : - pod/busybox-configmap created

[root@ip- kubernetes] kubectl exec -it <pod-name> /bin/sh

# env

Expected output : - busybox-memory-env=100MB

Create a POD and consume the secret from volume path

[root@ip- kubernetes] kubectl create configmap example-redis-config --from-file=redis-config

Expected Output:- configmap/example-redis-config created

[root@ip- kubernetes] kubectl create -f configmap-redis-file.yaml

Expected Output: - pod/redis created

[root@ip- kubernetes] kubectl exec -it redis cat /redis-master/redis.conf

Expected Output: - maxmemory 2mb

maxmemory-policy allkeys-lru

# Lab Exercise 5

# Jobs & CronJobs

[root@ip- kubernetes] cd gitrepo/kubernetes/day\_03\_kubernetes/yaml/jobs

Create a Jobs using the yaml below

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

Expected result : - job/hellojob created

[root@ip- kubernetes] kubectl get jobs

Expected Output: - Out the details of the job

[root@ip- kubernetes] kubectl get pods

Expected Output: - Display the pods which gets created by the job

Create a CronJobs using the yaml below

[root@ip- Kubernetes] cd gitrepo/kubernetes/day\_03\_kubernetes/yaml/cronjobs

[root@ip- kubernetes] kubectl create -f cronjob-template.yaml

Expected result : - mycronjob created

[root@ip- kubernetes] kubectl get pods

Expected Output: - Display the pods which gets created by the pod