# 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 "<your git email address>"

# # git config --global user.name "<your git username>"

# # 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

# Volume

# emptyDir

# 

[root@ip- kubernetes] cd gitrepo/k8/day1/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/k8/day1/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 2

# Jobs & CronJobs

[root@ip- kubernetes] cd gitrepo/k8/day2/yaml/job

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/k8/day2/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

# Lab Exercise 3

# Secrets

[root@ip- kubernetes] cd gitrepo/k8/day2/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 $myusername

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 myusername

# Lab Exercise 4

# ConfigMap

[root@ip- kubernetes] cd gitrepo/k8/day2/yaml/configmap

Create a configMap

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

# Scheduling

**Manual**

[root@ip- kubernetes] cd gitrepo/k8/day2/yaml/scheduling/manual

[root@ip- kubernetes] kubectl get nodes

Copy one of the worker node name where you want to manually schedule the pod

For Exp:- ip-xxxxx-worker1

[root@ip- kubernetes]vi manual-scheduling.yaml

Update the nodeName: <your worker node ip>. Save the file and apply

[root@ip- kubernetes] kubectl apply -f manual-scheduling.yaml

Expected Result:-The Pod should get provisioned in the configured node.

**Taints and Tolerations**

[root@ip- kubernetes] cd gitrepo/k8/day2/yaml/scheduling/taints

Apply the taint on one of the worker node

[root@ip- kubernetes] kubectl taint nodes <nodename> color=green:NoSchedule

Expected output : node/<nodename> tainted

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

[root@ip- kubernetes] kubectl get pods

Expected Output: - The Nginx pod should get provisioned in the tainted node (

NB: - Please keep in mind that, the taints and toleration does not guarantee the pod will get provisioned in the tainted node. It ensure that only the PODs tolerant to the taint created shall arrive at the Node.

To remove the taint from the node

[root@ip- kubernetes] kubectl taint nodes <nodename> color=green:NoSchedule-

Why Master Node, the PODs are not provisioned. In order to remove the taint

kubectl taint nodes --all node-role.kubernetes.io/master-

**NodeAffinity**

[root@ip- kubernetes] cd gitrepo/k8/day2/yaml/scheduling/affinity

Apply the label on one of the worker node

[root@ip- kubernetes] kubectl get nodes

Select the node name where we would like to apply the label

[root@ip- kubernetes] kubectl label nodes <node-name> env=production

Expected output : node/<nodename> label applied

[root@ip- kubernetes] kubectl apply -f node-affinity.yaml

[root@ip- kubernetes] kubectl get pods

The POD should get provisioned only on the node we have matching label