POC : Kubernetes Assignments

Introduction

Kubernetes

- Kubernetes is an open-source container orchestration platform that for automating deployment, scaling, and management of containerized applications.

Features of kubernetes

- **Orchestration:**
  Clustering any no of containers on different hardwares
- **Auto scaling :**
  It is a feature in which the cluster is capable of increasing the number of nodes as the demand for service response increases and vice-versa.
- **Auto healing :**
  New containers in place of crashed containers
- **Load balancing :** Distributing Incoming Traffic across nodes
- **Rollback :** Going to previous versions

Kuberntes uses various types of objects.

1. **Pod:** This is a layer of abstraction on top of a container. This is the smallest object that kubernetes can work on. In the pod, we have the container. kubectl commands will work on the pod and pod communicates there instructions to the container.
2. **Service Object:** This is used for port mapping and network load balancing.
3. **NameSpace**: This is used for creating partitions in the cluster. Pods running in a namespace cannot communicate with other pods running in other namespace.
4. **Secrets** : This is used for passing encrypted data to the pods such as a password, a token, or a key.
5. **ReplicaSet / Replication Controller**: This is used for managing multiple replicas of a pod to perform activities like load balancing and autoscaling.
6. **Deployment**: This is used for performing all activites that a ReplicaSet can do. It can also handle rollling updates.

Command to create a pod

```
kubectl run --image nginx   webserver --port=5701  (webserver is pod name )
```

To see list of pod

```
kubectl get pods
```

To delete the pod

```
kubectl delete pods webserver
```

This one way of creating Pods by defining all parameteres in the command line itself.

Also Kubernetes performs container orchestration by using **Definition files or Manifest**. Definition files are yaml files.

Before Starting With this Exersise we should ready with

1. Master Node

2. Worker Node
3. Docker should be Installed on both Nodes
4. Installed Kubeadm in both master and Worker Nodes.
5. Initialization of Master Node.
6. And connect Worker Node with Master Node.

```
[root@master ec2-user]# hostnamectl set-hostname master
[root@master ec2-user]# exec bash
[root@master ec2-user]# sudo su -
Last login: Wed Oct 26 05:25:01 UTC 2022 on pts/0
[root@master ~]#
```

```
[root@ip-172-31-25-255 ec2-user]# hostnamectl set-hostname worker
[root@ip-172-31-25-255 ec2-user]# exec bash
[root@worker ec2-user]# sudo su -
Last login: Wed Oct 26 05:25:55 UTC 2022 on pts/0
[root@worker ~]#
```

```
[root@master ~]# docker --version
Docker version 20.10.17, build 100c701
[root@master ~]#
```

```
[root@worker ~]# docker --version
Docker version 20.10.17, build 100c701
[root@worker ~]#
```

```
Complete!
[root@master ~]# systemctl restart docker && systemctl enable docker && systemctl restart kubelet && systemctl enable kubelet
Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /usr/lib/systemd/system/kubelet.service.
[root@master ~]#
```

```
Complete!
[root@worker ~]# systemctl restart docker && systemctl enable docker && systemctl restart kubelet && systemctl enable kubelet
Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /usr/lib/systemd/system/kubelet.service.
[root@worker ~]#
```

```
kubeadm join 172.31.22.2:6443 --token jahg03.8b1rk5u48su7599d \
        --discovery-token-ca-cert-hash sha256:52c1d3483a94adfd1d6683a287a90fec85f5fa0a908be3f1300616ff1cc4f3cb
[root@master ~]#
```

kubeadm join 172.31.22.2:6443 --token jahg03.8b1rk5u48su7599d \
      --discovery-token-ca-cert-hash
sha256:52c1d3483a94adfd1d6683a287a90fec85f5fa0a908be3f1300616ff1cc4f3cb

kubectl get nodes

```
[root@master ~]# kubectl get nodes
NAME     STATUS     ROLES           AGE     VERSION
master   NotReady   control-plane   4m19s   v1.25.3
worker   NotReady   <none>          26s     v1.25.3
```

kubectl get nodes

```
[root@master ~]# kubectl get nodes
NAME     STATUS   ROLES           AGE     VERSION
master   Ready    control-plane   6m49s   v1.25.3
worker   Ready    <none>          2m56s   v1.25.3
```

---

1. List all the namespaces in the cluster

kubectl get namespaces
kubectl get ns

```
[root@master ~]# kubectl get namespaces
NAME              STATUS   AGE
default           Active   148m
kube-node-lease   Active   148m
kube-public       Active   148m
kube-system       Active   148m
[root@master ~]# kubectl get ns
NAME              STATUS   AGE
default           Active   149m
kube-node-lease   Active   149m
kube-public       Active   149m
kube-system       Active   149m
[root@master ~]#
```

2. List all the pods in all namespaces

kubectl get po --all-namespaces

```
[root@master ~]# kubectl get po --all-namespaces
NAMESPACE     NAME                                      READY   STATUS    RESTARTS   AGE
kube-system   calico-kube-controllers-59697b644f-ccw5c  1/1     Running   0          144m
kube-system   calico-node-5n27g                         1/1     Running   0          144m
kube-system   calico-node-x9r9k                         1/1     Running   0          144m
kube-system   coredns-565d847f94-972mg                  1/1     Running   0          149m
kube-system   coredns-565d847f94-hjthz                  1/1     Running   0          149m
kube-system   etcd-master                               1/1     Running   0          150m
kube-system   kube-apiserver-master                     1/1     Running   0          150m
kube-system   kube-controller-manager-master            1/1     Running   0          150m
kube-system   kube-proxy-j4tr5                          1/1     Running   0          146m
kube-system   kube-proxy-jdmmx                          1/1     Running   0          149m
kube-system   kube-scheduler-master                     1/1     Running   0          150m
[root@master ~]#
```

3. List all the pods in the particular namespace

kubectl get po -n <namespace name>

```
[root@master ~]# kubectl get po -n kube-system
NAME                                      READY   STATUS    RESTARTS   AGE
calico-kube-controllers-59697b644f-ccw5c  1/1     Running   0          145m
calico-node-5n27g                         1/1     Running   0          145m
calico-node-x9r9k                         1/1     Running   0          145m
coredns-565d847f94-972mg                  1/1     Running   0          151m
coredns-565d847f94-hjthz                  1/1     Running   0          151m
etcd-master                               1/1     Running   0          151m
kube-apiserver-master                     1/1     Running   0          151m
kube-controller-manager-master            1/1     Running   0          151m
kube-proxy-j4tr5                          1/1     Running   0          147m
kube-proxy-jdmmx                          1/1     Running   0          151m
kube-scheduler-master                     1/1     Running   0          151m
```

4. List all the services in the particular namespace

kubectl get svc -n <namespace name>

```
[root@master ~]# kubectl get svc -n kube-system
NAME       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)                  AGE
kube-dns   ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   152m
```

5. Create an nginx pod in a default namespace and verify the pod running

// creating a pod
kubectl run nginx --image=nginx --restart=Never
// List the pod
kubectl get po

```
[root@master ~]# kubectl run nginx --image=nginx --restart=Never
pod/nginx created
[root@master ~]# kubectl get po
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          15s
```

6. create a nginx pod with definition file or Yaml file

* create an empty yaml file

touch pod-definition1.yml

* Now need to define necessasary Key value pairs in that Yaml file

vim pod-definition1.yml

```
---
apiVersion: v1
kind: Pod
metadata:
 name: nginx-pod
 labels:
  author: arun
  version : v1
spec:
 containers:
  - name: appserver
    image: nginx
```

save the file
:wq

```
[root@master ~]# cat pod-definition1.yml
---
apiVersion: v1
kind: Pod
metadata:
 name: nginx-pod
 labels:
  author: arun
  version : v1
spec:
 containers:
  - name: appserver
    image: nginx
[root@master ~]#
```

*to get the list of pods

kubectl get pods (no pod is running)

```
[root@master ~]# kubectl get pods
No resources found in default namespace.
[root@master ~]#
```

* Now run the definition file

kubectl create -f pod-definition1.yml (Pod is created)

```
[root@master ~]# kubectl create -f pod-definition1.yml
pod/nginx-pod created
```

* Now get the list of pods

kubectl get pods

```
[root@master ~]# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          51s
[root@master ~]#
```

7. To get the list of pods on which node the pod is running

kubectl get pods -o wide

```
NAME        READY   STATUS    RESTARTS   AGE     IP              NODE     NOMINATED NODE   READINESS GATES
nginx-pod   1/1     Running   0          2m20s   192.168.171.69  worker   <none>           <none>
```

8. To Describe the pod

kubectl describe pods nginx-pod

All the informations about the Pod is available

```
[root@master ~]# kubectl describe pods nginx-pod
Name:             nginx-pod
Namespace:        default
Priority:         0
Service Account:  default
Node:             worker/172.31.28.216
Start Time:       Wed, 12 Oct 2022 06:52:55 +0000
Labels:           author=arun
                  version=v1
Annotations:      cni.projectcalico.org/containerID: 85757fc29211c23ad6f3fa27b2f9e848c1aa50d68f3a8a28b1d7184071bc8852
                  cni.projectcalico.org/podIP: 192.168.171.69/32
                  cni.projectcalico.org/podIPs: 192.168.171.69/32
Status:           Running
IP:               192.168.171.69
IPs:
  IP:  192.168.171.69
Containers:
  appserver:
    Container ID:   containerd://e86aeb0a1882d97b1ef4f7b51ce4945d0f09d3aadb53e1a03ed806012c92e004
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:2f770d2fe27bc85f68fd7fe6a63900ef7076bc703022fe81b980377fe3d27b70
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Wed, 12 Oct 2022 06:52:56 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
```

9. To acces the containers in the pod, enter the following command

kubectl exec -it podname -c containername bash
kubectl exec -it nginx-pod -c appserver bash

```
[root@master ~]# kubectl exec -it nginx-pod -c appserver bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@nginx-pod:/#
```

enter exit
to get out of the container.

```
root@nginx-pod:/# exit
exit
[root@master ~]#
```

10. Delete the pod you just created

kubectl delete po nginx

```
[root@master ~]# kubectl delete po nginx
pod "nginx" deleted
```

11. Create the nginx pod with version 1.17.4 and expose it on port 80

kubectl run nginx --image=nginx:1.17.4 --restart=Never --port=80

```
[root@master ~]# kubectl run nginx --image=nginx:1.17.4 --restart=Never --port=80
pod/nginx created
[root@master ~]# kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP               NODE     NOMINATED NODE   READINESS GATES
nginx   1/1     Running   0          32s   192.168.171.69   worker   <none>           <none>
```

12. Change the Image version to 1.15-alpine for the pod you just created and verify the image version is updated

kubectl set image pod/nginx nginx=nginx:1.15-alpine
kubectl describe po nginx
// another way it will open vi editor and change the version
kubeclt edit po nginx
kubectl describe po nginx

```
[root@master ~]# kubectl set image pod/nginx nginx=nginx:1.15-alpine
pod/nginx image updated
[root@master ~]# kubectl describe po nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             worker/172.31.25.255
Start Time:       Wed, 26 Oct 2022 08:59:32 +0000
Labels:           run=nginx
Annotations:      cni.projectcalico.org/containerID: d8773e8eddae0b8d1c7f470f206d30952bc8c7b87de5f44c6cfdc06481ec35d4
                  cni.projectcalico.org/podIP: 192.168.171.69/32
                  cni.projectcalico.org/podIPs: 192.168.171.69/32
Status:           Running
IP:               192.168.171.69
IPs:
  IP:  192.168.171.69
Containers:
  nginx:
    Container ID:   containerd://e9850d04ba35ad46bbec38c0591462456baaab10225f5df7287c336b45cb0b2d
    Image:          nginx:1.15-alpine
```

13. Delete the pod you just created without any delay (force delete)

kubectl delete po nginx --grace-period=0 --force

```
[root@master ~]# kubectl delete po nginx --grace-period=0 --force
Warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource may continue to run on the cl
ter indefinitely.
pod "nginx" force deleted
```

14. Create the nginx pod and execute the simple shell on the pod

// creating a pod
kubectl run nginx --image=nginx --restart=Never
// exec into the pod
kubectl exec -it nginx /bin/sh

```
[root@master ~]# kubectl run nginx --image=nginx --restart=Never
pod/nginx created
[root@master ~]# kubectl exec -it nginx /bin/sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
```

15. Get the IP Address of the pod you just created

kubectl get po nginx -o wide

```
[root@master ~]# kubectl get po nginx -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP               NODE     NOMINATED NODE   READINESS GATES
nginx   1/1     Running   0          66s   192.168.171.70   worker   <none>           <none>
```

16. Create a busybox pod and run command ls while creating it and check the logs

kubectl run busybox --image=busybox --restart=Never -- ls
kubectl logs busybox

```
[root@master ~]# kubectl run busybox --image=busybox --restart=Never -- ls
pod/busybox created
[root@master ~]# kubectl logs busybox
bin
dev
etc
home
proc
root
sys
tmp
usr
var
```

17. Create a busybox pod with command sleep 3600

kubectl run busybox2 --image=busybox --restart=Never -- /bin/sh -c "sleep 3600"

```
[root@master ~]# kubectl run busybox2 --image=busybox --restart=Never -- /bin/sh -c "sleep 3600"
pod/busybox2 created
```

18. Create a busybox pod and echo message 'How are you' and delete it manually

kubectl run busybox3 --image=nginx --restart=Never -it -- echo "How are you"
kubectl delete po busybox3

```
[root@master ~]# kubectl run busybox3 --image=nginx --restart=Never -it -- echo "How are you"
How are you
```

19. Create a busybox pod and echo message 'How are you' and have it deleted immediately

// notice the --rm flag
kubectl run busybox --image=nginx --restart=Never -it --rm -- echo "How are you"

```
[root@master ~]# kubectl run busybox2 --image=nginx --restart=Never -it --rm -- echo "How are you"
How are you
pod "busybox2" deleted
```

20. Create 5 nginx pods in which two of them is labeled env=prod and three of them is labeled env=dev

kubectl run nginx-dev1 --image=nginx --restart=Never --labels=env=dev
kubectl run nginx-dev2 --image=nginx --restart=Never --labels=env=dev

kubectl run nginx-dev3 --image=nginx --restart=Never --labels=env=dev
kubectl run nginx-prod1 --image=nginx --restart=Never --labels=env=prod
kubectl run nginx-prod2 --image=nginx --restart=Never --labels=env=prod

```
[root@master ~]# kubectl run nginx-dev1 --image=nginx --restart=Never --labels=env=dev
pod/nginx-dev1 created
[root@master ~]# kubectl run nginx-dev2 --image=nginx --restart=Never --labels=env=dev
pod/nginx-dev2 created
[root@master ~]# kubectl run nginx-dev3 --image=nginx --restart=Never --labels=env=dev
pod/nginx-dev3 created
[root@master ~]# kubectl run nginx-prod1 --image=nginx --restart=Never --labels=env=prod
pod/nginx-prod1 created
[root@master ~]# kubectl run nginx-prod2 --image=nginx --restart=Never --labels=env=prod
pod/nginx-prod2 created
```

21. Verify all the pods are created with correct labels

kubectl get pods --show-labels

```
[root@master ~]# kubectl get pods --show-labels
NAME           READY   STATUS      RESTARTS   AGE    LABELS
busybox        0/1     Completed   0          10m    run=busybox
nginx          1/1     Running     0          12m    run=nginx
nginx-dev1     1/1     Running     0          118s   env=dev
nginx-dev2     1/1     Running     0          118s   env=dev
nginx-dev3     1/1     Running     0          118s   env=dev
nginx-prod1    1/1     Running     0          118s   env=prod
nginx-prod2    1/1     Running     0          116s   env=prod
```

22. Get the pods with label env=dev

kubectl get pods -l env=dev

```
[root@master ~]# kubectl get pods -l env=dev
NAME          READY   STATUS    RESTARTS   AGE
nginx-dev1    1/1     Running   0          2m45s
nginx-dev2    1/1     Running   0          2m45s
nginx-dev3    1/1     Running   0          2m45s
```

23. Get the pods with label env=prod and also output the labels

kubectl get pods -l env=prod --show-labels

```
[root@master ~]# kubectl get pods -l env=prod --show-labels
NAME          READY   STATUS    RESTARTS   AGE     LABELS
nginx-prod1   1/1     Running   0          3m37s   env=prod
nginx-prod2   1/1     Running   0          3m35s   env=prod
```

24. Get the pods with labels env=dev and env=prod

kubectl get pods -l 'env in (dev,prod)'

```
[root@master ~]# kubectl get pods -l 'env in (dev,prod)'
NAME          READY   STATUS    RESTARTS   AGE
nginx-dev1    1/1     Running   0          4m30s
nginx-dev2    1/1     Running   0          4m30s
nginx-dev3    1/1     Running   0          4m30s
nginx-prod1   1/1     Running   0          4m30s
nginx-prod2   1/1     Running   0          4m28s
```