Lab 7 - Kubernetes

Initialize cluster

1. To free up resources, stop all vagrant VMs in devop-lab environment.

   ○ Clone the following Git repository :

   https://github.com/brahimhamdi/k8s-lab

   ○ In k8s-lab directory, execute following command to deploy k8s vagrants VMs :

   vagrant up

2. Kubernetes is already installed on all vagrant VMs. On master VM, initialize the cluster.

sudo kubeadm init --apiserver-advertise-address=192.168.56.10 --pod-network-cidr=10.244.0.0/16

```
vagrant@kube-control-plane:~$ sudo kubeadm init --apiserver-advertise-address=192.168.56.10 --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.27.4
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
```

   ○ If no errors, what's the output of the initializing command ?

```
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.56.10:6443 --token d8qczx.mki4o33z0k11i0ih \
        --discovery-token-ca-cert-hash sha256:15018c12c56b176cd793c7e16d6f2df85372427409f4d35f218b4c118cef8149
vagrant@kube-control-plane:~$
```

○ To start using your cluster as regular user, apply next commands

```
vagrant@k8s-master:~$ mkdir -p $HOME/.kube
vagrant@k8s-master:~$    sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
vagrant@k8s-master:~$    sudo chown $(id -u):$(id -g) $HOME/.kube/config
vagrant@k8s-master:~$
```

○ Apply flannel yaml file.

kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel.yml

```
vagrant@kube-control-plane:~$ kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
vagrant@kube-control-plane:~$ _
```

```
vagrant@kube-control-plane:~$ kubectl edit daemonset.apps/kube-flannel-ds -n kube-flannel
daemonset.apps/kube-flannel-ds edited
vagrant@kube-control-plane:~$
```

```
    k8s-app: flannel
    tier: node
  name: kube-flannel-ds
  namespace: kube-flannel
  resourceVersion: "3129"
  uid: b12f9b49-c341-47f8-8d8f-d44bedb7830e
spec:
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: flannel
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: flannel
        tier: node
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/os
                operator: In
                values:
                - linux
      containers:
      - args:
        - --ip-masq
        - --kube-subnet-mgr
        - --iface=eth1
        command:
        - /opt/bin/flanneld
        env:
```

3. Check the cluster info.

```
vagrant@kube-control-plane:~$ kubectl cluster-info
Kubernetes control plane is running at https://192.168.56.10:6443
CoreDNS is running at https://192.168.56.10:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
vagrant@kube-control-plane:~$
```

○ How kubernetes components looks like ?

```
vagrant@k8s-master:~$ kubectl get namespaces
NAME                STATUS   AGE
default             Active   13m
kube-flannel        Active   6m16s
kube-node-lease     Active   13m
kube-public         Active   13m
kube-system         Active   13m
vagrant@k8s-master:~$ kubectl get pod -n kube-system
NAME                                        READY   STATUS    RESTARTS   AGE
coredns-5d78c9869d-brhv4                    0/1     Pending   0          13m
coredns-5d78c9869d-nnhh8                    0/1     Pending   0          13m
etcd-k8s-master                             1/1     Running   10         13m
kube-apiserver-k8s-master                   1/1     Running   19         13m
kube-controller-manager-k8s-master          1/1     Running   20         13m
kube-proxy-54pxj                            1/1     Running   0          13m
kube-scheduler-k8s-master                   1/1     Running   20         13m
vagrant@k8s-master:~$
```

○ What is the IP address of DNS systems ?

```
vagrant@kube-control-plane:~$ kubectl get pod -n kube-system -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP             NODE                 NOMINATED NODE
S GATES
coredns-5d78c9869d-5gxxx      1/1     Running   0          19m   10.244.0.3     kube-control-plane   <none>
coredns-5d78c9869d-lfdrf      1/1     Running   0          19m   10.244.0.2     kube-control-plane   <none>
etcd-kube-control-plane       1/1     Running   0          20m   192.168.56.10  kube-control-plane   <none>
```

4. Join all nodes to the cluster.

○ On the master, check that all nodes are ready on the cluster.

sudo kubeadm token create --print-join-command

```
vagrant@kube-control-plane:~$ sudo kubeadm token create --print-join-command
kubeadm join 192.168.56.10:6443 --token 0n48cr.x0tzlke0pt4un26e --discovery-token-ca-cert-hash sha256:15018c12c56b176cd793c7e16d6f2df853724274
09f4d35f218b4c118cef8149
vagrant@kube-control-plane:~$
```

```
vagrant@kube-control-plane:~$ exit
logout
brahim@Training:~/k8s-lab$ vagrant ssh kube-node2
vagrant@kube-node2:~$ sudo kubeadm join 192.168.56.10:6443 --token 0n48cr.x0tzlke0pt4un26e --discovery-token-ca-cert-hash sha256:15018c12c56b1
76cd793c7e16d6f2df85372427409f4d35f218b4c118cef8149
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

vagrant@kube-node2:~$ exit
logout
brahim@Training:~/k8s-lab$ vagrant ssh kube-control-plane
Last login: Tue Aug  8 12:42:42 2023 from 10.0.2.2
vagrant@kube-control-plane:~$ kubectl get node -o wide
NAME               STATUS     ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION     CONTAI
NER-RUNTIME
kube-control-plane  Ready     control-plane   29m     v1.27.4   192.168.56.10   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   contai
nerd://1.6.22
kube-node1          Ready     <none>          2m6s    v1.27.4   192.168.56.11   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   contai
nerd://1.6.22
kube-node2          NotReady  <none>          13s     v1.27.4   192.168.56.12   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   contai
nerd://1.6.22
vagrant@kube-control-plane:~$
vagrant@kube-control-plane:~$
```

```
vagrant@kube-control-plane:~$ sudo kubeadm token create --print-join-command
kubeadm join 192.168.56.10:6443 --token 0n48cr.x0tzlke0pt4un26e --discovery-token-ca-cert-hash sha256:15018c12c56b176cd793c7e16d6f2df853724274
09f4d35f218b4c118cef8149
vagrant@kube-control-plane:~$
vagrant@kube-control-plane:~$ exit
logout
brahim@Training:~/k8s-lab$ vagrant ssh kube-node1
vagrant@kube-node1:~$ sudo kubeadm join 192.168.56.10:6443 --token 0n48cr.x0tzlke0pt4un26e --discovery-token-ca-cert-hash sha256:15018c12c56b1
76cd793c7e16d6f2df85372427409f4d35f218b4c118cef8149
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

vagrant@kube-node1:~$
vagrant@kube-node1:~$ exit
logout
brahim@Training:~/k8s-lab$ vagrant ssh kube-control-plane
Last login: Tue Aug  8 12:09:11 2023 from 10.0.2.2
vagrant@kube-control-plane:~$ kubectl get node -o wide
NAME               STATUS     ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION     CONTAIN
ER-RUNTIME
kube-control-plane  Ready     control-plane   27m     v1.27.4   192.168.56.10   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   contain
erd://1.6.22
kube-node1          NotReady  <none>          48s     v1.27.4   192.168.56.11   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   contain
erd://1.6.22
vagrant@kube-control-plane:~$
```

Manage pods

5. Create a yaml file for a *hasher* pod.

```
vagrant@k8s-master:~$ vim hasher.yaml
vagrant@k8s-master:~$ cat hasher.yaml
apiVersion: v1
kind: Pod
metadata:
  name: hasher
spec:
  containers:
  - name: hasher
    image: brahimhamdi/hasher
vagrant@k8s-master:~$ kubectl  apply -f hasher.yaml
pod/hasher created
vagrant@k8s-master:~$
vagrant@k8s-master:~$ kubectl get pod -o wide
NAME     READY   STATUS            RESTARTS   AGE   IP       NODE         NOMINATED NODE   READINESS GATES
hasher   0/1     ContainerCreating  0          6s    <none>   k8s-worker2  <none>           <none>
vagrant@k8s-master:~$
```

6. Apply the yaml file.

  ○ On which node the pod is created ?

  ○ What is the pod's IP address ?

  ○ What is the container's name and ID ?

  ○ What is the image's name and ID ?

```
vagrant@k8s-master:~$ kubectl describe pod/hasher
Name:            hasher
Namespace:       default
Priority:        0
Service Account: default
Node:            k8s-worker2/192.168.205.102
Start Time:      Tue, 13 Jun 2023 13:36:56 +0000
Labels:          <none>
Annotations:     <none>
Status:          Running
IP:              10.244.2.63
IPs:
  IP:  10.244.2.63
Containers:
  hasher:
    Container ID:   containerd://755099de8a6f3fd62d23b228d212c78f755a0ecb6a867c755776fba923f84c87
    Image:          brahimhamdi/hasher
    Image ID:       docker.io/brahimhamdi/hasher@sha256:a37377f07840109415eb7df07ae830bc617d0f3ac3c98c904b7a8647868785f5
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Tue, 13 Jun 2023 13:38:38 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-9fkdl (ro)
Conditions:
  Type            Status
  Initialized     True
  Ready           True
  ContainersReady True
  PodScheduled    True
Volumes:
```

7. Remove the pod from the cluster.

```
vagrant@k8s-master:~$ kubectl delete pod hasher
pod "hasher" deleted
vagrant@k8s-master:~$ kubectl get pod
No resources found in default namespace.
vagrant@k8s-master:~$
```

Manage deployments and services

8. Create yaml file to describe *dockercoins* application deployment.

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: dockercoins
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: worker
  namespace: dockercoins
spec:
# replicas: 3
  selector:
    matchLabels:
      app: dockercoins
      tier: worker
  template:
    metadata:
      labels:
        app: dockercoins
        tier: worker
    spec:
      containers:
      - name: worker
        image: brahimhamdi/worker
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rng
  namespace: dockercoins
spec:
# replicas: 3
  selector:
    matchLabels:
      app: dockercoins
      tier: rng
  template:
    metadata:
      labels:
        app: dockercoins
        tier: rng
    spec:
      containers:
      - name: rng
        image: brahimhamdi/rng
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hasher
  namespace: dockercoins
spec:
```

```yaml
# replicas: 3
  selector:
    matchLabels:
      app: dockercoins
      tier: hasher
  template:
    metadata:
      labels:
        app: dockercoins
        tier: hasher
    spec:
      containers:
      - name: hasher
        image: brahimhamdi/hasher
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
  namespace: dockercoins
spec:
# replicas: 3
  selector:
    matchLabels:
      app: dockercoins
      tier: redis
  template:
    metadata:
      labels:
        app: dockercoins
        tier: redis
    spec:
      containers:
      - name: redis
        image: redis
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webui
  namespace: dockercoins
spec:
# replicas: 3
  selector:
    matchLabels:
      app: dockercoins
      tier: webui
  template:
    metadata:
      labels:
        app: dockercoins
        tier: webui
    spec:
      containers:
      - name: webui
        image: brahimhamdi/webui
---
apiVersion: v1
```

```yaml
kind: Service
metadata:
  name: rng
  namespace: dockercoins
spec:
  selector:
    app: dockercoins
    tier: rng
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  name: hasher
  namespace: dockercoins
spec:
  selector:
    app: dockercoins
    tier: hasher
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  name: redis
  namespace: dockercoins
spec:
  selector:
    app: dockercoins
    tier: redis
  ports:
    - protocol: TCP
      port: 6379
      targetPort: 6379
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  name: webui
  namespace: dockercoins
spec:
  selector:
    app: dockercoins
    tier: webui
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30001
```

            Brahim HAMDI

type: NodePort

9. Apply the yaml file and check the application.

```
vagrant@k8s-master:~$ kubectl apply -f dockercoins.yaml
namespace/dockercoins created
deployment.apps/worker created
deployment.apps/rng created
deployment.apps/hasher created
deployment.apps/redis created
deployment.apps/webui created
service/rng created
service/hasher created
service/redis created
service/webui created
vagrant@k8s-master:~$
```

```
vagrant@k8s-master:~$ kubectl get all -n dockercoins
NAME                          READY   STATUS    RESTARTS   AGE
pod/hasher-7f9d944db9-d2xbl   1/1     Running   0          3m24s
pod/redis-78579d7b98-l4sp2    1/1     Running   0          3m24s
pod/rng-544477487c-c8dn8      1/1     Running   0          3m24s
pod/webui-c9697458-8857v      1/1     Running   0          3m24s
pod/worker-5f7877988-frxhg    1/1     Running   0          3m24s

NAME              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service/hasher    ClusterIP   10.104.236.71    <none>        80/TCP         3m23s
service/redis     ClusterIP   10.100.243.140   <none>        6379/TCP       3m23s
service/rng       ClusterIP   10.111.114.165   <none>        80/TCP         3m23s
service/webui     NodePort    10.108.202.27    <none>        80:30001/TCP   3m23s

NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hasher       1/1     1            1           3m24s
deployment.apps/redis        1/1     1            1           3m24s
deployment.apps/rng          1/1     1            1           3m24s
deployment.apps/webui        1/1     1            1           3m24s
deployment.apps/worker       1/1     1            1           3m24s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/hasher-7f9d944db9       1         1         1       3m24s
replicaset.apps/redis-78579d7b98        1         1         1       3m24s
replicaset.apps/rng-544477487c          1         1         1       3m24s
replicaset.apps/webui-c9697458          1         1         1       3m24s
replicaset.apps/worker-5f7877988        1         1         1       3m24s
vagrant@k8s-master:~$
```

```
vagrant@k8s-master:~$ vim dockercoins.yaml
vagrant@k8s-master:~$ kubectl apply -f dockercoins.yaml
namespace/dockercoins unchanged
deployment.apps/worker configured
deployment.apps/rng unchanged
deployment.apps/hasher unchanged
deployment.apps/redis unchanged
deployment.apps/webui unchanged
service/rng unchanged
service/hasher unchanged
service/redis unchanged
service/webui unchanged
vagrant@k8s-master:~$
```

DockerCoin Miner WebUI
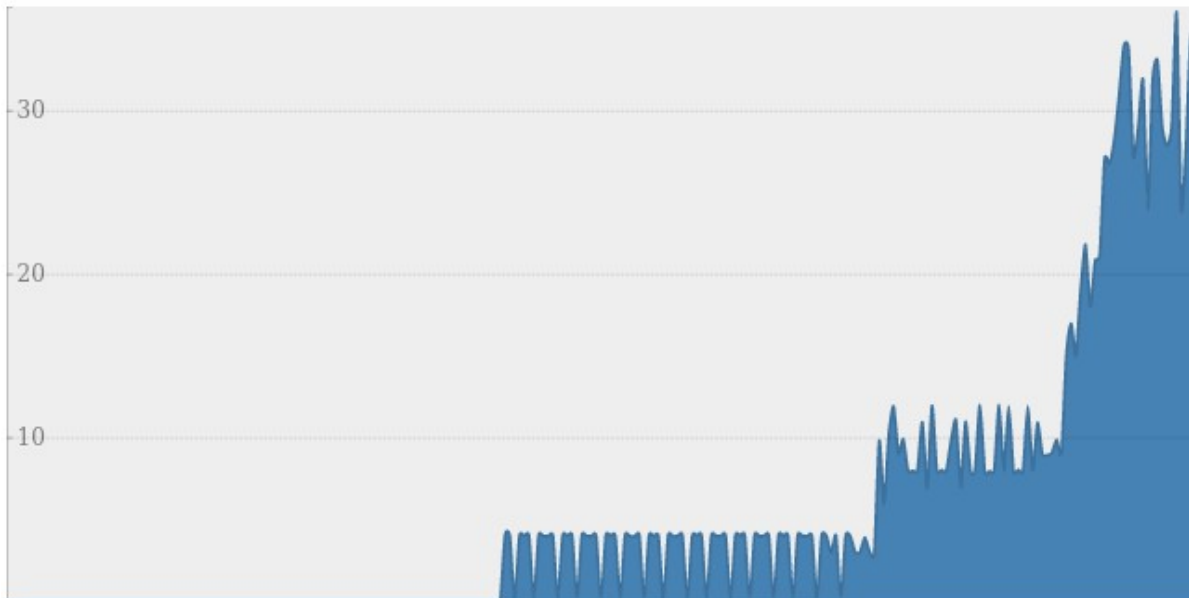
Current mining speed: ~28.1 hashes/second (Tweet this!)