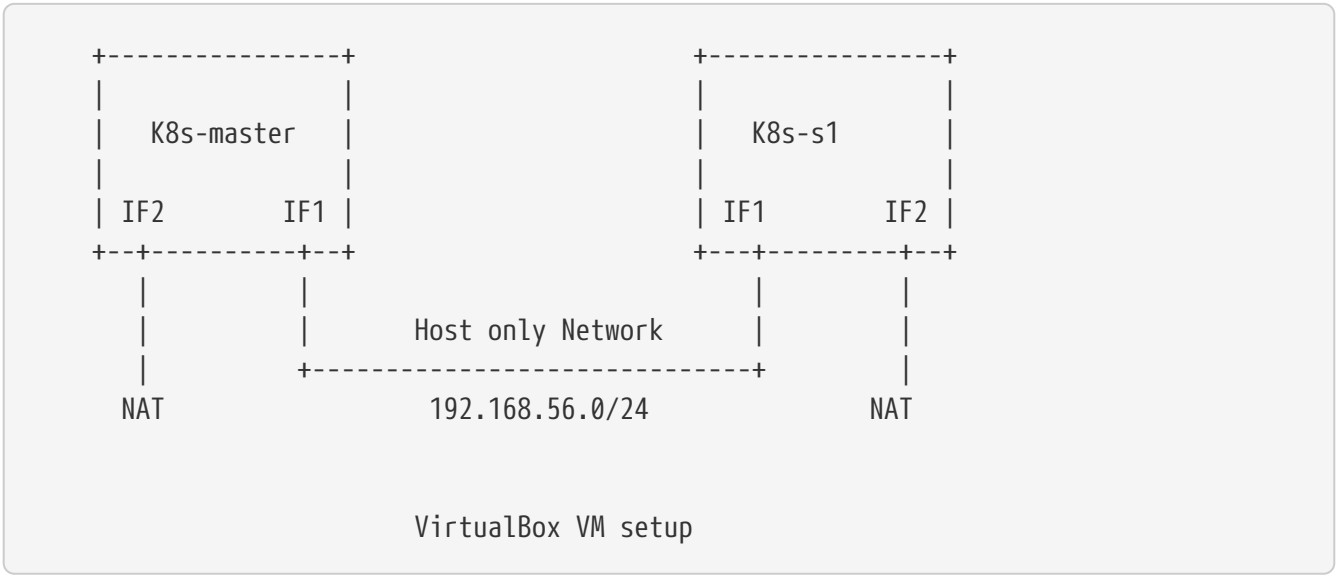
Kubernetes MultiNode cluster on VirtualBox

Table of Contents

A step by step description of bringing up kubernetes cluster using VirtualBox 6.0 with ubuntu 18.04 server.

VirtualBox setup

Create two VMs k8s-master and k8s-s1 as shown in below diagram.



VirtualBox VM setup

VM hardware configuration.

- Kubernetes master/controller node hardware requirement

Component	Value (min)
Processor	2 core
RAM	4GB
Network Adapter-1	Host only network
Network Adapter-2	NAT

- kubernetes worker/slave node hardware requirement

Component	Value (min)
Processor	1 core
RAM	2GB
Network Adapter-1	Host only network
Network Adapter-2	NAT

while creating disk partition do not select partition with LVM.

Prepare Servers:

Execute the below step in both the VMs.

Setup network.

- Shutdown the VMs.
- Modify VM network, with Adapter 1 for Host Only Network and Adapter 2 for NAT.
- Start VM.
- Execute below command and note down the interfaces (generally for Virtual box VMs network interface will start with enp0s).

```
$ sudo ifconfig -a
```

- Setup static network. Open file /etc/netplan/50-cloud-init.yaml and add below configuration.

  On k8s-master

```
network:
 ethernets:
  enp0s3:
   addresses: [192.168.56.10/24]
   dhcp4: false
  enp0s8:
   addresses: []
   dhcp4: true
 version: 2
```

  On k8s-s1

```
network:
 ethernets:
  enp0s3:
   addresses: [192.168.56.12/24]
   dhcp4: false
  enp0s8:
   addresses: []
   dhcp4: true
 version: 2
```

- Reboot VMs.
- Update Packages if any
- Update apt sources.

```
$ sudo apt update
```

- Install packages

```
$ sudo apt upgrade
```

Install docker

- Install utility packages

```
$ sudo apt install apt-transport-https ca-certificates curl software-properties-
common
```

- Add GPG key

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

- Add docker repository to apt sources.

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu bionic stable"
```

- Update apt sources and install docker

```
$ sudo apt update ; sudo apt install docker-ce
```

- Verify docker service is started.

```
$ sudo systemctl status docker
```

Install and configure kubernetes.

General Installation

Execute the below steps on both the VMs.

1. Add Kubernetes source gpg.

```
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add
```

2. Next add kubernetes repository

```
$ sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
```

Browese http://apt.kubernetes.io/ and search for the corresponding ubuntu version, since there was no kubernetes-bionic, installing kubernetes-xenial in my case.

3. Install `kubeadm`

```
$ sudo apt install kubeadm
```

4. Turn off swap

```
$ sudo swapoff -a
```

5. Commanent out any line containing swap in `/etc/fstab`

If 4 and 5 steps not followed, kubelet service will not start.

6. Reboot VMs.

Configure Master VM.

1. Initialize kubernetes master node by executing below command.

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

2. As output of above command shows to execute below command to enable kubectl command. Execute the below command.

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

3. Note down (copy to textpad) the kubeadm join command which needs to be executed on slave nodes to join to the cluster, we not run the command now, instead we will execute once dashboard is installed as we need dashboard to be installed on master node.
4. Install pod network.

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

5. Verify all necessary pods are started

```
$ kubectl get pods --all-namespaces
```

output:

```
NAMESPACE     NAME                                 READY   STATUS    RESTARTS   AGE
kube-system   coredns-86c58d9df4-c68gd             1/1     Running   0
6m41s
kube-system   coredns-86c58d9df4-q5bht             1/1     Running   0
6m41s
kube-system   etcd-k8s-master                      1/1     Running   0
6m6s
kube-system   kube-apiserver-k8s-master            1/1     Running   0
5m59s
kube-system   kube-controller-manager-k8s-master   1/1     Running   0
5m56s
kube-system   kube-flannel-ds-amd64-stb29          1/1     Running   0          49s
kube-system   kube-proxy-882ms                     1/1     Running   0
6m41s
kube-system   kube-scheduler-k8s-master            1/1     Running   0
5m54s
```

Configure Kubernetes Dashboad.

1. Deploy dashboard.

```
$ kubectl create -f
https://raw.githubusercontent.com/kubernetes/dashboard/master/aio/deploy/recommende
d/kubernetes-dashboard.yaml
```

2. Wait till dashboard pod is running.

```
$ kubectl get pods --all-namespaces
```

output:

```
NAMESPACE     NAME                                    READY   STATUS    RESTARTS
AGE
kube-system   coredns-86c58d9df4-c68gd                1/1     Running   0
11m
kube-system   coredns-86c58d9df4-q5bht                1/1     Running   0
11m
kube-system   etcd-k8s-master                         1/1     Running   0
10m
kube-system   kube-apiserver-k8s-master               1/1     Running   0
10m
kube-system   kube-controller-manager-k8s-master      1/1     Running   0
10m
kube-system   kube-flannel-ds-amd64-stb29             1/1     Running   0
5m18s
kube-system   kube-proxy-882ms                        1/1     Running   0
11m
kube-system   kube-scheduler-k8s-master               1/1     Running   0
10m
kube-system   kubernetes-dashboard-57df4db6b-5phx2    1/1     Running   0
35s
```

3. By default dashboard cannot be accessed from outside the VM, if you are using ubuntu desktop you can run below command and access the dashboard using proxy at url http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/

```
$ kubectl proxy
```

   1. However if you want dashboard be accessed from external ip editing kubernetes-dashboard service and changing type from ClusterIP to NodePort.

```
$ kubectl edit service kubernetes-dashboard -n kube-system
```

      The file content should something as shown below

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2019-01-21T18:06:35Z"
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system
  resourceVersion: "1885"
  selfLink: /api/v1/namespaces/kube-system/services/kubernetes-dashboard
  uid: 4a2d8f61-1da7-11e9-9d52-080027aba7cb
spec:
  clusterIP: 10.110.253.116
  externalTrafficPolicy: Cluster
  ports:
  - nodePort: 32608
    port: 443
    protocol: TCP
    targetPort: 8443
  selector:
    k8s-app: kubernetes-dashboard
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

4. Execute the below command and note down the port

```
$ kubectl get service --all-namespaces
```

output:

```
NAMESPACE     NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)         AGE
default       kubernetes             ClusterIP   10.96.0.1        <none>
443/TCP         21m
kube-system   kube-dns               ClusterIP   10.96.0.10       <none>
53/UDP,53/TCP   21m
kube-system   kubernetes-dashboard   NodePort    10.110.253.116   <none>
443:32608/TCP   10m
```

The dashboard port is **32608** in my case.

5. Now we can access dashboard at URL. https://192.168.56.10:32608

Port will be dynamically generated and port should be replaced from step 5.

Create service Account and access dashboard.

1. Create a service account

   ```
   $ kubectl create serviceaccount admin-user -n kube-system
   ```

   Verification : Below command should list the admin-user account

   ```
   $ kubectl get serviceaccount --all-namespaces
   ```

2. Create Cluster Role binding for the user.

   ```
   $  kubectl create clusterrolebinding admin-user -n kube-system \
        --clusterrole=cluster-admin \
        --serviceaccount=kube-system:admin-user
   ```

3. Generate the Bearer Token to access Dashboard

   ```
   $ kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep
   admin-user | awk '{print $1}')
   ```

output:

```
Name:         admin-user-token-4nwz2
Namespace:    kube-system
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: admin-user
              kubernetes.io/service-account.uid: a1e3ca50-1dab-11e9-9d52-
080027aba7cb

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1025 bytes
namespace:  11 bytes
token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3
ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0Z
XMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2VyLXRva2VuLTRud3oyIiwia3Vi
ZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImFkbWluLXVzZXIiLCJ
rdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiJhMWUzY2E1MC0xZG
FiLTExZTktOWQ1Mi0wODAwMjdhYmE3Y2IiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1ze
XN0ZW06YWRtaW4tdXNlciJ9.YHRkrY1dPsrf1N4LU6qGqCPPl617faeBbHelJAdWXD3TvvZMYnQdMvZuWtF
ZjVMxXPdgXDud17eCffDXBg5bRAs1sxd7B37IbXVULrYFoMR-
B0MjOa3eLx1edO_gvE6ZqpyPpdWxC0hWYI0P9cQ78oyZEZ0RDNctTus0qRpVrHpP5ZIMhfRPknV8zxxF-
zGf8Xg8ni1NxUOHHB-
DYO1T6gd4v65JgD2ohLS4N9rLpq_MrA7nc13R4sE6zDIgYi5V7kZYz0Zx72qAaV4oOGMDTr0FPP7q3m9SrH
8uO3UOUe9tkp_ce8-7V9hJW8AbPHu3rLNBw2dOGnOk59yNe3jv5w
```

Copy the token and paste it into token feild in the URL to Dashboard and login to dashboard.

Configuration and joining slave node.

1. Now go to slave node and execute the join command previously saved when you were executing kubeadm on master.

```
$  kubeadm join 192.168.56.10:6443 --token t0j1zi.v5lojsnpjh9r0rbn --discovery
-token-ca-cert-hash
sha256:40b1142d9002003ab5b085776b8b8cba4a41ceaafab06429c49eaedc2b2939fa
```

The above command is sample, the values are dynamically generated.

2. Now go back to master and execute the below command, you should be able to see slave node added.

```
$ kubectl get nodes
```

output:

```
NAME         STATUS   ROLES    AGE     VERSION
k8s-master   Ready    master   56m     v1.13.2
k8s-s1       Ready    <none>   2m49s   v1.13.2
```