Kubernetes MultiNode cluster on VirtualBox

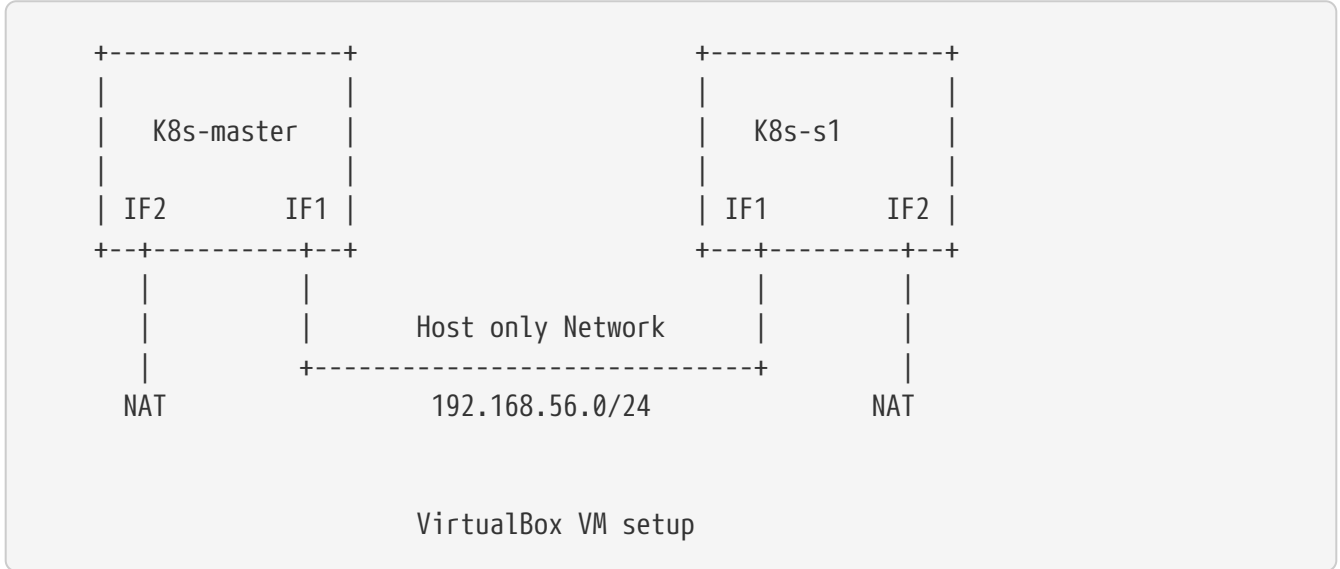
Table of Contents

VirtualBox setup . . . 1
VM hardware configuration. . . . 2
Prepare Servers . . . 2
  Setup network. . . . 2
  Install docker . . . 4
Install and configure kubernetes. . . . 4
  Prepare server . . . 4
  Configure Master VM. . . . 6
  Configure Kubernetes Dashboad. . . . 7
  Configure slave node. . . . 11
  Install Helm . . . 11
  Install Ingress controller . . . 12
Troubleshoot. . . . 12
  References 13

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

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.

k8s-master

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

k8s-s1

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

- Update `/etc/hosts` such that command `hostname -i` outputs ip which is routable from other VM.

  example(master)

  ```
  192.168.56.10 k8s-master
  192.168.56.12 k8s-s1
  ```

  example(slave)

  ```
  192.168.56.10 k8s-master
  192.168.56.12 k8s-s1
  ```

  This step is required since we've chosen host-only adapter and our default route is provider by NAT network, if we choose Bridge network this step can be skipped.

- Reboot VMs.
- Update Packages.
  - 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.

Prepare server

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"
```

Browse 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 update; sudo apt install kubeadm kubelet kubectl
```

4. Turn off swap

```
$ sudo swapoff -a
```

5. Turn off swap permanently in `/etc/fstab` by executing belo command.

```
$ sudo sed -i '/swap/s/^\(.*\)$/#\1/g' /etc/fstab
```

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

6. Add private ip to kubernetes arguments

   While installing `kubeadm` you might get error related to port forwarding if you are using private IP which is true in our case. To resolve this issue, Create file `/etc/default/kubelet` and following content in the file.

```
KUBELET_EXTRA_ARGS=--node-ip=192.168.56.10
```

7. Use systemd for cgroups.

   Create file `/etc/docker/daemon.json` and add following content.

```
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
```

8. Reload configuration and restart docker service.

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

9. Reboot VMs.

Configure Master VM.

1. Initialize kubernetes master node by executing below command.

```
$ kubeadm init --apiserver-advertise-address=192.168.56.9 \
   --apiserver-cert-extra-sans=192.168.56.9  \
   --pod-network-cidr=10.244.0.0/16 -v 1
```

If decide to change pod network remember to change the pod network in `flannel.yaml` while setting up pod network in below step.

2. As output of above command shows to execute below command, run below command to update auth details for `kubectl` 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.
   a. If you are using bridge network which has default route then you can directly install flannel using below command. However if you had given host-only adapter IP address for API server. then skip this step and follow next steps.

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

   b. Download flannel.yaml

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

   c. Open `kube-flannel.yaml` file search for daemonset kind and add additional arugment `--iface=<host-only-adaper-interface>`, note there are multiple daemonset definitions, either update all of them or execute `kubectl get nodes --show-labels` and get the arch and update corresponding arch daemonset only.

```
      containers:
      - name: kube-flannel
        image: quay.io/coreos/flannel:v0.11.0-amd64
        command:
        - /opt/bin/flanneld
        args:
        - --ip-masq
        - --kube-subnet-mgr
        - --iface=enp0s3 ①
        resources:
          requests:
            cpu: "100m"
            memory: "50Mi"
          limits:
```

① Added my host-only adapter here

d. if you had changed pod network while executing `kubeadm init`, replace `10.244.0.0/16` which chosen pod network above in `net-conf.json` section.

e. Now Create pod netwrok by applying updated yaml file

```
$ kubectl apply -f kube-flannel.yaml
```

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.

Install Dashboard

1. Deploy dashboard.

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

   a. 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: ClusterIP ①
status:
  loadBalancer: {}
```

① Replace **ClusterIP** with **NodePort**

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.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2VyLXRva2VudLTRud3oyIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLml
vL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiJhMWUzY2E1MC0xZGFiLTExZTktOWQ1Mi0wODAwMjdhYmE3Y2IiLCJzdWIiOiJzeXN
0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.YHRkrY1dPsrf1N4LU6qGqCPPl617faeBbHelJAdWXD3TvvZMYnQdMvZuWt
FZjVMxXPdgXDud17eCffDXBg5bRAs1sxd7B37IbXVULrYFoMR-
B0MjOa3eLx1edO_gvE6ZqpyPpdWxC0hWYI0P9cQ78oyZEZ0RDNctTus0qRpVrHpP5ZIMhfRPknV8zxxF-zGf8Xg8ni1NxUOHHB-
DYO1T6gd4v65JgD2ohLS4N9rLpq_MrA7nc13R4sE6zDIgYi5V7kZYz0Zx72qAaV4oOGMDTr0FPP7q3m9SrH8uO3UOUe9tkp_ce8-
7V9hJW8AbPHu3rLNBw2dOGnOk59yNe3jv5w
```

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

Configuration 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
```

Install Helm

Helm package manager consists of helm client and tiller component which runs as pod on k8s. We will install helm client on our master server itself.

Install helm client.

1. Install helm client using snap on master server using below command.

```
$ sudo snap install helm --classic
```

2. Verify helm client is running by executing command `helm version`, it should give version details for client and for server side should return error until we install tiller.

Install Tiller.

Before installing tiller we need to create RBAC for the tiller pod.

1. Create service account for tiller.

```
$ kubectl create serviceaccount -n kube-system tiller
```

2. Provide cluster role for the above service account.

```
$ kubectl create clusterrolebinding tiller-cluster-rule \
   --clusterrole cluster-admin \
   --serviceaccount=kube-system:tiller
```

3. Now create tiller using the service account

```
helm init --service-account tiller
```

4. Wait until tiller pod comes up.

```
kubectl get po --all-namespaces --watch
```

5. Once tiller pod is up, run `helm version`, the output provide version details for both client and server.

Install Ingress controller

Optionally if you want to expose Any API/web application install ingress controllers such as NGINX, traefik etc.

Install Traefik Ingress controller.

1. Execute below helm command to install traefik ingress.

```
$ helm install --name treafik-ingress --namespace kube-system  --set
dashboard.enabled=true --set rbac.enabled=true stable/traefik
```

Installing NGINX ingress controller.

1. Execute below helm command to install nginx ingress.

```
$ helm install stable/nginx-ingress --name nginx-ingress --set rbac.create=true
```

Troubleshoot.

https://kubernetes.io/docs/tasks/administer-cluster/dns-debugging-resolution/ - DNS not resolving

https://kubernetes.io/docs/setup/independent/troubleshooting-kubeadm/ - kubeadm

References

https://github.com/helm/charts/tree/master/stable/traefik - traefik

https://github.com/helm/charts/tree/master/stable/nginx-ingress - nginx

https://kubernetes.io/docs/concepts/services-networking/ingress/ - ingress controller concepts