A SIMPLE K3S CLUSTER DEPLOYED ON UBUNTU CORE 20

Ver. 0.1

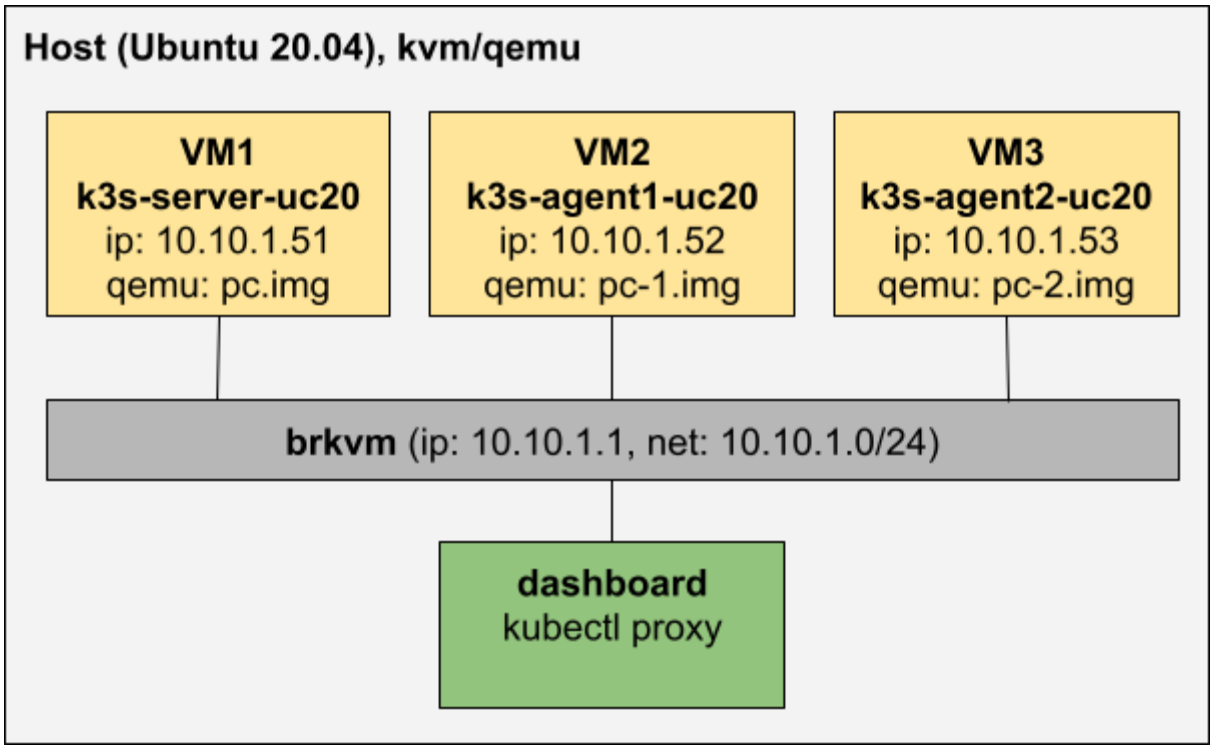
February 2021

Diego Bruno

TABLE OF CONTENT

Introduction

This document describes a simple deployment of k3s using ubuntu core 20 and qemu on amd64 architecture.
As described in the diagram below, the cluster is composed of three virtual machines: one k3s server (VM1), and two k3s agents (VM2 and VM3). The three VMs are hosted on the same Ubuntu 20.04 host.
A bridged network (*brkvm*) is used to give internet access to the VMs and to give connectivity to each others.
The document provides all the steps required to setup such an environment, including host and VMs setup, snap installation and configuration, k8s dashboard configuration.



k3s as a snap

Ubuntu Core imposes that all the software running on an Ubuntu Core system has to be a snap.
For this reason, a k3s snap called *k3s-dbruno* has been developed.
More information about the *k3s-dbruno* can be found here:
https://snapcraft.io/k3s-dbruno
https://github.com/dbruno74/k3s-dbruno

Host setup

Preparation

- Install qemu and test if kvm acceleration can be used:
  ```
  sudo apt install qemu-kvm
  kvm-ok
  ```

- Install the OVMF package
  ```
  sudo apt install ovmf
  ```

- Download uc20 for amd64 from here. Pick ubuntu-core-20-amd64.img.xz.
  ```
  tar xf ubuntu-core-20-amd64.img.xz
  cp ubuntu-core-20-amd64.img pc.img
  cp ubuntu-core-20-amd64.img pc-2.img
  ```

Network setup

1. create the bridge
   ```
   sudo brctl addbr brkvm
   sudo ip addr add 10.10.1.1/24 dev brkvm
   sudo ip link set brkvm up
   sudo mkdir /etc/qemu
   sudo touch /etc/qemu/bridge.conf
   sudo echo "allow brkvm" >> /etc/qemu/bridge.conf
   ```

2. Create the file /etc/qemu/bridge.conf and add the following:
   ```
   allow brkvm
   ```

3. Define NAT rule to allow the VMs to access internet

   `sudo iptables -t nat -A POSTROUTING -s 10.10.1.0/24 -o <your eth adapter accessing internet> -j MASQUERADE`

   NOTE: in case uc20 vm is not able to reach internet, try to disable the firewall on the host

   ```
   sudo ufw disable
   ```

VM1 (k3s server) setup

1. Start qemu

```
sudo qemu-system-x86_64 -smp 2 -m 2048 \
 -net nic,model=virtio,macaddr=52:54:00:12:34:10 \
 -net bridge,br=brkvm \
 -drive
file=/usr/share/OVMF/OVMF_CODE.fd,if=pflash,format=raw,unit=0,readonly=on \
 -drive file=pc.img,cache=none,format=raw,id=disk1,if=none \
 -device virtio-blk-pci,drive=disk1,bootindex=1 -machine accel=kvm
```

2. Configure the network and ssh access
    a. Press enter to configure
    b. Click OK
    c. Select “ens3” interface and press ENTER
    d. Select “edit IPv4” and press ENTER
    e. Press ENTER and select “Manual”
    f. Fill the fields as described here:

```
Subnet: 10.10.1.0/24
Address: 10.10.1.51
Gateway: 10.10.1.1
Name servers: <your name servers>
Search domains: <empty>
```

    g. Click “Save”
    h. Select “Done” and press ENTER
    i. Enter your SSO account email address, then select “Done” and press ENTER

NOTE: you can still set/change network configuration later on in `/etc/netplan/00-snapd-config.yaml` as described below:

```
network:
  ethernets:
    ens3:
      dhcp4: false
      addresses: [10.10.1.51/24]
      gateway4: 10.10.1.1
      nameservers:
        addresses: [<your nameserver>]
  version: 2
```

3. Connect to the vm

```
ssh <your SSO login name>@10.10.1.51
```

4. Set the hostname and reconnect
   ```
   sudo hostnamectl set-hostname 'k3s-server-uc20'
   ssh <your SSO login name>@10.10.1.51
   ```

5. Install k3s snap from the snap store
   a. Install k3s snap
      ```
      snap install k3s-dbruno --edge --devmode
      ```

6. Apply workaround on `/run/user`
   ```
   sudo mkdir -p /run/user/0/snap.k3s
   sudo chmod 700 /run/user/0
   ```

7. Apply cri-container.d apparmor profile
   a. Install wget-simosx snap
      ```
      snap install --beta wget-simosx
      snap connect wget-simosx:home
      ```
   b. Download `cri-containerd-apparmor.d`
      ```
      wget-simosx.wget
      https://raw.githubusercontent.com/dbruno74/k3s/main/cri-containerd.apparmor.d
      ```
   c. Load the profile
      ```
      sudo cp cri-containerd.apparmor.d /var/lib/snapd/apparmor/profiles
      sudo apparmor_parser -r
      /var/lib/snapd/apparmor/profiles/cri-containerd.apparmor.d
      ```

8. Start k3s server, and wait for 2-3 minutes it completely starts

   ```
   snap start k3s-dbruno.k3s-daemon
   ```

9. If you need to get the server log:
   ```
   snap logs k3s-dbruno.k3s-daemon -f
   ```

10. Create a convenient alias for kubectl.
    a. Add to ~/.bashrc the following line
       ```
       alias kubectl='sudo k3s-dbruno.k3s kubectl'
       ```
    b. Logout and login again

11. Connect with another shell, and check the server and pods are running

    ```
    ssh <your SSO login name>@10.10.1.51

    $ kubectl get nodes
    NAME                STATUS   ROLES                  AGE     VERSION
    k3s-server-uc20     Ready    control-plane,master   3m      v1.20.0+k3s2

    $ kubectl get pods --all-namespaces
    NAMESPACE     NAME                                      READY   STATUS
    RESTARTS   AGE
    ```

```
kube-system   metrics-server-86cbb8457f-jgc4v           1/1     Running     0
99m
kube-system   helm-install-traefik-lbnp9                0/1     Completed   0
99m
kube-system   coredns-854c77959c-xcwnr                  1/1     Running     0
99m
kube-system   local-path-provisioner-7c458769fb-nvhhb   1/1     Running     1
99m
kube-system   svclb-traefik-ncnxz                       2/2     Running     0
69s
kube-system   traefik-6f9cbd9bd4-snh5n                  1/1     Running     0
99m
```

12. Fix the note-token symbolic link

```
sudo rm /var/lib/rancher/k3s/server/node-token
sudo ln -s /var/snap/k3s/current/var/lib/rancher/k3s/server/token
/var/snap/k3s/current/var/lib/rancher/k3s/server/node-token
```

13. Get the node-token

```
dbruno74@ubuntu:~$ sudo cat
/var/snap/k3s/current/var/lib/rancher/k3s/server/node-token
K107dc295855d74090b399ec6429c7a846b47f4c25f93115d17bec83106435236a3::server:11
0e2695f46e01c5f12c95978a5e3104
```

    This will be used to start workers

VM2 (k3s worker 1) setup

1. Start qemu

```
sudo qemu-system-x86_64 -smp 2 -m 2048 \
 -net nic,model=virtio,macaddr=52:54:00:12:34:11 \
 -net bridge,br=brkvm \
 -drive
file=/usr/share/OVMF/OVMF_CODE.fd,if=pflash,format=raw,unit=0,readonly=on \
 -drive file=pc-1.img,cache=none,format=raw,id=disk1,if=none \
 -device virtio-blk-pci,drive=disk1,bootindex=1 -machine accel=kvm
```

2. Configure the network and ssh access
    a. Press enter to configure
    b. Click OK
    c. Select “ens3” interface and press ENTER
    d. Select “edit IPv4” and press ENTER
    e. Press ENTER and select “Manual”
    f. Fill the fields as described here:

```
Subnet: 10.10.1.0/24
Address: 10.10.1.52
Gateway: 10.10.1.1
Name servers: <your name servers>
Search domains: <empty>
```

    g. Click “Save”
    h. Select “Done” and press ENTER
    i. Enter your SSO account email address, then select “Done” and press ENTER

NOTE: you can still set/change network configuration later on in `/etc/netplan/00-snapd-config.yaml` as described below:

```
network:
  ethernets:
    ens3:
      dhcp4: false
      addresses: [10.10.1.52/24]
      gateway4: 10.10.1.1
      nameservers:
        addresses: [<your nameservers>]
  version: 2
```

3. Connect to the vm

```
ssh <your SSO login name>@10.10.1.52
```

4. Set the hostname and reconnect
   ```
   sudo hostnamectl set-hostname 'ke3s-agent1-uc20'
   ssh <your SSO login name>@10.10.1.51
   ```

5. Install k3s snap from the snap store
   a. Install k3s snap
      ```
      snap install k3s-dbruno --edge --devmode
      ```

6. Apply workaround on `/run/user`
   ```
   sudo mkdir -p /run/user/0/snap.k3s
   sudo chmod 700 /run/user/0
   ```

7. Apply cri-container.d apparmor profile
   a. Install wget-simosx snap
      ```
      snap install --beta wget-simosx
      snap connect wget-simosx:home
      ```
   b. Download `cri-containerd-apparmor.d`
      ```
      wget-simosx.wget
      https://raw.githubusercontent.com/dbruno74/k3s/main/cri-containerd.appar
      mor.d
      ```
   c. Load the profile
      ```
      sudo cp cri-containerd.apparmor.d /var/lib/snapd/apparmor/profiles
      sudo apparmor_parser -r
      /var/lib/snapd/apparmor/profiles/cri-containerd.apparmor.d
      ```

8. Check connectivity with VM1

   ```
   dbruno74@ubuntu:~$ ping 10.10.1.51
   PING 10.10.1.51 (10.10.1.51) 56(84) bytes of data.
   64 bytes from 10.10.1.51: icmp_seq=1 ttl=64 time=0.723 ms
   64 bytes from 10.10.1.51: icmp_seq=2 ttl=64 time=0.751 ms
   ^C
   ```

9. Start k3s agent
   a. Set k3s command line
      ```
      set k3s-dbruno k3s-cmd-line="agent --server https://10.10.1.51:6443
      --token <node-token printed on VM1 previously>"
      ```

   ```
   snap start k3s-dbruno.k3s-daemon
   ```

10. If you need to get the server log:
    ```
    snap logs k3s-dbruno.k3s-daemon -f
    ```

11. Create a convenient alias for kubectl.
    a. Add to ~/.bashrc the following line
       ```
       alias kubectl='sudo k3s-dbruno.k3s kubectl'
       ```
    b. Logout and login again

12. Check nodes on VM1

```
$ ip a
(...)
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff
    inet 10.10.1.51/24 brd 10.10.1.255 scope global ens3

$ kubectl get nodes
NAME              STATUS   ROLES                  AGE    VERSION
k3s-agent1-uc20   Ready    <none>                 10m    v1.20.0+k3s2
k3s-server-uc20   Ready    control-plane,master   151m   v1.20.0+k3s2
```

VM3 (k3s worker 2) setup

13. Start qemu

```
sudo qemu-system-x86_64 -smp 2 -m 2048 \
 -net nic,model=virtio,macaddr=52:54:00:12:34:12 \
 -net bridge,br=brkvm \
 -drive
file=/usr/share/OVMF/OVMF_CODE.fd,if=pflash,format=raw,unit=0,readonly=on \
 -drive file=pc-2.img,cache=none,format=raw,id=disk1,if=none \
 -device virtio-blk-pci,drive=disk1,bootindex=1 -machine accel=kvm
```

14. Configure the network and ssh access
    a. Press enter to configure
    b. Click OK
    c. Select “ens3” interface and press ENTER
    d. Select “edit IPv4” and press ENTER
    e. Press ENTER and select “Manual”
    f. Fill the fields as described here:

```
Subnet: 10.10.1.0/24
Address: 10.10.1.53
Gateway: 10.10.1.1
Name servers: <your name servers>
Search domains: <empty>
```

    g. Click “Save”
    h. Select “Done” and press ENTER
    i. Enter your SSO account email address, then select “Done” and press ENTER

NOTE: you can still set/change network configuration later on in `/etc/netplan/00-snapd-config.yaml` as described below:

```
network:
  ethernets:
    ens3:
      dhcp4: false
      addresses: [10.10.1.53/24]
      gateway4: 10.10.1.1
      nameservers:
        addresses: [<your nameservers>]
  version: 2
```

15. Connect to the vm

```
ssh <your SSO login name>@10.10.1.53
```

16. Set the hostname and reconnect
    ```
    sudo hostnamectl set-hostname 'ke3s-agent2-uc20'
    ssh <your SSO login name>@10.10.1.51
    ```

1. Install k3s snap from the snap store
    a. Install k3s snap
        ```
        snap install k3s-dbruno --edge --devmode
        ```

2. Apply workaround on `/run/user`
    ```
    sudo mkdir -p /run/user/0/snap.k3s-dbruno
    sudo chmod 700 /run/user/0
    ```

3. Apply cri-container.d apparmor profile
    a. Install wget-simosx snap
        ```
        snap install --beta wget-simosx
        snap connect wget-simosx:home
        ```
    b. Download `cri-containerd-apparmor.d`
        ```
        wget-simosx.wget
        https://raw.githubusercontent.com/dbruno74/k3s/main/cri-containerd.appar
        mor.d
        ```
    c. Load the profile
        ```
        sudo cp cri-containerd.apparmor.d /var/lib/snapd/apparmor/profiles
        sudo apparmor_parser -r
        /var/lib/snapd/apparmor/profiles/cri-containerd.apparmor.d
        ```

4. Check connectivity with VM1

    ```
    dbruno74@ubuntu:~$ ping 10.10.1.51
    PING 10.10.1.51 (10.10.1.51) 56(84) bytes of data.
    64 bytes from 10.10.1.51: icmp_seq=1 ttl=64 time=0.723 ms
    64 bytes from 10.10.1.51: icmp_seq=2 ttl=64 time=0.751 ms
    ^C
    ```

5. Start k3s agent
    a. Set k3s command line
        ```
        set k3s-dbruno k3s-cmd-line="agent --server https://10.10.1.51:6443
        --token <node-token printed on VM1 previously>"
        ```

    ```
    snap start k3s-dbruno.k3s-daemon
    ```

6. If you need to get the server log:
    ```
    snap logs k3s-dbruno.k3s-daemon -f
    ```

7. Create a convenient alias for kubectl.
    a. Add to ~/.bashrc the following line
        ```
        alias kubectl='sudo k3s-dbruno.k3s kubectl'
        ```
    b. Logout and login again

8. Check nodes on VM1

```
$ ip a
(...)
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff
    inet 10.10.1.51/24 brd 10.10.1.255 scope global ens3

$ sudo k3s-dbruno.k3s kubectl get nodes
NAME              STATUS   ROLES                  AGE    VERSION
k3s-agent1-uc20   Ready    <none>                 10m    v1.20.0+k3s2
k3s-server-uc20   Ready    control-plane,master   151m   v1.20.0+k3s2
```

Dashboard installation

1. Deploy the dashboard on k3s server (VM1) as described below:
   https://rancher.com/docs/k3s/latest/en/installation/kube-dashboard/
   Do not launch kubectl proxy on VM1, it will be launched on the host as described below.

2. Install kubectl on your host.

```
curl -LO "https://dl.k8s.io/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
echo "$(<kubectl.sha256) kubectl" | sha256sum --check
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
kubectl version --client
```

3. Copy k3s.yaml locally from k3s server (VM1),
   /var/snap/k3s-dbruno/current/etc/rancher/k3s/k3s.yaml

4. Obtain the bearer token from k3s server (VM1),

```
sudo k3s kubectl -n kubernetes-dashboard describe secret admin-user-token |
grep ^token
```

5. Launch kubectl proxy

```
kubectl proxy --kubeconfig k3s.yaml
```

6. Access the dashboard and login with the bearer token just printed
   http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/