2021 SU VE489 Project

Group 16

Name: 陆昊融，娄辰飞

Student ID: 518370910194,

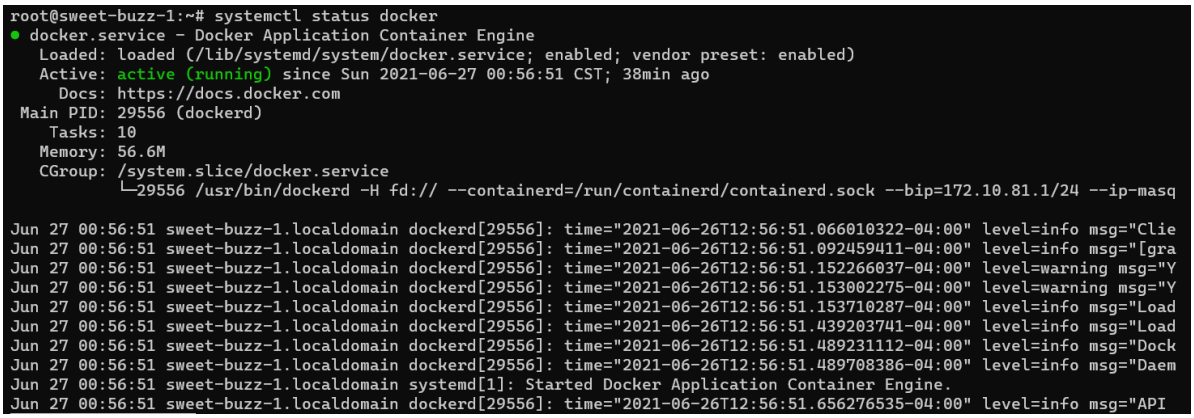
Demo Video Link: https://jbox.sjtu.edu.cn/l/YFlZQc

Task 1 Procedure

1. Install Docker on the computer

As we use a computer with Debian GNU/Linux 10 (buster) as the host machine, we just follow the Docker's official document https://docs.docker.com/engine/install/debian/ to install Docker. After installation, we check the status of Docker by running

```
systemctl status docker
```



We can see that docker runs well on our computer.

2. Pull, run, and start and attach a docker image, name the docker image as ‘rookie’

Our host machine is located in Osaka, Japan, so usually the connection is not a problem. However, if the computer is located in China, the connection to the official Docker hub may be unstable and slow. Then we can use aliyun docker hub mirror instead by changing,

```
vim /etc/docker/daemon.json
# add "registry-mirrors": ["https://registry.cn-hangzhou.aliyuncs.com"] into
daemon.json
```

Then we pull the latest version of Ubuntu from the mirror,

```
docker pull ubuntu:latest
docker image list # check the image list
```

```
root@sweet-buzz-1:~# docker image list | grep ubuntu
ubuntu                                  latest              9873176a8ff5   8 days ago     72.7MB
```

We can see that the image is pulled properly, and then we can start to create a container "rookie" based on this image,

```
docker run -it -d --name rookie ubuntu:latest /bin/bash
# -i, --interactive                    Keep STDIN open even if not attached
# -t, --tty                            Allocate a pseudo-TTY
# -d, --detach                         Run container in background and print
container ID
#     --name string                    Assign a name to the container
docker ps -a # check the container status
```

```
root@ide:~# docker ps -a | grep rookie
acddd8f6dff0        ubuntu:latest       "/bin/bash"             21 minutes ago      Up 2 seconds                                  rookie
```

We can see the container "rookie" is running properly.

3. Start bash in the docker image rookie

If we want to attach the running docker's standard input/output in the terminal, we can run,

```
rookieID=`docker ps -a | grep rookie | awk '{print $1}'` # get rookie's ID
docker start $rookieID
docker attach $rookieID
```

```
root@sweet-buzz-1:~# rookieID=`docker ps -a | grep rookie | awk '{print $1}'`
root@sweet-buzz-1:~# docker start $rookieID
5e537f4ec621
root@sweet-buzz-1:~# docker attach $rookieID
root@5e537f4ec621:/#
root@5e537f4ec621:/#
root@5e537f4ec621:/#
```

We successfully start the `/bin/bash` of "rookie".

4. Install etcd/flannel on the host machine

```
# Install etcd
tar xzvf etcd-v3.3.10-linux-amd64.tar.gz
cd etcd-v3.3.10-linux-amd64
chmod +x {etcd,etcdctl}
cp {etcd,etcdctl} /usr/bin/

# Install flannel
wget https://github.com/flannel-io/flannel/releases/download/v0.14.0/flanneld-
amd64
chmod +x flanneld-amd64
cp flanneld-amd64 /usr/bin/flanneld

# Check versions
etcd --version
etcdctl --version
flanneld --version
```

```
root@sweet-buzz-1:~# etcd --version
etcd Version: 3.3.10
Git SHA: 27fc7e2
Go Version: go1.10.4
Go OS/Arch: linux/amd64
root@sweet-buzz-1:~# etcdctl --version
etcdctl version: 3.3.10
API version: 2
root@sweet-buzz-1:~# flanneld --version
v0.14.0
```

After downloading the binaries, we need to configure etcd and flannel. Since they are both running in the background, for simplicity, we turn them into a service respectively, and use `systemd` to manage them.

```
vim /lib/systemd/system/etcd.service

# Copy following content to /lib/systemd/system/etcd.service
# Change 192.243.120.147 to your host machine's IP
################################################################################
[Unit]
Description=etcd
After=network.target

[Service]
Type=notify
Environment=ETCD_NAME=etcd-1
Environment=ETCD_DATA_DIR=/var/lib/etcd
Environment=ETCD_LISTEN_CLIENT_URLS=http://192.243.120.147:2379,http://127.0.0.1
:2379
Environment=ETCD_LISTEN_PEER_URLS=http://192.243.120.147:2380
Environment=ETCD_ADVERTISE_CLIENT_URLS=http://192.243.120.147:2379,http://127.0.
0.1:2379
Environment=ETCD_INITIAL_ADVERTISE_PEER_URLS=http://192.243.120.147:2380
Environment=ETCD_INITIAL_CLUSTER_STATE=new
Environment=ETCD_INITIAL_CLUSTER_TOKEN=etcd-cluster-token
Environment=ETCD_INITIAL_CLUSTER=etcd-1=http://192.243.120.147:2380
ExecStart=/usr/bin/etcd --enable-v2
################################################################################

vim /lib/systemd/system/flanneld.service

# Copy following content to /lib/systemd/system/flanneld.service
################################################################################
[Unit]
Description=flanneld
Before=docker.service

[Service]
ExecStart=/usr/bin/flanneld --etcd-prefix docker-flannel/network

[Install]
WantedBy=multi-user.target
RequiredBy=docker.service
################################################################################
```

```
systemctl daemon-reload
systemctl restart etcd
systemctl restart flanneld
```

5. Use flannel to configure an overlay network

```
etcdctl set /docker-flannel/network/config '{"Network":"172.10.0.0/16",
"SubnetMin": "172.10.1.0", "SubnetMax": "172.10.254.0", "Backend": {"Type":
"vxlan"}}'

etcdctl get /docker-flannel/network/config # Check network config
systemctl restart flanneld

# Check the flannel running status
cat /run/flannel/subnet.env
# You should see something like this
#############################################################################
FLANNEL_NETWORK=172.10.0.0/16
FLANNEL_SUBNET=172.10.81.1/24 # Copu this subnet, we will add this to
docker.service
FLANNEL_MTU=1450
FLANNEL_IPMASQ=false
#############################################################################
```

Now the configuration has been done and flanneld is running well. Then we need to let flannel assign an IP address to each docker container respectively. To do this, we need to modify the `docker.service`

```
vim /lib/systemd/system/docker.service

# modify this line by adding the exec options
#############################################################################
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
--bip=172.10.81.1/24 --ip-masq=true --mtu=1450
#############################################################################

systemctl daemon-reload
systemctl restart docker
```

Now the IP address should be properly assigned to each container.

6. Configure SSH in "rookie" and Final Result

Before deploying the SSH Server, we need to lay some foundations,

```
docker start rookie
docker attach rookie

# In the container rookie
echo "nameserver 8.8.8.8" | tee /etc/resolv.conf > /dev/null # Add DNS 8.8.8.8
apt update -y
apt install vim net-tools iputils-ping iproute2 -y
ifconfig # Check IP
```

If everything is OK, you should see the network interface `eth0` now has an IP address according to the subnet you set before,

```
root@5e537f4ec621:/etc# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1450
        inet 172.10.81.2  netmask 255.255.255.0  broadcast 172.10.81.255
        ether 02:42:ac:0a:51:02  txqueuelen 0  (Ethernet)
        RX packets 14  bytes 1116 (1.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

After checking the IP, we can install SSH Server on "rookie",

```
apt install openssh-server -y
vim /etc/ssh/sshd_config

# Enable PermitRootLogin
#################################################################################
PermitRootLogin yes
#################################################################################

/etc/init.d/ssh start # Start SSH Server
service ssh status # Check SSH Status
passwd # Change root password for SSH login
```

Now we can use SSH to access rookie from the new container (pawn) via the overlay network, for example,

```
docker run -it -d --name pawn ubuntu:latest /bin/bash
docker start pawn
docker attach pawn

# In the container pawn
echo "nameserver 8.8.8.8" | tee /etc/resolv.conf > /dev/null # Add DNS 8.8.8.8
apt update -y
apt install vim net-tools iputils-ping iproute2 openssh-client -y
ssh root@${IP of rookie}
```

```
root@sweet-buzz-1:~# docker start pawn
pawn
root@sweet-buzz-1:~# docker attach pawn
root@f6a2d08606db:/# ssh root@172.10.81.2
root@172.10.81.2's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 4.19.0-5-amd64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sun Jun 27 02:35:27 2021 from 172.10.81.1
root@5e537f4ec621:~#
root@5e537f4ec621:~#
root@5e537f4ec621:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1450
        inet 172.10.81.2  netmask 255.255.255.0  broadcast 172.10.81.255
        ether 02:42:ac:0a:51:02  txqueuelen 0  (Ethernet)
        RX packets 531  bytes 1039483 (1.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 430  bytes 34218 (34.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Finally, we successfully access rookie from the new container (pawn) via the overlay network.

Discussion

- As the container is not booted with systemd as init system (PID 1), we are not able to use `systemctl` to manage the SSH Server on rookie. This makes it hard to start SSH Server whenever the container starts. A workaround solution is add `service sshd start` into `/etc/rc.local`, which is executed when the container starts.
- The flannel (v0.14) seems not to support the newest version of etcd (v3.5.0), so I replace etcd v3.5.0 with etcd v3.3.10. However, downgrade operations will make the existing cluster unusable. Therefore, we need to delete etcd directories `/var/lib/etcd` before launching etcd.