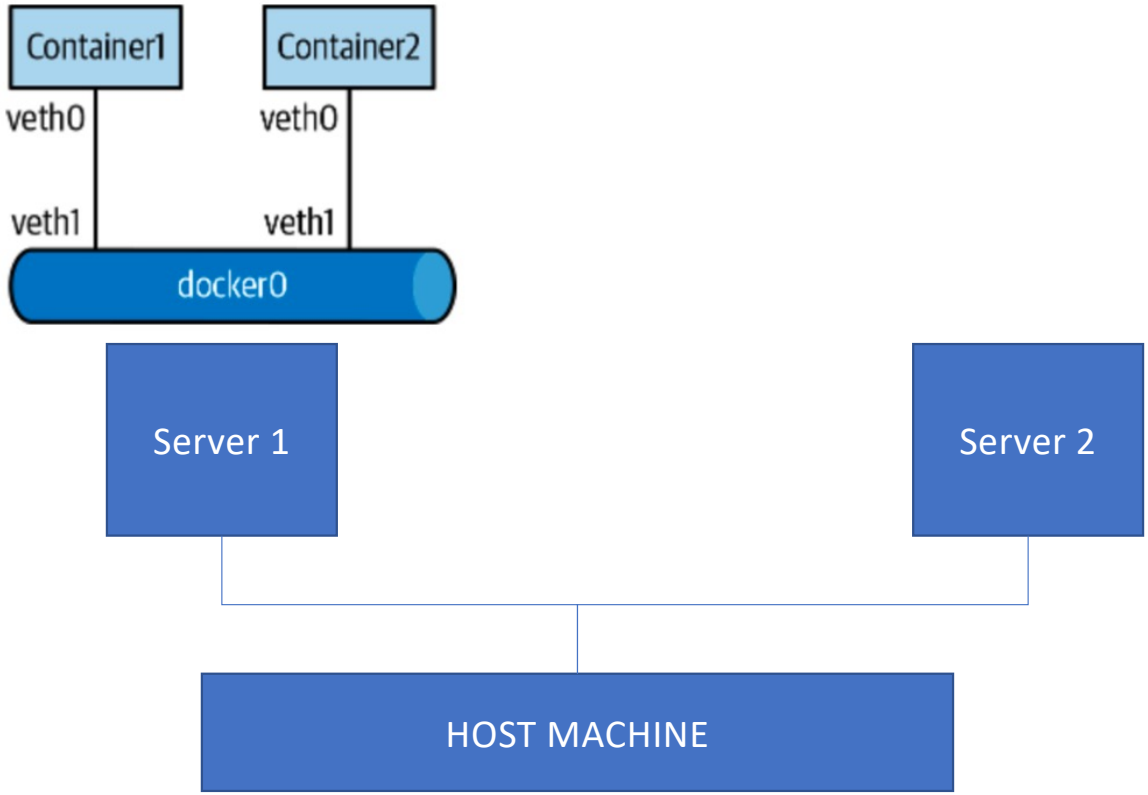
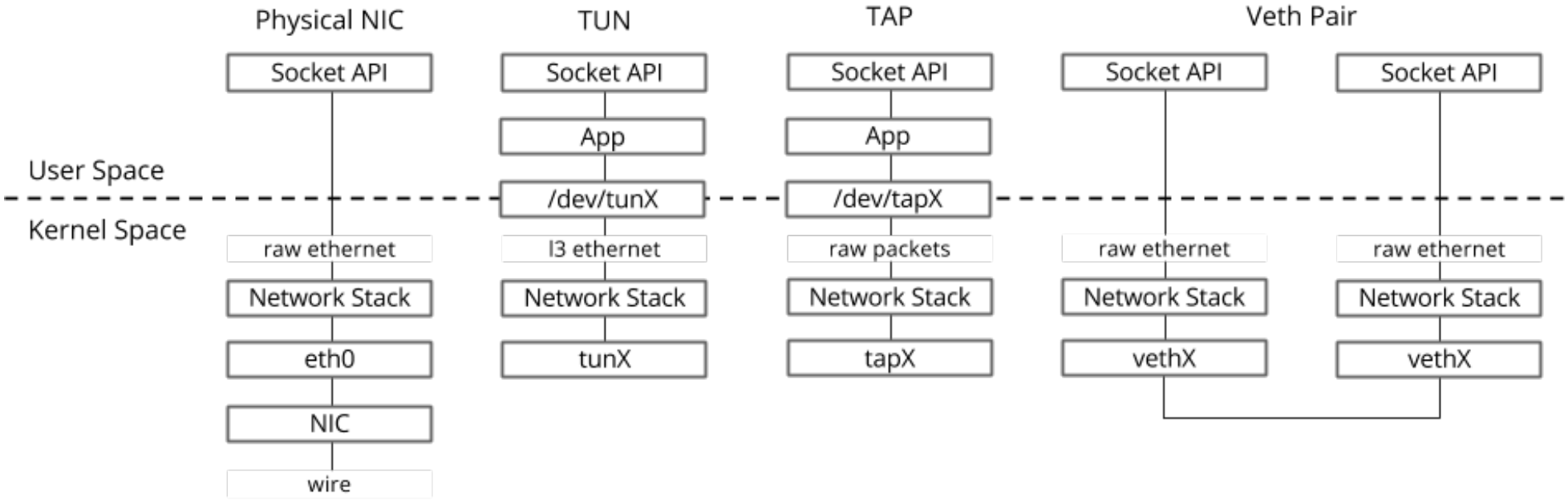
Container Networking



Reference: Cloud Native Data Center Networking – Chapter 7

Types of virtual interfaces



Network Namespaces

- `ip netns add space1`
- `ip netns add space2`
- `ip netns list`

- `ip link add veth-s1 type veth peer name veth-s2`
- `ip link list | grep veth`

- `ip link set veth-s1 netns space1`
- `ip link set veth-s2 netns space2`
- `ip link list | grep veth`

- `ip netns exec space1 ip link`

Testing namespaces

- `sudo ip netns exec space1 ip address add 10.0.0.11/24 dev veth-s1`
- `sudo ip netns exec space1 ip link set veth-s1 up`
- `sudo ip netns exec space2 ip address add 10.0.0.12/24 dev veth-s2`
- `sudo ip netns exec space2 ip link set veth-s2 up`

- `sudo ip netns exec space1 ping 10.0.0.12`

Quick recap on Dockers

- Install some pre-requisites

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

- Add the key of the official docker repository in the system

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

- Add the Docker repository

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

- Install Docker

```
sudo apt install docker-ce
```

- add this user to docker group

```
sudo gpasswd -a $USER docker
```

Docker network information

- docker network ls

```
student@student_server2:~$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
883b3a60f75e        bridge              bridge              local
0daf15d946aa        host                host                local
1b3c105b224d        none                null                local
```

Base commands

```
docker pull alpine
docker search alpine
docker run -it alpine

docker ps -a          // list running/stopped containers
docker rm ID          // rm an existing container
docker images
docker rmi ID                 // rm an existing image

docker run -dti               // detached, with terminal, interactive
docker attach alpine          // attach to container's console
ctrl+p then q                 //detaches from a container
```

Running containers

```
docker run -dit --name alpine1 alpine
```

Veth interfaces and containers

On host server

```
ip -d link
```

```
10: veth9782499@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP mode DEFAU
LT group default
    link/ether 26:bd:d5:ab:f8:9d brd ff:ff:ff:ff:ff:ff link-netnsid 2 promiscuity 1
    veth
```

On container

```
ip link
```

```
9: eth0@if10: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
```

Bridged networking: Creating a custom network

- `docker network create --driver bridge alpine-net`

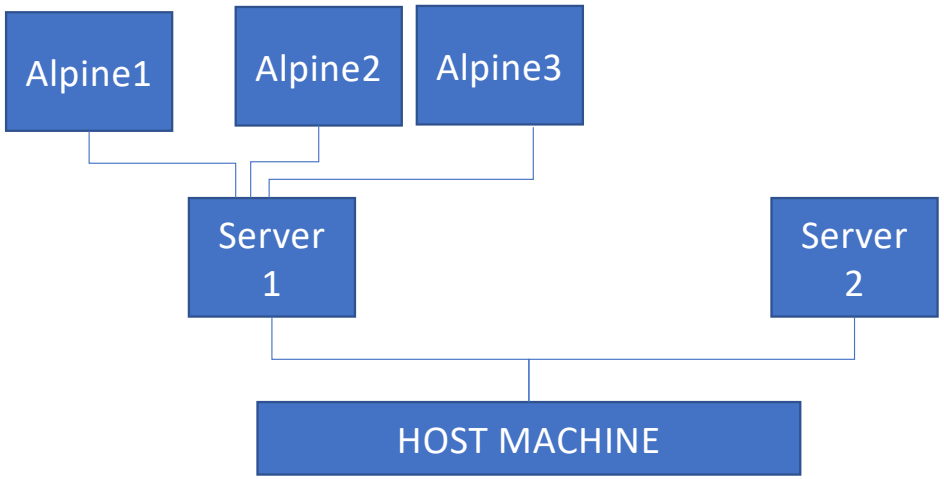
- `docker run -dit --name alpine2 alpine`
- `docker network connect alpine-net alpine2`

- `docker run -dit --name alpine3 --network alpine-net alpine`

- `docker network inspect alpine-net`

Test1

- Ping: A1 <-> A2
- Ping: A2 <-> A3
- Ping: A1 <-> A3
- Ping: Server 1 <-> Ax
- Ping: Server 2 -> Ax
- Ping: Server 2 <- Ax



Docker NAT

```
sudo iptables -L -t nat
```

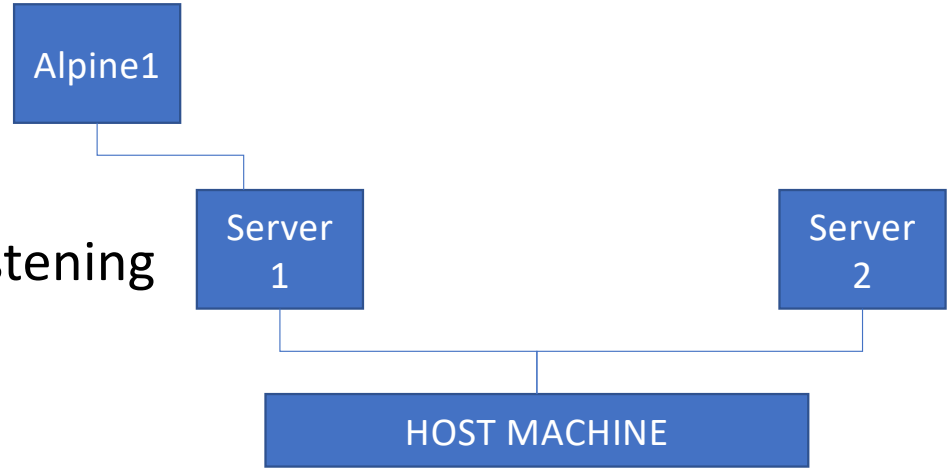
```
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  172.17.0.0/16        anywhere
```

Host network

- On server1

```
docker run -dit --network host --name alpine1 alpine
docker attach alpine1
```

Test2:



On alpine1

`nc -u -p 2020 -l` // netcat listening

- On server2

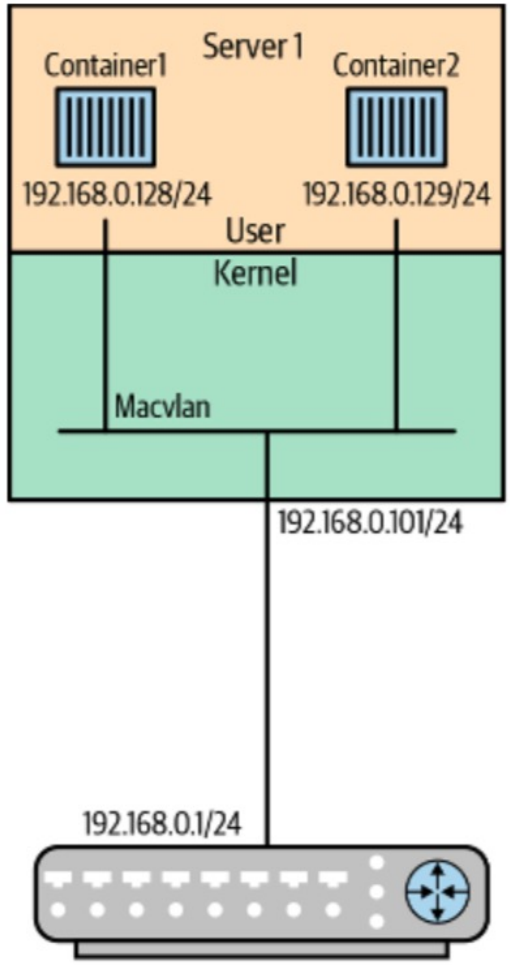
`nc 192.168.56.3 2020 -u`

- On server1

`Tcpdump -i enp0s3 port 2020`

macvlan

- Makes container appear as they are physically connected to the physical network
- For applications that expects direct connection to the network
- E.g., legacy applications, traffic monitoring



(a) Macvlan Driver in Linux Kernel

Macvlan: create

- Create network

```
docker network create -d macvlan \
    --subnet=192.168.56.0/24 \
    --gateway=192.168.56.106 \
    --ip-range=192.168.56.128/25 \
    -o parent=enp0s3 \
    macv
```

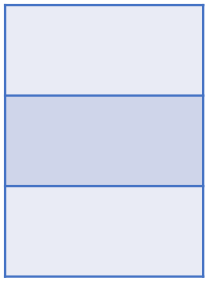
Macvlan

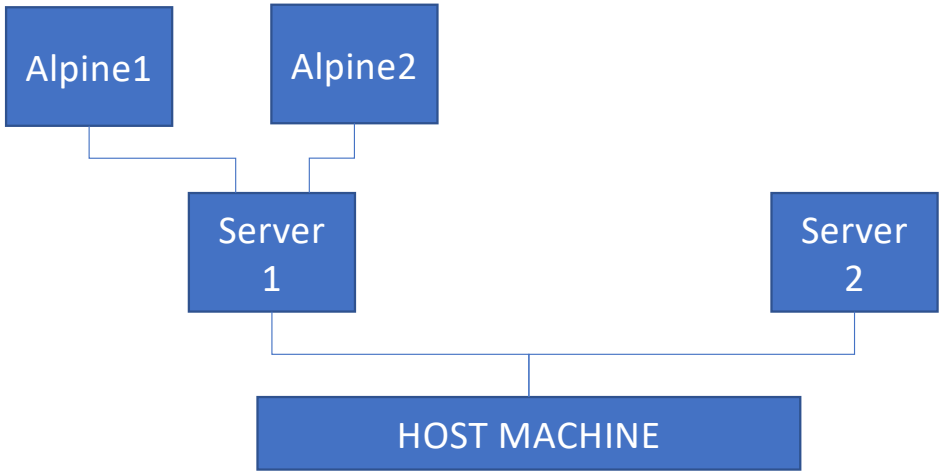
- Attach hosts

```
docker run -dit --network macv --name alpine1 alpine
```

Test3

- ping: A1<->A2
- Ping A1,A2 <-> server1
- Ping A1,A2 <-> server2





Macvlan: promiscuous

- On Vbox: enable promiscuous mode for the interface connected to vboxnet
- On host machine: enable promiscuous mode for enp0s3

```
sudo ip link set enp0s3 promisc on
```

Test4:

- ping: C1<->C2
- Ping C1,C2 <-> server1
- Ping C1, C2 <-> server2