Docker for Beginners

Day 4 – Volumes and Networking

- Lucas Albuquerque -
lucas.albuquerque@nutanix.com

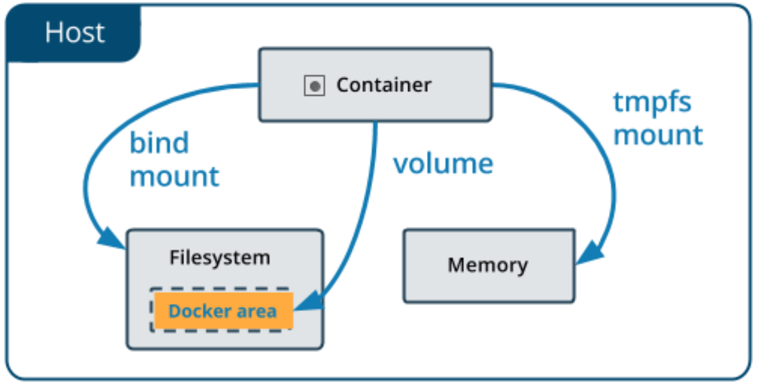
Docker Volumes

- Containers are ephemeral and mere processing units by definition
- Docker container is destroyed, any stored data will be lost.
- Docker volumes are very useful when we need to persist data in Docker containers or share data between containers.
- There are three ways to keep data safe in case of a container removal:

✓ Bind Mounts

✓ Named Volumes

✓ tmpfs



Docker Volumes

Bind Mounts

- A bind mount is just mapping a directory on the host machine to a directory in the container.
- However, when the container is removed, it does not affect the directory.
- Content can be modified by other applications

```
$ docker run -d --name nginx-app \
-v /path/to/app/directory:/usr/share/nginx/html \
hutger/nginx-app:0.1
```

Docker Volumes

Named Volumes

- Volumes which you create manually with docker ***volume create VOLUME_NAME***.
- Created in /var/lib/docker/volumes and can be referenced to by only their name.
- It is not an Union FS storage entity;
- Should not be modified by other applications;

```
$ docker volume create nginx_vol
$ docker volume ls

$ docker run -d --name nginx-app \
-v nginx_vol:/usr/share/nginx/html \
hutger/nginx-app:0.1
```

Docker Volumes

Named Volumes

- You can create or connect to remote volumes using backend drivers:

```
$ docker volume create --driver local \
    --opt type=nfs \
    --opt o=addr=192.168.1.1,rw \
    --opt device=:/path/to/dir \
    nginx-vol
```

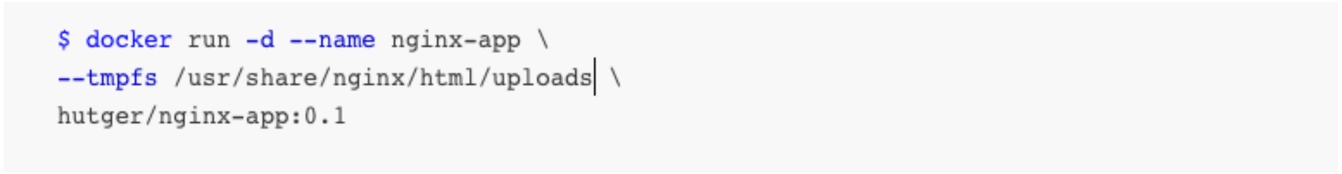
- Backend driver options by adding plugins: Nutanix DVP, Azure, Flocker, DRDB, GCE, GlusterFS, 3Par, NetApp, vSphere, etc.

https://docs.docker.com/engine/extend/legacy_plugins/

Docker Volumes

tmpfs

- If you do not want the data to persist either on the host or container
- For security reasons or to protect the performance of the container
- tmpfs mount is temporary, and only persisted in the host memory
- Unlike volumes and bind mounts, you can't share tmpfs mounts
- This functionality is only available if you're running Docker on Linux



```
$ docker run -d --name nginx-app \
--tmpfs /usr/share/nginx/html/uploads \
hutger/nginx-app:0.1
```

Docker Networking

Linux Bridge

- A **Linux bridge** is a L2 device that is the virtual implementation of a network switch inside the Linux kernel.

Network Namespaces

- Isolated network stack in the kernel with its own interfaces, routes, and firewall rules.
- It is a security aspect of containers and Linux, used to isolate containers.

Virtual Ethernet Devices

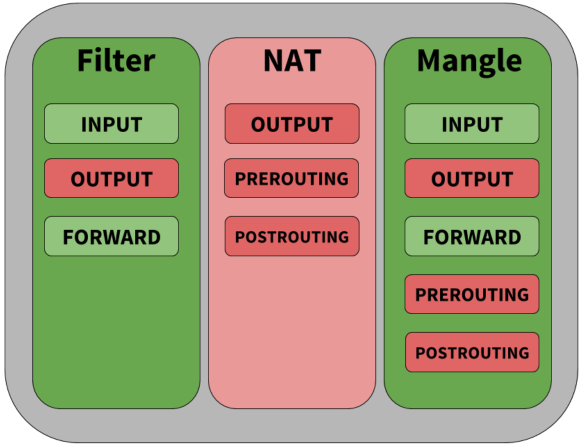
- A veth is a full duplex link that has a single interface in each namespace.
- Docker network drivers utilize veths to provide explicit connections between namespaces when Docker networks are created.

NUTANIX

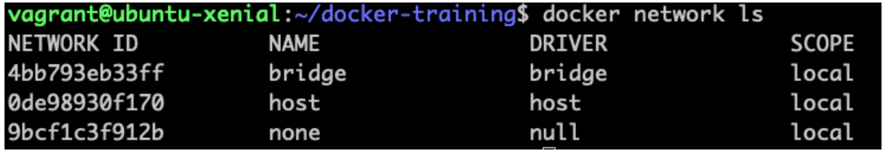
Docker Networking

IPtables

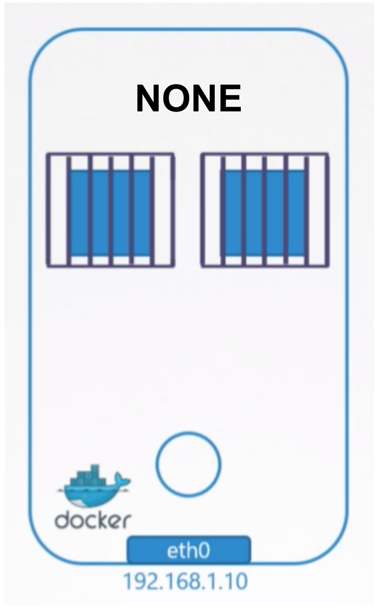
- It's a feature rich L3/L4 firewall that provides rule chains for packet marking, masquerading, and dropping.
- Docker network drivers utilize iptables extensively to segment network traffic, provide host port mapping, and to mark traffic for load balancing decisions.

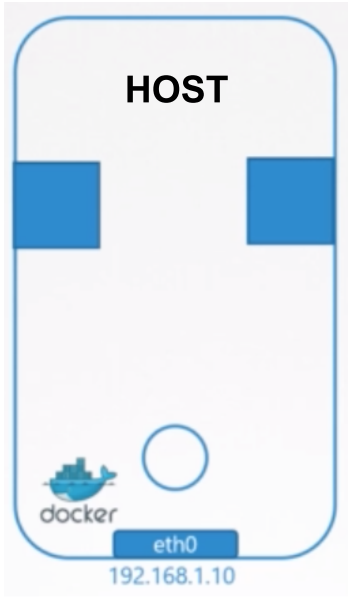


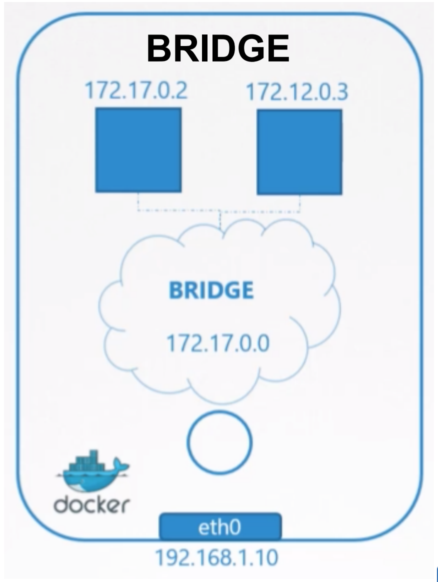
Docker Networking



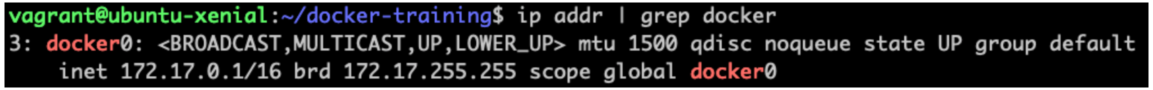
```
vagrant@ubuntu-xenial:~/docker-training$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
4bb793eb33ff        bridge              bridge              local
0de98930f170        host                host                local
9bcf1c3f912b        none                null                local
```





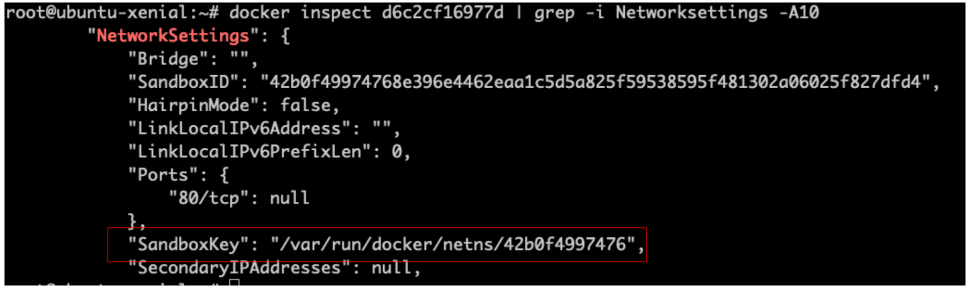


Docker Networking



```
vagrant@ubuntu-xenial:~/docker-training$ ip addr | grep docker
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
```

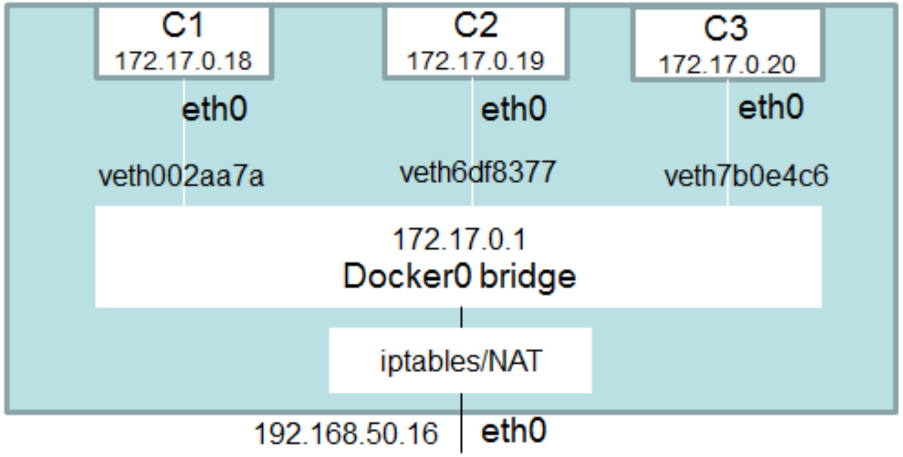
- Everytime a new container is create, a new network space is also created:



```
root@ubuntu-xenial:~# docker inspect d6c2cf16977d | grep -i Networksettings -A10
        "NetworkSettings": {
            "Bridge": "",
            "SandboxID": "42b0f49974768e396e4462eaa1c5d5a825f59538595f481302a06025f827dfd4",
            "HairpinMode": false,
            "LinkLocalIPv6Address": "",
            "LinkLocalIPv6PrefixLen": 0,
            "Ports": {
                "80/tcp": null
            },
            "SandboxKey": "/var/run/docker/netns/42b0f4997476",
            "SecondaryIPAddresses": null,
```

Docker Networking

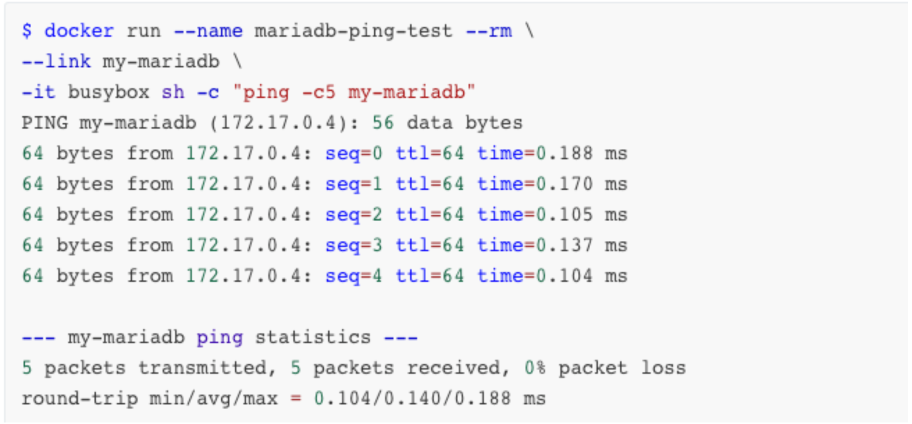
- Each container is connected to bridge **docker0** using a veth;
- Each veth connects to eth0 on container and to **docker0** on host (vethxxxx);
- Netfilter running on host is responsible for all NATs and REDIRECTs from/to containers;



Docker Networking

Linking Container

- Docker doesn't provide an native naming service solution
- In order to make reference to a different container by name, you can use the resource --link while running a container:



```
$ docker run --name mariadb-ping-test --rm \
--link my-mariadb \
-it busybox sh -c "ping -c5 my-mariadb"
PING my-mariadb (172.17.0.4): 56 data bytes
64 bytes from 172.17.0.4: seq=0 ttl=64 time=0.188 ms
64 bytes from 172.17.0.4: seq=1 ttl=64 time=0.170 ms
64 bytes from 172.17.0.4: seq=2 ttl=64 time=0.105 ms
64 bytes from 172.17.0.4: seq=3 ttl=64 time=0.137 ms
64 bytes from 172.17.0.4: seq=4 ttl=64 time=0.104 ms

--- my-mariadb ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.104/0.140/0.188 ms
```

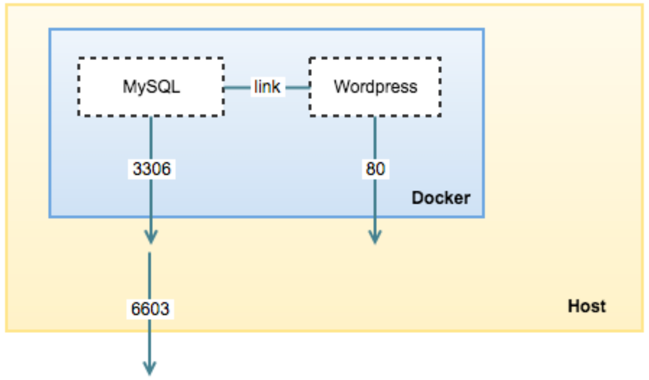
- The magic is made by basically adding an new entry on /etc/hosts of the running container with the link information.

```
172.17.0.4    my-mariadb
```

Docker Networking

Publishing Ports

- By default, the container is assigned an IP for every Docker network it connects to.
- The IP is assigned from the pool assigned to the network. Docker daemon acts as a DHCP server;
- Containers inside of the same host can communicate to each other (as long as the IPs are known)
- Publish service is a way to allow external users to connect to container services;



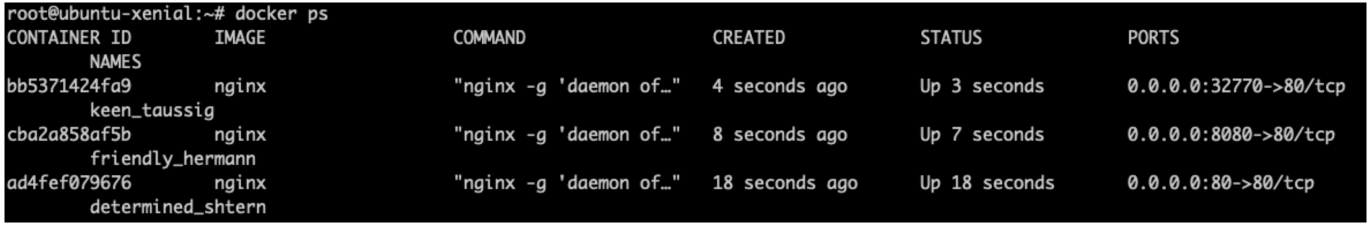
Docker Networking

Publishing Ports

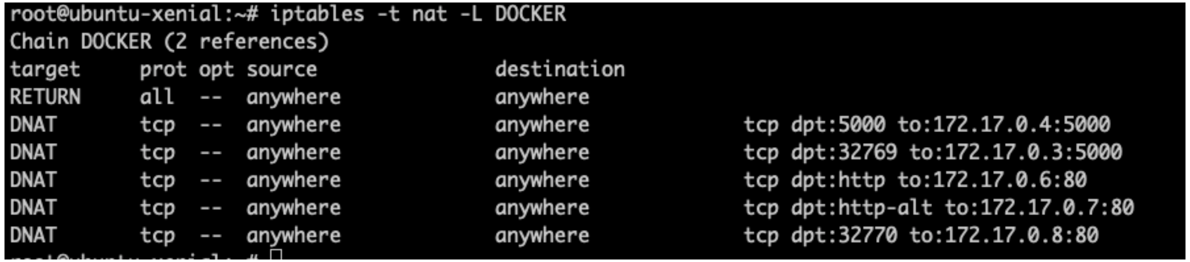
```
  # It will make a bind from port 80/TCP on host to port 80/TCP on the container
1 $ docker run -d -p 80:80 nginx

  # It will make a bind from port 8080/TCP on host to port 80/TCP on the container
2 $ docker run -d -p 8080:80 nginx

  # Publish all exposed ports to random ports. It relies on EXPOSE settings in Dockerfile.
3 $ docker run -d -P nginx
```



```
  root@ubuntu-xenial:~# docker ps
  CONTAINER ID        IMAGE                   COMMAND                  CREATED             STATUS              PORTS
          NAMES
3 bb5371424fa9        nginx                   "nginx -g 'daemon of…"   4 seconds ago       Up 3 seconds        0.0.0.0:32770->80/tcp
          keen_taussig
2 cba2a858af5b        nginx                   "nginx -g 'daemon of…"   8 seconds ago       Up 7 seconds        0.0.0.0:8080->80/tcp
          friendly_hermann
1 ad4fef079676        nginx                   "nginx -g 'daemon of…"   18 seconds ago      Up 18 seconds       0.0.0.0:80->80/tcp
          determined_shtern
```



```
  root@ubuntu-xenial:~# iptables -t nat -L DOCKER
  Chain DOCKER (2 references)
  target     prot opt source               destination
  RETURN     all  --  anywhere             anywhere
  DNAT       tcp  --  anywhere             anywhere             tcp dpt:5000 to:172.17.0.4:5000
  DNAT       tcp  --  anywhere             anywhere             tcp dpt:32769 to:172.17.0.3:5000
1 DNAT       tcp  --  anywhere             anywhere             tcp dpt:http to:172.17.0.6:80
2 DNAT       tcp  --  anywhere             anywhere             tcp dpt:http-alt to:172.17.0.7:80
3 DNAT       tcp  --  anywhere             anywhere             tcp dpt:32770 to:172.17.0.8:80
```

NUTANIX

Docker Volumes and Networking

See you tomorrow…

- Lucas Albuquerque -
lucas.albuquerque@nutanix.com

NUTANIX®

@nutanix

Thank You