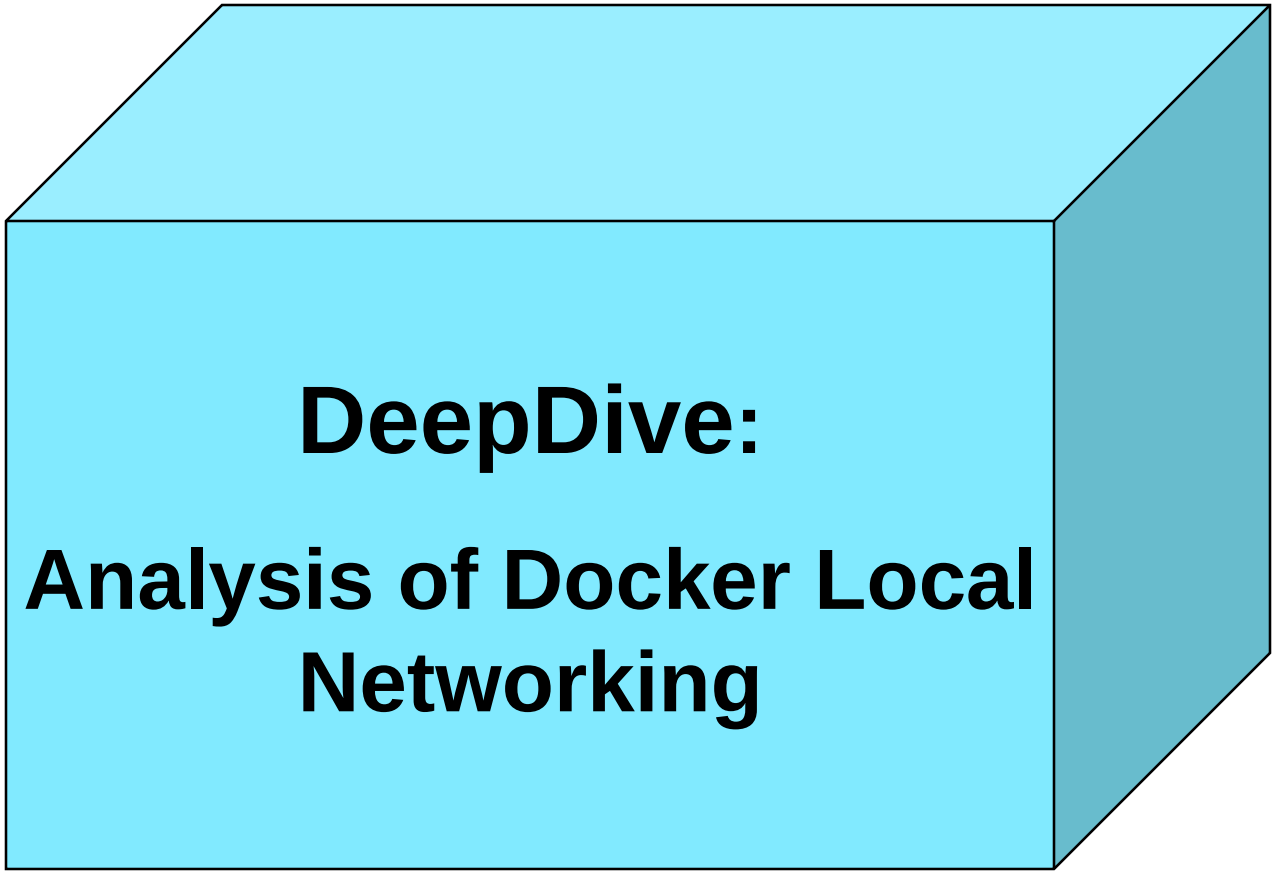

DeepDive:
Analysis of Docker Local Networking

Docker local networking structure

- The docker local networking structure is very complex
  - Every docker container running on the local system is a communicating micro service
  - A lot of interfaces on the docker host
  - Local virtual networks build by bridged subnets.
  - Internal routing and gateway routing.
- Building blocks of the local networking infrastructure
  - Interfaces
    → `ip addr show` / `ip a`
  - Bridges
    → `brctl show`
  - Subnets
    → via interfaces
  - Routing tables
    - `ip route show table main` / `ip route show` / `ip r` :
      Content of routing table main manageable by an administrator. Useful in most cases.
    - `ip route show table local` :
      routing table of local addresses managed by the kernel

http://www.system-rescue-cd.org/networking/Advanced-networking-and-policy-routing/
https://diego.assencio.com/?index=d71346b8737ee449bb09496784c9b344

Our network analysis methodology

- **Building the docker infrastructure step-by-step:**
  1. **basis: Ubuntu server 18.04. with one static-ipv4-interface (and with ssh)**
  2. **add: docker server/client (no container)**
  3. **add: running one simple container providing a webserver on port 80**
  4. **initialize docker swarm**

- **Analyze every building step by (only IPv4):**
  - **Interfaces**
  - **Bridges and subnets**
  - **Routing table**
  - **Connections and listening ports :**

    `netstat -an` **use grep in addition if necessary**

    `-a` **all active unix sockets,** `-t` **tcp sockets,** `-u` **udp sockets**

    `-n` **show ports as numbers (instead of resolving dns)**

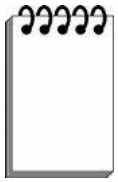
    `-l` **only ports bound to listen**

    `-p` **show program name / PID**

Step 1: Ubuntu server 18.04. with only one static ipv4 interface

- **Interfaces**

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether ca:67:51:5a:6d:91 brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.42/24 brd 192.168.178.255 scope global ens18
       valid_lft forever preferred_lft forever
    inet6 fd00:affe::c867:51ff:fe5a:6d91/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 7172sec preferred_lft 3572sec
    inet6 2001:16b8:9a:4000:c867:51ff:fe5a:6d91/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 7172sec preferred_lft 3572sec
    inet6 fe80::c867:51ff:fe5a:6d91/64 scope link
       valid_lft forever preferred_lft forever
```

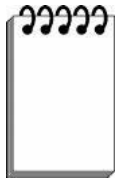


Step 1: Ubuntu server 18.04. with only one static ipv4 interface

- **Bridges and subnets: none**

```
root@dh-home2 ~ >brctl show
bridge name     bridge id               STP enabled     interfaces
```

What does STP mean?



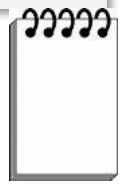
Step 1: Ubuntu server 18.04. with only one static ipv4 interface

Routing table (`ip r`)

```
default via 192.168.178.1 dev ens18 proto static
192.168.178.0/24 dev ens18 proto kernel scope link src 192.168.178.42
```

- **Listening ports (via `netstat --inet -taup`)**

```
root@dh-home2 ~ >netstat --inet -taup
Aktive Internetverbindungen (Server und stehende Verbindungen)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN      514/systemd-resolve
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN      804/sshd
tcp        0     64 dh-home2.home:ssh       pc-home2.home:54716     VERBUNDEN   1060/sshd: root@pts
udp        0      0 localhost:domain        0.0.0.0:*                           514/systemd-resolve
```



Step 2: Ubuntu server 18.04. with pure docker client/server

- **Verify docker installation**

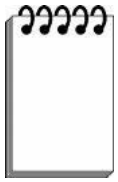
```
root@dh-home2 ~ >docker version
Client:
 Version:           18.09.6
 API version:       1.39
 Go version:        go1.10.8
 Git commit:        481bc77
 Built:             Sat May  4 02:35:57 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          18.09.6
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.8
  Git commit:       481bc77
  Built:            Sat May  4 01:59:36 2019
  OS/Arch:          linux/amd64
  Experimental:     false
```

Step 2: Ubuntu server 18.04. with pure docker client/server

- **Interfaces**

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
                                                    UP group default qlen 1000
    link/ether ca:67:51:5a:6d:91 brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.42/24 brd 192.168.178.255 scope global ens18
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
                                                    group default
    link/ether 02:42:a5:fc:1c:e7 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 scope global docker0
       valid_lft forever preferred_lft forever
```

- **Questions on docker0 Interface:**
  - **How to interpret interface docker0 in this context?**
  - **Why is it down?**



Step 2: Ubuntu server 18.04. with pure docker client/server

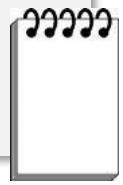
- **Bridges and subnets**

```
root@dh-home2 ~ >brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.024278cc117b       no
```

- **Questions on bridge `docker0`:**
  - **Why are there no interfaces?**
  - **Which subnet belongs to bridge `docker0` ?**
  - **Can you give a coherent explanation of the relationship between `docker0`-Bridge and `docker0`-Interface?**

- **Routing table (`ip r`):**

```
root@dh-home2 ~ >ip r
default via 192.168.178.1 dev ens18 proto static
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.178.0/24 dev ens18 proto kernel scope link src 192.168.178.42
```



Step 2: Ubuntu server 18.04. with pure docker client/server

- List docker networks

```
root@dh-home2 ~ >docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
ff333722eebf        bridge              bridge              local
0b7427f9e1ff        host                host                local
ea62cf66f475        none                null                local
```

Bridge: Default bridged network that is present on all Docker hosts.

Host: The host network adds a container on the host's network stack. there is no isolation between the host machine and the container.

none: Adds a container to a container-specific network stack. That container lacks a network interface. You have only a loop back address without interface.

User defined networks: You can define your own bridges and interfaces

Overlay network: Between containers running on several host (swarm).

Macvlan Bridge: For using VLANs

Step 2: Ubuntu server 18.04. with pure docker client/server

```
root@dh-home2 ~ >docker network inspect bridge
[    {   "Name": "bridge",
         "Id": "ff333722eebf666570da7a75ee7a3764bfb8b879d271c94417c50c36916dd411",
         "Created": "2019-05-19T11:59:16.569963486+02:00",
         "Scope": "local",
         "Driver": "bridge",
         "EnableIPv6": false,
         "IPAM": {
             "Driver": "default",
             "Options": null,
             "Config": [
                 {   "Subnet": "172.17.0.0/16",
                     "Gateway": "172.17.0.1"     } ]    },
         "Internal": false,
         "Attachable": false,
         "Ingress": false,
         "ConfigFrom": {
             "Network": "" },
         "ConfigOnly": false,
         "Containers": {},
         "Options": {
             "com.docker.network.bridge.default_bridge": "true",
             "com.docker.network.bridge.enable_icc": "true",
             "com.docker.network.bridge.enable_ip_masquerade": "true",
             "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
             "com.docker.network.bridge.name": "docker0",
             "com.docker.network.driver.mtu": "1500" },
         "Labels": {}
    }
]
```

Inspect docker networks

Step 2: Ubuntu server 18.04. with pure docker client/server

- **Listening ports (via `netstat -taup`)**

```
root@dh-home2 ~ >netstat --inet -taup
Aktive Internetverbindungen (Server und stehende Verbindungen)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN      515/systemd-resolve
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN      841/sshd
tcp        0     64 dh-home2.home:ssh       pc-home2.home:55497     VERBUNDEN   1059/sshd: root@pts
udp        0      0 localhost:domain        0.0.0.0:*                           515/systemd-resolve
```

Step 3: Ubuntu server 18.04. running one simple container (jennerwein/whoami at port 60000)

- **Interfaces**

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
qlen 1000
    link/ether ca:67:51:5a:6d:91 brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.42/24 brd 192.168.178.255 scope global ens18
       valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:f4:70:87:67 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
5: veth9c996da@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master
          docker0 state UP group default
    link/ether 42:62:55:48:ac:18 brd ff:ff:ff:ff:ff:ff link-netnsid 0
```

- **Questions: Where is interface 4? What means `veth9c996da@if4` ?**

Step 3: Ubuntu server 16.04. running one simple container

- **Bridges and subnets**

```
root@dh-home2 ~ >brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.0242f4708767       no              veth9c996da
```

- **Routing table (`ip r`)**

```
default via 192.168.178.1 dev ens18 proto static
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
192.168.178.0/24 dev ens18 proto kernel scope link src 192.168.178.42
```

- **Listening ports (via `netstat -tulp`)**

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN      526/systemd-resolve
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN      835/sshd
tcp        0     64 dh-home2.home:ssh       pc-home2.home:52125     VERBUNDEN   1049/sshd: root@pts
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN      835/sshd
tcp6       0      0 [::]:60000              [::]:*                  LISTEN      3491/docker-proxy
udp        0      0 localhost:domain        0.0.0.0:*                           526/systemd-resolve
```

Zugriff auch mit IPv4 möglich!

Step 3: Ubuntu server 18.04. running one simple container

Go inside the container and look around! (docker exec -it whoami-port60000 sh)

- **Inside container: Interfaces**

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
4: eth0@if5: <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
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

- **Inside container: Bridges and subnets: none**
- **Inside container: Routing table (`ip r`)**

```
default via 172.17.0.1 dev eth0
172.17.0.0/16 dev eth0  src 172.17.0.2
```

Step 3: Ubuntu server 18.04. running one simple container

- **Inside container**: Listening ports (via `netstat -taupn`)

```
/usr/src/app # netstat -taupn
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 :::8080                 :::*                    LISTEN      21/node
```

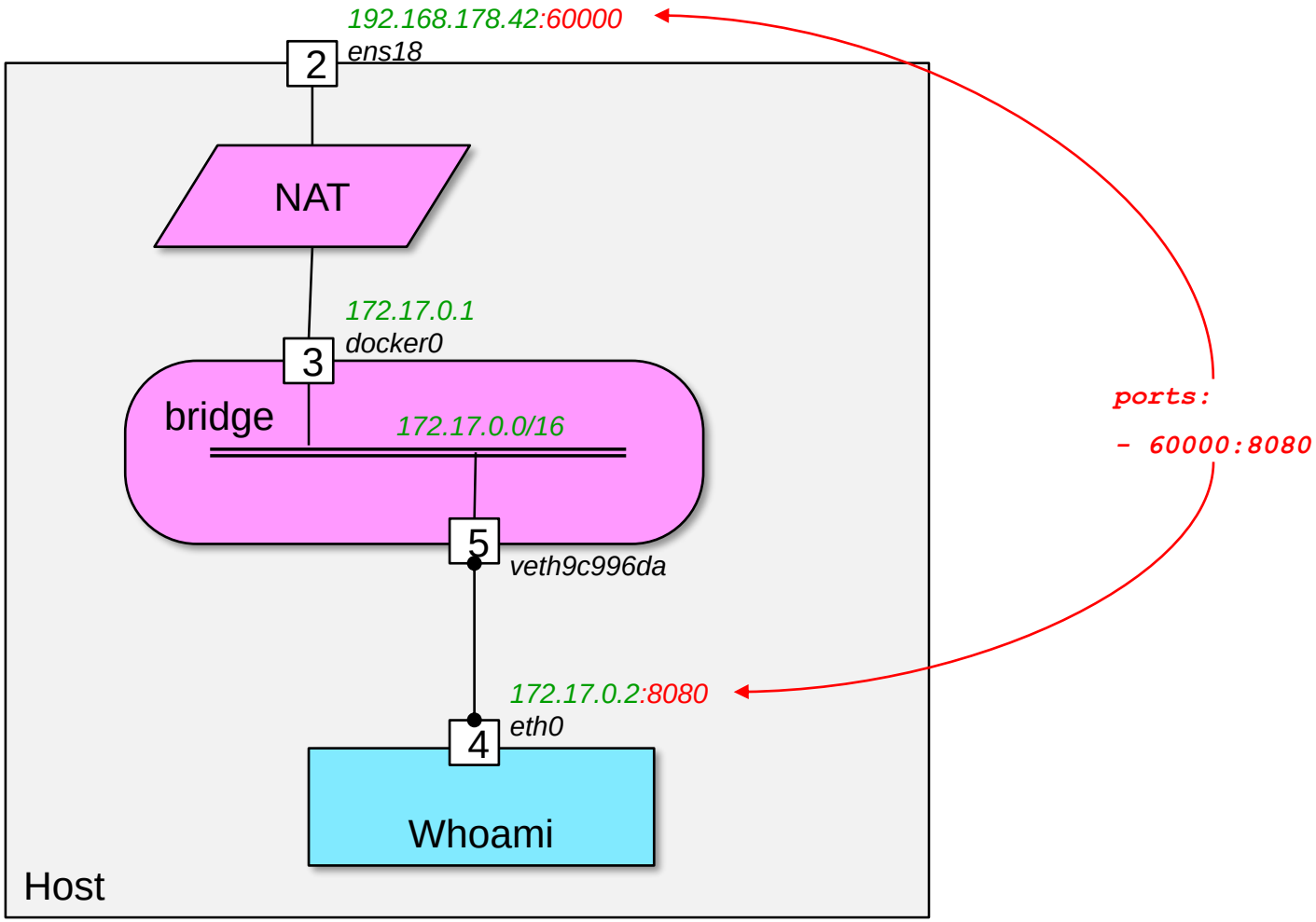
Started with PID 21

- **Inside container**: Running processes within the whoami container

```
/usr/src/app # psaux
sh: psaux: not found
/usr/src/app # ps aux
PID   USER     TIME  COMMAND
    1 root      0:00 npm
   21 root      0:00 node server.js
   64 root      0:00 sh
   83 root      0:00 ps aux
```

```
/usr/src/app # pstree -p
npm(1)---node(21)
```

Results summarized in a figure



Step 4: Ubuntu server 18.04. + docker swarm init

(docker swarm init --advertise-addr 192.168.178.42)

- **Interfaces**

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether ca:67:51:5a:6d:91 brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.42/24 brd 192.168.178.255 scope global ens18
       valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:89:0a:3b:05 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
5: vethbe934c7@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
                          master docker0 state UP group default
10: docker_gwbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:52:08:ac:47 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global docker_gwbridge
       valid_lft forever preferred_lft forever
12: veth8149801@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
                             master docker_gwbridge state UP group default
    link/ether ba:0a:6b:8d:86:a8 brd ff:ff:ff:ff:ff:ff link-netnsid 2
```

Step 4: Ubuntu server 18.04. + docker swarm init

- **Bridges and subnets**

```
root@dh-home2 ~ >brctl show
bridge name             bridge id               STP enabled     interfaces
docker0                 8000.0242890a3b05       no              vethbe934c7
docker_gwbridge         8000.02425208ac47       no              veth8149801
```

- **Routing table (`ip r`)**

```
root@dh-home2 ~ >ip r
default via 192.168.178.1 dev ens18 proto static
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
172.18.0.0/16 dev docker_gwbridge proto kernel scope link src 172.18.0.1
192.168.178.0/24 dev ens18 proto kernel scope link src 192.168.178.42
```

Step 4: Ubuntu server 18.04. + docker swarm init

- List docker networks

```
root@dh-home2 ~ >docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
be58999f4828        bridge              bridge              local
16716489872e        docker_gwbridge     bridge              local
0b7427f9e1ff        host                host                local
e4wfza5xluss        ingress             overlay             swarm
ea62cf66f475        none                null                local
```

Step 4: Ubuntu server 18.04. + docker swarm init

- **Inspect docker_gwbridge**

```
.
.
"Containers": {
            "ingress-sbox": {
                "Name": "gateway_ingress-sbox",
                "EndpointID":
"d8dce927f316f829474cab6482c553ad2e98d7893bf13dcae28ec4d37295838c",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        },
.
.
```

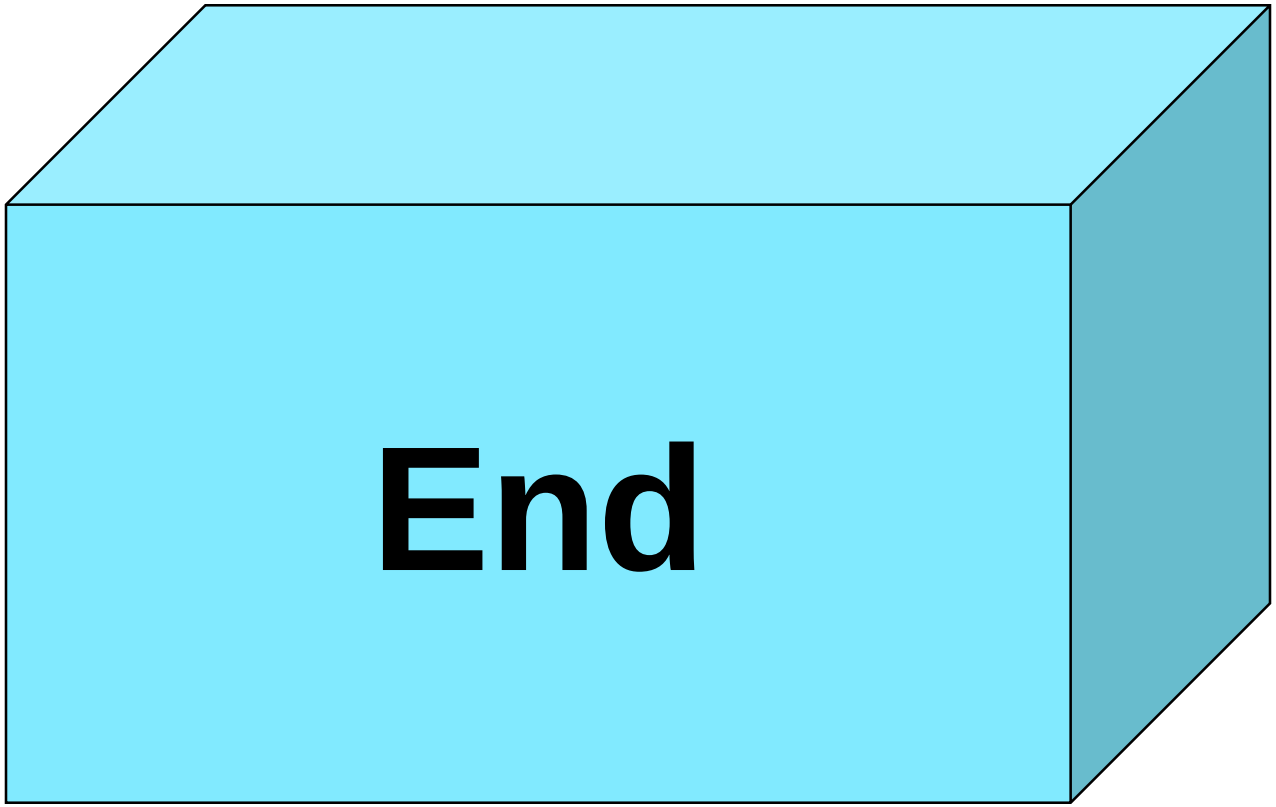
- **The docker_gwbridge is similar to docker0.
But it is not used to connect a container to the external network. It is used for exposing a container in docker swarm in conjunction with the ingress network.**

Step 4: Ubuntu server 18.04. + docker swarm init

- **Inspect ingress**

```
.
.
"Containers": {
            "ingress-sbox": {
                "Name": "ingress-endpoint",
                "EndpointID":
"fba43bb6196757352a548e9210bf40f6534e30b496c51a018f1e59dab90f8010",
                "MacAddress": "02:42:0a:ff:00:02",
                "IPv4Address": "10.255.0.2/16",
                "IPv6Address": ""
            }
        },
.
.
```

- **The Network `10.255.0.2/16` is the ingress network**


End