[Docker SecDevOps] Capítulo 5 : Redes

conceptos básicos

- net namespace: componente kernel Linux, encargado de crear cada instancia correspondiente a cada contenedor, aislado, evitando que acceda a internet u otro contenedor.
- Open vSwitch: switch programable virtual que permite multicapas, tunneling, y automatización
- IPables: firewall integrado de Linux
- Apparmor: aplicación que permite asignar unos perfiles de seguridad según la aplicación que se ejecute
- NAT
- instalación herramientas: iproute2, bridge-utils

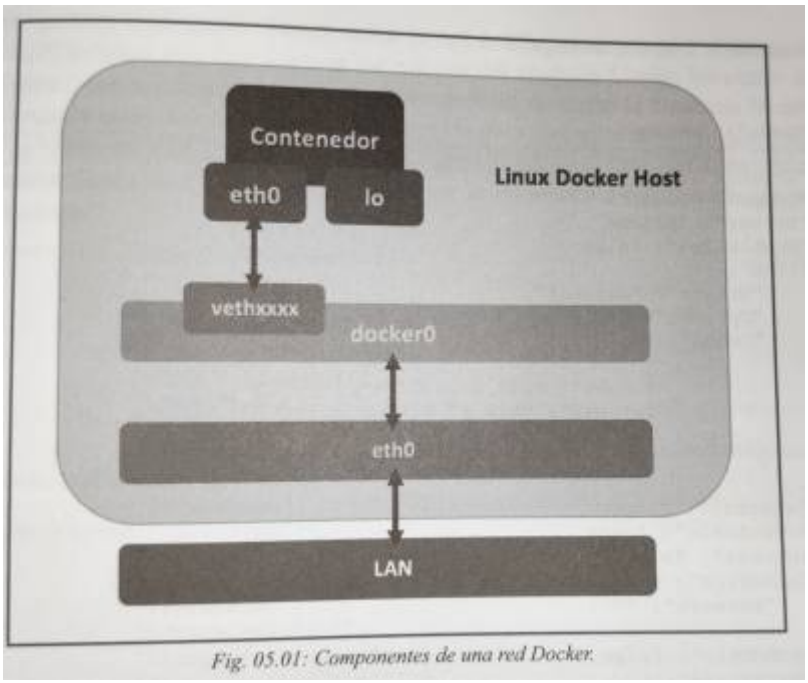
- El demonio docker crea una interfaz de red virtual (bridge) con el nombre **docker0** y la IP **172.17.0.1/24**
- También se crean por defecto 3 redes distintas → `docker network ls`:
  - **bridge**: asociada a la interfaz **docker0**. Aquí se conectan por defecto todos los contenedores, actuando como una especie de proxy
  - **none**: se usa cuando queramos asignar el interface loopback (lo). Sin red asociada
  - **host**: : asociada a la interfaz eth0 y es la que utiliza el host de Docker
- Cuando se crea un contenedor, se crean 2 interfaces asociadas:
  - **eth0**, asociada a **docker0** (y el rango de esta), para comunicación entre contenedores. Estas comunicaciones se realizan a través de una interfaz de Linux llamada *Virtual Ethernet (veth)*, que se crea con cada contenedor con el nombre **vethxxxxx**
  - **lo**, 127.0.0.01. para comunicaciones del mismo contenedor
  - 

Fig. 05.01: Componentes de una red Docker.

- En el proceso de creación del contenedor, se le puede especificar a `docker run` el parámetro **--net**:
  - --net default: interfaz **brige** por defecto usada
  - --net=none: sin configuración de red
  - --netcontainer1:container2: compartición entre los dos contenedores del *namespace*
  - --net=host: compartición del *namespace* con el host
  - para saber en que red está:
    - `docker network inspect bridge` (sección Containers)
      - además ofrece información del rango y el gateway (sección IPAM → Config)
    - `docker container instect <CONTAINER_ID>`
      - `-f` o `--format`: buscar información concreta en el resultado del comando
        - `docker inspect -f`

`range.networksettings.networksipaddressend` <CONTAINER_ID>

tipos de redes

bridge

- conecta por defecto todos los contenedores en la misma red privada.
- el contenedor está aislado del host
- solo se puede usar con un solo host
- Para que los contenedores tengan acceso a internet, hace falta usar NAT y mapeo de puertos
- Permite usar el mismo puerto para diferentes contenedores (con diferente IP dentro de la red bridge)
- Permite el acceso a través del mapeo de puertos

host

- desaparece el aislamiento del contenedor
- comparte red con el host, igual de expuesto
- dos contenedores no pueden compartir puerto (como hacíamos en bridge)
- ejecutar aplicaciones o entornos de desarollo con necesidad el mayor rendimiento posible de red.
- por contra, los servicios a ejecutar son más limitados.

over lay

- entornos distribuidos con más de un host
- docker swarm
  - `docker swarm init`
    - `docker swarm join-token manager`
  - `docker network create -d overlay network-overlay`
    - `--attachable`: To create an overlay network which can be used by swarm services **or** standalone containers to communicate with other standalone containers running on other Docker daemons
  - en este modo tiene habilitado el cifrado de información
    - el parámetro igualmente es `--opt encrypted`

MacVLAN

- configuración varias capas tipo 2 en una única interfaz física → permite tener varias sub-interfaces virtuales de red (VLAN), con MAC propia (generada aleatoriamente) e IP. Esta VLAN es la encargada de comunicar a ambos contenedores además de conectar el host usando la interfaz de red del mismo.
- requiere docker > 1.12.0
- `docker network create -d macvlan --subnet=192.168.1.0/24 --gateway=192.168.1.1 -o parent=eth0 mired_vmaclan`
- `docker container run --net=mired_vmaclan --ip=192.168.1.3 -itd ubuntu /bin/sh`

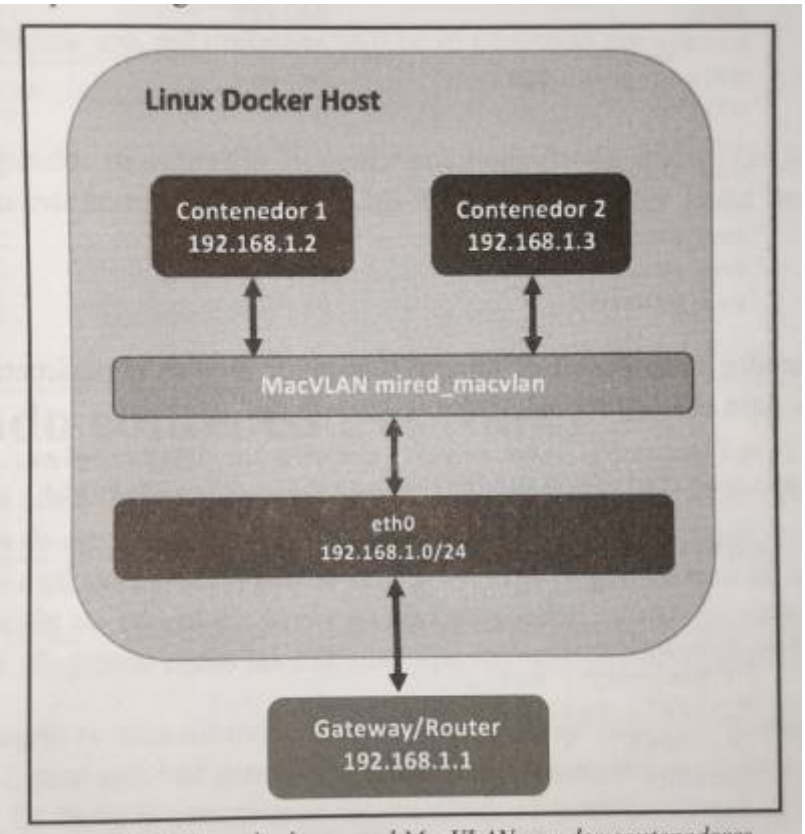


Fig. 05.04: Ejemplo de una red MacVLAN con dos contenedores.

- permite conectar contenedores sin usar la red bridge con la ventaja de crear diferentes subinterfaces a medida para una conexión más sencilla y óptima. Permite aplicar las ventajas de VLAN a docker (como trunking)

creación y gestión de redes Docker

- 2 redes bridge aisladas entre si:
  - `docker network create --driver=bridge --subnet=10.1.1.0/24 --gateway=10.1.1.254 red_prueba1`
  - `docker network create --driver=bridge --subnet=192.168.10.0/24 --gateway=192.168.10.254 red_prueba2`
    - `docker run --net=red_prueba1 -itd –name=contenedor_nginx nginx`
    - `docker container inspect contenedor_nginx`
  - añadir contenedor a (otra) red: `docker network { connect | disconnet } red_prueba1 <CONTENEDOR>`
    - parece ser que desde `docker run` no se puede asignar un contenedor a 2 redes.
  - elmiminar red:`docker network rm red_prueba1`

enlazando contenedores (link)

- contenedores en el mismo host
- comunicación unidireccional
- no se muestran puertos del contenedor en la red externa
- se pueden enlazar varios comandos link en la misma instruccción
- queda reflejado en **/etc/hosts**
- crea variables de entorno → `env`
- `docker run -it --name apache3 --link apache2:alias_apache2 httpd /bin/bash`
  - **apache3** tiene acceso a **apache2**

From:
https://miguelangel.torresegea.es/wiki/ - **miguel angel torres egea**

Permanent link:
https://miguelangel.torresegea.es/wiki/info:libros:docker-sec-dev-ops:cap5

Last update: **28/04/2022 06:37**