Tercer laboratorio

Documentación oficial de Docker

Especificación de Compose

1. Trabajar con redes en docker

Docker ofrece un conjunto robusto de herramientas para la gestión de redes. Este documento detalla cómo utilizar `docker network` y sus subcomandos, las configuraciones de red predeterminadas, personalizaciones avanzadas, y cómo conectar contenedores entre sí o con servicios externos.

Gestión de Redes en Docker

Comandos básicos de `docker network`

```
docker network

Usage:  docker network COMMAND

Manage networks

Commands:
  connect     Connect a container to a network
  create      Create a network
  disconnect  Disconnect a container from a network
  inspect     Display detailed information on one or more networks
  ls          List networks
  prune       Remove all unused networks
  rm          Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.
```

Listar redes existentes

```
docker network ls
```

Salida ejemplo:

```
NETWORK ID     NAME      DRIVER    SCOPE
34e8e7fd1dd8   bridge    bridge    local
e9a4a3af6bb3   host      host      local
d951c6c1dc60   none      null      local
```

Redes predeterminadas:

- **bridge**: Red por defecto (docker0). En Linux, durante la instalación, se crea una interfaz de red virtual llamada `docker0`. Los contenedores usan esta red a menos que se especifique otra.
- **none**: Los contenedores no tienen interfaz de red, solo `loopback` (lo).
- **host**: El contenedor comparte la configuración de red del servidor Docker Engine.

Inspeccionar redes

```
ip a show docker0
```

Salida ejemplo:

```
8: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
group default
    link/ether 02:42:d0:cd:13:cf 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
```

```
docker network inspect bridge
```

Salida ejemplo:

```
"Subnet": "172.17.0.0/16",
"Gateway": "172.17.0.1"
```

2. Exponer servicios de contenedores

Publicación de puertos

Ejecutar contenedores con puertos expuestos:

```
docker run -dtiP --name web1 httpd
```

Publicar manualmente un puerto:

```
docker run -dti -p 8085:80 --name web2 nginx
```

Consultar las reglas de iptables:

```
iptables -t nat -L DOCKER -v -n
```

Listar contenedores activos:

```
docker ps -a
```

Consultar puertos asignados:

```
docker port web1
```

Explicación de opciones:

- `-P`: Publica un puerto aleatorio libre del host.
- `-p`: Publica un puerto específico definido manualmente.

3. Personalización del Daemon de Docker

Configurar el archivo `daemon.json`:

```
vi /etc/docker/daemon.json
```

Contenido ejemplo:

```
{
  "bip": "192.168.50.1/24",
  "ip": "192.168.33.10",
  "hosts": ["unix:///var/run/docker.sock", "tcp://0.0.0.0:2376"]
}
```

Editar configuración del servicio:

```
systemctl edit --full docker.service
```

Modificar la línea ExecStart:

```
#ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
ExecStart=/usr/bin/dockerd
```

Reiniciar Docker:

```
systemctl daemon-reload
systemctl restart docker
```

4. Creación de Redes Personalizadas

Comando `docker network create`

Consultar opciones:

```
docker network create --help
```

Crear red con el controlador predeterminado:

```
docker network create desarrollo
```

Inspeccionar la red creada:

```
docker network inspect desarrollo
```

Salida ejemplo:

```
"Subnet": "172.18.0.0/16",
"Gateway": "172.18.0.1"
```

Redes con configuración personalizada

Crear red con rango y puerta de enlace definidos:

```
docker network create \
  --subnet 192.168.100.0/24 \
  --ip-range 192.168.100.100/30 \
  --gateway 192.168.100.100 produccion
```

Crear red sin acceso a Internet:

```
docker network create --internal interna-lan
```

Asignación de IP fija a contenedores

Solo posible con redes creadas con `--subnet`:

```
docker network create --subnet 192.168.100.1/24 produccion
```

Asignar IP fija:

```
docker run -dti --net produccion --ip 192.168.100.200 --name conte3 httpd
```

5. Conexión entre Contenedores

Enlazar contenedores

Ejemplo: Conectar un contenedor de WordPress con MySQL:

```
docker run -dti --name servidor_mysql -e MYSQL_ROOT_PASSWORD=000000 mysql:5.7

docker run -dti --name servidor_wp -p 8000:80 --link servidor_mysql:mysql
wordpress:5.6.2-php7.3
```

Verificar configuración:

```
docker exec -ti servidor_wp env
```

Contenido de `/etc/hosts` en el contenedor:

```
docker exec -ti servidor_wp cat /etc/hosts
```

Conectar contenedor a servicio externo

Ejemplo: Conectar WordPress con una base de datos en una máquina virtual:

```
docker run -dti --rm --name servidor_wp -p 85:80 \
  --add-host=mysqldb:192.168.1.5 \
  -e WORDPRESS_DB_HOST=mysqldb \
  -e WORDPRESS_DB_USER=root \
  -e WORDPRESS_DB_PASSWORD=000000 \
  -e WORDPRESS_DB_NAME=wordpress wordpress:5.6.2-php7.3
```

Verificar contenido de `/etc/hosts`:

```
docker exec -ti servidor_wp cat /etc/hosts
```

En Docker Swarm

Crear servicio conectado a una base de datos externa:

```
docker service create --name wp -p 85:80 \
  --host=mysqldb:192.168.1.5 \
  -e WORDPRESS_DB_HOST=mysqldb \
  -e WORDPRESS_DB_USER=root \
  -e WORDPRESS_DB_PASSWORD=000000 \
  -e WORDPRESS_DB_NAME=wordpress wordpress
```