

## Otras utilidades

### Índice

Contenedores.....	1
Comandos comunes Docker/Podman:.....	1
Específicos para Podman:.....	1
Específicos para Docker:.....	3
Kubernetes.....	4
Bases de datos.....	4
Visualstudio.....	5
Proxys.....	5
Servidores WEB.....	5

---

### Contenedores

Los contenedores son como vm, solo que usan el kernel del SO, no su propio kernel (Por eso no verás un contenedor Windows en un linux y viceversa). Son más rápidos que una vm de instalar, configurar, ya que puedes preparar imágenes por ejemplo para un servicio httpd...

Una de las herramientas más conocidas de contenedores es Docker. También podemos usar Podman. Sirve para manejar PODS (Como contenedores múltiples)

Para instalar podman el paquete se llama podman. `yum install podman`

Para el manual tenemos `man podman`

(Para la RHCSA)

En los comandos añadimos delante lo que estemos usando (ya sea docker o Podman)

#### Comandos comunes Docker/Podman:

`pull nombreimagen`: Descarga la imagen del repositorio. **Ej:** `pull docker.io/library/httpd`

`images`: Ve todas las imágenes que hay en el sistema.

`ps`: Enumera los contenedores que se están ejecutando, con `-a` vemos también los que se ejecutaron.

`rm ID/name`: Borramos el contenedor, con `rmi` borramos imagen.

#### Específicos para Podman:

Con podman `help` vemos los comandos de podman (sigue la misma idea que Docker, los comandos son parecidos).

Para buscar imágenes hacemos `search httpd`: Vemos las estrellas que tiene y si es oficial entre otros, también vemos de qué librería viene.

`run -dt nombreimg`: Ejecutamos la imagen que queramos en una nueva terminal. Con `-p` ponemos el puerto si es necesario

**Ej:** `run -dt -p 8080:80/tcp docker.io/library/httpd`

`stop ID/namecont`: Lo paramos, o con el ID o con el nombre (Eso lo vemos con el `ps`, no confundir el nombre del contenedor con el nombre de la imagen).

`logs ID`: Vemos los logs del contenedor.

`top`: Vemos los procesos al igual que en el `top` normal

`inspect nombreimagen`: Da un montón de info del file que genera el contenedor.

Otra opción para esto es `skopeo`, lo instalamos con `yum` o `dnf`:

`skopeo inspect nombreimagen`: La ventaja es que no tenemos que descargarla para inspeccionarla.

### - Configurar un contenedor como servicio e iniciarlo automáticamente:

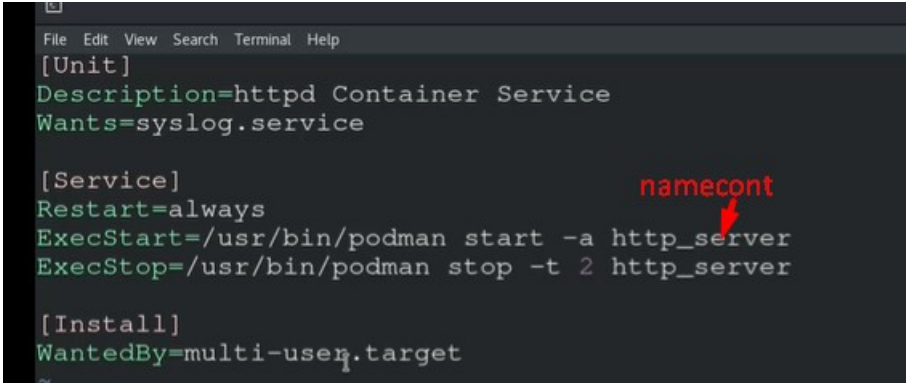
Este sin `podman`, ya que aquí tocamos la conf de SELINUX:

`setsebool -P container_manage_cgroup on`

`run -p --name namecont -p 8080:80 nombreimg`

Vemos que se ha hecho con `ps`. Ahora configuramos el contenedor como un servicio:

Luego creamos el archivo `/etc/systemd/system/httpd-container.service`



```
[Unit]
Description=httpd Container Service
Wants=syslog.service

[Service]
Restart=always
ExecStart=/usr/bin/podman start -a http_server
ExecStop=/usr/bin/podman stop -t 2 http_server

[Install]
WantedBy=multi-user.target
```

Para hacernos una idea podemos mirar en el examen el archivo `/etc/systemd/system/syslog.service`

Para recargar la config del `systemctl` hacemos `systemctl daemon reload`

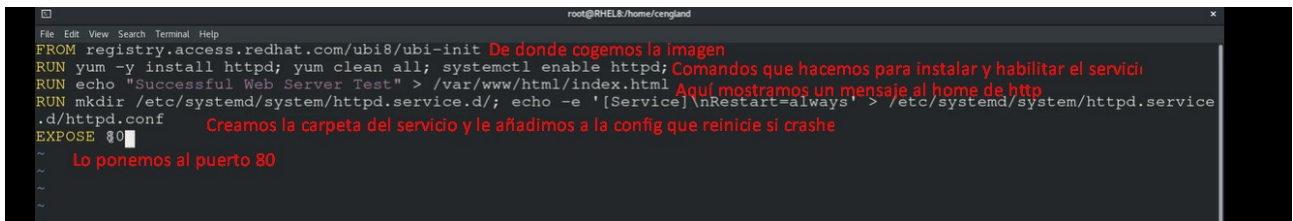
Luego para probarlo iniciamos el servicio (`systemctl start httpd-container.service`), si no funciona, asegurarse de que el contenedor está parado.

Ahora para que se inicie siempre (que estemos como `multi-user`, lo configurado) hacemos `enable` del servicio.

### - Correr servicios dentro de contenedores:

Con las dockerfile podemos crear nuestros propios contenedores o modificar los ya existentes.

Para crear una dockerfile creamos en un directorio nuevo el archivo Dockerfile:



```
FROM registry.access.redhat.com/ubi8/ubi-init
RUN yum -y install httpd; yum clean all; systemctl enable httpd;
RUN echo "Successful Web Server Test" > /var/www/html/index.html
RUN mkdir /etc/systemd/system/httpd.service.d; echo -e '[Service]
Restart=always' > /etc/systemd/system/httpd.service.d/httpd.conf
EXPOSE 80
```

Ahora para crear el contenedor hacemos: `podman build -t nombreimg .`

Luego para correr el contenedor hacemos:

`podman run -d -name=nombrecont -p 80:80 nombreimg`

### Para ejecutar un contenedor con privilegios de acceso a recursos locales:

`podman run --privileged -it -v /home/rutamaq:/mnt nombreimg /bin/bash`

El priveleged para los privilegios del volumen, -it interactive terminal, -v indica el volumen.

Ahora el /mnt del contenedor estará montado en /home/rutamaqde la máquina original. Con eso compartimos un directorio entre el contenedor y la máquina.

### Específicos para Docker:

`--version`: Ve la versión

`run -it -d -name nombrecont nombreimagen`: Ejecutan la imagen de la ventana. Con -p ponemos el puerto si es necesario. Con `--network` le decimos la red si tenemos varias.

`exec -it nombrecont bash`: Entrar en el contenedor para ejecutar comandos. (Puede cambiar lo de bash, lo vemos con `docker ps` en la columna command)

`restart idcont`: Reiniciamos el contenedor. Con `stop` lo paramos y con `start` lo volvemos a iniciar.

`kill idcont`: Paramos el contenedor

`stats`: Vemos lo que están gastando de memoria, cpu...

`network ls`: Ve las redes actuales de docker. Podemos cambiar ls por create nombre para crear una nueva red.

`inspect bridge`: Ve en detalle lo que le pidamos por ejemplo las bridge network.

Docker Compose:

Sirve para crear varios contenedores de docker con un comando. Apt install docker-compose

Hay que crear carpeta para usarlo y un archivo (y hacer allí los comandos). Docker-compose.yaml con un contenido así:

```
version: "3"
services:
  website:
    image: nginx
    ports:
      - "8081:80"
    restart: always
```

Lo iniciamos con: docker-compose up -d (La -d es para que esté en segundo plano)

Con docker-compose ps vemos los

contenedores iniciados con compose.

Con docker-compose down lo eliminamos, con stop lo paramos.

Si queremos dos contenedores ponemos algo así:

```
version: "3"
services:
  website:
    image: nginx
    ports:
      - "8081:80"
    restart: always
  website2:
    image: nginx
    ports:
      - "8082:80"
    restart: always
```

Para la red ponemos algo tal que así:

```
networks:
  coffee:
    ipam:
      driver: default
      config:
        - subnet: "192.168.92.0/24"
```

## Kubernetes

Herramienta de orquestación de contenedores (para administrar muchos contenedores a la vez). Tenemos un contenedor master y en cada contenedor kubelet y kube-proxy

## Bases de datos

Sistemas de gestión de BBDD comunes: MySQL, Microsoft SQL Server, PostgreSQL, Oracle...

Hay dos tipos de BBDD relacionales (comunes, las tablas están “conectadas”) y no relacionales.

### Comandos Mysql (una vez conectados acaban con ;):

mysql: Nos conectamos a mysql. Parámetros

-u user: Decimos el usuario

-P puerto: Decimos el puerto

-h:

show databases ;: Vemos las bbdd. (Para tablas tables)

`use bbdd;` Entramos en una bbdd.

`create database bbdd;` Crea una base de datos. (Para tabla table)

`describe table;` Da info sobre una tabla.

`select * from table where columnaedad = "35" or columnaedad = "26";` Muestra los datos pedidos. Para datos int podemos hacer `<` para menor y `>` para mayor. Where not excluye valores. Delete from borra datos. Order by columna edad asc ordena desc.

## Visualstudio

### Proxys

Un servidor proxy proporciona una puerta de enlace entre los usuarios e Internet. Es un servidor denominado “intermediario”, porque está entre los usuarios finales y las páginas web que visitan en línea. Sirve para incrementar la seguridad.

#### Diferencias con el firewall:

Firewall trabaja en los datos de la red y la capa de transporte, mientras que el servidor proxy también trabaja o procesa los datos de la capa de aplicación.

El firewall es más costoso.

Un servidor de firewall no permite ningún acceso no autorizado. Un servidor proxy proporciona comunicación o conexión a través de la red.

Los proxys pueden ser reverse proxy (que es el que está en el lado de la web e intercepta las solicitudes de los clientes) o forward (del lado de los clientes, entre ellos e internet)

### Servidores WEB

Es un servicio que expone el puerto 80 y el 443. Hay varios servidores diferentes, como NGINX, apache o tomcat.

#### NGINX:

Instalamos con apt (o lo que sea necesario)

El directorio principal es /etc/nginx.

Allí tenemos carpetas importantes como sites-available o sites-enabled

Sites-available:

Tenemos el archivo de conf default .

Normalmente se hace un copia para cada sitio y se le pone de nombre de archivo el subdominio

Ejemplo: pagina1migue.com

En ese archivo tenemos:

root: Donde están los archivos de nuestro sitio (/var/www/html en default)

Para sitios nuevos también quitar las líneas de default server

Sites-enabled: Normalmente se crean links simbólicos a los archivos de sites-available para desactivar y activar fácilmente los sitios (ln -s)

Descomentamos también el deny en el .ht (por si copias sitios que antes estaban en apache denegar acceso al htaccess)

Añadir modulo para leer php:

instalamos php-fpm con apt

Modificamos el archivo de configuracion paginamigue.com descomentando las líneas de php (o la de sock o la otra)

Modificamos también la ruta si en el archivo no coincide la versión.

Con nginx -t comprobamos que nuestros archivos están bien