

# Cómo instalar Docker en Parrot OS 5.1

1. Abrimos un terminal.
2. Actualizamos el equipo:

```
sudo apt update
```

3. instalamos Docker en nuestro equipo:

```
sudo apt-get install docker.io
```

4. Comprobamos la versión instalada:

```
docker -v
```

5. Vemos la lista de comandos:

```
docker
```

6. Descargamos un contenedor de prueba:

```
sudo docker pull hello-world
```

7. Ejecutamos el contenedor descargado:

```
sudo docker run hello-world
```

# Usos, instrucciones y ejemplos de Docker

- Lanzar un contenedor

```
docker run --name nombre_contenedor -ti parrot.run/core
```

- Detener el contenedor

```
docker stop pcore-1
```

- Reanudar un contenedor detenido anteriormente

```
docker start pcore-1
```

- Retirar un contenedor después de su uso

```
docker rm pcore-1
```

- Lista de todos los contenedores en ejecución

```
docker ps -a
```

- Iniciar varios contenedores

En el terminal 1:

```
docker run --name pentest1 -ti parrot.run/security
```

En el terminal 2:

```
docker run --name pentest2 -ti parrot.run/security
```

En el terminal 3:

```
docker run --name msf-listener -ti parrot.run/metasploit
```

- Retirar todos los contenedores

```
docker rm $(docker ps -qa)
```

- Iniciar un contenedor y eliminarlo automáticamente al salir

```
docker run --rm -ti parrot.run/core
```

## Utiliza volúmenes para compartir archivos con el host:

Es una buena práctica no mantener contenedores docker persistentes, sino eliminarlos en cada uso y asegurarse de guardar los archivos importantes en un volumen docker.

El siguiente comando crea una carpeta de trabajo dentro del directorio actual y la monta en /work dentro del contenedor.

```
docker run --rm -ti -v $PWD/work:/work parrot.run/core
```

- Utilice los volúmenes para compartir archivos en varios contenedores

En el terminal 1:

```
docker run --name pentest -ti -v $PWD/work:/work  
parrot.run/security
```

En el terminal 2:

```
docker run --rm --network host -v $PWD/work:/work -ti  
parrot.run/security
```

En el terminal 3:

```
docker run --rm -v $PWD/work:/work -ti parrot.run/metasploit
```

- Abrir un puerto del contenedor al host

Cada contenedor docker tiene su propio espacio de red conectado a una LAN virtual.

Todo el tráfico desde el interior del contenedor docker será NATted por el equipo anfitrión.

Si necesitas exponer un puerto a otras máquinas fuera de tu ordenador local, utiliza el siguiente ejemplo:

```
docker run --rm -p 8080:80 -ti parrot.run/core
```

Tenga en cuenta que el primer puerto es el puerto que se abrirá en su host, y el segundo es el puerto del contenedor para enlazar.

Aquí hay una referencia del uso de la bandera -p:

```
-p <host port>:<container port> (e.g. -p 8080:80)
```

```
-p <host port>:<container port>/<protocol> (e.g. -p 8080:80/tcp)
```

En caso de que haya varias direcciones en la red del host:

```
-p <address>:<host port>:<container port> (e.g. -p 192.168.1.30:8080:80)
```

- Utilizar el host de red en lugar de Docker NAT

Cada contenedor Docker tiene su propio espacio de red conectado a una LAN virtual.

Todo el tráfico desde el contenedor docker será NATted por el equipo anfitrión.

Si necesitas hacer que el contenedor docker comparta el mismo espacio de red de la máquina anfitriona, entonces utiliza la bandera **--network host** como se muestra a continuación

```
docker run --rm --network host -ti parrot.run/core
```

## Contenedores para pentesting

### DVWA:

```
docker run --rm -it -p 80:80 vulnerables/web-dvwa
```

### NINJAS:

```
docker run -d -p 8899:80 opendns/security-ninjas
```

### JUICE-SHOP:

```
docker run -d -p 3000:3000 bkimminich/juice-shop
```

### HACKAZON:

```
docker run --name hackazon -d -p 80:80 mutzel/all-in-one-hackazon:postinstall  
supervisord -n
```

### WEBGOAT:

```
docker pull webgoat/webgoat-8.0 docker run -p 8080:8080 -t webgoat/webgoat-8.0
```