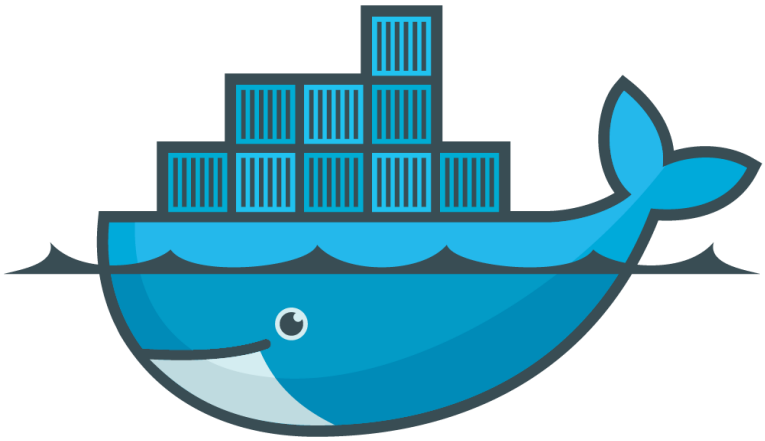
Referencia de Comandos

Docker



Autor: @adrianlois

https://zonasystem.com

ÍNDICE

Instalación Docker en Ubuntu 18.04

Instalación Docker

Actualizar los repositorios existentes.

```
sudo apt update
```

Instalar paquetes previos que permitan a apt usar paquetes a través de HTTPS.

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Agregar la clave GPG del repositorio oficial de Docker.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Agregar el repositorio de Docker en las fuentes de APT.

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
bionic stable"
```

Finalmente, se instala Docker.

```
sudo apt install docker-ce
```

Se comprueba que el daemon de Docker está iniciado y el proceso está habilita para iniciarse en el arranque.

```
sudo systemctl status docker
```

Si no estuviese iniciado y/o habilitado para el arranque del sistema.

```
sudo systemctl start docker
sudo systemctl enable docker
```

Ejecutar el comando Docker sin sudo o desde otro usuario

De forma predeterminada el comando Docker solo puede ser ejecutado por el usuario root o por un usuario del grupo docker. Donde “username” sería el nombre del usuario.

```
sudo usermod -aG docker username
```

Para comprobar que el usuario forma parte del grupo docker se pude ejecutar.

```
id -nG
```

Referencia comandos Docker

https://docs.docker.com/engine/reference/commandline/docker

Comandos gestión Docker

Listar contenedores en ejecución. Listar todos los contenedores en ejecución y parados (-a indica todos).

```
docker ps –a
```

Listar imágenes descargadas en local (-a muestra todas las imágenes incluidas las intermedias).

```
docker images
```

Descargar imagen a caché local desde un repositorio público.

```
docker pull ubuntu:18.04
```

Loguearse en Docker Hub, será necesario estar autenticado para subir (push) una imagen.

```
docker login
docker login -u <usuario> -p <password>
```

Subir una imagen a un repositorio (Docker Hub).

```
docker push <imagen_local>
```

Crear una imagen de un contenedor personalizado en ejecución.

```
docker commit <ID o nombre_contenedor> <nombre_imagen_nueva>
```

Para subir una imagen creada a un repositorio como puede ser Docker Hub, es neecsario “tagearla” antes.

```
docker tag <imagen_local> <nombre_tag_imagen>
docker push <nombre_tag_imagen_creada>
```

Construir una imagen a partir de un Dockerfile (--file será necesario si el nombre del fichero no es “Dockerfile” y no está situado el mismo directorio).

```
docker build --tag <nombre_imagen> --file <fichero_dockerfile>
```

Eliminar una imagen local (-f o --force fuerza la eliminación).

```
docker rmi <id o nombre_imagen>
```

Eliminar un contenedor, si el contenedor está en ejecución será necesario detenerlo (docker stop <contenedor>) antes de eliminarlo se puede indicar el parámetro -f o --force (esto fuerza la detención y eliminación del contenedor).

```
docker rm <id o nombre_imagen>
```

Eliminar todos contenedores, redes, imágenes o volúmenes "colgantes" sin referencia o no utilizados

```
docker system prune
#Parámetros
--all       #Eliminar todas las imágenes no utilizadas, no solo las que cuelgan.
--force     #Forzar el eliminado, no pedir confirmación.
--volumes   #Eliminar volúmenes no utilizados o colgantes.
--filter    #Proporcionar valores de filtro.
```

Crear e iniciar un contenedor basado en una imagen Ubuntu 18.04, que ejecuta una Shell en modo interactivo (-it) como entrypoint.

```
docker run –name <nombre_contenedor> -h <nombre_hostname> -it ubuntu:18.04 /bin/bash
```

Crear y ejecutar un contenedor con un comando/servicio en segundo plano un comando como entrypoint (-d modo detached, no interactivo) (-c comando dentro del modo interactivo bash).

```
docker run -d --name <nombre_contenedor> -h <hostname> <nombre_imagen> bash -c
"<comando>"
docker run -d --name mi_web -h web ubuntu_web bash -c "apache2ctl -D FOREGROUND"
```

Crear e iniciar un contenedor en modo interactivo con una Shell (-i interactivo -t asociar una tty).

```
docker run -it --name test debian bash
```

Crear contenedores. El contenedor se crea pero no se inicia (sería necesario usar después docker start). “docker créate” dispone de multitud parámetros.

```
docker create -it --storage-opt size=120G fedora /bin/bash
```

Iniciar, detener y reiniciar contenedores.

```
docker start <id o nombre_imagen>
docker stop <id o nombre_imagen>
docker restart <id o nombre_imagen>
```

Publicar puertos en contenedores (aunque se ejecute una imagen que ya predefina exposición de puertos, igualmente habría que establecerlos en la ejecución del contenedor).

```
#Publicar el puerto 80 del contenedor en un puerto aleatorio del host.
docker run -it --name test -p 80 debian bash

#Publicar el puerto 80 del contenedor al 8080 del host.
docker run -it --name test -p 8080:80 debian test
```

```
#Publicar todos los puertos expuestos a puertos aleatorios del host.
docker run -it --name test -P debian bash
```

Listar los puertos de contenedores.

```
docker port <id o nombre_imagen>
```

Finalizar (matar) contenedores.

```
docker kill <id o nombre_imagen>
```

Buscar registros de logs de contenedores (-f muestra logs a tiempo real).

```
docker logs -f <id o nombre_imagen>
```

Mostrar los procesos en ejecución de contenedores.

```
docker top <id o nombre_imagen>
```

Buscar imágenes en repositorios (Docker Hub).

```
docker search <término>
```

Importar, exportar y guardar imágenes en un archivo tar.

```
docker save --outpout <nombre_contenedor> <nombre_empaquetado_contenido.tar>
docker export --output <nombre_contenedor> <nombre_empaquetado_contenido.tar>
docker import <nombre_empaquetado_contenido.tar> <nombre_contenedor>
```

Cargar una imagen desde un archivo tar o STDIN *(Standard Input)*.

```
docker load --input <test.tar>
```

Gestionar Docker.

```
docker df                              #Mostrar el uso del disco docker.
docker system events o docker events   #Mostrar eventos en tiempo real.
docker system info o docker info       #Mostrar información de todo el sistema.
```

Desactivar procesos de contenedores.

```
docker unpause <id o nombre_imagen>
```

Actualizar la configuración de contenedores.

```
docker update <id o nombre_imagen>
```

Mostrar la versión de Docker.

```
docker version
```

Inspeccionar cambios de archivos o directorios de contenedores.

```
docker diff <id o nombre_imagen>
```

Inspeccionar contenedores (devuelve información de bajo nivel).

```
docker inspect <id o nombre_imagen>
```

Mostrar estadísticas de uso de los recursos de contenedores (CPU, Memoria, I/O, PIDs).

```
docker stats   #Muestra todos los contenedores actuales en ejecución.
docker stats <id o nombre_imagen>
```

Gestionar volúmenes Docker.

```
docker volume create    #Crear un volumen.
docker volume inspect   #Mostrar información detallada de volúmenes.
docker volume ls        #Listar volúmenes.
docker volume rm        #Eliminar volúmenes.
docker volume prune     #Eliminar volúmenes locales no utilizados.
```

Gestión de redes Docker.

```
docker network create       #Crear una red.
docker network connect      #Conectar un contenedor a una red.
docker network disconnect   #Desconectar un contenedor de una red.
docker network inspect      #Mostrar información detallada de redes.
docker network ls           #Listar redes.
docker network rm           #Eliminar redes.
docker network prune        #Eliminar todas las redes no utilizadas.
```

Montar un volumen de un directorio local en un contenedor (útil para trabajar con volúmenes de datos que se están editando habitualmente).

```
docker run -it -v /home/user/codigoWeb:/var/www/html debian bash
```

Crear un contenedor que usará un volumen asociado a otro contenedor existente.

```
#Crear el contenedor sin bash, simplemente para tenerlo como referencia para crear posteriormente otros contenedores basados en este usando su volumen de datos como referencia.
docker run -it -v /tmp:/var/www/html --name container_code debian /bin/false


#Asociar el volumen creado en el contenedor anterior con un nuevo contenedor usando el ID del contenedor.
docker run -it --name code -h code --volumenes-from ec3456a3d16cb debian bash
```

Copiar datos desde el contenedor al host local (opcional -a --archive copia toda la información uid/gid).

```
docker cp -a <id_contenedor>:<path_contenedor> <path_host_local>
```

Copiar desde el host local al contenedor.

```
docker cp -a <path_host_local> <id_contenedor>:<path_contenedor>
```

Ejecutar un comando en un contenedor en ejecución. Por ejemplo una Shell bash en modo interactivo (útil para acceder a un contenedor en ejecución).

https://docs.docker.com/engine/reference/commandline/exec

```
docker exec -it nombre_contenedor /bin/bash
```

- **Salir de un contenedor con bash interactivo**

```
# Con la combinación de teclas: Ctrl+p+q
# exit: si el contenedor se inició en modo --detach o -d
```

- **Live restore Docker**

Permite que los contenedores permanezcan en ejecución aunque el daemon no esté disponible (por ejemplo en un reinicio del daemon dockerd). Ayuda a reducir la inactividad del contenedor debido a fallas del daemon, interrupciones planificadas o actualizaciones.

https://docs.docker.com/config/containers/live-restore

Para habilitar es necesario crear el fichero "*/etc/docker/daemon.json*", con el contenido.

```
{
  "live-restore": true
}
```

- **Iniciar contenedores automáticamente**

Iniciar automáticamente un contenedor con "--restart always" se trata de una política de reinicio que siempre reinicia el contenedor si este se detiene (útil para auto iniciar el contenedor después de reiniciar la máquina host).

```
docker run -d --restart always --name test -h test debian

#Políticas de reinicio automático de contenedores
no              #No reinicia automáticamente el contenedor. (valor por defecto)
on-failure      #Reinicia el contenedor si sale debido a un error.
unless-stopped   #Reinicia el contenedor a menos que se detenga explícitamente o el propio Docker se detenga o reinicie.
always          #Siempre reinicia el contenedor si se detiene.
```

Comandos Docker Swarm

Referencia de comandos Docker Swarm.

https://docs.docker.com/engine/reference/commandline/swarm

https://docs.docker.com/engine/reference/commandline/swarm_update

Iniciar, unirse, dejar y actualizar un cluster Swarm.

```
docker swarm init     #Inicializar un Swarm.
docker swarm join      #Unirse a un Swarm como nodo worker y/o manager (dependiendo si se trata de un –token manager o worker).
docker swarm leave    #Dejar un Swarm.
docker swarm update   #Actualizar un Swarm, dispone de varias opciones.
```

Comandos Docker Machine

Referencia de comandos Docker Machine.

https://docs.docker.com/machine/reference

```
docker-machine create   #Crear un nodo.
docker-machine kill     #Detener una nodo.
docker-machine ls       #Listar nodos.
docker-machine ssh      #Iniciar sesión en un nodo activo.
docker-machine rm       #Eliminar un nodo.
docker-machine env      #Establecer variables de entorno para definir que docker debe ejecutar un comando en una máquina en particular.
```

Crear dos nodos usando driver de Amazon Web Services, estableciendo región y zona de disponibilidad, red VPC y tamaño de disco.

```
docker-machine create --driver amazonec2 --amazonec2-region "us-east-2" --amazonec2-zone "b" --amazonec2-vpc-id vpc-0d1dd38... --amazonec2-root-size 8 test
```

Comandos Docker Node

Referencia de comandos Docker Node.

https://docs.docker.com/engine/reference/commandline/node

```
docker node inspect   #Mostrar información detallada en uno o más nodos.
docker node ls        #Listar nodos del Swarm.
docker node ps        #Enumerar las tareas que se ejecutan en uno o más nodos.
docker node rm        #Eliminar uno o más nodos del Swarm.
docker node update    #Actualizar un nodo.
```

Promover un nodo worker a manager.

```
docker node promote <nodo>
```

Degradar un nodo manager a worker.

```
docker node demote <nodo>
```

Cambiar nodo a solo Manager y no Manager+Worker (que sería la función por defecto).

```
docker node update --availability drain <nodo>
```

- **Docker node update**

Referencia de comandos Docker node update.

https://docs.docker.com/engine/reference/commandline/node_update

```
--availability   #Disponibilidad del nodo ("active"|"pause"|"drain").
--label-add      #Agregar o actualizar una etiqueta de nodo (key=value).
--label-rm       #Eliminar una etiqueta de nodo si existe.
--role           #Rol del nodo ("worker"|"manager").
```

Comandos Docker Service

Referencia de comandos Docker Service.

https://docs.docker.com/engine/reference/commandline/service

```
docker service create     #Crear un nuevo servicio.
docker service inspect    #Mostrar información detallada sobre uno o más servicios.
docker service logs       #Obtener los registros de un servicio o tarea.
docker service ls         #Listar servicios.
docker service ps         #Listar las tareas de uno o más servicios.
docker service rm         #Eliminar uno o más servicios.
docker service rollback   #Revertir cambios a la configuración de un servicio.
docker service scale      #Escalar uno o múltiples servicios replicados.
docker service update     #Actualizar un servicio.
```

Crear un servicio. Se debe crear desde un nodo manager.

- *--publish*: Publicar puertos
- *--replicas*: Número de réplicas totales a repartir entre los nodos existentes.
- *--update-parallelism*: Ejecución del número de contenedores/tareas por nodo.
- *--update-delay*: Retraso entre las actualizaciones (ms|s|m|h)
- *--restart-condition on-failure*: Reiniciar la tarea en caso de fallo.
- *--constraint node.role manager*: Los nodos solo admitirán tareas de un nodo manager
- *test*: Imagen a ejecutar.

```
docker service create --name test -p 80:80 --replicas 3 --update-parallelism 1 --update-
delay 5s --restart-condition on-failure --constraint 'node.role == manager' test
```

Otra forma de crear un servicio es añadiendo \ para un salto de línea y seguir definiendo los parámetros del servicio.

```
docker service create \
--name test \
--publish 80:80 \
--replicas 3 \
--update-parallelism 1 \
--update-delay 5s \
--restart-condition on-failure \
--constraint 'node.role == manager' \
test
```

- **Docker service update**

Referencia de comandos Docker service update.

https://docs.docker.com/engine/reference/commandline/service_update

Actualizar réplicas de un Swarm de modo que se pueda controlar un escalado elástico.

```
docker service update --replicas=50 test
docker service scale test=50
```

Forzar actualización de un servicio, incluso si ningún cambio lo requiere. Útil para actualizar un cluster de nodos Swarm en los que se actualizaron réplicas en un servicio con un paralelismo concreto y Docker Swarm ignoró el balanceo de tareas (réplicas del servicio) a nuevos nodos worker unidos al Swarm.

```
docker service update --force test
```

Comandos Docker Stack

Referencia de comandos Docker Stack.

https://docs.docker.com/engine/reference/commandline/stack

```
docker stack deploy     #Implementar o actualizar un stack.
docker stack ls         #Lista de pilas.
docker stack ps         #Listar las tareas en la pila.
docker stack rm         #Eliminar una o más pilas.
docker stack services   #Listar los servicios en la pila.
```

Implementar o actualizar un stack a partir de un fichero “Docker-Compose”. Debe ejecutarse desde un nodo manager.

```
docker stack deploy --compose-file docker-compose.yml test
```