Primer laboratorio

Referencia de comandos de Docker Engine

0. Pasos previos

Parar todos los contenedores:

```
docker stop $(docker ps -a -q)
```

Eliminar todos los contenedores:

```
docker rm -f $(docker ps -a -q)
```

Eliminar todas las imagenes:

```
docker rmi -f $(docker images -a -q)
```

1. Trabajar con imágenes

Documentación relevante:

- Información de Docker
- Herramienta `dive` para trabajar con imágenes

Listar imágenes locales:

```
docker images
```

Este comando muestra todas las imágenes de Docker almacenadas localmente en el sistema. Incluye detalles como el nombre de la imagen, la etiqueta, el ID de la imagen, la fecha de creación y el tamaño.

Buscar imágenes en Docker Hub:

```
docker search nginx
```

Busca imágenes relacionadas con "nginx" en Docker Hub y muestra una lista con el nombre, descripción, número de estrellas, estado oficial, y si es automatizada.

Ayuda sobre el comando `docker search`:

```
docker search --help
```

Proporciona una lista de opciones y parámetros disponibles para usar con el comando `docker search`.

Buscar imágenes oficiales:

```
docker search -f is-official=true nginx
```

Busca imágenes relacionadas con "nginx" y filtra los resultados para mostrar solo aquellas marcadas como oficiales.

Descargar una imagen específica:

```
docker pull nginx
```

Descarga la última versión de la imagen de "nginx" desde Docker Hub al sistema local.

Descargar otra imagen específica:

```
docker pull httpd
```

Descarga la última versión de la imagen "httpd" desde Docker Hub.

Descargar una versión específica de una imagen:

```
docker pull mysql:5.7
```

Descarga la versión `5.7` de la imagen "mysql" desde Docker Hub.

Descargar la última versión de una imagen:

```
docker pull mysql
```

Descarga la última versión disponible de la imagen "mysql".

Directorio donde se almacenan las imágenes en linux:

```
/var/lib/docker/image
```

Este es el directorio predeterminado donde Docker almacena los archivos relacionados con las imágenes en el sistema.

Filtrar imágenes según las estrellas:

```
docker search --filter=stars=20 nginx
```

Busca imágenes relacionadas con "nginx" que tengan al menos 20 estrellas en Docker Hub.

```
docker search --filter stars=10 nginx
```

Busca imágenes relacionadas con "nginx" que tengan al menos 10 estrellas.

Ver el historial de una imagen:

```
docker history nginx
```

Muestra las capas de la imagen "nginx" con información sobre el tamaño y el comando utilizado para crear cada capa.

Inspeccionar una imagen:

```
docker inspect nginx
```

Proporciona detalles en formato JSON sobre la imagen "nginx", incluyendo metadatos como el ID, el tamaño y las configuraciones.

Eliminar una imagen específica:

```
docker rmi nginx
```

Elimina la imagen "nginx" del sistema local si no está siendo utilizada por ningún contenedor.

Forzar la eliminación de una imagen:

```
docker rmi -f nginx
```

Elimina la imagen "nginx" del sistema local, incluso si está siendo utilizada por contenedores.

Ayuda sobre el comando `docker rmi`:

```
docker rmi --help
```

Muestra información y opciones disponibles para el comando `docker rmi`.

Buscar imágenes de un usuario específico:

```
docker search tuUsuario
```

Busca imágenes relacionadas con el usuario "tuUsuario" en Docker Hub.

Descargar una imagen de un repositorio personal:

```
docker pull tuUsuario/intranet
```

Descarga la imagen "intranet" del repositorio del usuario "tuUsuario".

Aquí tienes la documentación para cada uno de los comandos relacionados con contenedores, con el mismo formato que utilizaste:

2. Trabajar con contenedores

2.1. Uso básico de contenedores

Ayuda sobre `docker run`:

```
docker run --help
```

Muestra las opciones disponibles y detalles sobre cómo utilizar el comando `docker run`.

Ayuda sobre `docker create`:

```
docker create --help
```

Muestra las opciones disponibles y detalles sobre cómo utilizar el comando `docker create`.

Comando	Crea contenedor	Inicia contenedor automáticamente
`docker create`	Sí	No
`docker run`	Sí	Sí

Lanzar un contenedor de forma interactiva en segundo plano:

```
docker run -dti httpd
```

Inicia un contenedor usando la imagen `httpd` en segundo plano (`-d`) y con un terminal interactivo (`-ti`).

Lanzar un contenedor con nombre y eliminación automática:

```
docker run -dti --rm --name web1 httpd
```

Inicia un contenedor llamado `web1` en segundo plano con un terminal interactivo. Se elimina automáticamente (`--rm`) cuando se detiene.

Lanzar un contenedor y ejecutar un comando específico:

```
docker run -dti --name web-2 httpd echo hola
```

Inicia un contenedor llamado `web-2` en segundo plano y ejecuta el comando `echo hola` dentro del contenedor.

Crear un contenedor sin iniciarlo:

```
docker create --rm --name web-create httpd
```

Crea un contenedor llamado `web-create` con la imagen `httpd`, pero no lo inicia.

Iniciar un contenedor previamente creado:

```
docker start web-create
```

Inicia el contenedor `web-create` que fue creado previamente con `docker create`.

Lanzar un contenedor con el comando completo:

```
docker container run -dti --rm --name web3 httpd
```

Inicia un contenedor llamado `web3` con eliminación automática, en segundo plano, usando la imagen `httpd`.

Lanzar un contenedor `nginx`:

```
docker run -dti --name web2 nginx
```

Inicia un contenedor llamado `web2` con la imagen `nginx`.

```
docker run -dti nginx
```

Inicia un contenedor sin nombre usando la imagen `nginx`.

Lanzar un contenedor con variables de entorno:

```
docker run -dti --name bd2 -e MYSQL_ROOT_PASSWORD=000000 mysql
```

Inicia un contenedor llamado `bd2` con la imagen `mysql`, configurando la contraseña del usuario root mediante la variable de entorno `MYSQL_ROOT_PASSWORD`.

Ejemplo de un contenedor fallido:

```
docker run -dti --name bd3 mysql
```

No funcionará porque falta configurar la variable de entorno `MYSQL_ROOT_PASSWORD` requerida por la imagen `mysql`.

Ejecutar un comando en un contenedor y adjuntarlo:

```
docker run -d --name topdemo ubuntu /usr/bin/top -b
docker attach topdemo
```

Inicia un contenedor llamado `topdemo` con la imagen `ubuntu` y ejecuta el comando `/usr/bin/top -b`. Luego, conecta el terminal al contenedor con `docker attach`.

Ejecutar un comando dentro de un contenedor en ejecución:

```
docker exec -ti bd2 /bin/bash
```

Inicia una sesión interactiva de Bash en el contenedor `bd2`.

Iniciar sesión en MySQL dentro del contenedor:

```
mysql -u root -p
```

Ejecutado desde dentro del contenedor para acceder al servidor MySQL como usuario root.

Inspeccionar un contenedor:

```
docker inspect web1
```

Muestra información detallada en formato JSON sobre el contenedor `web1`, incluyendo configuraciones, estado, red, etc.

2.2. Listar contenedores:

Ver todos los contenedores (incluyendo los detenidos):

```
docker ps -a
```

Ver el último contenedor lanzado:

```
docker ps -l
```

Ayuda sobre `docker container ls`:

```
docker container ls --help
```

Muestra opciones y detalles sobre el comando para listar contenedores.

Ver todos los contenedores con el nuevo comando:

```
docker container ls --all
```

2.3. Interactuar con contenedores:

Inspeccionar un contenedor:

```
docker inspect web1
```

Ejecutar un comando dentro de un contenedor:

```
docker exec -ti web1 /bin/bash
```

Detener un contenedor:

```
docker stop nombre-contenedor
```

Iniciar un contenedor detenido:

```
docker start nombre-contenedor
```

Eliminar un contenedor:

```
docker rm nombre-contenedor
```

Forzar la eliminación de un contenedor:

```
docker rm -f nombre-contenedor
```

Ver los logs de un contenedor:

```
docker logs nombre-contenedor
```

Ayuda sobre el comando `docker logs`:

```
docker logs --help
```

Seguir los logs en tiempo real:

```
docker logs -f nombre-contenedor
```

Ver logs con marcas de tiempo:

```
docker logs -t nombre-contenedor
```

Ver los procesos en un contenedor:

```
docker top contenedor1
```

Reiniciar un contenedor:

```
docker restart nombre-contenedor
```

Pausar un contenedor:

```
docker pause nombre-contenedor
```

Reanudar un contenedor pausado:

```
docker unpause nombre-contenedor
```

Salir de una sesión de contenedor:

```
exit
```

Inspeccionar de nuevo un contenedor:

```
docker inspect web1
```

2.4. Modificar configuración y ver estadísticas de contenedores

Ayuda sobre `docker update`:

```
docker update --help
```

Muestra las opciones disponibles para el comando `docker update`, que permite modificar en caliente la configuración de uno o más contenedores.

Modificar la memoria y el swap de un contenedor:

```
docker update -m 512m --memory-swap -1 web1
```

Ajusta la configuración del contenedor `web1` para:

- Limitar la memoria a **512 MB** (`-m 512m`).
- Deshabilitar el límite de swap (`--memory-swap -1`).

Ver estadísticas en tiempo real:

```
docker stats
```

Muestra estadísticas en tiempo real de los recursos utilizados por los contenedores en ejecución, como:

- Uso de CPU.
- Memoria.
- Red y disco.

2.5. Políticas de reinicio en contenedores

Política `--restart`:

Opción	Descripción
`off`	No reinicia el contenedor automáticamente (por defecto).
`on-failure`	Reinicia el contenedor solo si termina con un error.

Opción	Descripción
`always`	Siempre reinicia el contenedor si está detenido, incluso manualmente.
`unless-stopped`	Reinicia el contenedor si no fue detenido manualmente con `docker stop`.

Ejemplo de reinicio automático con `--restart`:

```
docker run -dtiP --name web5 --restart always nginx
```

Inicia un contenedor llamado `web5` con la imagen `nginx` y lo configura para reiniciarse automáticamente en cualquier caso.

Especificar un puerto con reinicio condicional:

```
docker run -dti -p 8084:80 --name web1 --restart unless-stopped httpd
```

Inicia un contenedor llamado `web1` con la imagen `httpd`, mapeando el puerto **8084** del host al puerto **80** del contenedor. El contenedor se reinicia automáticamente salvo que sea detenido manualmente.

2.6. Otras opciones comunes para contenedores

Eliminar un contenedor al detenerse:

```
--rm
```

Elimina el contenedor automáticamente cuando se detiene.

Cambiar el directorio de trabajo:

```
-w <directorio>
```

Especifica el directorio de trabajo dentro del contenedor.

Especificar el usuario para ejecutar comandos:

```
-u <usuario>
```

Define el usuario que ejecutará los comandos dentro del contenedor.

2.7. Ver estadísticas y modificar recursos en tiempo real

Ver estadísticas de los contenedores:

```
docker stats
```

Muestra métricas en vivo para todos los contenedores en ejecución.

Modificar los recursos de un contenedor:

```
docker update -m 200M --memory-swap -1 web5
```

Ajusta la memoria del contenedor `web5` a **200 MB**, desactivando el límite de swap.

2.8. Configuración de puertos y redirección de tráfico

Ejemplo de redirección de puertos:

```
docker run -dtiP --name web11 httpd
```

Inicia un contenedor llamado `web11` con la imagen `httpd` y selecciona un puerto aleatorio libre en el servidor (`-P`).

Especificar un puerto manualmente:

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

Inicia un contenedor llamado `web2` con la imagen `nginx` y mapea el puerto **8085** del host al puerto **80** del contenedor.

Configuración de bases de datos con redirección:

```
docker run -dti -p 3307:3306 --name bd2 -e MYSQL_ROOT_PASSWORD=000000 mysql
```

Inicia un contenedor llamado `bd2` con la imagen `mysql`, redirigiendo el puerto **3307** del host al puerto **3306** del contenedor y configurando la contraseña del usuario root.

Ver las reglas de iptables para Docker:

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

Muestra las reglas de NAT en iptables configuradas por Docker para redirigir el tráfico.

Consultar puertos expuestos por un contenedor:

```
docker port web11
```

Muestra los puertos abiertos en el contenedor `web11` y sus equivalentes en el host.

2.9. Configurar puertos en una imagen Docker

Exponer un rango de puertos en un Dockerfile:

```
EXPOSE 7000-8000
```

Declara que la imagen Docker puede usar los puertos **7000** a **8000**.

Especificar un rango de puertos al ejecutar un contenedor:

```
docker run --expose=7000-8000
```

Expone el rango de puertos sin asociarlos a ningún puerto del host.

```
docker run -p 7000-8000:7000-8000 imagen-docker
```

Mapea el rango de puertos **7000-8000** del host al contenedor.