Preguntas Claves de Docker (@LeiferMendez)

Video: https://www.youtube.com/@LeiferMendez/videos

¿Cómo crearías una red de Docker y por qué sería útil en el contexto de contenedores?

Respuesta:

Para crear una red de Docker, se utiliza el comando `docker network create`. Una red de Docker es útil porque permite la comunicación entre contenedores, facilitando la conectividad y el intercambio de datos entre aplicaciones en diferentes contenedores.

Ejemplo de Código:

```
docker network create mi_red_docker
```

¿Cómo puedes verificar qué imágenes de Docker tienes en tu sistema y cómo puedes filtrarlas?

Respuesta:

Puedes usar el comando `docker images` para ver las imágenes de Docker en tu sistema. Para filtrarlas, puedes usar `grep` junto con el nombre que estás buscando. Por ejemplo, `docker images | grep api-ts` mostrará solo las imágenes que contienen "api-ts" en su nombre.

Ejemplo de Código:

```
docker images | grep api-ts
```

Explique los elementos clave en el comando `docker build --no-cache -f Dockerfile -t api-ts-one-state ..`

Respuesta:

- `--no-cache` : Evita el uso de la caché durante la construcción.
- `-f Dockerfile` : Especifica el nombre del archivo Dockerfile.
- `-t api-ts-one-state` : Etiqueta la imagen resultante como "api-ts-one-state".
- `.` : Indica que se debe buscar el Dockerfile en el directorio actual.

Ejemplo de Código:

```
docker build --no-cache -f Dockerfile -t api-ts-one-state .
```

¿Cómo ejecutarías un contenedor de Docker y expondrías un puerto para acceder a la aplicación?

Respuesta:

Se puede usar el comando `docker run` con la opción `-p` para mapear puertos. Por ejemplo, `docker run -p 3000:3000/tcp` ejecutará un contenedor y mapeará el puerto 3000 del host al puerto 3000 del contenedor.

Ejemplo de Código:

```
docker run -p 3000:3000/tcp nombre_de_la_imagen
```

¿Por qué es importante usar variables de entorno en el comando `docker run` para la configuración de una aplicación?

Respuesta:

Las variables de entorno en el comando `docker run` permiten configurar dinámicamente aspectos de la aplicación, como las conexiones a bases de datos. Esto hace que la aplicación sea más flexible y fácil de configurar sin necesidad de modificar directamente la configuración del contenedor.

Ejemplo de Código:

```
docker run -e 'NOMBRE_VARIABLE=valor' nombre_de_la_imagen
```

¿Cómo puedes ver la lista de contenedores en ejecución en tu sistema?

Respuesta:

Puedes usar el comando `docker ps` para listar los contenedores en ejecución. Este comando muestra información como el ID del contenedor, el nombre, el estado, los puertos mapeados y más.

Ejemplo de Código:

```
docker ps
```

¿Cuál es el comando para detener un contenedor en ejecución?

Respuesta:

El comando para detener un contenedor es `docker stop` seguido del nombre o ID del contenedor que deseas detener.

Ejemplo de Código:

```
docker stop nombre_del_contenedor_o_ID
```

¿Cómo puedes eliminar un contenedor que ya ha sido detenido?

Respuesta:

Puedes utilizar el comando `docker rm` seguido del nombre o ID del contenedor que quieres eliminar. Es importante destacar que el contenedor debe estar detenido para ser eliminado.

Ejemplo de Código:

```
docker rm nombre_del_contenedor_o_ID
```

¿Cómo puedes ejecutar un contenedor Docker y qué significan los diferentes parámetros en el comando?

Respuesta:

Puedes usar el comando `docker run` para ejecutar un contenedor. Los parámetros significan:

- `--rm` : Elimina el contenedor automáticamente después de detenerlo.
- `-it` : Habilita la interactividad para acceder a la terminal del contenedor.
- `-p 3000:3000/tcp` : Mapea el puerto 3000 del host al puerto 3000 del contenedor.
- `--network mi_red_docker` : Conecta el contenedor a la red llamada `mi_red_docker`.
- `-e 'DB_URI=...'` : Configura una variable de entorno para la conexión a MongoDB.
- `-e 'NAME_APP=app2'` : Configura una variable de entorno con el nombre de la aplicación.
- `api-ts-one-state:latest` : Especifica la imagen a utilizar y su etiqueta.

Ejemplo de Código:

```
docker run --rm -it -p 3000:3000/tcp --network mi_red_docker -e 'DB_URI=...' -e 'NAME_APP=app2' api-ts-one-state:latest
```

Explica los diferentes parámetros en el comando `docker run --rm -it -p 8080:80/tcp --network network-my-app -v "ruta/nginx/conf.d:/etc/nginx/conf.d" --name mi_contenedor nginx-one:latest .`

Respuesta:

- `--rm` : Elimina automáticamente el contenedor después de detenerlo.
- `-it` : Habilita la interactividad para acceder a la terminal del contenedor.
- `-p 8080:80/tcp` : Mapea el puerto 8080 del host al puerto 80 del contenedor.
- `--network network-my-app` : Conecta el contenedor a la red llamada `network-my-app` .
- `-v "ruta/nginx/conf.d:/etc/nginx/conf.d"` : Monta un volumen para utilizar la configuración de Nginx desde el directorio local `ruta/nginx/conf.d` en el contenedor en la ruta `/etc/nginx/conf.d` .
- `--name mi_contenedor` : Asigna un nombre al contenedor.
- `nginx-one:latest` : Especifica la imagen a utilizar y su etiqueta.

Ejemplo de Código:

```
docker run --rm -it -p 8080:80/tcp \
  --network network-my-app \
  -v "rutat/nginx/conf.d:/etc/nginx/conf.d" \
  --name mi_contenedor \
  nginx-one:latest
```