Curso Docker

Arturo Silvelo

Try New Roads

Tabla de Contenido

Introducción

Contenedores

Redes

Volúmenes

Imágenes

¿Qué son las Imágenes de Docker?

- Las imágenes de Docker son plantillas de solo lectura utilizadas para crear contenedores.
- Contienen el sistema de archivos y todas las dependencias necesarias para ejecutar una aplicación.
- Se pueden compartir y distribuir fácilmente a través de registros como Docker Hub.
- Para crear imágenes personalizadas, se utiliza un `Dockerfile`, que es un archivo de texto con instrucciones para construir la imagen.

¿Qué es un Dockerfile?

- Un Dockerfile es un archivo de texto que define los pasos necesarios para construir una imagen de Docker, como instalar dependencias, copiar archivos, y configurar el entorno.
- Se utiliza con el comando `docker build` para generar una imagen personalizada y reproducible.
- Un Dockerfile se construye usando una serie de instrucciones o palabras reservadas, cada una de las cuales genera una nueva capa en la imagen.
- Las capas son almacenadas de manera eficiente, y Docker reutiliza las capas que no han cambiado, lo que acelera las construcciones subsecuentes.
- La capa final de la imagen es el contenedor que se ejecutará, proporcionando el entorno listo para ejecutar la aplicación.

Instrucciones Comunes en Dockerfile

- `FROM`: Define la imagen base a partir de la cual se construye la nueva imagen.

  ```
  FROM nginx:latest
  ```

- `WORKDIR`: Cambia el directorio donde se ejecutarán los siguientes comandos.
- `ENV`: Define las variables de entorno.
- `RUN`: Ejecuta un comando dentro de la imagen durante su construcción.

  ```
  RUN apt-get update && apt-get install -y
      iputils-ping nano
  ```

- `VOLUME`: Crea un punto de montaje para volúmenes.

  ```
  VOLUME ["/etc/nginx/conf.d"]
  ```

Instrucciones Comunes en Dockerfile

- **COPY/ADD**: Copia archivos o directorios del host al contenedor.

```
COPY ./index.html /usr/share/nginx/html/index.
    html
```

- **EXPOSE**: Informa a Docker que el contenedor escucha en un puerto específico. No mapea automáticamente el puerto al host.

```
EXPOSE 80
```

- **CMD**: Especifica el comando que se ejecutará cuando se inicie el contenedor. Es esencial para definir el comportamiento del contenedor.

```
CMD ["nginx", "-g", "daemon␣off;"]
```

[1]

[1] Más información: `https://docs.docker.com/reference/dockerfile/`

El archivo `.dockerignore`

- El archivo `.dockerignore` especifica qué archivos o directorios no deben ser copiados a la imagen de Docker.
- Similar al `.gitignore`, ayuda a evitar archivos innecesarios en la imagen.
- Mejora la eficiencia al reducir el tamaño de la imagen y acelera el proceso de construcción.
- Protege la seguridad al evitar incluir archivos sensibles en la imagen.

Ejercicio 1: Crear una Imagen Docker para el Entorno de Desarrollo de una Aplicación NestJS

Crear una imagen Docker para la aplicación NestJS con el entorno de desarrollo configurado.

1. Crear un archivo `Dockerfile` en la raíz de tu proyecto.
2. Usar `node:20` como imagen base.
3. Establecer un directorio de trabajo (`/usr/local/app`)
4. Copiar fichero de dependencias (`package.json`)
5. Definir variables (USE_MEMORY_DB, DATABASE_URI, PORT) y configurar puerto (3000).
6. Instalar las dependencias (`npm install --legacy-peer-deps`)
7. Copiar el contenido.
8. Ejecutar el comando (`npm run start:dev`)

Ejercicio 1: Crear una Imagen Docker para el Entorno de Desarrollo de una Aplicación NestJS

9. Genera la imagen con un nombre.
10. Inicia un contenedor con la imagen creada. (Comprobar que funciona y es accesible)
11. Inicia otro contenedor con un bind mount del `src`. (Comprobar que las modificaciones realizadas actualizan el backend, `main.ts` modificar textos del config)

Docker Compose