¿Qué es una Internal Developer Platform (IDP) y para que se utiliza ?

Lo que hace es proporcionar un entorno estandarizado donde los desarrolladores pueden construir, probar, implementar y gestionar aplicaciones de manera más eficiente.

El principal objetivo de una IDP es lograr que el trabajo diario de los equipos de desarrollo sea más eficiente, y fácil de gestionar. Para ello, consolidan y simplifican los elementos del proceso de desarrollo.También facilita la colaboración entre equipos, asegura la gestión eficiente de la infraestructura y la implementación continua de código

Alguna de sus principales características son:

- **Reducción de la carga cognitiva**: Simplificar el trabajo de los desarrolladores para que puedan concentrarse en tareas más importantes.
- **Sin abstracción total**: Mantener visible el contexto y las tecnologías subyacentes para que los desarrolladores no pierdan de vista elementos esenciales.

Esto mejora la eficiencia y la productividad sin sacrificar el entendimiento profundo de las tecnologías empleadas.

DockerFile:

Para crear un Dockerfile,se deben definir los siguientes requisitos:

1. Sistema operativo o base de la imagen.

2. Dependencias de software necesarias (librerías, paquetes, etc.).

3. Scripts de inicio o comandos para construir el entorno.

4. Variables de entorno requeridas.

La imagen base depende del entorno y las dependencias necesarias para la aplicación. Las más comunes son:

- Para una aplicación Node.js: `node:14` o la versión más reciente.

- Para una aplicación Python: `python:3.9`.

- Para una aplicación basada en Linux: `ubuntu:20.04` o la versión estable.

Los puertos a exponer dependen del servicio que esté en ejecución dentro del contenedor. Los ejemplos más comunes son:

- Aplicación web en Node.js: generalmente el puerto `3000`.

- Servidor web Apache o Nginx: usualmente el puerto `80` para HTTP y `443` para HTTPS.

- Base de datos MySQL: generalmente el puerto `3306`.

El archivo de configuración puede variar según la aplicación.

Ejemplos comunes:

- Un archivo `config.json` para configuraciones de la aplicación.

- Un archivo `.env` para variables de entorno que el contenedor utilizará.

Esto asegura que el contenedor tenga todas las configuraciones necesarias al momento de iniciarse.

Code Server

Servicio, puertos y volúmenes consideran necesarios

```yaml
version: '3'
services:
  code-server:
    image: codercom/code-server:latest
    container_name: code-server
```

Esta es la definición de un servicio para code server en el archivo `docker-compose.yml`

Version : Define la versión de docker-compose. Generalmente se usa la versión 3.

Services : Aquí se definen los servicios que se quieren ejecutar. Cada servicio tendrá sus propias configuraciones como imagen, puertos, volúmenes, etc.

image: codercom/code-server:latest : Define la imagen base del servicio.

container_name: code-server : Asigna un nombre al contenedor.

puertos:

Necesitarás mapear el puerto del contenedor al puerto del host para acceder al code server.

```
ports:
  - "8080:8080"
```

Mapea el puerto 8080 del contenedor al puerto 8080 del host.

volúmenes:

Para persistir los datos entre reinicios del contenedor, es importante montar volúmenes. Aquí un ejemplo:

```
volumes:
  - ./path/to/code-server-data:/home/coder/project
```

Monta un volumen desde el host al contenedor.

Código completo con servicios , puertos y volúmenes:

```
version: '3'

services:
  code-server:
    image: codercom/code-server:latest
    container_name: code-server
    ports:
      - "8080:8080"
    volumes:
      - ./path/to/code-server-data:/home/coder/project
```

1) Levantamos SQL Server utilizando Docker:

PS C:\Users\santi> docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=nico123!' -p 1433:1433 --name sqlserver -d mcr.microsoft.com/mssql/server:2019-latest >>

Unable to find image 'mcr.microsoft.com/mssql/server:2019-latest' locally 2019-latest: Pulling from mssql/server a6fcb2138200: Download complete 54bb8e7d5056: Download complete b43df23e6f02: Download complete

2) Verificamos que el contenedor esté en ejecución:

docker ps

>> CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 392a5f9287ad mcr.microsoft.com/mssql/server:2019-latest "/opt/mssql/bin/perm…" 14 seconds ago Up 7 seconds 0.0.0.0:1433->1433/tcp sqlserver CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 392a5f9287ad mcr.microsoft.com/mssql/server:2019-latest "/opt/mssql/bin/perm…" 14 seconds ago Up 7 seconds 0.0.0.0:1433->1433/tcp sqlserver PS C:\Users\santi>

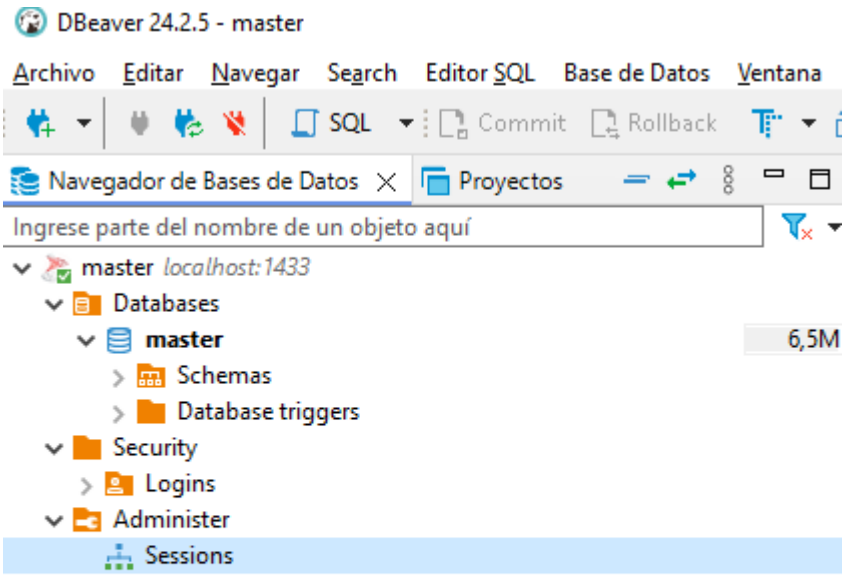
3)Luego creamos la nueva base de datos en DBeaver y establecimos la coneccion:

Host: `localhost`

Puerto: `1433`

Nombre de usuario: `sa`

Contraseña: `nico123!`



4) Creamos el archivo docker-compose.yaml para gestionar el contenedor de SQL Server de manera más eficiente :

```
version: '3.8'

services:

  sqlserver:

    image: mcr.microsoft.com/mssql/server:2019-latest

    container_name: sqlserver

    environment:

      - ACCEPT_EULA=Y

      - SA_PASSWORD=nico123!

    ports:

      - "1433:1433"

    volumes:

      - sqlserverdata:/var/opt/mssql

volumes:

  sqlserverdata:
```

5) Aquí ejecutamos “docker-compose up -d” para levantar el contenedor utilizando Docker Compose.
Esto levanta el contenedor de SQL Server y gestiona los volúmenes de datos automáticamente.

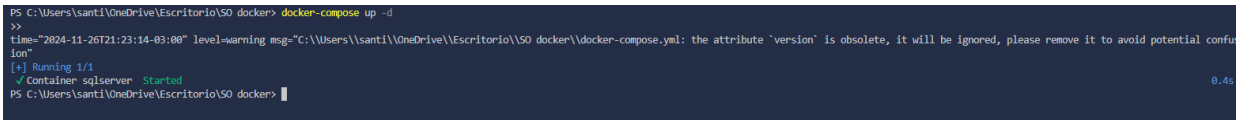


Foto del docker desktop