Taller 0 - Docker - BDD

1. Introducción a Docker

¿Por qué Docker?

- **Motivación:**
  Docker permite empaquetar aplicaciones y sus dependencias en contenedores aislados, lo que facilita el despliegue, la portabilidad y la escalabilidad.
- **Beneficios:**
  - Consistencia en distintos entornos (desarrollo, prueba y producción).
  - Aislamiento de procesos y recursos.
  - Optimización en el uso de recursos del sistema.

¿Qué es Docker?

- **Definición Básica:**
  Docker es una plataforma de virtualización a nivel de sistema operativo que utiliza contenedores para ejecutar aplicaciones de forma aislada.
- **Elementos Clave:**
  - **Dockerfile:** Archivo de texto con las instrucciones para construir una imagen Docker.
  - **Docker Compose:** Permite definir y gestionar múltiples contenedores a través de un archivo en formato YAML.

---

2. Vocabulario Fundamental

- **Imagen:**
  Es un paquete inmutable que contiene todo lo necesario para ejecutar una aplicación (código, dependencias, configuración y sistema operativo base).
- **Contenedor:**
  Es una instancia en ejecución de una imagen. Funciona de forma aislada y utiliza los recursos definidos.

- **Volumen:**
  Permite persistir y compartir datos entre contenedores o entre el host y el contenedor.
- **Network:**
  Es la red virtual que permite la comunicación entre contenedores.
- **Servicio:**
  Es la definición de una aplicación o componente que se ejecuta en uno o varios contenedores.
- **YAML:**
  Formato de serialización de datos utilizado para definir la configuración en Docker Compose.
- **Docker CLI:**
  Es la interfaz de línea de comandos para interactuar con Docker.
- **Registro:**
  Repositorio de imágenes Docker (ej. Docker Hub).

---

3. Instalación y Recursos

- **Instalación de Docker:**
  Se recomienda seguir las guías oficiales (por ejemplo, para Ubuntu) y utilizar tutoriales en línea (como el video sugerido en el taller).
- **Links Útiles:**
  - Docker Hub
  - Docker Docs
  - Docker CLI Cheat Sheet
  - Docker Compose CLI Cheat Sheet
  - Docker Engine Installation (Ubuntu)
  - Tutorial instalación Docker Engine (YouTube FIUBA)
  - Docker Taller (YouTube FIUBA)
  - Página de la cátedra: Base de Datos FIUBA

---

4. Comandos Básicos de Docker

- **docker ps (-a):**
  Lista los contenedores en ejecución y, con la opción `a`, también los detenidos.
- **docker exec (-it):**
  Permite ejecutar comandos dentro de un contenedor en ejecución, por ejemplo, para abrir una sesión interactiva.
- **docker compose up (—build, -d):**
  Construye y levanta los contenedores definidos en un archivo docker-compose.yaml. La opción `d` lo hace en modo "desconectado".
- **docker compose down (-v):**
  Detiene y elimina los contenedores, redes y volúmenes creados.
- **docker logs (-f):**
  Muestra los logs de un contenedor en tiempo real.
- **docker volume ls:**
  Lista los volúmenes creados en Docker.
- **docker cp:**
  Copia archivos entre el host y el contenedor.

---

5. Ejemplo de Archivo docker-compose.yaml

El siguiente archivo define un servicio para una base de datos PostgreSQL:

```
services:
  db:
    image: postgres:17.4 # Importante: Usuarios ARM, verifique compatibilid
ad
    ports:
      - "5432:5432"
    environment:
      POSTGRES_USER: usuario
      POSTGRES_PASSWORD: 123
      POSTGRES_DB: bdd_db
    volumes:
      - db_data:/var/lib/postgresql/data
```

```
volumes:
  db_data:
```

Explicación:

- **services:**
  Define los servicios a ejecutar. Aquí se configura el servicio `db`.
- **image:**
  Se utiliza la imagen `postgres:17.4` para el contenedor de la base de datos.
- **ports:**
  Se mapea el puerto 5432 del contenedor al 5432 del host, permitiendo el acceso externo.
- **environment:**
  Variables de entorno para configurar el usuario, contraseña y nombre de la base de datos.
- **volumes:**
  Se asigna un volumen persistente (`db_data`) para almacenar los datos de PostgreSQL y conservarlos entre reinicios.

> Nota: El comando sugerido para acceder a la base de datos es:
>
> `docker exec -it <container_name> psql -U usuario bdd_db`
