Docker

Docker és una plataforma que permet empaquetar aplicacions i les seves dependències en contenidors lleugers i aïllats. Aquests contenidors asseguren que l'aplicació funcioni de manera consistent en qualsevol entorn, des del desenvolupament fins a la producció.

Docker Desktop

Docker Desktop és una aplicació que facilita la instal·lació i gestió de Docker en entorns locals. Proporciona una interfície gràfica d'usuari que simplifica la creació, execució i administració de contenidors, cosa que resulta molt útil per a desenvolupadors que volen treballar amb aplicacions empaquetades en contenidors sense haver de configurar manualment Docker des de la línia de comandes.

Accedim a la pàgina oficial de Docker per descarregar-lo:
https://www.docker.com/products/docker-desktop/

Des de la interfície de línia de comandes integrada a Docker Desktop o des de PowerShell es poden executar ordres per gestionar contenidors i altres recursos.
Pots saber la versió de Docker usant: `docker --version`.

Comandes bàsiques

- **docker images:**
  Llista totes les imatges disponibles localment. Una imatge és una plantilla per crear contenidors, empaquetant aplicacions i les seves dependències.

- **docker pull**
  Descarrega una imatge des del repositori, per defecte, Docker Hub. Exemple: `docker pull ubuntu`.

- **docker rmi:**
  Elimina una imatge local. Exemple: `docker rmi ubuntu`.

- **docker create:**
  Crea un contenidor a partir d'una imatge.
  Exemple: `docker create --name nom_contenidor ubuntu`

- **docker start:**
  Inicia un contenidor aturat. Exemple: `docker start nom_contenidor`

- **docker stop:**
  Atura un contenidor en execució. Exemple: `docker stop nom_contenidor`.

- **docker run:**
  Descarrega la imatge i crea i inicia un contenidor a partir d'aquesta imatge.
  Exemple: `docker run --name nom_contenidor ubuntu`.

- **docker rm:**
  Elimina un contenidor aturat. Exemple: `docker rm nom_contenidor`.

- **docker ps:**
  Mostra els contenidors en execució. Es poden veure tots els contenidors, incloent-hi els aturats, afegint l'opció -a: `docker ps -a`.

- **docker logs:**
  Mostra els logs d'un contenidor, permetent veure la sortida i esdeveniments.
  Exemple: `docker logs nom_contenidor`.

- **docker exec:**
  Executa comandes dins d'un contenidor en execució.
  Exemple: `docker exec -it nom_contenidor /bin/bash`. Obre una sessió interactiva de bash dins del contenidor.

Dockerfile

El fitxer Dockerfile conté instruccions per construir una imatge Docker personalitzada de forma automatitzada. Dins d'aquest fitxer, s'especifica la imatge base, es defineixen les operacions per instal·lar dependències, copiar fitxers i configurar l'entorn d'execució del contenidor.

Exemple, creem una imatge personalitzada de Linux Ubuntu:

```
# Selecciona la imatge base
FROM ubuntu:20.04

# Actualitza el sistema i instal·la curl
RUN apt-get update && apt-get install -y curl

# Defineix el directori de treball dins del contenidor
WORKDIR /app

# Copia els fitxers locals dins el contenidor, al directori de treball
COPY . .

# Comanda per defecte en executar el contenidor
CMD ["echo", "Hola, món!"]
```

- **docker build:**
  Construeix una imatge a partir d'un Dockerfile.
  Exemple: `docker build -t nom_imatge .`.
  El punt indica que el Dockerfile es troba al directori actual.

- **docker history:**
  Mostra l'historial de construcció d'una imatge. Exemple: `docker history nom-imatge`.

Xarxes Docker

Docker utilitza xarxes per permetre la comunicació entre contenidors i amb l'amfitrió.

- **docker network ls:**
  Llista les xarxes disponibles. Aquesta comanda mostra totes les xarxes personalitzades i les creades per Docker.

- **docker network create:**
  Crea una xarxa personalitzada que pot ser usada per connectar diversos contenidors, per defecte és de tipus bridge. Exemple: `docker network create nom-xarxa`.

- **docker network inspect:**
  Permet veure la configuració i els contenidors connectats a la xarxa especificada.
  Exemple: `docker network inspect nom-xarxa`.

Volums

Els volums a Docker són mecanismes per persistir dades fora del cicle de vida d'un contenidor, permetent que la informació es mantingui encara que el contenidor es reinicii o s'elimini.

- **docker volume ls:**
  Mostra tots els volums existents.

- **docker volume create:**
  Crea un volum. Exemple: `docker volume create nom-volum`.

- **docker volume inspect:**
  Mostra informació detallada sobre un volum, com ara la ubicació física a l'amfitrió.
  Exemple: `docker volume inspect nom-volum`.

Docker Compose

Docker Compose és una eina que permet definir i executar aplicacions amb múltiples contenidors de manera senzilla. Es basa en un fitxer de configuració anomenat **docker-compose.yml**, en el qual pots especificar els diferents serveis, xarxes i volums que conformen la teva aplicació.

Exemple, definim dos serveis: un servidor web amb Nginx i un servei de memòria cau amb Redis:

```
services:
  web:
    image: nginx:latest
    ports:
      - "80:80"
  redis:
    image: redis:latest
```

Mapeja els ports, indica que el port 80 de l'amfitrió (primer port) s'enllaça amb el port 80 del contenidor (segon port).

- **docker compose up**
  Inicia els serveis definits al fitxer docker-compose.yml. Si els contenidors no existeixen, els crea a partir de les imatges especificades. Usant el modificador --build podem forçar la reconstrucció de les imatges abans de crear els contenidors, assegurant així que es prenguin en compte els canvis recents en el codi i en el Dockerfile: `docker-compose up --build`.

- **docker compose down**
  Atura tots els serveis i elimina els contenidors, xarxes i altres recursos creats per Docker Compose, deixant net l'entorn.

ACTIVITATS

En aquestes activitats aprendrem a desplegar una aplicació Node.js amb Express i una base de dades MongoDB utilitzant Docker i Docker Compose.
L'aplicació a desplegar la pots clonar des de:
https://github.com/xbaubes/DesenvolupamentWeb/tree/main/Backend/Node.js/Express.js/mongoose/VolcanoApp
No has d'instal·lar cap dependència de l'aplicació, totes s'instal·laran únicament als contenidors de Docker.

Sempre que creis un element, llista'l i fes-ne una captura.

*Substitueix ABC per sigles que t'identifiquin.

1. **Xarxa per comunicar contenidors**

    - Crea una xarxa bridge que permeti que els nostres contenidors es comuniquin entre ells. L'anomenem «volcano-networkABC». Mostra-la.

2. **Contenidor MongoDB**

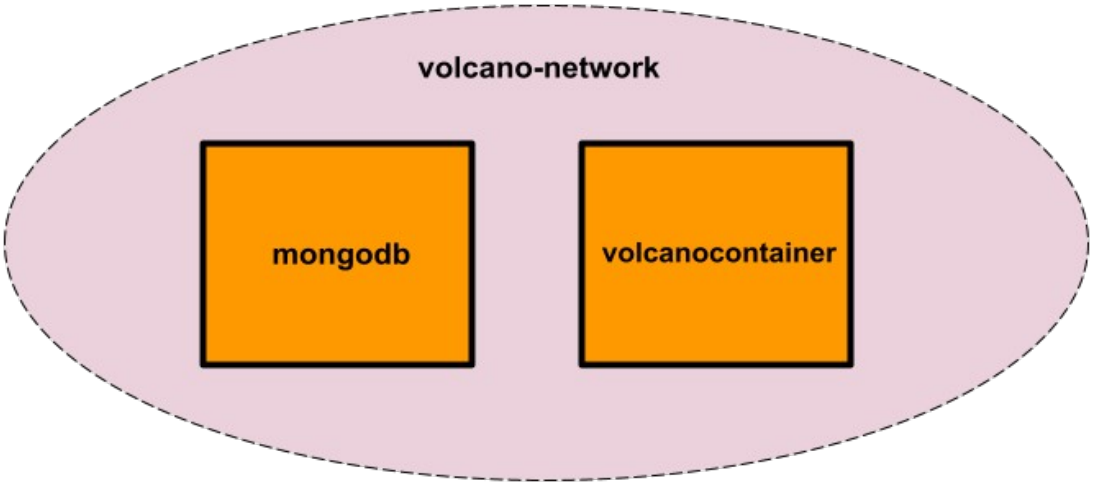
    - Descarrega la imatge oficial de MongoDB des de Docker Hub. Mostra-la.
    - Crea el contenidor de MongoDB, l'anomenem «mongodbABC», configura les següents característiques:
        - Fa servir la xarxa «volcano-networkABC», creada anteriorment.
        - Utilitza el port predeterminat de MongoDB: 27017.
        - Requereix autenticació per connectar-se a la base de dades, utilitza les credencials que trobaràs al codi de l'aplicació.
        - Mostra els logs de creació.
    - Inicia el contenidor i llista els contenidors en funcionament.
    - Accedeix a la consola de MongoDB dins del contenidor com a admin i llista'n les bases de dades.

3. **Imatge per l'aplicació Node.js amb Dockerfile**

    - Defineix un fitxer Dockerfile per Node.js amb les següents característiques:
        - Usa l'última versió de Node.js.
        - El directori de treball dins del contenidor ha de ser: «/home/app».
        - Utilitza el port definit al fitxer .env.
        - Copia els fitxers package.json i package-lock.json dins el contenidor per poder aplicar la comanda «npm install» i crear l'entorn de dependències que requereix l'aplicació.
        - Copia la resta de fitxers.
        - Inicia el servidor amb la comanda «npm start» definida al package.json.
    - Crea una imatge de nom «volcanoappdockerABC» a partir del Dockerfile generat. Valida que s'ha creat correctament.
    - OPCIONAL: Crea un fitxer Dockerfile per MongoDB.

4. Contenidor per l’aplicació

- Modifica la connexió amb la base de dades definida al fitxer .env per tal que el nostre contenidor Express s’hi pugui connectar. S’ha de canviar «localhost» pel nom del contenidor de mongoDB: «mongodbABC».
- Creació del contenidor per l’aplicació Express a partir de la imatge creada, assigna-li el nom «volcanocontainerABP», configura les següents característiques:
  - Utilitza el port definit al fitxer .env.
  - Assegura’t de guardar el contingut del fitxer .env, ja que no podrem incorporar-lo al contenidor, doncs està registrat al fitxer .dockerignore.
  - Fa servir la mateixa xarxa que el contenidor de MongoDB per tal de poder-se comunicar entre ells.
  - Mostra els logs de creació.
- Inicia el contenidor i documenta-ho.
- Testeja l’aplicació accedint-hi des del navegador.
- Mostra les col·leccions guardades a la base de dades creades des de l’aplicació.



5. Automatització amb Docker Compose

- Abans d’executar Docker Compose, elimina els contenidors creats manualment. Els noms dels nous contenidors seran els mateixos.
- Ha de crear el contenidor per Node.js amb l’aplicació i el contenidor per MongoDB units per la xarxa creada anteriorment:
  - Utilitza el fitxer Dockerfile creat anteriorment per crear la imatge de Node.js.
  - El contenidor de Node.js requereix que el contenidor de MongoDB estigui en funcionament.
  - Assegura’t de guardar el contingut del fitxer .env i les variables d’entorn, ja que no l’incorporarem al contenidors a l’estar registrat al fitxer .dockerignore.
  - Utilitza volumes per la persistència de les dades de la base de dades, usa una ubicació adequada per MongoDB.
- Inicia els contenidors.
- Prova l’aplicació accedint-hi des del navegador.
- Mostra els logs.

BIBLIOGRAFIA I WEBGRAFIA

«aTopeCode». http://www.atopecode.net/mongodb-con-docker-compose-y-mongo-express/
«dockerdocs». https://docs.docker.com/manuals/
https://docs.docker.com/reference/samples/express/