Explicación do Ficheiro "Cheat-Sheet-Docker_A3.pdf"

Este ficheiro é unha referencia rápida (*cheat-sheet*) para traballar con Docker nun sistema Debian GNU/Linux. A continuación, explicamos os seus contidos principais:

1. Instalación de Docker

Hai dúas opcións para instalar Docker:

- **Opción 1**: Seguir as instrucións oficiais de Docker.
  - **URL de instalación de Docker**
  - **URL para Docker Compose**
- **Opción 2**: Instalar os paquetes `docker.io` e `docker-compose` directamente desde os repositorios de Debian:

  ```
  apt update && apt -y install docker.io && apt -y install docker-compose
  ```

2. Comandos Básicos de Docker

- **Listar imaxes locais:**

  ```
  docker image ls #Mostra as imaxes Docker almacenadas localmente.
  ```

- **Iniciar sesión en Docker Hub:**

  ```
  docker login #Permite iniciar sesión no Docker Hub para subir ou descargar imaxes.
  ```

- **Etiquetar unha imaxe:**

  ```
  docker tag kalia:https ricardofc/repoedu-ccbysa:kalia-https #Asigna unha nova etiqueta a unha imaxe local para preparala para subir ao Docker Hub.
  ```

- **Subir unha imaxe ao Docker Hub:**

  ```
  docker push ricardofc/repoedu-ccbysa:kalia-https #Envía a imaxe etiquetada ao repositorio remoto no Docker Hub.
  ```

- **Descargar unha imaxe:**

  ```
  docker pull ricardofc/repoedu-ccbysa:kalia-https #Descarga unha imaxe específica desde o Docker Hub.
  ```

- **Construír unha imaxe:**

  ```
  docker build -t kalia:https . #Constrúe unha imaxe Docker a partir dun Dockerfile presente no directorio actual.
  ```

- **Executar un contedor:**

  ```
  docker run -p 443:443 --name kalia -dit kalia:https #Executa un contedor en segundo plano (-d), asignando o nome "kalia", mapeando o porto 443 do host ao porto 443 do con
  ```

- **Listar contedores en execución:**

  ```
  docker container ls #Mostra os contedores que están en execución.
  ```

- **Acceder a un contedor en execución:**

  ```
  docker exec -ti kalia /bin/bash #Abre unha sesión interactiva dentro do contedor chamado "kalia".
  ```

3. Explicación do Dockerfile

O Dockerfile é un script que define como construír unha imaxe Docker. Aquí está a explicación liña por liña:

```
FROM kalilinux/kali-rolling:latest
LABEL maintainer="ricardofc "

## Apache2
RUN apt-get update \
    && apt-get -y install apache2 \
    && a2ensite default-ssl \
    && a2enmod ssl

ENTRYPOINT apachectl -D FOREGROUND
```

- **`FROM kalilinux/kali-rolling:latest`**: Define a imaxe base para construír a nova imaxe. Neste caso, usa Kali Linux.
- **`LABEL maintainer="ricardofc "`**: Engade metadatos á imaxe, especificando o mantenedor.
- **`RUN apt-get update ...`**: Actualiza os paquetes do sistema e instala Apache2. Activa o sitio SSL predeterminado e o módulo SSL.
- **`ENTRYPOINT apachectl -D FOREGROUND`**: Define o comando que se executará cando o contedor inicie. Neste caso, inicia Apache en primeiro plano.

4. Explicación do docker-compose.yml

O arquivo docker-compose.yml permite definir e executar aplicacións multi-contedor. Aquí está a explicación liña por liña:

```
version: "3.1"
services:
  https:
    image: kalia:https
    ports:
      - "443:443"
    volumes:
      - ./web:/var/www/html
    networks:
      - frontend
networks:
  frontend:
    driver: bridge
```

- **version: "3.1"**: Especifica a versión da sintaxe de Docker Compose utilizada.
- **services**: Define os servizos (contedores) que forman parte da aplicación.
- **https**: Nome do servizo.
- **image: kalia:https**: Usa a imaxe "kalia:https" para este servizo.
- **ports: - "443:443"**: Mapea o porto 443 do host ao porto 443 do contedor.
- **volumes: - ./web:/var/www/html**: Monta o directorio local "./web" no camiño "/var/www/html" do contedor.
- **networks: - frontend**: Asigna o servizo á rede "frontend".
- **networks/frontend/driver: bridge**: Define unha rede de tipo "bridge" chamada "frontend".

5. Comandos de Docker Compose

- **Subir os servizos:**

  ```
  docker-compose up #Inicia os contedores definidos no arquivo docker-compose.yml.
  ```

- **Parar os servizos:**

  ```
  docker-compose down #Para e elimina os contedores e redes creadas por docker-compose.
  ```

- **Reconstruír os servizos:**

  ```
  docker-compose build #Reconstrúe as imaxes dos servizos definidos.
  ```

docker run e docker exec

- Para interactuar cun contedor, sempre necesitas iniciar un contedor primeiro (con docker run).
- Unha vez que o contedor está en execución, podes usar docker exec para interactuar con el.
- Se queres abrir unha consola directamente, podes facelo usando docker run coa opción -it.

Que é unha rede bridge?

Unha rede **bridge** é o tipo de rede predeterminado en Docker. Permite que os contedores nunha mesma rede se comuniquen entre eles, pero non están directamente expostos á rede externa (por defecto).

Puntos clave:

- **Comunicación entre contedores:**
  - Os contedores conectados á mesma rede *bridge* poden comunicarse uns cos outros usando os seus nomes como dirección DNS.
  - Por exemplo, se tes dous servizos (servizo1 e servizo2) conectados á rede frontend, servizo1 pode acceder a servizo2 simplemente usando o nome servizo2.
- **Illamento:**
  - Os contedores nunha rede *bridge* están illados da rede externa por defecto. Para permitir o acceso externo, debes mapear portos usando a opción ports no docker-compose.yml.
- **Controlador predeterminado:**
  - Se non se especifica un controlador, Docker usa automaticamente o controlador *bridge*.

Por que usar redes personalizadas como frontend?

Usar redes personalizadas (como frontend) ten varias vantaxes:

- **Seguridade:**
  - Podes controlar que contedores están conectados a que redes, limitando a comunicación entre eles.
- **Facilidade de comunicación:**
  - Os contedores conectados á mesma rede poden comunicarse uns cos outros usando os seus nomes como DNS, sen necesidade de configurar IPs manuais.
- **Organización:**
  - Ao agrupar contedores en redes específicas, podes organizar mellor a túa aplicación. Por exemplo:
    - Unha rede para servizos front-end (frontend).
    - Outra rede para servizos back-end (backend).

Ricardo Feijoo Costa