Tercera Práctica: Docker

Descripción

Docker es una plataforma abierta para desarrollar, enviar y ejecutar aplicaciones. Docker permite separar las aplicaciones de su infraestructura para que pueda entregar el software rápidamente. Con Docker, es posible gestionar su infraestructura de la misma manera que gestiona sus aplicaciones. Al aprovechar las metodologías de Docker para enviar, probar y desplegar el código rápidamente, puede reducir significativamente el tiempo que transcurre entre la escritura del código y su ejecución en producción.

Para realizar esta práctica será necesario realizar todos los pasos documentando con comentarios, imágenes lo que estáis haciendo. Se deberá mostrar los resultados al ejecutar los comandos y tendréis que explicar lo que se está haciendo con vuestras propias palabras.

Setup

Si usáis Windows
https://docs.docker.com/desktop/install/windows-install/
Si usáis Linux
https://docs.docker.com/desktop/install/linux-install/

Correr el primer contenedor

Ahora que tienes todo configurado, es hora de ensuciarse las manos. En esta sección, vas a ejecutar un contenedor Alpine Linux (una distribución de linux ligera) en tu sistema y a probar el comando docker run.

```
docker pull alpine
```

El comando pull obtiene la imagen del sistema operativo Alpine del registro Docker y la guarda en nuestro sistema. Puedes usar el comando docker images para ver una lista de todas las imágenes en tu sistema.

```
docker images
```

Muy bien. Ahora vamos a ejecutar un contenedor Docker basado en esta imagen. Para ello vamos a utilizar el comando docker run.

```
docker run alpine ls -l
```

¿Qué ha pasado? Entre bastidores, han pasado muchas cosas. Cuando se llama a ejecutar ese comando se realizan las siguientes tareas:

- El cliente Docker contacta con el demonio Docker
- El demonio Docker comprueba en el almacén local si la imagen (alpine en este caso) está disponible localmente, y si no, la descarga de Docker Store. (Ya que hemos emitido docker pull alpine antes, el paso de descarga no es necesario)
- El demonio Docker crea el contenedor y luego ejecuta un comando en ese contenedor.
- El demonio Docker transmite la salida del comando al cliente Docker

Cuando ejecutas docker run alpine, proporcionas un comando (ls -l), así que Docker inicia el comando especificado y ves el listado.

Intentemos algo más emocionante.

```
docker run alpine echo "hello from alpine"
```

Bien, esto es una salida real. En este caso, el cliente Docker ejecutó el comando **echo** en nuestro contenedor con SO Alpine y luego salió de él. Si te has dado cuenta, todo eso ocurrió bastante rápido. Imagina arrancar una máquina virtual, ejecutar un comando y luego matarla. ¡Ahora sabes por qué dicen que los contenedores son rápidos!

Prueba con otro comando.

```
docker run alpine /bin/sh
```

Espera, ¡no ha pasado nada! ¿Es un error? Bueno, no. Estos shells interactivos saldrán después de ejecutar cualquier comando con script, a menos que se ejecuten en una terminal interactiva - así que para que este ejemplo no salga, necesitas

```
docker run -it alpine /bin/sh
```

Ahora estás dentro del shell del contenedor y puedes probar algunos comandos. Salga del contenedor dando el comando exit.

Bien, ahora es el momento de ver el comando docker ps. El comando docker ps te muestra todos los contenedores que se están ejecutando actualmente.

```
docker ps
```

Como no hay contenedores en ejecución, se ve una línea en blanco. Probemos una variante más útil:

```
docker ps -a
```

Lo que ve arriba es una lista de todos los contenedores que ha ejecutado. Observa que la columna STATUS muestra que estos contenedores se cerraron hace unos minutos. Probablemente te estés preguntando si hay una forma de ejecutar más de un comando en un contenedor. Intentemos eso ahora:

```
docker run -it alpine /bin/sh
```

Ejecutar el comando run con las banderas -it nos adjunta a una consola interactiva dentro del contenedor. Ahora puedes ejecutar todos los comandos que quieras en el contenedor. Tómate un tiempo para ejecutar tus comandos favoritos.

Ahora abre otra consola y ejecuta el comando

```
docker ps
```

¿Aparece algún contender? ¿Por qué?

Esto concluye un recorrido relámpago del comando docker run, que probablemente será el comando que usarás más a menudo. Tiene sentido dedicar algo de tiempo a sentirse cómodo con él. Para saber más sobre run, utiliza docker run --help para ver una lista de todas las banderas que soporta. A medida que avanza, veremos algunas variantes más de docker run.

Terminología

En la última sección, has visto un montón de jerga específica de Docker que puede ser confusa para algunos. Así que antes de seguir adelante, vamos a aclarar algunos términos que se utilizan con frecuencia en el ecosistema Docker.

Imágenes: Las imágenes son plantillas de sólo lectura que contienen instrucciones para crear un contenedor. Una imagen Docker crea contenedores que se ejecutan en la plataforma Docker. Piensa en una imagen como un plano o una instantánea de lo que habrá en un contenedor cuando se ejecute.

Contenedores: Un contenedor es un lugar aislado donde una aplicación se ejecuta sin afectar al resto del sistema y sin que el sistema afecte a la aplicación. Como el contenedor se ejecuta de forma nativa en Linux y comparte el kernel de la máquina anfitriona, es ligero y no utiliza más memoria que otros ejecutables.

Docker daemon - El servicio de fondo que se ejecuta en el host y que gestiona la construcción, ejecución y distribución de los contenedores Docker.
Cliente Docker - La herramienta de línea de comandos que permite al usuario interactuar con el demonio Docker.
Docker Store - Un registro de imágenes Docker, donde puedes encontrar contenedores de confianza y listos para la empresa, plugins y ediciones Docker. Lo utilizarás más adelante en este tutorial.

WebApps con Docker

Muy bien. Así que ya has visto la ejecución de Docker, has jugado con un contenedor Docker y también has aprendido algo de terminología. Ahora estás listo para llegar a la cosa real - el despliegue de aplicaciones web con Docker.

Ejecutar un sitio web estático en un contenedor

Primero, usaremos Docker para ejecutar un sitio web estático en un contenedor. El sitio web se basa en una imagen existente. Sacaremos una imagen Docker de Docker Store, ejecutaremos el contenedor y veremos lo fácil que es configurar un servidor web.

La imagen que vamos a utilizar es un sitio web de una sola página que ya fue creado para esta demostración y está disponible en la Docker Store como dockersamples/static-site. Puedes descargar y ejecutar la imagen directamente de una sola vez utilizando docker run de la siguiente manera.

```
docker run -d dockersamples/static-site
```

Entonces, ¿qué sucede cuando se ejecuta este comando?

Como la imagen no existe en tu host Docker, el demonio Docker primero la obtiene del registro y luego la ejecuta como un contenedor.

Ahora que el servidor se está ejecutando, ¿ves el sitio web? ¿En qué puerto se está ejecutando? Y lo que es más importante, ¿cómo se accede al contenedor directamente desde nuestra máquina anfitriona?

En este caso, el cliente no le dijo al motor Docker que publicara ninguno de los puertos, así que necesitas volver a ejecutar el comando docker run para añadir esta instrucción.

Vamos a volver a ejecutar el comando con algunas banderas nuevas para publicar los puertos y pasar su nombre al contenedor para personalizar el mensaje mostrado. Volveremos a utilizar la opción -d para ejecutar el contenedor en modo separado.

En primer lugar, detengamos el contenedor que acabamos de lanzar. Para ello, necesitamos el ID del contenedor.

Como hemos ejecutado el contenedor en modo detached, no tenemos que lanzar otro terminal para hacer esto. Ejecuta docker ps para ver los contenedores en ejecución.

```
docker ps
```

Fíjese en la columna ID DEL CONTENEDOR. Tendrá que utilizar este valor de ID de contenedor, una larga secuencia de caracteres, para identificar el contenedor que desea detener, y luego eliminarlo. El ejemplo siguiente proporciona el ID de contenedor en nuestro sistema; usted debe utilizar el valor que ve en su terminal.

```
docker stop ID
docker rm   ID
```

Ahora, vamos a lanzar un contenedor como se muestra a continuación:

```
docker run --name static-site -e AUTHOR="TuNombre" -d -P dockersamples/static-site
```

En el comando anterior:

-d creará un contenedor con el proceso separado de nuestro terminal

-P publicará todos los puertos expuestos del contenedor a puertos aleatorios en el host Docker

-e es la forma de pasar variables de entorno al contenedor

--name permite especificar el nombre del contenedor

AUTOR es el nombre de la variable de entorno y Your Name es el valor que puedes pasar

Ahora puedes ver los puertos ejecutando el comando docker port.

```
docker port static-site
```

Ahora puede abrir http://<SU_DIRECCIÓN_IP>:[SU_PUERTO] para ver su sitio.

Accede al sitio web y saca una captura de pantalla.

Para desplegar esto en un servidor real sólo tendrías que instalar Docker, y ejecutar el comando docker de arriba.

Ahora que has visto cómo ejecutar un servidor web dentro de un contenedor Docker, ¿cómo crear tu propia imagen Docker? Esta es la pregunta que exploraremos en la siguiente sección.

Pero primero, vamos a parar y eliminar los contenedores ya que no los vas a utilizar más.

```
docker stop static-site
docker rm static-site
```

Ejecuta docker ps para asegurarte de que los contenedores han desaparecido.

```
docker ps
```

Imagenes Docker

En esta sección, vamos a profundizar en lo que son las imágenes Docker. Construirás tu propia imagen, usarás esa imagen para ejecutar una aplicación localmente.

Las imágenes Docker son la base de los contenedores. En el ejemplo anterior, has sacado la imagen dockersamples/static-site del registro y has pedido al cliente Docker que ejecute un contenedor basado en esa imagen. Para ver la lista de imágenes que están disponibles localmente en su sistema, ejecute el comando docker images.

```
docker images
```

Arriba hay una lista de imágenes que he sacado del registro y las que he creado yo mismo (pronto veremos cómo). Usted tendrá una lista diferente de imágenes en su máquina. El TAG se refiere a una instantánea particular de la imagen y el ID es el identificador único correspondiente a esa imagen.

Para simplificar, puedes pensar en una imagen como un repositorio git - las imágenes pueden ser confirmadas con cambios y tener múltiples versiones. Cuando no se proporciona un número de versión específico, el cliente utiliza por defecto la más reciente.

Por ejemplo, usted podría obtener una versión específica de la imagen de ubuntu de la siguiente manera:

```
docker pull ubuntu:12.04
```

Si no se especifica el número de versión de la imagen, entonces, como se ha mencionado, el cliente Docker utilizará por defecto una versión llamada latest.

Así, por ejemplo, el comando docker pull que se da a continuación extraerá una imagen llamada ubuntu:latest:

```
docker pull ubuntu
```

Para obtener una nueva imagen de Docker puedes obtenerla de un registro (como el Docker Store) o crear la tuya propia. Hay cientos de miles de imágenes disponibles en Docker Store. También puedes buscar imágenes directamente desde la línea de comandos utilizando docker search.

Una distinción importante con respecto a las imágenes es entre las imágenes base y las imágenes hijo(child).

- Las imágenes base son imágenes que no tienen imágenes padre, normalmente imágenes con un SO como ubuntu, alpine o debian.
- Las imágenes hijo son imágenes que se basan en las imágenes base y añaden funcionalidad adicional.

Otro concepto clave es la idea de imágenes oficiales e imágenes de usuario. (Ambas pueden ser imágenes base o imágenes hijo).

- Las imágenes oficiales son imágenes sancionadas por Docker. Docker, Inc. patrocina un equipo dedicado que es responsable de revisar y publicar todo el contenido de los Repositorios Oficiales. Este equipo trabaja en colaboración con los mantenedores de software upstream, expertos en seguridad y la comunidad Docker en general. Estos no llevan el prefijo de una organización o nombre de usuario. En la lista de imágenes anterior, las imágenes python, node, alpine y nginx son imágenes oficiales (base). Para saber más sobre ellas, consulta la documentación de las imágenes oficiales.
- Las imágenes de usuario son imágenes creadas y compartidas por usuarios como tú. Se basan en las imágenes base y añaden funcionalidades adicionales. Suelen tener el formato usuario/nombre de la imagen. El valor del usuario en el nombre de la imagen es el nombre de su usuario u organización de Docker Store.

Para los propósitos de este taller, hemos creado una pequeña y divertida aplicación Python Flask que muestra un gato .gif al azar cada vez que se carga.

Crear tu primera APP

Comienza creando un directorio llamado flask-app donde crearemos los siguientes archivos:

- app.py
- requirements.txt
- templates/index.html
- Dockerfile

Asegúrate de cd flask-app antes de empezar a crear los archivos, porque no querrás empezar a añadir un montón de otros archivos al azar a tu imagen.

Estos archivos están en https://github.com/DAW-2022-2023/DWEC

DockerFile

Un Dockerfile es un archivo de texto que contiene una lista de comandos que el demonio Docker llama mientras crea una imagen. El Dockerfile contiene toda la información que Docker necesita saber para ejecutar la aplicación - una imagen Docker base desde la que ejecutar, la ubicación del código de tu proyecto, cualquier dependencia que tenga, y qué comandos ejecutar al inicio. Es una forma sencilla de automatizar el proceso de creación de imágenes. La mejor parte es que los comandos que escribes en un Dockerfile son casi idénticos a sus comandos equivalentes de Linux. Esto significa que realmente no tienes que aprender una nueva sintaxis para crear tus propios Dockerfiles.

Construir la imagen

Ahora que tienes tu Dockerfile, puedes construir tu imagen. El comando docker build se encarga de crear una imagen docker a partir de un Dockerfile.

El comando docker build es bastante simple - toma un nombre de etiqueta opcional con la bandera -t, y la ubicación del directorio que contiene el Dockerfile.

Como consejo haz que el root de tu consola sea el directorio Tema_2 obtenido al clonar el repositorio y ejecuta el siguiente comando.

```
docker build -t myfirstapp ./
```

Ejecuta docker images y mira si tu imagen (myfirstapp) aparece.

Correr tu imagen

El siguiente paso en esta sección es ejecutar la imagen y ver si realmente funciona.

```
docker run -p 8888:5000 --name cats myfirstapp
```

Dirígete a http://localhost:8888 y tu aplicación debería estar activa. Pulsa el botón Actualizar en el navegador web para ver más imágenes de gatos.

Haz tres capturas de pantalla de la web mostrando gatos diferentes.