UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
PRACTICAS INICIALES
Año 2,203

MANUAL DE USUARIO

202200007 - Natalia Calderon Echeverria

202106651 - Luis Carlos Corleto Marroquín

202200131 - Carlos Samuel Aguilar Acosta

202200033 - Isai Dardón Mayén

202002907 - Moises Ivan D. Alvarado García

201909035 - Josue Javier Aguilar Lopez

INDICE

Contenido

Introducción

Docker es una plataforma de contenedores que revolucionó la forma en que las aplicaciones se desarrollan, distribuyen y ejecutan. A medida que las aplicaciones se volvieron más complejas y diversas, surgió la necesidad de una solución que permitiera a los desarrolladores crear, empaquetar y desplegar sus aplicaciones de manera coherente en diferentes entornos.

En lugar de ejecutar aplicaciones directamente en el sistema operativo del anfitrión, Docker introduce el concepto de contenedores. Un contenedor es una unidad ligera y autónoma que contiene todo lo necesario para ejecutar una aplicación, incluidos el código, las bibliotecas, las dependencias y las configuraciones. Esto asegura que la aplicación se ejecute de manera consistente en diferentes entornos, ya sea en un entorno de desarrollo local, en un servidor de prueba o en la nube.

Conceptos Clave de Docker:

1. Imagen

2. Contenedor

3. Dockerfile

4. Registro

5. Orquestación

Docker

Docker es una plataforma de virtualización a nivel de sistema operativo que se utiliza para desarrollar, desplegar y ejecutar aplicaciones en contenedores. es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Un contenedor es un paquete de software que incluye todo lo necesario para ejecutar una aplicación, incluyendo el código fuente, las dependencias y las bibliotecas.

Los contenedores son una forma de virtualización a nivel de sistema operativo. Esto significa que los contenedores comparten el kernel del sistema operativo con otros contenedores, lo que los hace más ligeros y eficientes que las máquinas virtuales tradicionales.

Algunos de los conceptos clave asociados con Docker son:

1. Imagen: Una imagen de Docker es una plantilla que contiene un sistema de archivos y metadatos necesarios para ejecutar una aplicación. Las imágenes son estáticas y se utilizan como base para crear contenedores.

2. Contenedor: Un contenedor Docker es una instancia en ejecución de una imagen. Los contenedores están aislados unos de otros y del sistema anfitrión, lo que garantiza que las aplicaciones sean consistentes y no entren en conflicto entre sí.

3. Dockerfile: Un Dockerfile es un archivo de texto que contiene una serie de instrucciones para construir una imagen. Estas instrucciones pueden incluir la instalación de paquetes, la configuración de variables de entorno y la copia de archivos.

4. Registro: Un registro de Docker es un repositorio donde se almacenan y administran imágenes de contenedores. Docker Hub es el registro público más conocido, pero también puedes configurar tus propios registros privados.

5. Orquestación: La orquestación de contenedores implica administrar, coordinar y escalar múltiples contenedores como parte de una aplicación más grande.

Importancia de Docker

1. Portabilidad: Docker permite empaquetar aplicaciones y todas sus dependencias en contenedores. Esto hace que las aplicaciones sean altamente portátiles, lo que significa que puedes desarrollar, probar y ejecutar las mismas aplicaciones en diferentes entornos, desde tu máquina local hasta la nube, sin preocuparte por las diferencias de configuración.

2. Eficiencia: Los contenedores comparten el mismo núcleo del sistema operativo y utilizan menos recursos en comparación con máquinas virtuales completas. Esto los hace más eficientes en términos de uso de recursos y tiempo de inicio.

3. Consistencia: Los contenedores de Docker garantizan que las aplicaciones se ejecuten de manera consistente en cualquier entorno, independientemente de las diferencias en la infraestructura subyacente.

4. Desarrollo Ágil: Docker simplifica el proceso de desarrollo al permitir a los equipos de desarrollo y operaciones trabajar en los mismos entornos.

5. Escalabilidad: Docker facilita la escalabilidad de las aplicaciones. Puedes crear múltiples instancias de contenedores idénticos para manejar una mayor carga y equilibrarla fácilmente utilizando herramientas de orquestación como Kubernetes o Docker Swarm.

6. Aislamiento: Docker proporciona aislamiento a nivel de sistema operativo para las aplicaciones en contenedores. Esto asegura que una aplicación en un contenedor no afecte a otras aplicaciones en contenedores ni al sistema anfitrión, lo que aumenta la seguridad y la confiabilidad.

En resumen, Docker es importante porque simplifica y mejora drásticamente la forma en que las aplicaciones se desarrollan, distribuyen y ejecutan. Ayuda a las organizaciones a ser más ágiles, a aumentar la eficiencia y a garantizar la consistencia y la confiabilidad de las aplicaciones en diferentes entornos.

Beneficios de Docker

Entre las principales ventajas de utilizar Docker se encuentran:

- **Portabilidad mejorada.** Este es una de sus principales ventajas ya que ofrece al usuario la posibilidad de crear o instalar una aplicación compleja y asegurarse que sí funcionará en otro espacio, ya que los contenedores Docker incorporan todo lo que requiere una aplicación para funcionar y prácticamente sin la intervención del usuario.
- **Transferencia simple.** Las aplicaciones basadas en Docker pueden transferirse desde equipos de desarrollo locales a implementaciones de producción en AWS.
- **Automatización**. Con la ayuda de las tareas cron y los contenedores Docker, los usuarios pueden automatizar sus tareas.
- **Apoyo.** Docker cuenta con un canal en Slack dedicado, un foro de la comunidad Docker y miles de colaboradores en portales web para desarrolladores como por ejemplo: Stack Overflow.
- **Contenedores automatizados.** Docker diseña automáticamente un contenedor basado en el código fuente de la aplicación.
- **Control de versiones del contenedor.** Docker permite rastrear versiones de una imagen del contenedor, retroceder a versiones anteriores y rastrear quién creó una versión y cómo.
- **Reutilización de contenedores y bibliotecas compartidas.** Los contenedores ya creados se pueden usar como plantillas para hacer contenedores nuevos. Una vez creados, los desarrolladores tienen la opción de ir a una fuente abierta de registro y encontrar miles de contenedores facilitados por otros usuarios.
- **Diseño prioritario para DevOps y desarrolladores**. Con esta herramienta los desarrolladores pueden diseñar y poner en funcionamiento aplicaciones como contenedores portátiles y ligeros.
- **Eficiencia.** Los contenedores comparten el mismo núcleo del sistema operativo y utilizan menos recursos en comparación con máquinas virtuales completas. Esto los hace más eficientes en términos de uso de recursos y tiempo de inicio.
- **Consistencia.** Los contenedores de Docker garantizan que las aplicaciones se ejecuten de manera consistente en cualquier entorno, independientemente de las diferencias en la infraestructura subyacente.

Instalar Docker

1. Abre una terminal.

2. Luego, el sistema debe actualizarse para que sea más seguro y confiable para la instalación de Docker. Ejecuta los siguientes dos comandos:

```
sudo apt update
sudo apt upgrade
```

3. Una vez que hayas actualizado el sistema, necesitarás instalar algunos paquetes necesarios antes de instalar Docker. Puedes hacerlo con la ayuda de un solo comando:

```
sudo apt-get install  curl apt-transport-https ca-certificates software-properties-common
```

Para comprender mejor el comando anterior, aquí hay una breve descripción de lo que significa:

- **apt-transport-https:** permite que el administrador de paquetes transfiera datos a través de https
- **ca-certificates:** permite que el navegador web y el sistema verifiquen los certificados de seguridad
- **curl:** transfiere datos
- **software-properties-common:** agrega scripts para administrar el software

4. Ahora tienes que agregar los repositorios de Docker. Esto facilitará mucho el proceso de instalación y al mismo tiempo podrás utilizar el método de instalación oficialmente compatible. Primero, agrega la clave GPG, ingresando el siguiente comando en la línea de comando:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

A continuación, agrega el repositorio:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
```

Después de eso, actualiza la información del repositorio:

```
sudo apt update
```

5. Usa el comando apt para instalar Docker:

```
sudo apt install docker-ce
```

6. Una vez que se completa la instalación, es buena idea verificar el estado del servicio:

```
sudo systemctl status docker
```

Comandos Básicos

1. **Crear y ejecutar un contenedor a partir de una imagen:** Esto creará y ejecutará un nuevo contenedor basado en la imagen especificada. El indicador -it permite la interacción con el contenedor en un terminal interactivo.

```
docker run -it nombre-de-la-imagen
```

2. **Listar contenedores en ejecución:** Muestra una lista de los contenedores en ejecución.

```
docker ps
```

3. **Listar todos los contenedores (incluidos los detenidos):** Muestra todos los contenedores, tanto en ejecución como detenidos.

```
docker ps -a
```

4. **Detener un contenedor en ejecución:**

```
docker stop ID-o-NOMBRE
```

5. **Iniciar un contenedor detenido:**

```
docker start ID-o-NOMBRE
```

6. **Eliminar un contenedor detenido:**

```
docker rm ID-o-NOMBRE
```

7. **Listar imágenes locales:**

```
docker images
```

8. **Eliminar una imagen:** Elimina una imagen local. Asegúrate de que no haya contenedores basados en esa imagen antes de eliminarla.

```
docker rmi nombre-de-la-imagen
```

9. **Ver registros de un contenedor:** Muestra los registros de salida de un contenedor.

```
docker logs ID-o-NOMBRE
```

10. **Ejecutar un comando en un contenedor en ejecución:** Ejecuta un comando dentro de un contenedor en ejecución.

```
docker exec -it ID-o-NOMBRE comando
```

Dockerfile

Un Dockerfile es un archivo de texto plano que contiene una serie de instrucciones necesarias para crear una imagen que, posteriormente, se convertirá en una sola aplicación utilizada para un determinado propósito. Similar a lo explicado anteriormente, y la base del funcionamiento de Docker es mediante Dockerfile.

Es como la receta necesaria para un banquete, en este caso el Dockerfile es necesario para la imagen que queramos construir, el Dockerfile es la receta y el gran banquete será nuestra imagen.

Importancia de Dockerfile

1. Reproducibilidad: Uno de los problemas más comunes en el desarrollo de software es la inconsistencia entre los entornos de desarrollo, pruebas y producción. Dockerfile resuelve este problema al permitir a los desarrolladores describir todas las dependencias, configuraciones y pasos necesarios para crear un entorno de ejecución específico. Esto asegura que el mismo entorno se pueda replicar en cualquier lugar.

2. Automatización: Dockerfile facilita la automatización de la construcción de imágenes. Al describir cada paso en el proceso de construcción, los desarrolladores pueden crear fácilmente scripts y flujos de trabajo automatizados para construir imágenes consistentes y confiables.

3. Control de Versiones: Dockerfile es parte del código fuente de una aplicación y puede ser almacenado en sistemas de control de versiones como Git. Esto permite que el Dockerfile se mantenga junto con el código fuente de la aplicación y evite posibles problemas de versiones desalineadas.

4. Personalización: Dockerfile ofrece un alto grado de personalización en la construcción de imágenes. Los desarrolladores pueden elegir una imagen base adecuada y luego agregar software, configuraciones y scripts específicos para las necesidades de la aplicación.

5. Eficiencia: Docker utiliza una arquitectura de capas para construir imágenes, lo que significa que si se han realizado cambios en una capa, solo esa capa necesita ser reconstruida. Esto hace que el proceso de construcción sea rápido y eficiente.

Sintaxis Básica

Características
Dentro de las características de un Dockerfile se encuentra su ejecución, que se lleva a cabo siguiendo ciertos pasos de fácil aplicación, iniciando con la indicación de las instrucciones que el usuario considere necesario para la creación de la imagen. Después de esto, se debe ejecutar el comando docker build, que hará que el Dockefile esté disponible para su uso y crear contenedores. Esto implica que esta herramienta funciona bajo el esquema Dockerfile > Docker Image > Docker Container.

Además de esto, un dcokerfile se caracteriza por definir las instrucciones para crear una nueva imagen, pero siempre inicia con una imagen base que ya existe en el sistema. De manera que la imagen creada surge partir de esta base, pero cuenta con una serie de diferencias y propiedades distintas que distingue una de la otra.

Funcionamiento de Dockerfile
En lo que respecta al funcionamiento de un archivo de texto Dockerfile, es posible agregar que cada una de las instrucciones incluidas en esta herramienta para crear una imagen se ubica en una línea distinta y se ejecutan una tras otra, es decir, de forma consecutiva como si se tratara de un script por lotes.

De modo que la creación de una imagen de Docker depende de las instrucciones del Dockerfile. Estas, a su vez, se inician una vez se ejecute el comando docker build dentro de la plataforma de contenedores.

Para el funcionamiento de Dockerfile se debe tener en cuenta también el llamado contexto docker build context, es decir, la herramienta que se encarga de indicar cuáles son los archivos, directorios y ficheros a los que tiene acceso la opción de docker build. Por tanto, los datos e información pertenecientes al directorio fuente se transfieren al docker daemon una vez se ejecute el comando docker build. Esto significa que, para la serie de pasos incluidas en un Dockerfile, podrán tener acceso a la información relacionada con los archivos del build context.

En los casos donde el usuario no necesite utilizar la totalidad de archivos incluidos en el directorio fuente local en el build context, puede acudir a la herramienta de .dockerignore, que se encargará de excluir los elementos que no requiera la imagen final.

Dentro de las opciones más relevantes para un Dockerfile se encuentran herramientas encargadas de labores, como el establecimiento de la imagen base, el cambio de usuario o los elementos preestablecidos para el arranque de un contenedor en Docker, entre otros.

Comandos Importantes de Dockerfile

Aquí hay una lista de algunos comandos importantes en Dockerfile:

- FROM: Define la imagen base a partir de la cual se construirá la nueva imagen.

```
FROM ubuntu:latest
```

- RUN: Ejecuta comandos en el contenedor durante la construcción de la imagen.

```
RUN apt-get update && apt-get install -y nginx
```

- COPY y ADD: Copian archivos y directorios del sistema de archivos local al contenedor.

```
COPY mi_app /var/www/html/
```

- WORKDIR: Establece el directorio de trabajo para las instrucciones siguientes.

```
WORKDIR /app
```

- EXPOSE: Indica los puertos en los que el contenedor estará escuchando en tiempo de ejecución.

```
EXPOSE 80
```

- ENV: Establece variables de entorno en el contenedor.

```
ENV DEBUG_MODE=true
```

- CMD y ENTRYPOINT: Definen el comando por defecto que se ejecutará cuando el contenedor se inicie.

```
CMD ["nginx", "-g", "daemon off;"]
```

- VOLUME: Crea un volumen en el contenedor para almacenar datos persistentes.

```
VOLUME /data
```

- ARG: Define argumentos que se pueden pasar durante la construcción de la imagen.

```
ARG version=latest
```

- LABEL: Agrega metadatos a la imagen para proporcionar información adicional.

```
LABEL maintainer="nombre@ejemplo.com"
```

- RUN vs CMD vs ENTRYPOINT: Son comandos diferentes con propósitos distintos: RUN se utiliza para ejecutar comandos en la etapa de construcción, mientras que CMD y ENTRYPOINT se usan para definir el comando por defecto al iniciar un contenedor.

Recuerda que la construcción de un Dockerfile implica una secuencia de capas y operaciones, y que la elección de los comandos y su orden depende de los requisitos específicos de tu aplicación.

Docker Hub

Docker Hub es un servicio en línea desarrollado por Docker, Inc. que desempeña un papel fundamental en el ecosistema de Docker. Se trata de un registro público y privado de imágenes de contenedores, diseñado para simplificar la distribución y colaboración en el desarrollo y despliegue de aplicaciones basadas en contenedores. Docker Hub proporciona un lugar centralizado para almacenar, acceder y compartir imágenes de contenedores, lo que acelera y facilita el proceso de implementación.

Características

Registro de Imágenes
Docker Hub actúa como un registro, que es un repositorio centralizado donde los desarrolladores pueden almacenar y compartir imágenes de contenedores. Estas imágenes contienen aplicaciones, bibliotecas, dependencias y configuraciones necesarias para ejecutar aplicaciones en entornos aislados. Al proporcionar un registro de imágenes confiable y escalable, Docker Hub permite a los equipos de desarrollo y operaciones compartir soluciones completas y consistentes.

1. Imágenes Públicas y Privadas
Docker Hub admite tanto imágenes de contenedores públicas como privadas. Las imágenes públicas son accesibles para cualquier persona y se utilizan comúnmente para compartir proyectos de código abierto. Por otro lado, las imágenes privadas están protegidas y solo son accesibles para aquellos a quienes el propietario otorga permiso, lo que brinda una forma segura de compartir imágenes internas o comerciales.

2. Colaboración y Distribución
Docker Hub facilita la colaboración y distribución entre equipos de desarrollo y operaciones. Los desarrolladores pueden cargar sus imágenes en Docker Hub y permitir que otros miembros del equipo las descarguen y ejecuten. Esto simplifica la colaboración en proyectos, ya que todos los miembros del equipo pueden acceder a una versión confiable y coherente de la aplicación.

3. Versionamiento de Imágenes
Las imágenes en Docker Hub pueden estar etiquetadas con versiones específicas. Esto permite a los usuarios acceder a versiones anteriores o específicas de una imagen. El etiquetado adecuado es esencial para garantizar la consistencia en diferentes entornos y evitar problemas de compatibilidad.

4. Integración con Docker CLI
La integración con la línea de comandos de Docker (Docker CLI) facilita la descarga y el uso de imágenes directamente desde Docker Hub. Los desarrolladores pueden usar comandos simples para descargar imágenes y ejecutar contenedores basados en esas imágenes.

5. Automatización de Procesos
Docker Hub permite la automatización de procesos, como la construcción y la actualización de imágenes cuando se realizan cambios en un repositorio de código fuente. Esto es particularmente útil en flujos de trabajo de desarrollo continuo (CI/CD), donde los contenedores se construyen y prueban automáticamente antes de implementarse en entornos de producción.

6. Organizaciones y Equipos
Docker Hub permite crear organizaciones y equipos dentro de esas organizaciones. Esto es especialmente útil en entornos empresariales donde varios equipos colaboran en diferentes proyectos. Las organizaciones pueden tener repositorios compartidos y gestionar el acceso de los miembros del equipo a esas imágenes.

7. Automatización de Compilación (Automated Builds)
Docker Hub proporciona una característica llamada "Automated Builds" (compilaciones automatizadas) que permite vincular tu repositorio de código fuente a Docker Hub. Cuando realizas cambios en el repositorio, Docker Hub puede detectar estos cambios y automáticamente construir una nueva imagen basada en los archivos Dockerfile presentes en el repositorio. Esto simplifica la tarea de mantener imágenes actualizadas en Docker Hub.

8. Webhooks y Notificaciones
Docker Hub también admite la configuración de webhooks, que son llamadas HTTP que se realizan cuando ocurren ciertos eventos, como la construcción exitosa de una imagen. Estos webhooks pueden ser utilizados para integrar Docker Hub con sistemas de automatización y notificación, como sistemas de seguimiento de problemas o canales de comunicación internos.

9. Imágenes Oficiales y Validación
Docker Hub aloja imágenes oficiales que son mantenedas por los propios equipos de Docker, Inc. Estas imágenes cumplen con estándares de calidad y seguridad, y son ampliamente utilizadas como bases confiables para construir aplicaciones. La validación y el mantenimiento continuo de estas imágenes oficiales garantizan la calidad y la confiabilidad.

10. Docker Hub Enterprise
Además de Docker Hub público, Docker también ofrece una versión empresarial llamada Docker Hub Enterprise. Esta versión permite a las organizaciones implementar su propio registro de imágenes privado y seguro en su infraestructura local, lo que brinda un mayor control y seguridad sobre el almacenamiento y la distribución de imágenes.

Crear una Cuenta

Se mostrara una guía paso a paso para crear una cuenta en Docker Hub:

Accede al Sitio Web de Docker Hub:
Abre tu navegador web y ve al sitio web oficial de Docker Hub en https://hub.docker.com/

Registro:
En la página principal de Docker Hub, verás un botón "Sign Up" (Registrarse) en la esquina superior derecha. Haz clic en ese botón para comenzar el proceso de registro.

Elige tu Plan:
Docker Hub ofrece diferentes planes, incluyendo opciones gratuitas y opciones de pago con características adicionales. Selecciona el plan que mejor se adapte a tus necesidades. Para comenzar, elige el plan gratuito haciendo clic en "Continue".

Crea tu Cuenta:
A continuación, deberás completar el formulario de registro. Ingresa la siguiente información:

- Tu dirección de correo electrónico: Asegúrate de usar una dirección válida, ya que recibirás un correo electrónico para verificarla.
- Nombre de usuario: Elige un nombre de usuario único que utilizarás para acceder a tu cuenta en Docker Hub.
- Contraseña: Crea una contraseña segura para tu cuenta.
- Opcional: Puedes optar por recibir actualizaciones por correo electrónico de Docker.

Verifica tu Correo Electrónico:
Una vez que hayas completado el formulario de registro, recibirás un correo electrónico de Docker Hub en la dirección que proporcionaste. Abre el correo electrónico y haz clic en el enlace de verificación para confirmar tu dirección de correo electrónico.

Completa el Registro:
Después de verificar tu dirección de correo electrónico, se te redirigirá de nuevo al sitio web de Docker Hub. Aquí deberás proporcionar algunos detalles adicionales, como tu nombre completo y una breve descripción. Luego, haz clic en "Continue".

Preferencias de Comunicación:
Docker Hub te dará la opción de recibir correos electrónicos sobre anuncios, actualizaciones y noticias relacionadas con Docker. Selecciona tus preferencias y haz clic en "Continue".

¡Listo!:
Una vez que hayas completado todos los pasos anteriores, habrás creado exitosamente tu cuenta en Docker Hub. Ahora puedes acceder a tu cuenta utilizando tu nombre de usuario y contraseña.

Recuerda que después de crear tu cuenta, podrás utilizarla para subir y descargar imágenes de contenedores desde y hacia Docker Hub, lo que facilita la colaboración y el uso compartido de tus aplicaciones basadas en contenedores.

Subir una Imagen

Subir una imagen a Docker Hub es un proceso relativamente sencillo pero requiere que tengas una cuenta en Docker Hub y que hayas creado una imagen localmente. Aquí tienes los pasos para subir una imagen a Docker Hub:

Paso 1: Crear una Cuenta en Docker Hub
Si aún no tienes una cuenta en Docker Hub, ve al sitio web de Docker Hub (https://hub.docker.com/) y crea una cuenta. Confirma tu dirección de correo electrónico siguiendo las instrucciones proporcionadas por Docker Hub.

Paso 2: Construir una Imagen Localmente
Antes de subir una imagen a Docker Hub, debes construir la imagen localmente en tu máquina utilizando un Dockerfile. Asegúrate de haber etiquetado correctamente la imagen con un nombre y una etiqueta significativos.

```
docker build -t nombre_de_usuario/nombre_imagen:etiqueta .
```

Reemplaza nombre_de_usuario, nombre_imagen y etiqueta con los valores apropiados.

Paso 3: Iniciar Sesión en Docker Hub
Abre una terminal y utiliza el comando docker login para iniciar sesión en tu cuenta de Docker Hub.

```
docker login
```

Se te pedirá que ingreses tu nombre de usuario, contraseña y, posiblemente, un código de autenticación de dos factores si lo tienes habilitado.

Paso 4: Etiquetar la Imagen
Si aún no has etiquetado la imagen correctamente, asegúrate de hacerlo antes de subirla. Utiliza el mismo nombre de usuario, nombre de imagen y etiqueta que utilizaste al crear la imagen.

```
docker tag nombre_de_usuario/nombre_imagen:etiqueta nombre_de_usuario/nombre_imagen:etiqueta
```

Paso 5: Subir la Imagen a Docker Hub
Una vez que hayas etiquetado la imagen, puedes subirla a Docker Hub utilizando el comando docker push.

```
docker push nombre_de_usuario/nombre_imagen:etiqueta
```

Este comando enviará la imagen a tu repositorio en Docker Hub.

Paso 6: Verificar en Docker Hub
Después de que la imagen se haya subido correctamente, puedes acceder a tu cuenta de Docker Hub en línea para verificar que la imagen esté presente en tu repositorio.

IMPORTANTE PARA LOS PASOS QUE SE DIERON:
Recuerda que estos pasos asumen que ya tienes una imagen construida localmente. Si aún no has creado la imagen, asegúrate de crearla utilizando un Dockerfile antes de intentar subirla a Docker Hub.

Descargar una Imagen

Se dará a continuación los pasos para descargar una imagen desde Docker Hub:

Abre una Terminal o línea de comandos:
Abre una terminal en tu sistema operativo. Puedes usar la Terminal en macOS y Linux, o el Símbolo del sistema en Windows.

Ejecuta el Comando docker pull:
Utiliza el comando docker pull seguido del nombre de usuario del propietario de la imagen, el nombre de la imagen y opcionalmente la etiqueta de versión.

```
docker pull nombre_de_usuario/nombre_imagen:etiqueta
```

Por ejemplo, si quieres descargar la última versión de la imagen de Ubuntu:

```
docker pull ubuntu:latest
```

Si deseas descargar una versión específica, reemplaza latest con la etiqueta correspondiente.

Espera a que se Descargue la Imagen:
Una vez que ingreses el comando, Docker comenzará a descargar la imagen desde Docker Hub. El tiempo de descarga dependerá del tamaño de la imagen y de tu velocidad de conexión a Internet.

Verifica la Descarga:
Una vez que la descarga se haya completado, verás mensajes en la terminal indicando el progreso. Cuando finalice, habrás descargado exitosamente la imagen desde Docker Hub.

Confirma la Descarga:
Puedes confirmar que la imagen se ha descargado correctamente ejecutando el comando docker images. Esto mostrará una lista de todas las imágenes que tienes en tu sistema. Busca la imagen que acabas de descargar en la lista.

```
docker images
```

¡Listo! Ahora has descargado exitosamente una imagen desde Docker Hub y estás listo para usarla para crear y ejecutar contenedore

Comandos Importantes de Docker Hub

Docker Hub es principalmente una plataforma en línea para administrar imágenes de contenedores y no tiene una línea de comandos propia como Docker CLI. Sin embargo, aquí hay algunos comandos importantes relacionados con Docker Hub que puedes ejecutar en tu terminal:

Iniciar Sesión en Docker Hub:
Para iniciar sesión en Docker Hub desde la línea de comandos, utiliza el siguiente comando:

```
docker login
```

Se te pedirá que ingreses tu nombre de usuario, contraseña y, si está habilitada, la autenticación de dos factores.

Etiquetar una Imagen:
Antes de subir una imagen a Docker Hub, es importante etiquetarla adecuadamente:

```
docker tag nombre_imagen:etiqueta nombre_de_usuario/nombre_imagen:etiqueta
```

Reemplaza nombre_imagen, nombre_de_usuario y etiqueta con los valores apropiados.

Subir una Imagen a Docker Hub:
Para subir una imagen etiquetada a Docker Hub, utiliza el comando docker push:

```
docker push nombre_de_usuario/nombre_imagen:etiqueta
```

Esto enviará la imagen al repositorio de Docker Hub asociado con tu cuenta.

Descargar una Imagen de Docker Hub:
Puedes descargar una imagen desde Docker Hub utilizando el comando docker pull:

```
docker pull nombre_de_usuario/nombre_imagen:etiqueta
```

Reemplaza nombre_de_usuario, nombre_imagen y etiqueta con los valores correctos.

Ver Información de la Imagen en Docker Hub:
A través de Docker CLI, puedes obtener información básica sobre una imagen en Docker Hub utilizando su nombre de usuario y nombre de imagen:

```
docker search nombre_de_usuario/nombre_imagen
```

Esto te mostrará las imágenes coincidentes en Docker Hub con detalles básicos.

Listar Imágenes Descargadas:
Puedes listar todas las imágenes que tienes localmente en tu sistema utilizando:

```
docker images
```

Esto te mostrará una lista de imágenes descargadas y creadas localmente.

Eliminar una Imagen Local:
Si deseas eliminar una imagen localmente, puedes usar:

```
docker rmi nombre_imagen:etiqueta
```

Esto eliminará la imagen de tu sistema local. Ten en cuenta que esto no elimina la imagen de Docker Hub si la has subido allí.

Eliminar Imágenes no Utilizadas:
Puedes eliminar todas las imágenes que no estén siendo utilizadas por ningún contenedor:

```
docker image prune
```

Esto eliminará imágenes no utilizadas y liberará espacio en disco.

Recuerda que Docker Hub es principalmente una plataforma en línea, y gran parte de la administración, búsqueda y visualización de imágenes se realiza a través de su sitio web. Los comandos de Docker CLI están más centrados en la gestión de contenedores e imágenes locales.

Conclusión

Docker es una plataforma de código abierto que permite a los desarrolladores empaquetar y enviar sus aplicaciones en contenedores, lo que facilita la implementación y el escalado. Un Dockerfile es un archivo de texto que contiene una serie de instrucciones para construir una imagen de Docker, lo que permite a los desarrolladores especificar exactamente cómo se deben construir sus contenedores. Docker Hub, por otro lado, es la biblioteca y comunidad más grande del mundo para imágenes de contenedores, donde puedes buscar más de 100,000 imágenes de contenedores de proveedores de software, proyectos de código abierto y la comunidad.

En resumen, Docker te permite empaquetar y enviar tus aplicaciones en contenedores, Dockerfile te permite crear imágenes personalizadas y Docker Hub te permite compartir y encontrar imágenes para tus proyectos. Estas herramientas trabajan juntas para simplificar y mejorar el proceso de desarrollo, implementación y distribución de aplicaciones.