Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Practicas Iniciales

Sección F-

Cat. Ing. Herman Igor Veliz Linares

# MANUAL DE USO

Carlos Javier Pérez Pocon – 202206425

Doni Esau Sican Tepeu - 202004181

Hamilton Hermelindo Bethancourt Zapeta 202105095

Danny Josué Ixcal Sija – 201807105

Ricardo Andreé Paniagua Rodenas – 202111219

Francisco Josué Galindo Mansilla - 202201511

# Docker

Docker es una plataforma que permite crear, ejecutar y gestionar contenedores de manera fácil y eficiente. Los contenedores permiten empaquetar aplicaciones y sus dependencias en un entorno aislado, lo que facilita la portabilidad entre diferentes sistemas.

Contenedor: Un contenedor Docker o Docker Container es una instancia de imagen Docker que se ejecuta en un microservicio individual o en un stack de aplicaciones completo. Cuando se lanza un contenedor, se añade una capa modificable a la imagen. Esto se utiliza para almacenar cualquier cambio realizado en el contenedor durante el tiempo de ejecución.

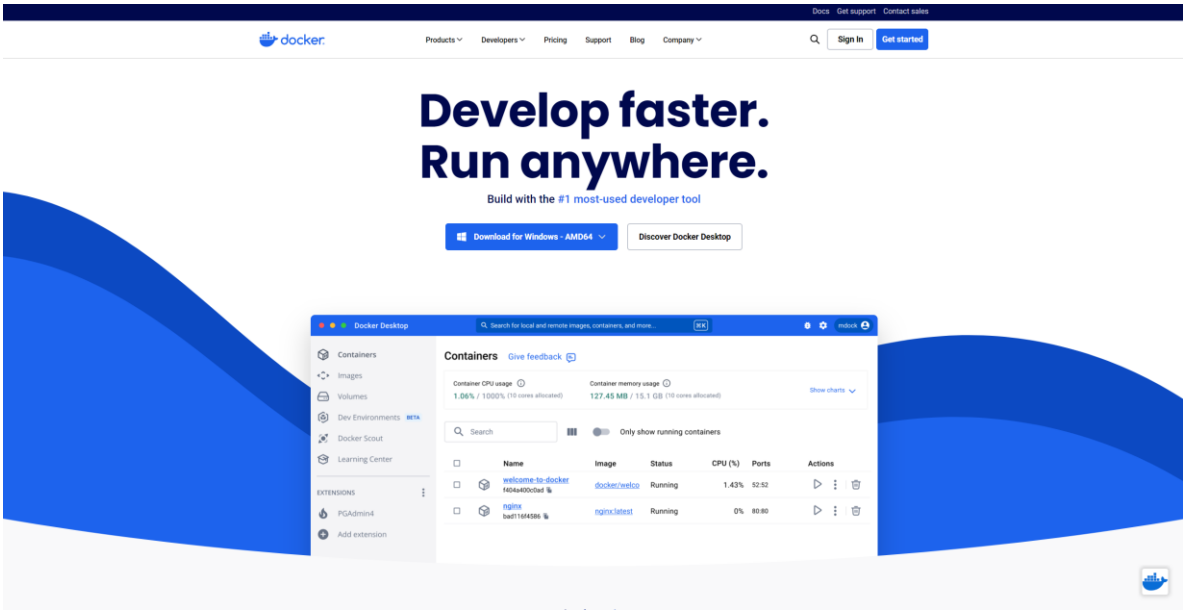
Imagen: Una imagen Docker es un modelo de «solo lectura» que se utiliza para crear contenedores Docker. Se compone de varias capas que agrupan todas las instalaciones, dependencias, bibliotecas, procesos y código de aplicación necesarios para un entorno de contenedores totalmente operativo.

Dockerfile: Cada contenedor Docker comienza con un Dockerfile. Se trata de un archivo de texto escrito con una sintaxis comprensible y que contiene las instrucciones para crear una imagen Docker.

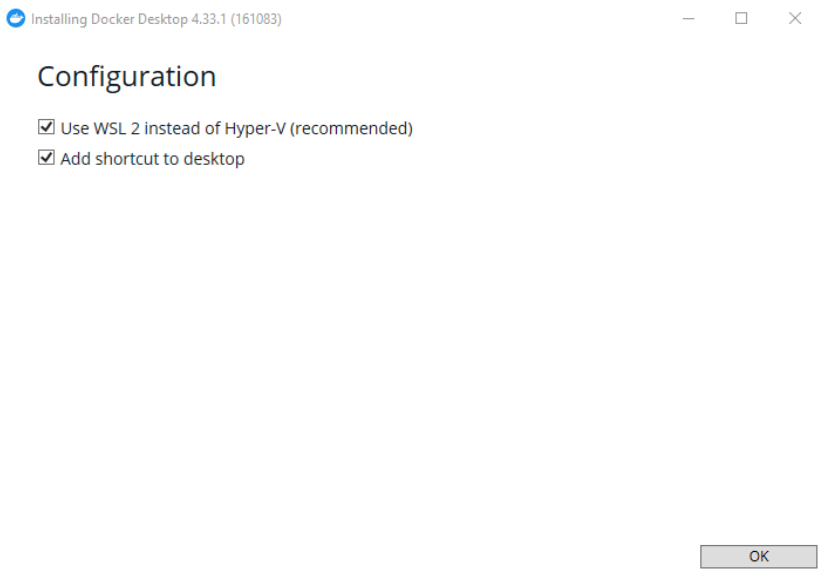
DockerHub: Es un repositorio en la nube donde puedes almacenar y compartir imágenes de Docker.

# Instalación de Docker

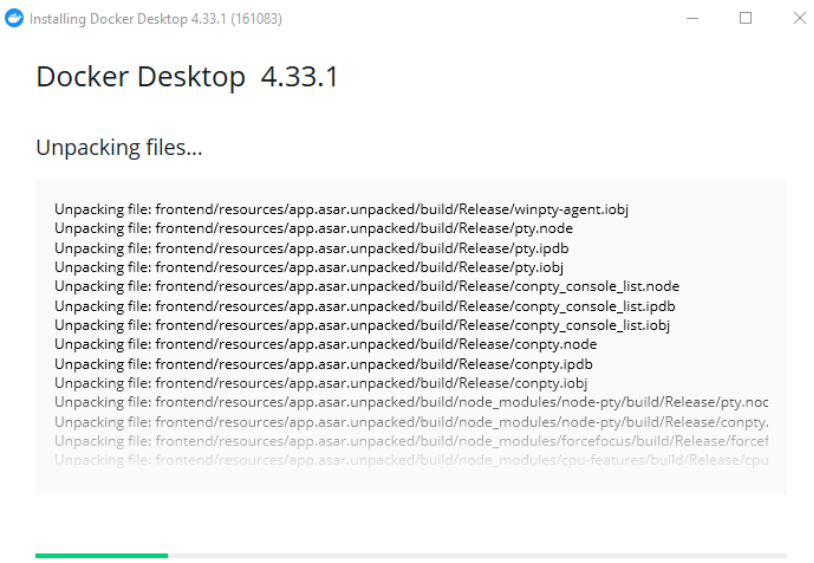
Docker está disponible en múltiples plataformas como Linux, macOS, y Windows. Para instalar Docker debemos dirigirnos a la página oficial: https://www.docker.com/



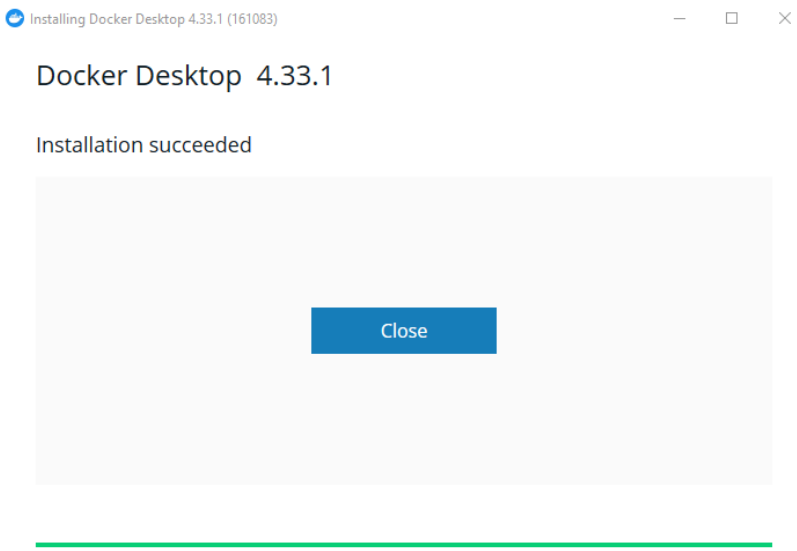

Una vez descargado debemos ejecutar el programa, el cual nos mostrará una configuración predeterminada, la cual nos recomiendo hacer uso de WS, que es un subsistema de Windows, para Linux.



Una vez aceptado se empezará a instalar todas las librearías y dependencias necesarias.

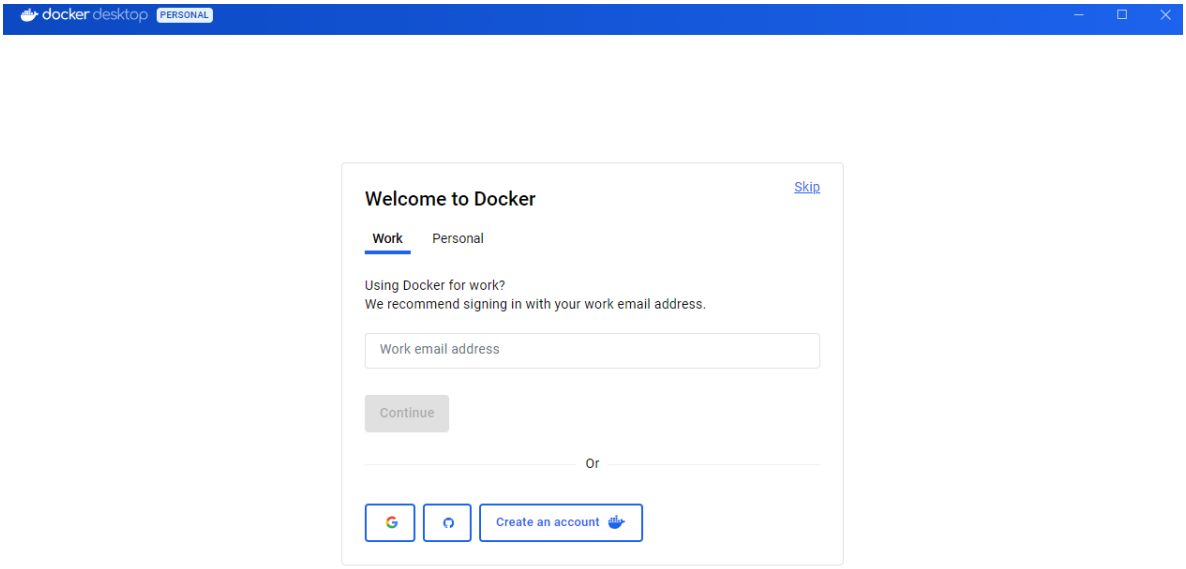



Una vez instaladas todas las librerías, se completa la instalación.

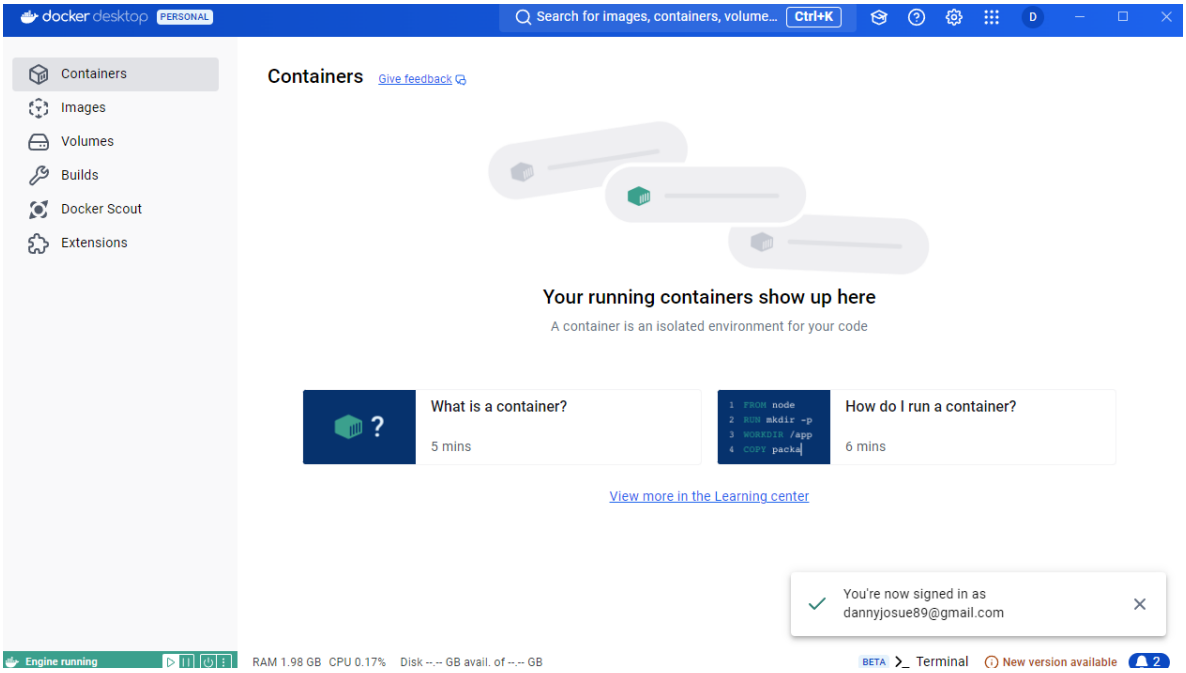



Después de esto se debe reiniciar el sistema para que todos los cambios sean aplicados de manera correcta.

Al iniciar el programa DockerHub nos solicitará iniciar sesión, esto lo podemos hacer fácilmente iniciando sesión directamente con Google.




Una vez se tenga la sesión iniciada, ya podremos acceder a todas las funcionalidades de DockerHub.



## Creación de Dockerfile

Para poder levantar una aplicación se debe crear un archivo Dockerfile, este debe contener una serie de instrucciones en forma de comandos para construir una imagen de Docker personalizada.

Para esto debemos dirigirnos a la carpeta raíz de nuestro proyecto y crear un archivo con el nombre ***Dockerfile,*** para este caso el ejemplo es tomando como base un código de Python.

```
# Usa una imagen base oficial de Python importada desde DockerFile
FROM python:3.9-slim

# Establecer el directorio de trabajo dentro del contenedor
WORKDIR /usr/src/app

# Copiar el archivo requirements.txt y las dependencias necesarias
COPY requirements.txt ./

# Instalar las dependencias necesarias
RUN pip install --no-cache-dir -r requirements.txt

# Copiar el resto del código fuente de la aplicación
COPY . .

# Exponer el puerto 5000 (puerto por defecto para Flask)
EXPOSE 5000

# Define el comando para correr la aplicación
CMD ["python", "app.py"]
```

Para crear una imagen a partir del Dockerfile debemos realizar los siguientes comandos:

Crear un archivo requirements.txt que contenga las dependencias del programa, en este caso Flask.

```
Flask==2.0.1
```

Para construir la imagen se ejecuta el siguiente comando en la carpeta raíz:

```
docker build -t aplicacion1.
```

Con esto ya tendríamos creada la imagen que luego nos permitirá subir a DockerHub.

## Obtener imágenes desde DockerHub

Para poder descargar imágenes desde DockerHub debemos abrir una terminal desde nuestra carpeta raíz y ejecutar los siguientes comandos:

```
Sdocker pull python:3.9-slim
```

Este comando realiza la descarga de una imagen base de Python en su versión 3.9

## Subir imágenes a DockerHub

Para poder crear y subir nuestras propias imagnes a DockerHub, debemos iniciar sesión con el siguiente comando.

```
docker login
```

Luego de ejecutar el comando nos solicitará nuestras credenciales, mismas que creamos al final de la instalación de la aplicación.

Una vez realizado esto, debemos darle un nombre a la imagen, esto se hace a través de una etiqueta:

```
docker tag aplicación1 usuarioDeDockerHub/aplicación1:latest
```

Para este caso etiquetamos la imagen con el nombre ***aplicacion1.***

Por último, para subir la imagen ejecutamos el siguiente comando, que no hace más que realizar un push a nuestro repositorio Docker.

```
docker push usuario_dockerhub/aplicación1:latest
```

## Correr imágenes desde DockerHub

Una vez tengamos nuestra imagen, ya sea creada o descargada desde Docker, podemos ejecutar el siguiente comando:

```
docker run -d -p 5000:5000 aplicación1
```

Este comando indica que el puerto 5000 del contenedor se configurará con el puerto 5000 de nuestra computadora, lo que hará posible ingresar a nuestra aplicación desde Http://localhost:5000.

Para poder visualizar las imágenes que estamos corriendo se usa el siguiente comando:

```
docker ps
```

Esto muestra la información de todos los contenedores activos.

## Visualizar la aplicación

### Cliente

Para visualizar que la aplicación este corriendo en el puerto indicado, lo hacemos desde el siguiente comando:

```
docker run -d -p 5000:5000 aplicacion1
```

### Servidor

Para poder ver que la aplicación esta operativa en una API debemos realizar una solicitud GET, la cual responde al siguiente comando:

```
curl http://localhost:5000/api
```

El cual accederá a un archivo api el cual debe contener un endpoint que devuelva una solicitud GET, la cual mostrar la respuesta en la terminal del servidor.

## Detener y eliminar contenedores e imágenes

Para detener un contenedor se realizar mediante el siguiente comando:

```
docker stop nombreDelContenedor
```

Para eliminarlo:

```
docker rm nombreDelContenedor
```

## Comando extra para Docker

Estos comandos no son esenciales para la creación de los contenedores, pero pueden ser útiles en caso de tener algún error el ejecutar el programa:

Ver detalles de una imagen, esto debe mostrar detalles como configuraciones, capas, entre otras características importantes:

```
docker inspect nombreDeLaImagen
```

Para ver las salidas que puede haber en un contenedor se utiliza el siguiente comando, esto es útil para la depuración de nuestra aplicación:

```
docker logs nombreDelContenedor
```

Para el reinicio del contenedor se hace mediante el siguiente comando:

```
docker restart nombreDelContenedor
```

Por último, para poder conectarse a un contenedor que se esta ejecutando, lo podemos realiza de la siguiente manera, esto permitirá acceder a líneas de comandos dentro del propio contenedor, esto es ideal para depuración:

```
docker exec -it nombreDelContenedor /bin/bash
```