



ESCUELA DE
INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



Día, Fecha:

Jueves, 31/10/2024

Hora de inicio:

10:40 - 12:20

Introducción a la Programación y Computación 2 [P]

Denilson Florentín de León Aguilar

Recordatorio Grabar Clase

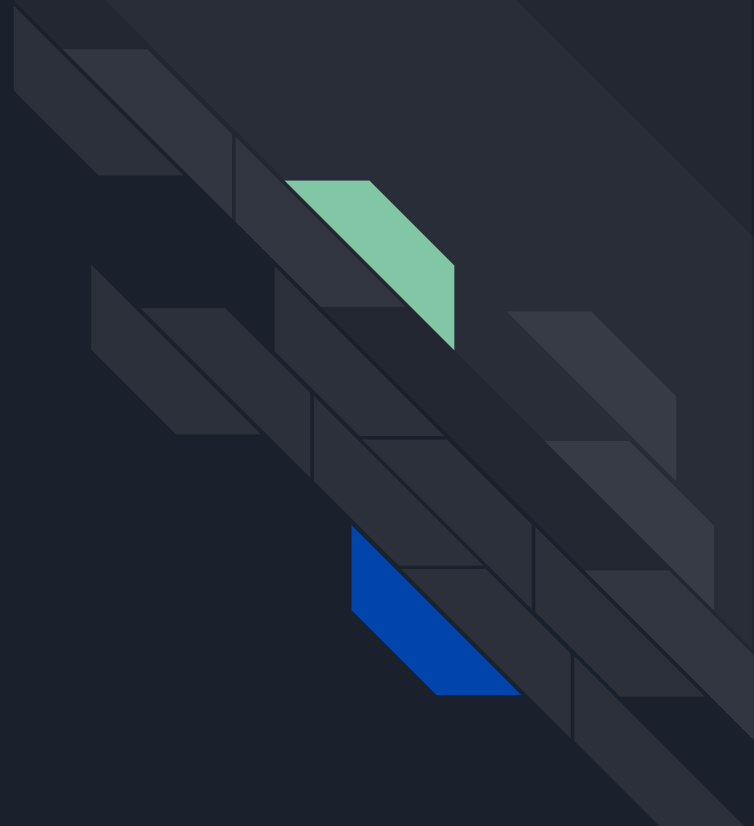


Contenido clase 14

- Notas finales
- Docker
- Docker-compose



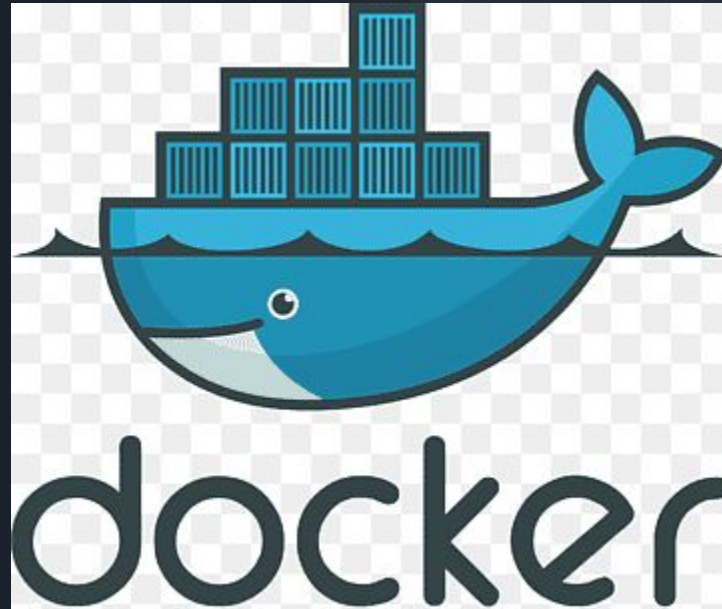
Notas Finales Publicadas



Felicidades por
finalizar el curso

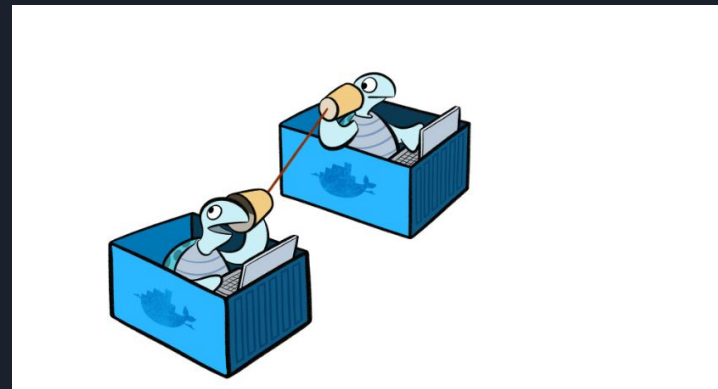


DOCKER



¿Qué es Docker?

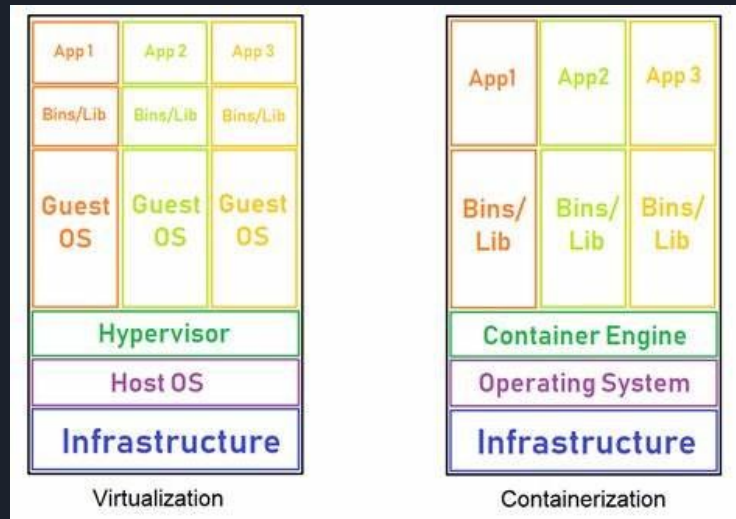
Docker es una plataforma de código abierto diseñada para simplificar la creación, implementación y ejecución de aplicaciones mediante contenedores. Los contenedores permiten empaquetar una aplicación y todas sus dependencias en una unidad estándar para el desarrollo y ejecución, garantizando que la aplicación se ejecute de la misma manera en cualquier entorno.



Diferencia entre Docker y Virtualización (VM)

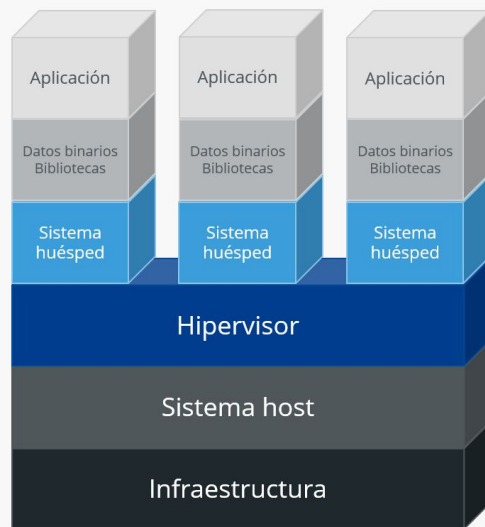
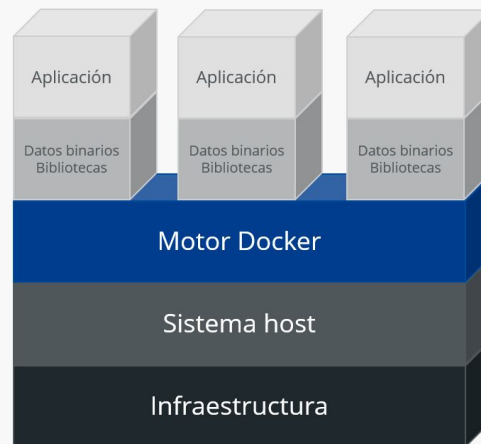
Docker: Utiliza la virtualización a nivel de sistema operativo para ejecutar contenedores. Los contenedores comparten el mismo kernel del sistema operativo anfitrión, lo que los hace más livianos y eficientes en términos de recursos.

Virtualización (VM): Utiliza la virtualización a nivel de hardware para ejecutar máquinas virtuales. Cada máquina virtual incluye su propio sistema operativo completo, lo que puede resultar en un uso más intensivo de recursos y una mayor complejidad en la gestión.



Diferencia entre Docker y Virtualización (VM)

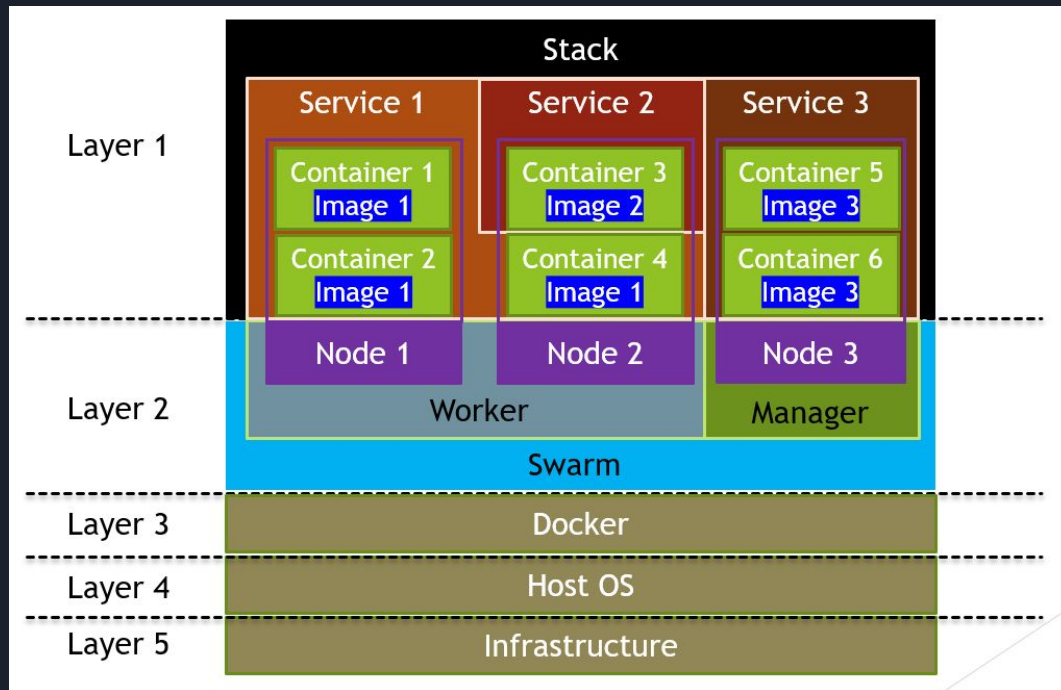
Comparación entre máquinas virtuales y contenedores de Docker

**IONOS****Máquinas Virtuales (VM)****Contenedores de Docker**

¿Para qué sirve Docker?

Desarrollo de Aplicaciones: Docker simplifica el proceso de desarrollo de aplicaciones al permitir a los desarrolladores crear, probar y distribuir aplicaciones de manera consistente y reproducible en cualquier entorno.

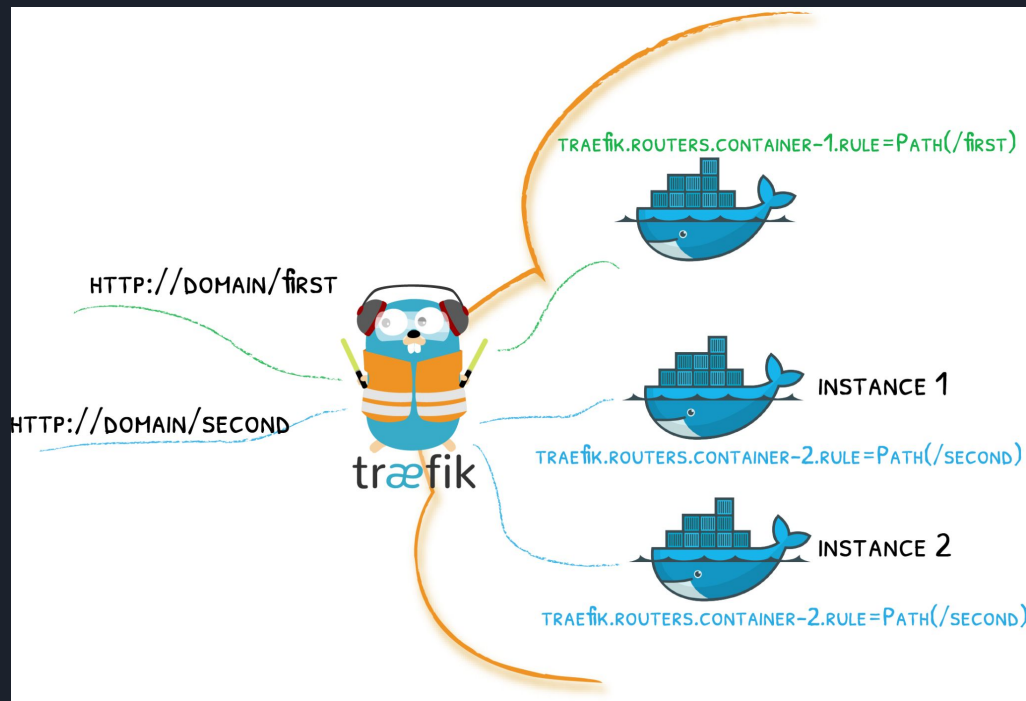
Implementación de Aplicaciones: Docker facilita la implementación de aplicaciones al proporcionar un entorno aislado y portátil para ejecutar aplicaciones en cualquier infraestructura, ya sea en la nube, en servidores locales o en máquinas virtuales.

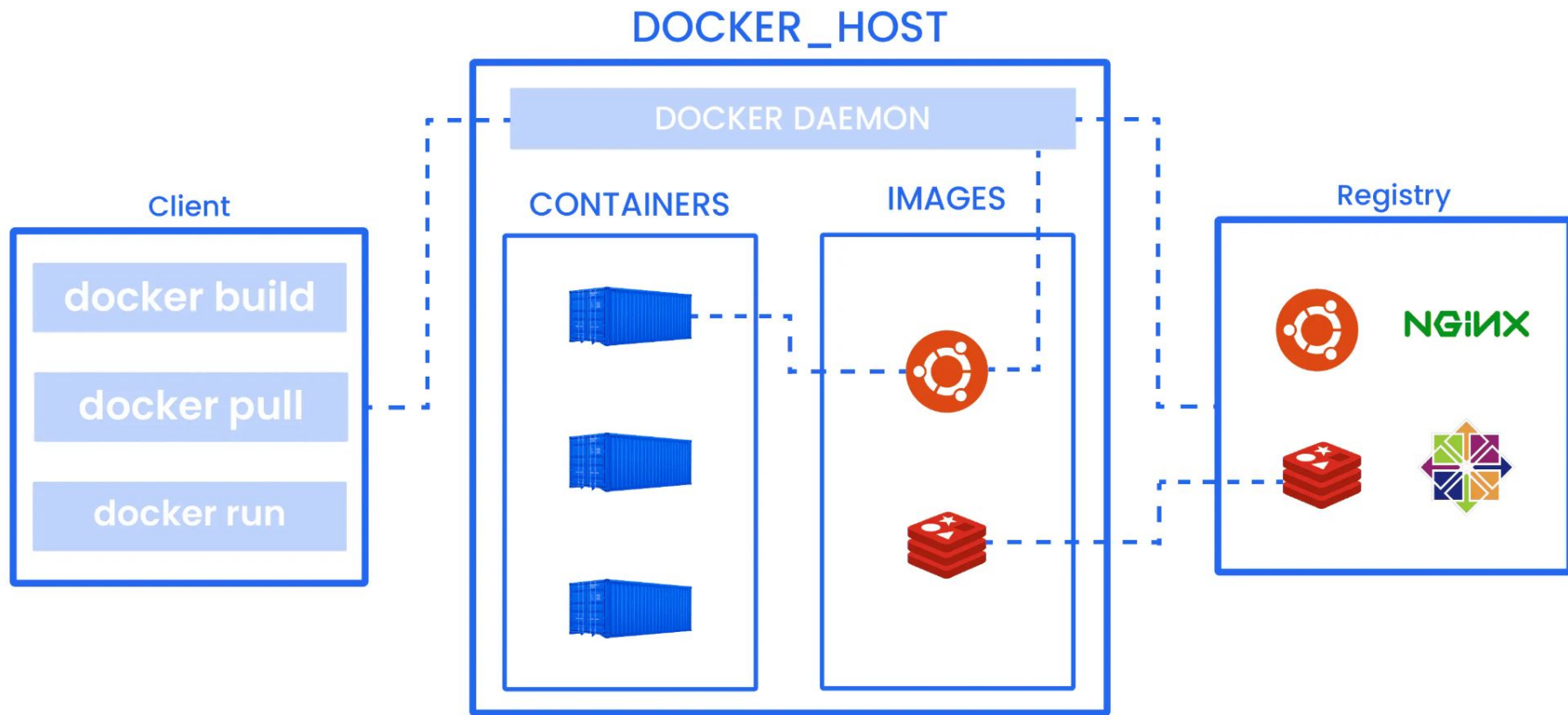


¿Para qué sirve Docker?

Escalabilidad: Docker facilita la escalabilidad horizontal de aplicaciones al permitir la creación rápida de múltiples instancias de contenedores para satisfacer la demanda variable de recursos.

Gestión de Infraestructura: Docker simplifica la gestión de infraestructura al permitir la automatización de la creación, configuración y despliegue de aplicaciones mediante herramientas de orquestación como Docker Compose y Kubernetes.







¿Cómo usar Docker?

Escalabilidad: Docker facilita la escalabilidad horizontal de aplicaciones al permitir la creación rápida de múltiples instancias de contenedores para satisfacer la demanda variable de recursos.

Gestión de Infraestructura: Docker simplifica la gestión de infraestructura al permitir la automatización de la creación, configuración y despliegue de aplicaciones mediante herramientas de orquestación como Docker Compose y Kubernetes.



1. Instalación de Docker

Antes de comenzar, asegúrate de tener Docker instalado en tu sistema. Puedes descargar e instalar Docker desde el sitio web oficial de Docker según el sistema operativo que estés utilizando.

Los pasos para ubuntu son:

- Agregar repositorios.
- Instalar con apt-get install
- Comprobar

```
# Add Docker's official GPG key:
```

```
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl
```

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
```

```
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
# Add the repository to Apt sources:
```

```
echo \
```

```
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]  
https://download.docker.com/linux/ubuntu \
```

```
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
```

```
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

1. Instalación de Docker

Instalamos una vez agregados los repository

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin  
docker-compose-plugin
```

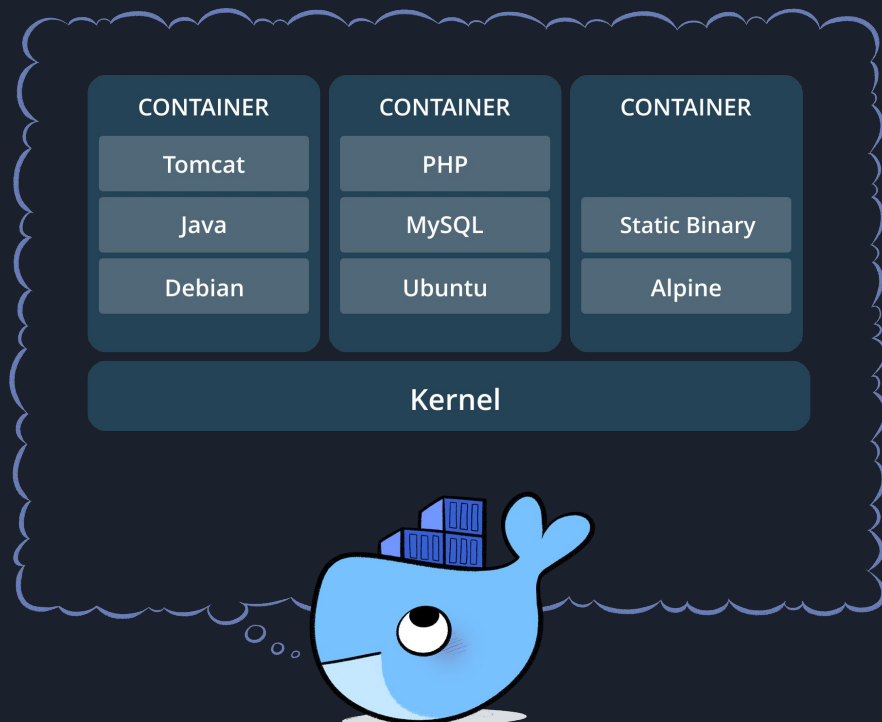
Comprobamos estado

```
sudo docker run hello-world
```


2. Creación de un Dockerfile

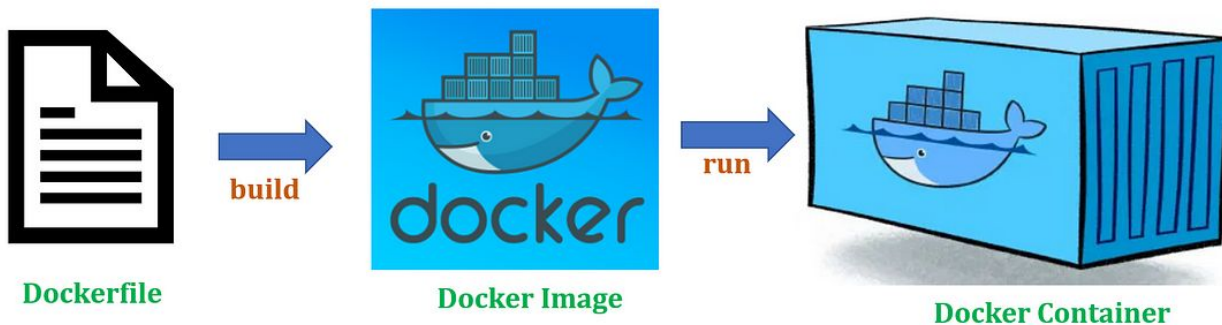
Un Dockerfile es un archivo de texto que contiene instrucciones para construir una imagen Docker. Para crear un Dockerfile, sigue estos pasos:

- Crea un nuevo archivo llamado Dockerfile en el directorio raíz de tu proyecto.
- Dentro del archivo, especifica la imagen base que deseas utilizar, las dependencias necesarias y los comandos de ejecución de tu aplicación.



2. Creación de un Dockerfile

Una vez teniendo el Dockerfile en el directorio a empaquetar, se ejecuta Build y se crea la imagen

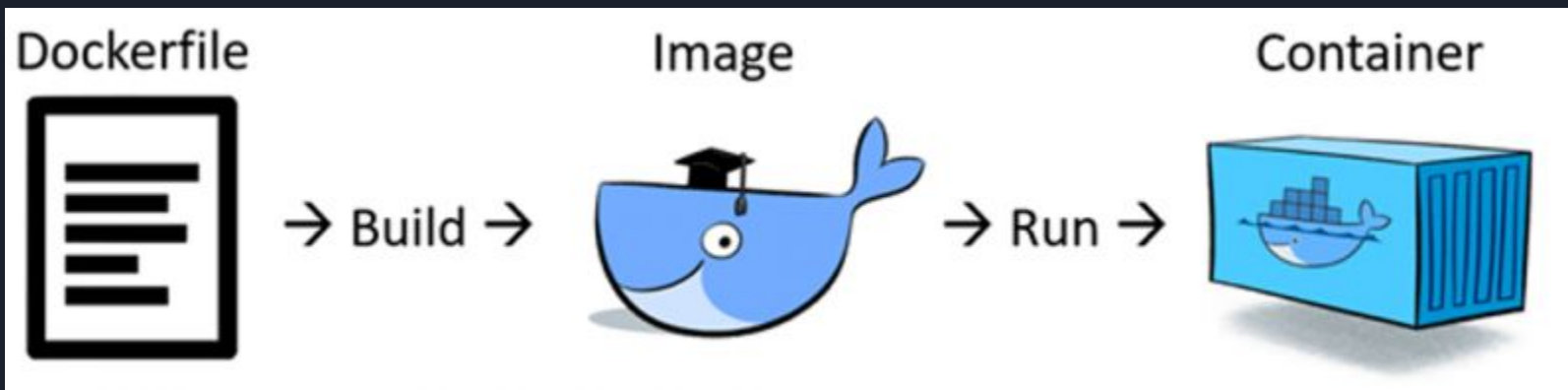


3. Construcción de la imagen Docker

Una vez que hayas creado el Dockerfile, puedes construir la imagen Docker ejecutando el siguiente comando en tu terminal:

```
docker build -t nombre_de_la_imagen .
```

Este comando construirá una nueva imagen Docker utilizando las instrucciones definidas en el Dockerfile y la etiquetará con el nombre especificado.

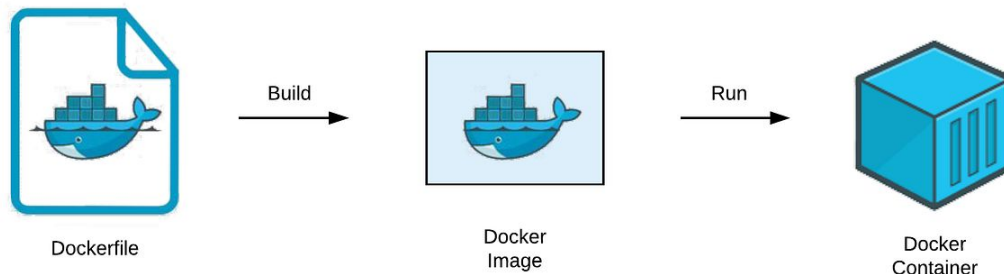


4. Ejecución de un contenedor Docker

Una vez que hayas construido la imagen Docker, puedes ejecutar un contenedor basado en esa imagen utilizando el siguiente comando:

```
docker run nombre_de_la_imagen
```

Este comando iniciará un nuevo contenedor basado en la imagen especificada y ejecutará la aplicación dentro del contenedor.



5. Gestión de contenedores Docker

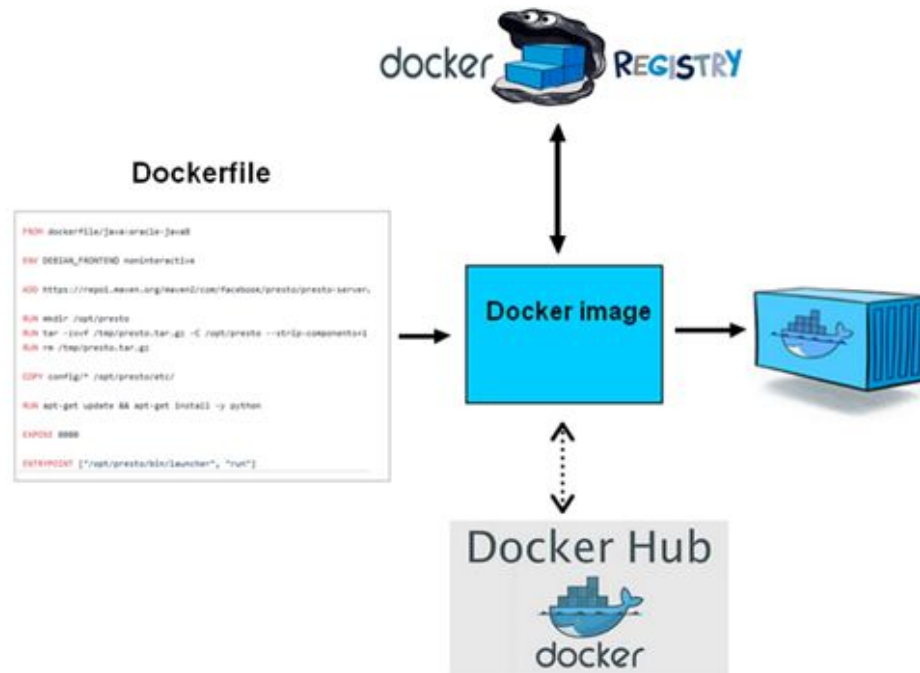
Puedes administrar contenedores Docker utilizando una variedad de comandos. Algunos de los comandos más comunes incluyen:

- **docker ps:** Muestra una lista de contenedores en ejecución.
- **docker stop <ID_del_contenedor>:** Detiene un contenedor en ejecución.
- **docker rm <ID_del_contenedor>:** Elimina un contenedor.
- **docker logs <ID_del_contenedor>:** Muestra los registros de salida de un contenedor.

```
docker ps
docker stop <ID_del_contenedor>
docker rm <ID_del_contenedor>
docker logs <ID_del_contenedor>
```

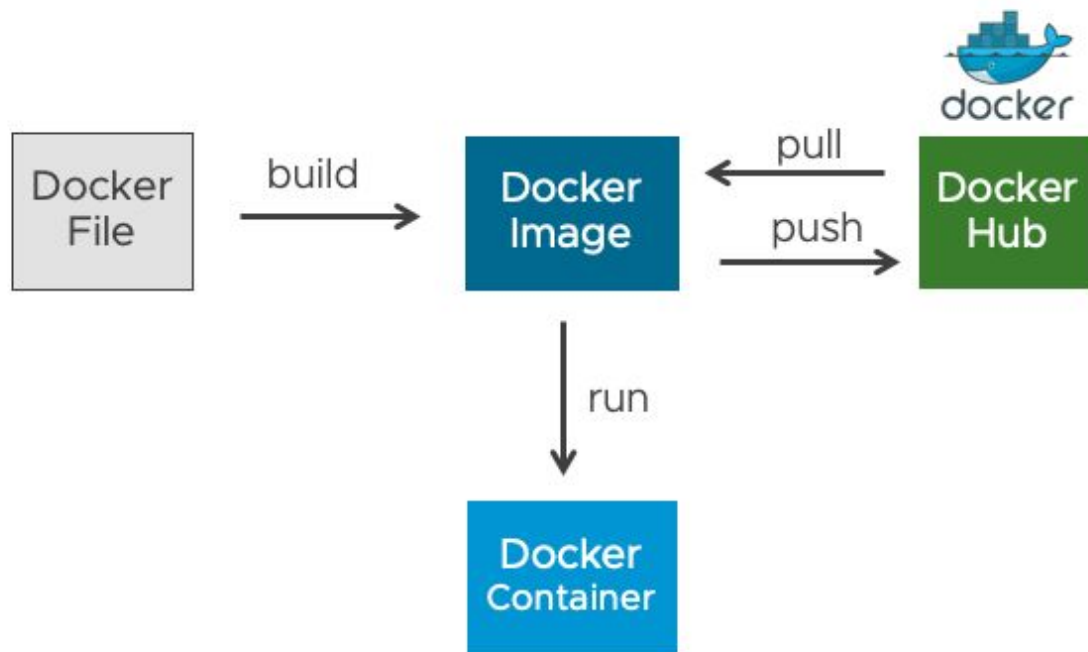
6. Distribución de imágenes Docker

Una vez que hayas construido una imagen Docker, puedes distribuirla a otros usuarios compartiéndola en un registro de Docker público o privado, como Docker Hub.



7. Monitoreo y depuración

Docker proporciona herramientas para monitorear y depurar contenedores en ejecución. Puedes usar comandos como `docker stats` para monitorear el uso de recursos de tus contenedores y `docker exec` para ejecutar comandos dentro de un contenedor en ejecución.

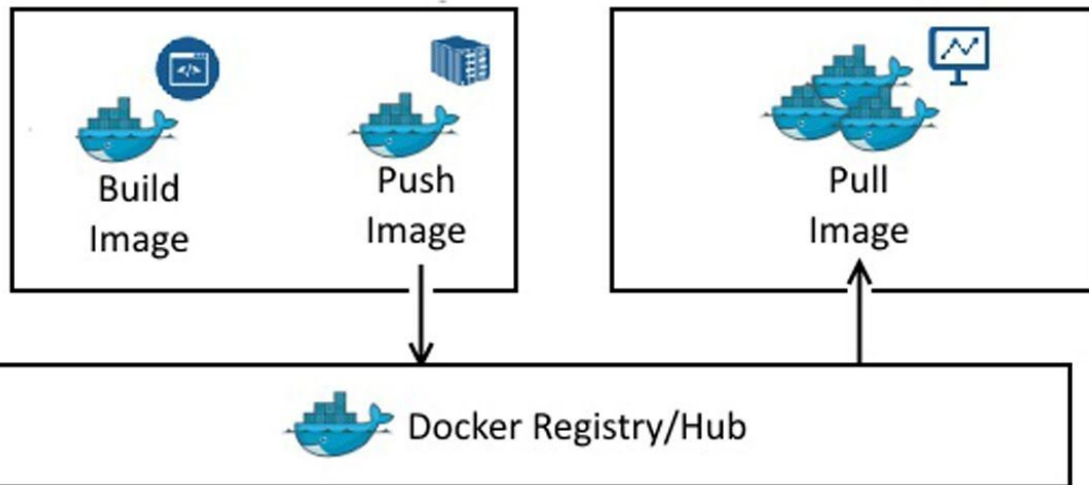


Recordatorio Captura de pantalla

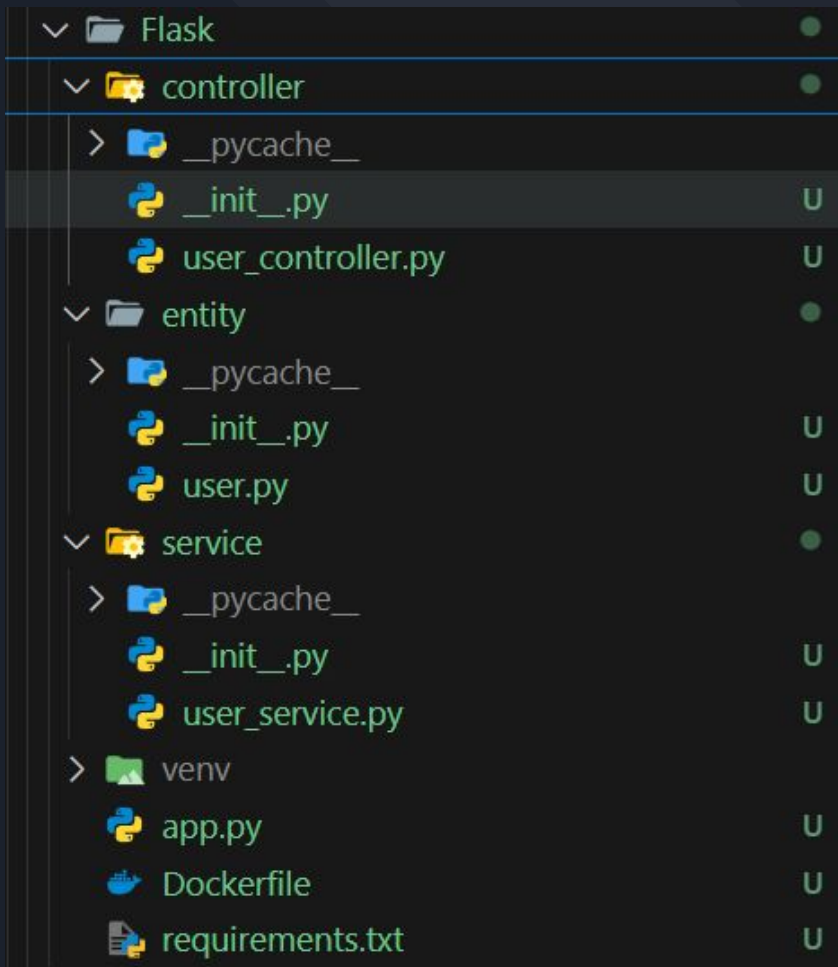


Ejemplo Docker

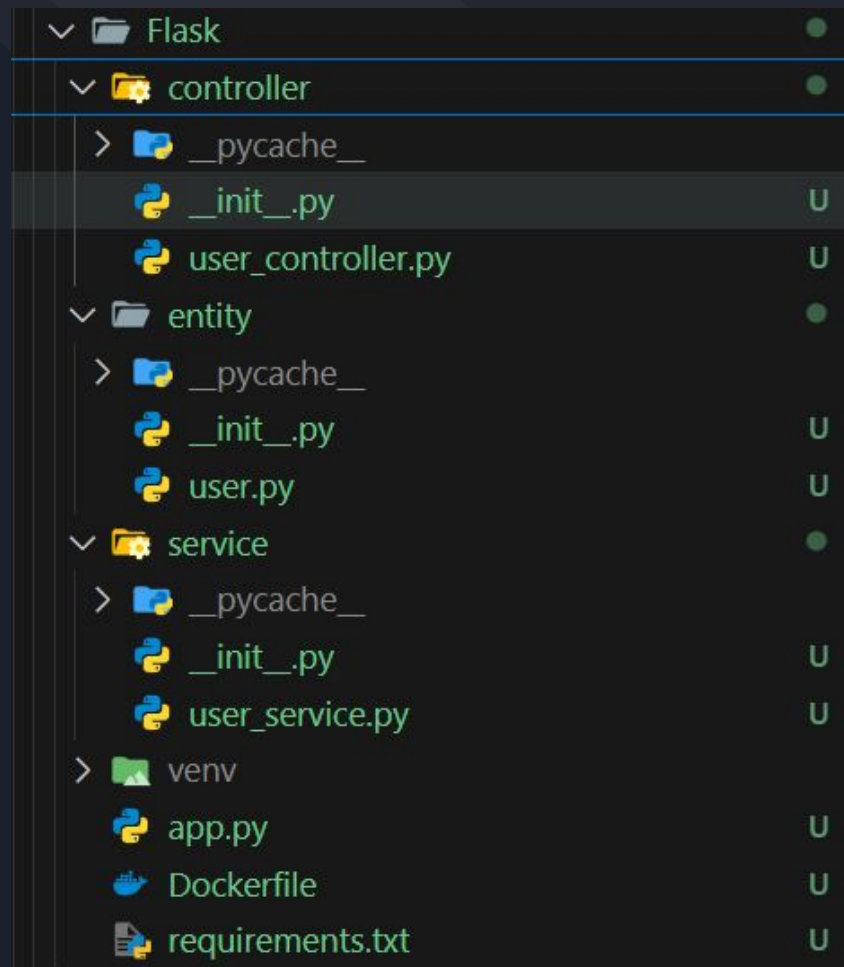
How to Build Images in Docker (Containerization/Dockerization)



Ejemplo Docker



El ejemplo es un servidor de Flask. Se trata de un registro de usuarios, se recomendó una buena organización de módulos y carpetas para un desarrollo ordenado. Además, se adjuntó el archivo Dockerfile.

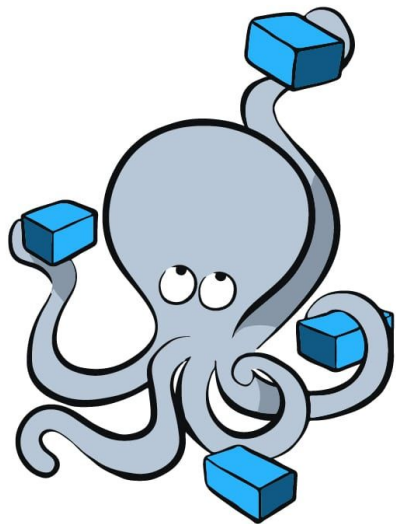


Docker-compose

<https://docs.docker.com/compose/>



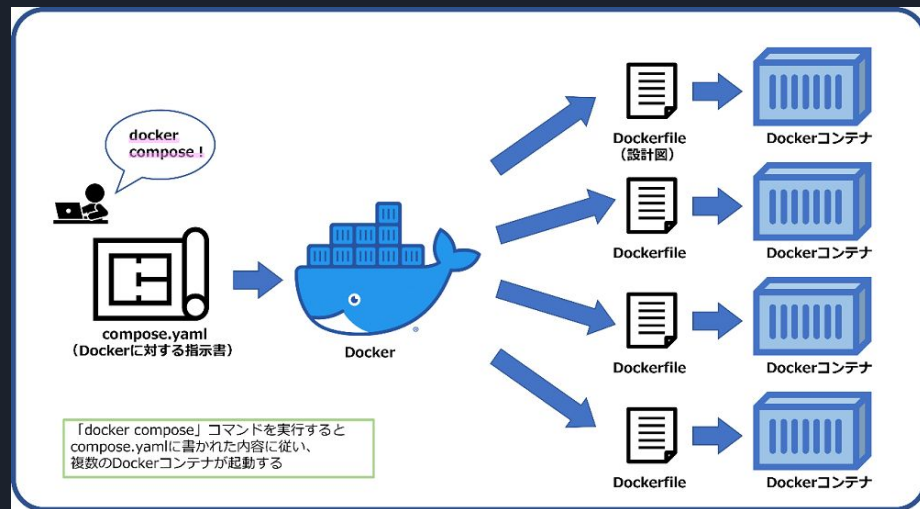
Docker-compose



docker
Compose

Docker-compose

Docker Compose es una herramienta que facilita la ejecución y administración de múltiples contenedores Docker a la vez. Permite definir, en un solo archivo, los diferentes servicios que forman parte de una aplicación (como bases de datos, aplicaciones web, servidores de caché, etc.). En lugar de ejecutar contenedores manualmente uno por uno, Docker Compose se encarga de levantar todos los servicios que se hayan definido en el archivo de configuración.





1. ¿Cómo funciona Docker-compose?

Docker Compose utiliza un archivo de configuración (generalmente llamado `docker-compose.yml`) en el que se describe cada uno de los servicios que componen la aplicación. Dentro de este archivo, especificas:

- Las imágenes Docker de cada contenedor.
- La configuración de red entre ellos.
- Variables de entorno y volúmenes compartidos.
- Dependencias entre servicios (por ejemplo, si una base de datos debe estar en funcionamiento antes de lanzar la aplicación principal).

Al usar Docker Compose, la herramienta lee el archivo `docker-compose.yml` y ejecuta cada uno de los contenedores necesarios en el orden especificado, logrando así que todos trabajen de manera integrada y sincronizada.



2. ¿Cómo lo hace Docker Compose?

Docker Compose se encarga de la creación y la orquestación de múltiples contenedores a través de los siguientes pasos clave:

Definición del Archivo de Configuración (`docker-compose.yml`):


- El archivo de configuración permite describir cada servicio de la aplicación, como el tipo de imagen, puertos, redes y volúmenes.

Creación y Ejecución de Contenedores:

- Una vez definido el archivo, con el comando `docker-compose up`, Docker Compose crea y ejecuta todos los contenedores especificados. Docker Compose analiza las dependencias y levanta cada contenedor en el orden necesario para asegurar la conexión entre ellos.

Gestión del Ciclo de Vida de Contenedores:

- Con comandos como `docker-compose down`, puedes detener y eliminar contenedores y redes creados por Docker Compose. Además, permite escalabilidad de los servicios (por ejemplo, puedes replicar servicios con `docker-compose up --scale servicio=n`).



3. Escenarios en los que Docker Compose es útil

Docker Compose es especialmente valioso en escenarios como:

Desarrollo de Aplicaciones Multiservicio:

- Ideal para aplicaciones que requieren varios componentes, como una API, una base de datos y un servidor de caché. En lugar de configurar estos servicios de forma individual, Docker Compose permite gestionarlos en conjunto.

Pruebas de Integración:

- Docker Compose es muy útil para crear entornos de prueba que replican entornos de producción, permitiendo hacer pruebas de integración entre distintos componentes de la aplicación.

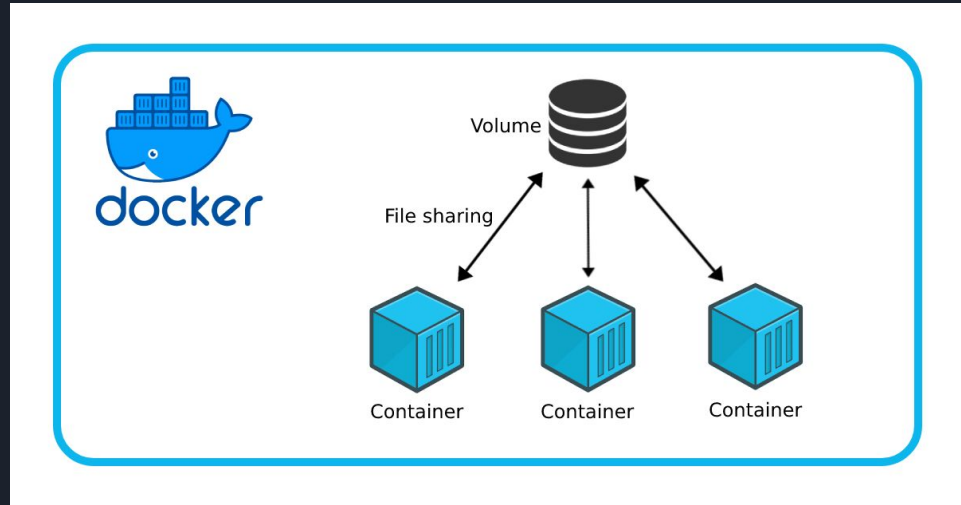
Entornos de Pruebas Locales o en CI/CD:

- En el ciclo de desarrollo, sobre todo en entornos de Integración Continua (CI), Docker Compose permite lanzar rápidamente un entorno de pruebas completo.

3. Escenarios en los que Docker Compose es útil

En este ejemplo, se tienen 3 contenedores sobre el mismo sistema backend, las cuales podrían recibir y ejecutar las peticiones del sistema principal.

Además, se visualiza que todos ellos comparten un mismo volumen que se encarga de administrar la información que necesitan, por lo que si un contenedor falla, puede ser reemplazado sin afectar al sistema.



4. Ejemplo básico de Docker Compose

Para una aplicación con una API en Flask y una base de datos PostgreSQL, el archivo docker-compose.yml podría verse así:

Aquí, definimos dos servicios:

- **Web:** Que usa una imagen de la API en Flask y expone el puerto 5000.
- **DB:** Un servicio PostgreSQL configurado con variables de entorno para el usuario, la contraseña y el nombre de la base de datos.

Para ejecutar estos contenedores, solo necesitarías ejecutar en la terminal:

```
docker-compose up
```

```
yaml

version: '3'
services:
  web:
    image: myflaskapp:latest
    ports:
      - "5000:5000"
    depends_on:
      - db
  db:
    image: postgres:latest
    environment:
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword
      POSTGRES_DB: mydatabase
```



5. Conclusiones sobre Docker-Compose

Docker Compose facilita el manejo de aplicaciones complejas, ofreciendo una forma ordenada y escalable de manejar contenedores que dependen unos de otros. Además, te ayuda a reproducir entornos de forma consistente, agilizando el desarrollo y pruebas de tus aplicaciones.