1. Introducción a Docker Compose:

Docker Compose es una herramienta que te permite definir y gestionar aplicaciones Docker de múltiples contenedores. Te permite describir la pila completa de la aplicación, incluyendo múltiples servicios, redes y volúmenes, en un único archivo YAML. Esta introducción teórica te ayudará a comprender los conceptos fundamentales de Docker Compose.

1.1. Simplificando las Aplicaciones de Múltiples Contenedores:

- Muchas aplicaciones modernas están compuestas por varios servicios, cada uno ejecutándose en su propio contenedor. Estos servicios a menudo necesitan comunicarse e interactuar entre sí.
- Docker Compose simplifica el proceso de gestionar y coordinar estas aplicaciones de múltiples contenedores al permitirte definir y controlar toda la pila con un único archivo de configuración.

1.2. Archivo YAML de Docker Compose:

- Se utiliza un archivo YAML de Docker Compose (***docker-compose.yml***) para definir los componentes de una aplicación de múltiples contenedores. Especifica servicios, redes, volúmenes y otras configuraciones.
- El archivo YAML es legible por humanos y te permite definir las relaciones, dependencias y ajustes para cada servicio en la pila de tu aplicación.

1.3. Servicios:

- Un servicio es una unidad de una aplicación definida en el archivo YAML de Docker Compose. Corresponde a un contenedor y define cómo debe configurarse y ejecutarse el contenedor.
- Cada servicio en el archivo ***docker-compose.yml*** puede tener su propia imagen, variables de entorno, puertos, volúmenes y otras configuraciones.

1.4. Redes:

- Docker Compose te permite crear redes personalizadas para tus servicios. Las redes permiten la comunicación entre servicios al mismo tiempo que los aíslan del tráfico externo.
- Los servicios dentro de la misma red pueden comunicarse utilizando los nombres de los servicios como nombres de host.

1.5. Volúmenes:

- Los volúmenes en Docker Compose te permiten persistir y compartir datos entre contenedores. Se utilizan para almacenar datos que deben persistir más allá del ciclo de vida del contenedor.
- Los volúmenes son especialmente útiles para contenedores de bases de datos o contenedores con archivos de configuración importantes.

1.6. Ventajas de Docker Compose:

- **Simplicidad:** Docker Compose abstrae la complejidad de gestionar múltiples contenedores, facilitando la definición y el mantenimiento de pilas de aplicaciones complejas.
- **Reproducibilidad:** Con un único archivo docker-compose.yml, puedes definir toda la pila de la aplicación, garantizando la consistencia en diferentes entornos.
- **Portabilidad:** Las configuraciones de Docker Compose pueden ser versionadas y compartidas, lo que permite a los equipos colaborar y asegurarse de que la misma aplicación se ejecute de manera consistente en todas partes.
- **Aislamiento:** Los servicios en Docker Compose se ejecutan en contenedores aislados, proporcionando un aislamiento a nivel de proceso mientras permiten una comunicación fluida.

1.7. Casos de Uso:

- Docker Compose es especialmente útil para entornos de desarrollo y pruebas, donde necesitas crear rápidamente toda una pila de aplicaciones para pruebas locales o depuración.
- También es valioso para crear configuraciones repetibles en pipelines de integración continua y despliegue.

1.8 Verificación de la Instalación de Docker Compose:

Después de instalar Docker Desktop, deberías tener Docker Compose disponible automáticamente. Para verificar la instalación, abre una terminal y ejecuta el siguiente comando:

```
docker-compose --version
```

Este comando mostrará la versión de Docker Compose instalada si la instalación fue exitosa.

In [1]:
```
!docker-compose --version
```

```
Docker Compose version 2.19.0
```

Resumen:

Docker Compose simplifica la gestión de aplicaciones de múltiples contenedores al proporcionar una manera declarativa de definir y orquestar servicios, redes y volúmenes. Es una herramienta valiosa tanto para desarrolladores como para equipos de DevOps, ya que permite optimizar el despliegue y prueba de aplicaciones complejas.

2. Ejecuta Tu Primer Contenedor de Docker:

Paso 1: Crea un Directorio para tu Proyecto:

Abre una terminal y crea un nuevo directorio para tu proyecto de clúster Dask:

In [ ]:
```
ls
```

In [ ]:
```
!mkdir dask-cluster
```

In [ ]:
```
!cd dask-cluster
```

Paso 2: Crea un Archivo docker-compose.yml:

Dentro de tu directorio de proyecto, crea un archivo llamado **docker-compose.yml** utilizando un editor de texto de tu elección. Este archivo definirá la configuración de tu clúster Dask.

Paso 3: Define Servicios en docker-compose.yml:

Agrega el siguiente contenido a tu archivo docker-compose.yml:

In [18]:
```
!cat dask-cluster/docker-compose.yml
```

```
version: '3'
services:
  scheduler:
    image: daskdev/dask:latest
    command: ["dask-scheduler"]
    ports:
      - "8786:8786"

  worker:
    image: daskdev/dask:latest
    command: ["dask-worker", "scheduler:8786"]
    depends_on:
      - scheduler
```

Esta configuración define dos servicios: **scheduler** y **worker**.

- El servicio ***scheduler*** utiliza la imagen ***daskdev/dask*** y ejecuta el programador de Dask mediante el comando dask-scheduler. Expone el puerto 8786 para la comunicación.
- El servicio ***worker*** también utiliza la misma imagen y ejecuta un worker de Dask mediante el comando ***dask-worker***. Depende del servicio ***scheduler*** y se conecta al puerto 8786 del programador.

Paso 4: Inicia el Clúster Dask:

En tu terminal, navega hasta el directorio de tu proyecto y ejecuta el siguiente comando para iniciar el clúster Dask:

In [4]:
```
!docker ps
```

```
CONTAINER ID   IMAGE        COMMAND                  CREATED             STAT
US             PORTS                                NAMES
5f593c3d2104   httpd        "httpd-foreground"       About an hour ago   Up A
bout an hour   0.0.0.0:80->80/tcp                   6webarchitecture-apache-1
82582c8cd29c   phpmyadmin   "/docker-entrypoint.…"   About an hour ago   Up A
bout an hour   0.0.0.0:8080->80/tcp                 6webarchitecture-phpmyadmi
n-1
3e8fc19f89ef   mysql        "docker-entrypoint.s…"   About an hour ago   Up A
bout an hour   33060/tcp, 0.0.0.0:3307->3306/tcp    6webarchitecture-mysql-1
```

In [5]:
```
!docker-compose up
```

```
no configuration file provided: not found
```

Ejecuta el comando `docker-compose up` con la opción `-f` o `--file` seguida de la ruta hacia tu archivo docker-compose.yml.

In [ ]:
```
!docker-compose -f dask-cluster/docker-compose.yml up
```

```
[+] Running 0/0
 ⠋ worker Pulling
0.1s
 ⠋ scheduler Pulling
0.1s
[+] Running 0/2
 ⠙ worker Pulling
0.2s
 ⠙ scheduler Pulling
0.2s
[+] Running 0/2
 ⠹ worker Pulling
0.3s
 ⠹ scheduler Pulling
0.3s
[+] Running 0/2
 ⠸ worker Pulling
0.4s
 ⠸ scheduler Pulling
```

Docker Compose will pull the necessary images and start the scheduler and worker containers

Paso 5: Verificar el clúster:

Una vez que hayas ejecutado el comando `docker-compose up` , puedes comprobar el estado del clúster Dask visitando la dirección `http://localhost:8786` en tu navegador web. Esto te llevará al panel de control del clúster Dask, donde podrás supervisar y administrar tus trabajadores y tareas.

In [ ]:
```
!docker ps
```

Paso 6: Detener el Clúster:

Para detener el clúster Dask, presiona **Ctrl+C** en la terminal donde iniciaste el clúster con `docker-compose up .`

Paso 7: Limpiar:

Si deseas eliminar los contenedores y la red creados por Docker Compose, ejecuta el siguiente comando en el directorio de tu proyecto:

In [21]:
```
!docker-compose down
```

```
no configuration file provided: not found
```

Resumen:

¡Listo! Has creado una configuración de Docker Compose para un clúster Dask en Python. Esta configuración te permite gestionar y escalar fácilmente tu clúster Dask para tareas de cómputo distribuido.

In [ ]:
```

```