Docker Compose

Explicación de práctica 5

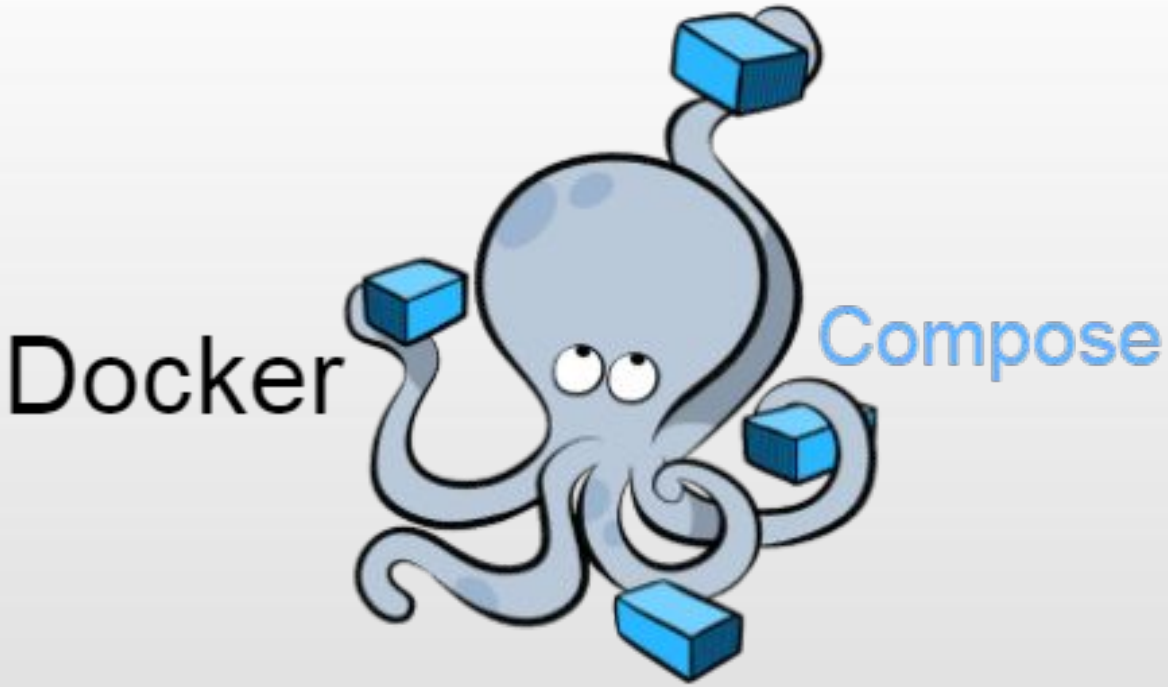
Sistemas Operativos

Facultad de Informática
Universidad Nacional de La Plata

2024

Agenda

1. Docker compose



¿Qué es docker compose?

Como vimos, docker permite empaquetar y ejecutar una aplicación/proceso en contenedores.

Por ahora vimos cómo desplegar UN contenedor que puede correr UN proceso determinado, o UN servicio determinado.

¿Qué es docker compose?

- Los sistemas modernos suelen estar formados por varios componentes que se intercomunican entre sí.
- En la arquitectura de Microservicios esto se lleva al extremo.
- Es habitual usar containers Docker para ejecutar cada componente.

Ejemplo básico:

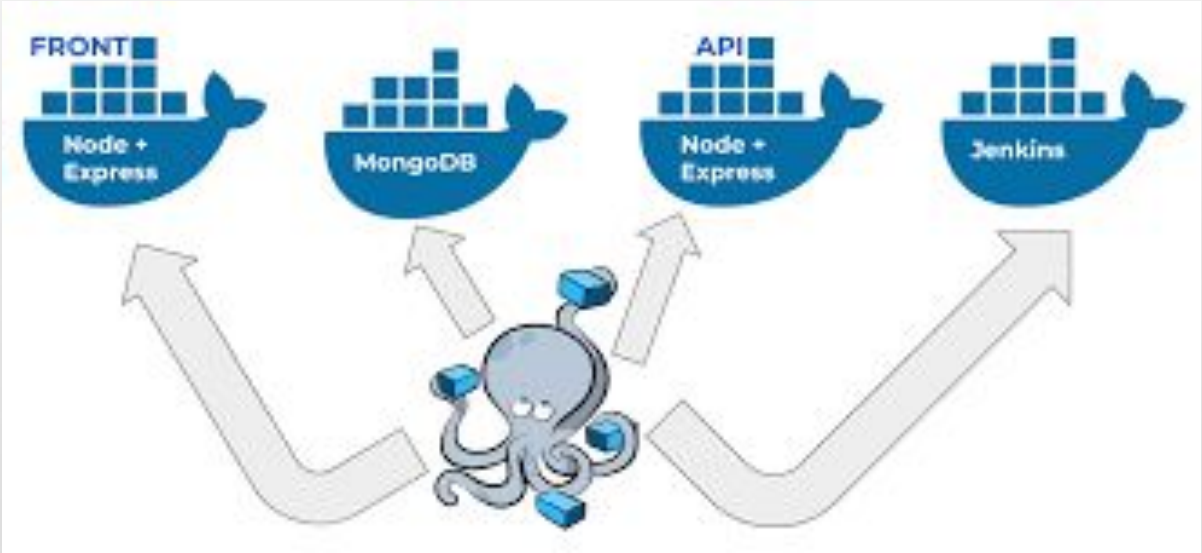
App + Base de datos.

Por lo tanto, para desplegar una app "dockerizada" generalmente voy a necesitar desplegar un conjunto de contenedores.

¿Qué es docker compose?

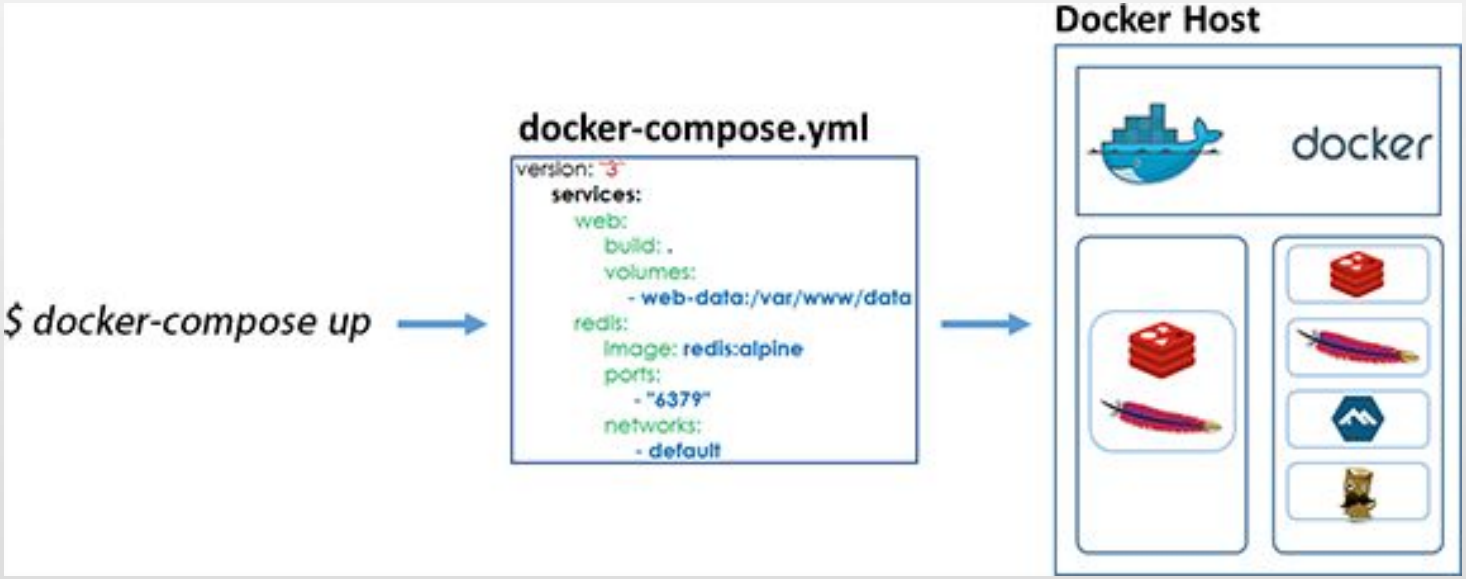
Docker-compose es una herramienta que facilita el despliegue de aplicaciones compuestas por múltiples contenedores.

Este conjunto de contenedores se comunican entre sí para brindar la funcionalidad objetivo.



¿Qué es docker compose?

Consideramos entonces a docker-compose como el conjunto de la herramienta (el binario) y los archivos de configuración llamados "archivo compose" o "compose file" que definen los recursos deseados.



En la actualidad existen 2 formas de utilizar docker compose.

1. Con la aplicación Docker Desktop
2. Utilizando el plugin Compose

En los ejemplos utilizaremos el Compose Plugin

Más información en
https://docs.docker.com/compose/install/

- Actualmente Compose es un plugin de Docker. Es la herramienta que se invoca para interpretar los "compose file" y desplegar los contenedores allí definidos.

- El "compose file" es un archivo escrito en el lenguaje de marcado YAML que describe las características de los contenedores a desplegar. Usualmente llamado docker-compose.yml.

Toda la configuración se define en el archivo compose!

- Cada componente se denomina “service”
- Un service:
  - Define uno o más containers con la misma imagen y configuración.
  - Se conecta con otros services a través de “networks”.
  - Almacenan datos persistentes en “volumes”.
  - Por default name = domain name

https://docs.docker.com/compose/compose-application-model/

Estructura básica de un archivo docker-compose.yml o `compose.yml`:

```yaml
version: "3.9"
services:
  web:
    build: .
    ports:
      - "8000:5000"
    volumes:
      - .:/code
    environment:
      FLASK_ENV: development
  redis:
    image: "redis:alpine"
```

Actualmente existen tres versiones para el formato de archivo de docker-compose.yml:

- Versión 1: Está obsoleta.
- Versión 2.x
- Versión 3.x: Es la última versión y la recomendada por Docker.

Las versiones 2.x y 3.x comparten estructura pero no algunas opciones. Veremos la sintaxis de la V3.

Cuando se invoca el binario compose, este busca en la carpeta desde donde es invocado algún archivo con nombre docker-compose.yml o compose.yml.

También se le puede pasar como parámetro cuál es el "compose file" a utilizar.

En base al contenido del "compose file", se iniciarán los distintos contenedores con las características definidas para cada uno.

Comandos básicos

- Verificar la versión instalada: **docker compose -v**
- Crear todos los contenedores: **docker compose create**
- Iniciar todos los contenedores: **docker compose up**
- Frenar todos los contenedores: **docker compose down**
- Iniciar contenedores en background:
  docker compose up -d
- Listar los contenedores iniciados: **docker compose ps**
- Eliminar todos los contenedores: **docker compose rm**
- Ver los logs de los contenedores: **docker compose logs**

Simplicidad: Todo definido en un archivo con una estructura determinada y documentada.

Replicabilidad: Si necesito volver a desplegar los servicios, simplemente corro el mismo archivo.

"Shareability": Facilidad de compartir, solo necesito proveer el "compose file" (NOTA: las imágenes docker deben estar accesibles)

Versionable: Facilidad de usar control de versiones (por ej git).

Ventajas docker-compose

Vamos a comparar cómo iniciar un contenedor `mysql` en la versión 5.7, utilizando variables de entorno para los nombres de usuario, nombre de base de datos y contraseñas, publicando el puerto "`3306`" en el host, que use un volumen nombrado "`data`" para los datos de la bbdd ("`/var/lib/mysql`") y que tenga como política de reinicio "`always`" y nombre "`database`".

Comparemos cómo hacerlo usando directamente docker y docker-compose.

Con docker:

Primero tengo que crear el volumen ”data”:

```
~$ docker volume create data
```

Luego puedo iniciar el contenedor:

```
~$ docker run --name=database --publish=3306:3306 \
--restart always -v data:/var/lib/mysql \
-e “MYSQL_ROOT_PASSWORD=rootpassword” \
-e “MYSQL_DATABASE=database” \
-e “MYSQL_USER=user” \
-e “MYSQL_PASSWORD=password”  \
mysql:5.7
```

en cambio, con docker-compose:

docker-compose.yml:

```yaml
version: "3.9"
services:
  database:
    image: mysql:5.7
    volumes:
      - data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: database
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    ports:
      - 3306:3306
volumes:
  data: {}
```

y en el directorio ejecutar:

docker compose up.

Pensemos,

- ¿Qué es más claro?
- ¿Qué es más replicable?
- ¿Qué es más fácil de compartir?
- ¿Qué es "más" versionable?
- ¿Y si en lugar de solo 1 contenedor, tuviera que iniciar 5 definiendo todo para cada uno?

Referencias

- https://docs.docker.com/compose/install/
- https://github.com/docker/compose
- https://docs.docker.com/compose/compose-file/
- https://docs.docker.com/compose/reference/
- https://docs.docker.com/compose/gettingstarted/

¿Preguntas?


200
DOSCIENTOS
PESOS
BANCO CENTRAL DE LA
REPÚBLICA ARGENTINA
200
200