<codoa codo/>

Agencia de Aprendizaje a lo largo de la vida

DJANGO
Clase 34

Docker

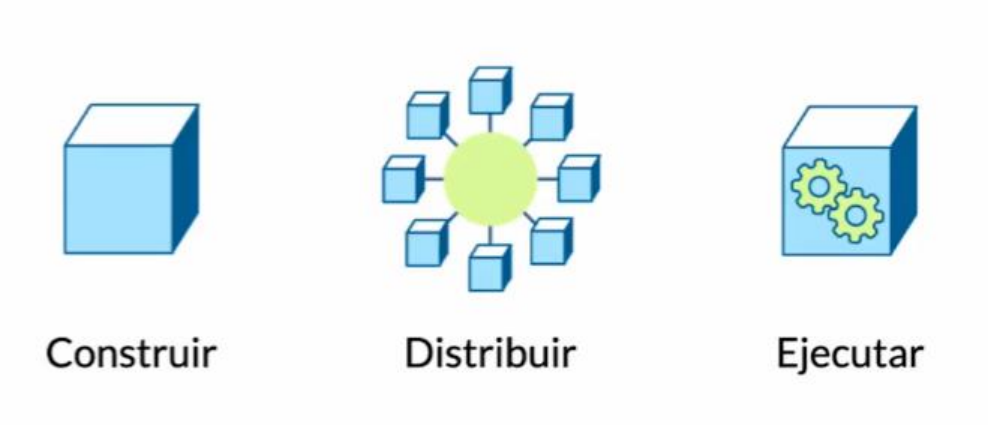
BA

Les damos la bienvenida

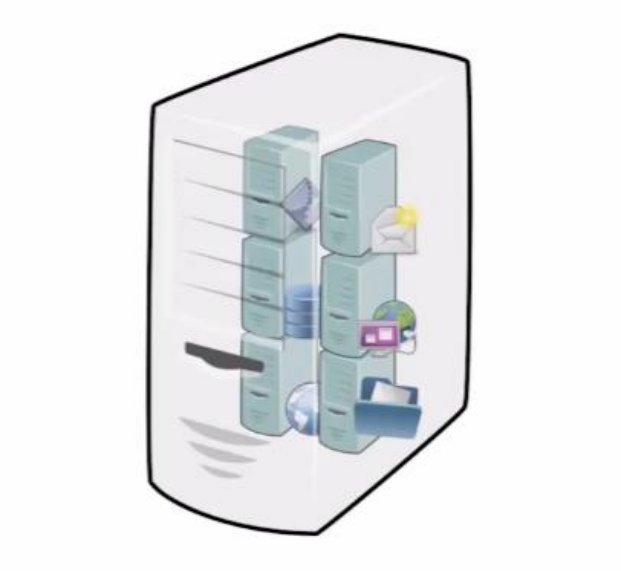
Vamos a comenzar a grabar la clase

Problemas en el desarrollo de software

Existen 3 grandes problemas



Virtualización



Una posible solución, aunque presenta los siguientes problemas

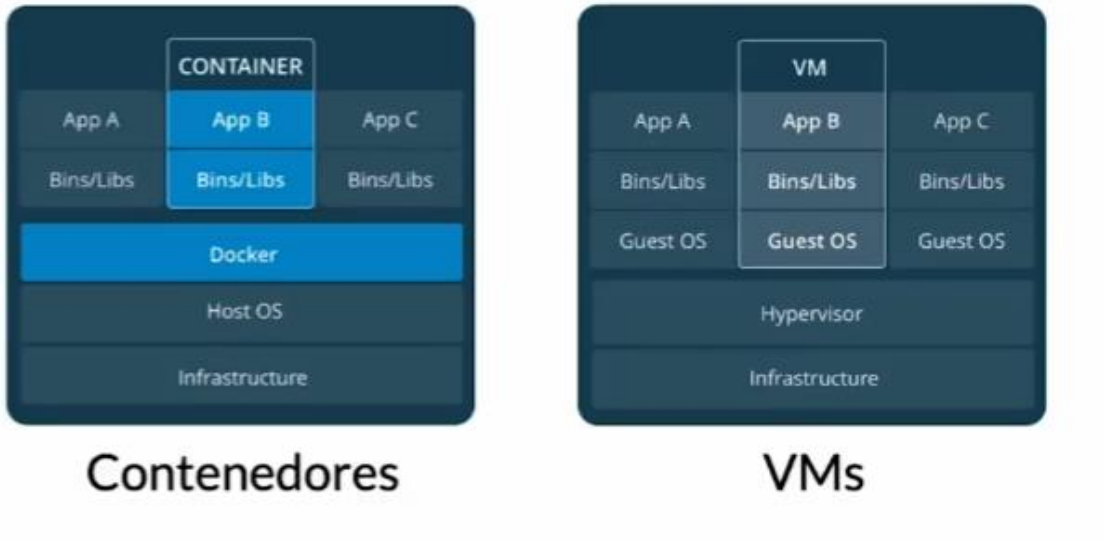
- Peso
- Costo de administración
- Múltiple formatos

Containerización

El empleo de contenedores para construir y desplegar software

- Flexibles
- Livianos
- Portables
- Bajo acoplamiento
- Escalables
- Seguros

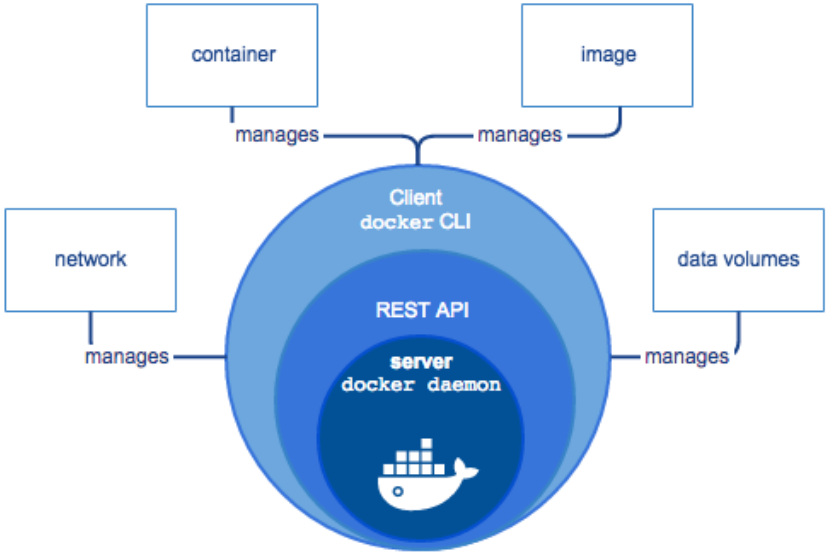
Contenedores vs VMs



¿Qué es Docker?

Docker es una plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas **contenedores** que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución.

¿Cómo funciona Docker?



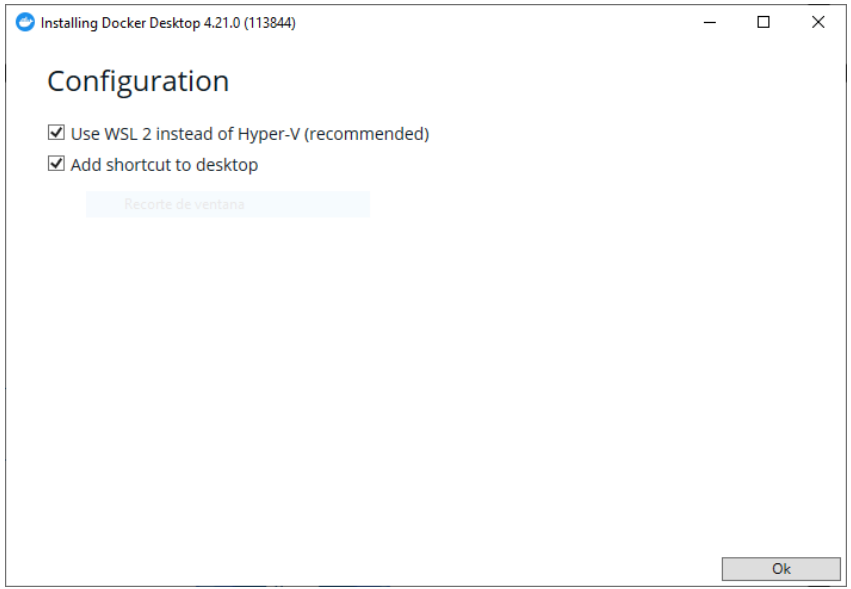
Instalación de Docker

Descarga desde https://www.docker.com/ el instalador dependiendo de tu sistema operativo.

En Windows 10 necesitamos habilitar Hyper-V
https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v

Por último, es recomendable crear un cuenta en Docker Hub
https://hub.docker.com/

Instalación de Docker (Windows 10)



Destildar la opción “Use WSL 2” si que no se tiene habilitada WSL en la instalación de Windows.

Comandos

Crear contenedores desde línea de comandos
docker run hello-world
docker ps (muestra los contenedores activos)
docker ps –a (muestra todos los contenedores)
docker run --name hello-cac hello-world (crea un contenedor llamado hello-cac a partir de la imagen hello-world)
docker rm <ID o nombre> (borro un contenedor)
docker rm -f <ID o nombre> (borro si esta corriendo)

Exponemos contendores
docker run -d -p 33060:3306 --name mysql-db -e MYSQL_ROOT_PASSWORD=secret mysql
(Se crea un contenedor de mysql, -p indica el puerto del anfitrión:puerto del contenedor)
docker exec -it mysql-db mysql –p (podemos ingresar al contenedor y ejecutar el comando mysql como usuario root)

Comandos - continuación

Creamos volúmenes
docker volume ls (listo los volumes)
docker volume create dbdata (creo un volumen, reserva un espacio del disco del anfriotion)
#Se crea un contenedor con un volumen de datos montado
docker run -d -p 33060:3306 --name mysql-db -e MYSQL_ROOT_PASSWORD=secret -v dbdata: /var/lib/mysql mysql

Redes
docker network ls (listo las redes)
docker network create --atachable cacnet (creo la red cacnet que pueda ser alcanzada por otros contenedores)
docker network connect cacnet db (conecto el contenedor “mysql-db” a la red “cacnet”)

Dockerfile

Dockerfile, nos permite crear imágenes con código nuestro, que nos va a permitir luego crear contendores. Creamos este archivo en el directorio de nuestro proyecto.

```
#version de la imagen queremos crear el contenedor
FROM python:3.9

#Especificamos un directorio de trabajo, es como hacer un mkdir y cd
WORKDIR /proyecto_23319

#Copiamos el archivo requirements dentro del contenedor
COPY requirements.txt /proyecto_23319/requirements.txt

#Ejecutamos la instalación de las dependecias dentro del contenedor
RUN pip install --no-cache-dir --upgrade -r /proyecto_23319/requirements.txt

#Copiamos el resto del contenido de este directorio al workdir del contenedor
COPY . /proyecto_23319/

#Exponemos el puerto 8000 para que sea vinculable con el anfitrion
EXPOSE 8000
```

docker-compose.yml

```yaml
#version del docker composer que vamos a utilizar
version: '3.8'
#especifiamos que servicios queremos que tengan, son los distintos componentes que tiene nuestra aplicacion
services:
  # cada servicio podria parecerse un contenedor, pero un servicio puede contener uno o más contenedores de la misma imagen
  proyecto_23319:
    build:
      context: .
      dockerfile: Dockerfile
    # indicamos las dependencias entre los servicios
    depends_on:
      - mysql-db
    #podemos indicar el comando principal del contenedor
    command: bash -c "python manage.py makemigrations && python manage.py migrate && python manage.py runserver 0.0.0.0:8000"
    ports:
      - "8090:8000"
    #especificamos los volumens del servicio
    volumes:
      #quiero que se monte en la ruta actual del proyecto, en el directorio del contenedor, permite que cuando se hagan
      #cambios en el proyecto, el contendor tome los cambios tambien
      - .:/proyecto_23319

  mysql-db:
    image: mysql
    #podemos definir variables de entorno
    environment:
      - MYSQL_ROOT_PASSWORD=cac2023
    #exponer los puertos
    ports:
      - "33060:3306"
    volumes:
      - dbdata:/var/lib/mysql

volumes:
  dbdata:
    driver: local
```

Nos permite escribir en forma declarativa la arquitectura de servicios que nuestra aplicación necesita, como deben comunicarse entre sí, como hay que manejar los archivos, en un pequeño archivo donde nosotros declaramos lo que queremos que pase y Docker por detrás corre todos los comandos.

Comandos – docker-compose

Estos comandos deben ejecutarse en el directorio donde se encuentra el archivo docker-compose.yml

docker-compose up –d (para poner en ejecución los contenedores, intermente crea las imágenes y luego los contenedores de acuerdo a lo especificado en el archivo docker-compose.yml)
docker-compose stop (para detener en ejecución de los contenedores)
docker-compose star (para poner en ejecución a los contenedores)
docker-compose down (para detener en ejecución y eliminar los contenedores)
docker-compose ps (lista los contenedores en ejecución)

No te olvides de completar la asistencia y consultar dudas

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

TODO EN EL AULA VIRTUAL