Fundamentos de Docker

Clase 7 – 21/3/2024

Bootcamp de DevOps @ Código Facilito


YAML

👋

Alejandra Bricio

cloud, scripting, k8s
@alebricio

- Site Reliability Engineer @ PayPal
- AWS Solutions Architect – Associate
- AWS Jalisco User Group – Leader
- AWS Community Day Mexico – Organizadora
- Tutora Bootcamp de Ciencias Computacionales @ Código Facilito
  - Redes Computacionales
  - Sistemas Distribuidos

Fundamentos de Docker – Agenda

1. ¿Qué es Docker? ¿Para qué sirve?
2. Docker vs Máquinas Virtuales
3. Instalando Docker localmente
4. Imágenes vs Contenedores
5. Registros Públicos y Privados
6. Iniciando Contenedores
7. Creando imágenes de Docker (Dockerfile)
8. Comandos principales
9. Versionamiento de Imágenes
10. Flujo de trabajo con Docker

¿Qué es docker ?

¿Por qué fue creado?

¿Qué problema resuelve?

¿Qué es  docker® ?

- Software de virtualización
- Permite **desarrollar** y **desplegar** aplicaciones mucho más fácil
- Empaqueta aplicaciones con todas las
  - dependencias necesarias,
  - configuración,
  - herramientas de sistema y
  - “runtime”
- Artifacto portable, fácil de distribuir y compartir

¿Qué es docker ?

¿Por qué fue creado?

¿Qué problema resuelve?

¿Por qué fue creado?

¿Qué problema resuelve?

¿Cómo era el desarrollo antes de los contenedores?

¿Cómo era el desarrollo antes de los contenedores?

Cada desarrollador tiene que instalar y configurar los servicios directamente en su máquina

- Configuraciones locales inconsistentes
- Conflictos entre versiones de dependencias.
- Procesos manuales y sujetos a errores.

Desarrollo con contenedores

postgres
redis
…

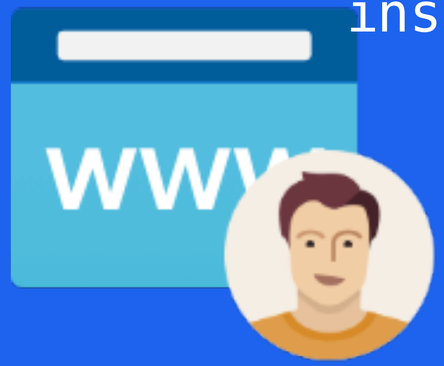
Desarrollo con contenedores

- Entornos consistentes y reproducibles.
- Dependencias encapsuladas y versionadas por contenedor.
- Eficiencia en uso de recursos.

postgres
redis
…

Operaciones antes de contenedores

- Artifacto
- Instrucciones de instalación



Development → **Operations**

Operaciones después de contenedores

Development → Operations

Virtualización

Virtualización

OS Capa Aplicaciones

OS Kernel

Hardware

OS = Sistema Operativo

Virtualización

OS Capa Aplicaciones

OS Kernel

Hardware

OS = Sistema Operativo

docker®

OS Capa Aplicaciones

OS Kernel

Hardware

OS Capa Aplicaciones

OS Kernel

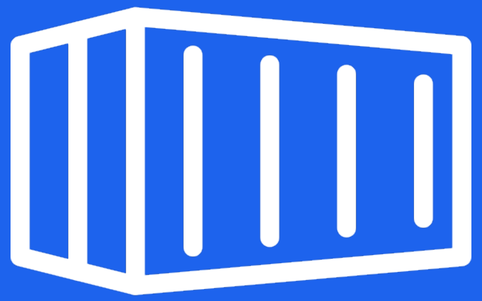
Hardware


VM
VM
App A
App B
OS
OS
OS Host
Máquina física


Aplicaciones contenerizadas
App A
App B
docker
OS Host
Máquina física

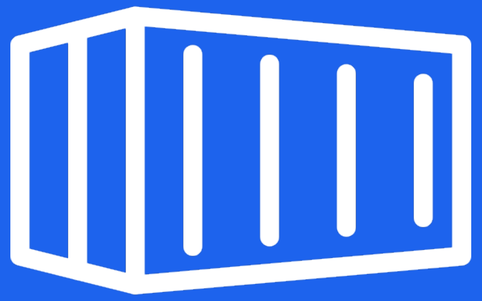
Uso de recursos	Alto	Bajo
Escalabilidad	Velocidad de inicialización: minutos	Velocidad de inicialización: segundos

Instalando  docker®



Imágenes vs Contenedores

imagen



contenedor

Definición	Plantilla inmutable para crear un contenedor.	Instancia en ejecución de una imagen.
Estado	Inmutable, no cambia una vez creada.	Mutable, cambia durante su vida útil.
Uso	Usada para construir contenedores.	Ejecuta aplicaciones en entornos aislados.
Almacenamiento	Almacenada en un registro Docker, como Docker Hub.	Existe solo mientras está en ejecución.
Creación	Se crea a través de un archivo Dockerfile.	Se crea al instanciar una imagen.

Registros públicos y privados

Ejecutar un contenedor

1. git clone https://github.com/docker/welcome-to-docker
2. Inspeccionar Dockerfile
3. cd /path/to/welcome-to-docker/
4. docker build -t welcome-to-docker .
5. Ejecutar contenedor Docker Desktop
   1. Puerto 8089
6. Explorar frontend: http://localhost:8089

Ejecutar una imagen de Docker Hub

1. Abrir Docker Desktop

2. Ejecutar contenedor
Imagen: docker/welcome-to-docker
Puerto: 8088

3. Explorar contenedor

4. Detener contenedor

docker commands

Categoría	Comando	Descripción
Imágenes	docker pull <nombre-imagen>:<tag>	Descargar una imagen desde Docker Hub.
	docker images	Listar imágenes descargadas.
	docker rmi <nombre-imagen>:<tag>	Eliminar una imagen local.
Contenedores	[opciones] <nombre-imagen>	Crear y arrancar un contenedor.
	docker ps	Listar contenedores en ejecución.
	docker ps -a	Listar todos los contenedores.
	docker stop <nombre-contenedor>	Detener un contenedor.
	docker rm <nombre-contenedor>	Eliminar un contenedor detenido.
Volúmenes	docker volume create <nombre-volumen>	Crear un volumen.
	docker volume ls	Listar volúmenes.
	docker volume rm <nombre-volumen>	Eliminar un volumen.

docker build

Opción	Formato del Comando	Descripción
Etiqueta	docker build -t mi-imagen:tag .	Construye una imagen y la etiqueta con un nombre y tag específicos.
Dockerfile	docker build -f /path/to/a/Dockerfile .	Especifica un Dockerfile diferente al predeterminado ('./Dockerfile').
Variable de Entorno	docker build --build-arg VAR=valor .	Construye una imagen y pasa variables de entorno en tiempo de construcción.
No Cache	docker build --no-cache .	Construye una imagen sin usar la caché de las capas para todas las instrucciones en el Dockerfile.
Pull	docker build --pull .	Siempre intenta extraer una versión más reciente de la imagen base antes de construir.

docker run

Opción	Formato del Comando	Descripción
Nombre	`--name <nombre>`	Asigna un nombre específico al contenedor.
Detached	`-d <imagen>`	Ejecuta el contenedor en background (modo detached).
Puertos	`-p <host>:<contenedor> <imagen>`	Mapea un puerto del host a un puerto del contenedor.
Variables de Entorno	`-e "VAR=valor" <imagen>`	Establece una variable de entorno en el contenedor.
Volumen	`-v <host-dir>:<cont-dir> <imagen>`	Monta un volumen desde el host en el contenedor.
Interactivo	`-it <imagen>`	Ejecuta el contenedor en modo interactivo (permite entrada de teclado).

Ejercicio

1. git clone https://github.com/alebricio/getting-started-docker.git

Fundamentos de Docker

1. ¿Qué es Docker? ¿Para qué sirve?
2. Docker vs Máquinas Virtuales
3. Instalando Docker localmente
4. Imágenes vs Contenedores
5. Registros Públicos y Privados
6. Iniciando Contenedores
7. Creando imágenes de Docker (Dockerfile)
8. Comandos principales
9. Versionamiento de Imágenes
10. Flujo de trabajo con Docker

@alebricio