



# Despliegue de Aplicaciones: Taller de Docker (1)

Evolución y Gestión de la Configuración

Paquetes (deb, msi, rpm...)

VirtualEnv

Contenedores

VM

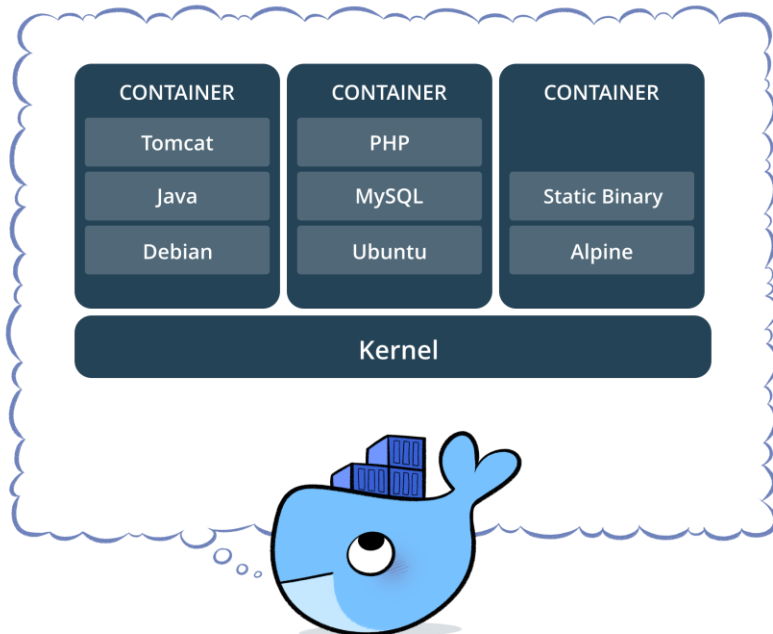
Permite tener "instalaciones" de módulos y paquetes Python de manera simultanea

Con Contenedores aislamos dependencias más allá de python

Permiten aislar todas las dependencias del sistema

Overhead y aislamiento

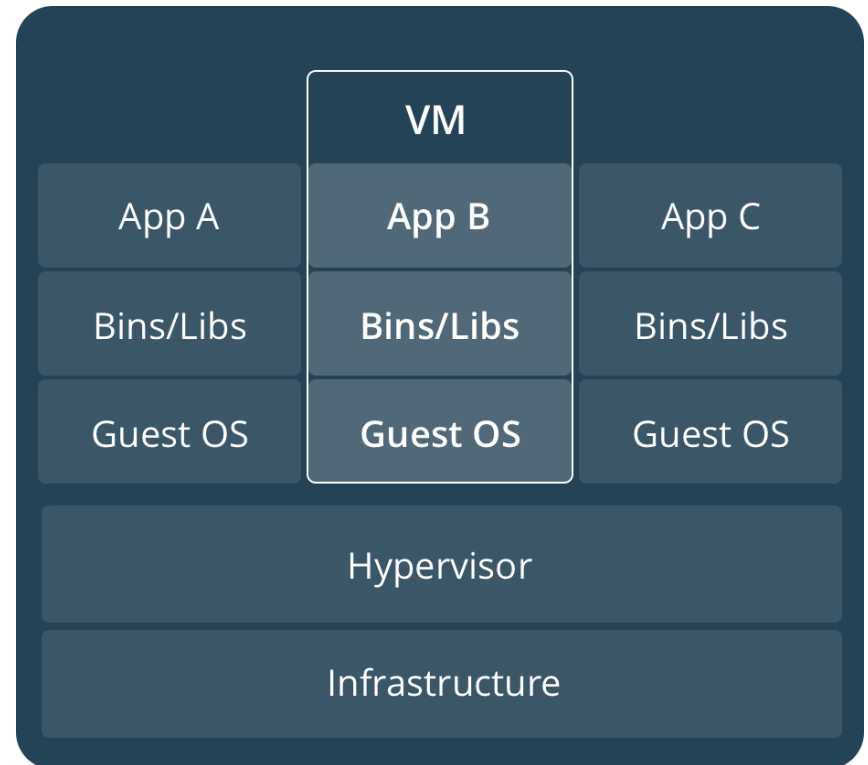
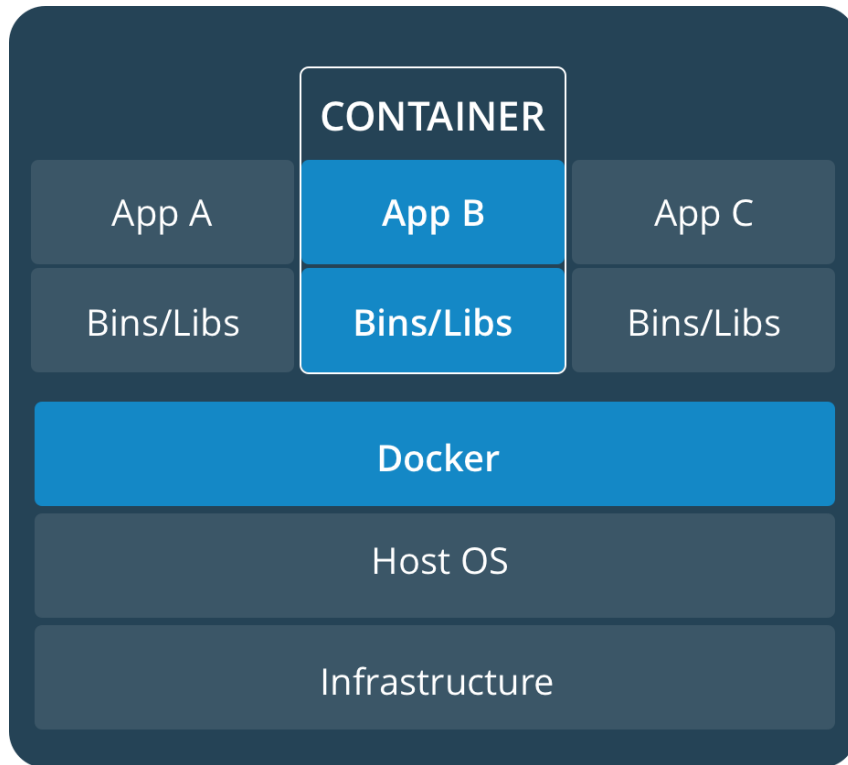
# INTRODUCCIÓN A CONTENEDORES



## ¿Qué es un contenedor?

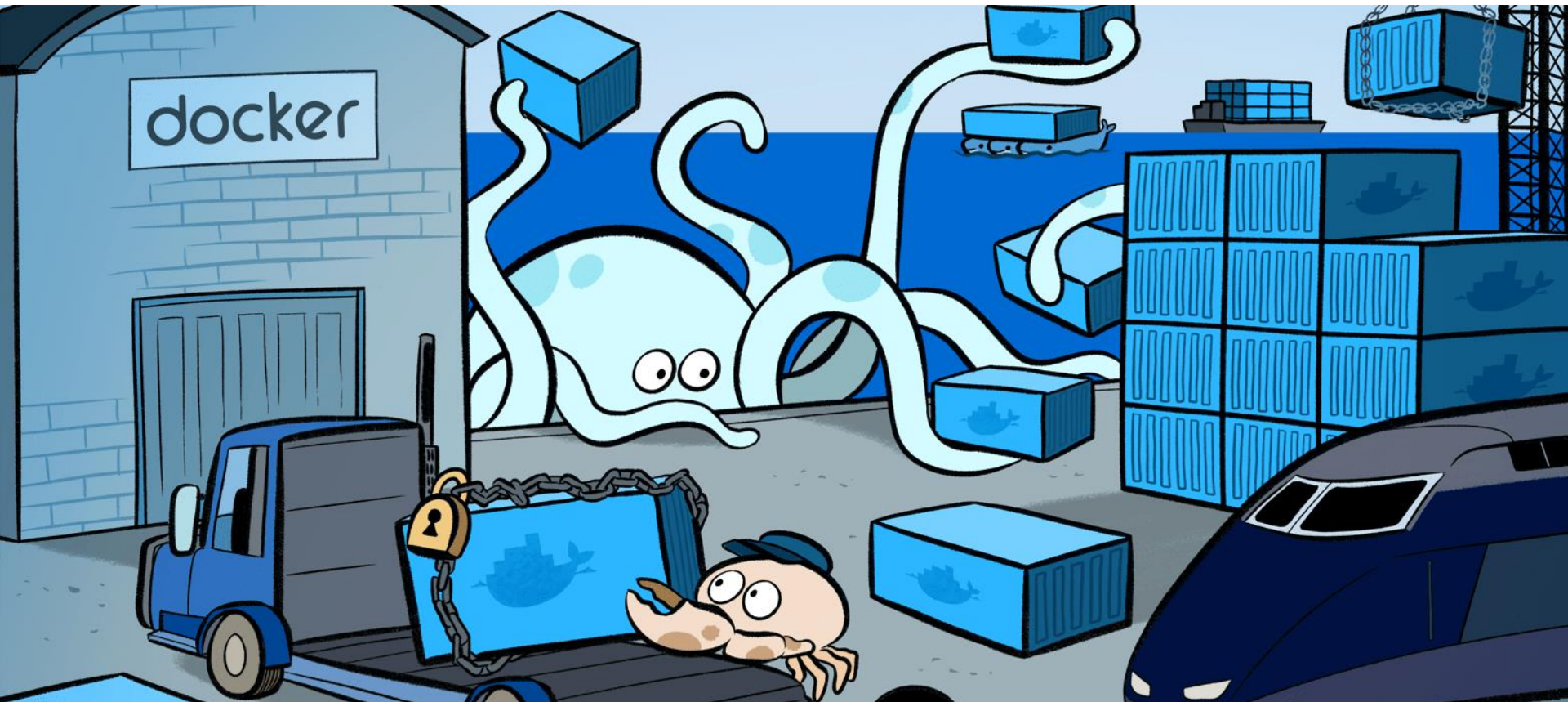
- Una forma de empaquetar software en un formato que incluye todo lo necesario para hacerlo funcionar y se ejecuta aislado del resto de la máquina
- Tiene dos conceptos muy relacionados:
  - La imagen, que es un paquete ejecutable que incluye todo lo necesario para ejecutar un software
  - El contenedor, que es la instancia en ejecución de una imagen, es decir, lo que la imagen

## ¿Y esto no es lo mismo que una máquina virtual?



## ¿Qué son los gestores de contenedores?

- Podman y Docker son software para la gestión de contenedores



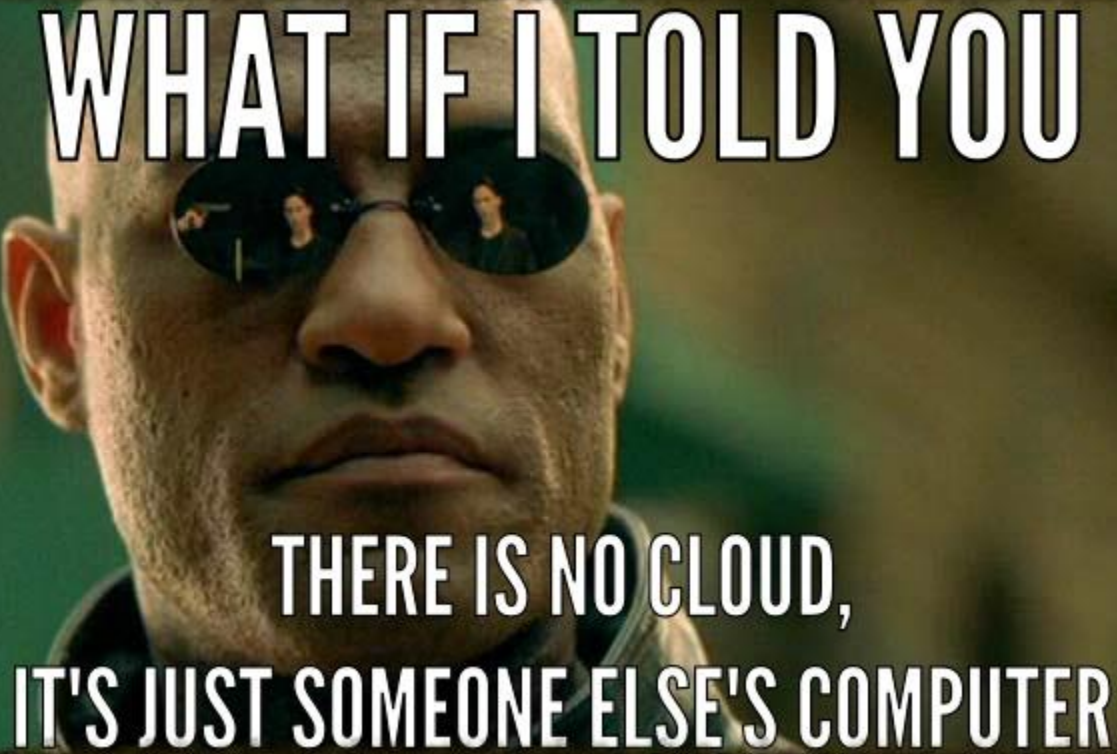


**CONTAINERS  
ARE  
LINUX.®**



redhat.

¿ Y porque usamos contenedores si ya tenemos máquinas virtuales?

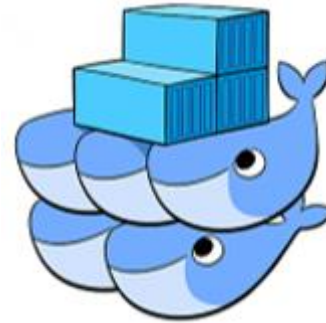


WHAT IF I TOLD YOU  
THERE IS NO CLOUD,  
IT'S JUST SOMEONE ELSE'S COMPUTER



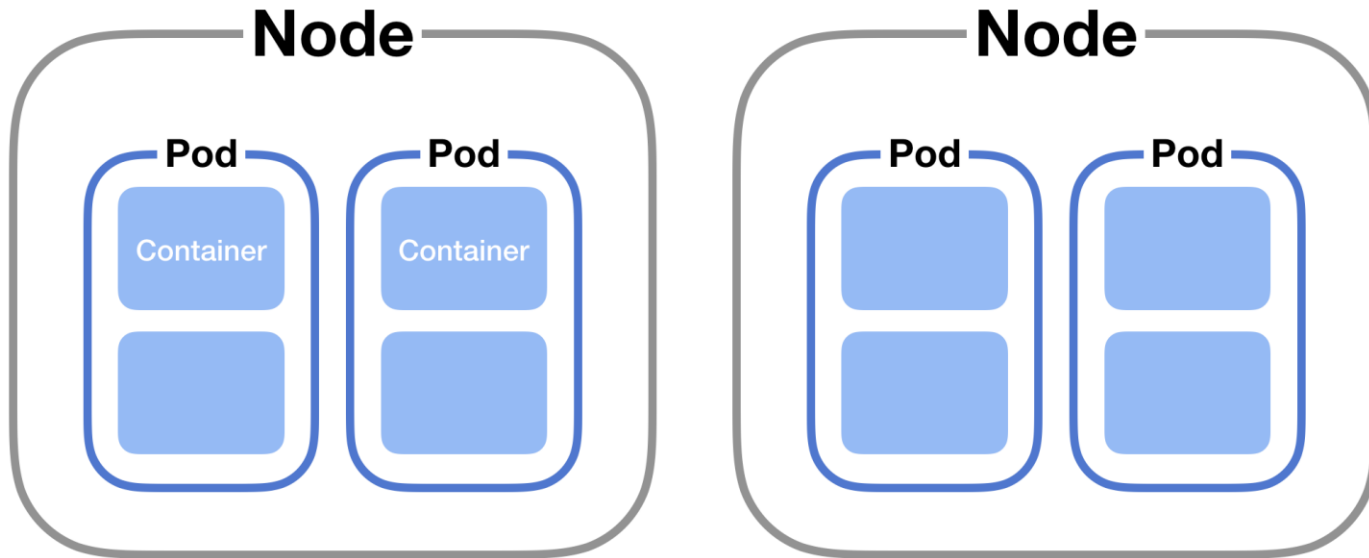


## ¡Contenedores en la nube!



# Containers to the cloud

## Cluster



# Automatically replicating pods

Kubernetes

Node

Node

Node



## Nomenclatura común

- Imagen
  - Contenidos de la aplicación (binarios, fuentes) y de sus dependencias
- Contendor
  - Imagen en ejecución (normalmente uno o más procesos)
- Volumen
  - Datos persistentes de una aplicación
- Pod
  - Conjunto de contenedores para desplegar en la nube
- Kubernetes y Docker swarm
  - Gestores de pods en la nube
- Docker compose
  - Forma original de crear pods (en desuso para despliegue)

# PRIMEROS PASOS



Instala Docker o podman  
(si no funciona <http://play-with-docker.com> )

**IMPORTANTE.** Todos los comandos con docker se pueden ejecutar con la misma sintaxis en podman.

## Nuestro “hello world” con Podman

```
> podman run hello-world
```

Tambien  
funciona  
con docker

Lanzar una imagen

Nombre de la imagen

*container*      ¿Qué pasó?  
*! Dependiendo del gestor!*

Client

- docker build
- docker pull
- docker run

DOCKER\_HOST

Docker daemon

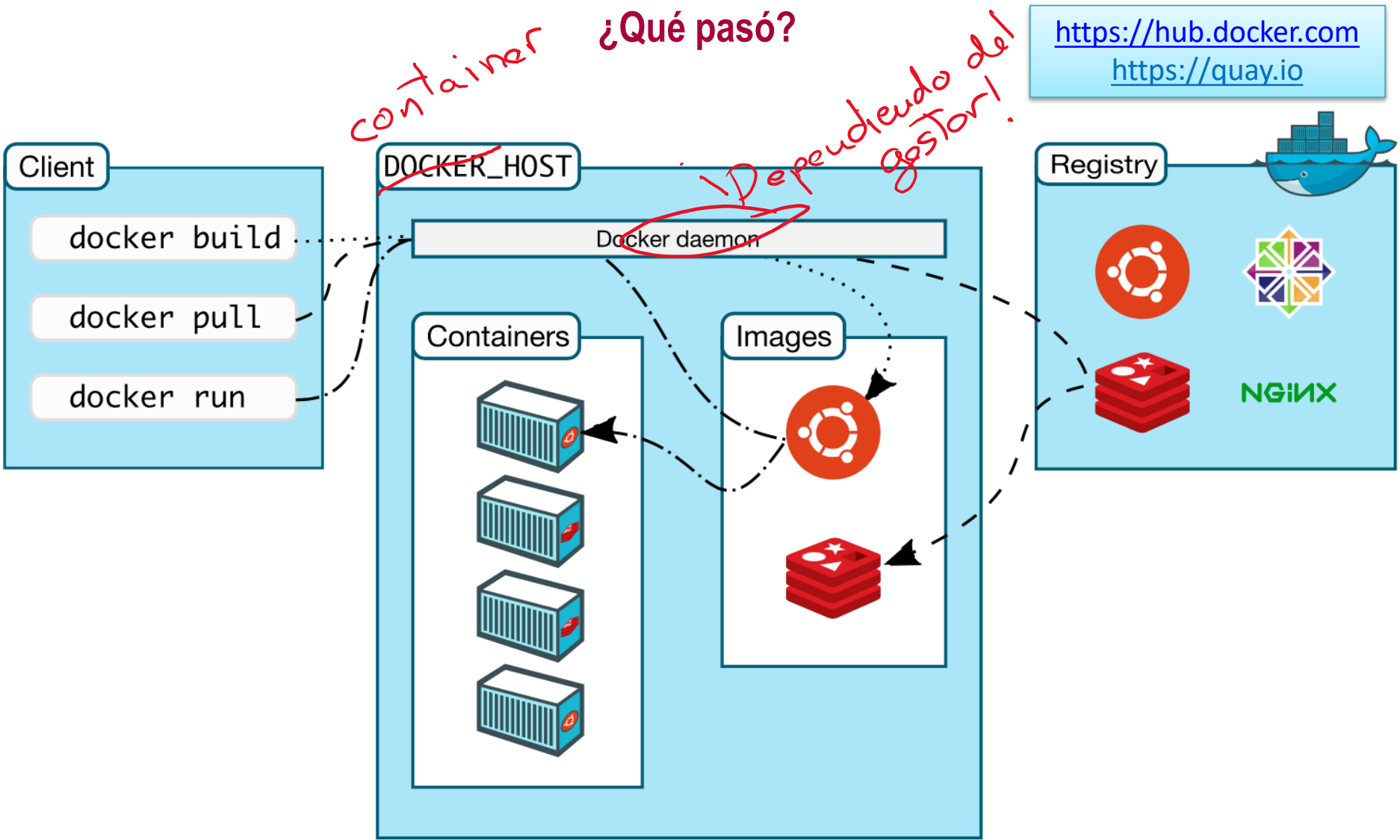
Containers

Images

<https://hub.docker.com>  
<https://quay.io>

Registry

NGINX



## Otro ejemplo

```
> podman run -i -t ubuntu /bin/bash
```

Interactivo



Comando a ejecutar



## Otro más

Puerto local : Puerto contenedor

```
> podman run -p 8000:80 -d  
kitematic/hello-world-nginx
```

En segundo plano



## Otro más

```
> podman run -p 8010:80 -d -v  
/home/<user>/misitioweb:/website_files  
kitematic/hello-world-nginx
```

Edita el index.html que ha aparecido en nginx\_files y prueba cómo se actualiza dinámicamente

¿Puedo tener más de una máquina?

> podman ps

## Esta es una lista de comandos básicos:

```
podman run -d -p 4000:80 friendlyname#Run "friendlyname" mapping port 4000 to 80
podman container ls # List all running containers
podman container ls -a # List all containers, even those not running
podman container stop <hash> # Gracefully stop the specified container
podman container kill <hash> # Force shutdown of the specified container
podman container rm <hash> # Remove specified container from this machine
podman container rm $(podman container ls -a -q) # Remove all containers
podman image ls -a # List all images on this machine
podman image rm <image id> # Remove specified image from this machine
podman image rm $(podman image ls -a -q) # Remove all images from this machine
podman logs <containerName> # Shows the log of a container
```

# ¿Y de dónde salen las imágenes?

mysql

FILTER E

## Recommended



**official**  
**mysql**

MySQL is a widely used, open-source relational database management system (RDBMS).

♡ 4.1K ↓ 69M    **CREATE**

<https://hub.docker.com>  
<https://quay.io>

## Other Repositories



**mysql**  
**mysql-server**

Optimized MySQL Server Docker images. Created, maintained and supported by the MySQL team a...


♡ 284 ↓ 4M    **CREATE**



**tozd**  
**mysql**

MySQL (MariaDB fork) Docker image.


♡ 1 ↓ 2K    **CREATE**



**alterway**  
**mysql**

Docker Mysql

♡ 3 ↓ 1K    **CREATE**



**centurylink**  
**mysql**

Image containing mysql. Optimized to be linked to another image/container.


♡ 49 ↓ 6M    **CREATE**



**cloudposse**  
**mysql**

Improved `mysql` service with support for `mysqld\_safe` and `fixtures` loaded from...

♡ 0 ↓ 540K    **CREATE**



**bitnami**  
**mysql**

Bitnami MySQL Docker Image

♡ 4 ↓ 1K    **CREATE**

# DOCKERIZANDO APLICACIONES



## ¿Qué queremos conseguir?

- Tener empaquetada nuestra aplicación y sus dependencias en una imagen para poder desplegarla donde queramos simplemente con

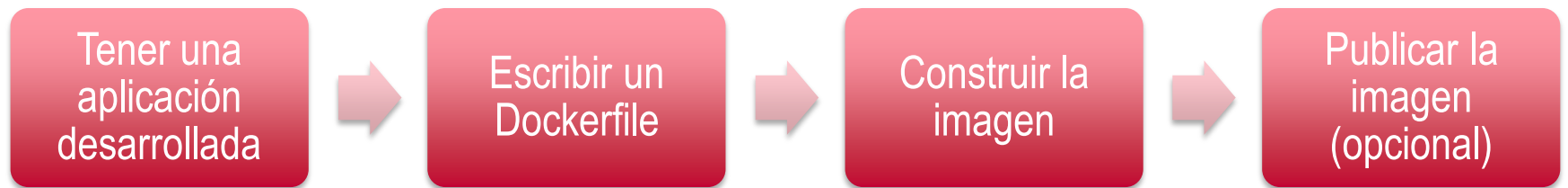
```
> podman run miAplicacion
```

- <https://github.com/EGCETSII/1920-Practica-1>

## Imágenes de docker

- Una imagen es una colección de archivos
- Se parte de una imagen base y luego se construyen imágenes personalizadas encima
- Un Dockerfile o un Containerfile es un fichero que describe las instrucciones para construir una nueva imagen
- Las imágenes están en capas y cada capa representa un diff de la capa anterior

## Pasos para Dockerizar una aplicación



# Nuestra aplicación: Un “Hello world” hecho en python con el framework Flask

```
# Importamos el modulo de flask para poder usar ese framkework
from flask import Flask

# Constructor de Flask
app = Flask(__name__)

# En flask tenemos distintas rutas para distintas funciones
@app.route('/')

# '/' está asociada a la función hello_world().
def hello_world():
    return 'Hello World'

# '/hello/name está asociada a la función hello_name().
@app.route('/hello/<name>')
def hello_name(name):
    return 'Hello %s!' % name

# Función principal
if __name__ == '__main__':
    app.run()
```

## El Dockerfile

```
# Base image
FROM python:3

COPY requirements.txt ./ # añadir Flask al fichero tras
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD [ "python", "./holamundo.py" ]
```

Consejos para escribir Dockerfiles: [https://docs.docker.com/engine/userguide/eng-image/dockerfile\\_best-practices/](https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices/)



## Construimos la imagen y la comprobamos

```
> podman build -t flaskhello .
```

```
> podman images
```

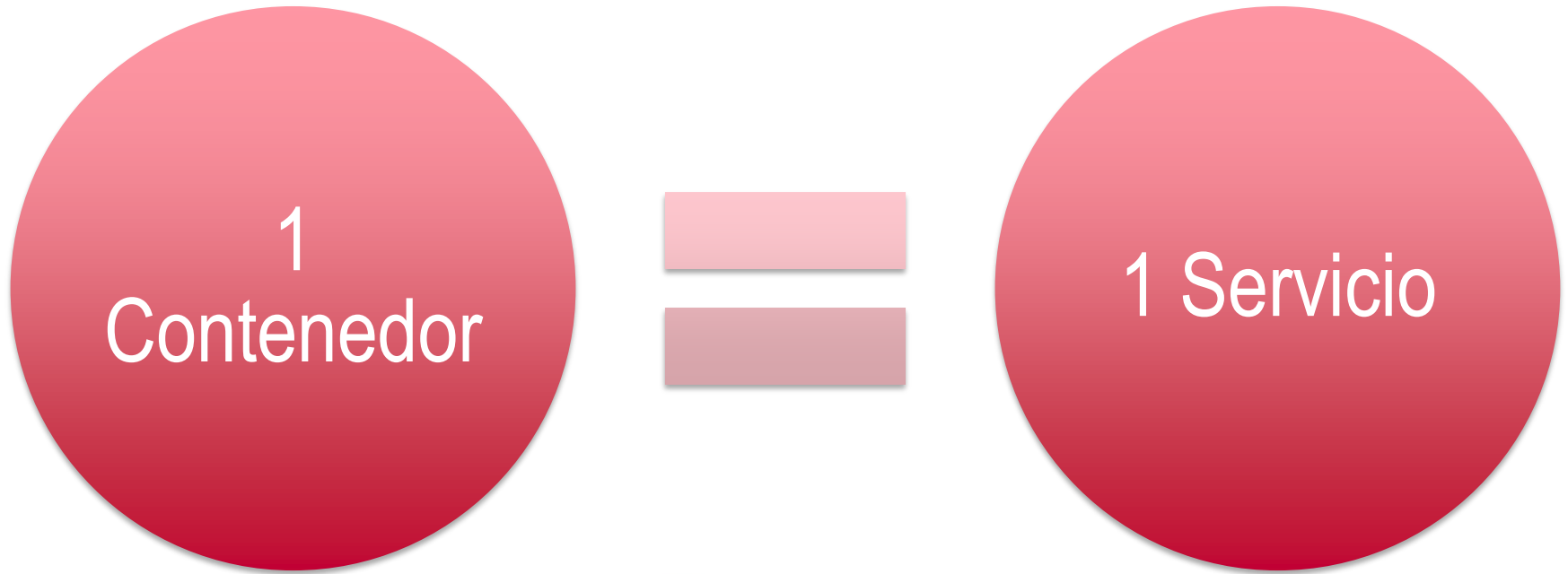
```
> podman run -it --rm -p 5000:5000 flaskhello
```

## Generemos el fichero Kubernetes desde nuestro dockerfile (opcional)

- #First we run our app within a pod giving it a name
  - `podman run --name mypod myapp`
- `podman generate kube mypod`
- `podman pod ps`

**EJECUTANDO DECIDE EN CONTENEDORES CON  
DOCKER COMPOSE**

En Docker se recomienda seguir el principio de responsabilidad única:



# ¿QUÉ PENSAIS QUE ESTÁ MAL EN ESTA DEFINICION?

version: '3.4'

services:

db:

restart: always

container\_name: decide\_db

image: postgres:alpine

volumes:

- db:/var/lib/postgresql/data

networks:

- decide

web:

restart: always

container\_name: decide\_web

image: decide\_web:latest

build: .

command: ash -c "python manage.py migrate && gunicorn -w 5 decide.wsgi --timeout=500 -b 0.0.0.0:5000"

expose:

- "5000"

*compose*

## Arreglar las dependencias

*Dockerfile*

---

db:  
restart: always  
container\_name: decide\_db  
image: postgres:10.15-alpine  
volumes:  
- db:/var/lib/postgresql/data  
networks:  
- decide  
environment:  
- POSTGRES\_PASSWORD=postgres

```
from python:3.7-alpine

RUN apk add --no-cache git postgresql-dev gcc libc-dev
RUN apk add --no-cache gcc g++ make libffi-dev python3-dev build-base

RUN pip install gunicorn
RUN pip install pycopg2
RUN pip install ipdb
RUN pip install ipython

WORKDIR /app

RUN git clone https://github.com/jagalindo/decide.git .
RUN pip install -r requirements.txt
```

*requirements*

```
Django==2.0
pycryptodome==3.6.6
djangorestframework==3.7.7
django-cors-headers==2.1.0
requests==2.18.4
django-filter==1.1.0
psycopg2-binary==2.7.4
django-rest-swagger==2.2.0
coverage==4.5.2
django-nose==1.4.6
jsonnet==0.12.1
```

---

**¿Por qué hay que especificar el binario de pycspg2?**



# CONCLUSIONES

Lectura recomendada: <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>



## ¿Para qué me sirve Docker como desarrollador?

- Entornos de desarrollo:
  - Compartibles
  - Seguros
  - Limpios
  - Extensibles
- Asegura el mismo entorno en:
  - Todos los desarrolladores
  - Pruebas
  - Producción
- Facilita gestionar varias versiones de una misma aplicación
- Ahorra costes en el despliegue

## ¿Para qué me sirve como administrador?

- Despliegue independiente de la tecnología (Java, PHP, NodeJS...)
- Elimina inconsistencias entre entornos de desarrollo, prueba y producción
- Permite desplegar de forma similar en:
  - El portátil del desarrollador
  - En máquinas virtuales en un data center
  - En servidores cloud (AWS, Azure, DigitalOcean...)
  - En una mezcla de ellos
- Ofrece facilidades de escalado y gestión de clústeres
- Es más barato que las máquinas virtuales

# Docker en la Universidad

- **Regístrate** en <http://dockr.ly/students> y obtendrás:
  - Acceso al *Docker Student Kit*.
  - Últimas **novedades** y actualizaciones sobre Docker.
  - **Invitaciones** y códigos de **descuento** a **eventos** de Docker para estudiantes.
  - Posibilidad de conseguir **acceso prioritario** a betas y lanzamientos de productos.
  - Oportunidad de convertirte en *Docker Ambassador*.
  - Acceso al **canal de Slack** de Docker (#docker-students).

# Recursos

- **Cursos:**

- Laboratorios virtuales gratuitos: <http://training.play-with-docker.com/>
- Cursos gratuitos oficiales: <http://training.docker.com/category/self-paced-online>

- **Libros:**

- Docker Cookbook: <http://shop.oreilly.com/product/0636920036791.do>
- Using Docker: <http://shop.oreilly.com/product/0636920035671.do>
- Docker: Up & Running: <http://shop.oreilly.com/product/0636920036142.do>

## Agradecimientos

- Parte de estas transparencias están muy inspiradas (incluso copiadas) de una presentación de Docker de Antonio Gámez (<http://personal.us.es/agamez2/conferencias/docker-y-kubernetes-el-futuro-de-la-distribucion-de-aplicaciones-en-la-nube/> )