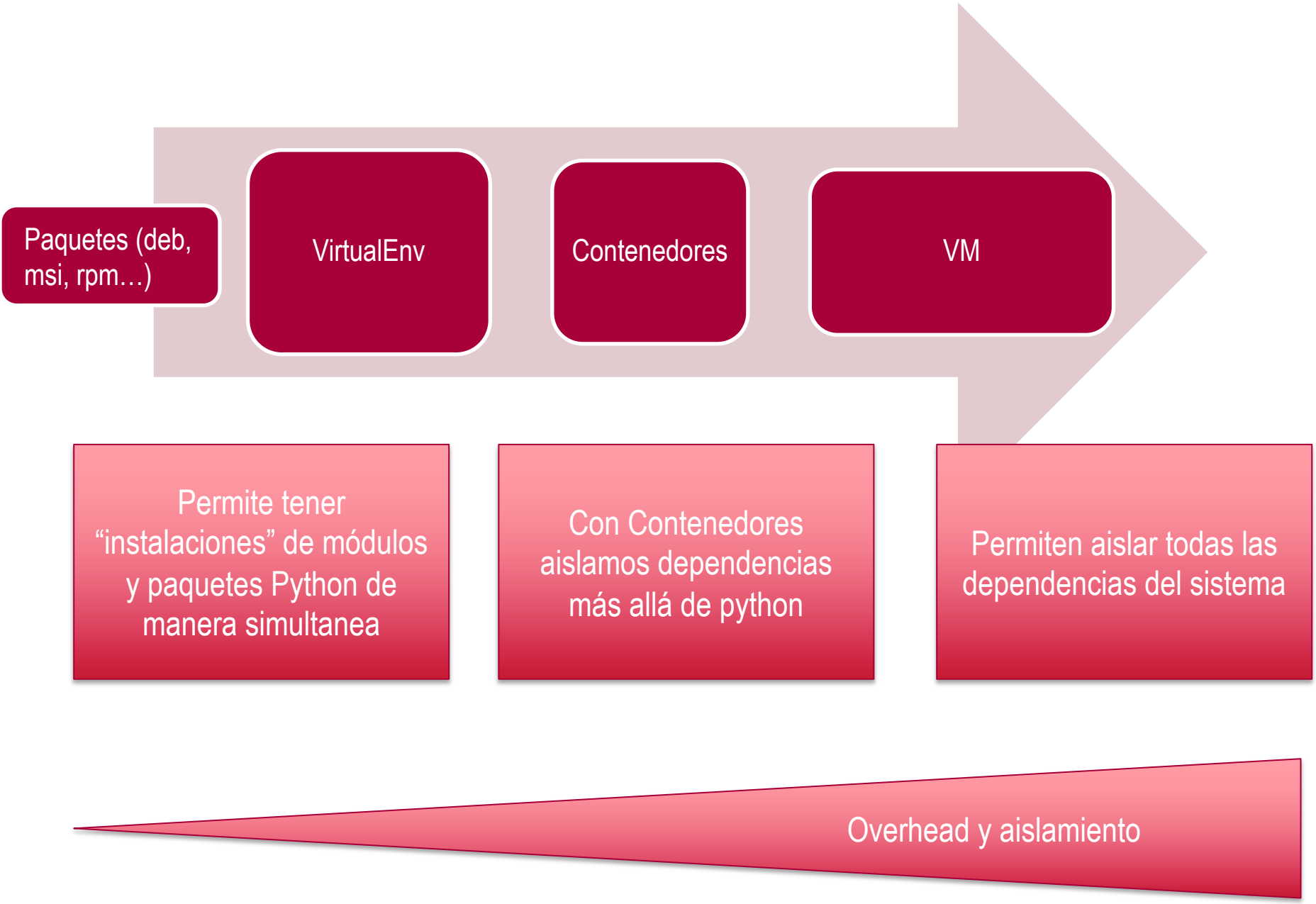
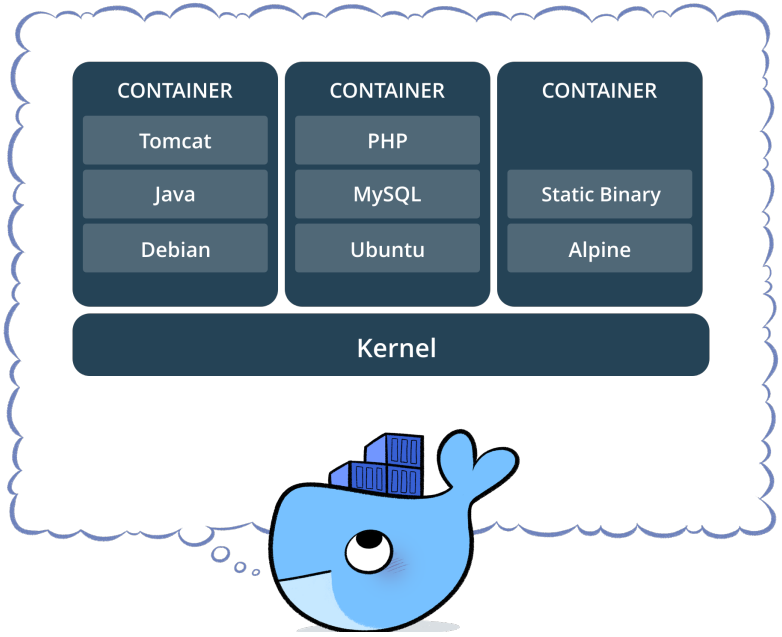
Despliegue de Aplicaciones en Contenedores y Máquinas Virtuales

Evolución y Gestión de la Configuración

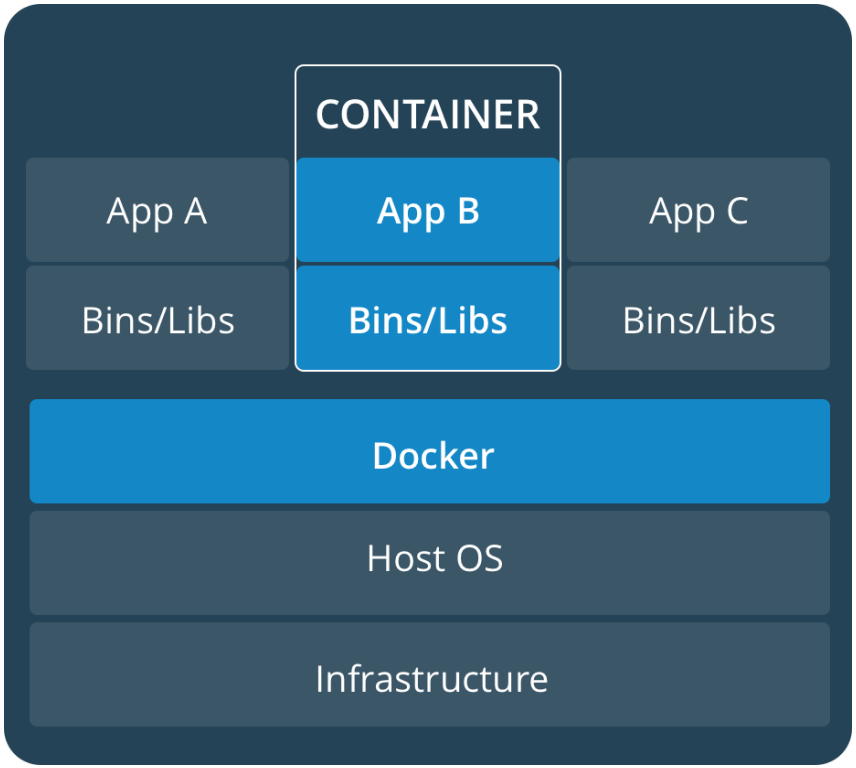

Paquetes (deb, msi, rpm…)
VirtualEnv
Contenedores
VM
Permite tener "instalaciones" de módulos y paquetes Python de manera simultanea
Con Contenedores aislamos dependencias más allá de python
Permiten aislar todas las dependencias del sistema
Overhead y aislamiento

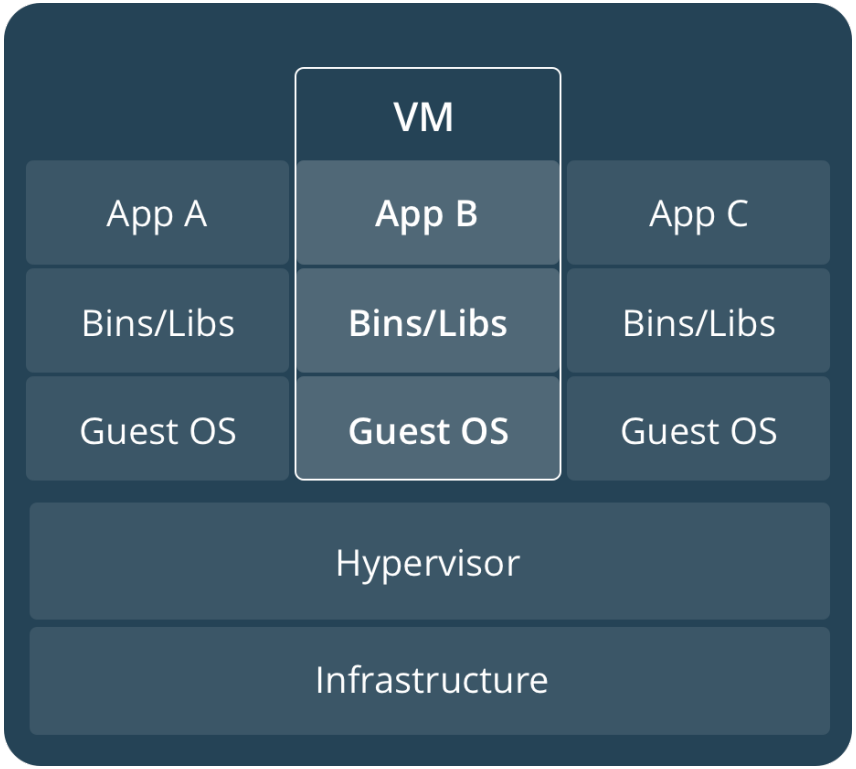


¿Qué es un contenedor?

- 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

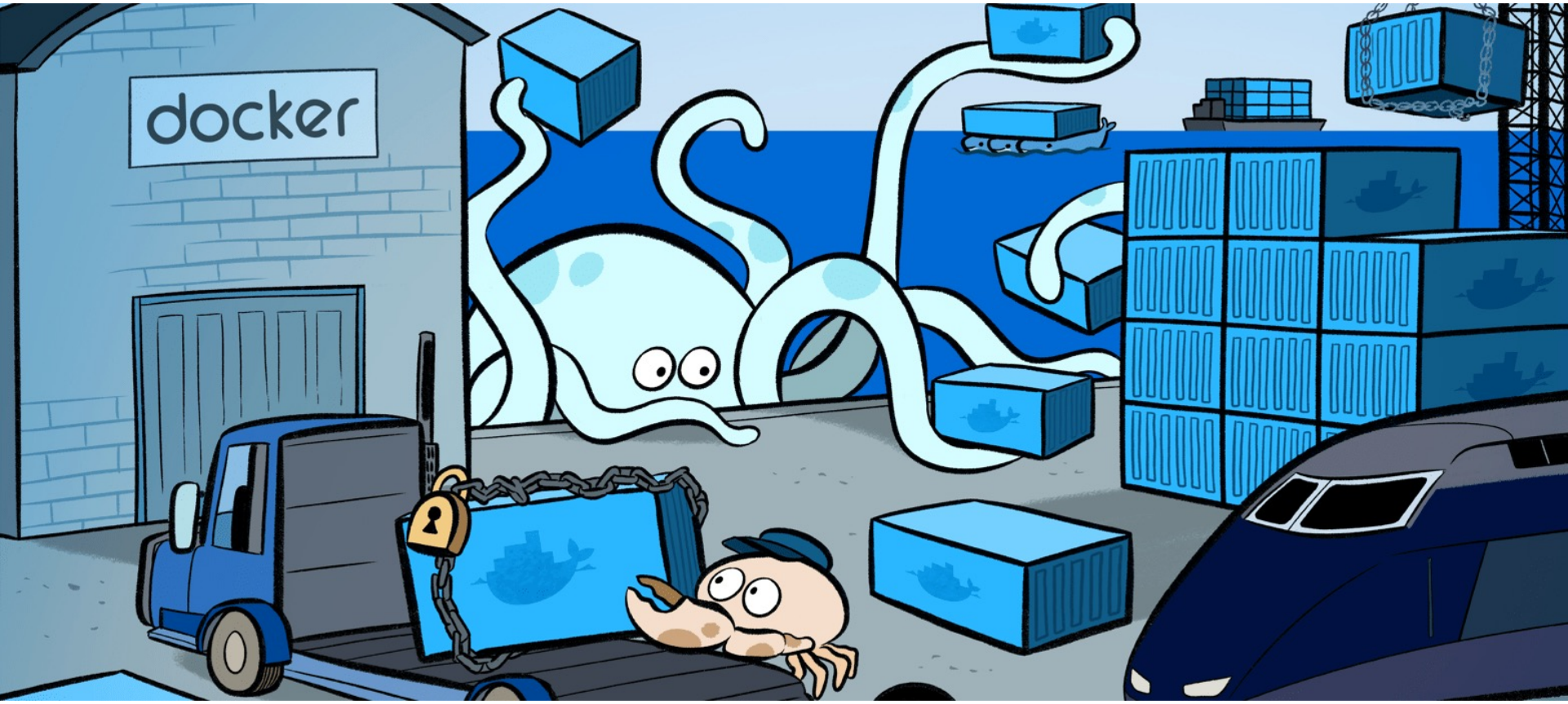
¿Y esto no es lo mismo que una máquina virtual?





¿Qué son los gestores de contenedores?

- Podman y Docker son software para la gestión de contenedores

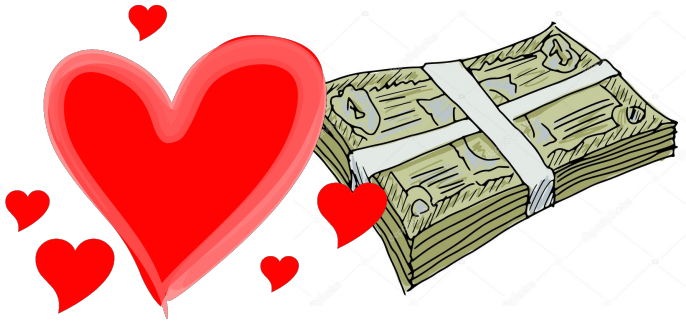



CONTAINERS
ARE
LINUX®
redhat.

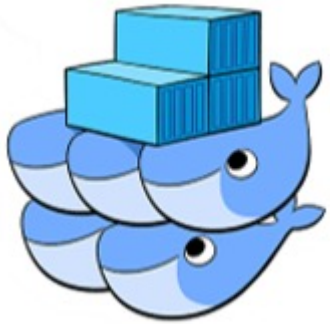
¿ Y porque usamos contenedores si ya tenemos máquinas virtuales?

Azure

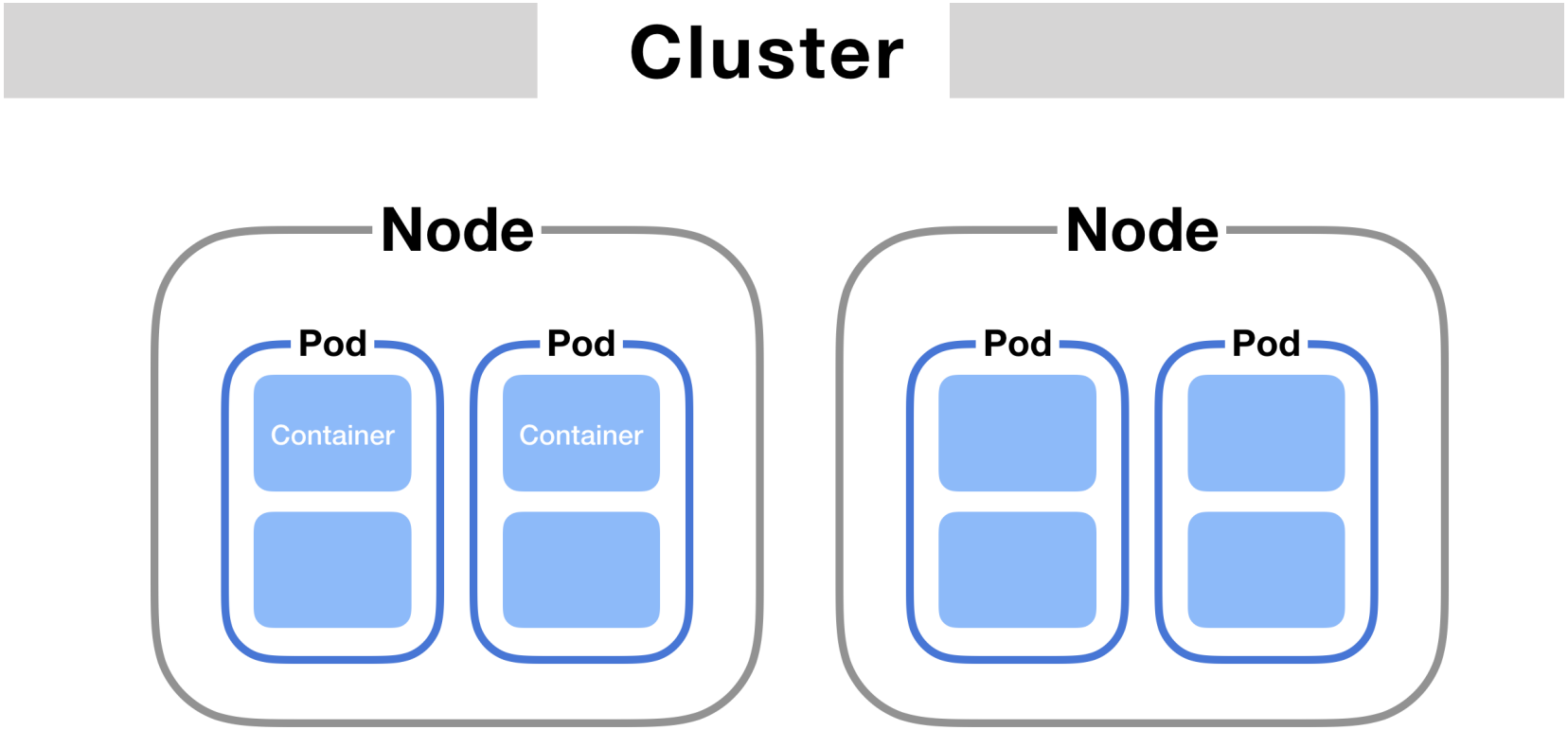
Amazon
EC2



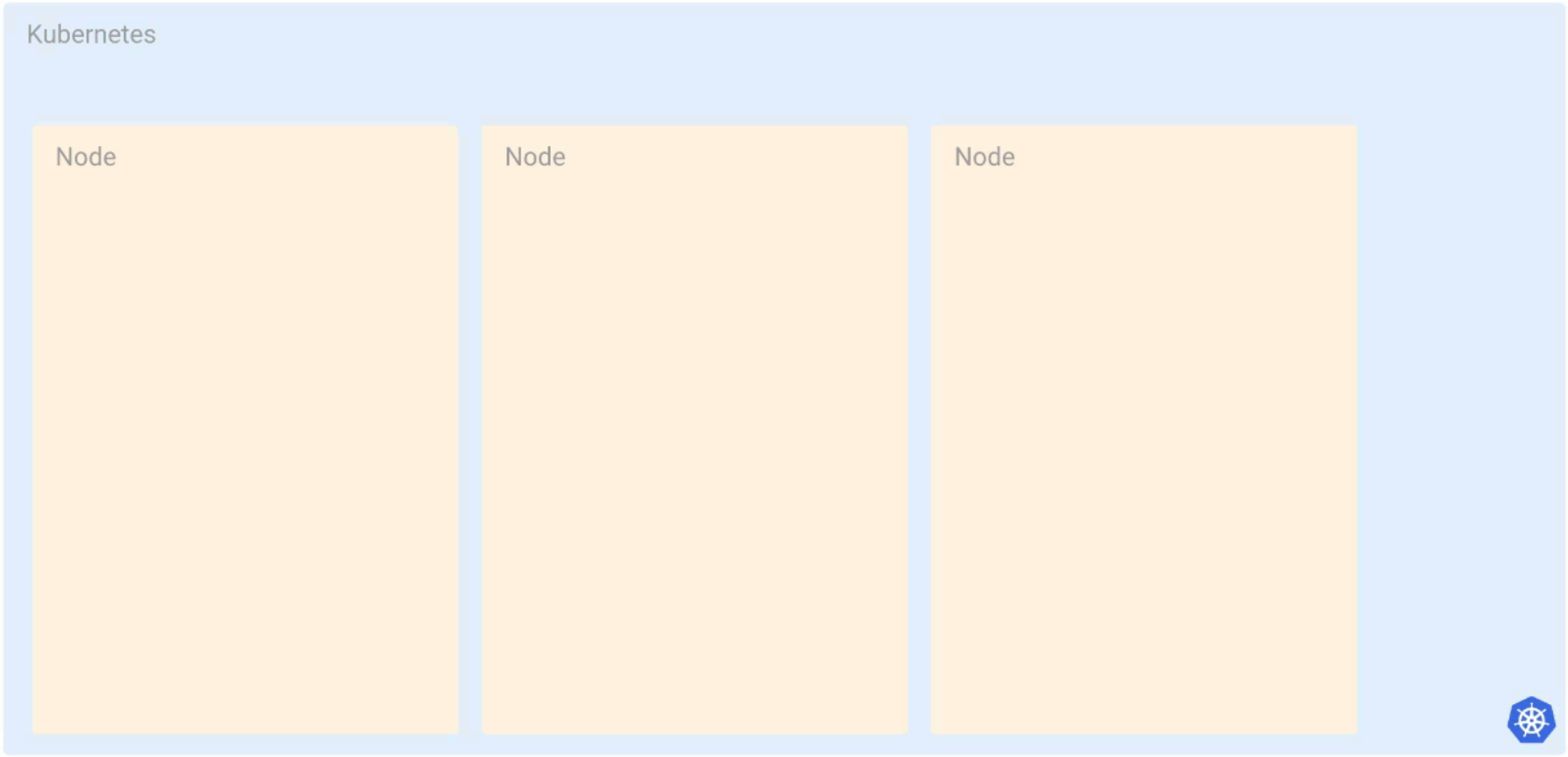
¡Contenedores en la nube!



Containers to the cloud



Automatically replicating pods



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 swarn
  - Gestores de pods en la nube
- Docker compose
  - Forma original de crear pods (en desuso para despliegue)

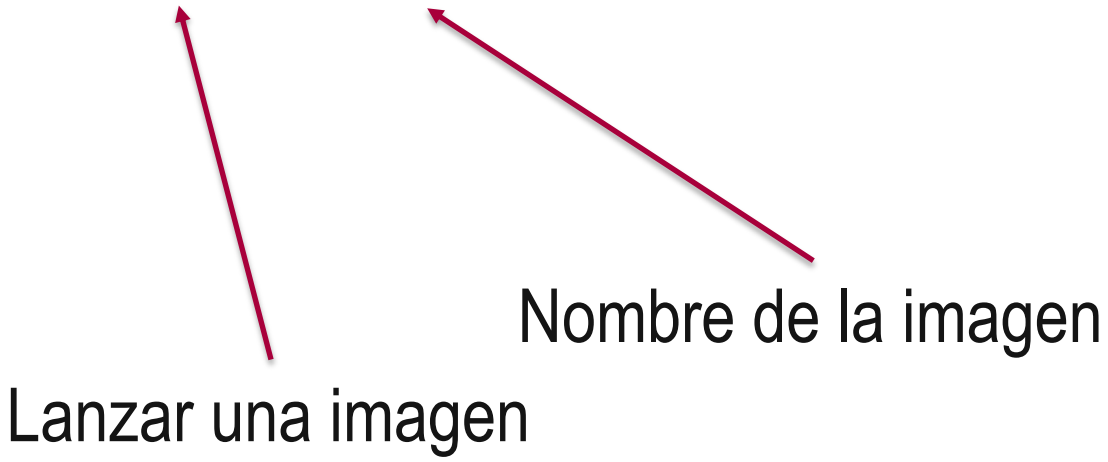
PRIMEROS PASOS

Instala Docker

(si no funciona http://play-with-docker.com)

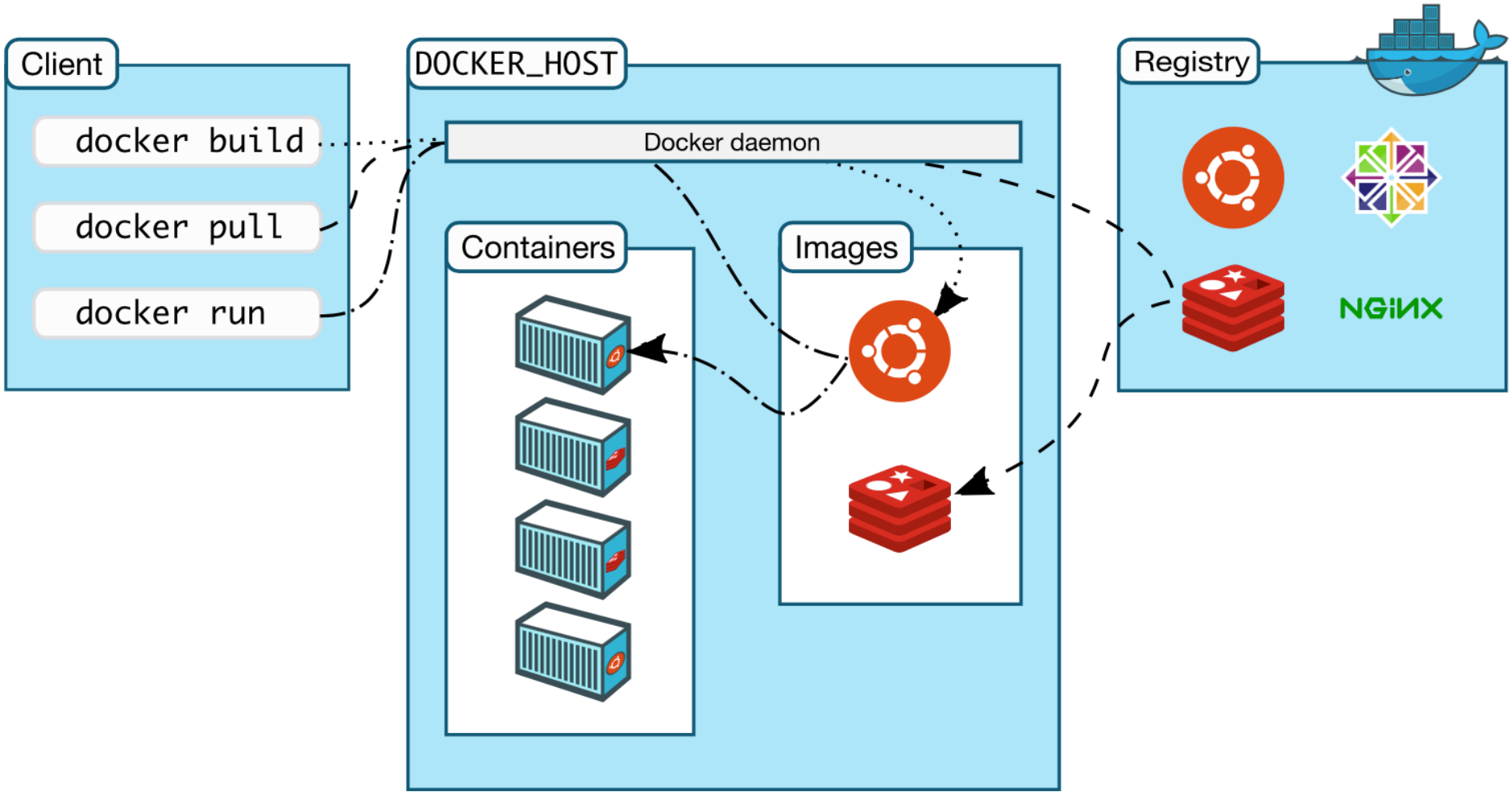
Nuestro “hello world” con Docker

```
> docker run hello-world
```

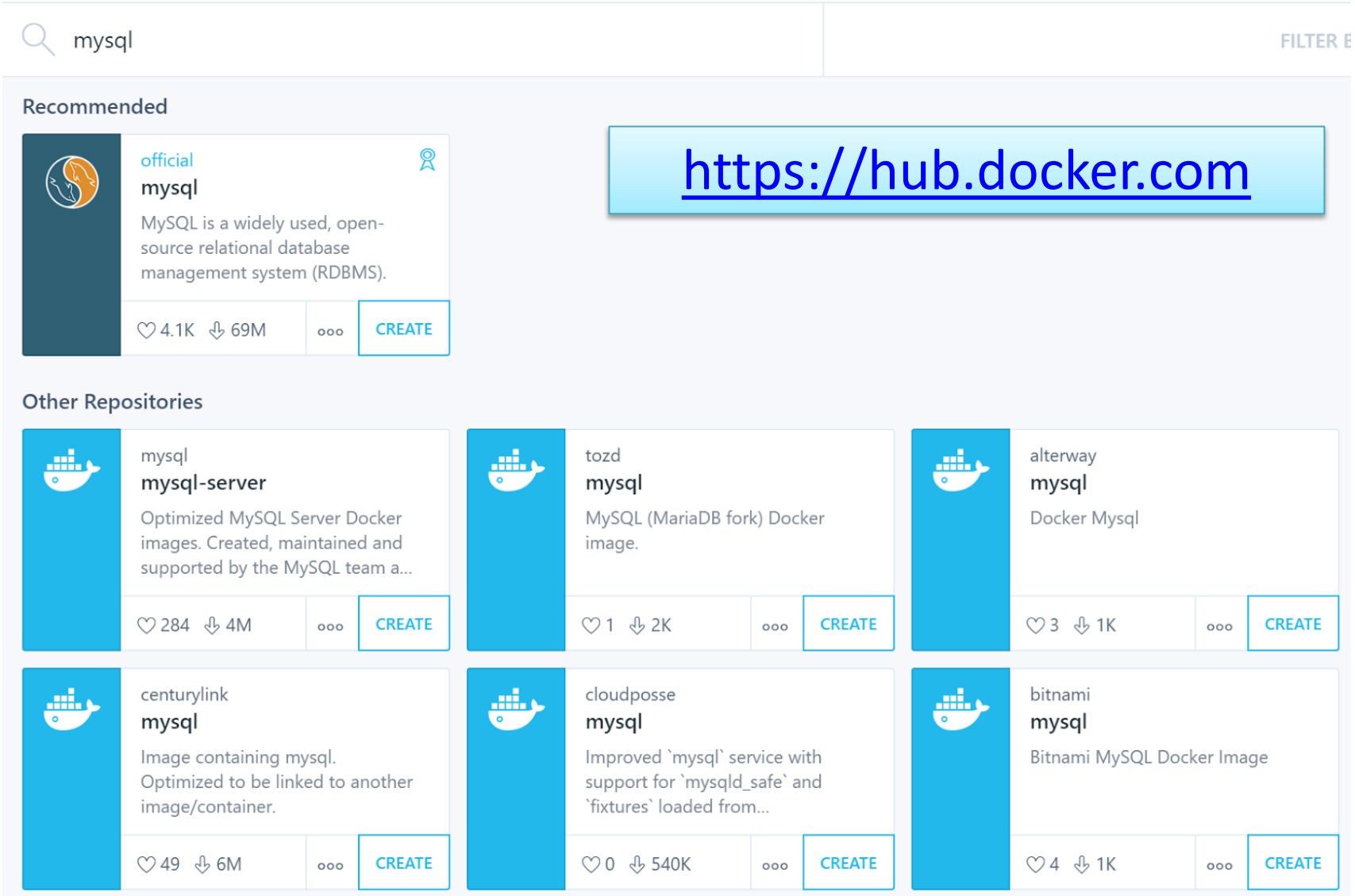


¿Qué ha pasado?

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



¿Y de dónde salen las imágenes?



Otro ejemplo

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

Otro más

Puerto local : Puerto contenedor

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

En segundo plano

Otro más

```
> docker 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

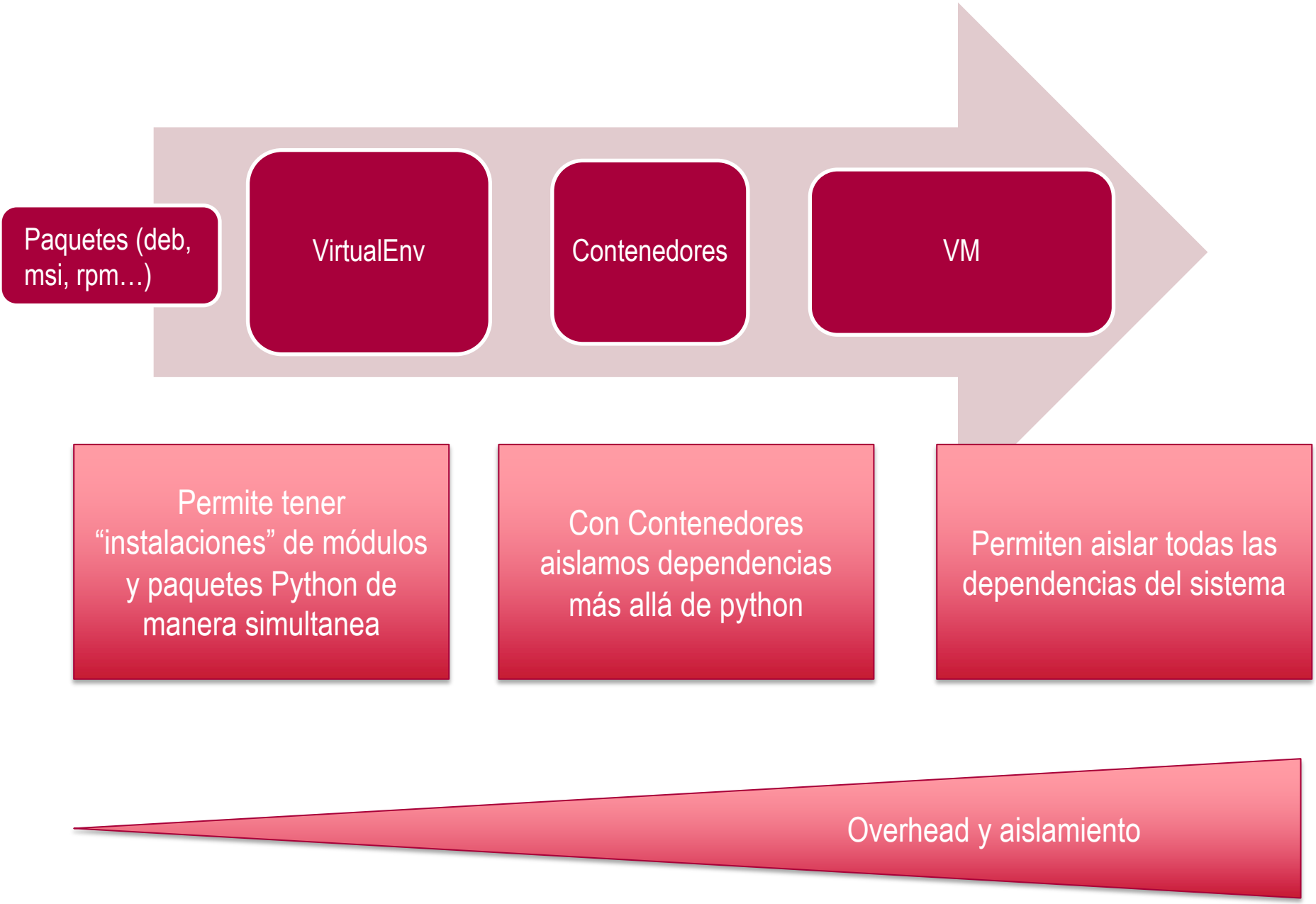
¿Puedo tener más de una máquina?

```
> docker ps
```

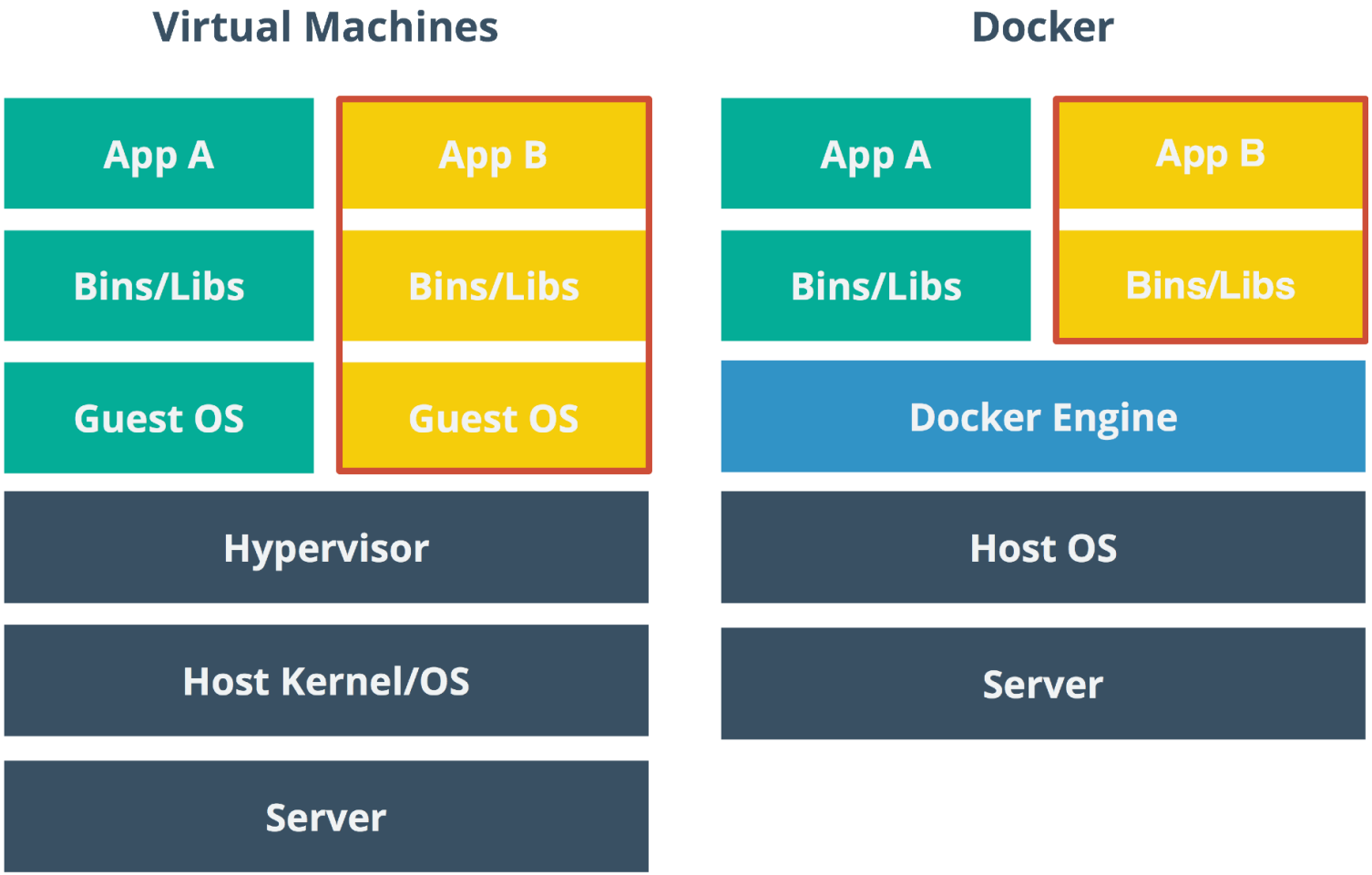
Esta es una lista de comandos básicos:

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

INTRODUCCIÓN A MAQUINAS VIRTUALES


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

VM vs contenedores



https://www.linuxparty.es/109-virtualizacion/9544-docker-virtualiza-aplicaciones-con-contenedores

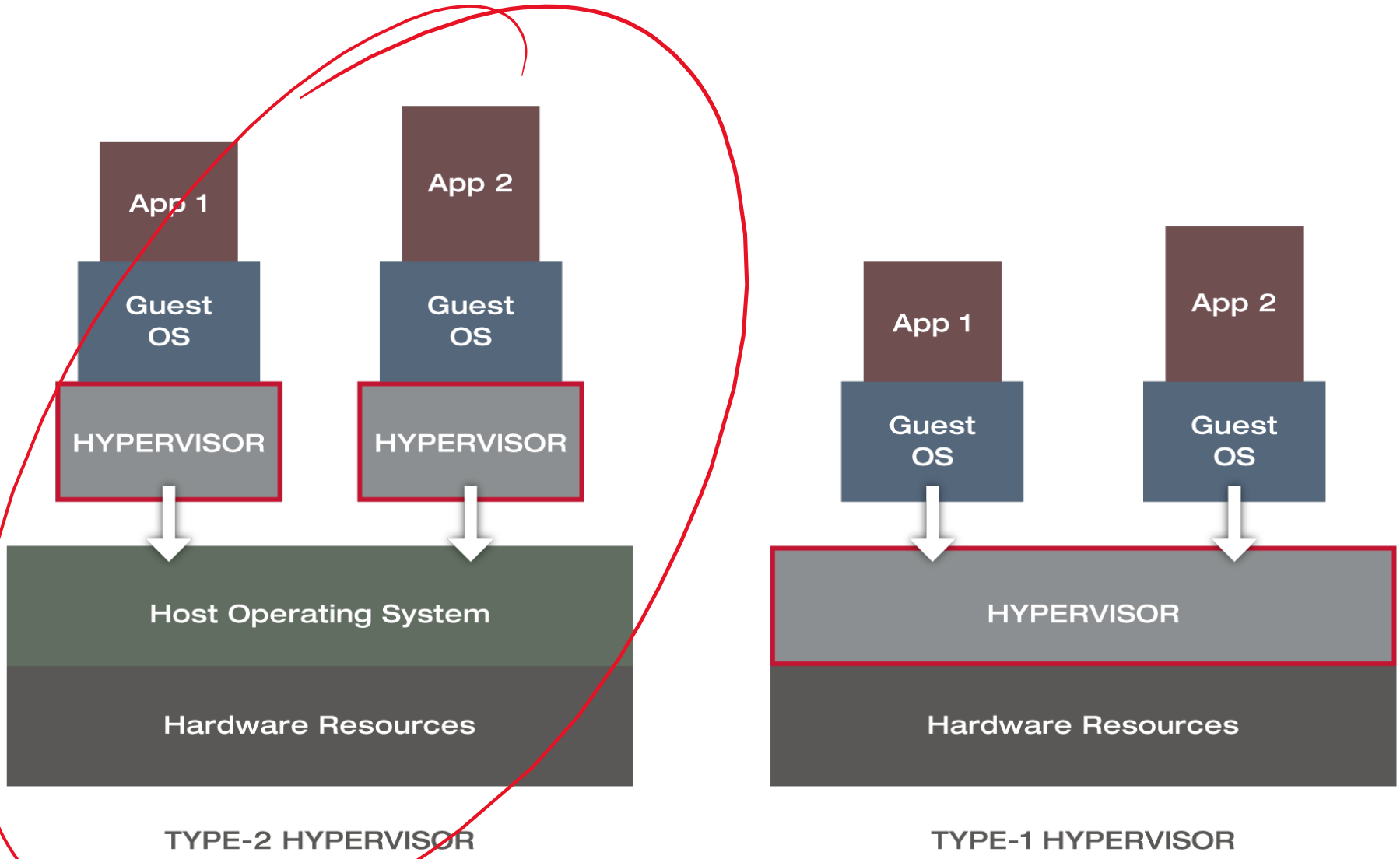
Contenedores

Máquinas virtuales

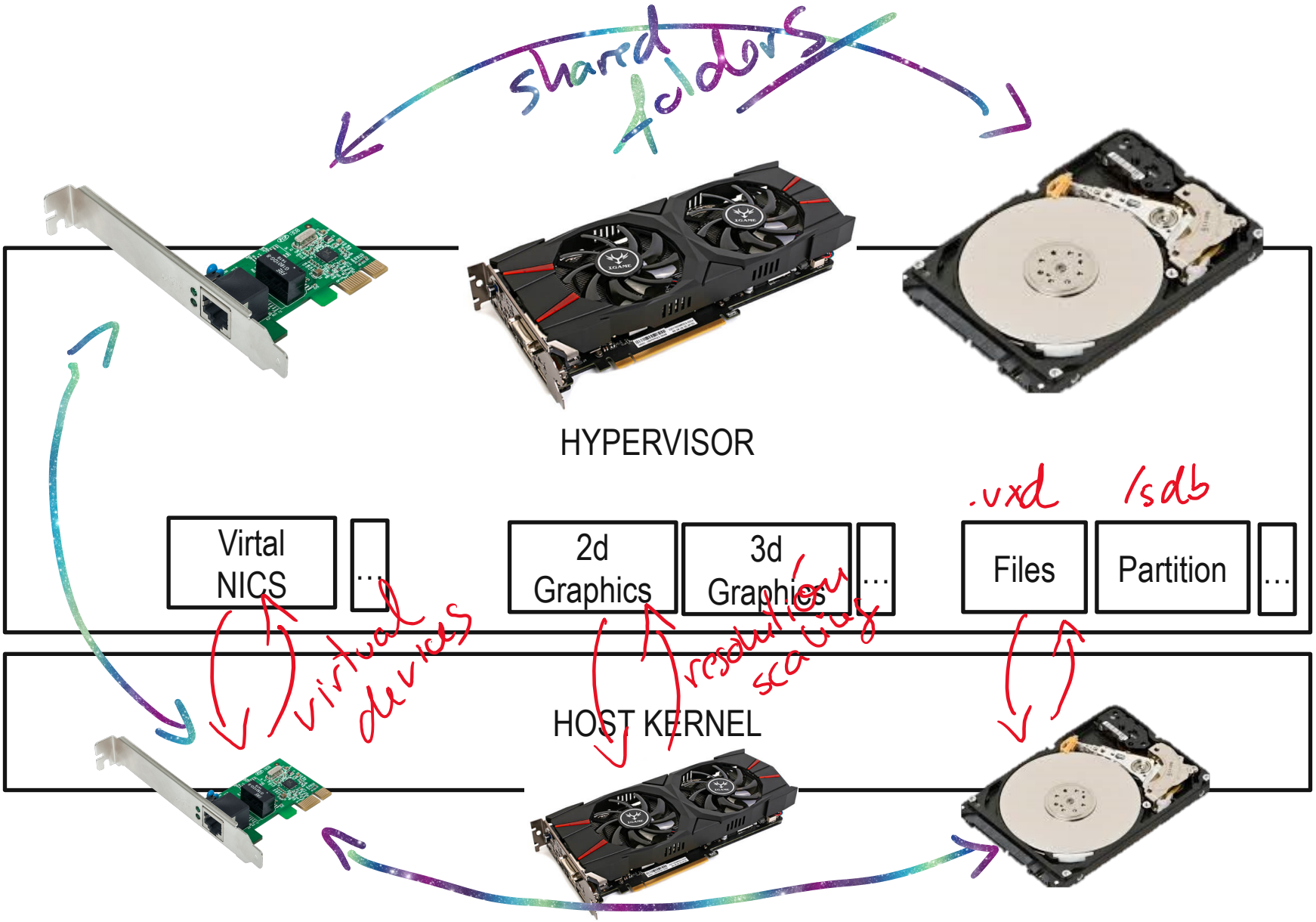
- Aislamiento parcial del sistema operativo host. Menos seguro con respecto a ataques.
- Ejecuta el mismo núcleo que el host. E.g. un host Windows soporta contenedores Windows.
- Actualiza el Dockerfile, genera una nueva imagen, sube de nuevo al host de imágenes

- Aislamiento completo del sistema operativo. Más seguro con respecto a ataques a la infraestructura.
- Ejecuta cualquier sistema operativo como invitado.
- Cuando actualizamos, necesitamos descargar e instale las actualizaciones del sistema operativo en cada VM. Instalar una nueva versión del sistema operativo requiere actualizar o, a menudo, sólo crear una VM completamente nueva. Esto puede llevar mucho tiempo, especialmente si tiene muchas máquinas virtuales.
- Compartir archivos mediante protocolos de red

Tipos de hipervisores



Se necesitan implementar drivers virtuales para todos los dispositivos



Modelos comerciales de hypervisores tipo 2

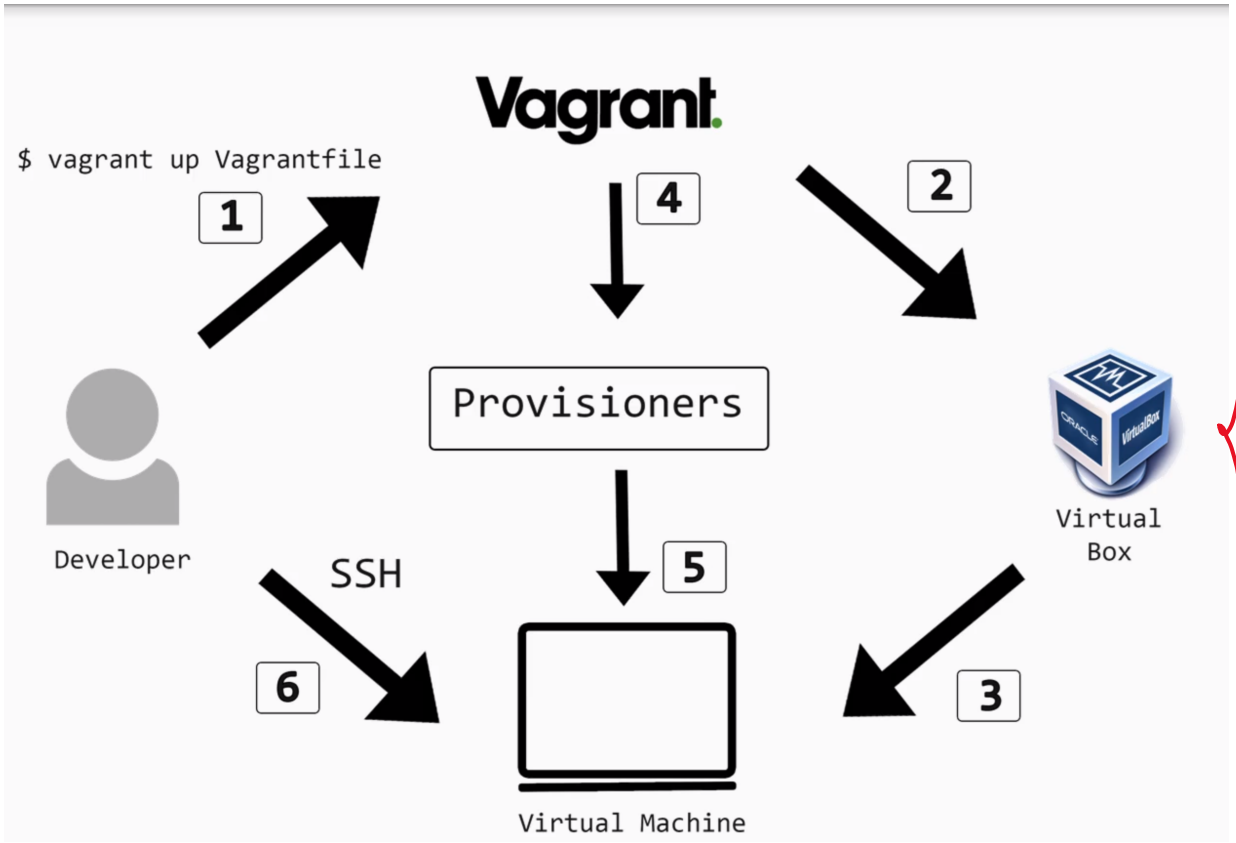
Parallels

Microsoft
Hyper-V

vmware®

GESTORES DE VM (soportando distintos hipervisores)

VAGRANT WORKFLOW



- Backend of Vagrant
- VirtualBox
- VMware
- Hyper-V
- vCloud
- AWS

https://cwzhou.win/2019/08/03/vagrant/

Hay dos etapas principales

- Primera etapa. Creación de la vm en el hipervisor
  - Configuración de red
  - Discos duros
  - Drivers gráficos
- Segunda etapa. Aprovisionamiento del software
  - Script sh de instalación
  - Ansible

PRIMEROS PASOS

Nuestro “hello world” con Vagrant

```
> vagrant init ubuntu/trusty32
```

Otro ejemplo

```
> vagrant up
```

Enciende la máquina virtual

Otro más

```
> Vagrant ssh
```

```
> Vagrant ssh –c “cat /etc/sources.list”
```

Otro más

```
> vagrant init obihann/nginx \
    --box-version 0.0.1
  vagrant up
```

config.vm.network "forwarded_port", guest: 80, host: 8080

Creamos un .html de ejemplo

Esta es una lista de comandos básicos:

- Adding a vagrant box:
  - Syntax: vagrant box add
  - Example: vagrant box add ubuntu/trusty32
- Listing and removing vagrant boxes:
  - vagrant box list
  - vagrant box remove
- Creating a VM environment:
  - Syntax: vagrant init
  - Example: vagrant init ubuntu/trusty32
- Starting a VM environment:
  - vagrant up ubuntu/trusty32
  - vagrant up
- Connecting:
  - vagrant ssh ubuntu/trusty32
  - vagrant ssh
- Stopping, restarting, and destroying
  - vagrant halt
  - vagrant reload
  - vagrant destroy

¿Para qué me sirven estas soluciones 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 sirven 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
- Es más caro que los contenedores

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/)