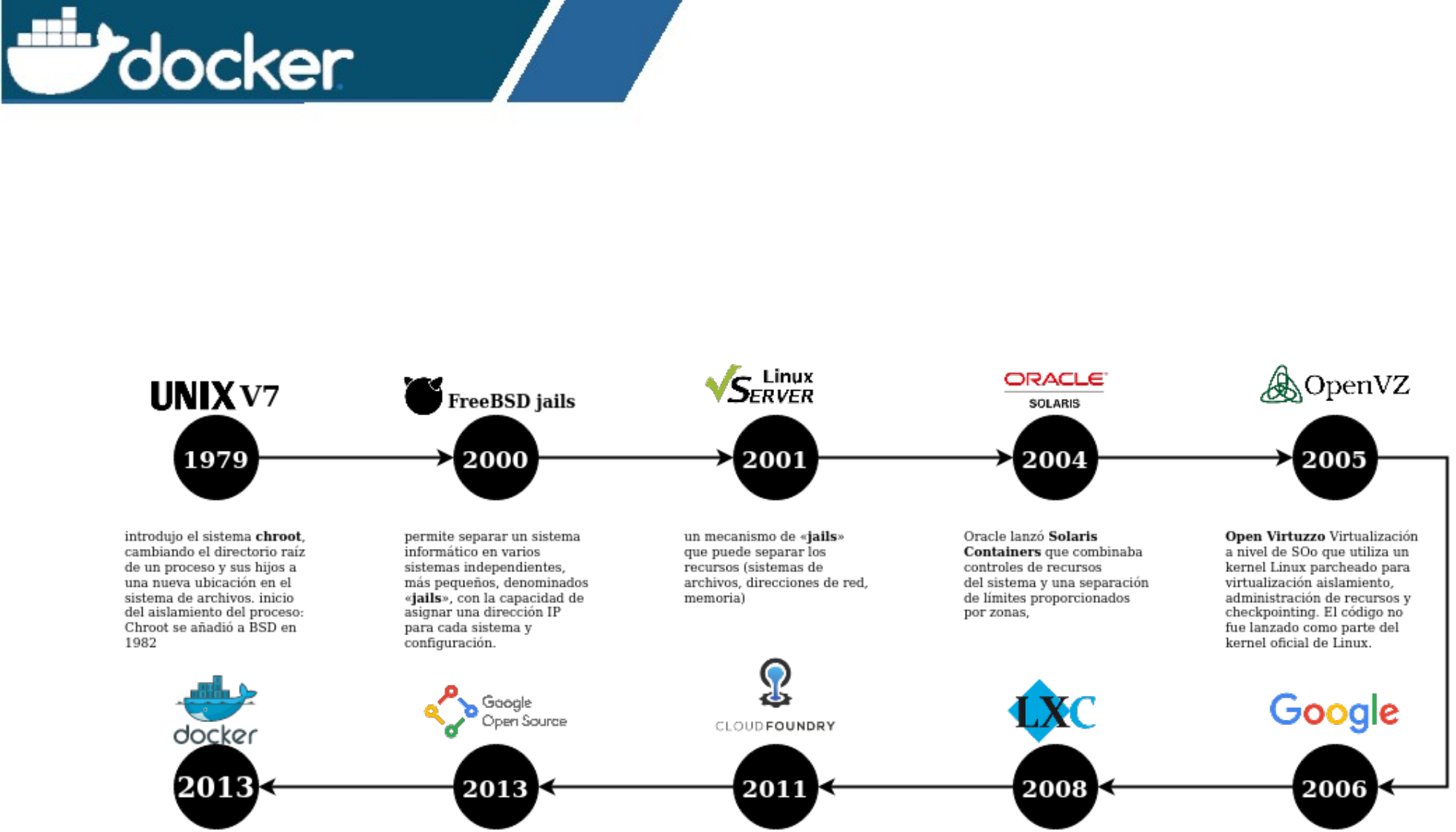
Escuela de Ingeniería de
Sistemas e Informatica

Jathinson Meneses M.
Docente Ingeniería de Sistemas UIS
Henry A. Jimenez H.
Estudiante Maestría en Ingeniería
de Sistemas e Informática UIS

Workshop
Introducción a
docker®

Universidad Industrial de Santander
Super Computación y Cálculo Científico UIS
CAGE

docker


UNIX V7
1979
introdujo el sistema **chroot**, cambiando el directorio raíz de un proceso y sus hijos a una nueva ubicación en el sistema de archivos. inicio del aislamiento del proceso: Chroot se añadió a BSD en 1982
FreeBSD jails
2000
permite separar un sistema informático en varios sistemas independientes, más pequeños, denominados «**jails**», con la capacidad de asignar una dirección IP para cada sistema y configuración.
Linux SERVER
2001
un mecanismo de «**jails**» que puede separar los recursos (sistemas de archivos, direcciones de red, memoria)
ORACLE SOLARIS
2004
Oracle lanzó **Solaris Containers** que combinaba controles de recursos del sistema y una separación de límites proporcionados por zonas,
OpenVZ
2005
Open Virtuzzo Virtualización a nivel de SOo que utiliza un kernel Linux parcheado para virtualización aislamiento, administración de recursos y checkpointing. El código no fue lanzado como parte del kernel oficial de Linux.
Google
2006
Process Containers diseñado para limitar, contabilizar y aislar el uso de recursos (CPU, memoria, I/O de disco, red) de una colección de procesos. Fue renombrado «**Grupos de control (cgroups)**» un año después y finalmente se fusionó con el kernel 2.6.24 de Linux.
LXC
2008
LXC (LinuX Containers) fue la primera y más completa implementación del gestor de contenedores Linux. Se dió en el año 2008 utilizando cgroups y namespaces de Linux, y funcionaba en un solo kernel de Linux sin necesidad de parches.
CLOUDFOUNDRY
2011
Warden aísla entornos en cualquier sistema operativo, ejecutándose como un daemon y proporcionando una API para la gestión de contenedores. Desarrolló un modelo cliente-servidor para administrar una colección de contenedores a través de varios hosts
Google Open Source
2013
Let Me Contain That For You (LMCTFY) se lanzó en 2013 como una versión de código abierto de los contenedores de Google, proporcionando contenedores de aplicaciones Linux.
docker
2013
Todos estos sistemas tenían sus adoptantes y devotos, pero cuando **Docker** emergió en 2013 en el **2016** primer producto comercial

Qué es

Se refiere a varias cosas. Esto incluye un proyecto de la comunidad open source; las herramientas del proyecto open source; Docker Inc., la empresa que es la principal promotora de ese proyecto; y las herramientas que la empresa admite formalmente. El hecho de que las tecnologías y la empresa compartan el mismo nombre puede ser confuso.

Docker Software de TI es una tecnología de creación de contenedores que permite la creación y el uso de contenedores de Linux

Comunidad open source Docker trabaja para mejorar estas tecnologías a fin de beneficiar a todos los usuarios de forma gratuita.

Docker Inc desarrolla el trabajo de la comunidad Docker, lo hace más seguro y comparte estos avances con el resto de la comunidad. También respalda las tecnologías mejoradas y reforzadas para los clientes empresariales.

Como funciona

La tecnología Docker usa el kernel de Linux y las funciones de este, como Cgroups y namespaces, para segregar los procesos.

ofreciendo un modelo de implementación basado en imágenes permitiendo compartir una aplicación, o un conjunto de servicios, con todas sus dependencias en varios entornos, también automatiza la implementación de la aplicación en este entorno de contenedores.

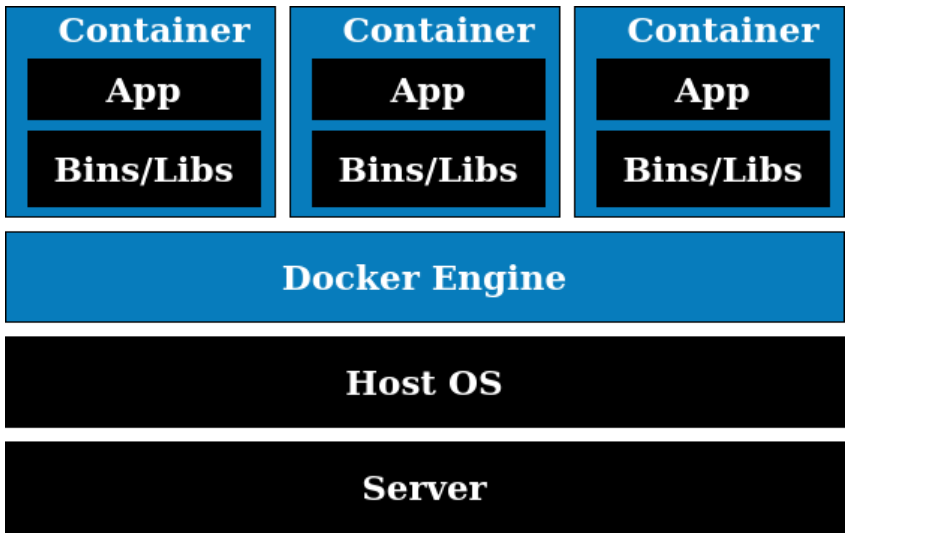
Como funciona

El server es el servidor físico.

El Host OS es la maquina base Linux o Windows.

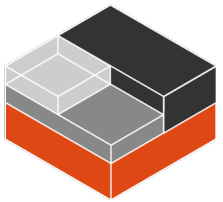
El Docker engine: Conjunto de componentes centrales que permite dirigir el Docker engine por líneas de comando y gestionar las imágenes.

Cada Apps que corre en los Docker containers.



Que son las Imágenes?

Una imagen de contenedor es un paquete ligero, independiente, ejecutable de una pieza de software que incluye todo lo necesario para ejecutarlo: código, tiempo de ejecución, herramientas del sistema, bibliotecas del sistema, configuraciones



Para qué sirven

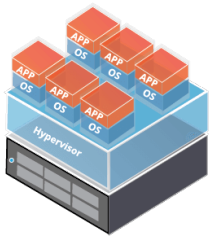
imagen para contener sistemas operativos como Ubuntu con un servidor Apache y tu aplicación web instalada.

para crear contenedores, y nunca cambian.

Para contener elementos básicos como Java, Ubuntu, Apache…etc, que se pueden descargar y utilizar. Normalmente cuando creas imágenes, partimos de una imagen padre a la que le vamos añadiendo cosas.

Qué es la virtualización?

Tecnología que permite crear múltiples entornos simulados o recursos dedicados desde un solo sistema de hardware físico. El software llamado "hipervisor" se conecta directamente con el hardware y permite dividir un sistema en entornos separados, distintos y seguros, máquinas virtuales (VM).



Tipos de virtualización

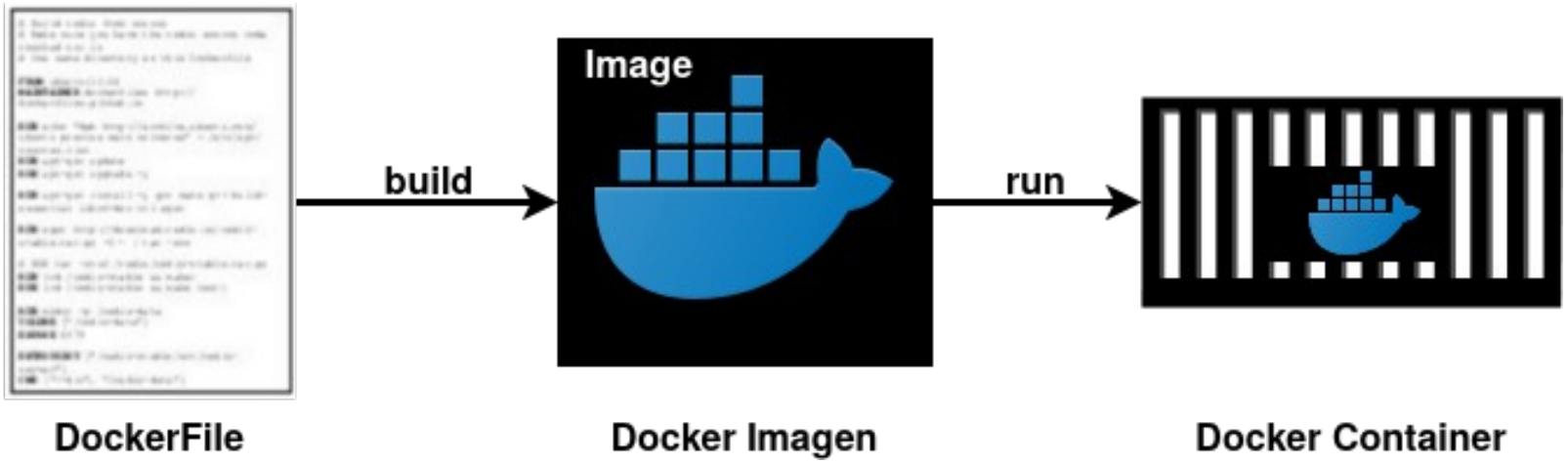
Existen diferentes tipos de distintos procedimientos para conseguir como: virtualizar determinados recursos Virtualización de servidor o por hardware, Virtualización de software o por sistema operativo, Virtualización de red, Virtualización de almacenamiento, Virtualización de memoria y Virtualización de escritorio

Flujo de trabajo de Docker

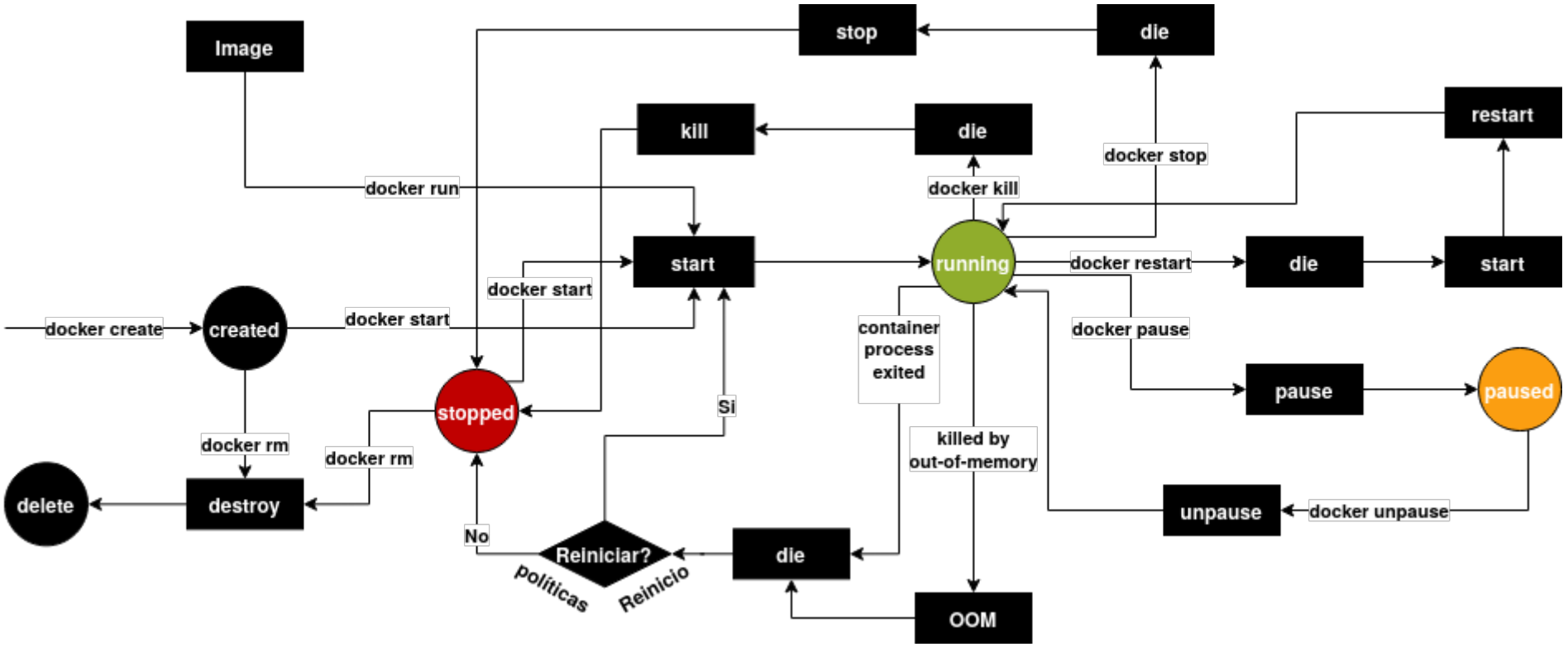
Dockerfile: documento de texto que contiene todos los comandos que un usuario podría llamar en la línea de comandos para ensamblar una imagen.

Imagen: colección ordenada de cambios en el sistema de archivos y los parámetros de ejecución correspondientes para su uso en tiempo de ejecución del contenedor.

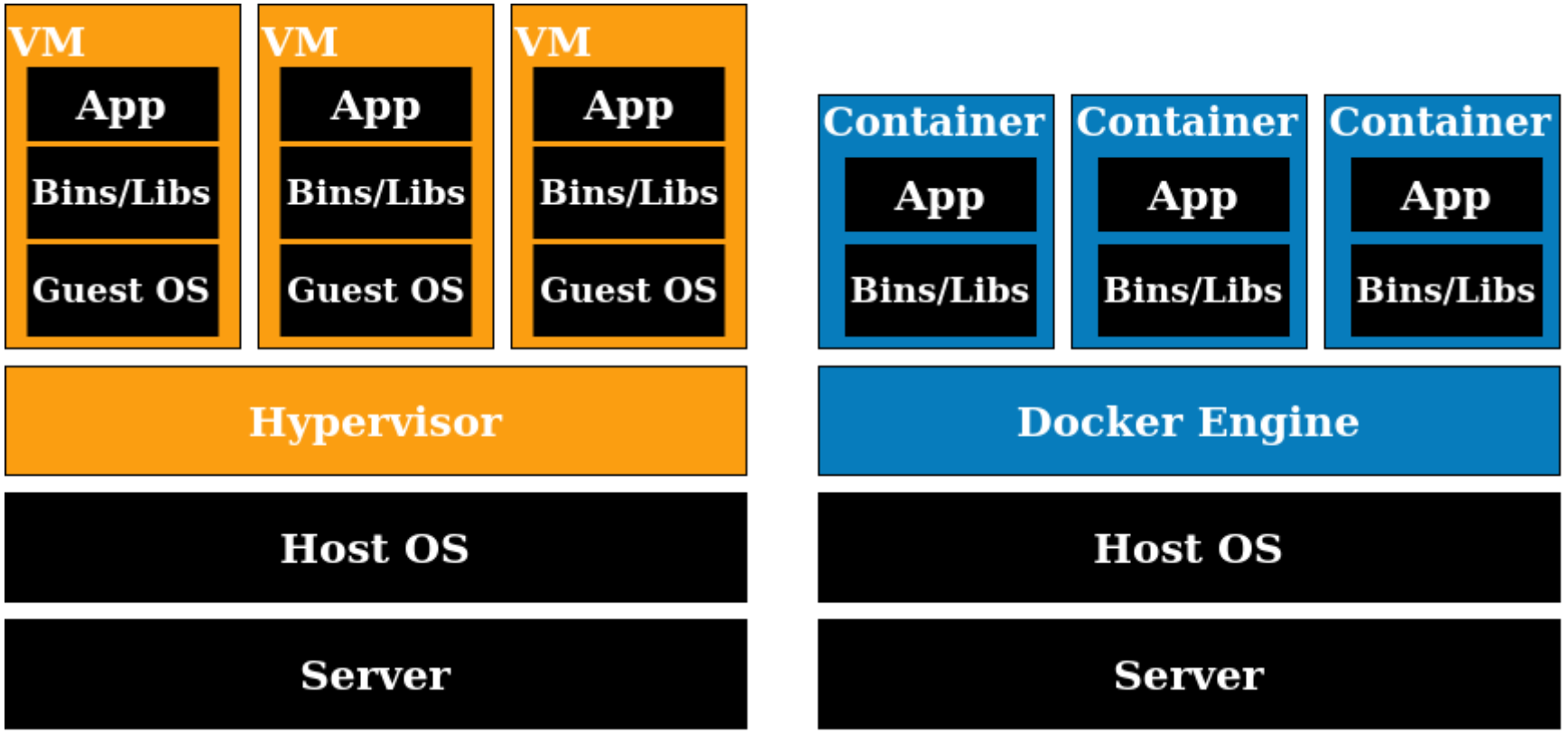
Container: Instancia en tiempo de ejecución de una imagen docker.



Flujo de Estados de Docker



Maquinas Virtuales vs Docker



Maquinas Virtuales vs Docker

Similitudes

Maquinas Virtuales	Docker
Los procesos en una VM se puede ver en otras VM s	Los procesos en un contenedor pueden ver los procesos en otro contenedor
Cada vm tiene su propio root filesytem	Cada Contenedor tiene su propio root filesytem, no su propio kernel
Cada VM tiene su propio adaptador de red	Docker puede obtener adaptadores de red virtuales. Puede tener por searado IP y pouertos
VM es una instancia en ejecución de archivos físicos(VMX y VMDK)	Los contenedores Docker corren de instancias de imagenes Docker
SO del Host puede ser diferente al de los SO del Guest	SO del Host puede ser diferente al de los SO del Contenedor

Maquinas Virtuales vs Docker

Diferencias

Maquinas Virtuales	Docker
Cada VM ejecuta su propio SO	todos los contenedores comparten el kernel del host
El tiempo de arranque es en minutos	Los contenedores inician en segundos
VM,s snapshots es usado, con moderación	Las imágenes se crean de forma incremental sobre otras capas similares
No hay control de versiones,	Las imágenes son diferenciadas y cada versión es controlada. Docker Hub similar a GitHub
No puede ejecutar más que un par de máquinas virtuales en una computadora promedio	Pueden correr muchos Contenedores in un computadora
Solo se puede iniciar un VM desde un set de archivos VMX y VMDK	Múltiples contenedores Docker pueden iniciar desde una imagen Docker

Maquinas Virtuales vs Docker

Factor	Maquinas Virtuales	Containers
Soporte de OS	Requiere guest OS	El SO del Host puede ser compartido
Tiempo de Arranque	Requiere tiempo de inicio parecido al SO tradicional	El inicio es muy rápido
Estandarización	las Maquina Virtuales son generalmente de un SO especifico	Los Contenedores son de Aplicaciones especificas
potabilidad	Menos potabilidad	Mas Portable
Necesidad de servidores	Mas Requerimiento del Servidor	Menos cantidad requerimientos del servidor
Seguridad	cada sistema operativo que se ejecuta en una máquina virtual puede proporcionar seguridad depende de los datos del usuario. la seguridad depende también del hipervisor	la seguridad puede ser una sobrescrita en el kernel y compartida entre las aplicaciones. entonces podemos decir que carece de medidas de seguridad
Baja redundancia	más información redundante ya que cada máquina virtual tiene su propio SO	se comparte un solo SO, por lo que no hay mucha información redundante
Acceso al Hardware	El acceso directo al Hardware no es posible	Es posible el acceso directo al Hardware
recursos distribuidos	gran cantidad de recursos requeridos	Se necesitan menos recursos en comparación a VM
requerimiento de Memoria	Gran requerimiento de memoria ya que cada guest tiene su propio SO	se requiere menos memoria ya que el sistema operativo se comparte entre las aplicaciones
Compartir archivos y librerías	compartir archivos y librerías no es posible	Los archivos pueden ser compartidos usando comando Linux como scp

Instalar contenedores directamente en el sistema operativo.?

A favor

elimina sobrecarga, utilizan los recursos del sistema de manera más eficiente ,El uso eficiente de los recursos de memoria, procesamiento y redes

En Contra

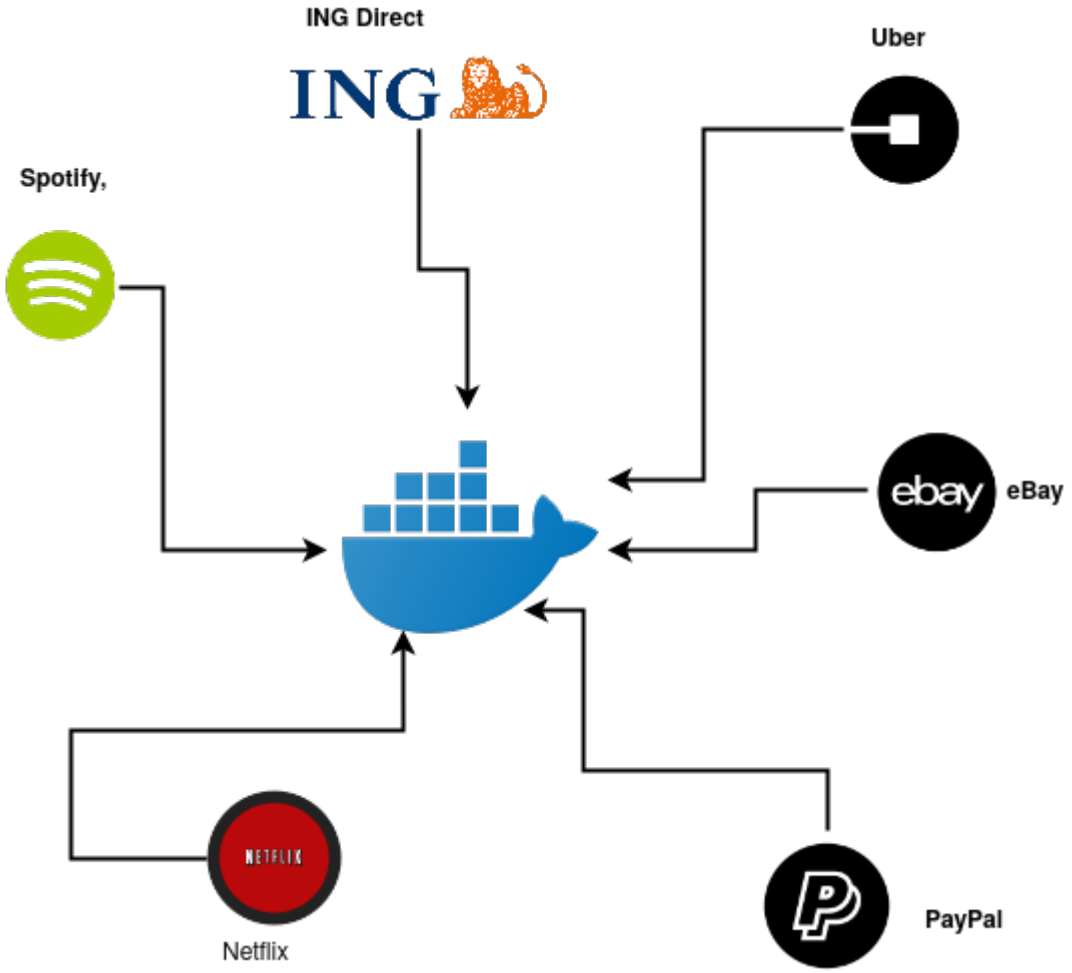
La Seguridad, oportunidad para un ataque si el contenedor pudiera verse comprometido.

La falta de aislamiento también puede permitir que diferentes contenedores monopolicen los recursos del host.

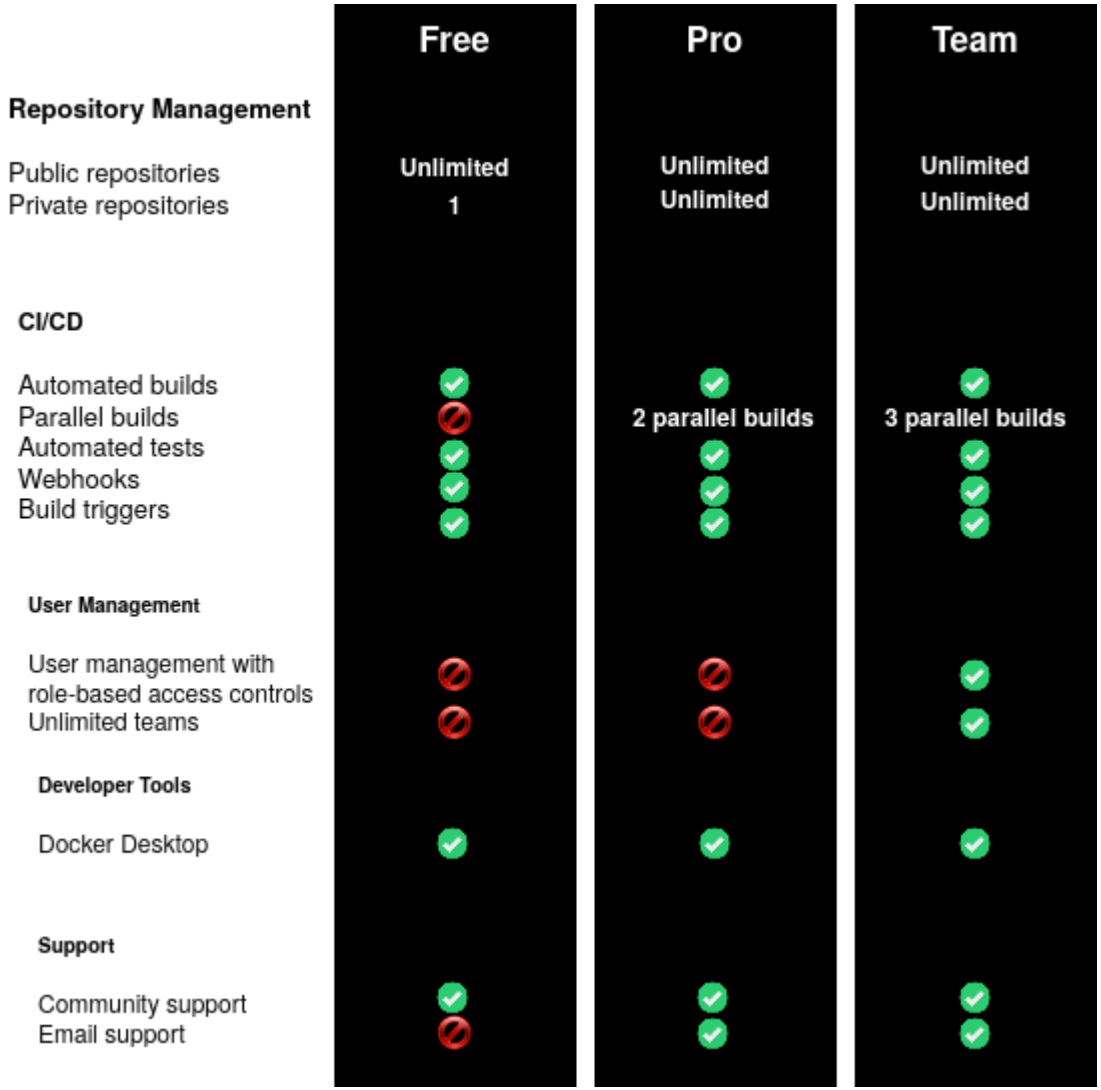
Dado que el contenedor hace uso del núcleo del host, podría haber problemas al intentar portarlos entre diferentes plataformas

Las actualizaciones físicas del servidor son difíciles

Empresa que usan Docker



Docker Planes



	Free	Pro	Team
Repository Management			
Public repositories	Unlimited	Unlimited	Unlimited
Private repositories	1	Unlimited	Unlimited
CI/CD			
Automated builds	✓	✓	✓
Parallel builds	✗	2 parallel builds	3 parallel builds
Automated tests	✓	✓	✓
Webhooks	✓	✓	✓
Build triggers	✓	✓	✓
User Management			
User management with role-based access controls	✗	✗	✓
Unlimited teams	✗	✗	✓
Developer Tools			
Docker Desktop	✓	✓	✓
Support			
Community support	✓	✓	✓
Email support	✗	✓	✓

Práctica
Introducción a Docker

Requisitos para la práctica.

- Docker Hub Account.
- Maquina linux en Play With Docker con instancia iniciada (easy)
- Maquina linux (virtual o real) con docker Community edition Instalado (Solo para valientes)

Comenzando con Docker Image

Verificar versión de docker.

Para verificar la versión de docker que tenemos instalada ejecutamos en una consola de comandos:

```
docker --version
```

Hello World en Docker.

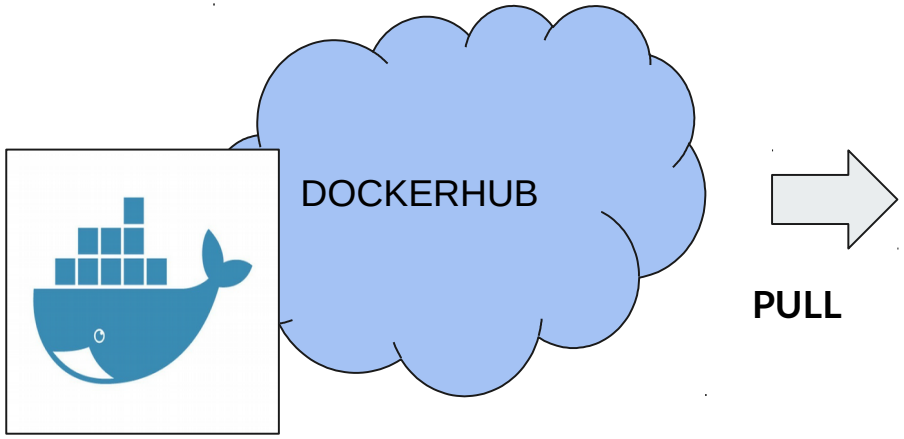
Imagen **Contenedor en Ejecución**

Hello World en Docker.

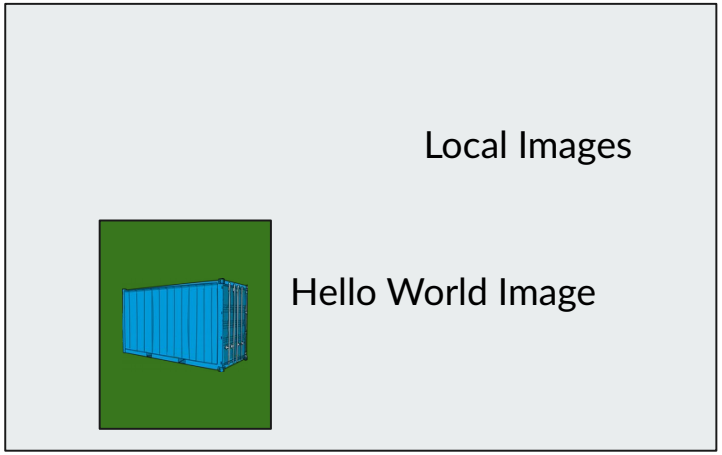
Para descargar la imagen y desplegar el contenedor docker del ejercicio hello world ejecutamos:

```
docker run hello-world
```

Hello World en Docker.



PULL



RUN

La imagen local existe?

- SI: Crear contenedor en ejecución
- No: Buscar en Dockerhub y descargar, luego crear contenedor en ejecución.

Hello World en Docker.

Vamos a ver qué imágenes tenemos en el repositorio local con el comando

```
docker images
```

```
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
hello-world   latest    bf756fb1ae65   6 months ago    13.3kB
```

Hello World en Docker.

Vamos a ver qué contenedores están corriendo en nuestro servidor/equipo.

```
docker ps
```

```
CONTAINER ID   IMAGE   COMMAND   CREATED   STATUS   PORTS   NAMES
```

Hello World en Docker.

Vamos a ver todos los contenedores están corriendo en nuestro servidor/equipo.

```
docker ps -a
```

Hello World en Docker.

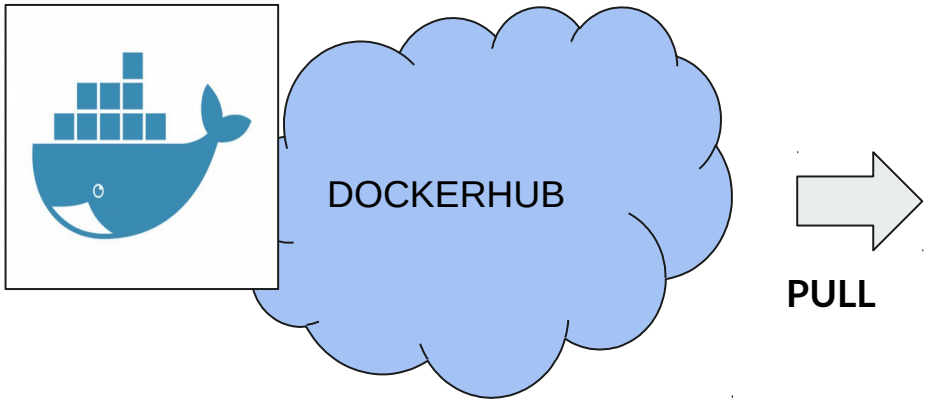
Para inspeccionar una imagen en particular..

```
docker inspect <imagename>
```

```
docker inspect hello-world
```

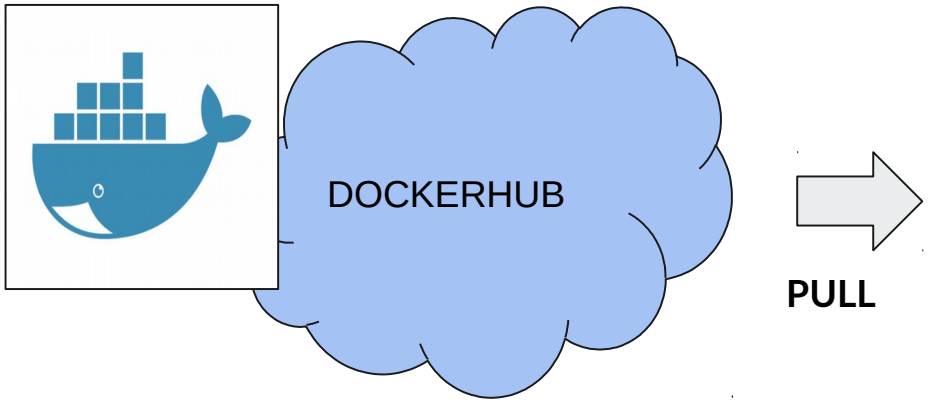
Imagenes en Docker.

Imagenes en Docker.

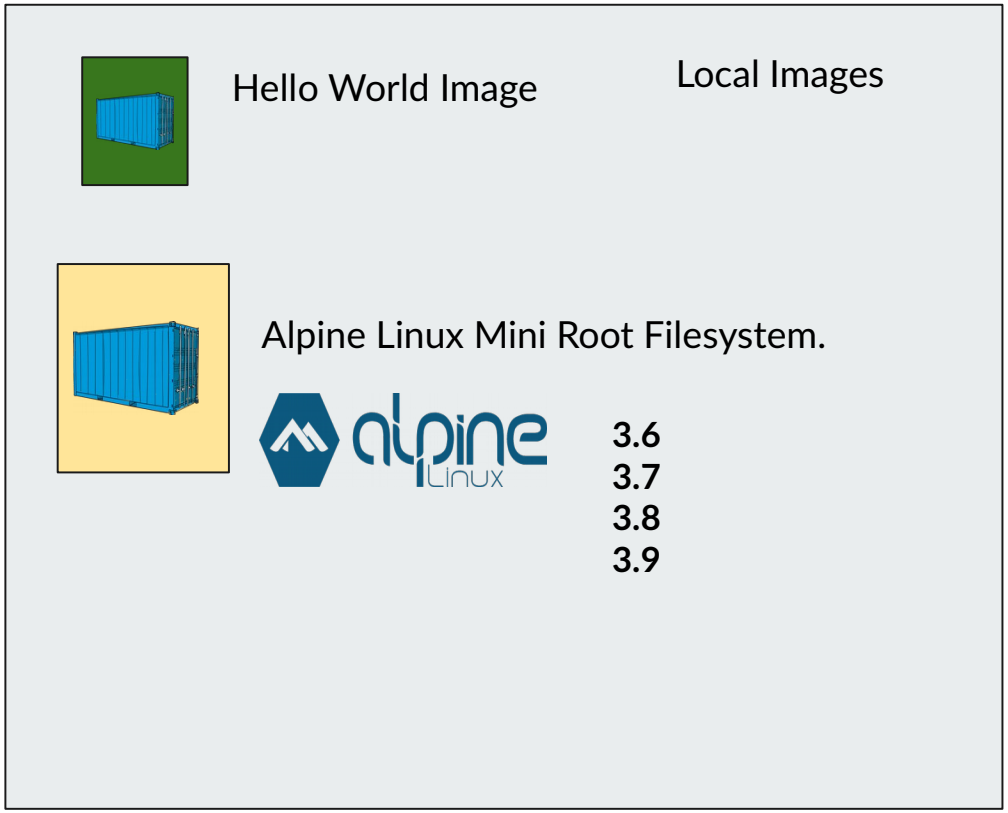


- Listar Imágenes.
- Listar Imagenes particulares.
- Ver el ID completo de las imágenes.
- Buscar imagenes por un filtro.
- Eliminar imágenes.

Imagenes en Docker.



PULL



- `docker pull alpine:3.6`
- `docker pull alpine:3.7`
- `docker pull alpine:3.8`
- `docker pull alpine:3.9`

Imagenes en Docker.

Vamos a ver qué imágenes tenemos en el repositorio local con el comando

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	bf756fb1ae65	6 months ago	13.3kB
alpine	3.6	43773d1dba76	16 months ago	4.03MB
alpine	3.7	6d1ef012b567	16 months ago	4.21MB
alpine	3.8	dac705114996	16 months ago	4.41MB
alpine	3.9	5cb3aa00f899	16 months ago	5.53MB

Información de imágenes en Docker.

Para ver información de una imagen específica.

```
docker images alpine:3.6
```

```
REPOSITORY   TAG        IMAGE ID       CREATED          SIZE
alpine       3.6        43773d1dba76   16 months ago    4.03MB
```

Información de imágenes en Docker.

Para filtrar información de un conjunto de imágenes.

```
docker images --filter=reference='alpine'
```

```
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
alpine       3.6       43773d1dba76   16 months ago    4.03MB
alpine       3.7       6d1ef012b567   16 months ago    4.21MB
alpine       3.8       dac705114996   16 months ago    4.41MB
alpine       3.9       5cb3aa00f899   16 months ago    5.53MB
```

Información de imágenes en Docker.

Para conocer los comandos que podemos ejecutar con imágenes.

```
docker images --help
```

Información de imágenes en Docker.

Por ejemplo ver el ID completo de las imágenes.

```
docker images --no-trunc
```

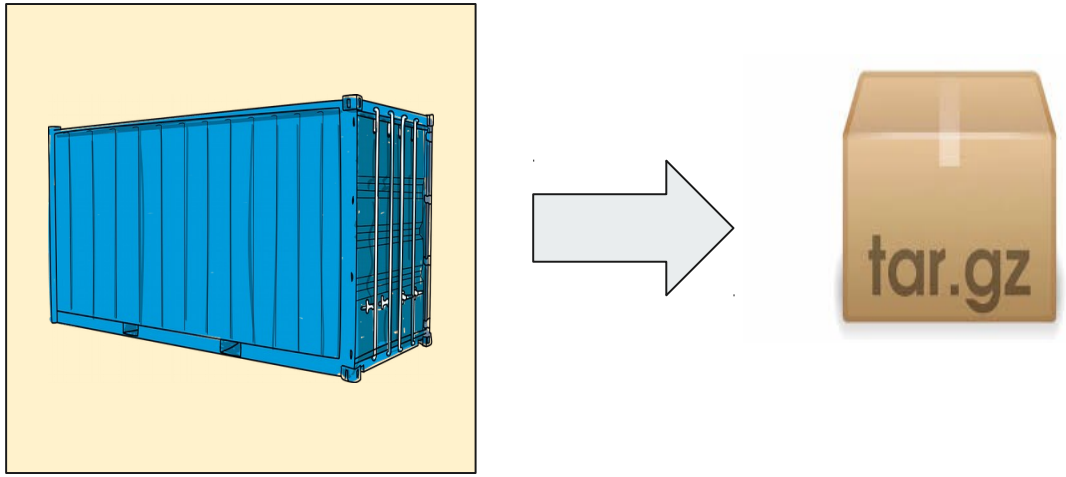
Eliminar imágenes en Docker.

Para eliminar una o varias imágenes específicas.

```
docker rmi <your-image-id/name> <your-image-id/name>
docker rmi 43773d1dba76
```

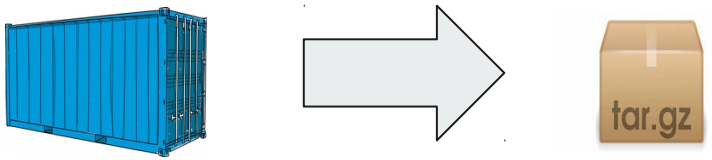
NO se pueden eliminar imágenes de contenedores que estén desplegados.

Guardar imágenes y contenedores como archivos Tar para compartir

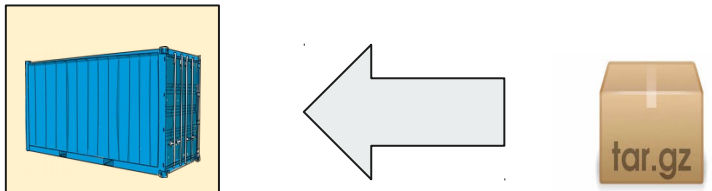


Comandos Básicos

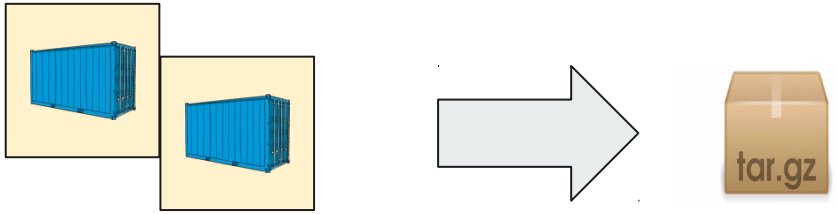
docker export



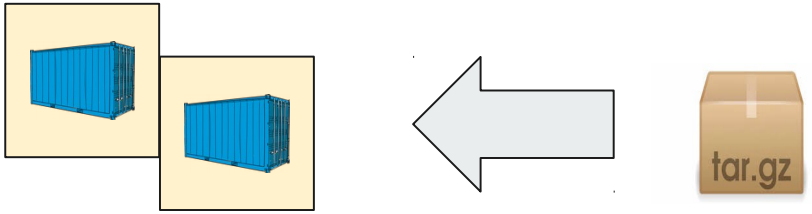
docker import



docker save



docker load



Creando contenedor con Nginx.

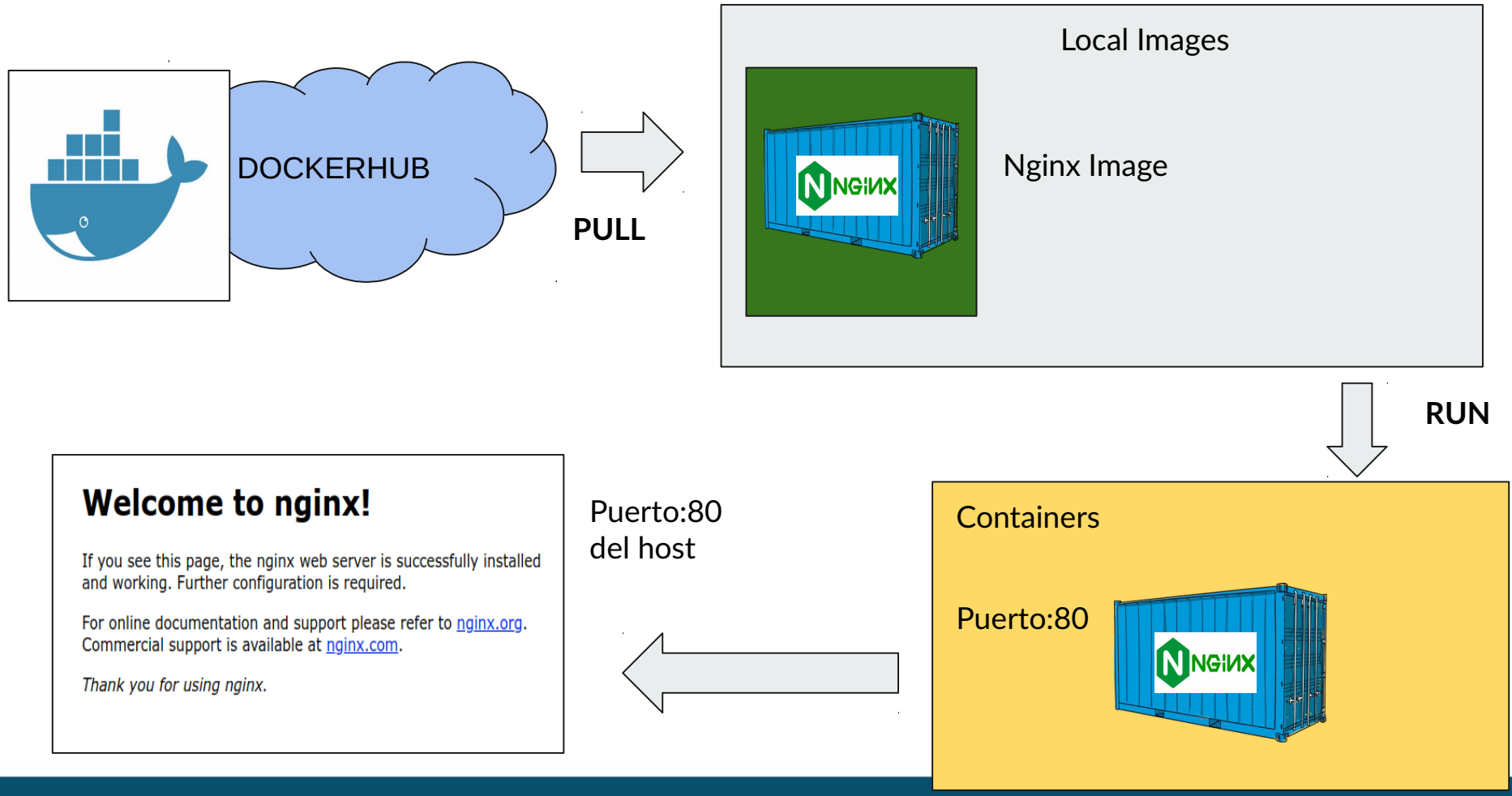
Para crear el servidor con Nginx (Nginx es un servidor web/proxy inverso ligero de alto rendimiento).

```
docker run --name nginx1 -d -p 80:80 nginx
```

-d Ejecute el contenedor en segundo plano e imprima la ID del contenedor

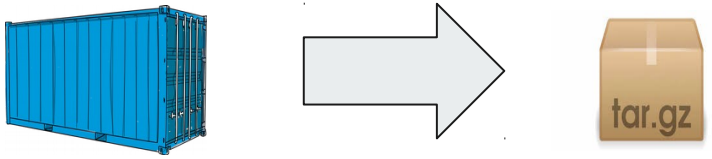
-p Publicar los puertos de un contenedor en el host

Servidor Web NGINX Con Docker



Comandos Básicos

docker export



Exportar el sistema de archivos de un contenedor como un archivo tar

Ver los contenedores que están en ejecución.

Para ver todos (-a) los contenedores que están en ejecución actualmente ejecutamos.

```
docker ps -a
```

```
CONTAINER ID  IMAGE  COMMAND                  CREATED         STATUS          PORTS                NAMES
8956a0c0a1f2  nginx  "/docker-entrypoint.…"   11 minutes ago  Up 11 minutes   0.0.0.0:80->80/tcp   nginx1
```

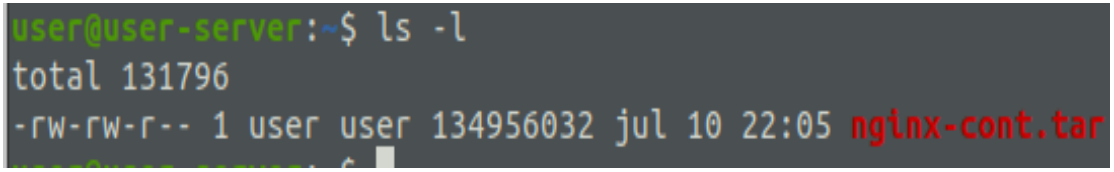
Exportar el contenedor en un archivo Tar.

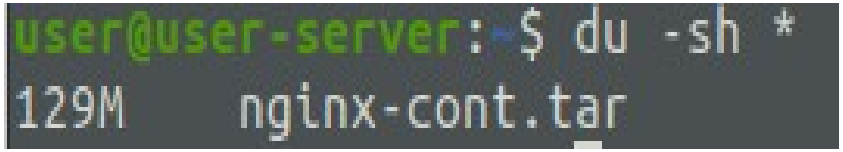
Para exportar el contenedor de Nginx a un archivo Tar ejecutamos:

```
docker export nginx1 > nginx-cont.tar
```

Podemos listar el archivo y ver su tamaño con los comandos:

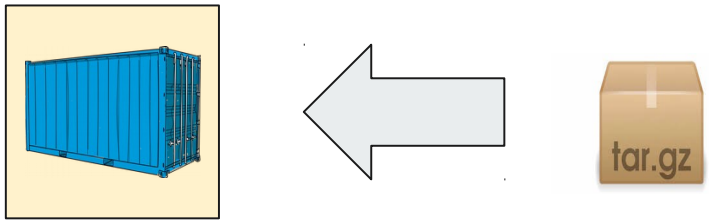
```
ls -l | du -sh *
```





Comandos Básicos

docker import

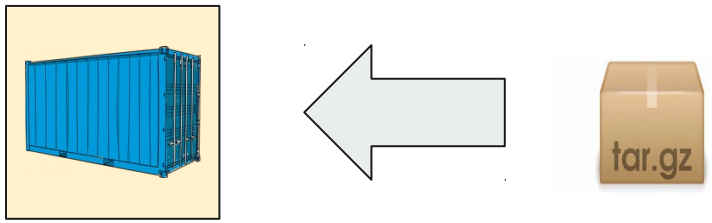


Importar el contenido de un archivo tar para crear una imagen del sistema de archivos de un contenedor.

Importar el archivo Tar como una Imagen.

Para importar el archivo tar como imagen ejecutamos:

```
docker import - mynginx < nginx-cont.tar
```



Imagenes en Docker.

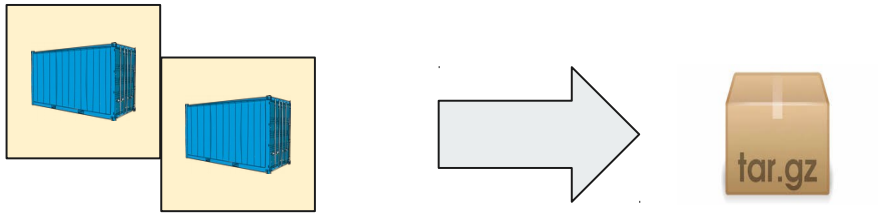
Vamos a ver qué imágenes tenemos en el repositorio local con el comando

```
docker images
```

```
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
mynginx      latest    ae9120001c7c   4 seconds ago    130MB
nginx        latest    2622e6cca7eb   4 weeks ago      132MB
```

Comandos Básicos

docker save



Guardar una o más imágenes en un archivo tar.

Imagenes en Docker.

Para exportar una o varias imágenes del repositorio en un archivo Tar

```
docker save -o nginx-img.tar nginx
```

```
ls -l | du -sh *
```

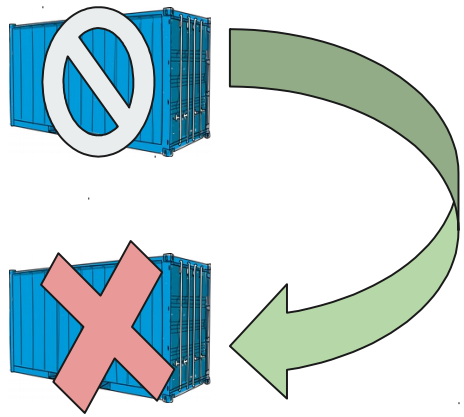
```
user@user-server:~$ ls -l
total 265280
-rw-rw-r-- 1 user user 134956032 jul 10 22:05 nginx-cont.tar
-rw------- 1 user user 136687616 jul 10 22:06 nignx-img.tar
```

```
user@user-server:~$ du -sh *
129M    nginx-cont.tar
131M    nignx-img.tar
```

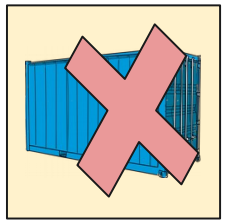
Comandos Básicos

docker stop <container-id/name>

docker rm <container-id/name>



docker rmi <image-id>



Ver los contenedores que están en ejecución.

Para ver todos (-a) los contenedores que están en ejecución actualmente ejecutamos.

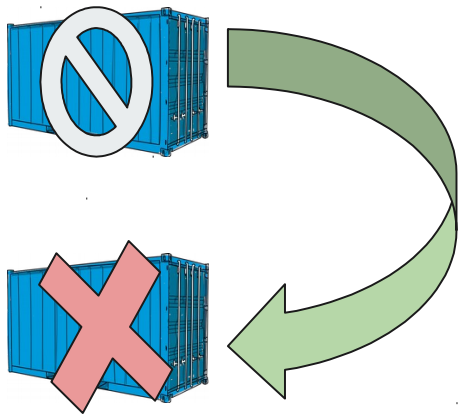
```
docker ps -a
```

```
CONTAINER ID   IMAGE   COMMAND                  CREATED         STATUS          PORTS                NAMES
```

Comandos Básicos

docker stop nginx1

docker rm nginx1



Detener y Remover un contenedor.

Imagenes en Docker.

Vamos a ver qué imágenes tenemos en el repositorio local con el comando

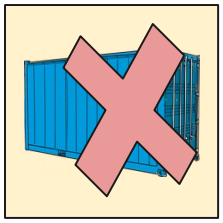
```
docker images
```

```
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
mynginx      latest    ae9120001c7c   4 seconds ago    130MB
nginx        latest    2622e6cca7eb   4 weeks ago      132MB
```

Comandos Básicos

docker rmi mynginx

docker rmi nginx



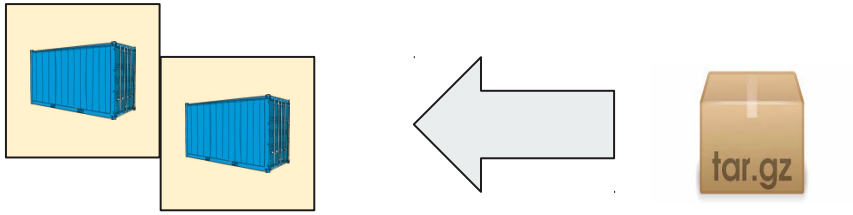
Imagenes en Docker.

Cargar imágenes desde un archivo Tar

```
docker load < nginx-img.tar
```

---

docker load



Cargar una o más imágenes desde un archivo Tar

Imagenes en Docker.

Vamos a ver qué imágenes tenemos en el repositorio local con el comando

```
docker images
```

```
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
nginx        latest    2622e6cca7eb   4 weeks ago     132MB
```

Acceso y gestión de contenedor Docker

Contenedor con versión mínima de UBUNTU.

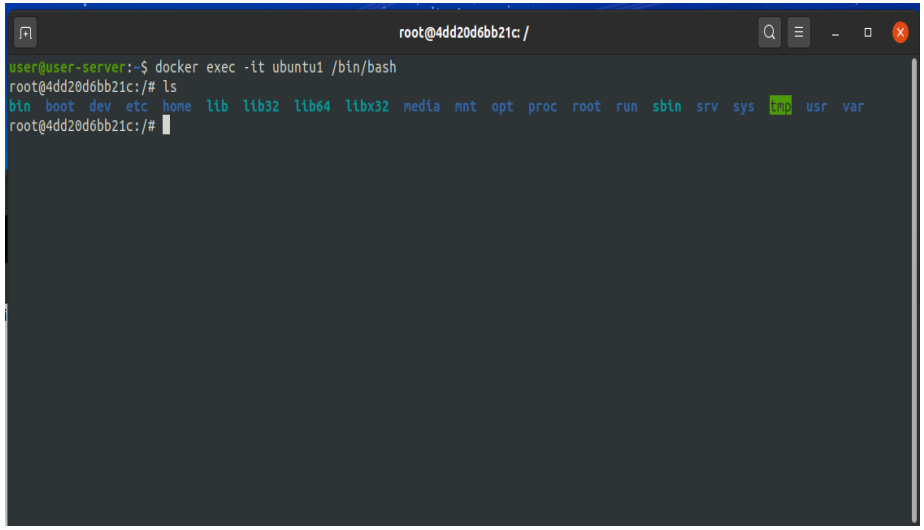
Vamos a desplegar una versión mínima de ubuntu en un contenedor con el comando:

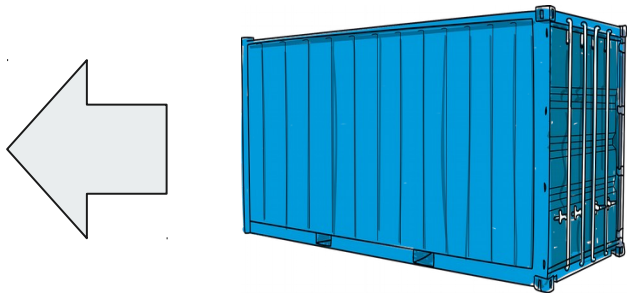
```
docker run --name ubuntu1 -dit ubuntu
```

Accediendo a la consola del Contenedor.

Para acceder a la consola del contenedor que llamamos (ubuntu1):

```
docker exec -it ubuntu1 /bin/bash
```





Saliendo de la consola del Contenedor.

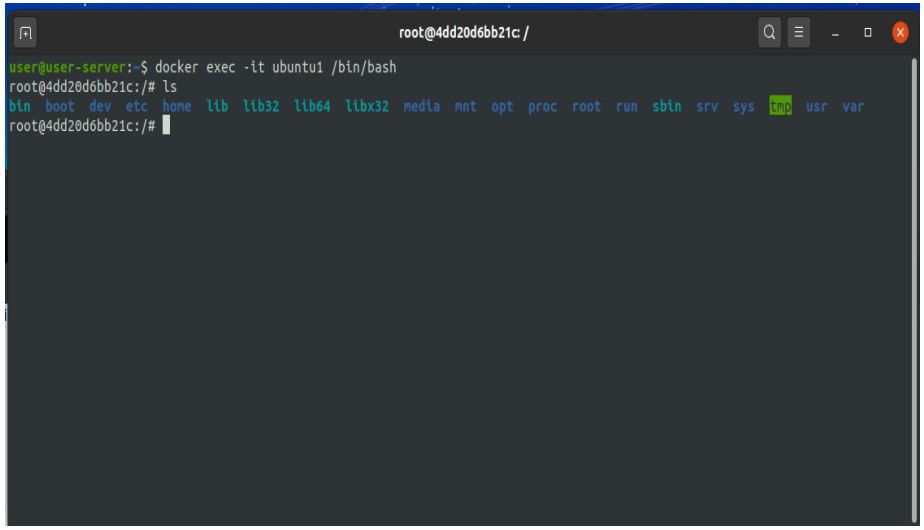
Para salir de la consola del contenedor que llamamos (ubuntu1):

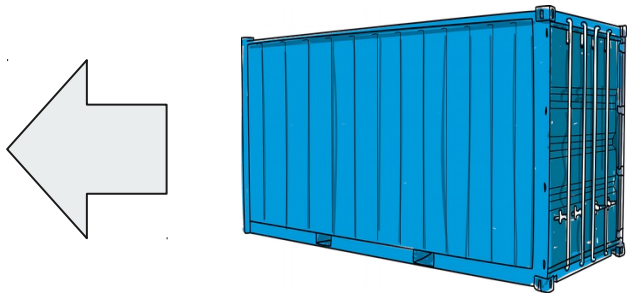
```
exit
```

Accediendo a la consola del Contenedor.

Para acceder a la consola del contenedor que llamamos (ubuntu1):

```
docker attach ubuntu1
```





Saliendo de la consola del Contenedor.

Para salir de la consola del contenedor que llamamos (ubuntu1):

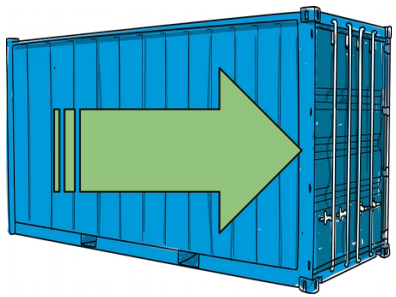
```
exit
```

Cuando usamos el comando **docker attach** al dar la instrucción exit para salir el contenedor se detiene, similar a docker stop.

Iniciando nuevamente el Contenedor.

Para iniciar el contenedor que llamamos (ubuntu1):

```
docker start ubuntu1
```



Gracias por la atención. ¿Preguntas?

Referencias:

Docker Oficial Page:
https://www.docker.com/

Collabnix Docker Beginners Track:
http://dockerlabs.collabnix.com/beginners/README.html