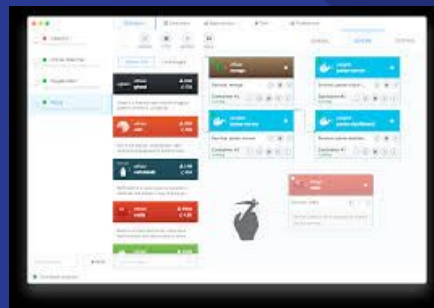
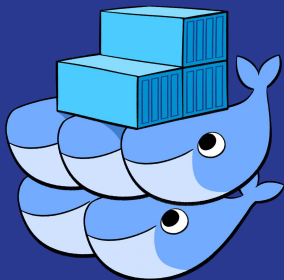




kubernetes

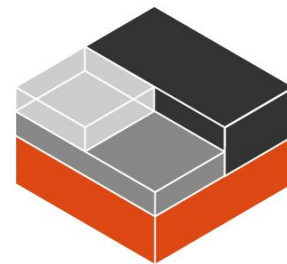


docker

Contenedores con Docker.

I) Comenzando.

Daniel A. Cialdella C., José M. Cousiño y Pablo P. Corral L.



Espacio ofrecido por
Travel Labs Madrid



28-03-2019



RANCHER

Presentación.

¿ Quiénes somos y porqué hicimos Dockertips ?

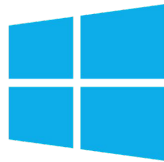
2015. Comenzamos probando Docker.

2016. Implementaciones para desarrolladores.

2017. www.dockertips.com y publicaciones.

2018. Puesta en marcha de varios Proyectos.

2019. Comunidad establecida, compartiendo experiencias y dando servicios.



ITProfesionales
I.T. Profesionales para profesionales.

security
stability
reliability

debian
GNU/Linux

DEBIAN

martes, 14 de abril de 2015

Docker la nueva evolucion

En esta semana estamos probando Docker.
Un buen link aqui.
<http://kencochrane.net/blog/2013/08/the-docker-guidebook/>

Para mi, es un salto cuántico, tal como lo fue por el 98 con VMWare.
Un producto para evaluar, entender, usar y prepararse para el día a día.

Otros links con mas datos.
<http://blog.thoward37.me/articles/where-are-docker-images-stored/>
<http://stackoverflow.com/questions/21486004/how-do-i-move-a-docker-containers-image-to-a-persistent-disk>
<http://stackoverflow.com/questions/23935141/how-to-copy-docker-images-from-one-host-to-another-without-via-repository>



MeetUps Docker.

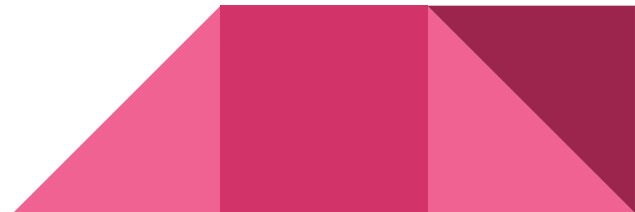
I) Introducción.

- ¿Qué es la tecnología Contenedores?
- Comparativa con virtualización y con aislamiento de APPs.
- Sistemas operativos donde corre y entornos.
- Conceptos generales de Contenedores.
- Imágen, contenedor, almacenamiento, red, recursos asignados.
- Ecosistema de Contenedores. (productos, servicios)
- Herramientas de control básica de contenedores.

II) Docker-Compose vs Docker-Swarm

III) Kubernetes.

IV) Casos reales aplicados.



Características de Contenedores.

- ¿Qué son los contenedores? Abstracción, aislamiento, portabilidad.
- Entorno limpio, seguro, reproducible y aislado para aplicaciones.
- Incluye sus dependencias y bibliotecas autocontenidas.
- Admite automatización de procesos. Instant reply.
- Tecnologías previas. (chroot, jail, Solaris containers, lxc, openvz 2005)
- Funciona como demonio/servicio. Sobre Linux/OSx/Windows.
- Modo local, servidor o en Internet (“la nube”), centralizado o distribuido.
- Aislamiento, recursos, entorno standard, empaquetado, pruebas y seguridad.
- Soluciones básicas (C.E.), profesionales (E.E.) y distribuidas.

Docker CE, openshift , coreos , openvz , linux containers, lxc, lxd , docker stats
c advisor, scout, datadog, rocker, prometheus+grafana+influxdb
dockerana, weave , nebula , docker compose, docker-swarm, openstack
minikube, haven, Kubernetes, Kitematic, Dockstation, portainer
captain, shipyard, docker compose ui, rancher, nomad, podman

<https://dockertips.com/ecosistema>

Instalación.



Instalación.

- ❏ Linux

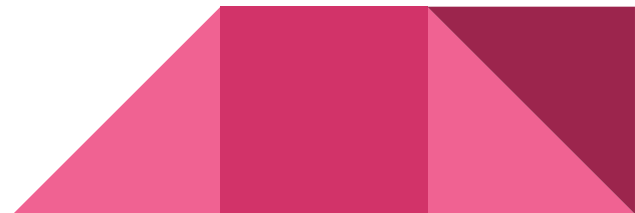
https://dockertips.com/instalando_docker

- ❏ Windows

<https://dockertips.com/instalando-docker-en-windows-10>

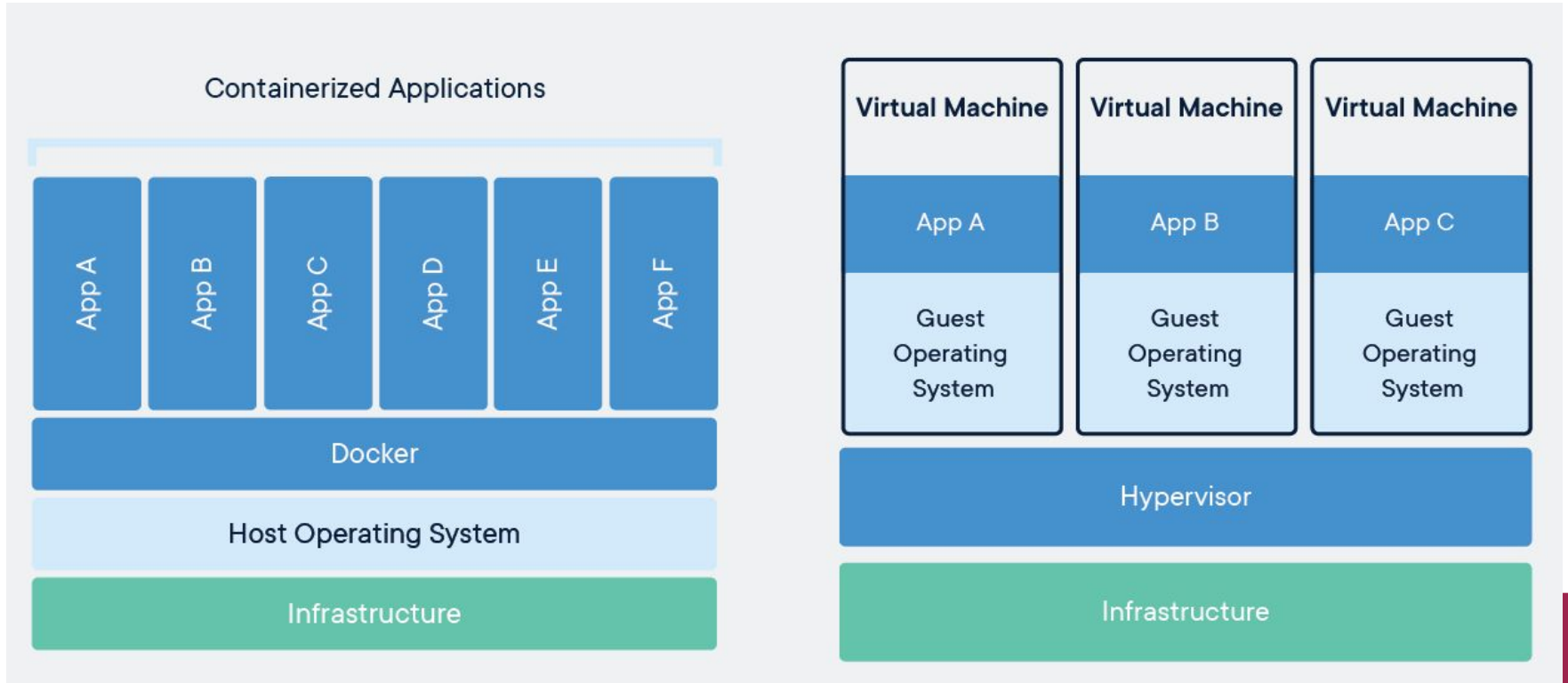
- ❏ Mac

<https://dockertips.com/instalar-docker-en-mac-osx>



Máquinas virtuales vs Docker.

Comparativa Contenedor vs VM.





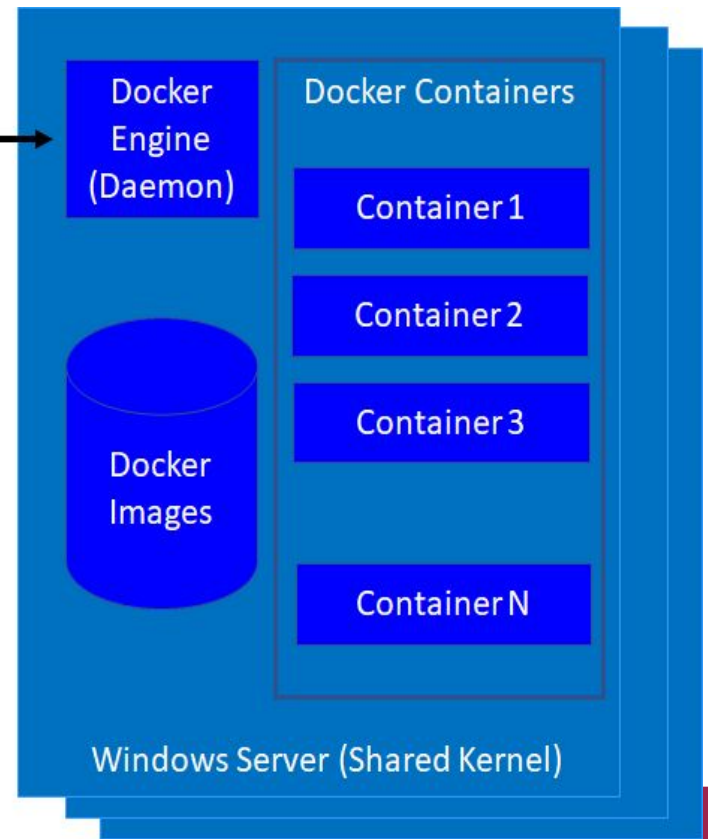
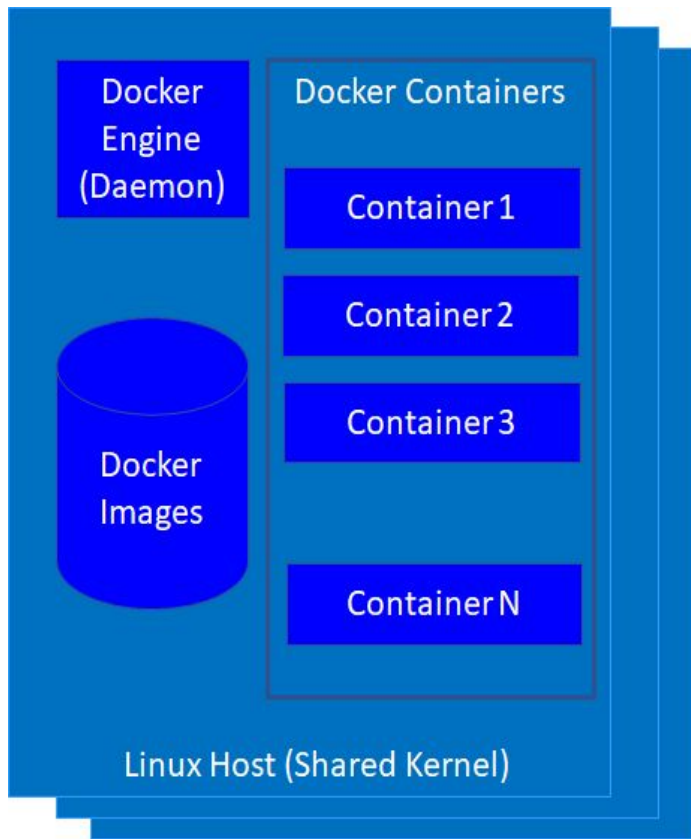
Arquitectura y componentes.

Componentes.

- Un demonio/servicio (dockerd).
- Un programa de control (docker).
- Creación de Imágenes o usando las que existen.
 - Sitio web. <https://hub.docker.com/> Repositorio de imágenes.
- El contenedor como unidad central.
- Cada contenedor tendrá IP, Espacio, nombre. (CPU y RAM compartidas)
- Usa el kernel del equipo real.
- Herramienta de monitorización, gestión, interacción y arranque.


https://dockertips.com/monitorizacion_en_docker







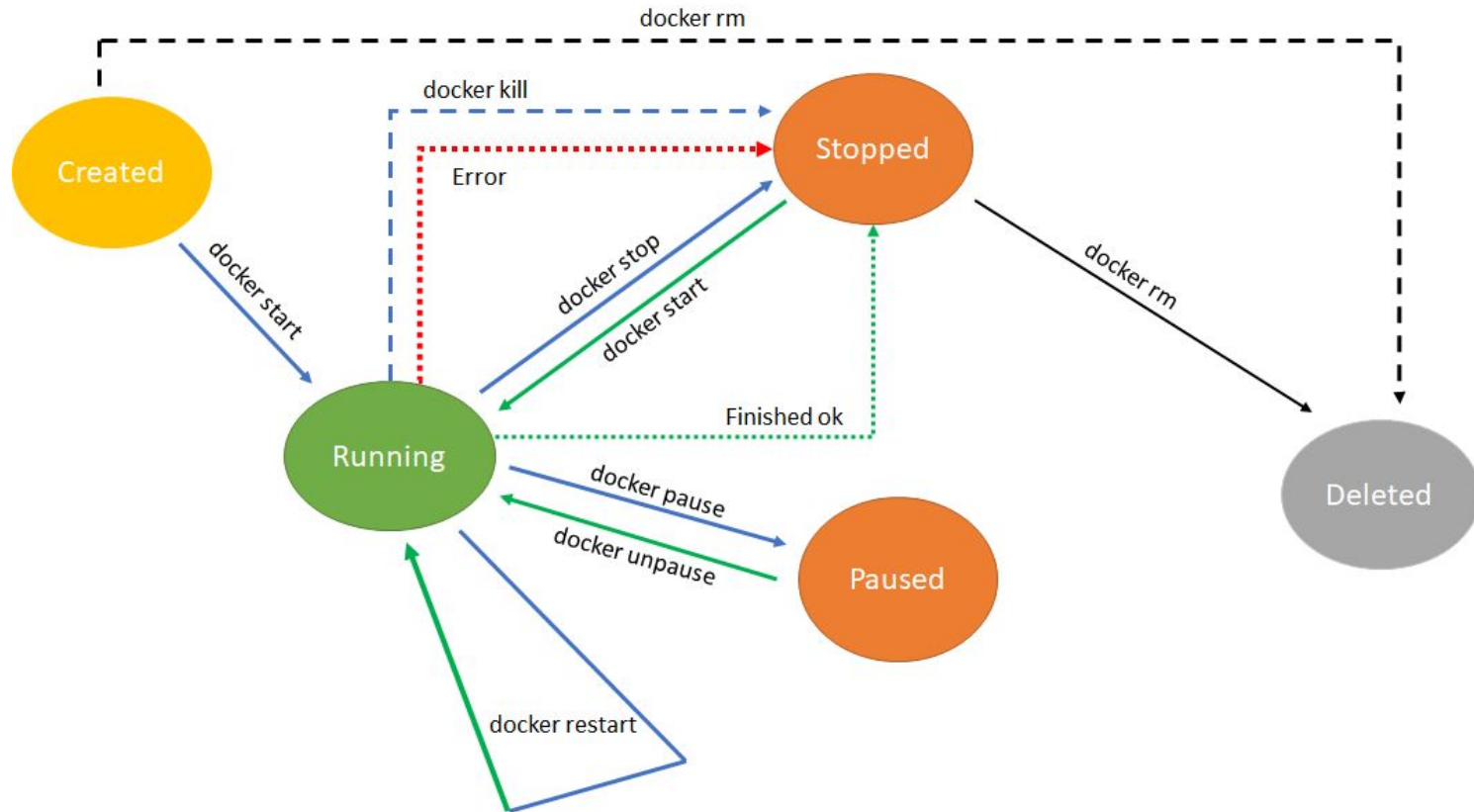
Contenido de un contenedor.

- **Librerías del sistema operativo:** dentro del contenedor se tiene todo lo necesario de una sistema operativo para aislarlo del sistema que lo que ejecuta (Host)
 - **Herramientas del sistema:** herramientas que nos facilitan el trabajo con los contenedores:
 - Editores de texto
 - Monitorización
 - Registro
 - **Runtime:** el software que se necesita para ejecutar la aplicación dentro del contenedor:
 - Runtime de Net, Net Core, ...
 - Máquina virtual de java
 - **Código de la aplicación** con sus recursos.
- 

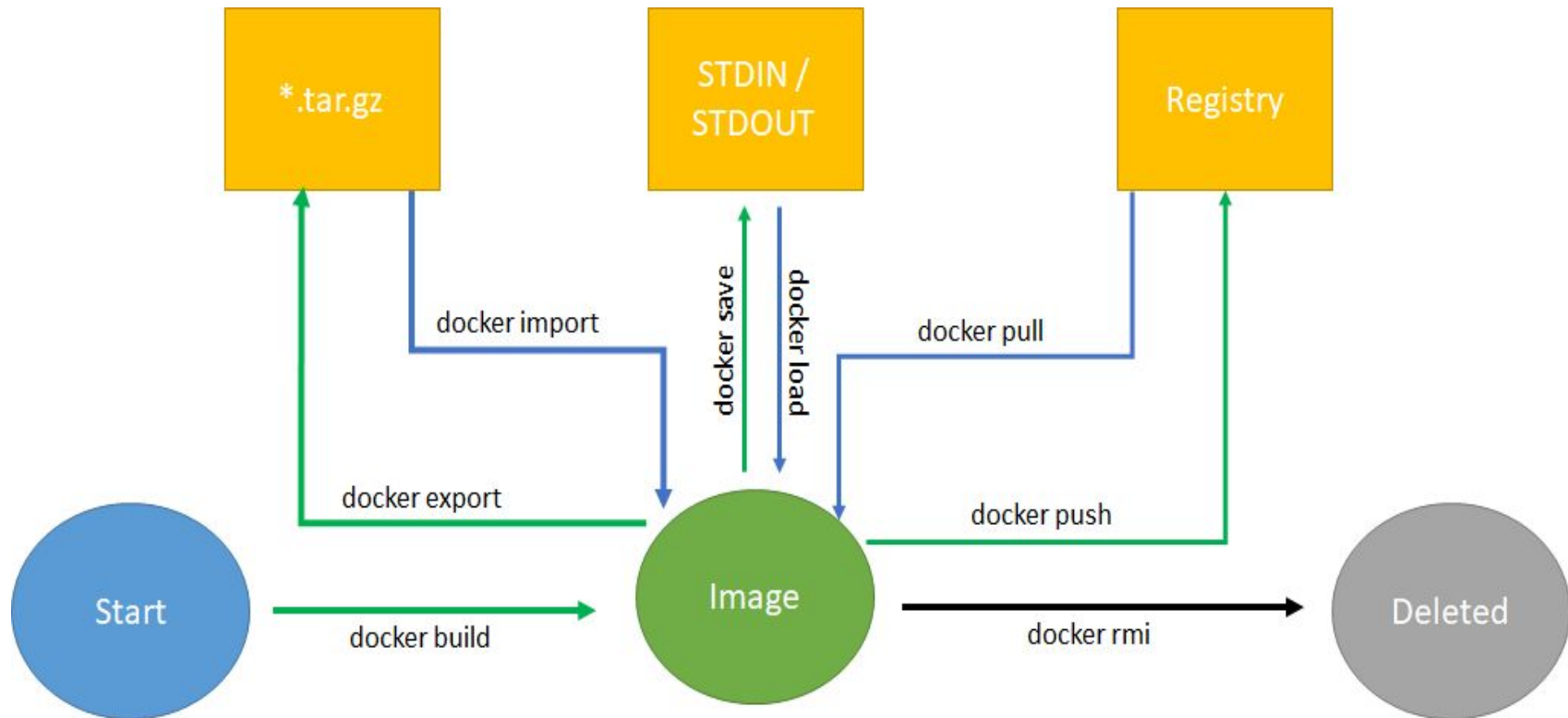
Imágenes y Contenedores.

- Una **imagen** describe los contenido que tienen el contenedor:
 - Configuración
 - Proceso
 - Aplicación
- Un **contenedor** es el resultado de ejecutar una imagen
- Es parecido a la programación orientada a objetos:
 - Clase es la imagen
 - Objeto es el contenedor.
- Las imágenes no se ejecutan.
- Los contenedores tienen un ciclo de vida.
- Las imágenes se pueden copiar entre hosts, el contenedor no.
- Solo se puede crear contenedores de imágenes descargadas en el sistema

Ciclo de vida.



Manejo e interacción.





Docker Hub.

- Docker Hub es un servicio de registro en la nube de imágenes de contenedores.

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

Google, Amazon, Azure y otros.

- Ofrece las siguientes características:
 - Repositorio de imagenes (públicos y privados)
 - Compilaciones automatizadas
 - WebHooks (acciones después de una compilación)
 - Integración con GitHub y Bitbucket
- Búsqueda de imágenes con. docker search “ubuntu”
- Las imágenes que se suban tienen que tener la siguiente estructura
<nombre usuario>/<nombre repositorio>[:tag]





Docker CLI.

- Docker es una herramienta basada en la CLI (Interfaz de línea de comandos)
- Todos los comandos se ejecutan desde la línea de comandos.
- Principales comandos:
 - **pull**: descarga una imagen.
 - **images**: permite listar las imágenes que hay en la máquina.
 - **build**: construye una imagen personalizada.
 - **run**: crea un contenedor a partir de una imagen.
 - **start**: inicia un contenedor.
 - **stop**: para un contenedor.
 - **rm**: elimina un contenedor.
 - **rmi**: elimina una imagen.
 - **exec** corre un programa en un contenedor.
 - **ps**: muestra los contenedores en ejecución.
Con -a muestra los contenedores que están detenidos.






Fichero Dockerfile.

- Para crear imágenes personalizadas se necesita de un fichero llamado Dockerfile.
- Este fichero contiene un conjunto de instrucciones para que Docker cree la imagen.
- Es un fichero de texto plano donde cada línea es una instrucción para crear la imagen.
- Importante, el fichero no tiene extensión.
- Este fichero no es parte de la imagen resultante.



- Una imagen contiene:
 - Un conjunto de archivos. Es un sistema de archivos completo.
 - Un conjunto de puertos expuestos: lista de los puertos expuestos al exterior. Son puertos del contenedor no del Host.
 - Una configuración en forma de variables de entorno.
 - Un comando inicial que es el que se ejecuta por defecto en todos los contenedores.
 - Todos los comandos de Dockerfile están orientados a especificar estos conceptos.
- 

- Comandos de Dockerfile:

- **FROM:** indica la imagen base. Generalmente se utiliza una del repositorio de DockerHub o de cualquier otro repositorio.
- **ENV:** define una variable de entorno.
- **WORKDIR:** establece un directorio en el que se despliega la aplicación. Se pueden usar varios comandos WORKDIR.
- **COPY:** copia ficheros de la aplicación.
- **RUN:** ejecuta un comando
- **CMD o ENTRYPOINT:** indica el comando que se ejecuta al iniciar el contenedor. (entrypoint admite parámetros). Admite ejecutar otro distinto en el momento del arranque.



Dockerfile Web Api Net Framework

```
FROM microsoft/aspnet:4.7.2-windowsservercore-1803
ARG source
WORKDIR /inetpub/wwwroot
COPY ${source:-obj/Docker/publish} .
```

Dockerfile Angular Nginx

```
FROM nginx
LABEL author="JMC"
COPY ./dist/testangular /usr/share/nginx/html
EXPOSE 80 443
CMD ["nginx", "-g", "daemon off;"]
```

Crear un fichero Dockerfile

```
FROM ubuntu:17.10
RUN apt-get update
RUN apt-get install -y nmap apache2 htop
RUN echo "test" > test.txt
RUN mkdir -p /demo1
EXPOSE 80 443
```

docker build -t apache2 . (genera una imagen local en base a una del hub)

docker run --name ubu3 -d -it apache2:latest /bin/bash (genera un nuevo contenedor)

docker exec -it ubu3 /bin/bash Luego "ll"

Crear un fichero Dockerfile - WINDOWS

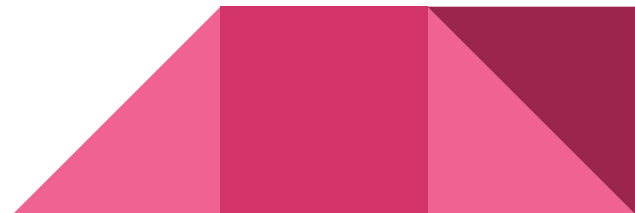
```
FROM microsoft/iis
```

```
RUN echo "Hello World - Dockerfile" > c:\inetput\wwwroot
```

```
# docker build -t myserver . (genera una imagen local en base a una del hub)
```

```
# docker run --name iisserver -d -p 9999:80 myserver (genera un nuevo contenedor)
```

```
# docker exec -it iisserver cmd se accede al explorador de archivos del contenedor
```



Demo.



Ejecutar contenedores.

Poner a funcionar un contenedor Linux.

- `docker run --name ubu -d -it ubuntu:17.10 /bin/bash`
- `docker images`
- `docker ps -a`
- `docker start/stop ubu`
- `docker exec -it ubu /bin/bash`
- `uname -a` (compartir kernel, ram y cpu con S.O. real)
- `apt update; apt install htop nmap; htop ; nmap 127.0.0.1`

https://dockertips.com/ubuntu_1804

https://dockertips.com/modos_run_docker



Poner a funcionar un contenedor Windows.

- `docker run --name iisserver -d -it nanoserver/iis powershell`
- `docker images`
- `docker ps -a`
- `docker start/stop iisserver`
- `docker exec -it iisserver cmd`



Comunicaciones con el contenedor.

- ping 172.17.0.1 y ping 172.17.0.2.
- apt install apache2; /etc/init.d/apache2 start
- nmap 172.17.0.2
- Acceder a la página <http://172.17.0.2>
- ls -ail /var/lib/docker/containers/
- **exit** sale del contenedor pero no lo detiene.





Almacenamiento.

Almacenamiento.

- Dos tipos de almacenamiento, interno y conectado al equipo real.
- `docker run --name ubu2 -v /tmp:/host -d -it ubuntu:17.10 /bin/bash`
- `docker exec -it ubu2 /bin/bash`
- El `/host` muestra el contenido de la carpeta “host” apuntando a “/tmp”
- Cada contenedor tiene una zona en el disco real.



Limpieza. `docker system prune -a`

```
docker rm ubu --force
```

```
docker rmi ubuntu:17.10 --force
```

```
du -h /var/lib/docker
```

```
rm -r /var/lib/docker
```

```
/etc/init.d/docker restart
```

```
docker ps -a
```

```
docker images
```

```
# docker system prune -a
```

```
WARNING! This will remove:
```

- all stopped containers
- all networks not used by at least one container
- all images without at least one container associated to them
- all build cache

```
Are you sure you want to continue? [y/N] y
```

```
Deleted Containers:
```

```
1450281e0150a64a07edd6ff27a96cb488556e8c618afb7acf768b6b3c954a27
```

```
Deleted Images:
```

```
untagged: ubuntu:latest
```

```
untagged:
```

```
ubuntu@sha256:945039273a7b927869a07b375dc3148de16865de44dec839867297  
7e050a072e
```

```
untagged: my-apache2:latest
```

```
deleted:
```

```
sha256:f007c13967e9ae098a46393cc90b1a598f02dd9028e3c4f1f834a3f4953eb04f
```

```
Total reclaimed space: 305.2MB
```



Preguntas.

Gracias por participar.
dockertipshelp@gmail.com

Próximo Meetup, II) Docker-Compose vs Docker-Swarm

