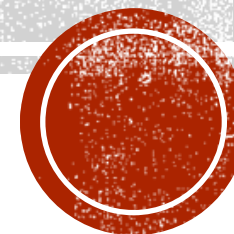


AMBIENTES DE DESARROLLO

GESTIÓN DE SERVIDORES / SERVICIOS



SITUACION ACTUAL

- “La complejidad cada vez mayor de las aplicaciones y la necesidad de acelerar el desarrollo están ejerciendo aún más presión sobre la infraestructura, procesos y equipos de TI.
- Los líderes en I&O (Infrastucture & Operation) deben mejorar las estrategias de implementación de servidores o crearán infraestructuras incapaces de dar soporte a la siguiente generación de aplicaciones.



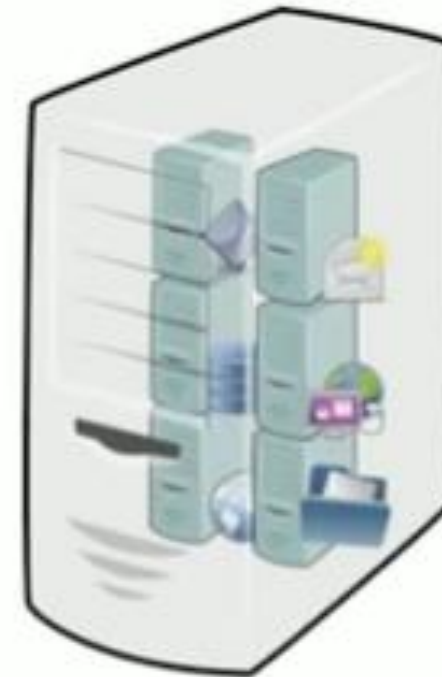
MAQUINAS VIRTUALES (VMS)

CARACTERISTICAS NEGATIVAS

- De volumen considerado, en el orden de los GB.
- Las VMs en el mismo host pueden repetirse en lo que contienen (ej. núcleo del S.O, drivers, etc).
- Problemas de update, parches, etc.

CARACTERISTICAS POSITIVAS

Las MV tienen mejor aislamiento (Intel Vtd/Vtx) otras alternativas.



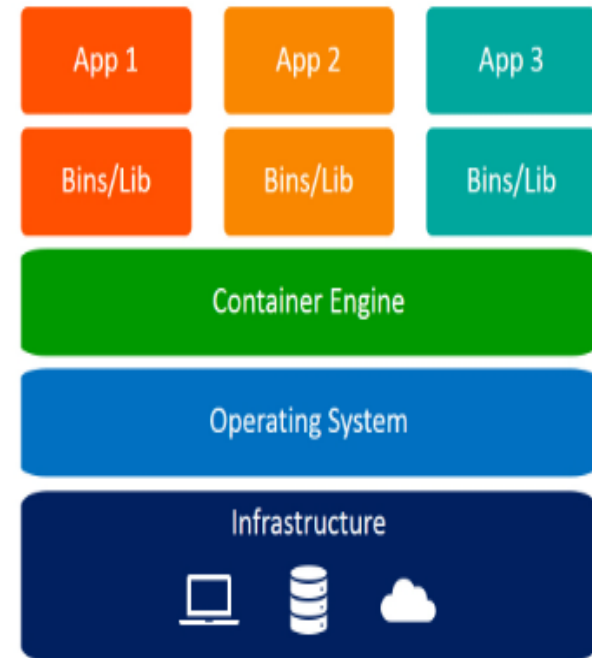
Virtual Machines



CONTENEDORES

CARACTERISTICAS

- **Autosuficiencia.** Tienen instaladas solo las dependencias/librerías que son necesarias para dar un servicio.
- **Portabilidad.** Funcionan igual en cualquier lado.
- **Ligereza.** Solo se ejecutan procesos, no un S.O. completo. Lo que pasa en el container queda en el container.
- **Eficientes.** Ya que comparten archivos inmutables con otros contenedores.
- **De volumen versátil** de los ordenes de los MB, contienen solo lo que las diferencian del S.O. en el que se ejecutan.



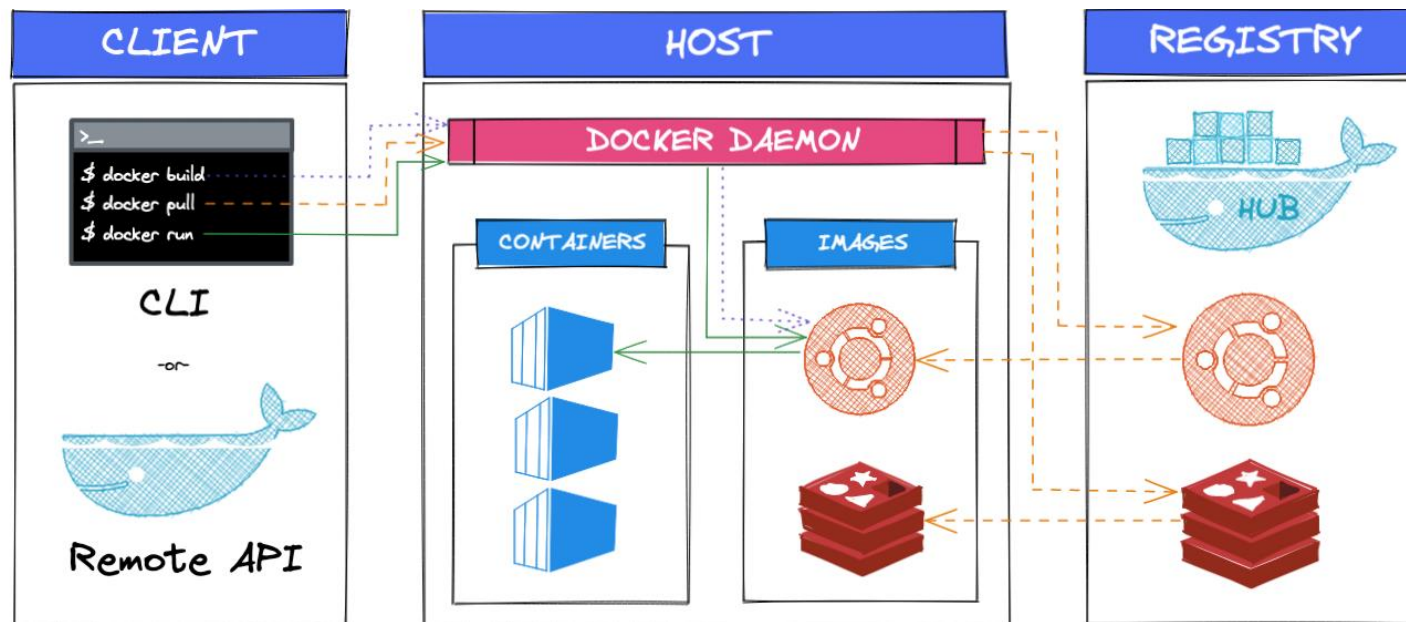
Containers



DOCKER - DEFINICION

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de “contenedores”. Este contenedor empaqueta todo lo necesario para que uno o más procesos (servicios o aplicaciones) funcionen: código, herramientas del sistema, bibliotecas del sistema, dependencias, etc.

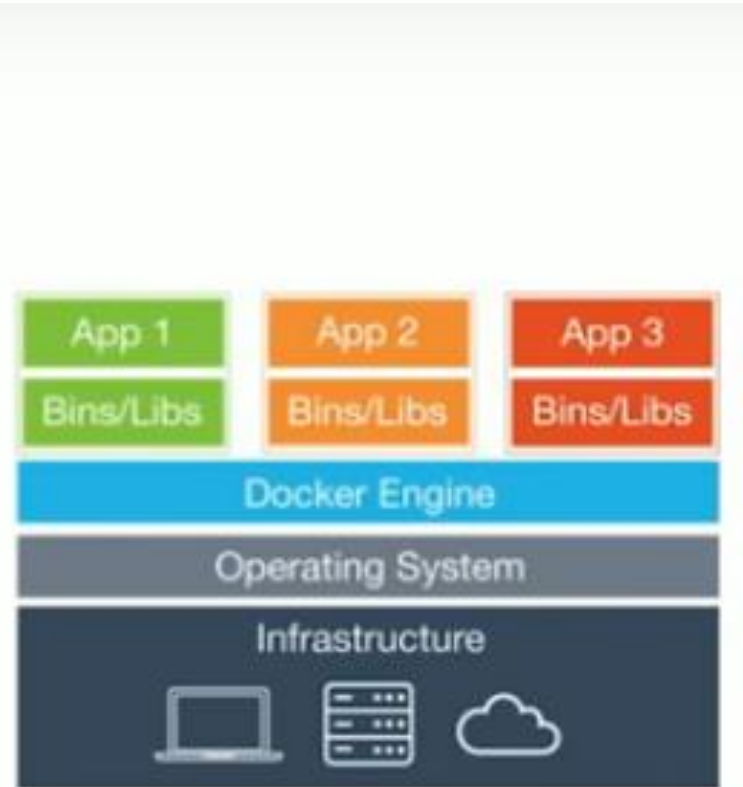
Usar contenedores Docker permite a desarrolladores y administradores de sistemas probar aplicaciones o servicios en un entorno seguro e igual al de producción, reduciendo los tiempos de pruebas y adaptaciones entre los entornos de prueba y producción.



VMS, CONTAINER - ARQUITECTURA



VMs



Contenedores



CONTENEDORES VS VMS, CUAL ELEGIR?

Existen escenarios idóneos para cada aproximación. – Puede que la combinación de ambos sea la opción más propicia.

- Docker Container es idóneo porque
 - Promueve la portabilidad entre diferentes cloud providers.
 - Es una plataforma abierta para desarrolladores. Permite aislar dependencias de las aplicaciones aislándolas en contenedores.(ej. ..)
 - Los contenedores son más escalables y seguros que otras estrategias.
- VM es idóneo donde
 - Existen escenarios de distribución de cargas más apropiado. Ej. cluster de BDs.
 - La gestión de la capacidad, seguridad y rendimiento pueden requerir herramientas aún no disponibles para entornos de contenedores.
 - Sólo puede usarse de forma nativa en entornos Unix con Kernel igual o superior a 3.8. Requerimiento Arquitectura de 64 bits.



REGISTRO — DOCKER HUB

Registro:

- Registro público (Docker Hub): mas de 10.000 imágenes disponibles. Es libre.
<https://hub.docker.com/>
- Registro privado: Sólo accesible bajo autorización. Es pago.

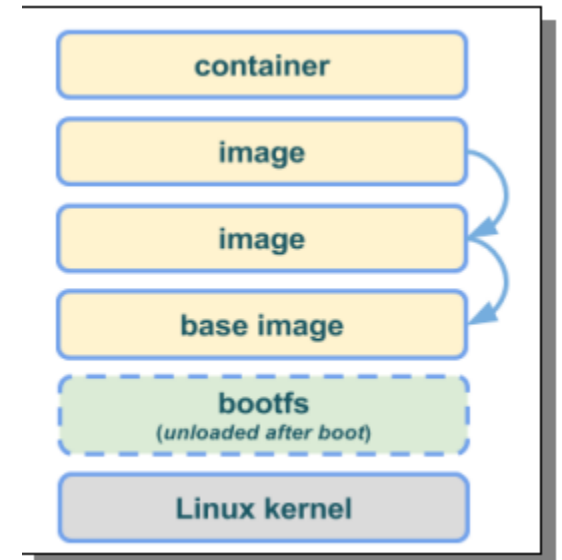
Registro local

- Este registro se instala por medio de un contenedor Docker para centralizar las imágenes de los contenedores.



IMAGENES

- Cada imagen (plantilla), también conocida como repositorio, es una sucesión de capas, es utilizada para crear nuevos contenedores. Las imágenes nunca cambian (solo lectura).
- Con el contenedor corriendo, cada vez que realizamos un cambio en el contenedor Docker añade una capa encima de la anterior con los cambios
- Hay muchas imágenes públicas con elementos básicos como Java, Ubuntu, Apache...etc, disponibles para descargar.
- Cuando lo que necesitamos es crear nuevas imágenes, partimos de una imagen padre básica, como por ejemplo una imagen de Debian a la que le vamos añadiendo programas.



IMÁGENES — EJ. COMANDOS DOCKER

- > `docker search ubuntu`

Busca imágenes en el docker hub que se correspondan con el texto ubuntu.

- > `docker pull ubuntu:14.04`

Descarga una imagen ubuntu (del repositorio oficial), la versión no es obligatoria por defecto tomará la última versión. Si nos fijamos en la descarga vemos 5 líneas que ponen “pull complete”. Esto es que la imagen está formada por 5 capas. Estas capas pueden ser reutilizadas por otras imágenes, que evitan así tener que volver a descargarlas, Ej. si descargáramos otra imagen (otra versión) de Ubuntu.

```
ubuntu@docker:~$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
5ba4f30e5bea: Pull complete
9d7d19c9dc56: Pull complete
ac6ad7efd0f9: Pull complete
e7491a747824: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:46fb5d001b88ad904c5c732b086b596b92cfb4a4840a3abd0e35dbb6870585e4
Status: Downloaded newer image for ubuntu:latest
ubuntu@docker:~$
```



IMÁGENES — EJ. COMANDOS DOCKER

- > `docker pull sgonzalez/centosopen:centos5`
Descarga una imagen centosopen (de un repositorio particular) especifica la versión.
- > `docker images`
Lista las imágenes que se encuentran descargadas en el host.
- > `docker rmi image`
Borra una imagen que hemos descargado y disponemos localmente.



CONTENEDORES – EJECUCIÓN CON RUN

El comando run primero crea una capa del contenedor sobre la que se puede escribir y a continuación ejecuta el comando especificado. Ej. Se correrá el comando echo hello world

```
> docker run ubuntu echo hello world
```

El contenedor que se creará a partir de una **imagen**. Si la imagen no esta disponible localmente se procederá primeramente a descargarla y luego se creará el contenedor.

Dicho contenedor sólo se ejecuta durante el tiempo que dura el **comando** que especifiquemos, funciona como un proceso.

```
> docker run -it --name test01 ubuntu /bin/bash
```



CONTENEDORES – EJECUCIÓN CON RUN

El comando con la opción `-d` corre un servicio que se mantendrá en ejecución y no parará el contenedor. En este caso estamos descargando un servidor apache y mapeando los puertos internos a los externos (80:80)

```
> docker run -d --name apache-server -p 80:80 httpd
```



CONTENEDORES - COMANDOS

<code>docker ps [-a]</code>	<code>docker inspect <ID></code>
<code>docker attach <ID></code>	<code>docker stop <ID></code>
<code>docker start <ID></code>	<code>docker rm [-f] <ID></code>
<code>docker logs [-f] <ID></code>	<code>docker create <image></code>
<code>docker top <ID></code>	<code>docker build [-f dockerfile] <dir></code>
<code>docker exec <ID> <cmd></code>	<code>docker load / save</code>



CONTENEDORES — COMANDOS ...

Podemos listar los contenedores que se están ejecutando actualmente o con `-a` se listarán todos los creados.

> `docker ps` //lista los contenedores que están en ejecución

> `docker ps -a` //lista todos los contenedores

```
$ docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                    NAMES
e194cb5836ec   httpd     "httpd-foreground"     44 minutes ago Up 13 minutes  0.0.0.0:80->80/tcp      apacheserver
```

> `docker stop apacheserver` //detiene un contenedor en ejecución

> `docker start apacheserver` //arranca un contenedor que esta parado

> `docker exec -it apacheserver /bin/bash` //ejecuta un comando en un contenedor

> `docker rm apacheserver` //elimina un contenedor que no esta en ejecución



DOCKER HUB — SUBIENDO IMAGENES ...

Subir una imagen a Docker hub

```
C:\Users\alfre>docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
nano/ubuntu     v1          00e083e57a52  51 minutes ago 115MB
ubuntu          latest      df5de72bdb3b  10 days ago   77.8MB
```

> docker tag local-repository:localversion remote-repository:remote-version

ej. docker tag nano/ubuntu:v1 aespi/ubuntutest:v01

```
C:\Users\alfre>docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
aespi/ubuntutest v01         00e083e57a52  49 minutes ago 115MB
nano/ubuntu     v1          00e083e57a52  49 minutes ago 115MB
ubuntu          latest      df5de72bdb3b  10 days ago   77.8MB
```

> docker login -u aespi -p xxyyyzz

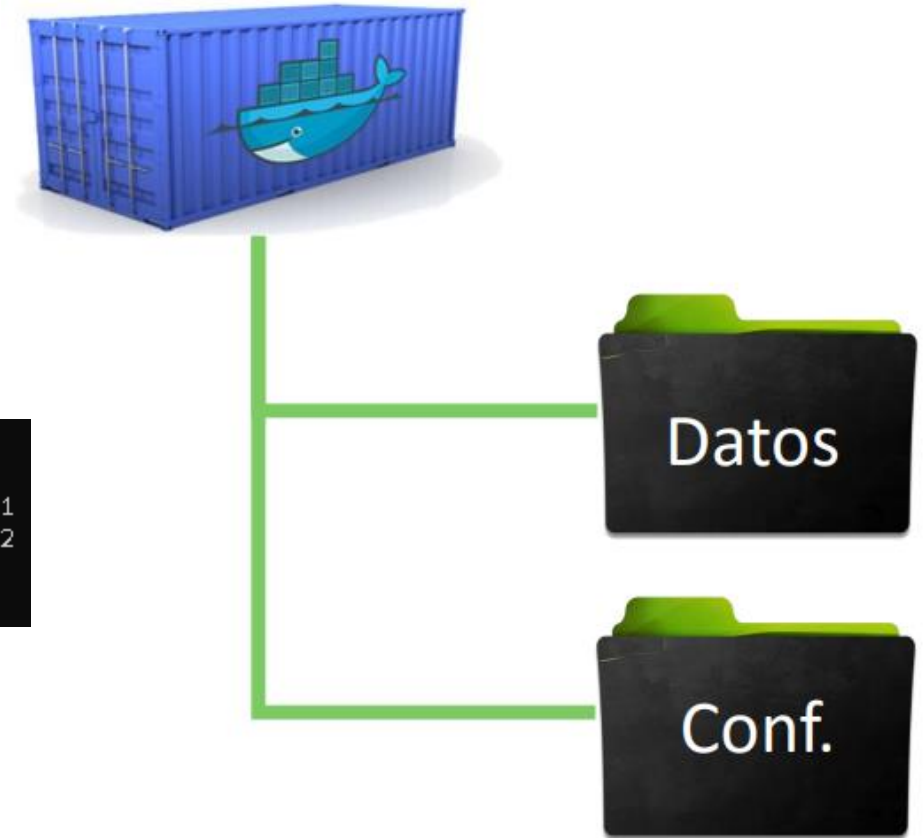
> docker push aespi/ubuntutest:v01



VOLUMENES

- Discos o directorios externos que podemos montar en el contenedor.
- Recursos externos (alojados en el host) que sobreviven al contenedor
- Usos: Configuración / Datos / Recursos

```
C:\Users\Users>docker volume ls
DRIVER      VOLUME NAME
local       0c9efafd0def1aa383f0026a0db8639f63136fd83b508d925b15753424b487a1
local       1b3031a720691694164f59293dfdff9e0f39f36468ab7bc465e05baab32ac372
local       jiraVolume
local       sitioangularapp
```



DOKERFILE

- Un Dockerfile es un archivo de texto (ej. de nombre Dockerfile) que incluye una serie de instrucciones que se necesitan ejecutar de manera consecutiva para cumplir con los procesos necesarios para la creación de una nueva imagen.

```
FROM ubuntu
RUN apt update
RUN apt install -y apache2
RUN apt install -y apache2-utils
RUN apt clean
EXPOSE 8082
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

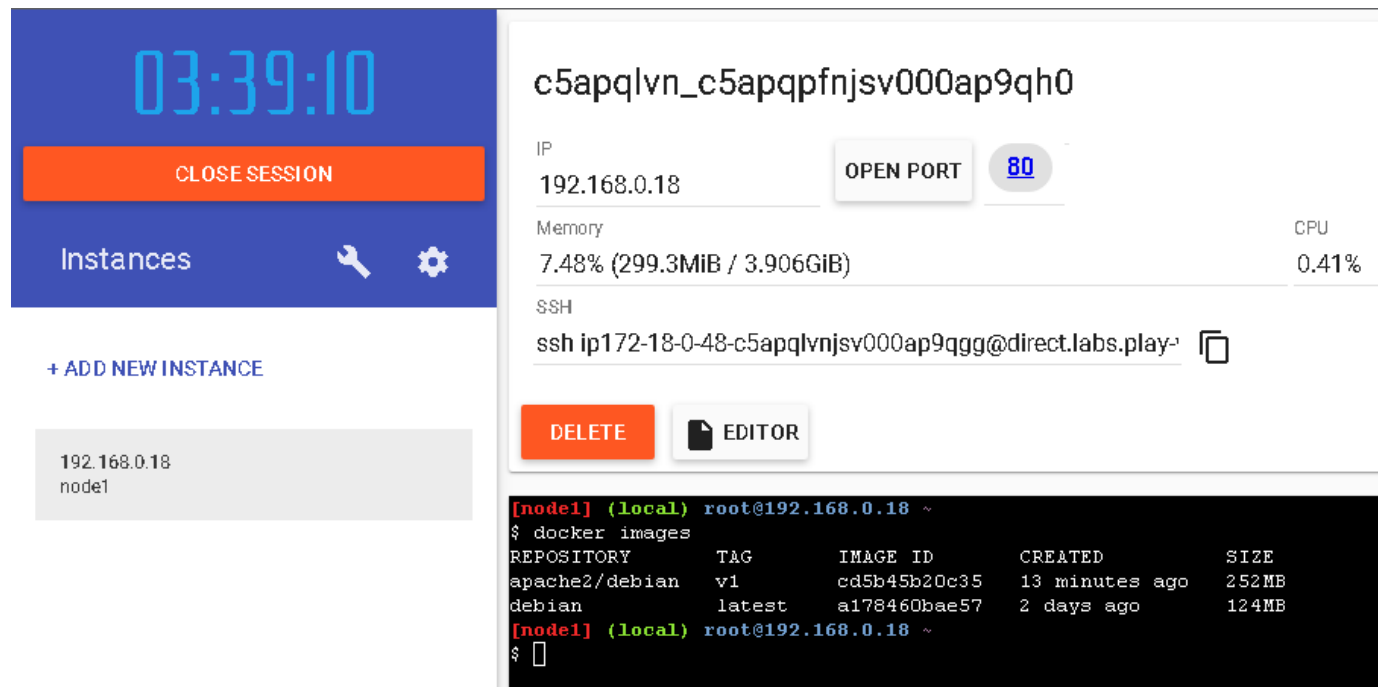
- Las imágenes en dockerfile se crean a partir de un comando en específico denominado docker build, que se encargará seguir las instrucciones que el usuario haya indicado en el archivo de texto.

```
C:\Users\Users>docker build -f Dockerfile -t miapache:1.0 .
```



PLAY DOCKER

- En caso de no poder instalar Docker en Windows o Linux podemos testear en línea los comandos Docker en una sesión gratuita de 4 hs. Este es un servicio oficial del sitio de docker. <https://labs.play-with-docker.com/>



The screenshot displays the Play with Docker interface. On the left, a blue sidebar contains a timer showing 03:39:10, a 'CLOSE SESSION' button, and an 'Instances' section with a '+ ADD NEW INSTANCE' button and a list of instances including '192.168.0.18 node1'. The main area shows details for instance 'c5apqlvn_c5apqpfjnjsv000ap9qh0', including its IP (192.168.0.18), memory usage (7.48%), CPU usage (0.41%), and an SSH command. Below this are 'DELETE' and 'EDITOR' buttons. At the bottom, a terminal window shows the command '\$ docker images' and its output:

```
[node1] (local) root@192.168.0.18 ~
$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
apache2/debian v1        cd5b45b20c35  13 minutes ago 252MB
debian        latest   a178460bae57  2 days ago    124MB
[node1] (local) root@192.168.0.18 ~
$
```

