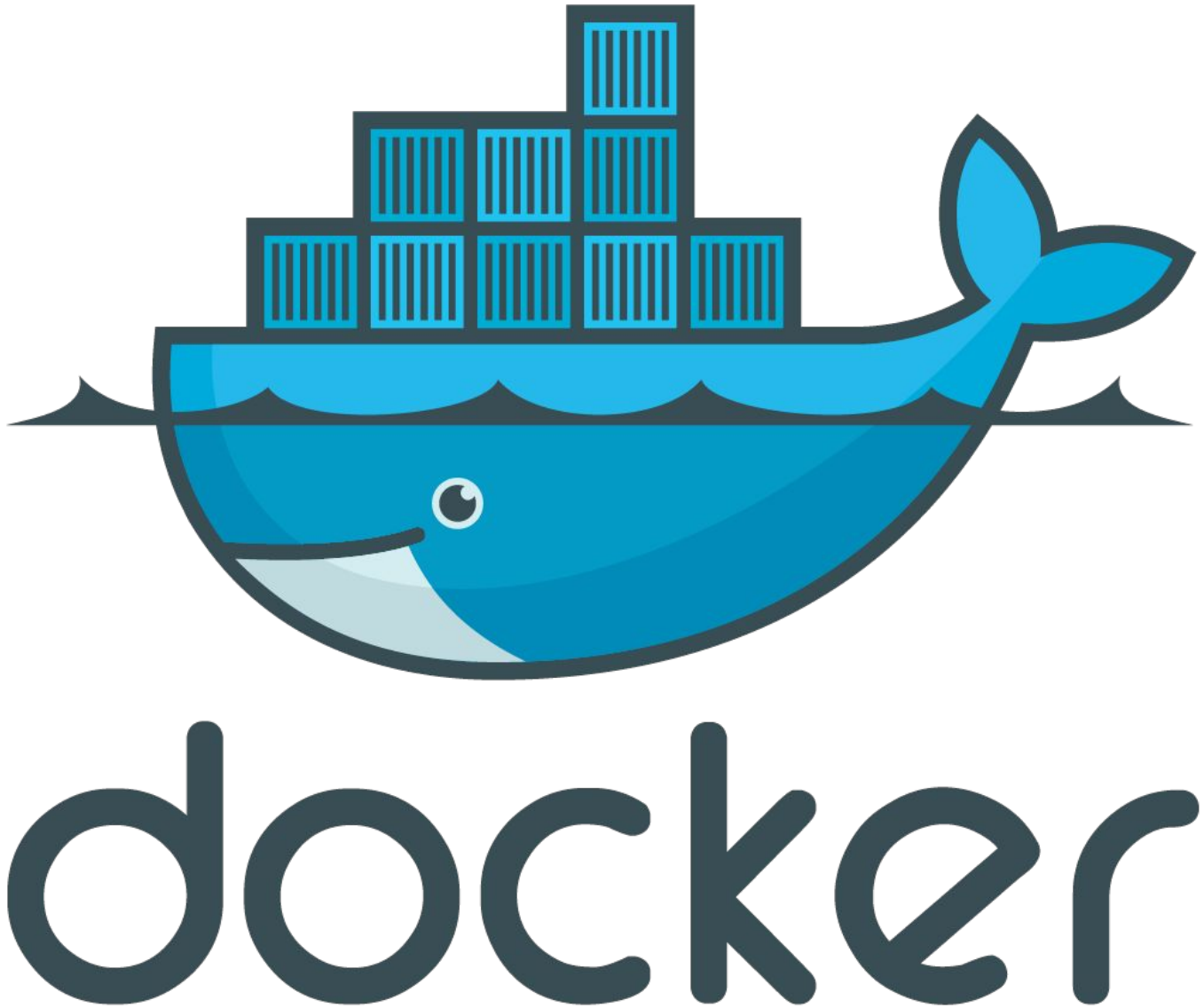
ARQUITECTURA Y SISTEMAS OPERATIVOS

Docker


docker

¿ QUE ES DOCKER ?

- Docker es una herramienta tanto para el sysadmin como developer.
- Ayuda a automatizar el despliegue de aplicaciones dentro de de contenedores de software, proporcinando una capa de adicional de abstracción eliminando el overhead que provoca corer un Sistema operative virtual para cada una sola aplicación.
- Además de la propia aplicación que gestiona los contenedores (a los que llamamos docker engine o docker), existe un repositorio de imágenes creadas por la comunidad, llamada docker hub.
- Tanto los developers como los sysadmin pueden compartir entornos de trabajo (imágenes).

¿ QUE ES DOCKER ?

- Corre en equipos de 64 bits.
- Gran parte de éxito de docker se basa en su fácil portabilidad y su ligereza.
- Soporta los sistemas operativos Windows, Mac y obviamente GNU/Linux.
- Docker es tanto cliente como servidor.

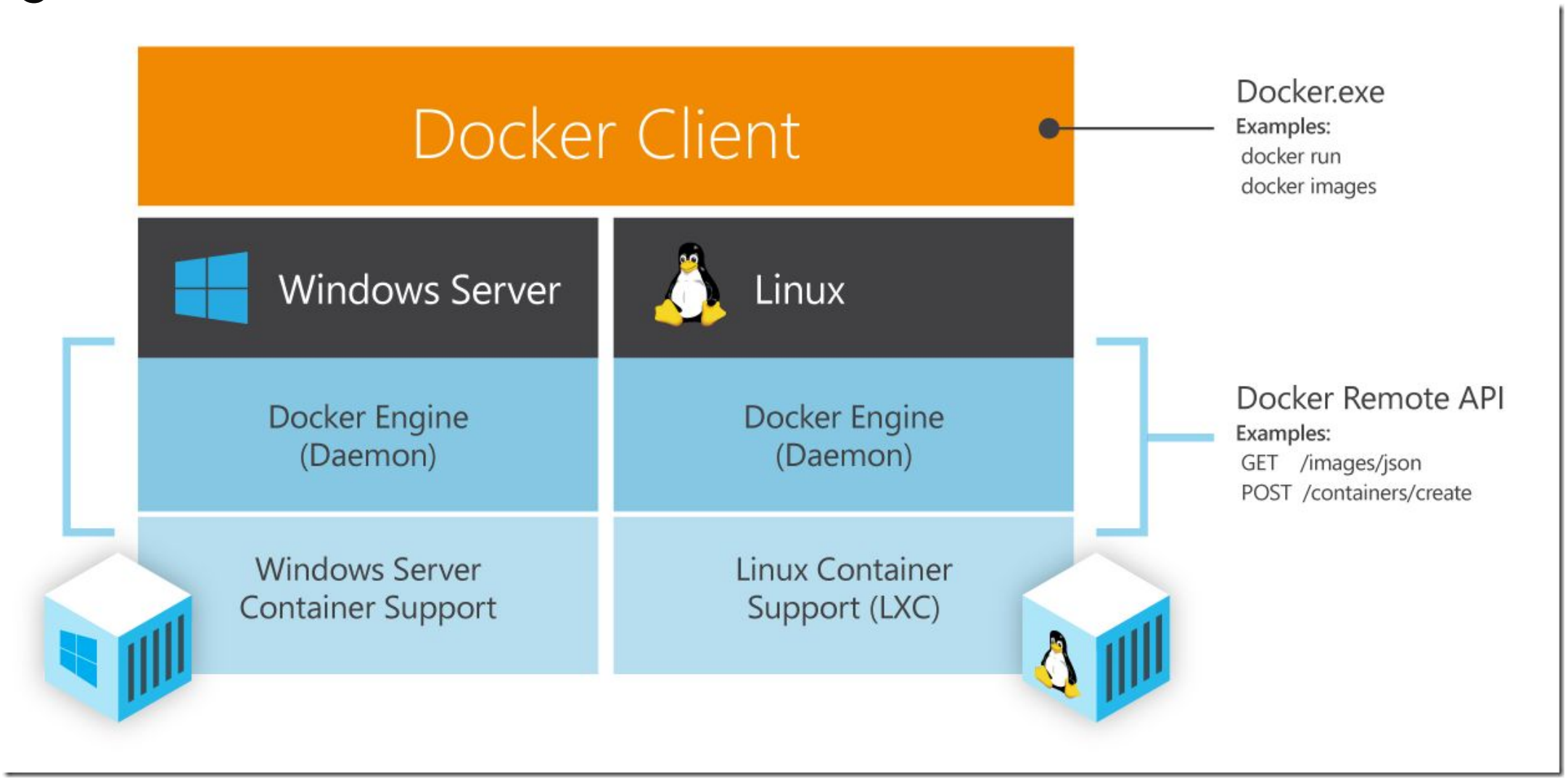
¿ QUE NO ES DOCKER ?

- Un gestor de entornos de desarrollo virtuales (vagrant).
- Un software de virtualización (hypervisor).

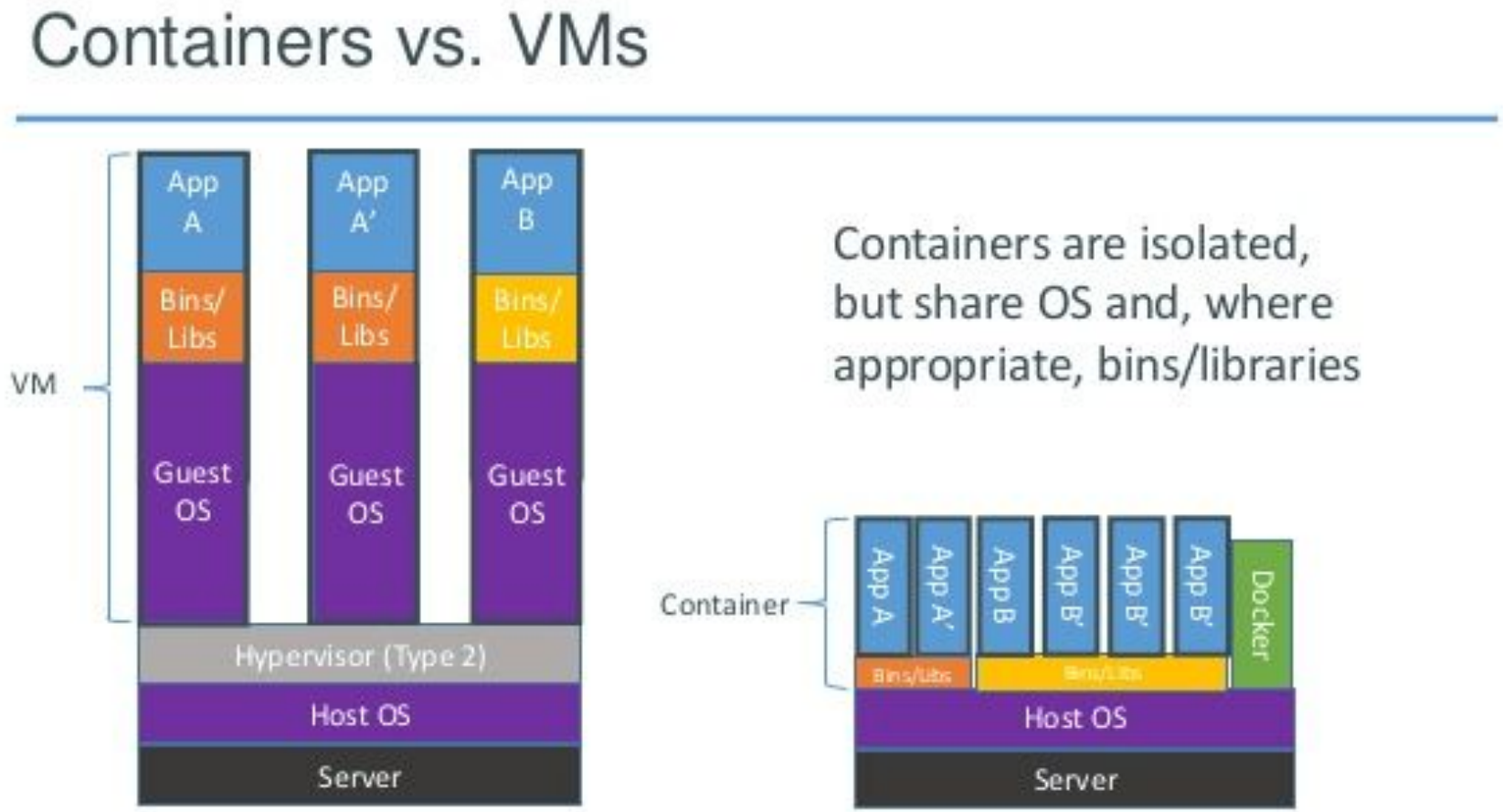
? (kvm, xen, vmware, virtualbox, etc).

- Un gestor de configuración (puppet/chef).
- Un simple contenedor de software (lxc).
- Aunque se parece como servidor.

¿ QUE ES DOCKER ?



¿ QUE ES DOCKER ?



¿ QUE ES DOCKER ?

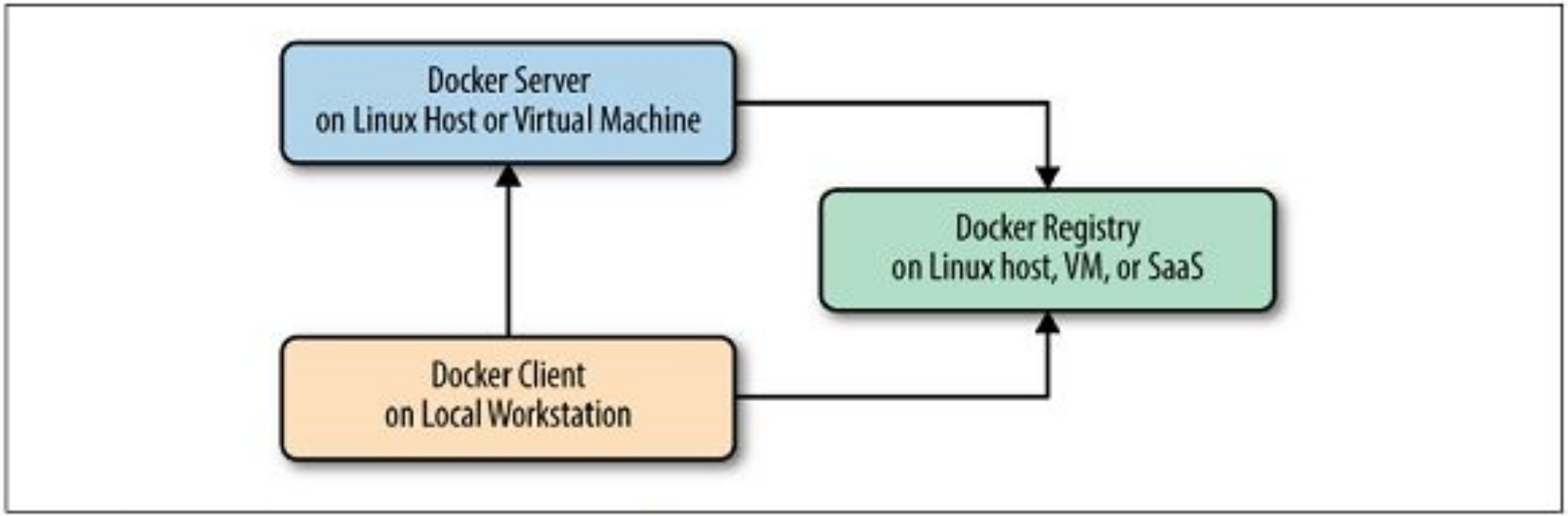
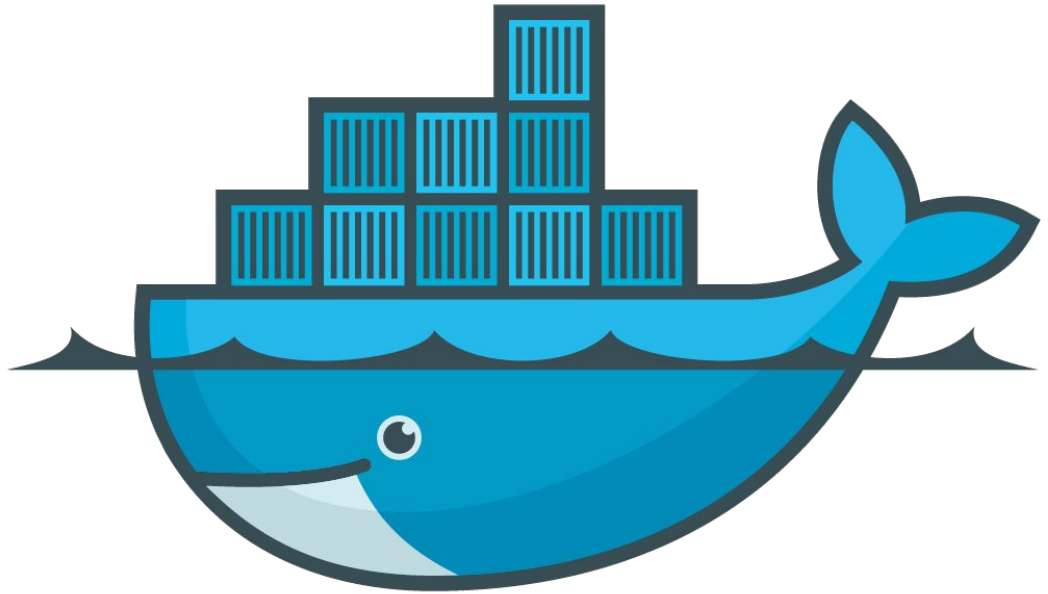
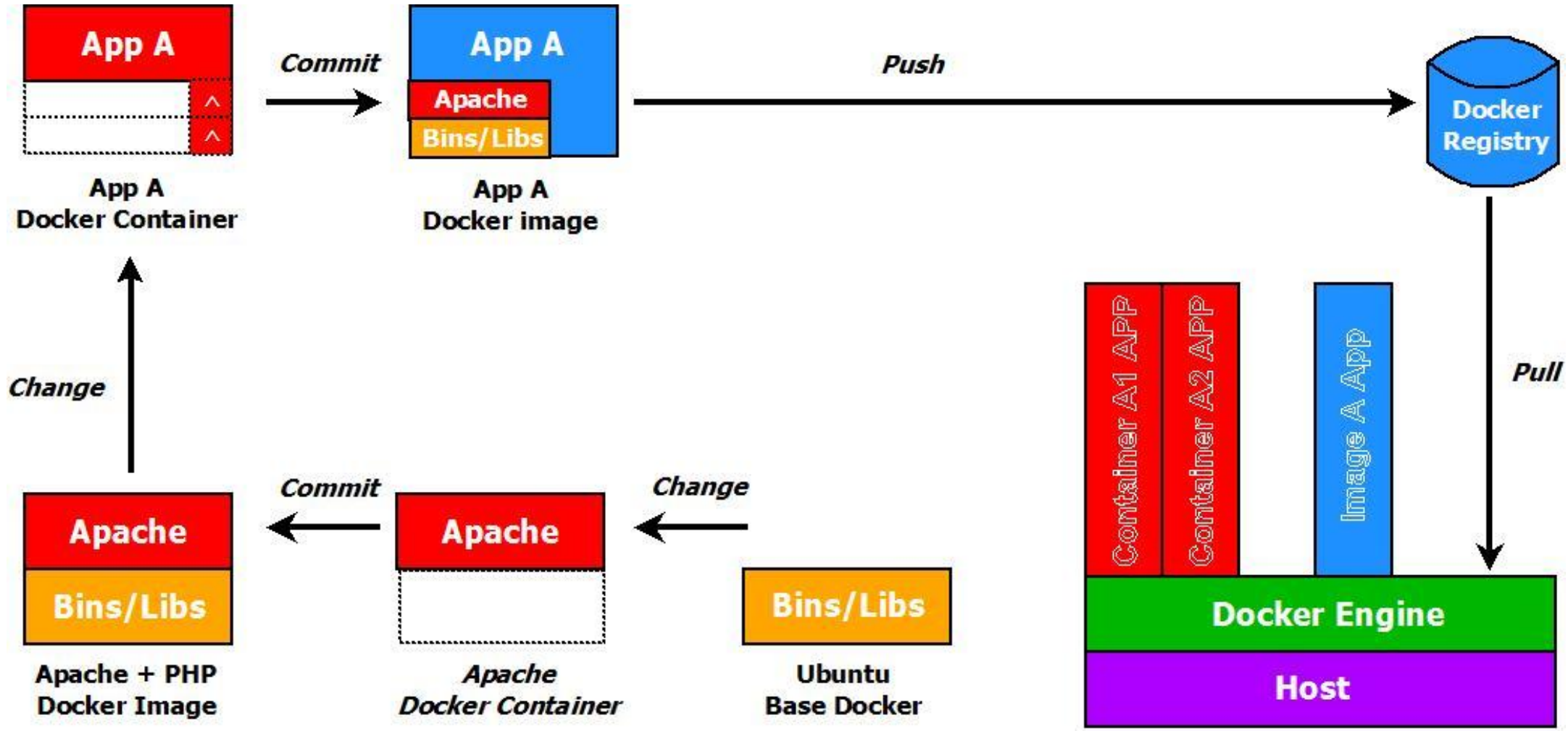


Figure 2-3. Docker client/server model



CICLO DE VIDA

CICLO DE VIDA

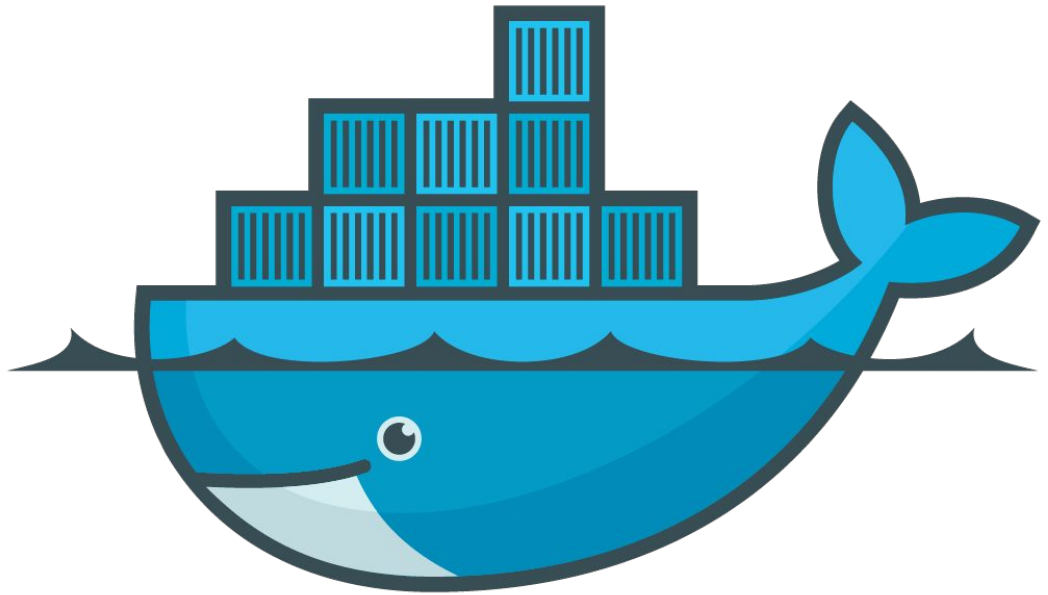


COMANDOS DE DOCKER CILO DE VIDA

- dockere pull
- docker images
- docker run
- docker stop & start
- docker ps
- docker attach
- docker logs / docker cp
- docker top

COMANDOS DE DOCKER CILO DE VIDA

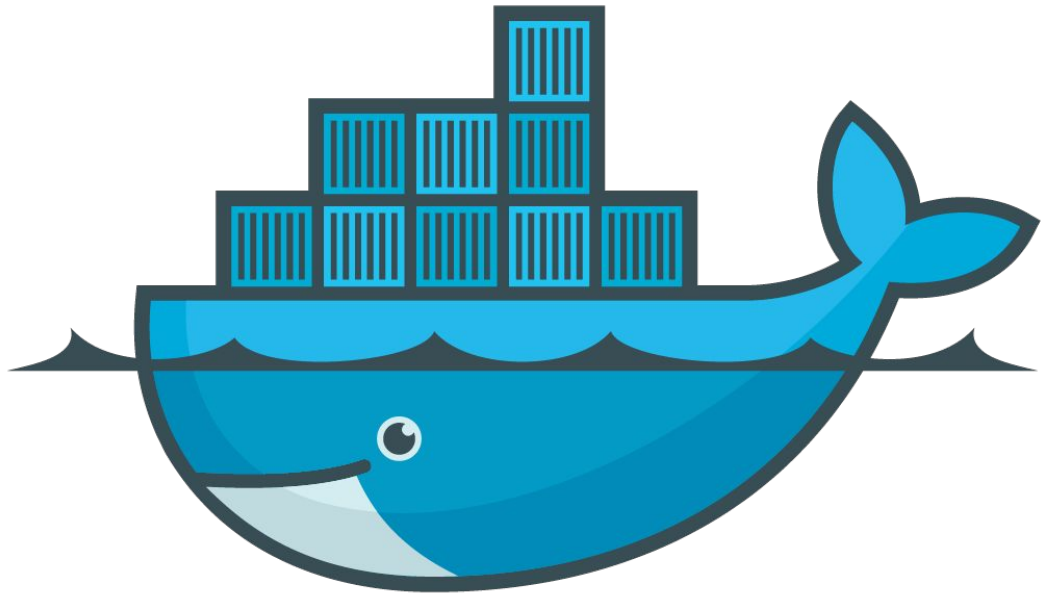
- dockere export / import
- docker tag
- docker commit
- docker push



WEB

ARQUITECTURA DE DOCKER

- WEB
  - https://www.docker.com
- HUB
  - https://hub.docker.com
- DOCUMENTACION
  - https://docs.docker.com
- API
  - https://docs.docker.com/engine/api/



docker

INSTALACION

INSTALACION

- Ubuntu
  - apt-get install docker.io
- Centos/OS
  - yum –y install docker docker-commn container-selinux
- Debian
  - apt-get install docker-engine

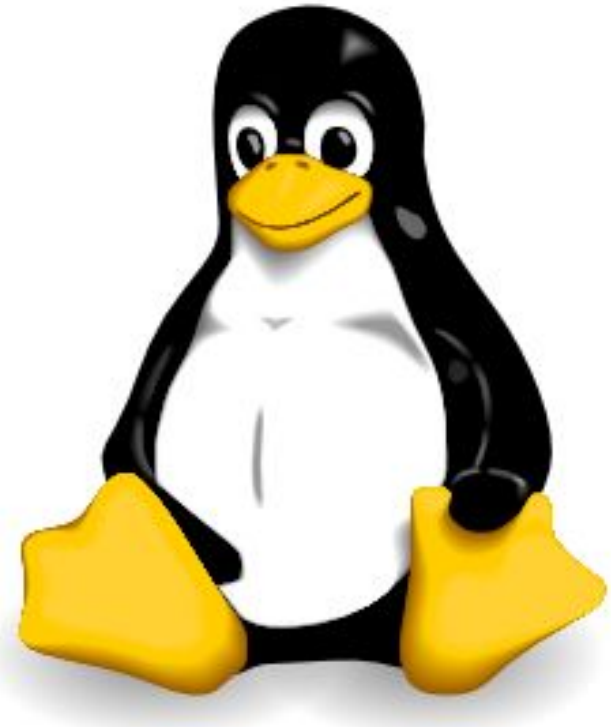
https://docs.docker.com/engine/installation/linux/

INSTALACION

```
localhost:~# ps aux | grep docker
 2106 root      0:21 /usr/bin/dockerd -p /run/docker.pid
 2237 root      0:36 containerd --config /var/run/docker/containerd/containerd.toml --log-level info
 2701 root      0:00 grep docker
```

INSTALACION

usermd –G docker –a USUARIO



GNU/Linux

COMANDOS

ifconfig

```
localhost:~# ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:05:C7:D4:60
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 08:00:27:9D:FD:59
          inet addr:192.168.1.109  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe9d:fd59/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10072 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1517 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3094468 (2.9 MiB)  TX bytes:164694 (160.8 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

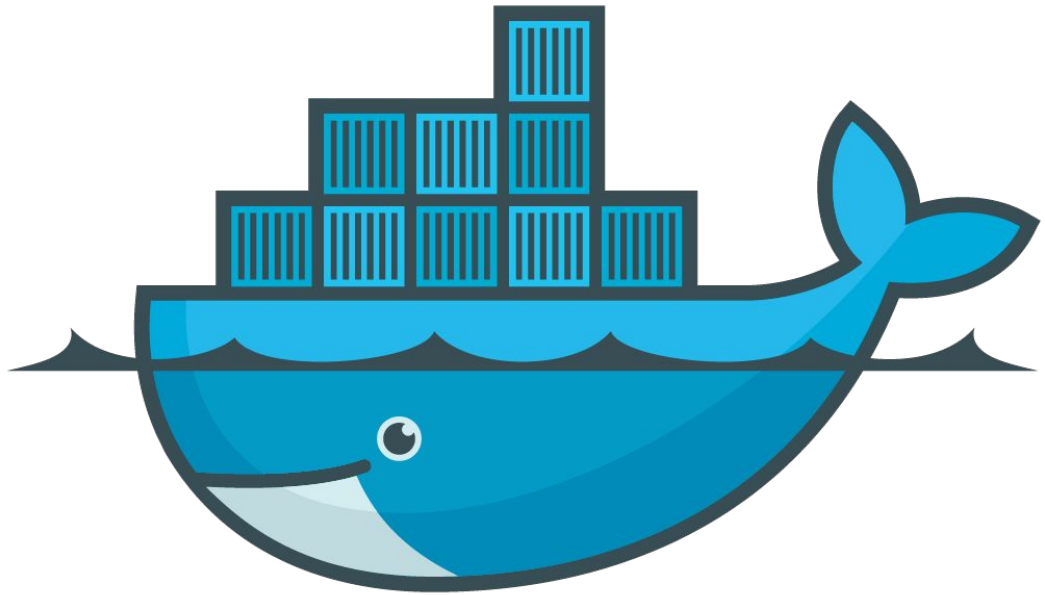
route

```
localhost:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.1.1     0.0.0.0         UG    202    0        0 eth0
172.17.0.0      *               255.255.0.0     U     0      0        0 docker0
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
```

COMANDO

iptables –L -n

```
localhost:~# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination
DOCKER-USER  all  --  0.0.0.0/0            0.0.0.0/0
DOCKER-ISOLATION-STAGE-1  all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           ctstate RELATED,ESTABLISHED
DOCKER     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain DOCKER (1 references)
target     prot opt source               destination

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
target     prot opt source               destination
DOCKER-ISOLATION-STAGE-2  all  --  0.0.0.0/0            0.0.0.0/0
RETURN     all  --  0.0.0.0/0            0.0.0.0/0

Chain DOCKER-ISOLATION-STAGE-2 (1 references)
target     prot opt source               destination
DROP       all  --  0.0.0.0/0            0.0.0.0/0
RETURN     all  --  0.0.0.0/0            0.0.0.0/0

Chain DOCKER-USER (1 references)
target     prot opt source               destination
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
```



docker

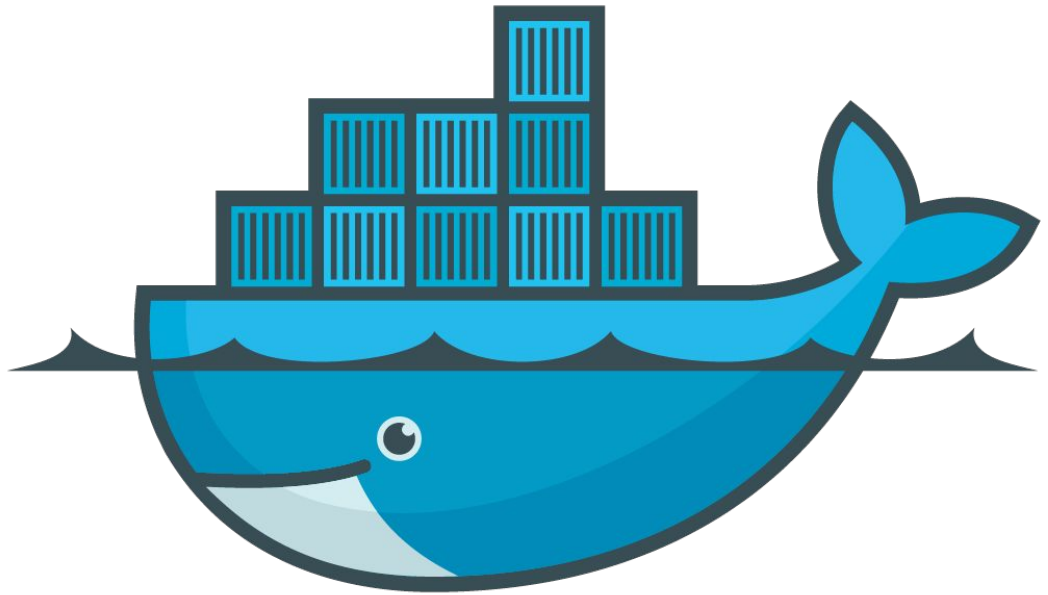
COMANDOS

COMANDOS

docker help

docker versión

docker info

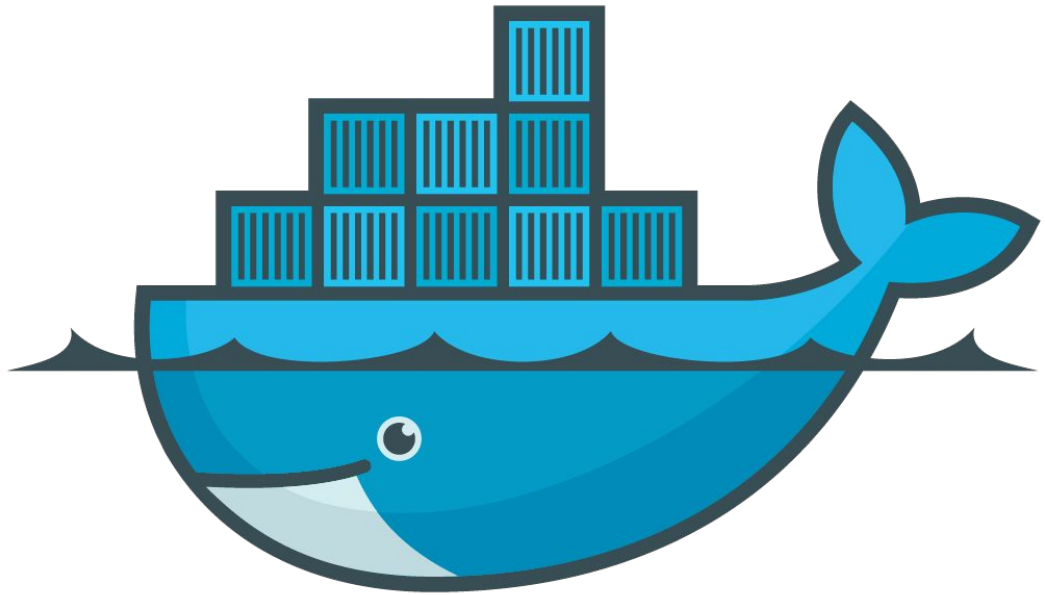


docker

IMAGENES

COMANDOS MANEJO DE IMAGENES

```
# docker search ubuntu

# docker pull ubuntu

# docker images

# docker rmi ubuntu:latest
```



docker

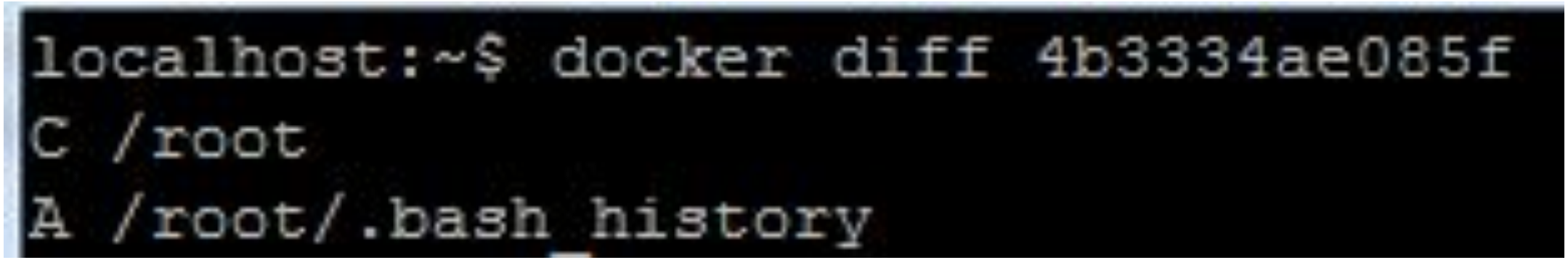
CONTENEDORES

COMANDOS CONTENEDORES

```
# docker run –ti ubuntu /bin/bash
# docker ps
# docker cp MI-ARCHIVO 36a9bb57726a:/DIRECTORIO
# docker ps -a
# docker logs 36a9bb57726a
# docker attach 36a9bb57726a
# docker rmi ubuntu:latest
```

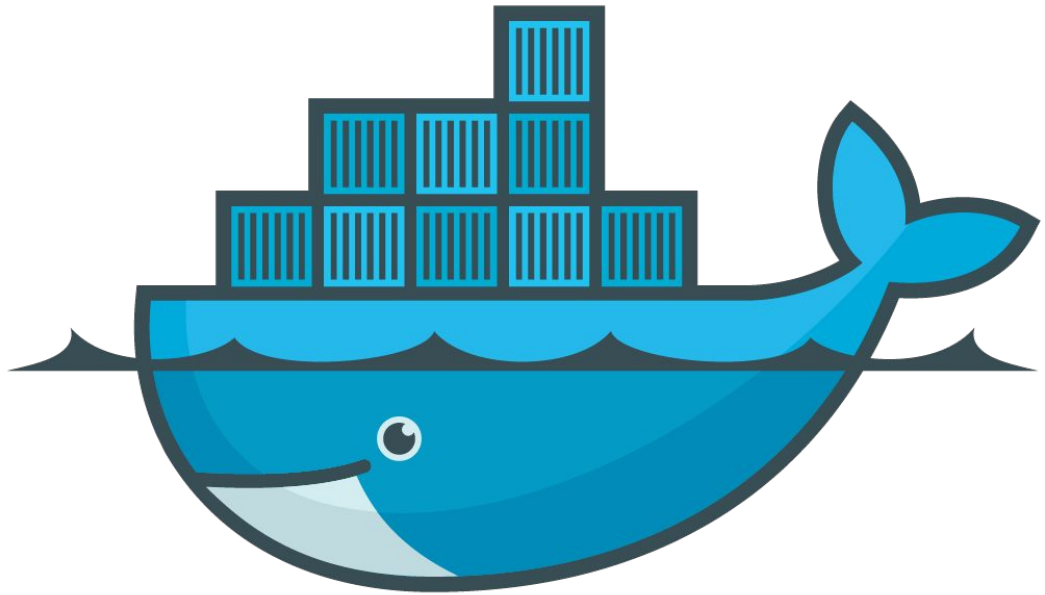
COMANDOS CONTENEDORES

docker diff 4b3334ae085f



COMANDOS CONTENEDORES

```
# docker stop 4b3334ae085f

# docker start 4b3334ae085f

# docker rm $(docker ps –a –q)

# docker rmi ubuntu
```



docker

Networking

NETWORKING

- **bridge:** La red standard que usarán todos lo contenedores, por defecto se hace un puente para pasar todo el tráfico.

- **host:** El contenedor usará el mismo IP del servidor real que tengamos.

- **overlay:** redes virtuales.

- **macvlan:** asignar una mac a un contenedor, tener una ip.

- **none:** Se utiliza para indicar que un contenedor no tiene asignada una red, desactiva la networking, no usar la red en un container.

NETWORKING

docker inspect 4314eb49079d

```
...
 "NetworkSettings": {
        "Bridge": "",
        "SandboxID": "5971a4c426a35c2532586741af36e016b9e2e1fa9625f63e691d10691baae677",
        "HairpinMode": false,
        "LinkLocalIPv6Address": "",
        "LinkLocalIPv6PrefixLen": 0,
        "Ports": {
            "80/tcp": [
                {
                    "HostIp": "0.0.0.0",
                    "HostPort": "8081"
                }
            ]
        },
..
```

NETWORKING

```
# docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
c109b8931626	bridge	bridge	local
27b615e6979f	docker_default	bridge	local
85c4498884dc	host	host	local
e8f05f120cc0	none	null	local
79a6f1e28276	opencti-docker_default	bridge	local

NETWORKING

docker network inspect bridge


```
[
    {
        "Name": "bridge",
        "Id": "c109b89316266fcf0afb99f292fe3b559263d4fddba0ef2565700964a01098a0",
        "Created": "2020-04-24T10:20:02.299959845-03:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
```

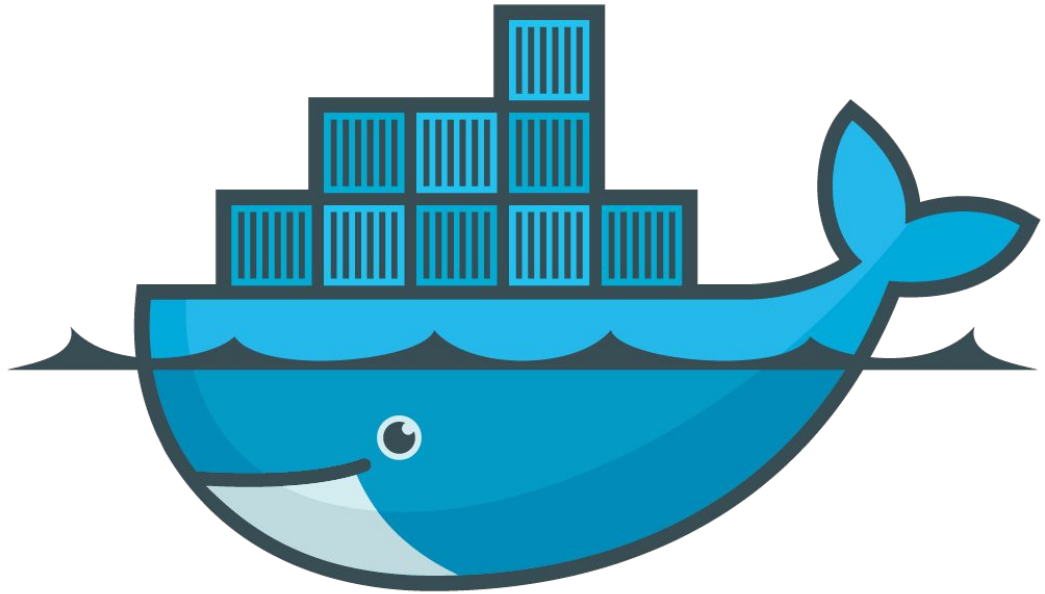
NETWORKING


```
 "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

NETWORKING

docker network create test

NETWORK ID	NAME	DRIVER	SCOPE
c109b8931626	bridge	bridge	local
27b615e6979f	docker_default	bridge	local
85c4498884dc	host	host	local
e8f05f120cc0	none	null	local
79a6f1e28276	opencti-docker_default	bridge	local



docker

DockerFile

DOCKERFILE

```
# Dockerfile



FROM ubuntu

RUN apt-get install
apache2

COPY code/ /var/www

CMD [ “apache2” ]
```

DOCKERFILE

```
localhost:~# cat dockerfile
FROM alpine:latest
MAINTAINER "Marcos Pablo Russo version: 0.1"
COPY repositories /etc/apk/repositories

RUN apk update && apk upgrade
RUN apk add alpine-base bash mc vim wget links \
            apache2 php7 php7-apache2 php7-pdo_mysql \
            php7-gd php7-session php7-mbstring php7-json \
            php7-zlib php7-xml php7-mcrypt php7-openssl

RUN rm -rf /var/cache/apk/*
RUN mkdir -p /run/apache2

EXPOSE 80
VOLUME /var/www/localhost/htdocs/

CMD ["httpd", "-D", "FOREGROUND"]
```

DOCKERFILE

```
# docker build -t marcospr1974:alpine-linux .
```

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

DOCKERFILE

```
localhost:~# docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
marcospr1974                alpine-apache       3a3b7fb35820        2 minutes ago       99.3MB
```

docker run –d --hostname web01 --name web01 –p 8080:80 marcospr1974:alpine-apache

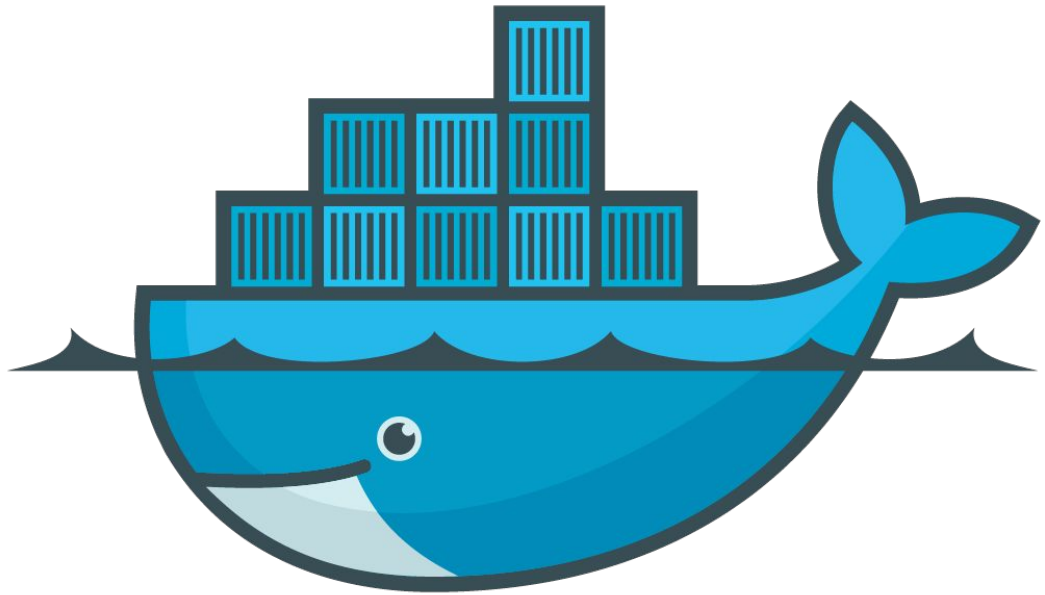
docker top web01

docker export web01 –o backup_web01.tar

DOCKERFILE

Multi Stage Build

```
FROM golang:1.7.3 as builder
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get –d –v golang.org/x/net/html
COPY app.go .
RUN CGO_ENABLED=0 GOOS=linux go build –a –installsuffix cgo –o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /go/src/github.com/alexellis/href-counter/app .
COPY --from=nginx:latest /etc/nginx/nginx.conf /nginx.conf
CMD [“./app”]
```



docker

Diferencia entre

CMD, RUN y ENTRYPOINT

DIFERENCIA ENTRE CMD, RUN Y ENTRYPOINT

* **RUN** > correr comando cuando se crea la imagen.

```
FROM Python:3.8-alpine
RUN apk add --update vim
```

* **CMD** > Comandos que se ejecuta en el contenedor.

```
FROM Python:3.8-alpine
RUN apk add --update vim

WORKDIR /usr/src/myapp
COPY . .
CMD ["python", "/usr/src/myapp/server.py"]
```

```
# docker run –d –p 8080:8080 hola-python
# docker run –d –ti –p 8080:8080 hola-python sh
```

Mediante el comando CMD que utilizo en el Dockerfile, puedo sobreescribirlo por **sh**, para meterme dentro del container y entonces no se ejecutara el servicio de Python.

DIFERENCIA ENTRE CMD, RUN Y ENTRYPOINT

* **ENTYPOINT** > comando que se ejecutando cuando levanta el container, pero no se puede reemplazar como en **CMD**.

```
FROM Python:3.8-alpine
RUN apk add --update vim

WORKDIR /usr/src/myapp
COPY . .
ENTYPOINT ["python"]
```

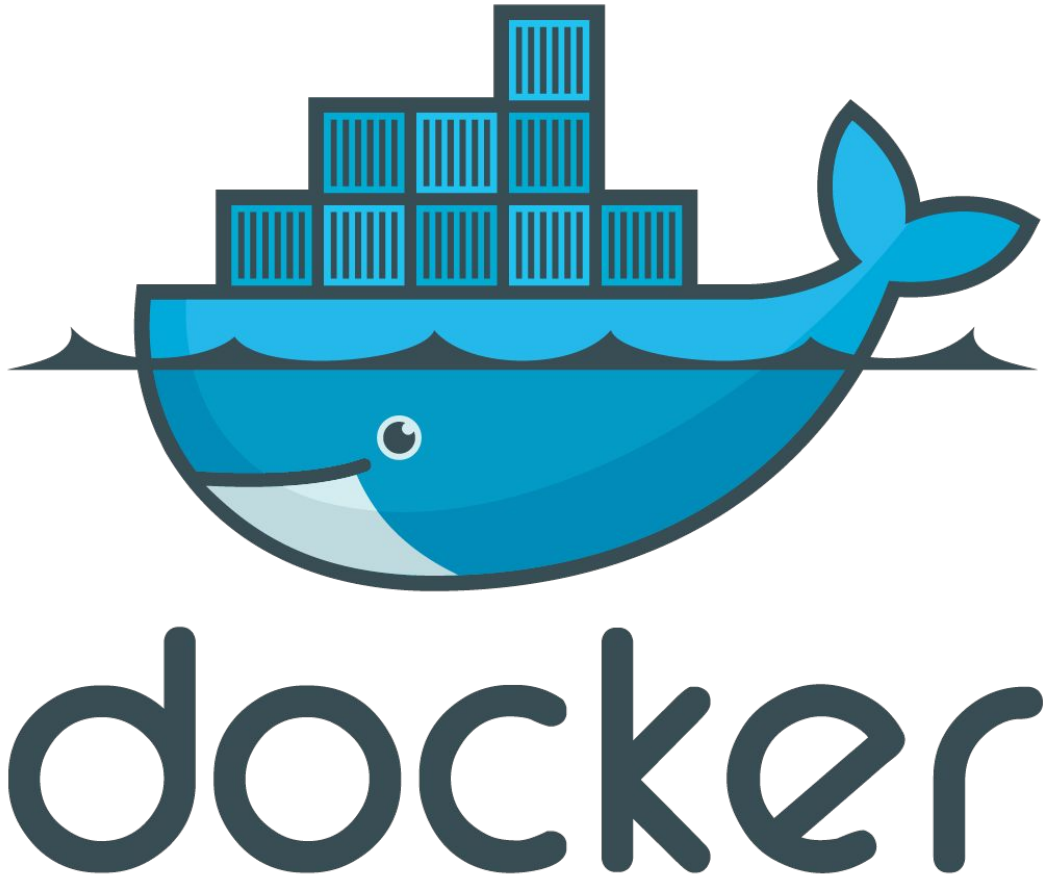
Cuando le paso como parámetro el comando **sh**, me da un error donde **python** trata de correr el comando **sh**. Lo que hago es pasarle argumentos al comando python.

```
# docker run –d –ti –p 8080:8080 hola-python sh
python: can't open file 'sh': [Errno 2] No such file or directory
```

Ahora le pasa como parámetro un programa de Python, para que sea interpretado por Python del container.

```
# docker run hola-python hello.py
```

```
# cat hello.py

print("Hello World")
```



Docker Backup

COMANDOS CONTENEDORES

Realizando Backup

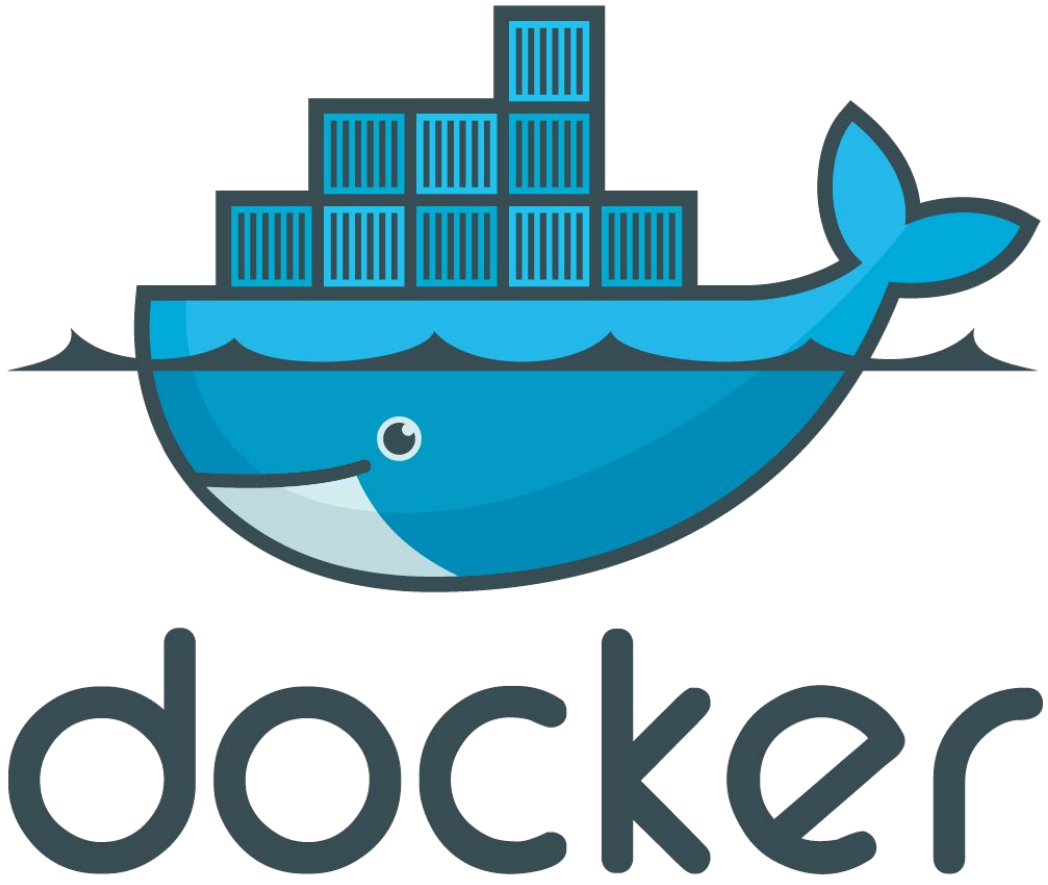
docker export web01 –o backup_web01.tar

docker export web01 –o backup_web01.tar marcospr1974:alpine-apache

Restaurando Backup

docker import backup_web01.tar

docker tag 23bc397a5bae web01



Docker Compose

DOCKER COMPOSE

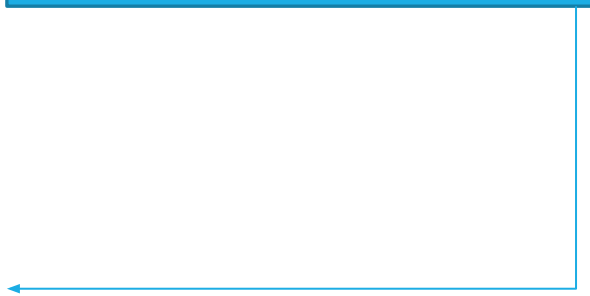
Compose es una herramienta para definir y ejecutar aplicaciones Docker de múltiples contenedores. Con Compose, utiliza un archivo Compose para configurar los servicios de su aplicación. Luego, utilizando un solo comando, creará e iniciará todos los servicios desde su configuración. Para obtener más información sobre todas las funciones de Compose, consulte la lista de funciones.

DOCKER COMPOSE

```
# vim docker-compose.yaml

version: '3.1

services:
    wordpress:
        image: wordpress:php7.2-apache
        ports:
            - 8081:80
        environment:
            - WORDPRESS_DB_HOST=mysql
            - WORDPRESS_DB_USER=root
            - WORDPRESS_DB_PASSWORD=root
            - WORDPRESS_DB_NAME=wordpress
        links:
            - mysql:mysql
```

O también se puede poner las **enviroment:**

```
WORDPRESS_DB_HOST: mysql
WORDPRESS_DB_USER: root
WORDPRESS_DB_PASSWORD: root
WORDPRESS_DB_NAME: wordpress
```



DOCKER COMPOSE

```
mysql:
    image: mysql:8.0.13
    ports:
      - 3302:3306
    command: --default-authentication-plugin=mysql_native_password
    environment:
      - MYSQL_DATABASE=wordpress
      - MYSQL_ROOT_PASSWORD=root
    volumes:
      - ~/Docker/mysql-data:/var/lib/mysql
```

DOCKER COMPOSE

Para ejecutar el archivo que armamos ejecutamos el siguiente comando:

docker-compose up –d

Para ver los container levantados:

docker-compose ps

```
 Name                 Command                State          Ports
-------------------------------------------------------------------------------------------
docker_mysql_1       docker-entrypoint.sh --def ...   Up      0.0.0.0:3302->3306/tcp, 33060/tcp
docker_wordpress_1   docker-entrypoint.sh apach ...   Up      0.0.0.0:8081->80/tcp
```

DOCKERFILE Y DOCKER COMPOSE

cat Dockerfile

```
FROM nginx:1.14.2-alpine
COPY index.html /usr/share/nginx/html
```

cat index.html

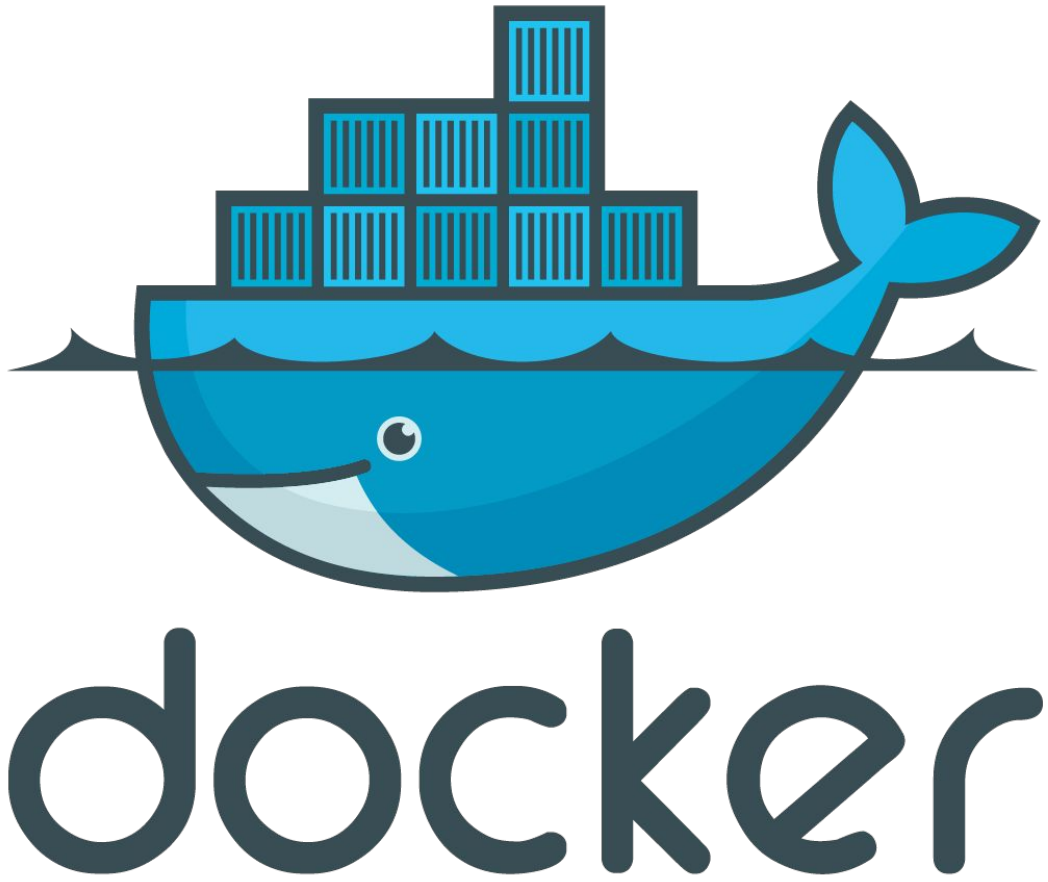
```
Hola Docker
```

cat docker-compose.yaml

```
versión: “2”
services:
    holadocker:
      build: .
      ports:
        - 80:80
```

DOCKERFILE Y DOCKER COMPOSE

```
# docker-compose build
# docker-compose up –d
# docker-compose ps
```



Docker Compose Override

DOCKER COMPOSE OVERRIDE

Esto me permite tener un archivo **docker-compose.yaml** donde puedo declarar por ejemplo las variables de entorno y en otro archivo llamado **docker-compose.override.yaml** lo que va hacer es justamente mergear ambos archivos en uno solo, donde utiliza como base el archivo **docker-compose.yaml** y cualquier cambios que haya en el archivo **docker-compose.override.yaml** va a pisar la base.

Por ejemplo si en **docker-compose.yaml** tengo una versión de **alpinelinux** 1.2 y en el archivo **docker-compose.override.yaml** la versión 1.1, se utilizara la versión 1.1.

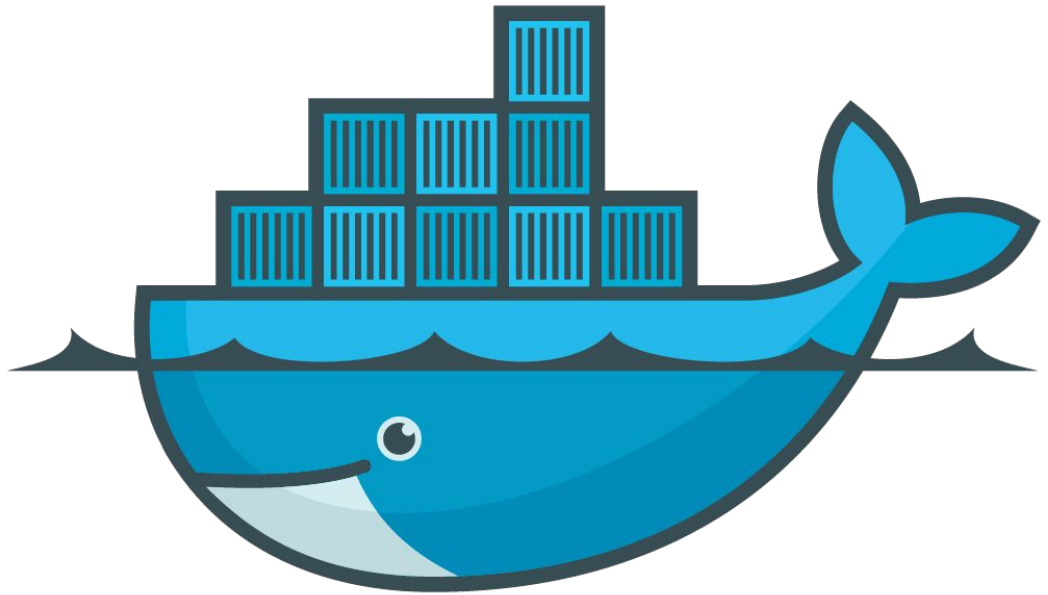
DOCKER COMPOSE OVERRIDE

cat docker-compose.yaml

```
versión: “3”
services:
    nginx:
        image: nginx:1.14.2-alpine
        environment:
            - MYSQL_PASSWORD=producción
```

cat docker-compose.override.yaml

```
versión: “3”
services:
    nginx:
        image: nginx:1.14.1-alpine
```



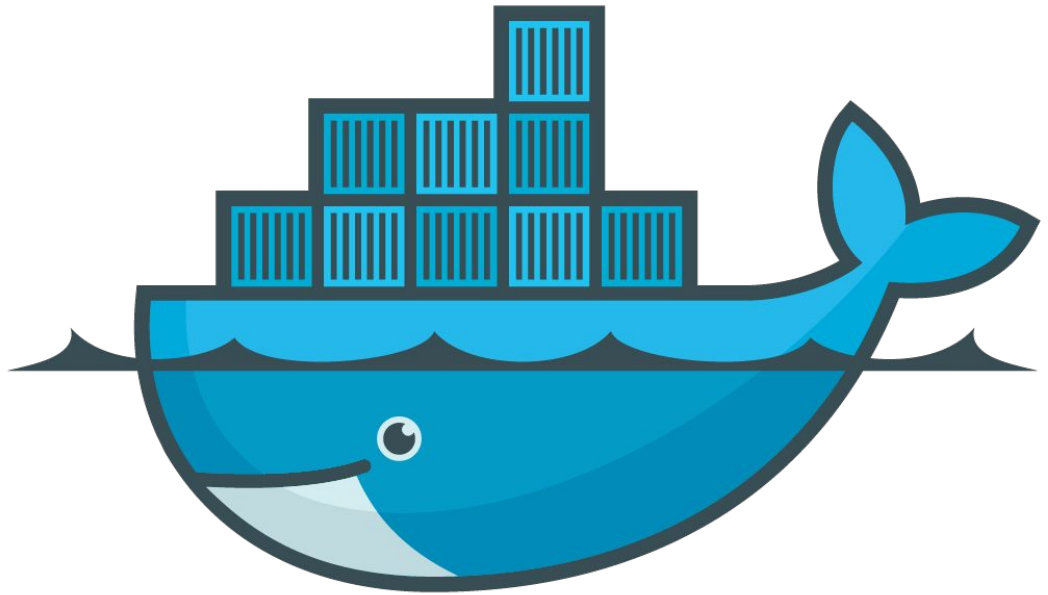
APP. GRAFICA

COMANDOS CONTENEDORES

```
# cat dockerfile

FROM ubuntu:latest

MAINTAINER "Marcos Pablo Russo version: 0.1"

RUN apt-get update && apt-get upgrade –y && apt-get install kodi sudo –y
RUN export uid=1000 gid=1000 && \
        mkdir –p /home/developer && \
        echo “developer:x:${uid}:${gid}:Developer,,,:/home/developer:/bin/bash” >> /etc/passwd
RUN echo “developer:x:${uid}:” >> /etc/group && \
        echo “developer ALL=(ALL) NOPASSWD: ALL” > /etc/sudoers.d/developer && \
        chmod 0440 /etc/sudoers.d/developer && \
        chown ${uid}:${gid} –R /home/developer

USER developer

ENV HOME /home/developer
CMD /usr/bin/kodi
```

COMANDOS CONTENEDORES

```
# docker build –t kodi .

# docker images


# docker run –ti –rm \
      -e DISPLAY=$DISPLAY \
            –v /tmp/.X11-unix:/tmp/.X11-unix \
      kodi
```



EJEMPLO

COMANDOS CONTENEDORES

```
# docker pull mariadb

# docker run –d --name dba00 –p 3306:3306 \
    –e MYSQL_ROOT_PASSWORD=‘debian’ mariadb

# docker ps
```

```
localhost:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
          NAMES
a728aeb6fb78        mariadb             "docker-entrypoint.s…"   14 seconds ago      Up 12 seconds       0.0.0.0:3306->3
306/tcp   dba00
```

COMANDOS CONTENEDORES

docker log dba00

```
2020-03-31 21:45:09+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 1:10.4.12+maria~bionic started.
2020-03-31 21:45:10+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2020-03-31 21:45:10+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 1:10.4.12+maria~bionic started.
2020-03-31 21:45:10+00:00 [Note] [Entrypoint]: Initializing database files


PLEASE REMEMBER TO SET A PASSWORD FOR THE MariaDB root USER !
To do so, start the server, then issue the following commands:

'/usr/bin/mysqladmin' -u root password 'new-password'
'/usr/bin/mysqladmin' -u root -h  password 'new-password'

Alternatively you can run:
'/usr/bin/mysql_secure_installation'

which will also give you the option of removing the test
databases and anonymous user created by default.  This is
strongly recommended for production servers.

See the MariaDB Knowledgebase at http://mariadb.com/kb or the
MySQL manual for more instructions.

Please report any problems at http://mariadb.org/jira
```

COMANDOS CONTENEDORES

```
# docker exec –ti dba00 /bin/bash

# mysql –u root mysql –p

# create database wordpress;

# grant all privileges on wordpress.* to wordpress@'%' identified by 'wordpress';

# docker inspect  dba00
```

COMANDOS CONTENEDORES

```
# docker pull ulsmith/alpine-apache-php7
# mkdir –p /Volumen/web00
# cd /Volumen/web00
# wget https://wordpress.org/latest.tar.gz
# docker run -d -v /Volumen/web00:/app/public --name web00 \
   --hostname web00 -p 80:80 --link dba00 ulsmith/alpine-apache-php7

# docker exec –ti web00 /bin/sh
```

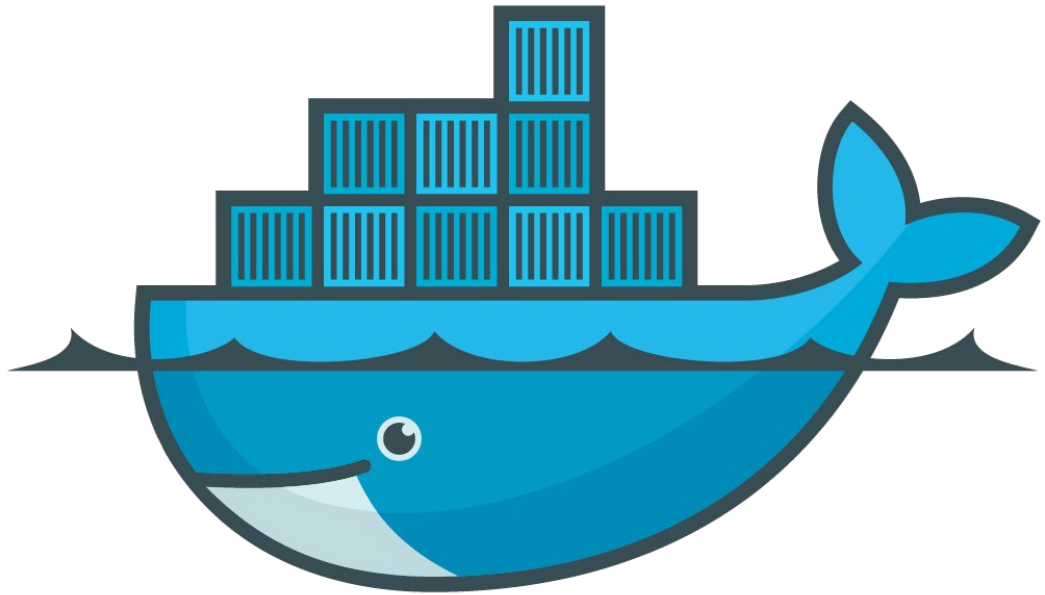
https://hub.docker.com/r/nimmis/alpine-apache

COMANDOS CONTENEDORES

```
localhost:/Volumen/web00# docker exec -ti web00 /bin/sh
eenv
/ # env
DBA00_ENV_GOSU_VERSION=1.10
DBA00_PORT_3306_TCP_PORT=3306
DBA00_PORT_3306_TCP_PROTO=tcp
HOSTNAME=web00
DBA00_ENV_MARIADB_MAJOR=10.4
SHLVL=1
HOME=/root
DBA00_PORT=tcp://172.17.0.2:3306
DBA00_PORT_3306_TCP=tcp://172.17.0.2:3306
DBA00_NAME=/web00/dba00
TERM=xterm
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
DBA00_ENV_GPG_KEYS=177F4010FE56CA3336300305F1656F24C74CD1D8
DBA00_ENV_MARIADB_VERSION=1:10.4.12+maria~bionic
PWD=/
DBA00_PORT_3306_TCP_ADDR=172.17.0.2
DBA00_ENV_MYSQL_ROOT_PASSWORD=debian
```

COMANDOS CONTENEDORES

docker stats



docker

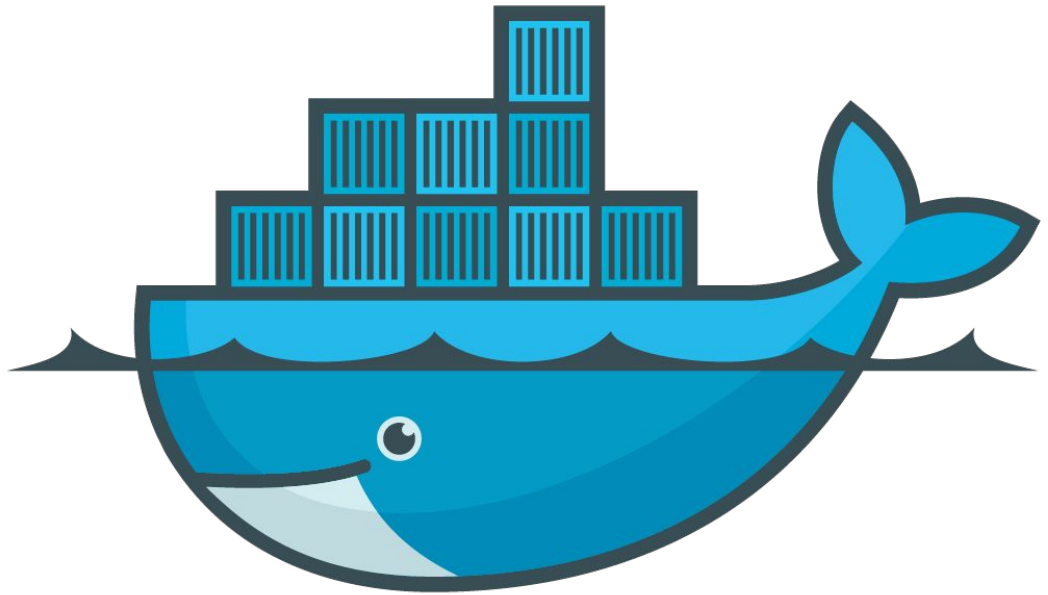
DOCKER HUB

COMANDOS CONTENEDORES

```
# docker login

Login with your Docker ID to push and pull images from Docker Hub. If you don't have a
Docker ID, head over to https://hub.docker.com to create one.

Username: usuario
Password:
Login Succeeded
```

COMANDOS CONTENEDORES

```
# docker tag ubuntu:mc marcospr1974/ubuntu:mc

# docker images

REPOSITORY              TAG         IMAGE ID        CREATED          SIZE
ubuntu                  mc          89882c858e15    2 minutes ago    216MB
marcospr1974/ubuntu     mc          89882c858e15    2 minutes ago    216MB
…

# docker push marcospr1974/ubuntu:mc

The push refers to repository [docker.io/marcospr1974/ubuntu]
1f411dbc3047: Pushed
059ad60bcacf: Mounted from library/ubuntu
8db5f072feec: Mounted from library/ubuntu
67885e448177: Mounted from library/ubuntu
ec75999a0cb1: Mounted from library/ubuntu
65bdd50ee76a: Mounted from library/ubuntu
mc: digest: sha256:905a4074bb191c28c46e33d2d6967eab79b46ac34bdb27b72835ee
      372ed01f60 size: 1569
```



docker

PORTAINER

PORTAINER

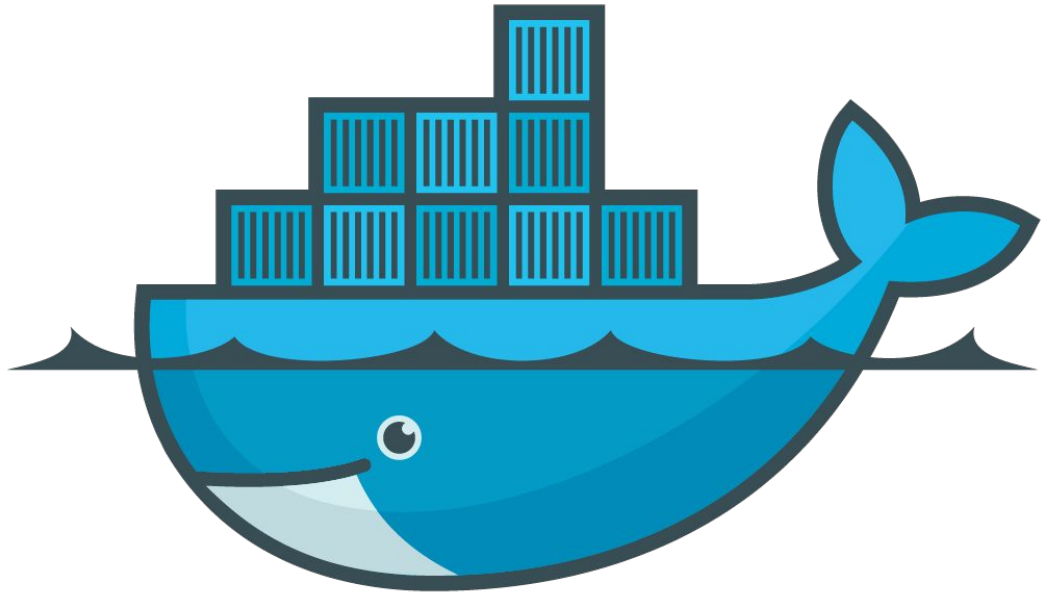
https://www.portainer.io/

```
# docker pull portainer/portainer

# docker volumen create portainer_data

# echo –n “mypassword” > /tmp/portainer_password

# docker run –d –p 8000:8000 –p 9000:9000 \
                -v /var/run/dockersock:/var/run/docker.sock \
                -v /tmp/portainer_password:/tmp/portainer_password \
                -v portainer_data:/data \
                --restart always \
                portainer/portainer \
          --admin-password-file /tmp/portainer_password
```

https://downloads.portainer.io/edge_agent_guide.pdf
https://portainer.readthedocs.io/en/stable/configuration.html

PORTAINER

http://127.0.0.1:9000



docker

TIPS

TIPS

Ver estado de los containter

docker stats

```
CONTAINER ID        NAME                                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O           PIDS
282bc17d2cd3        portainer                           0.00%               10.33MiB / 15.44GiB   0.07%               289kB / 4.42MB      70.8MB / 303kB      11
9af4e4763532        alpine-flask-tusalida_tusalida_1    4.55%               78.63MiB / 15.44GiB   0.50%               14.8kB / 0B         82.6MB / 0B         4
```

Borrar container,volumnes,network que no usamos y liberar espacio.

docker system prune

```
WARNING! This will remove:
  - all stopped containers
  - all networks not used by at least one container
  - all dangling images
  - all dangling build cache

Are you sure you want to continue? [y/N] y
Deleted Containers:
f3f5c5e881f61a9cd3e6b83ad39efe8743dfbbb678d9a3e8e0c2cedc000861b3

Deleted Networks:
5_lan
opencti-docker_default
rtorrent_default
docker_default

Total reclaimed space: 0B
```

TIPS

Ver estado de los containter

docker stats

```
CONTAINER ID        NAME                                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O           PIDS
282bc17d2cd3        portainer                           0.00%               10.33MiB / 15.44GiB   0.07%               289kB / 4.42MB      70.8MB / 303kB      11
9af4e4763532        alpine-flask-tusalida_tusalida_1    4.55%               78.63MiB / 15.44GiB   0.50%               14.8kB / 0B         82.6MB / 0B         4
```

Borrar container,volumnes,network que no usamos y liberar espacio.

docker system prune

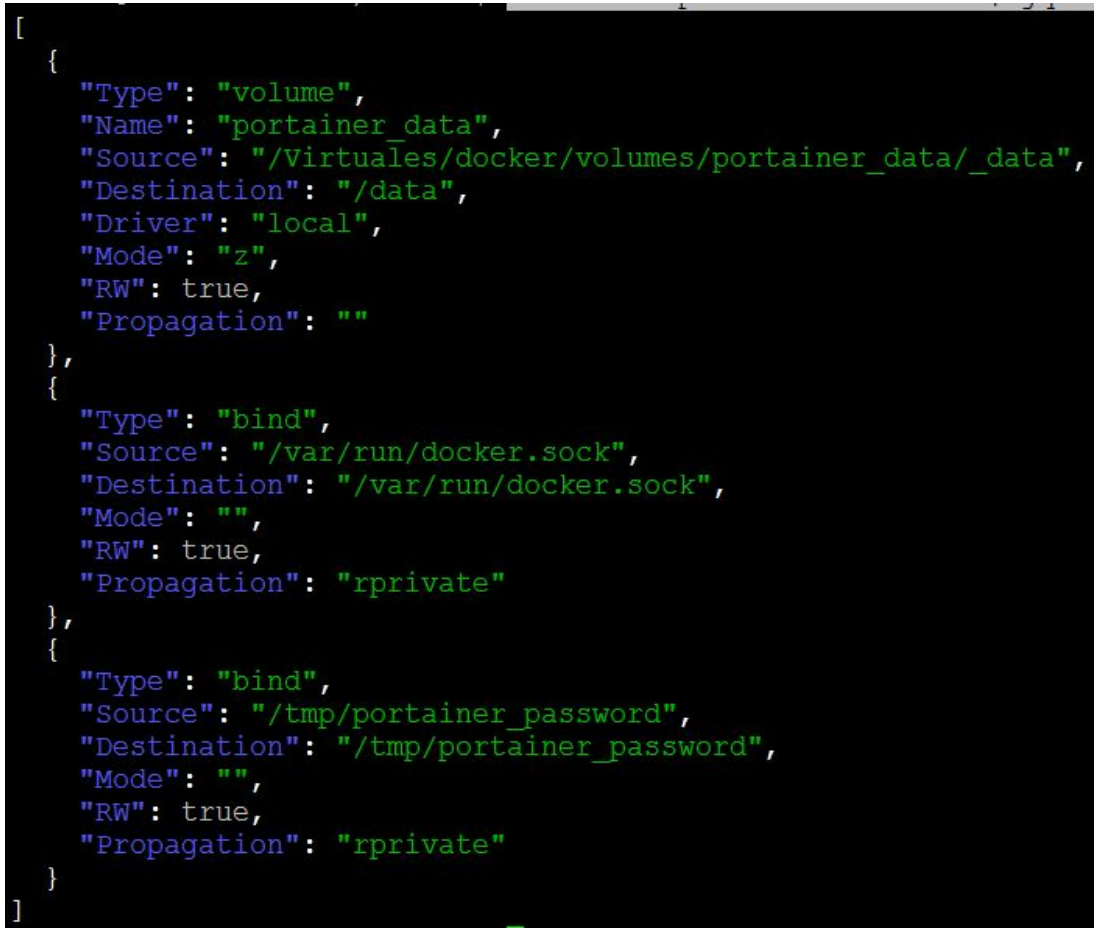
```
WARNING! This will remove:
  - all stopped containers
  - all networks not used by at least one container
  - all dangling images
  - all dangling build cache

Are you sure you want to continue? [y/N] y
Deleted Containers:
f3f5c5e881f61a9cd3e6b83ad39efe8743dfbbb678d9a3e8e0c2cedc000861b3

Deleted Networks:
5_lan
opencti-docker_default
rtorrent_default
docker_default

Total reclaimed space: 0B
```

TIPS

Convertir en json el resultado de inspect

apt-get install jq

docker inspect 282bc17d2cd3 | jq '.[].Mounts'



TIPS

Copiar del contenedor a mi directorio

docker cp 282bc17d2cd3:/etc/nginx/nginx.conf .

Diferencia de COPY y ADD en Dockerfile.

* COPY copia archivos.
* ADD puedo poner una ruta url sin tener instalador wget.

EJ: ADD example.com/archivo /tmp

TIPS

Docker Compose

```
nginx-proxy
    image: jwilder/nginx-proxy
    restart: always
    ports:
        - 80:80
        - 443:443
    volumes:
        - /var/run/docker.sock:/tmp/docker.sock.ro
        ....
```

Si se reinicia la pc, el demonio de docker levantara el containerd.

Levanta el volumen solo como lectura.

TIPS

Docker Compose

```
nginx-proxy
    image: jwilder/nginx-proxy
    restart: unless-stopped
    ports:
        - 80:80
        - 443:443
    volumes:
        - /var/run/docker.sock:/tmp/docker.sock.ro
        ....
```

Si nosotros bajamos el container, luego no lo levantara si reiniciamos la pc.

Levanta el volumen solo como lectura.

TIPS

Docker Compose

```
nginx-proxy
    image: jwilder/nginx-proxy
    restart: on-failure:10
    ports:
        - 80:80
        - 443:443
    volumes:
        - /var/run/docker.sock:/tmp/docker.sock.ro
        ....
```

Reinicia todas las veces hasta que falle 10 veces.

Levanta el volumen solo como lectura.

TIPS

Ver las ultimass 10 líneas del log

docker ps
docker logs --tail=10 5332975a7e1e

```
* Debugger PIN: 169-053-467
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 210-738-562
```

TIPS

Ver las ultimas 10 líneas del log y saber desde cuando es la fecha de esos logs.

docker ps
docker logs --tail=10 -t 5332975a7e1e

```
2020-04-25T19:52:58.194328628Z  * Debugger PIN: 169-053-467
2020-04-25T19:56:59.896376521Z  * Serving Flask app "app" (lazy loading)
2020-04-25T19:56:59.896590280Z  * Environment: production
2020-04-25T19:56:59.896596213Z    WARNING: This is a development server. Do not use it in a production deployment.
2020-04-25T19:56:59.896602684Z    Use a production WSGI server instead.
2020-04-25T19:56:59.896605451Z  * Debug mode: on
2020-04-25T19:56:59.906308137Z  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
2020-04-25T19:56:59.907362264Z  * Restarting with stat
2020-04-25T19:57:00.610398992Z  * Debugger is active!
2020-04-25T19:57:00.620897865Z  * Debugger PIN: 210-738-562
```

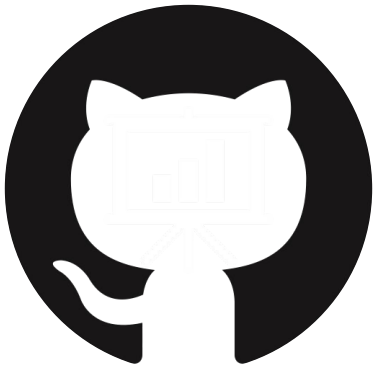
TIPS

Inicializar un contenedor por ejemplo de nginx, pero ejecutar antes un bash (Shell) antes de ejecutar nginx, que nos permite realizar correcciones, esto se realizar cambiando el entrypoint.

docker run –it –v /root/nginx.conf:/etc/nginx/nginx.conf --entrypoint=bash nginx

RECURSOS

SO-UTNFRA.SLACK.COM



GITHUB.COM/MARTIN919191/ARQUITECTURAYSISTEMASOPERATIVOS

SO.UTNFRA@GMAIL.COM