

DIGITAL STRATEGY & TECHNOLOGY

Curso: Docker



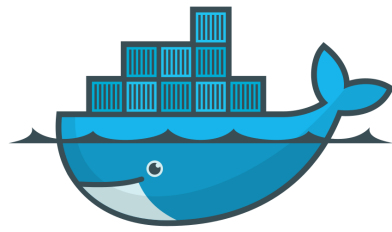
Miguel Ángel Timaná Paz
Solutions Specific Knowledge Leader
Technology
mtimanap@everis.com



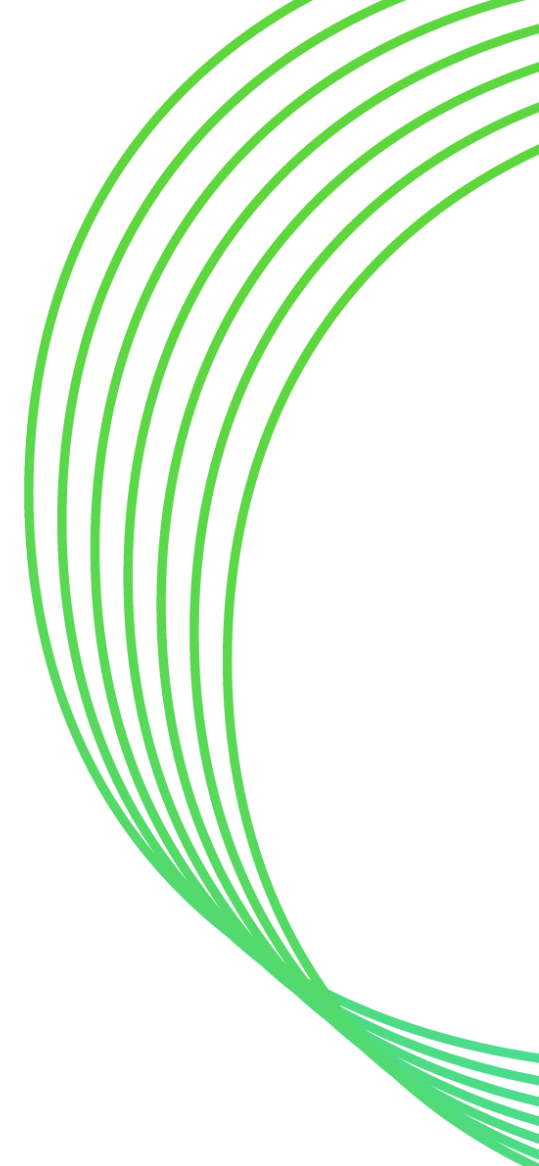
Confidentiality

This document is confidential and property of everis. Its use, reproduction or distribution without prior written permission from everis is strictly prohibited.

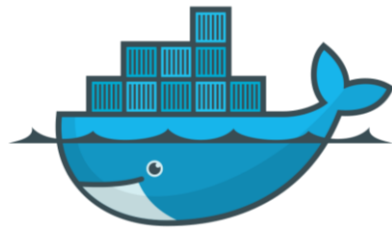
Objetivos



- ✓ Conocer que es Docker.
- ✓ Profundizar los conceptos imágenes, contenedores, redes.
- ✓ Que es Docker-composer.
- ✓ Que es Docker Swarm.
- ✓ Casos prácticos.



Agenda



01 Introducción a Docker

02 Imágenes

03 Contenedores

04 Docker-composer

05 Docker Swarm

06 Casos practicos

01 Introducción a Docker

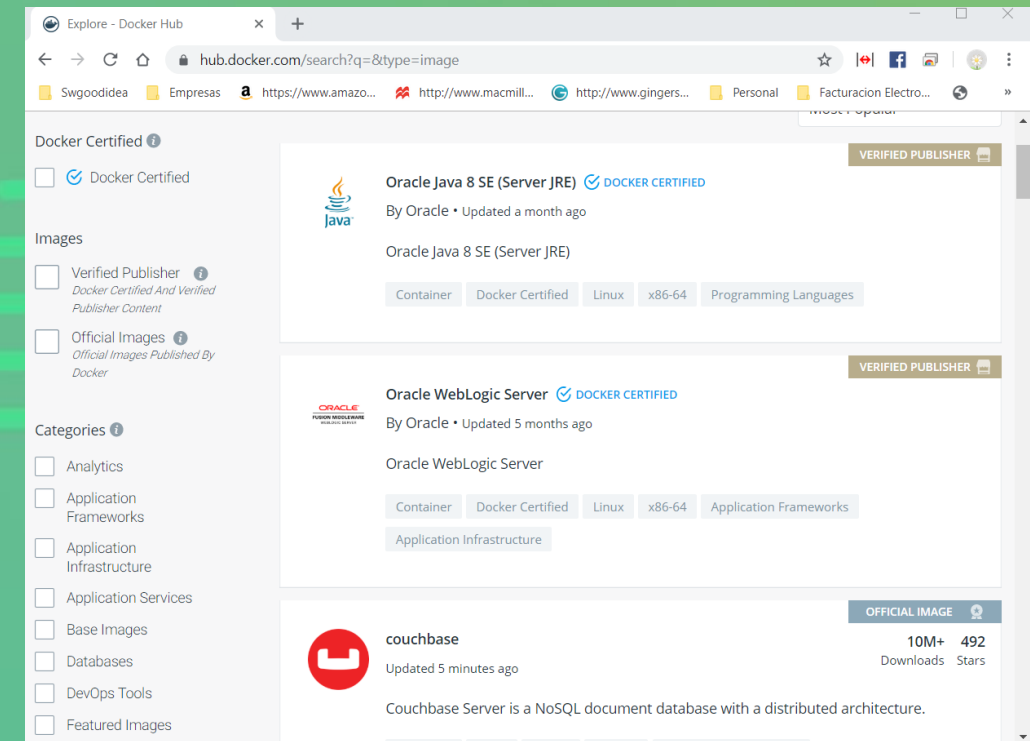


¿ Que es Docker ?

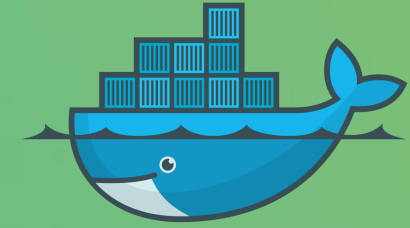
- Es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

¿ Que es docker hub ?

- Es un repositorio público en la nube, similar a Github, para distribuir los contenidos. Está mantenido por la propia Docker y hay multitud de imágenes, de carácter gratuito, que se pueden descargar y así no tener que hacer el trabajo desde cero al poder aprovechar "plantillas". También podemos crear nuestros propios repositorios privados e, incluso, dispone de una tienda.



01 Introducción a Docker



Docker Engine:

Es un demonio que corre sobre cualquier distribución de Linux y que expone una API externa para la gestión de imágenes y contenedores.

Docker Client:

Es el cliente de línea de comandos (CLI) que nos permite gestionar el Docker Engine. El cliente docker se puede configurar para trabajar con un Docker Engine local o remoto.

01 Introducción a Docker



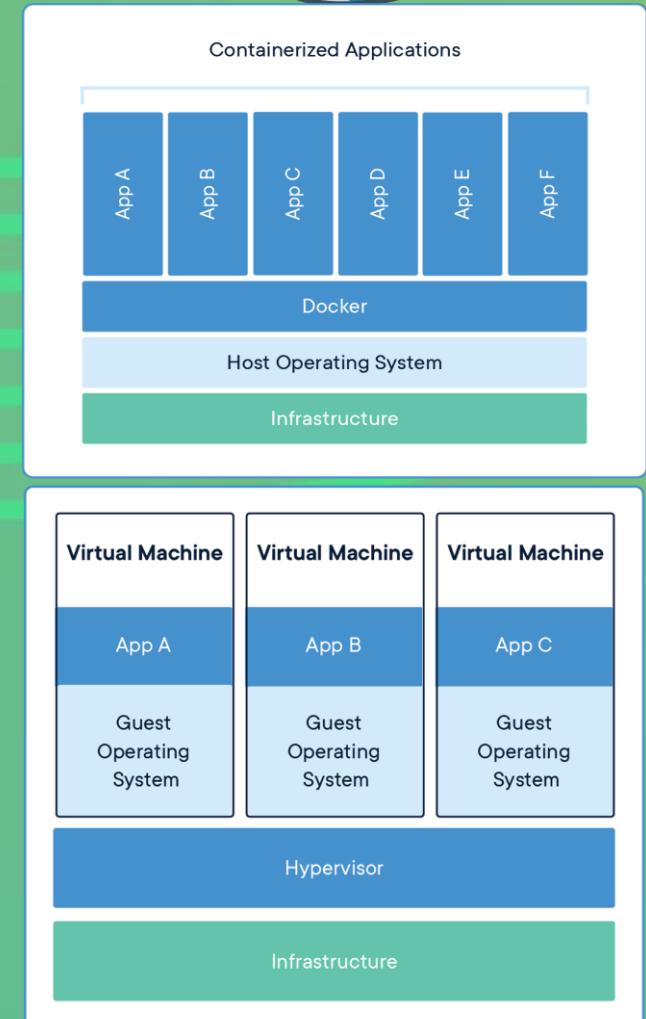
Containerization:

Es una alternativa ligera a la virtualización completa de la máquina que implica encapsular una aplicación en un contenedor con su propio entorno operativo. Los contenedores usan el mismo sistema operativo host (SO) repetidamente, en lugar de instalar un SO para cada VM invitada.

Los contenedores también proporcionan una forma de aislar aplicaciones y proporcionan una plataforma virtual para que las aplicaciones se ejecuten.

Virtualization:

Una máquina virtual es una copia de un servidor completo, básicamente tiene su propio sistema operativo, lo que significa la replicación de binarios, controladores o bibliotecas entre diferentes máquinas virtuales que se ejecutan en el mismo servidor y desperdicia recursos significativos del servidor.



01 Introducción a Docker



Instalación Ubuntu 18.04:

```
mtimanap@laptop:/#sudo su
```

```
root@laptop:/#apt-get update
```

```
root@laptop:/#apt-get install apt-transport-https ca-certificates curl software-properties-common
```

```
Do you want to continue? [Y/n] Y
```

```
root@laptop:/# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
ok
```

```
root@laptop:/# add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic  
stable"
```

```
Fetches 163 kB in 2s (104 kB/s)
```

```
root@laptop:/#apt-get update
```

```
Fetches 88,7 kB in 1s (63,6 kB/s)
```

```
Reading package lists... Done
```

01 Introducción a Docker



Instalación Ubuntu 18.04:

```
root@laptop:/#apt-cache policy docker-ce
docker-ce:
```

```
Installed: (none)
```

```
Candidate: 5:19.03.4~3-0~ubuntu-bionic
```

```
Version table:
```

```
5:19.03.4~3-0~ubuntu-bionic 500
```

```
root@laptop:/#apt-get install docker-ce
```

```
After this operation, 418 MB of additional disk space will be used.
```

```
Do you want to continue? [Y/n] Y
```

```
root@laptop:/#systemctl status docker
```

```
docker.service - Docker Application Container Engine
```

```
Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
```

```
Active: active (running) since Fri 2019-11-01 23:09:37 -05; 4min 42s ago
```

```
Docs: https://docs.docker.com
```



01 Introducción a Docker

Instalación Ubuntu 18.04:

```
root@laptop:/#exit
mtimanap@laptop:/#sudo usermod -aG docker ${USER}
```

```
[sudo] password for mtimanap:
```

```
mtimanap@laptop:/#reboot
```

```
mtimanap@laptop:/#docker ps
```

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

```
mtimanap@laptop:/#docker run --rm -p 80:80 nginx
```

```
Unable to find image 'nginx:latest' locally
```

```
latest: Pulling from library/nginx
```

```
8d691f585fa8: Pull complete
```

```
5b07f4e08ad0: Pull complete
```

```
abc291867bca: Pull complete
```

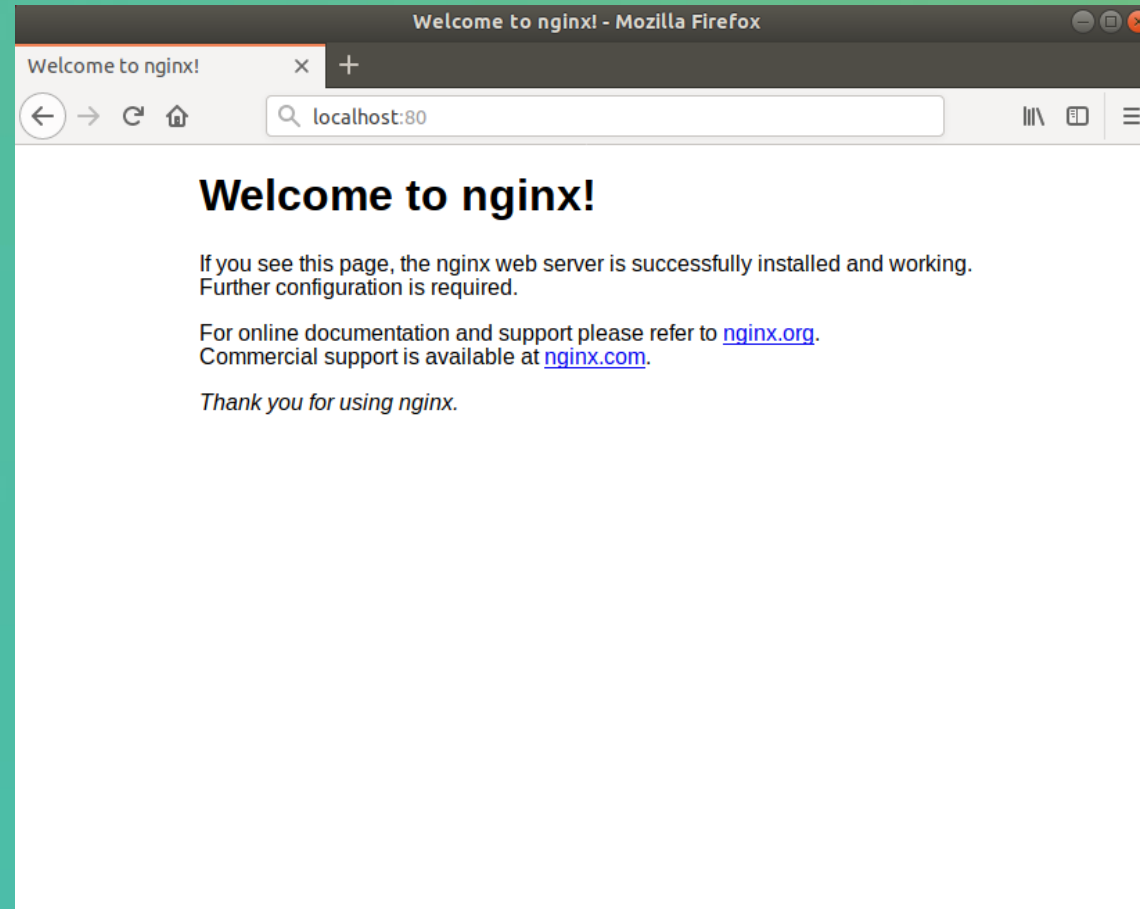
```
Digest: sha256:922c815aa4df050d4df476e92daed4231f466acc8ee90e0e774951b0fd7195a4
```

```
Status: Downloaded newer image for nginx:latest
```



01 Introducción a Docker

Instalación Ubuntu 18.04:



01 Introducción a Docker



Instalación Ubuntu 18.04:

```
mtimanap@laptop:/# docker run --rm -p 8080:8080 -p 9990:9990 jboss/wildfly
```



02 Imágenes



➤ Que es una imagen?

Son plantillas (que incluyen una aplicación, los binarios y las librerías necesarias) que se utilizan para construir contenedores Docker y ejecutarlos (los contenedores ejecutarán una imagen previamente compilada).

Por ejemplo una imagen podría contener un sistema operativo Ubuntu con un servidor Apache y tu aplicación web instalada.

Las imágenes se utilizan para crear contenedores, y nunca cambian. Hay muchas imágenes públicas con 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 (p.e: una imagen padre con Ubuntu y Apache, que hemos modificado para instalar nuestra aplicación).

02 Imágenes



Que es Docker Hub?

Es un repositorio público en la nube, similar a Github, para distribuir los contenidos. Está mantenido por la propia Docker y hay multitud de imágenes, de carácter gratuito, que se pueden descargar y así no tener que hacer el trabajo desde cero al poder aprovechar “plantillas”. También podemos crear nuestros propios repositorios privados e, incluso, dispone de una tienda.

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

Que otras alternativas tenemos?

- ✓ Azure Container Registry: <https://azure.microsoft.com/en-us/services/container-registry/>
- ✓ Amazon Elastic Container Registry (Amazon ECR):
<https://aws.amazon.com/es/getting-started/tutorials/deploy-docker-containers/>
- ✓ Google Container Registry: <https://cloud.google.com/container-registry/>
- ✓ IBM Cloud Container Registry: https://cloud.ibm.com/docs/services/Registry?topic=registry-registry_overview
- ✓ **Crear tu propio Container Registry**

02 Imagenes



#Para ver las imagenes en maquina local

```
mtimanap@laptop:/#docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	540a289bab6c	10 days ago	126MB
hello-world	latest	fce289e99eb9	10 months ago	1.84kB

#Para descargar una imagen

```
mtimanap@laptop:/#docker pull jboss/wildfly
```

```
mtimanap@laptop:/#docker pull postgres
```

Using default tag: latest

latest: Pulling from library/postgres

8d691f585fa8: Already exists

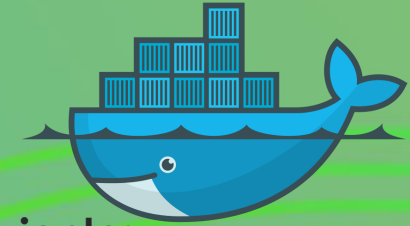
c991029393ff: Pull complete

d104c69c9175: Pull complete

#Para poder usar la imagen descargada de postgres

```
mtimanap@laptop:/#docker run --rm -e POSTGRES_PASSWORD=123456 -p 5432:5432 postgres
```


02 Imagenes



#Para crear nuestra imagen primero debemos crear un archivo Dockerfile con lo siguiente:

```
FROM alpine
```

```
CMD echo "Hola mundo"
```

#Después creamos la imagen de Docker con el siguiente comando:

```
mtimanap@laptop:/#docker build -t miimagen .
```

```
Sending build context to Docker daemon 2.048kB
```

```
Step 1/2 : FROM alpine
```

```
latest: Pulling from library/alpine
```

```
89d9c30c1d48: Pull complete
```

```
Digest: sha256:c19173c5ada610a5989151111163d28a67368362762534d8a8121ce95cf2bd5a
```

```
Status: Downloaded newer image for alpine:latest
```

```
---> 965ea09ff2eb
```

```
Step 2/2 : CMD echo "Hola mundo"
```

```
---> Running in f5d048efed5d
```

```
Removing intermediate container f5d048efed5d
```

```
---> 6cb15d46f2c2
```

```
Successfully built 6cb15d46f2c2
```

```
Successfully tagged miimagen:latest
```

02 Imagenes



#Listamos nuestras imagenes

```
mtimanap@laptop:/#docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
miimagen	latest	6cb15d46f2c2	2 minutes ago	5.55MB
nginx	latest	540a289bab6c	10 days ago	126MB
alpine	latest	965ea09ff2eb	11 days ago	5.55MB

#Para correr nuestra imagen creada

```
mtimanap@laptop:/#docker run --rm miimagen
```

```
Hola mundo
```

#Tagear una imagen

```
mtimanap@laptop:/#docker tag miimagen:latest miguelangeltimanapaz/miimage:1.0.RC1
```

```
mtimanap@laptop:/#docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
miimagen	latest	6cb15d46f2c2	11 minutes ago	5.55MB
miguelangeltimanapaz/miimage	1.0.RC1	6cb15d46f2c2	11 minutes ago	5.55MB
nginx	latest	540a289bab6c	10 days ago	126MB
alpine	latest	965ea09ff2eb	11 days ago	5.55MB
postgres	latest	f88dfa384cc4	2 weeks ago	348MB

02 Imagenes



#Crear imagen mas avanzada(crear archive dockerFileApp)

```
FROM store/oracle/serverjre:8
ENV APP_HOME=/opt/sales
WORKDIR $APP_HOME
COPY apiuat.cer $APP_HOME/apiuat.cer
COPY miapp-service-0.0.1-SNAPSHOT.war $APP_HOME/app-service-0.0.1-SNAPSHOT.war
RUN mkdir -p $APP_HOME/log/
RUN /usr/java/latest/bin/keytool -importcert -file $APP_HOME/apiuat.cer -keystore
/usr/java/latest/jre/lib/security/cacerts -alias "interbank uat" -storepass "changeit" -noprompt
ENTRYPOINT ["java","-Duser.timezone=GMT-05:00","-jar","app-service-0.0.1-SNAPSHOT.war","--
spring.profiles.active=uat"]
EXPOSE 80
```

#Crear imagen ya tagueada con archive especifico

```
mtimanap@laptop:/#docker build --tag=acre.azurecr.io/empresa/miapp:1.0-uat --file=dockerFileApp .
```

#Para ver toda la documentación de como construir una imagen

<https://docs.docker.com/engine/reference/commandline/build/>

02 Imagenes



#Listar todas las images

```
mtimanap@laptop:/# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
miguelangelmtimanapaz/miimage	1.0.RC1	6cb15d46f2c2	11 hours ago	5.55MB
acre.azurecr.io/empresa/miapp	1.0-uat	6cb15d46f2c2	11 hours ago	5.55MB

#Para ver el historial de una imagen: docker history [IMAGE ID]

```
mtimanap@laptop:/# docker history f88dfa384cc4
```

#Para inspeccionar una imagen: docker inspect [IMAGE ID]

```
mtimanap@laptop:/# docker inspect f88dfa384cc4
```

#Para eliminar una imagen: docker rmi [IMAGE ID]

```
mtimanap@laptop:/# docker rmi fce289e99eb9
```

#Para eliminar una imagen forzando: docker rmi -f [IMAGE ID]

```
mtimanap@laptop:/# docker image rmi -f fce289e99eb9
```

#Para exportar una imagen: docker save [REPOSITORY]:[TAG] | gzip > [FILE NAME].tar.gz

```
mtimanap@laptop:/# docker save postgres:latest | gzip > postgres.tar.gz
```

#Para importar imagen: docker load -i [File Name].tar.gz

```
mtimanap@laptop:/# docker load -i postgres.tar.gz
```


02 Imagenes



#Para subir mi imagen a docker hub

```
mtimanap@laptop:/#docker login --username=miguelangeltimanapaz
```

Password:

Login Succeeded

#Crear imagen tageando con tu nombre de usuario

```
mtimanap@laptop:/#docker build -t miguelangeltimanapaz/miimagen -f dockerFileApp .
```

#Subimos nuestra imagen a docker hub

```
mtimanap@laptop:/#docker push miguelangeltimanapaz/miimagen:latest
```

#Para subir mi imagen a azure container, primero nos logeamos a zure

```
mtimanap@laptop:/#az login --use-device-code
```

#Luego nos logeamos a nuestro registry

```
mtimanap@laptop:/#az acr login --name acreu2c008batouat01
```

#Finalmente subimos nuestra imagen al container registry de azure

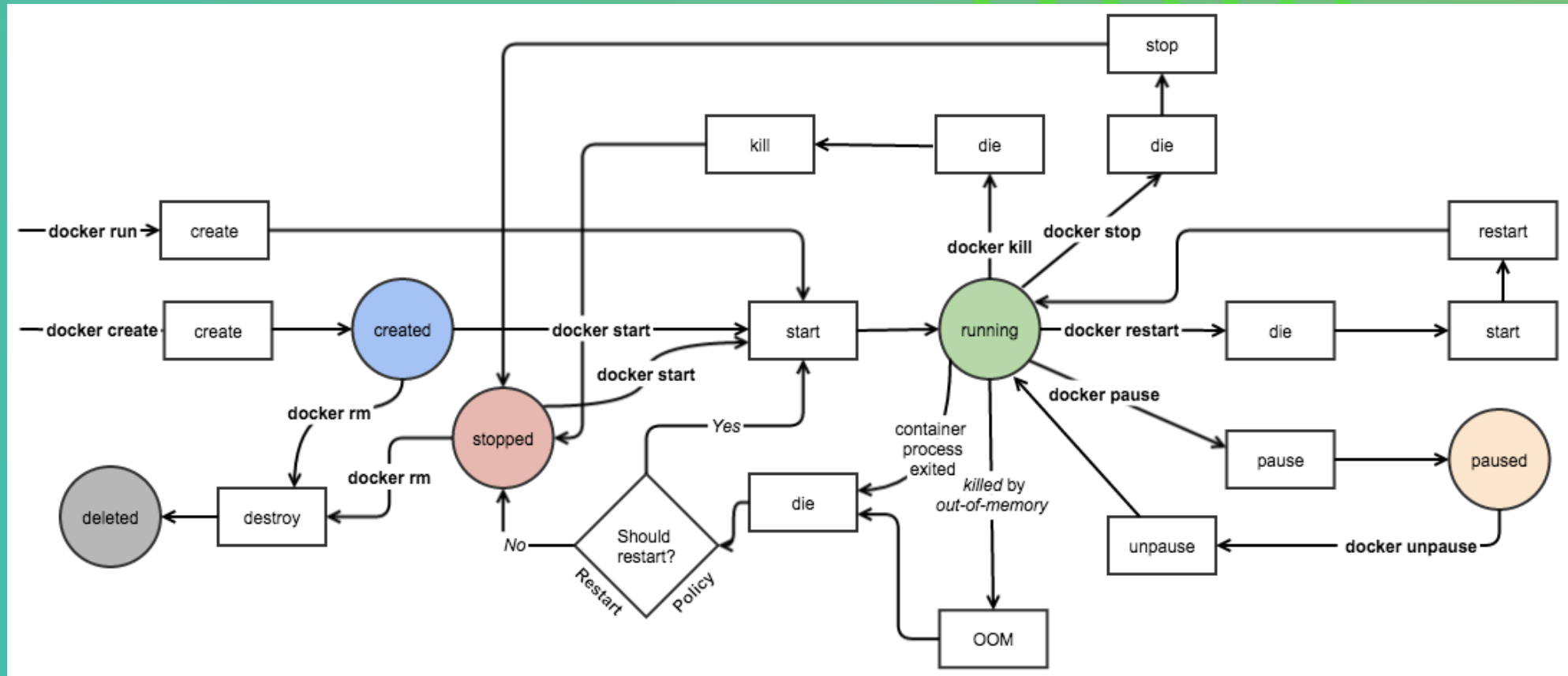
```
mtimanap@laptop:/#docker push acreu2c008batouat01.azurecr.io/interbank/digital-sales-services:1.0-uat
```

Para mas detalle:

<https://docs.docker.com/engine/reference/commandline/images/>

03 Contenedores

Ciclo de vida de un contenedor





03 Contenedores

#Para ver todos los contenedores con sus estados: docker ps -a

mtimanap@laptop:/#docker ps -a

#Crear mi primer contenedor

mtimanap@laptop:/#docker create -e POSTGRES_PASSWORD=123456 -p 5432:5432 postgres

Digest: sha256:a4a944788084a92bcaff6180833428f17cceb610e43c828b3a42345b33a608a7

Status: Downloaded newer image for postgres:latest

28bb193cbf9824b9581c60f9773fb75b53e9314e50f40a2370ef1b4441af401d

mtimanap@laptop:/#docker ps -a

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
28bb193cbf98	postgres	"docker-entrypoint.s..."	4 minutes ago	Created		vigilant_jepsen

#Para iniciar un container: docker container start [CONTAINER ID]

mtimanap@laptop:/#docker start 28bb193cbf98

28bb193cbf98

mtimanap@laptop:/#docker ps -a

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
28bb193cbf98	postgres	"docker-entrypoint.s..."	3 hours ago	Up 5 seconds	5432/tcp	vigilant_jepsen



03 Contenedores

#Ejecutar comandos dentro del container

```
mtimanap@laptop:/#docker exec -it 5f32fa3698a2 /bin/bash
root@5f32fa3698a2:/#psql -U postgres
psql (12.0 (Debian 12.0-2.pgdg100+1))
Type "help" for help.
postgres=#
```

#Para detener un container: docker stop [CONTAINER ID]

```
mtimanap@laptop:/#docker stop 28bb193cbf98
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
28bb193cbf98	postgres	"docker-entrypoint.s..."	3 hours ago	Exited (0) 4 seconds ago		vigilant_Jepsen

#Para eliminar un container: docker rm [CONTAINER ID]

```
mtimanap@laptop:/#docker rm 28bb193cbf98
28bb193cbf98
```

#Listamos de nuevo nuestros containers

```
mtimanap@laptop:/#docker ps -a
```

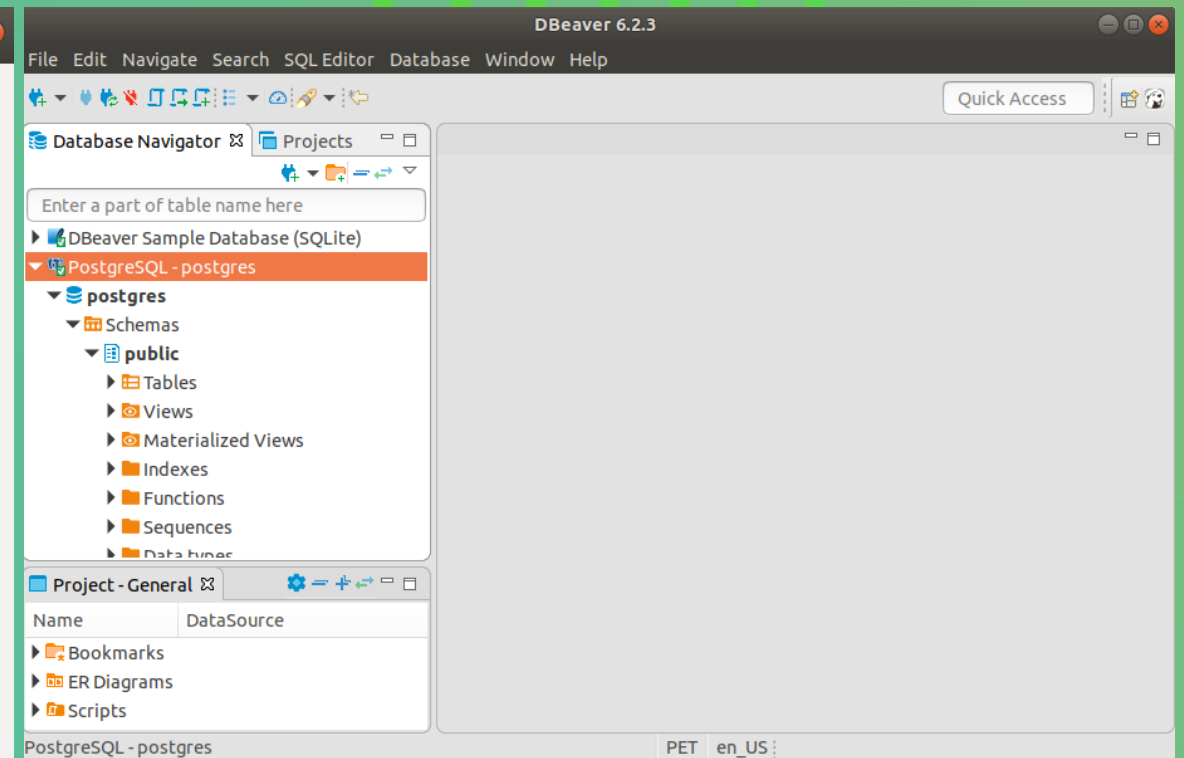
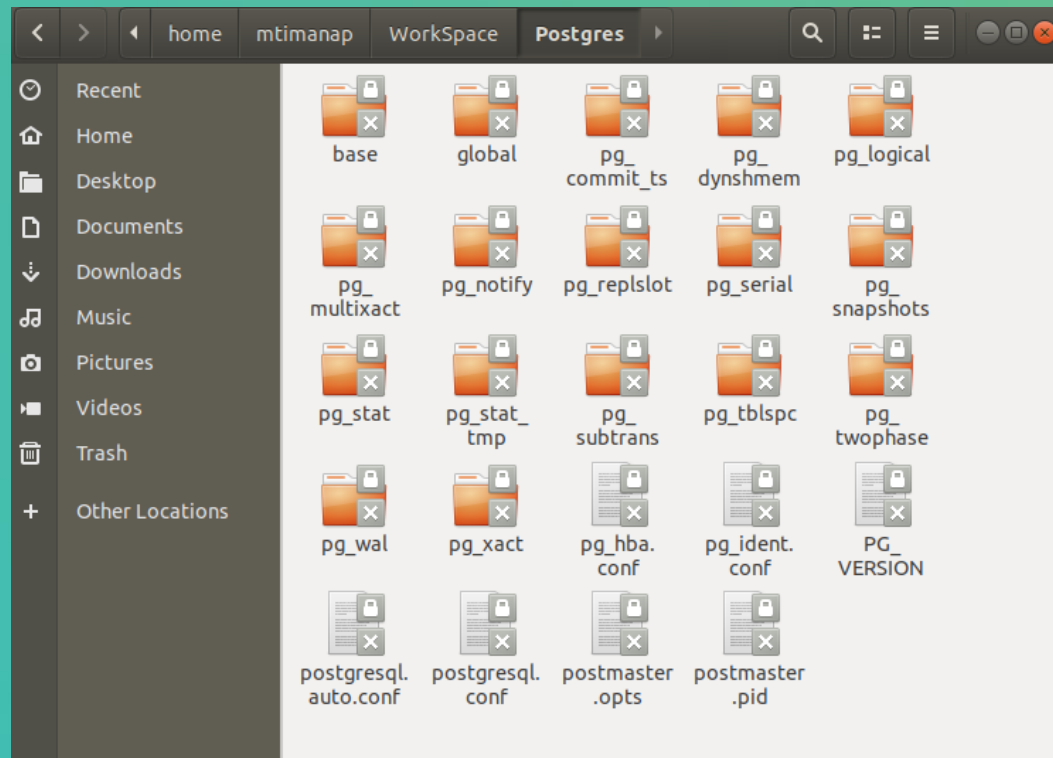
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

03 Contenedores



#Crear container con puerto y volumen en Postgres.

```
mtimanap@laptop:/#docker create -v /home/mtimanap/WorkSpace/Postgres:/Postgres -e PGDATA=/Postgres -e POSTGRES_PASSWORD=123456 -p 5432:5432 postgres
```



03 Contenedores



#Crear container con puerto y volumen en Postgres otra forma.

```
mtimanap@laptop:/#docker run --rm -v /home/mtimanap/WorkSpace/Postgres:/Postgres -e PGDATA=/Postgres -e POSTGRES_PASSWORD=123456 -p 5432:5432 postgres
```

```
019-11-03 14:02:53.701 UTC [21] LOG: database system was not properly shut down; automatic recovery in progress
```

```
2019-11-03 14:02:53.719 UTC [21] LOG: redo starts at 0/1645460
```

```
2019-11-03 14:02:53.720 UTC [21] LOG: invalid record length at 0/1645548: wanted 24, got 0
```

```
2019-11-03 14:02:53.720 UTC [21] LOG: redo done at 0/1645510
```

```
2019-11-03 14:02:53.764 UTC [1] LOG: database system is ready to accept connections
```

#Para ver los container

```
mtimanap@laptop:/#docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
60bc5159bf20	postgres	"docker-entrypoint.s..."	10 minutes ago	Up 10 minutes	0.0.0.0:5432->5432/tcp	quirky_kilby

```
mtimanap@laptop:/#docker run -d --rm -v /home/mtimanap/WorkSpace/Postgres:/Postgres -e PGDATA=/Postgres -e POSTGRES_PASSWORD=123456 -p 5432:5432 postgres
```

```
b49a6f6a9633ab813a70266f4bfe3f41c9d88a6f855b196d3e8005c6d1a81ad3
```

#Para ingresar al container

```
mtimanap@laptop:/#docker exec -it b49a6f6a9633 /bin/bash
```

```
root@b49a6f6a9633:/#psql -U postgres
```



03 Contenedores

#Copiar archivos entre contenedore y directorios local

```
mtimanap@laptop:/#docker run -d --rm -p 8080:8080 -p 9990:9990 jboss/wildfly
mtimanap@laptop:/#docker cp d545ca445eec:/opt/jboss/wildfly/standalone/log/server.log .
mtimanap@laptop:/#docker cp config.xml d545ca445eec:/opt
```

#Para ver las redes en docker

```
mtimanap@laptop:/#docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
44135ba8ca71	bridge	bridge	local
085b8632cc7f	host	host	local
c7c21d875854	none	null	local

#Iniciamos nuestra base de datos

```
mtimanap@laptop:/#docker run -d --rm -v /home/mtimanap/WorkSpace/Postgres:/Postgres -e PGDATA=/Postgres -e POSTGRES_PASSWORD=123456 -p 5432:5432 --name postgres postgres:latest
```

#Iniciamos nuestro servidor de aplicaciones: [link Legacy](#)

```
mtimanap@laptop:/#docker run -d --rm -p 8080:8080 --name jboss --link postgres jboss/wildfly
mtimanap@laptop:/#docker inspect postgres
mtimanap@laptop:/#docker inspect jboss
```

03 Contenedores



#Vemos los container que tenemos

```
mtimanap@laptop:/# docker ps -a
```

CONTAINER ID	IMAGE	CREATED	STATUS	PORTS	NAMES
3c63933963c3	jboss/wildfly	44 seconds ago	Up 43 seconds	0.0.0.0:8080->8080/tcp	jboss
b8baa3f47499	postgres:latest	16 minutes ago	Up 16 minutes	0.0.0.0:5432->5432/tcp	postgres

```
mtimanap@laptop:/# docker exec -it 3c63933963c3 /bin/bash
```

```
[jboss@3c63933963c3 ~]$ ping postgres
```

```
PING postgres (172.17.0.2) 56(84) bytes of data.
```

```
64 bytes from postgres (172.17.0.2): icmp_seq=1 ttl=64 time=0.064 ms
```

```
64 bytes from postgres (172.17.0.2): icmp_seq=2 ttl=64 time=0.102 ms
```

```
64 bytes from postgres (172.17.0.2): icmp_seq=3 ttl=64 time=0.101 ms
```

```
mtimanap@laptop:/# docker exec -it b8baa3f47499 /bin/bash
```

```
root@b8baa3f47499:/# ping jboss
```

```
bash: ping: command not found
```

```
#
```




03 Contenedores

#Crear una red

```
mtimanap@laptop:/#docker network create mired  
ecd48b83bf31787a4d67d60d4116ba32b2b4b3c7b038c4525aec3f94eab15219  
mtimanap@laptop:/#docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
44135ba8ca71	bridge	bridge	local
085b8632cc7f	host	host	local
ecd48b83bf31	mired	bridge	local
c7c21d875854	none	null	local

```
mtimanap@laptop:/#docker run --net=mired -d --rm -v /home/mtimanap/WorkSpace/Postgres:/Postgres -e  
PGDATA=/Postgres -e POSTGRES_PASSWORD=123456 -p 5432:5432 --name postgres postgres:latest
```

```
mtimanap@laptop:/#docker run --net=mired -d --rm -p 8080:8080 --name jboss jboss/wildfly
```

```
mtimanap@laptop:/#docker ps -a
```

CONTAINER ID	IMAGE	CREATED	STATUS	PORTS	NAME
15055fc90d6f	jboss/wildfly	About a minute ago	Up About a minute	0.0.0.0:8080->8080/tcp	jboss
e753d3e7510f	postgres:latest	2 minutes ago	Up 2 minutes	0.0.0.0:5432->5432/tcp	postgres

```
#docker exec -ti jboss ping postgres
```

03 Contenedores



#Hacemos ping desde jboss a postgres: `docker exec -ti [CONTAINER NAME A] ping [CONTAINER NAME B]`

```
mtimanap@laptop:/#docker exec -ti jboss ping postgres
```

```
PING postgres (172.18.0.2) 56(84) bytes of data.
```

```
64 bytes from postgres.mired (172.18.0.2): icmp_seq=1 ttl=64 time=0.056 ms
```

```
64 bytes from postgres.mired (172.18.0.2): icmp_seq=2 ttl=64 time=0.096 ms
```

```
mtimanap@laptop:/#docker exec -it 15055fc90d6f /bin/bash
```

```
[jboss@15055fc90d6f ~]$ ping postgres
```

```
PING postgres (172.18.0.2) 56(84) bytes of data.
```

```
64 bytes from postgres.mired (172.18.0.2): icmp_seq=1 ttl=64 time=0.045 ms
```

```
64 bytes from postgres.mired (172.18.0.2): icmp_seq=2 ttl=64 time=0.096 ms
```

Para más detalles:

<https://docs.docker.com/network/>

03 Contenedores



#Crear mi propio Container Registry

```
mtimanap@laptop:/#docker run -d -p 5000:5000 --restart=always --name registry -v /home/mtimanap/WorkSpace/RegistryServer:/var/lib/registry registry:2  
mtimanap@laptop:/#docker tag postgres:latest localhost:5000/postgres:latest  
mtimanap@laptop:/#docker push localhost:5000/postgres:latest
```

#Despues los clientes ya pueden descargar las imagenes desde nuestro servidor

```
mtimanap@laptop:/#docker push localhost:5000/postgres:latest
```

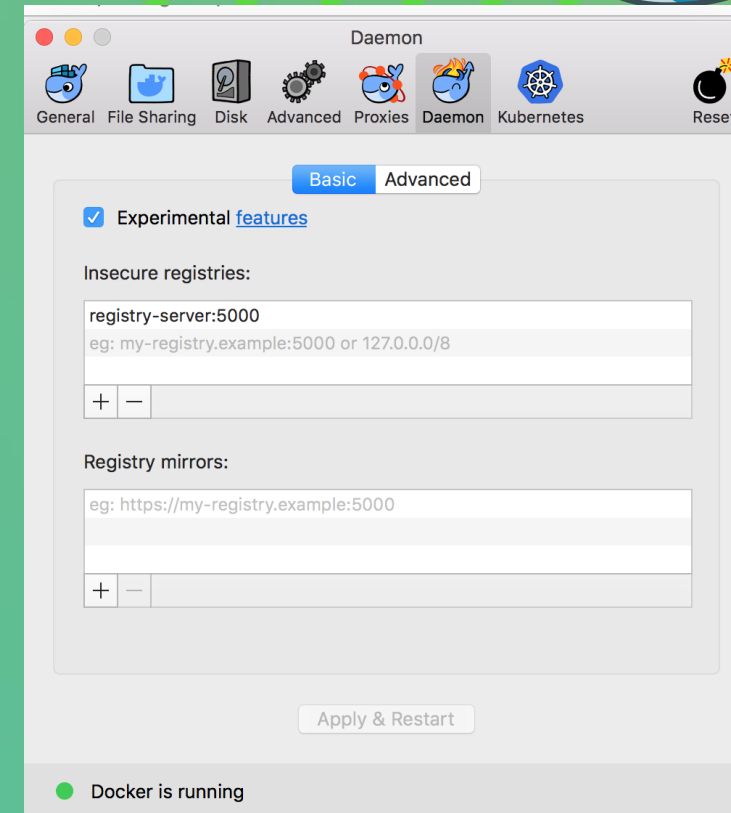
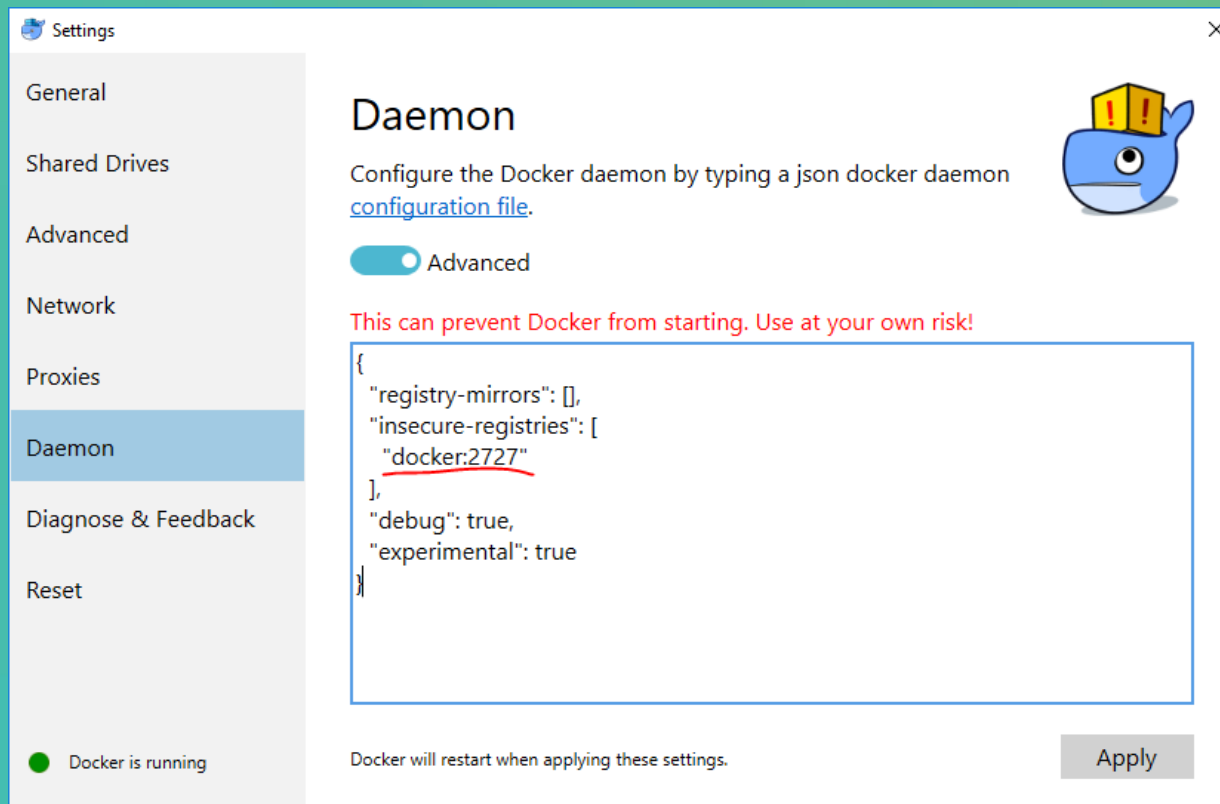
#Creamos or modificamos /etc/docker/daemon.json en el cliente

```
{ "insecure-registries":["myregistry.example.com:5000"] }  
reiniciamos docker daemon  
mtimanap@laptop:/#sudo /etc/init.d/docker restart
```

Para mayor información:

<https://docs.docker.com/registry/deploying/>

03 Contenedores



Para mayor información:

<https://docs.docker.com/registry/deploying/>



04 Docker-compose

Es una herramienta para definir y ejecutar aplicaciones Docker de contenedores múltiples. Con Compose, utiliza un archivo YAML para configurar los servicios de su aplicación. Luego, con un solo comando, crea e inicia todos los servicios desde su configuración.

Las características de Compose que lo hacen efectivo son:

- Múltiples entornos aislados en un solo host

- Conservar datos de volumen cuando se crean contenedores

- Solo recrear contenedores que han cambiado

- Variables y movimiento de una composición entre entornos.

```
mtimanap@laptop:/#sudo su
```

```
root@laptop:/#curl -L "https://github.com/docker/compose/releases/download/1.24.1/docker-  
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
root@laptop:/#chmod +x /usr/local/bin/docker-compose
```

```
mtimanap@laptop:/# docker-compose --version
```

```
docker-compose version 1.24.1, build 4667896b
```



04 Docker-compose

```
#Crear archivo docker-compose.yml
version: "3.7"
services:
```

```
...
```

```
volumes:
```

```
...
```

```
networks:
```

```
...
```

Services :

una aplicación web acoplada que consta de un front-end, un back-end y una base de datos:

```
#
```

```
mtimanap@laptop:/#docker-compose up
```



04 Docker-compose

```
#Crear archivo docker-compose.yml
version: "3"
services:
  servidorApp:
    image: jboss/wildfly:latest
    ports:
      - "8080:8080"
  baseDeDatos:
    image: postgres:latest
    ports:
      - "5432:5432"
#docker-compose up
```



04 Docker-compose

#Para ejecutar y en modo background

```
mtimanap@laptop:/#docker-compose up -d
```

#Para ejecutar con archivo especifico

```
mtimanap@laptop:/#docker-compose -f docker-compose-jboss-postgres1.yml up
```

#Iniciamos con archivo especifico y en background

```
mtimanap@laptop:/#docker-compose start
```

#Para detener

```
mtimanap@laptop:/#docker-compose stop
```

#Para eliminar

```
mtimanap@laptop:/#docker-compose down
```

#Para escalar servicios

```
mtimanap@laptop:/#docker-compose up --scale servidorApp=3 -d
```

#Para reiniciar

```
mtimanap@laptop:/#docker-compose restart
```

#Para ver el log

```
mtimanap@laptop:/#docker-compose logs -f
```

04 Docker-compose



#Para subir mi imagen a docker hub

```
mtimanap@laptop:/#docker-compose -f docker-compose-jboss-postgres2.yml up --build
```

#Para construir imagenes

```
docker-compose -f docker-compose-jboss-postgres2.yml -build
```

#Para construir images y ejecutarlo en background

```
docker-compose -f docker-compose-jboss-postgres2.yml up --build -d
```

#Para scalar servicios

```
docker-compose -f docker-compose-jboss-postgres2.yml up --scale servidorApp=3 -d
```

Para mas información

<https://docs.docker.com/compose/reference/overview/>

05 Docker swarm



Es un grupo de máquinas físicas o virtuales que ejecutan la aplicación Docker y que se han configurado para unirse en un clúster. Una vez que un grupo de máquinas se ha agrupado, aún puede ejecutar los comandos Docker a los que está acostumbrado, pero ahora serán ejecutados por las máquinas en su clúster. Las actividades del clúster están controladas por un swarm manager, y las máquinas que se han unido al clúster se denominan nodos.

Docker Swarm es una herramienta de orquestación de contenedores, lo que significa que le permite al usuario administrar múltiples contenedores desplegados en múltiples máquinas host.

05 Docker swarm

```
docker swarm init --advertise-addr eth0
```



06 Casos practicos

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

<https://docs.docker.com/develop/sdk/examples/>

<https://docs.docker.com/engine/api/v1.24/>

```
curl --unix-socket /var/run/docker.sock http://v1.24/containers/json
```


Referencias

- ✓ <https://www.docker.com/resources/what-container>
- ✓ <https://hub.docker.com/u/ibmcom>
- ✓ <https://docs.docker.com/reference/>
- ✓ <https://docs.docker.com/network/>
- ✓ <https://docs.docker.com>

A long-exposure photograph of the Golden Gate Bridge at night. The bridge's iconic orange-red towers and suspension cables are illuminated, with light trails from vehicles creating a sense of motion. The San Francisco city lights and the bay are visible in the background under a dark sky. In the top right corner, there are several horizontal teal lines of varying lengths.

Thanks!

Looking forward to work with you

April 13, 2020

everis

an NTT DATA Company