Contenedores: Iniciación a Docker y casos de uso prácticos

Alvaro Racero & Alejandro Som
SREs

Ebury

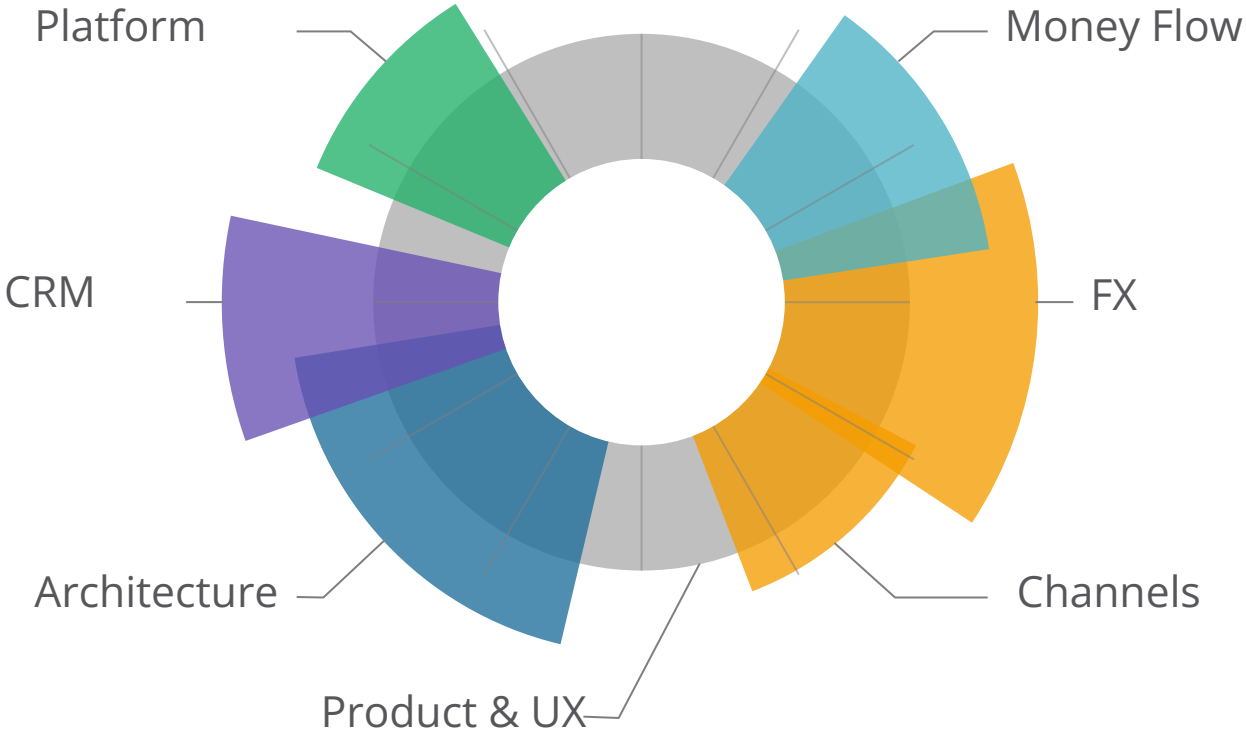
Proporcionamos soluciones para transacciones con divisas, ayudando a las industrias, eliminando las barreras comerciales globales a las que se enfrentan muchas organizaciones.

Ofrecemos:

- **Soluciones para cobros**: Recibir fondos de cualquier parte del mundo de forma fácil. Cambia tu saldo de una divisa a otra o utilízalo para realizar pagos (nacionales o internacionales) en más de 35 divisas sin necesidad de tener varias cuentas bancarias o presencia local.
- **Cuentas en divisa**: Cuentas en la divisa y pais que necesitas, utilizando una cuenta a tu nombre.
- **Transacciones en divisa**: Pagos en más de 100 divisas garantizando que lleguen a tiempo
- **Ebury Online**: Soluciones para gestionar y hacer tu negocio de forma ágil y sencilla incluyendo pagos y cobros en divisa

Ebury Tech. ¿Quienes somos?

Distribución de equipos | Áreas



Ebury Tech. ¿Quienes somos?

Distribución de equipos | Geográfica

America

Brasil
Argentina
Colombia

Europa

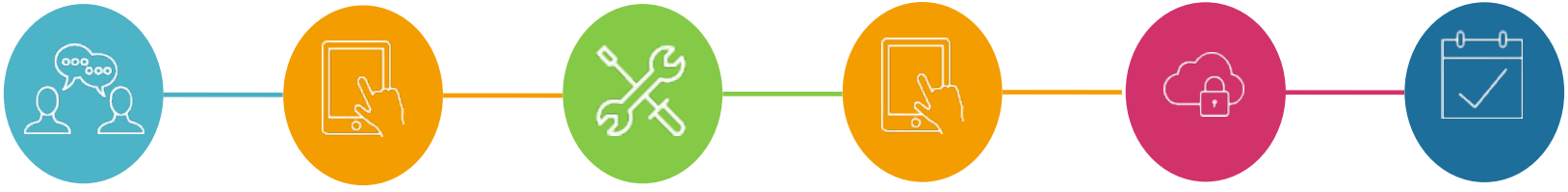
España
Letonia
Hungría
Reino Unido
Portugal
Holanda
Polonia
Turquía
Ucrania
Croacia
Bulgaria
Finlandia
Francia
Austria
Bosnia
Rumanía
Bielorrusia

Ebury Tech. ¿Cómo lo hacemos?

Planificación de Proyectos

Discusión de Requerimientos

Contexto, estimación y acuerdo de solución final



Propuesta de Épica

Tamaño de Camiseta

Opciones de implementación a alto nivel

Planificación de Épica

Quarter

RFC
Estimación 3 puntos

Épica lista para Revisión

Reunión de Revisión

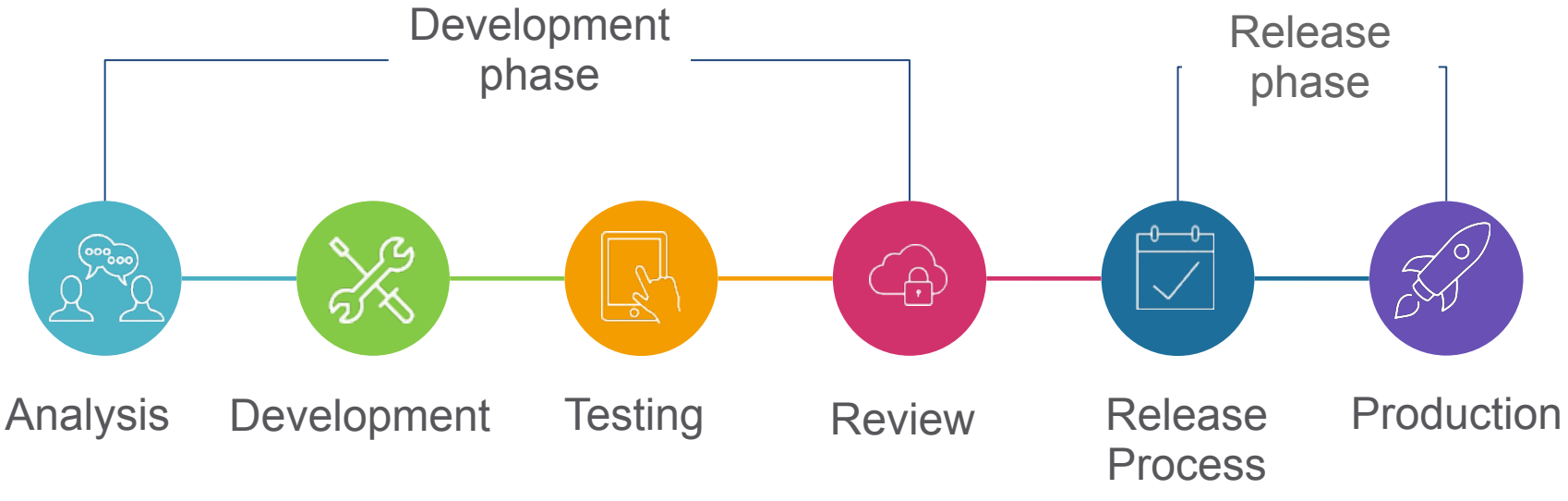
Épica Aprobada

Reporte de Iteración

Plan de Producto

Ebury Tech. ¿Cómo lo hacemos?

Software Development Life Cycle (SDLC)



Ebury Tech. ¿Cómo lo hacemos?

El día día

Ebury Tech. Apuesta por la nube

Alta disponibilidad

Configuraciones Activo-Activo-Activo frente a la clásica Activo-Pasivo.

Niveles superiores al 99,98% global.

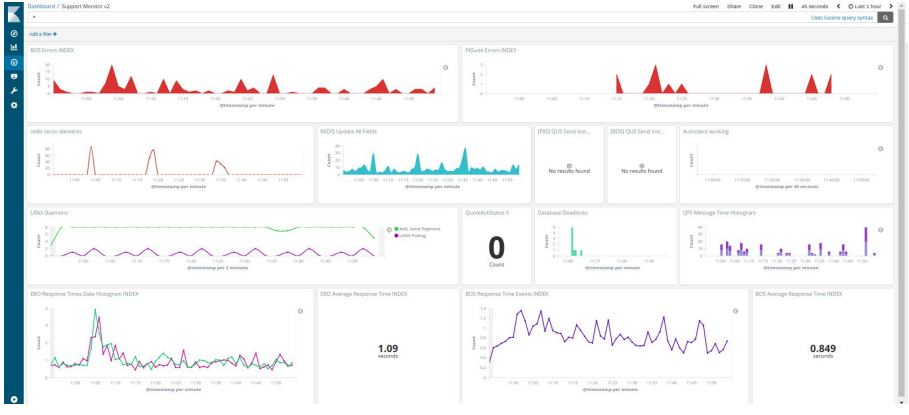
Herramientas visuales de monitorización.

Auto escalabilidad

Sistemas auto escalables que se auto-regulan para adaptarse a los picos de demanda.

Eficiencia y automatización

Pago por uso con alta automatización para encendido y apagado de sistemas.



amazon web services

Google Cloud Platform

salesforce

JIRA

Parte 1 - Teoría sobre contenedores.

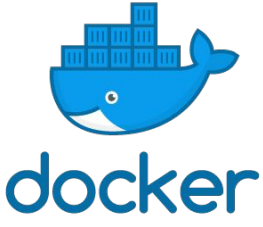
01	Introducción a los contenedores: ¿Qué son?
02	Diferencias entre MV y contenedores
03	Puntos fuertes y débiles del uso de contenedores
04	Introducción a Docker

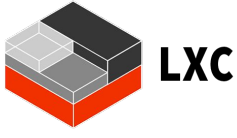
01 Contenedores

¿Qué son?

Contenedores

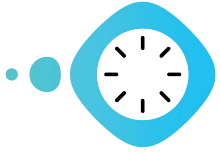
Los contenedores son un paquete de elementos que permiten ejecutar una aplicación determinada en cualquier sistema operativo.





- Docker
- Linux-VServer
- LXC
- LXD
- OpenVZ
- Systemd-nspawn

Contenedores

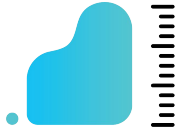


Vale, pero ¿Qué son?

Los contenedores son entornos de ejecución contenidos en sí mismos. Es decir, contienen todo lo necesario para que una aplicación funcione en otros entornos.

Virtualización

Tienen sus propios recursos de CPU, memoria, bloqueo de E/S y de redes aislados del sistema, pero comparten el Kernel del sistema operativo en el que se ejecutan.



Aislamiento

Usan namespaces y cgroups para el aislamiento de los procesos.

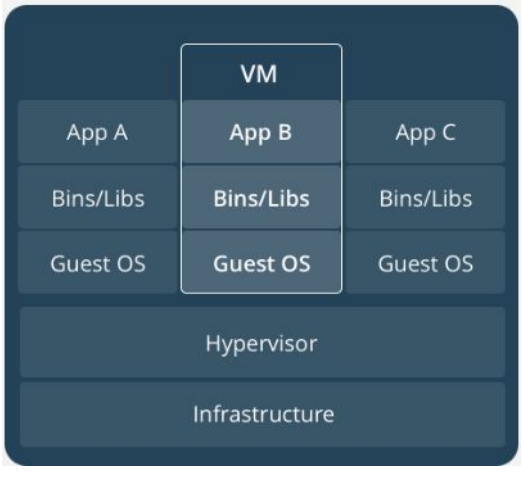
02 Diferencias entre MV y contenedores

Diferencias entre máquinas virtuales y contenedores

Máquinas virtuales

Sistemas operativos diferentes.

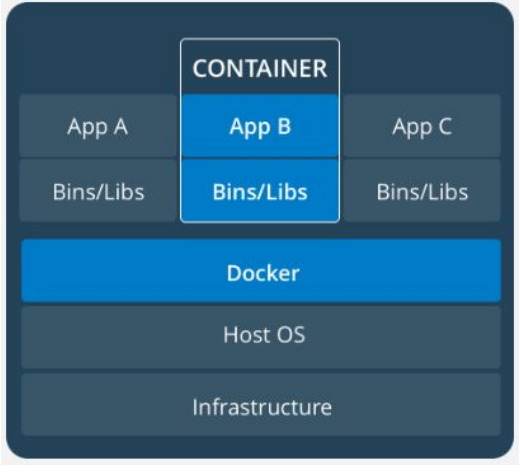
Cada máquina virtual tiene su propio sistema operativo.



Contenedores

Kernel compartido.

Los contenedores comparten el kernel entre ellos, que solo contienen los binarios y las librerías necesarias para su funcionamiento.



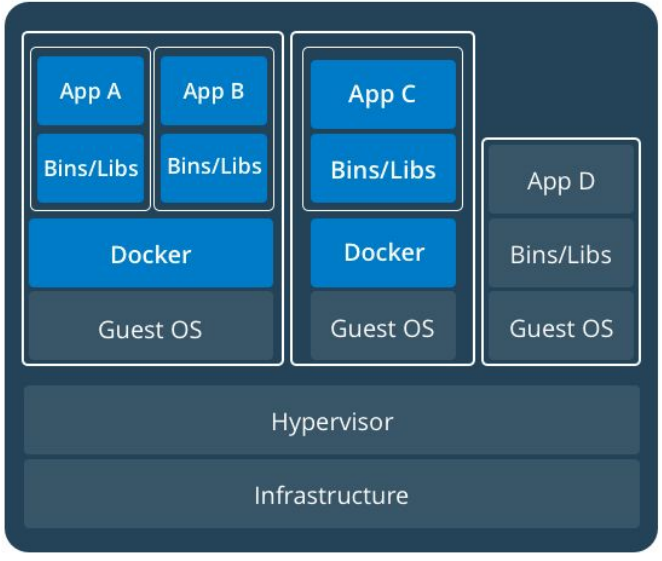
Diferencias entre máquinas virtuales y contenedores

Diferencias entre máquinas virtuales y contenedores

Uso combinado

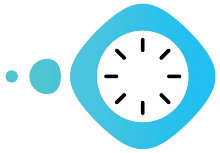
Máquinas virtuales y Docker

Escenario híbrido.



03 Puntos fuertes y débiles del uso de contenedores

Puntos fuertes del uso de contenedores

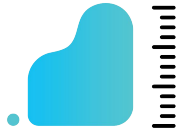


Ligereza

Un contenedor, por lo general, pesa muy poco. (E.J: nginx:alpine son solo 15MB)

Repositorios y Repetibilidad

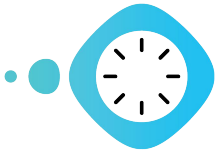
Facilidad para compartir contenedores en repositorios y moverlos entre entornos de desarrollo. Se crean a través de un archivo *Dockerfile*.



Orquestación

Junto a los contenedores, existen multitud de herramientas para su monitorización, despliegue, auto escalado, etc.

Puntos fuertes del uso de contenedores



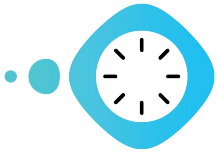
Entorno local de desarrollo

Posibilidad de levantar entornos con facilidad, con contenedores trabajando entre ellos.

Entornos de ejecución de pruebas

Entornos dinámicos con autoescalado para ejecución de pruebas unitarias o de integración.

Puntos débiles del uso de contenedores



Dificultad

Migrar aplicaciones enteras a Docker puede suponer un incremento de la complejidad de mantenimiento.

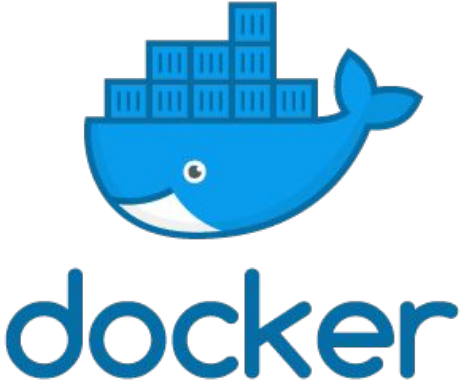
Seguridad

Compartir kernel da la posibilidad de que una vulnerabilidad en el mismo afecte a todos los contenedores.

05 Introducción a Docker

Introducción a Docker

Docker es la implementación de contenedores de software más popular.



- Images
- Dockerfiles
- Containers
- Registry
- Volumes

Conceptos básicos de Docker

01 **Images**
Una imagen es un archivo inerte e inmutable que es básicamente un snapshot de un contenedor. Similar al concepto de *ISO*. Versiones.

02 **Dockerfiles**
Un Dockerfile es el archivo en el que definimos las instrucciones necesarias para crear una imagen de Docker.

03 **Containers**
Un contenedor es una imagen de docker en funcionamiento.

04 **Registry**
El docker registry es un repositorio donde se suben las imágenes de docker. Docker nos ofrece el repositorio DockerHub para ser usado gratuitamente.

05 **Volumes**
El almacenamiento es efímero y cuando el contenedor se pare los datos internos desaparecen. Docker permite montar volúmenes que persisten más allá de la vida del contenedor.

Parte 2 - Práctica

01	Docker cli: Docker run
02	Dockerfile: ¿Qué es? - Creación Dockerfile
03	Docker build
04	Docker cli: Docker run
05	Docker Hub
06	Docker push
07	Docker run from Docker Hub

01 Docker cli: docker run

Docker-cli

Línea de comandos

Docker cli es el software que nos permite ejecutar comandos con Docker.

Docker container run

Lanzar contenedores

Con docker container run podemos desplegar contenedores.
https://docs.docker.com/engine/reference/run/

```
$ docker
Usage: docker [OPTIONS] COMMAND [ARG...]
       docker [ --help | -v | --version ]

A self-sufficient runtime for containers.

Options:
      --config string      Location of client config files (default "/root/.docker")
  -D, --debug              Enable debug mode
      --help               Print usage
  -H, --host value         Daemon socket(s) to connect to (default [])
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/root/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/root/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/root/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Commands:
    attach    Attach to a running container
    # [...]
```

01 Introducción a docker: Parte práctica

- Lanzad este comando:

```
docker container run -p 80:80 nginx:alpine
```

- docker container run: lanza el contenedor
- -p 80:80: mapea el puerto 80 del contenedor con el del host
- nginx:alpine: el nombre de la imagen a lanzar

- Ir a: http://localhost

01 Introducción a docker: Parte práctica

01 Introducción a docker: Parte práctica: Ejercicio 1

- ¿Cómo lanzar un contenedor en segundo plano?
- ¿Cómo ver los contenedores que se están ejecutando?
- ¿Cómo ver logs de un contenedor en segundo plano?
- Lanzad tres contenedores de nginx en diferentes puertos.

01 Introducción a docker: Parte práctica: Ejercicio 1

- ¿Cómo lanzar un contenedor en segundo plano?
  - docker container run -d ...
  - docker run -d ...

- ¿Como ver los contenedores que se están ejecutando?
  - docker container ls
  - docker ps

- ¿Como ver logs de un contenedor en segundo plano?
  - docker container logs “id del contenedor”
  - docker logs “id del contenedor”

- Lanzad tres contenedores de nginx en diferentes puertos
  - docker container run -d -p 80:80 nginx:alpine
  - docker container run -d -p 81:80 nginx:alpine
  - docker container run -d -p 82:80 nginx:alpine

02 Dockerfile: Qué es y creación

Un Dockerfile es un fichero que nos permite crear imágenes de docker a nuestro gusto, basadas en otras imágenes.

https://docs.docker.com/engine/reference/builder/

Dockerfile conceptos:

- **FROM**: la primera instrucción de un Dockerfile indica cual es la imagen base que utilizamos para nuestro Dockerfile.
- **ADD**: añade ficheros al contenedor durante su construcción.
- **RUN**: ejecuta comandos dentro del contenedor.

```
FROM debian:stretch-slim

ENV NGINX_VERSION 1.13.6-1~stretch

RUN apt-get update

EXPOSE 80 443

CMD ["nginx", "-g", "daemon off;"]
```

02 Dockerfile: Creando nuestro Dockerfile

- Creemos un contenedor de nginx:alpine pero desplegando nuestra web. Necesitamos crear un archivo que se llame “Dockerfile”.

```
FROM nginx:alpine

ADD index.html /usr/share/nginx/html/
```

- Necesitamos crear (en el mismo directorio) el archivo index.html. Este archivo puede contener cualquier código html.

03 Docker build

Docker cli: docker image build

Para crear imágenes a partir del dockerfile

Para poder construir nuestras imágenes, necesitamos ejecutar el comando “docker image build”. La sintaxis es la siguiente:

 # docker image build [-t nombre-de-la-imagen] [path contexto del dockerfile]

Ejemplo:

```
docker image build -t alvaro-racero-uma:latest .
```

03 Docker build

```
Sending build context to Docker daemon  3.072kB
Step 1/2 : FROM nginx:alpine
 ---> 315798907716
Step 2/2 : ADD index.html /usr/share/nginx/html/
 ---> da6a99d9ef58
Successfully built da6a99d9ef58
Successfully tagged alvaro-racero-uma:latest
```

02 Dockerfile: Ejercicio Dockerfile - 2

- Instrucciones:

Queremos lanzar un contenedor con una aplicación propia. Esta aplicación no es más que un script en sh con un echo (ver captura abajo).

El contenedor tomará de imagen base alpine:latest, copiará nuestra aplicación a /bin/nombredelaaplicación y luego declarará el comando a ejecutar en el contenedor. (Instrucción CMD de Dockerfile).

```
#!/bin/sh

echo "Hola Alvaro Racero"
```

04 Docker cli : docker run - nociones avanzadas

- Desde docker run siempre podemos sobreescribir el comando especificado en el Dockerfile.

```
alvaroracero@MLGPC159:~/uma/echo$ docker run alvaro-racero-echo echo "Test para UMA"
Test para UMA
```

- También podemos decir que el contenedor se ejecute de forma interactiva y conectado a nuestro terminal con las opciones “-ti”.

```
alvaroracero@MLGPC159:~/uma/echo$ docker run -it alvaro-racero-echo /bin/sh
/ #
```

02 Dockerfile: Ejercicio Dockerfile - 3

- Instrucciones:

Queremos lanzar un contenedor con con el comando “curl” instalado. Este contenedor estará basado en Ubuntu. Después, queremos poder ejecutar este contenedor para lanzar comandos “curl” a distintas webs.

Pista *“CMD” no nos vale para esto.*

FROM xxxxx

ENV http_proxy "http://proxy.laboratorios.ac.uma.es:3128"

ENV https_proxy "http://proxy.laboratorios.ac.uma.es:3128"

XXX

04 Docker Hub

Docker Hub

Docker Hub es un docker registry público donde subir nuestras imágenes

Docker hub nos permite registrarnos gratuitamente y subir nuestros contenedores de forma pública.

Exploración de repositorios oficiales.

Crear una cuenta en https://hub.docker.com/

01

Step 1

Registro en Docker Hub y creación de nuestro primer repositorio PÚBLICO.

02

Step 2

Uso del comando docker login para loguear nuestra pc en ese repositorio.

03

Step 3

Cambiar el tag de nuestro contenedor para ajustarlo al repositorio recién creado y subir el contenedor a Docker Hub con “docker push”.

04 Docker login & push

- docker login

```
alvaroracero@MLGPC159:~/uma/curl$ 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: alvaroracero
Password:
WARNING! Your password will be stored unencrypted in /home/alvaroracero/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

- docker images and docker tag

```
alvaroracero@MLGPC159:~/uma/echo$ docker tag alvaro-racero-echo:latest alvaroracero/echouma:latest
```

- docker push

```
alvaroracero@MLGPC159:~/uma/echo$ docker push alvaroracero/echouma:latest
The push refers to repository [docker.io/alvaroracero/echouma]
6fc7accc49d3: Pushed
7bff100f35cb: Mounted from library/alpine
latest: digest: sha256:92fdfe179d20de4f80f7cda997ad7d4816d842f98ccee7389b73bd78a7b09bc3 size: 735
```

04 Docker cli: docker run

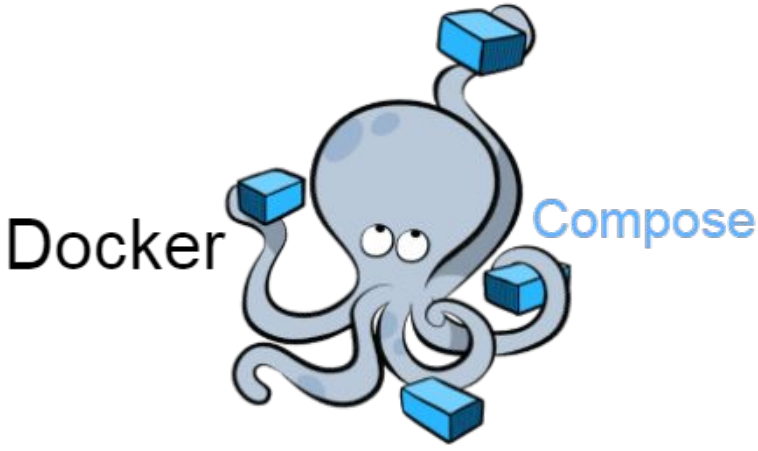
- Lanzad el contenedor de un compañero en vuestro local.

Parte 3 - Contenido avanzado

01	Orquestación

Docker Avanzado: Orquestación: docker-compose

- Docker compose es una herramienta para definir entornos de multicontenedores.
- Nos permite, en un solo archivo, definir varios contenedores y su interacción entre ellos.
- Guia de instalación: https://docs.docker.com/compose/install/
- Manual referencia: https://docs.docker.com/compose/reference/overview/



Docker Avanzado: Orquestación - docker-compose

Creando entornos multicontenedor con docker-compose

docker-compose.yml

- **version**: especificamos la versión a utilizar de docker-compose.
- **services**: definimos los contenedores
  - **nombrequequeramos**: define el nombre del servicio (que crea una entrada interna de DNS).
- Docker-compose up && docker-compose down

```
version: '2'

services:
  web:
    image: drupal:8.2-apache
    ports:
      - '8080:80'
  backend:
    image: myownapp
    build: .
```

Docker Avanzado: Orquestación - docker-compose

Ejercicio 1

- Queremos crear dos contenedores que interactúen entre ellos. Uno de ellos será un contenedor de wordpress (wordpress:5.0.3). Para el necesitamos hacer forward del puerto 80 del contenedor al 8080 nuestro (por ejemplo). El segundo contenedor será la base de datos de este contenedor. Vamos a utilizar mysql (msql:5.7).
  Pista: Los contenedores necesitan variables de entorno, por ejemplo en mysql MYSQL_ROOT_PASSWORD que especifica la contraseña.

- Ir a http://localhost:8080

Únete al equipo que está cambiando el futuro de las FinTech

Forma parte del hub tecnológico de desarrolladores, QAs y DevOps que está cambiando los servicios financieros

Visita la web **careers.ebury.com**

o manda tu CV a

recruitment.malaga@ebury.com

alvaro.racero@ebury.com

Ebury LABS

Visita el blog **labs.ebury.rocks**

Y nuestro Twitter **@EburyLabs**

¿Preguntas?