

# Contenerizando microservicios (Docker)

Bootcamp Java Microservicios

## Contenido

1.	Herramientas Necesarias .....	2
2.	Caso de Uso .....	2
3.	Instalando Dockers .....	2
4.	Docker Workflow .....	2
5.	Comandos Comunes en Docker .....	2
6.	Docker Hub .....	3
7.	Imágenes preconstruidas en Docker (Docker Pull) .....	3
8.	Creando una red interna (Docker network) .....	4
9.	Ejecutando imágenes preconstruidas en docker (Docker Run) .....	4
10.	Escribiendo nuestras propias imágenes (Dockerfile) .....	4
11.	Ejecutando imágenes propias en docker (Docker Run) .....	4
12.	Subiendo nuestras imágenes a repositorio (Docker Build) .....	5
13.	Docker Compose .....	6
14.	Enlaces de Interés .....	6

## 1. Herramientas Necesarias

- ✓ JDK Java 11
- ✓ IDE Java (IntelliJ IDEA 2022.1.2 Community, Spring Tool Suite).
- ✓ Microservicio desarrollado previamente.

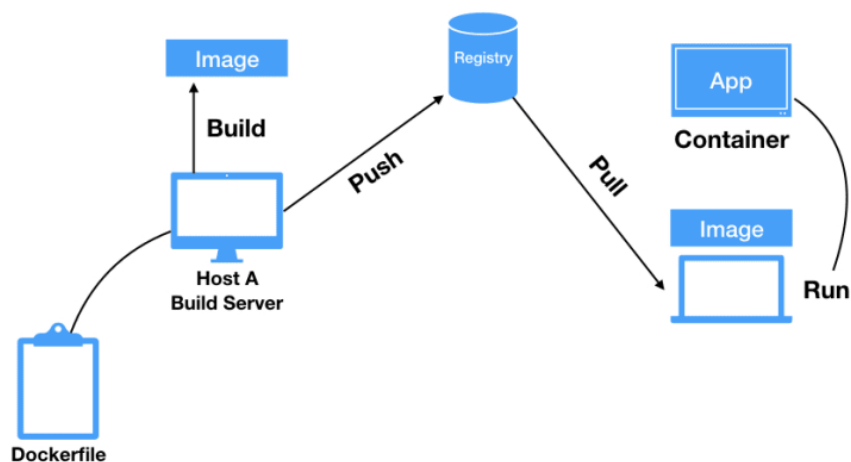
## 2. Caso de Uso

Se utilizará el mismo caso de uso del proyecto semanal

## 3. Instalando Dockers

- ✓ Instalar WSL desde PowerShell
- ✓ Reiniciar el equipo
- ✓ Crear usuario/password en Ubuntu.
- ✓ Ir a la página <https://www.docker.com/> y descargar Docker Desktop para Windows
- ✓ Referencia: <https://youtu.be/ZO4KWQfUBBc>

## 4. Docker Workflow



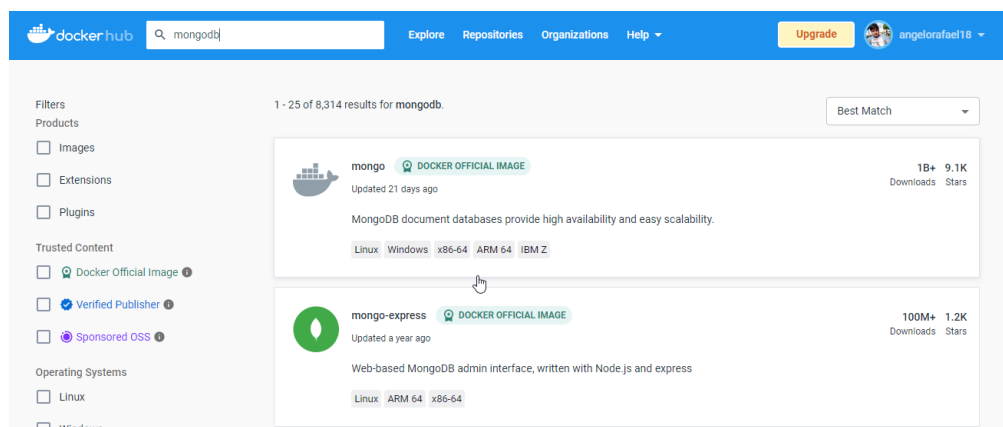
## 5. Comandos Comunes en Docker

- ✓ En la siguiente dirección <https://dockerlabs.collabnix.com/docker/cheatsheet/> encontrarán la mayoría de los comandos comunes más utilizados en Docker.

Cheatsheet for Docker CLI			
Run a new Container	Manage Containers	Manage Images	Info & Stats
Start a new Container from an Image <code>docker run IMAGE</code> <code>docker run nginx</code> ...and assign it a name <code>docker run --name CONTAINER IMAGE</code> <code>docker run --name web nginx</code> ...and map a port <code>docker run -p HOSTPORT:CONTAINERPORT IMAGE</code> <code>docker run -p 8080:80 nginx</code> ...and map all ports <code>docker run -P IMAGE</code> <code>docker run -P nginx</code> ...and start container in background <code>docker run -d IMAGE</code> <code>docker run -d nginx</code> ...and assign it a hostname <code>docker run --hostname HOSTNAME IMAGE</code> <code>docker run --hostname srv nginx</code> ...and add a dns entry <code>docker run --add-host HOSTNAME:IP IMAGE</code> ...and map a local directory into the container <code>docker run -v HOSTDIR:TARGETDIR IMAGE</code> <code>docker run -v ~/usr/share/nginx/html nginx</code> ...but change the entrypoint <code>docker run -it --entrypoint EXECUTABLE IMAGE</code> <code>docker run -it --entrypoint bash nginx</code>	Show a list of running containers <code>docker ps</code> Show a list of all containers <code>docker ps -a</code> Delete a container <code>docker rm CONTAINER</code> <code>docker rm web</code> Delete a running container <code>docker rm -f CONTAINER</code> <code>docker rm -f web</code> Delete stopped containers <code>docker container prune</code> Stop a running container <code>docker stop CONTAINER</code> <code>docker stop web</code> Start a stopped container <code>docker start CONTAINER</code> <code>docker start web</code> Copy a file from a container to the host <code>docker cp CONTAINER:SOURCE TARGET</code> <code>docker cp web:/index.html index.html</code> Copy a file from the host to a container <code>docker cp TARGET CONTAINER:SOURCE</code> <code>docker cp index.html web:/index.html</code> Start a shell inside a running container <code>docker exec -it CONTAINER EXECUTABLE</code> <code>docker exec -it web bash</code> Rename a container <code>docker rename OLD_NAME NEW_NAME</code> <code>docker rename 096 web</code> Create an image out of container <code>docker commit CONTAINER</code> <code>docker commit web</code>	Download an image <code>docker pull IMAGE[:TAG]</code> <code>docker pull nginx</code> Upload an image to a repository <code>docker push IMAGE</code> <code>docker push myimage:1.0</code> Delete an image <code>docker rmi IMAGE</code> Show a list of all images <code>docker images</code> Delete dangling images <code>docker image prune</code> Delete all unused images <code>docker image prune -a</code> Build an image from a Dockerfile <code>docker build DIRECTORY</code> <code>docker build .</code> Tag an image <code>docker tag IMAGE NEWIMAGE</code> <code>docker tag ubuntu ubuntu:18.04</code> Build and tag an image from a Dockerfile <code>docker build -t IMAGE DIRECTORY</code> <code>docker build -t myimage .</code> Save an image to tar file <code>docker save IMAGE &gt; FILE</code> <code>docker save nginx &gt; nginx.tar</code> Load an image from a tar file <code>docker load -i TARFILE</code> <code>docker load -i nginx.tar</code>	Show the logs of a container <code>docker logs CONTAINER</code> <code>docker logs web</code> Show stats of running containers <code>docker stats</code> Show processes of container <code>docker top CONTAINER</code> <code>docker top web</code> Show installed docker version <code>docker version</code> Get detailed info about an object <code>docker inspect NAME</code> <code>docker inspect nginx</code> Show all modified files in container <code>docker diff CONTAINER</code> <code>docker diff web</code> Show mapped ports of a container <code>docker port CONTAINER</code> <code>docker port web</code>

## 6. Docker Hub

- ✓ Registrarte en la siguiente Web <https://hub.docker.com/>



## 7. Imágenes preconstruidas en Docker (Docker Pull)

- ✓ Ejecutar el comando **docker pull mongo**

```
PS C:\Users\angulom\Downloads> docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
fb0b3276a519: Pull complete
c81bcd64a2d2: Pull complete
45ed91f63dfa: Pull complete
06d1770a2c11: Pull complete
a03d917eab2f: Pull complete
d0226311fde3: Pull complete
cf22b9bccca1: Pull complete
5a4955b612fd: Pull complete
bc8341d9c8d5: Pull complete
Digest: sha256:946d309038b2581d8913213333eb3f86142d95e770ec6a3e334ca9b43ebd402e
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest
```

- ✓ Ejecutar el comando **docker images**

```
PS C:\Users\aaangulom\Downloads> docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
docker101tutorial   latest          60bfe478c4dd   2 hours ago    28.9MB
alpine/git           latest          b80d2cac43e4   11 days ago    43.6MB
mongo                latest          1cca5cf68239   13 days ago    695MB
PS C:\Users\aaangulom\Downloads>
```

- ✓ La imagen se encontrará disponible para ser usada

## 8. Creando una red interna (Docker network)

- ✓ Ejecutar comando **docker network create development-network** esto creará una única red para poder interconectar los diferentes contenedores que crearemos.

```
PS C:\Repositorio\Personal\bootcampjava-adv\customer> docker network create development-network
ebc7e34ae9c32cecb6f7cd27f059025dad5a47d68633d50415df8fa962c8bed9
```

## 9. Ejecutando imágenes preconstruidas en docker (Docker Run)

- ✓ Ejecutar comando **docker run --name mongo-development --network development-network -d -p 27018:27017 mongo** para crear un contenedor a partir de una imagen obtenida.

```
PS C:\Users\aaangulom\Downloads> docker run --name mongo-development -d mongo
c9f393fe0e4f4716e56f103492df36ff4e1b0c8ec001a41984714b484859a4f7
```

- ✓ Ejecutar comando **docker ps** para visualizar el contenedor creado

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
c9f393fe0e4f	mongo	"docker-entrypoint.s..."	12 seconds ago	Up 11 seconds	27017/tcp	mongo-development

## 10. Escribiendo nuestras propias imágenes (Dockerfile)

- ✓ Crear archivo **Dockerfile** en la raíz del proyecto
- ✓ Escribir el siguiente contenido (Cambiar los nombres y puertos por el microservicio a desplegar)

```
FROM openjdk:11
VOLUME /tmp
EXPOSE 9043
ADD ./target/customer-0.0.1-SNAPSHOT.jar ms-customer.jar
ENTRYPOINT ["java", "-jar", "/ms-customer.jar"]
```

## 11. Ejecutando imágenes propias en docker (Docker Run)

- ✓ Navegar a directorio del microservicio (donde creamos el **Dockerfile**)
- ✓ Ejecutar el siguiente comando (cambiar los nombres por el microservicio a desplegar **docker build -t ms-customer .**

```
PS C:\Repositorio\Personal\bootcampjava-adv\customer> docker build -t ms-customer .
[+] Building 45.6s (8/8) FINISHED
```

- ✓ Ejecutar el siguiente comando para visualizar la imagen creada **docker images**

```
PS C:\Repositorio\Personal\bootcampjava-adv\customer> docker images
REPOSITORY          TAG          IMAGE ID        CREATED         SIZE
ms-customer         latest      b6e8abbd6dcd   About a minute ago  698MB
docker101tutorial   latest      60bfe478c4dd   2 hours ago     28.9MB
```

- ✓ Ejecutar el siguiente comando para crear contenedor a partir de imagen creada previamente **docker run --name ms-customer --network development-network -d -p 9043:9043 ms-customer**

```
PS C:\Repositorio\Personal\bootcampjava-adv\customer> docker run --name ms-customer -d -p 9083:9083 ms-customer
0b99a08c59d6bd118ef787d04fd8c8d9225f375dc7b34faa6759f13fd16851aa
```

- ✓ Ejecutar el siguiente comando para visualizar contenedor creado **docker ps**

```
PS C:\Repositorio\Personal\bootcampjava-adv\customer> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
0b99a08c59d6   ms-customer   "java -jar /ms-custo..."   About a minute ago   Up About a minute   0.0.0.0:9083->9083/tcp         ms-customer
9bd946fc426b   mongo        "docker-entrypoint.s..."   19 minutes ago     Up 19 minutes      0.0.0.0:27018->27017/tcp       mongo-development
```

## 12. Subiendo nuestras imágenes a repositorio (Docker Build)

- ✓ Ejecutar el siguiente comando para conectarte a docker hub **docker login**

```
PS C:\Repositorio\Personal\bootcampjava-adv\customer> docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
```

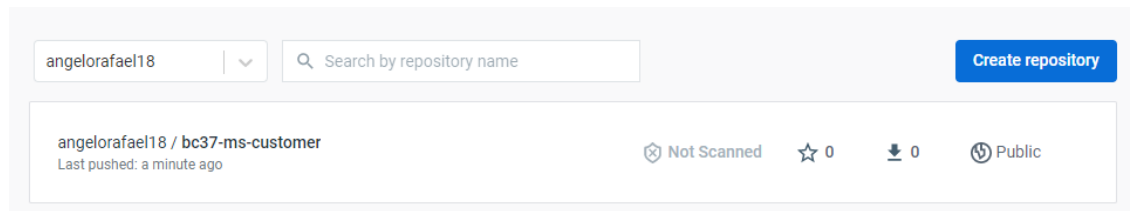
- ✓ Renombraremos la imagen que construimos para añadir el nombre del usuario **docker tag [ID Imagen] angelorafael18/bc37-ms-customer**

```
PS C:\Repositorio\Personal\bootcampjava-adv\customer> docker tag 247ef79f4a02 angelorafael18/bc37-ms-customer
PS C:\Repositorio\Personal\bootcampjava-adv\customer> docker images
REPOSITORY          TAG          IMAGE ID        CREATED         SIZE
<none>              <none>       cd8b8035522f   17 minutes ago  698MB
ms-customer         latest      247ef79f4a02   17 minutes ago  698MB
angelorafael18/bc37-ms-customer   latest      247ef79f4a02   17 minutes ago  698MB
```

- ✓ Ejecutaremos el siguiente comando para revisar que la imagen se haya subido al repositorio **docker push angelorafael18/bc37-ms-customer**

```
PS C:\Repositorio\Personal\bootcampjava-adv\customer> docker push angelorafael18/bc37-ms-customer
Using default tag: latest
The push refers to repository [docker.io/angelorafael18/bc37-ms-customer]
0f618a229b43: Pushed
7b7f3078e1db: Pushed
826c3d8bb29c: Pushed
b626401ef603: Pushed
9b55156abf26: Pushed
293d5db30c9f: Pushed
03127cdb479b: Pushed
9c742cd6c7a5: Pushed
latest: digest: sha256:afa7943080674b8759fb108ebe2ed0582da9b147e9bb8b730859fda5e62e757a size: 2007
PS C:\Repositorio\Personal\bootcampjava-adv\customer>
```

- ✓ Revisar en docker hub



## 13. Docker Compose

- ✓ Crear archivo **docker-compose.yml** en carpeta independiente a microservicios
- ✓ Agregar el siguiente contenido (Modificar los datos por los de los microservicios a desplegar)

```
version: "3.7"
services:
  ms-customer:
    image: angelorafael18/bc37-ms-customer
    ports:
      - 9043:9043

  mongo:
    image: mongo
    ports:
      - 27018:27017
```

- ✓ Ejecutar el comando **docker-compose up -d** el cual realizará lo declarado en el archivo **docker-compose.yml**

```
PS C:\Repositorio\Personal\bootcampjava-adv\docker> docker-compose up -d
[+] Running 2/3
- Container docker-mongo-1      Starting
- Container ms-customer        Recreated
- Container docker-ms-customer-1 Started
```

## 14. Enlaces de Interés

- ✓ <https://youtu.be/FA3VdlBo0nA>
- ✓ [https://youtu.be/CV\\_Uf3Dq-EU](https://youtu.be/CV_Uf3Dq-EU)