

[MDAS] - Principios y herramientas de desarrollo Entregable 1

1. Indica la diferencia entre el uso de la instrucción CMD y ENV (Dockerfile).

ENV:

La variable ENV nos permite setear las variables de entorno dentro de nuestro contenedor.

Utiliza **<key> =<value>**:

Ejemplo:

```
#Define the ENV variable
ENV nginx_vhost /etc/nginx/sites-available/default
ENV php_conf /etc/php/7.4/fpm/php.ini
ENV nginx_conf /etc/nginx/nginx.conf
ENV supervisor_conf /etc/supervisor/supervisord.conf
```

Esto nos facilitara la vida a la hora de interactuar con la consola de la maquina o para lanzar ejecuciones.

CMD:

Nos permite lanzar comandos dentro del cmd de la máquina. Ejecutar instrucciones como si estuviésemos en el terminal de la máquina.

Se utiliza normalmente al final del Dockerfile para lanzar la ejecución después de haberlo configurado.

Ejemplo:

CMD ["executable"," param1"," param2"]: **CMD ["addNumbers.py","2","4"]** Exec form

En este caso nos permite ejecutar el fichero addNumbers pasandole los parámetros 2 ,4.

Diferencia:

ENV nos permite especificarles las variables de entorno dentro de la maquina mientras que CMD nos permite lanzar ejecuciones/comandos como si estuviéramos dentro del CMD del contenedor.

2. Indica la diferencia entre el uso de la instrucción ADD y COPY (Dockerfile).

- **Los dos utilizan la misma lógica:** ADD/COPY <src> ... <dest>
- **COPY:** Toma un SRC y un destino. Sólo te permite copiar en un archivo o directorio local de tu host (la máquina que construye la imagen del Docker) en la propia imagen del Docker.

Host → Docker. Example: COPY /source/file/path /destination/path

- **ADD:** Te permite hacer lo mismo que el COPY pero también soporta otros tipos de fuentes (SRC). Puedes usar una URL en ves de un fichero local/directory. También te permite extraer un .tar file del SRC a la destinación (Docker).

Host → Docker. Example: ADD /source/file/path /destination/path

URL → Docker. Example: ADD http://source.file/url /destination/path

Tar File (identity, gzip, bzip2 or xz) → Docker. Example: ADD source.file.tar.gz /temp

3. Crea un contenedor con las siguientes especificaciones:

a) Instalar nginx:1.19.3 por línea de comandos.

```
test@ubuntu:~/Downloads$ sudo docker pull nginx:1.19.3
1.19.3: Pulling from library/nginx
bb79b6b2107f: Already exists
111447d5894d: Already exists
a95689b8e6cb: Already exists
1a0022e444c2: Already exists
32b7488a3833: Already exists
Digest: sha256:ed7f815851b5299f616220a63edac69a4cc200e7f536a56e421988da82e44ed8
Status: Downloaded newer image for nginx:1.19.3
docker.io/library/nginx:1.19.3
test@ubuntu:~/Downloads$ sudo docker images ps
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
test@ubuntu:~/Downloads$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
→ nginx             1.19.3             f35646e83998       3 weeks ago        133MB
test@ubuntu:~/Downloads$
```

b)

Dockerfile:

```
.Dockerfile
C: > Users > Budy > Desktop > MASTER > Docker1 > .Dockerfile
1 FROM nginx:1.19.3
2
3 VOLUME ./index.html /static_content
4
5 COPY ./index.html /usr/share/nginx/html/index.html
6
```

Index.html:

```
index.html X
C: > Users > Budy > Desktop > MASTER > Docker1 > index.html > ...
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>HOMEWORK 1</h1>
6
7 <p>Principis i eines de desenvolupament [2020/21]</p>
8
9 </body>
10 </html>
```

Comandos terminal:

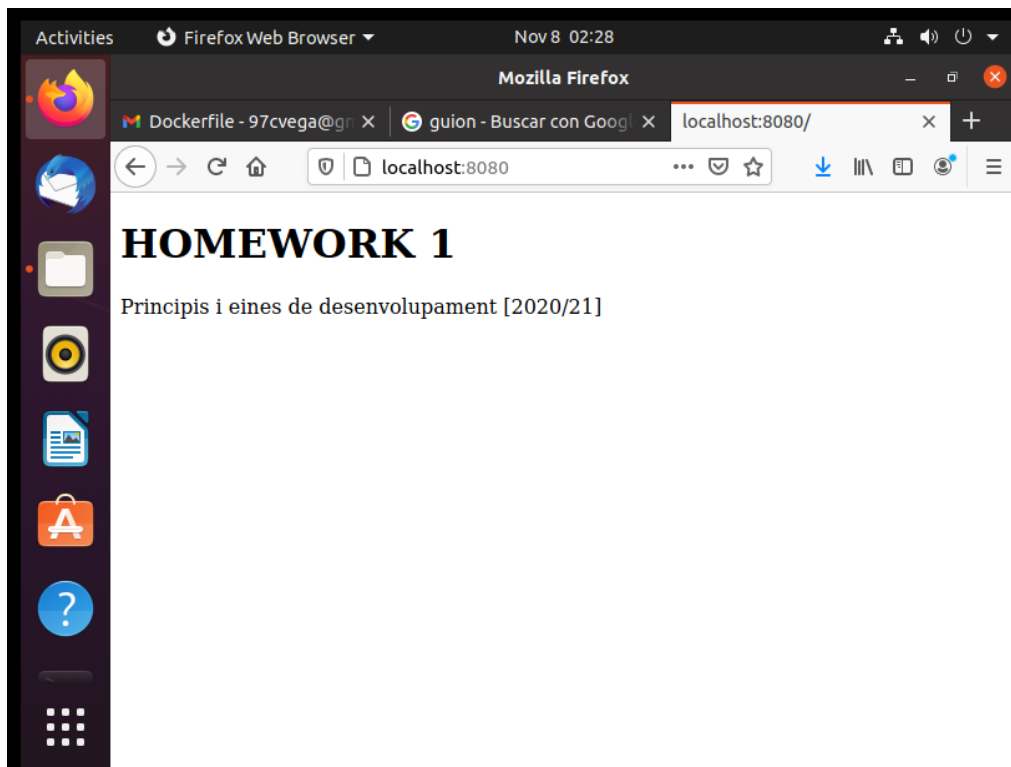
Sudo Docker build -t nginx:homework1 .

```
user@ubuntu:~/Downloads$ sudo docker build -t nginx:homework1 .
Sending build context to Docker daemon 4.096kB
Step 1/3 : FROM nginx:1.19.3
1.19.3: Pulling from library/nginx
bb79b6b2107f: Pull complete
111447d5894d: Pull complete
a95689b8e6cb: Pull complete
1a0022e444c2: Pull complete
32b7488a3833: Pull complete
Digest: sha256:ed7f815851b5299f616220a63edac69a4cc200e7f536a56e421988da82e44ed8
Status: Downloaded newer image for nginx:1.19.3
--> f35646e83998
Step 2/3 : VOLUME ./index.html /static_content
--> Running in 80c0c577656e
Removing intermediate container 80c0c577656e
--> 420e0da5e4fb
Step 3/3 : COPY ./index.html /usr/share/nginx/html/index.html
--> a9f9cb771e20
Successfully built a9f9cb771e20
Successfully tagged nginx:homework1
```

Sudo Docker run -d -p 8080:80 nginx:homework1

```
user@ubuntu:~/Downloads$ sudo docker run -d -p 8080:80 nginx:homework1
4abbcffda3a142bdfac5ee7f6936c5f79a661a10adadb66188865ddbbee93f8
```

Resultado final:



4. Crea una imagen docker a partir de un Dockerfile. Esta aplicación expondrá un servicio en el puerto 8080 y se deberá hacer uso de la instrucción HEALTHCHECK para comprobar si la aplicación está ofreciendo el servicio o por si el contrario existe un problema.

El healthcheck deberá parametrizarse con la siguiente configuración:

- La prueba se realizará cada 45 segundos.

```
HEALTHCHECK --interval=45s
```

- Por cada prueba realizada, se esperará que la aplicación responda en menos de 5 segundos. Si tras 5 segundos no se obtiene respuesta, se considera que la prueba habrá fallado.

```
--timeout=5s
```

- Ajustar el tiempo de espera de la primera prueba (Ejemplo: Si la aplicación del contenedor tarda en iniciarse 10s, configurar el parámetro a 15s).

```
--start-period=15s
```

- El número de reintentos será 2. Si fallan dos pruebas consecutivas, el contenedor deberá cambiar al estado “unhealthy”).

```
--retries=2
```

Full command (Dockerfile):

Creacion de script que nos dice si la pagina esta alive o no:

```
#!/bin/bash
curl -f -s -I "localhost:8080" &>/dev/null && echo 0 || echo 1
exit 0;
```

Healthcheck dockerfile:

```
HEALTHCHECK --interval=45s --timeout=5s --start-period=15s --retries=2 CMD
["sh", "/healthcheck.sh"]
```

Resultado:

```
---> Using cache
---> 021d422c4c4c
Successfully built 021d422c4c4c
Successfully tagged nginx:homework1
test@ubuntu:~/download$ sudo docker run -d -p 8080:80 nginx:homework1
[[C009952a53d730da3b3ba143af381b44387a2275863c6c6e4495d83e41f3b618
test@ubuntu:~/download$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
089952a53d73        nginx:homework1    "/docker-entrypoint..." 3 seconds ago       Up 2 seconds (health: starting)    0.0.0.0:8080->80/tcp    funny_ptoleny
test@ubuntu:~/download$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
089952a53d73        nginx:homework1    "/docker-entrypoint..." 27 seconds ago      Up 26 seconds (health: starting)    0.0.0.0:8080->80/tcp    funny_ptoleny
test@ubuntu:~/download$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
089952a53d73        nginx:homework1    "/docker-entrypoint..." 45 seconds ago      Up 44 seconds (health: starting)    0.0.0.0:8080->80/tcp    funny_ptoleny
test@ubuntu:~/download$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
089952a53d73        nginx:homework1    "/docker-entrypoint..." 46 seconds ago      Up 45 seconds (healthy)            0.0.0.0:8080->80/tcp    funny_ptoleny
test@ubuntu:~/download$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
089952a53d73        nginx:homework1    "/docker-entrypoint..." 47 seconds ago      Up 46 seconds (healthy)            0.0.0.0:8080->80/tcp    funny_ptoleny
test@ubuntu:~/download$
```

5. (Caso de uso) La compañía para la que trabajáis estudia la posibilidad de incorporar a nivel interno una herramienta para la monitorización de logs. Para ello, os han encomendado la tarea de realizar una “Proof of Concept” (PoC). Tras evaluar diferentes productos, habéis considerado que una buena opción es la utilización del producto Elastic stack, cumple con los requisitos y necesidades de la empresa.

Tras comentarlo con el CTO a última hora de la tarde, os ha solicitado que preparéis una presentación para mañana a primera hora. Dado el escaso margen para montar la demostración, la opción más ágil y rápida es utilizar una solución basada en contenedores donde levantaréis el motor de indexación (ElasticSearch) y la herramienta de visualización (Kibana).

Rellena el siguiente fichero **docker-compose** para que podáis hacer la demostración al CTO.

Hint: <https://github.com/dockersamples/example-voting-app>

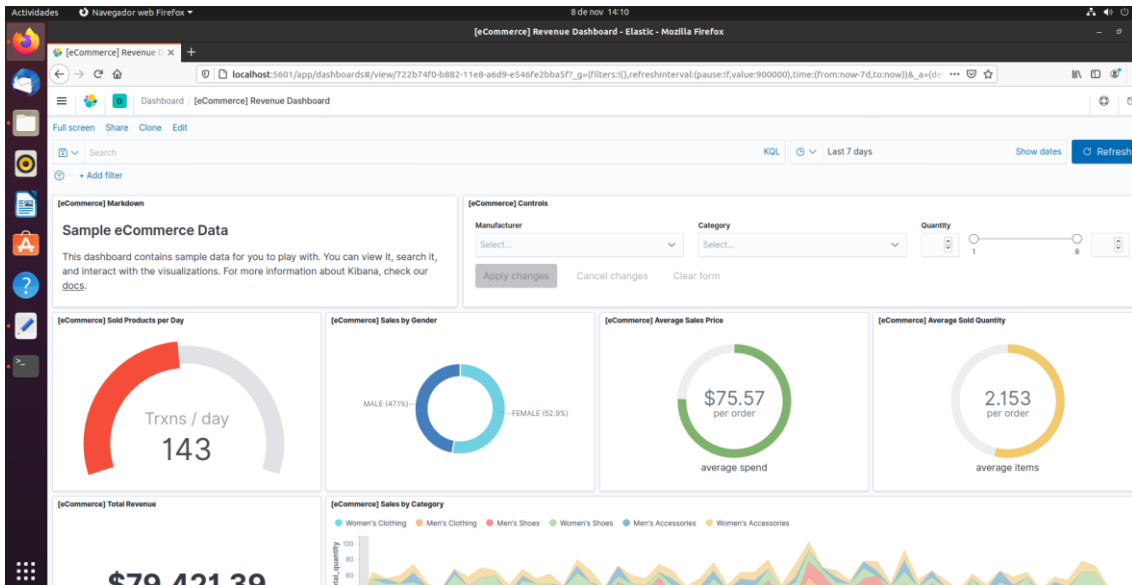
Para comprobar la correcta configuración deberemos acceder al portal de Kibana (<http://localhost:5601>):

Escoger la opción “Try our simple data”:

Seleccionar la opción “Add data”:

Y una vez se hayan cargado los datos, seleccionar View data > Dashboard:

```
docker-compose.yml
C: > Users > Budy > Desktop > MASTER > Docker1 > DockerMDAS > docker-exercises > hw-05 > docker-compose.yml
1  version: '3.6'
2  services:
3    elasticsearch:
4      image: elasticsearch:7.9.3
5      container_name: elastic
6      environment:
7        - discovery.type=single-node
8      networks:
9        - elastic-network
10     ports:
11       - "9200:9200"
12   kibana:
13     image: kibana:7.9.3
14     container_name: kibana
15     networks:
16       - elastic-network
17     environment:
18       - ELASTICSEARCH_HOST=elasticsearch
19       - ELASTICSEARCH_PORT=9200
20     ports:
21       - "5601:5601"
22       - "9300:9300"
23   networks:
24     elastic-network:
25       driver: bridge
```



IMPORTANTE:

Crear un proyecto en vuestro repositorio de código (Github, Gilab, ...) llamado **docker-exercises**. Dentro de ese proyecto, crear una carpeta para la primera entrega, por ejemplo **hw-01**.

Dentro de esa carpeta, crear un fichero por pregunta **answer_exercise_1.md**. Si necesitáis subir ficheros, crear una carpeta para ese ejercicio con todo el contenido. Si necesitáis hacer algún tipo de aclaración, hacerlo en el fichero README.md. En el enlace de entrega del campus, entregar un nota (notes_hw_1.txt) con el enlace del repositorio y el código del commit.