

Computació en Entorns Al Núvol

Pràctica 3 - Containers amb Docker LAB

Grup 19

- Adrià Muro Gómez (1665191)
- David Morillo Massagué (1666540)

Data: 16/05/2025

Introducció.....	3
Objectius.....	3
Exercici 1.....	4
Descripció de la tasca.....	4
Procediment.....	4
Creació de l'entorn de treball.....	4
Creació de l'script.....	4
Creació del Dockerfile.....	5
Construcció de la imatge Docker.....	5
Execució del contenidor.....	6
Eliminació i execució optimitzada.....	6
Monitorització i control.....	6
Preguntes.....	7
Exercici 2.....	8
Descripció de la tasca.....	8
Preguntes.....	8
Exercici 3.....	9
Descripció de la tasca.....	9
Procediment.....	9
Creació de l'script Python.....	9
Creació del Dockerfile.....	10
Preguntes.....	10
Conclusions.....	13

Introducció

En aquesta pràctica es treballa amb contenidors mitjançant la tecnologia *Docker*, dins l'entorn de desenvolupament *AWS Cloud9*. L'objectiu principal és familiaritzar-se amb la creació i execució de contenidors que encapsulin aplicacions o scripts escrits en diferents llenguatges de programació, com ara *Node.js* i *Python*.

Docker s'ha consolidat com una eina essencial en el desplegament d'aplicacions modernes, atès que proporciona un entorn lleuger, portable i consistent que facilita l'automatització i escalabilitat dels sistemes distribuïts.

Durant la pràctica, es realitzen tres exercicis progressius. Cada exercici està dissenyat per introduir un concepte o tècnica diferent relacionada amb la creació d'imatges *Docker*, la definició de *Dockerfiles*, i l'execució de contenidors que executen aplicacions web o scripts que interactuen amb serveis externs com *APIs*.

Tot això es duu a terme dins un entorn *Cloud9* sobre *AWS*, proporcionant així una experiència de treball realista i alineada amb les pràctiques actuals de desplegament en el núvol.

Objectius

Els objectius d'aquesta pràctica són els següents:

- Entendre el funcionament bàsic de Docker i els avantatges d'utilitzar contenidors en entorns distribuïts.
- Aprendre a escriure Dockerfiles per crear imatges personalitzades que incloguin aplicacions i dependències.
- Executar contenidors de forma local dins AWS Cloud9, monitoritzant la seva sortida i comportament.
- Integrar diferents llenguatges de programació dins de contenidors per desenvolupar solucions híbrides.
- Experimentar amb aplicacions que exposen serveis via HTTP i scripts que interactuen amb APIs externes.
- Adquirir autonomia en la construcció i execució d'entorns contenitzats en el núvol.

Exercici 1

Descripció de la tasca

En aquest primer exercici de la pràctica, l'objectiu és crear i executar una imatge Docker personalitzada que contingui un script Python anomenat `showtime.py`.

Aquest script imprimeix per pantalla l'hora actual cada 5 segons. El desenvolupament es fa dins l'entorn *AWS Cloud9*, aprofitant la seva integració amb una instància *EC2* com a host d'execució de contenidors.

Procediment

Creació de l'entorn de treball

Es crea una carpeta anomenada *containers* i s'hi accedeix des de l'entorn AWS Cloud9 per emmagatzemar-hi el codi.



Creació de l'script

Es crea un fitxer anomenat `showtime.py` que mostra l'hora actual cada cinc segons dins d'un bucle de 100 iteracions.

```
1 import time
2 import datetime
3
4 print("Hello Docker")
5 for i in range(100):
6     current_time = datetime.datetime.now()
7     print("Current time:", current_time)
8     time.sleep(5)
```

Creació del *Dockerfile*

Es genera un fitxer *Dockerfile* que defineix una imatge basada en *node:11-alpine*, instal·la *Python3*, crea un directori d'aplicació, copia l'script, li assigna permisos i defineix el directori de treball i el punt d'entrada.

```
showtime.py x Dockerfile +
1 # Start from a minimal Linux version: Alpine
2 FROM node:11-alpine
3
4 # install python3
5 RUN apk add --update --no-cache python3
6
7 # create a directory where we will install our application
8 RUN mkdir -p /opt/app
9
10 # copy our application into the directory
11 COPY showtime.py /opt/app
12 |
13 # Provide execution permissions to the script
14 RUN chmod 744 /opt/app/showtime.py
15
16 # Indicate where will the working directory for the application
17 WORKDIR /opt/app
18
19 # Indications of which process will be started
20 ENTRYPOINT [ "python3", "-u", "/opt/app/showtime.py"]
```

Construcció de la imatge Docker

Es construeix una imatge amb el nom *showtime* i la versió *v1* a partir del directori on es troba el *Dockerfile* i l'script (*~/environment/containers*).

```
voclabs:~/environment/containers $ docker build --tag showtime:v1 ~/environment/containers
[+] Building 9.7s (11/11) FINISHED docker:default
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 636B 0.0s
=> [internal] load metadata for docker.io/library/node:11-alpine 0.5s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/6] FROM docker.io/library/node:11-alpine@sha256:8bb56bab197299c8ff820f1a55462890caf08f57 3.8s
=> => resolve docker.io/library/node:11-alpine@sha256:8bb56bab197299c8ff820f1a55462890caf08f57 0.0s
=> => sha256:e7c96db7181be991f19a9fb6975cddb73c65f4a2681348e63a141a2192a5f10 2.76MB / 2.76MB 0.7s
=> => sha256:0119aca4464934fdf40121363b1cf0537e464c07c790df99058ba587b14aaad 21.66MB / 21.66MB 1.7s
=> => sha256:40df19605a18a2dc4e3c6c0cb5d0a93fd2f7615e2a92eccd743698c24c23fe84 1.33MB / 1.33MB 0.2s
=> => sha256:8bb56bab197299c8ff820f1a55462890caf08f57ffe3b91f5fa6945a4d505932 1.42kB / 1.42kB 0.0s
=> => sha256:914ff2c2145de019a19c080a9e42b5763c826194110ec8e02c8e92845799fba6 1.16kB / 1.16kB 0.0s
=> => sha256:f18da2f58c3dabd7d06c02f4a76719ba0c786737aa83d4bfc8f1e5cb0800b96a 5.66kB / 5.66kB 0.0s
=> => sha256:82194b8b4a64bd286671a1464d3fe319832aaa67c4e41c455fdffd295ae7aa73 279B / 279B 0.3s
=> => extracting sha256:e7c96db7181be991f19a9fb6975cddb73c65f4a2681348e63a141a2192a5f10 0.1s
=> => extracting sha256:0119aca4464934fdf40121363b1cf0537e464c07c790df99058ba587b14aaad 1.7s
=> => extracting sha256:40df19605a18a2dc4e3c6c0cb5d0a93fd2f7615e2a92eccd743698c24c23fe84 0.1s
=> => extracting sha256:82194b8b4a64bd286671a1464d3fe319832aaa67c4e41c455fdffd295ae7aa73 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 272B 0.0s
=> [2/6] RUN apk add --update --no-cache python3 2.0s
=> [3/6] RUN mkdir -p /opt/app 0.4s
```

Una vegada s'ha construït la imatge executem *docker images* a la terminal per comprovar que efectivament s'ha creat correctament.

```
voclabs:~/environment/containers $ docker images
REPOSITORY    TAG       IMAGE ID        CREATED         SIZE
showtime      v1        b8631d8945e3   4 minutes ago  128MB
```

Execució del contenidor

S'executa un contenidor anomenat *time* a partir de la imatge *showtime:v1*. El contenidor mostra l'hora per pantalla en temps real.

```
voclabs:~/environment/containers $ docker run --name time -it showtime:v1
Hello Docker
Current time: 2025-05-15 17:55:56.972243
Current time: 2025-05-15 17:56:01.977498
Current time: 2025-05-15 17:56:06.982490
Current time: 2025-05-15 17:56:11.985446
```

Eliminació i execució optimitzada

Un cop aturat el contenidor, el podem eliminar a partir de la comanda *docker rm time*.

```
voclabs:~/environment/containers $ docker rm time
time
voclabs:~/environment/containers $
```

Per tal de dur a terme una execució del contenidor més optimitzada, aquest es pot executar en segon pla, en el nostre cas amb *docker run --rm -d --name time showtime:v1*. Per comprovar que efectivament s'està executant en segon pla podem aplicar *docker logs time* per veure els seus resultats.

```
voclabs:~/environment/containers $ docker run --rm -d --name time showtime:v1
b40f4774016d06799ca79e904ba915ef6fe2e36bbbb23d35811bdbc07f00c6dd
voclabs:~/environment/containers $ docker logs time
Hello Docker
Current time: 2025-05-15 17:58:14.039632
Current time: 2025-05-15 17:58:19.042696
```

Monitorització i control

Es poden consultar els logs del contenidor, comprovar si està actiu amb *docker ps -a* i aturar-lo manualment si cal amb *docker stop time*.

```
voclabs:~/environment/containers $ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS   NAMES
b40f4774016d   showtime:v1   "python3 -u /opt/app..." 50 seconds ago Up 49 seconds           time

voclabs:~/environment/containers $ docker stop time
time
```

Preguntes

Quines són les capes d'aquest contenidor que heu creat? Fes una llista explicant cadascuna d'elles?

El contenidor *Docker* es crea a partir d'una imatge que es forma per capes. Cada capa correspon a una instrucció del *Dockerfile* i aporta canvis o afegeix contingut a la imatge. Aquestes capes es poden reutilitzar en diferents imatges si són iguals, fent el procés eficient.

Les capes del contenidor que hem creat són les següents:

- **Base: node:11-alpine**
Aquesta és la capa base que conté la imatge oficial de *Node.js* versió 11, construïda sobre Alpine Linux, una distribució lleugera. Aquesta capa inclou el sistema operatiu mínim i l'entorn *Node.js* necessari.
- **Instal·lació de Python 3**
Aquesta capa s'afegeix amb la comanda *RUN apk add --update --no-cache python3* i instal·la *Python 3* a la imatge base. Això permet executar *scripts* Python dins el contenidor.
- **Creació del directori /opt/app**
Amb la comanda *RUN mkdir -p /opt/app* es crea un directori dins la imatge on s'ubicarà l'aplicació.
- **Còpia de l'script showtime.py**
Aquesta capa copia el fitxer *showtime.py* des del sistema host al directori */opt/app* dins la imatge, fent que el codi de l'aplicació formi part del contenidor.
- **Canvi de permisos de l'script**
Amb *RUN chmod 744 /opt/app/showtime.py* es modifiquen els permisos del fitxer perquè sigui executable.
- **Establiment del directori de treball**
Amb la instrucció *WORKDIR /opt/app* es configura el directori de treball del contenidor, és a dir, el directori on s'executaran les ordres quan s'arrenqui.

- **Punt d'entrada (ENTRYPOINT)**

Finalment, amb `ENTRYPOINT ["python3", "-u", "/opt/app/showtime.py"]` es defineix la comanda que s'executarà en iniciar el contenidor, en aquest cas, executar l'script `Python showtime.py`.

Exercici 2

Descripció de la tasca

En aquest exercici es treballa amb una aplicació web bàsica desenvolupada amb *Node.js* i el *framework Express*. L'objectiu és crear una imatge *Docker* que contingui aquesta aplicació i que l'executi de manera que estigui disponible via navegador web a través d'un port específic.

L'usuari ha d'escriure un *Dockerfile* que instal·li *Node.js*, copii el codi de l'aplicació dins la imatge i configuri el port d'exposició.

Preguntes

Quines són les capes d'aquest contenidor que heu creat? Pots fer-ne una llista?

Les capes que es creen amb aquest *Dockerfile* són:

- **Base: node:11-alpine**
Imatge base que conté Alpine Linux amb *Node.js* versió 11 instal·lat. Proporciona un entorn lleuger per executar aplicacions *Node*.
- **Creació del directori /usr/src/app**
Amb la comanda `RUN mkdir -p /usr/src/app` es crea un directori on es posarà el codi de l'aplicació.
- **Establiment del directori de treball /usr/src/app**
Amb `WORKDIR /usr/src/app` es defineix el directori on s'executaran les ordres següents dins el contenidor.
- **Còpia del codi font al contenidor**
Amb `COPY . .` es copien tots els fitxers i carpetes del directori local (on es construeix la imatge) al directori de treball del contenidor `/usr/src/app`.
- **Instal·lació de dependències de Node.js**
Amb `RUN npm install` s'instal·len totes les dependències declarades al `package.json`, preparant l'entorn per executar l'aplicació.

- **Exposició del port 3000**
Amb *EXPOSE 3000* s'indica que el contenidor escoltarà per connexions al port 3000, que és on l'aplicació web estarà disponible.
- **Comanda per defecte per iniciar l'aplicació**
Amb *CMD ["npm", "run", "start"]* es defineix la comanda que s'executarà quan s'iniciï el contenidor: Arrencar l'aplicació amb *npm run start*.

Exercici 3

Descripció de la tasca

Aquest últim exercici es centra en l'execució d'un script Python que consulta una *API* externa per obtenir informació d'actualitat, com ara notícies. El repte consisteix a construir una imatge *Docker* que contingui totes les dependències necessàries, com ara el client de l'*API* i possibles biblioteques addicionals (com *requests* o *json*), i que executi automàticament el script quan el contenidor s'inicia.

Aquest exercici combina el coneixement adquirit als exercicis anteriors i afegeix la dificultat de gestionar dependències externes i comunicació amb serveis remots. A més, demostra la utilitat dels contenidors per empaquetar no només aplicacions completes, sinó també scripts i tasques puntuals que poden ser executades de manera segura i repetible, independentment del sistema host.

Procediment

Creació de l'script Python

Primer de tot, es crea un fitxer anomenat *api_example.py* amb el contingut següent:

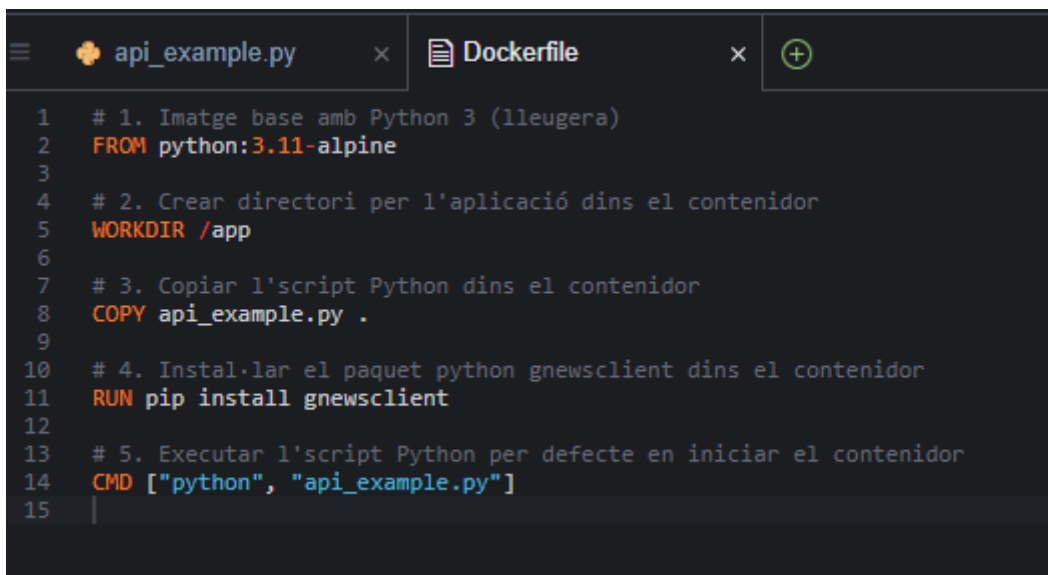
```
api_example.py
1  from gnewsclient import gnewsclient
2
3  client = gnewsclient.NewsClient(language='english',
4                                  location='United States (English)',
5                                  topic='World',
6                                  max_results=3)
7
8  news_list = client.get_news()
9
10 for item in news_list:
11     print("Title : ", item['title'])
12     print("Link : ", item['link'])
13     print("")
```

Aquest script fa una consulta a Google News mitjançant la llibreria *gnewsclient*. Recupera 3 notícies internacionals en anglès i imprimeix per pantalla el títol i l'enllaç de cada una.

Serveix com a exemple senzill per mostrar com consumir una *API* i visualitzar-ne la resposta dins d'un contenidor *Docker*.

Creació del Dockerfile

A continuació, es crea un fitxer anomenat Dockerfile amb el contingut següent, degudament comentat:

A screenshot of a code editor with two tabs: 'api_example.py' and 'Dockerfile'. The 'Dockerfile' tab is active, showing a multi-line script with comments in Spanish and Docker instructions. The instructions include setting the base image to python:3.11-alpine, creating a working directory at /app, copying the api_example.py file into the container, installing the gnewsclient package using pip, and setting the command to run python api_example.py when the container starts.

```
1 # 1. Imatge base amb Python 3 (lleugera)
2 FROM python:3.11-alpine
3
4 # 2. Crear directori per l'aplicació dins el contenidor
5 WORKDIR /app
6
7 # 3. Copiar l'script Python dins el contenidor
8 COPY api_example.py .
9
10 # 4. Instal·lar el paquet python gnewsclient dins el contenidor
11 RUN pip install gnewsclient
12
13 # 5. Executar l'script Python per defecte en iniciar el contenidor
14 CMD ["python", "api_example.py"]
15
```

Preguntes

Quins són els passos del vostre *Dockerfile*?

1. Tria una imatge base amb Python 3 (*python:3.11-alpine*).
 2. Estableix el directori de treball (*WORKDIR /app*).
 3. Copia l'script *api_example.py* dins el contenidor.
 4. Instal·la el paquet *gnewsclient* amb *pip install*.
 5. Defineix la **comanda per defecte per executar** l'script Python en iniciar el contenidor.
-

Quines són les operacions necessàries per copiar i executar el codi Python?

- *COPY api_example.py* .
Copia el fitxer local al directori de treball dins del contenidor.
- *CMD ["python", "api_example.py"]*
Indica la comanda que s'executarà per defecte quan el contenidor s'iniciï.

Quines són les instruccions per crear, executar i monitoritzar la sortida del vostre contenidor *Docker*?

- **Construcció de la imatge *Docker*:** *docker build -t my-python-news* .

```
voclabs:~/environment/containers $ docker build -t my-python-news .
[+] Building 2.4s (7/8)                                docker:default
=> [internal] load build definition from Dockerfile    0.0s
=> => transferring dockerfile: 499B                   0.0s
=> [internal] load metadata for docker.io/library/python:3.11-alpine 0.4s
=> [internal] load .dockerignore                      0.0s
=> => transferring context: 2B                         0.0s
=> [1/4] FROM docker.io/library/python:3.11-alpine@sha256:d0199977fdae5d1109a89d0b0014468465e0 1.8s
=> => resolve docker.io/library/python:3.11-alpine@sha256:d0199977fdae5d1109a89d0b0014468465e0 0.0s
=> => sha256:d0199977fdae5d1109a89d0b0014468465e014a9834d0a566ea50871b3255ad 10.30kB / 10.30kB 0.0s
=> => sha256:81d42a73add8e508771ee8e923f9f8392ec1c3d1e482d7d594891d06e78fb51c 1.74kB / 1.74kB 0.0s
=> => sha256:43665da319622502584882475cf1cdca6afe17495867f55be1bd36b3627533d1 5.16kB / 5.16kB 0.0s
=> => sha256:f18232174bc91741fdf3da96d85011092101a032a93a388b79e99e69c2d5c870 3.64MB / 3.64MB 0.1s
=> => sha256:eb8fdeee2de28d2dc60aa8b92f96d32d3b35f16aa134db72e39a9dce44183 460.21kB / 460.21kB 0.2s
```

- **Execució del contenidor:** *docker run --name news-container my-python-news*

```
voclabs:~/environment/containers $ docker run --name news-container my-python-news
/usr/local/lib/python3.11/site-packages/fuzzywuzzy/fuzz.py:11: UserWarning: Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning')
Title : An aide, a diplomat and a spy: Who is Putin sending to Turkey? - Reuters
Link : https://news.google.com/rss/articles/CBMiIAFBVV95cUxNdHY0dHd4VWRGZ21QcnRTRmNxVjJJaN3k0UTYxTTZrX0t0V2dSVmVhaHJ1SXVickExdzQ2T01vYjdnSkRUUkdWQVJ4UVVNX000V1J1RGRHbTZ3Ti1wS1huWjVzc0RJRnZnWm1mekFFY01tRzBZS1hwMkRtaG5HN0hVOE9zZU1vcFNjQWVCVkJ5S0M4Nkpk?oc=5

Title : Trump seals economic agreements in Middle East in final stretch and more top headlines - Fox News
Link : https://news.google.com/rss/articles/CBMipgFBVV95cUxOVng1ampHb1J4SXFpX1Y5Q1ZjUDFmaGkxeW5NT29SQ0NjVURVTXZOSDVPt0hPODZBMV1VZjM0ZC1qY2NaOG45NjUzMFlTe1VDQjhwdERyRX1iOFJ5T2ppM2M4TEI3Qy1TY1dHdUtQQ1BiVmdQdTZmMk1SbER2a2x6TmFbnVEX3NoR2pGx1V3T1ctbzJSMUZFStFUXdzdHb1pxX3V1X0930gGrAUFVX31xTE1EbDJCQTRVQ0p1eGJ3Rvd5U0U1uRUhLTTFVaTlZRDc5dWo4SzdGaUpOd1NtY2ZQTThpMnh1cHdkdERQb1dsZU5kNjB6aV9SN1E1MFBrCEZvWDFxUUtXOHd1NXA3Vk5FTnIxNU0tNVA2aTlRnZuQ51yXy1hUU0tT25qamZtMXc1dktXMRreWwxeC10TXdvdn1yNhp4d3EwRXh3NzNZaFd0cEk0TQ?oc=5

Title : What happened to Valeria Marquez, Mexican influencer shot live on TikTok? - Al Jazeera
Link : https://news.google.com/rss/articles/CBMisgFBVV95cUxOdVM3e1JXRfHoQjBibDVRv3ZzYnB1MTktYUduUD8fZDBjcUF5Y3hLQ3I2RXZlW1dIQVhXdeSCLUVQd1FRNDRwSjRjOGp5d2VhSnAzR3d2ZmxlWSX0VZnZnMQ0NxbGc2SEt6NG9Pa2VsaEFBwJbVYm1MV1JfbEFJRFAwR1VHbkRFRFJqUHD0cEFNaEjYdG9ONDM2N2JuUjU1anJWMG52a2hiY0NNWT15cE0yT2d80gG3AUFVX31xTE9sa
```


Conclusions

Durant aquest laboratori hem après a treballar amb contenidors *Docker* en un entorn de núvol utilitzant *AWS Cloud9*. Els coneixements i habilitats adquirits es poden resumir en els següents punts:

- 1. Comprensió del funcionament de Docker:**
Hem entès com *Docker* permet encapsular aplicacions i les seves dependències dins d'imatges lleugeres i portables. Això facilita l'execució consistent d'aplicacions en diferents entorns.
- 2. Creació i gestió de Dockerfiles:**
Hem après a escriure un fitxer *Dockerfile*, el qual defineix les instruccions per construir una imatge de contenidor. Aquest procés inclou des de la selecció d'una imatge base fins a la instal·lació de paquets, la còpia de fitxers i la definició del punt d'entrada de l'aplicació.
- 3. Construcció i execució d'imatges Docker:**
S'han generat imatges pròpies a partir del *Dockerfile* i hem executat contenidors que mostren la sortida del nostre script, tant en primer pla com en segon pla. També s'ha après a eliminar contenidors per gestionar els recursos del sistema.
- 4. Desplegament d'aplicacions web:**
A través de l'exercici 2, hem vist com empaquetar i desplegar una aplicació *Node.js* en un contenidor, exposant ports i instal·lant dependències mitjançant *npm*.
- 5. Contenidors personalitzats amb Python:**
Hem creat un contenidor que executa un script Python que accedeix a notícies utilitzant la llibreria *gnewsclient*. Això ens ha permès entendre com instal·lar paquets *Python* dins d'un contenidor i automatitzar l'execució de scripts.
- 6. Gestió de contenidors i supervisió:**
S'ha practicat l'ús d'ordres bàsiques per monitoritzar (*logs*), aturar, eliminar i verificar l'estat dels contenidors, millorant així la nostra autonomia en la gestió del cicle de vida dels contenidors.

En definitiva, aquest laboratori ens ha proporcionat una base sòlida per entendre i aplicar *Docker* en entorns de desenvolupament i execució d'aplicacions distribuïdes. Aquesta experiència serà molt útil per a futurs projectes en els quals calgui desplegar serveis de manera eficient i escalable al núvol.