

Universitat de Barcelona

FACULTAT DE MATEMÀTIQUES I INFORMÀTICA

# INFORME PRÀCTICA 1: DOCKER

*Sistemes Operatius 2*

Junjie Li i Manuel Liu Wang

Octubre 2022

## 1 Introducció

En aquesta pràctica ens introduïrem al Docker, el Docker ens permet manipular de forma senzilla els contenidors, així tots els usuaris connectats a aquell "contenedor" treballar amb les mateixes versions, documents, etc sense haver de tenir-los instal·lats.

També podem controlar l'espai de directoris/fitxers als quals poden accedir, el nombre màxim de CPUs que poden utilitzar, la memòria RAM màxima que poden utilitzar, el nombre màxim de processos que es poden executar a un contenidor, entre altres coses.

## 2 Experiments amb els contenidors

### 2.2 Docker: aplicació statistics

**Pregunta** Com es pot "despertar" un contenidor que ha sigut "aturat" prèviament de forma que es puguin introduir noves instruccions dins del contenidor?

Per "despertar" un contenidor "aturat" prèviament a l'haver fet èxit, s'han de cridar a les comandes **docker container start [idContenedor]** , on la id es pot obtenir fent **docker container ls -a**, i la comanda **docker container attach [idContenedor]** per accedeix.

.....

### 2.3 Docker: fork-bomb

**Pregunta** Quants processos fork-bomb s'estan executant dins del contenidor? Com ho compteu? Podeu fer servir instruccions de la línia de comandes (p.ex.bash) per saber-ho?

S'estan executant **32 processos** dins del contenidor. Aquest nombre està a la dreta del tot a la columna PIDS, després d'executar la comanda **docker ps -a**, d'altra banda, es el nombre de forks que havíem limitat amb la comanda **docker run --ulimit nproc=32:64 --cpus 1 -ti fork-bomb** .

.....

**Pregunta** Quanta CPU està utilitzant el contenidor? Està utilitzant 1, 2 o més CPUs? Per mirar-ho tingueu en compte que en un ordinador amb 4 CPUs el màxim de CPU que es pot ocupar és d'un 400%.

La CPU utilitzada, de les 4 CPU's que té l'ordinador, el percentatge utilitzat varia amb el temps, aquest percentatge oscil·la entre 94% i 106%, fent la mitja, ens dona un 100%, per tant, podem dir que està utilitzant una CPU, tal com s'havia limitat en la comanda **docker run --ulimit nproc=32:64 --cpus 1 -ti fork-bomb**.

.....

**Pregunta** Observeu que en aquest cas hem executat el contenidor en mode interactiu. És a dir, apareix la línia de comandes que ens permet executar una comanda. Proveu de fer-ho! Per exemple, executeu un ls o ps. Per què dóna el bash un missatge d'error?

Perquè abans amb la comanda **docker run --ulimit nproc=32:64 --cpus 1 -ti fork-bomb** havíem limitat el nombre de processos executables, llavors a l'executar un ls o ps excedeix el límit imposat prèviament, per tant, dona error.

.....

## 2.4 Docker: connexió via socket

### Execució sense contenidors

**Pregunta** Proveu d'executar els dos servidors en dos terminals diferents, sense fer servir cap contenidor. Què és el que succeeix en intentar fer-ho?

La petició no és acceptada pel dispositiu. mostra un missatge de: **bind() ha fallat. Prova un altre port.**

.....

**Pregunta** Per què succeeix?

Perquè no pot haver-hi més d'un servidor en un port.

.....

### Execució amb contenidors

**Pregunta** Observeu, al README, que per fer el mapat de ports es fa servir l'opció "-p" per executar el contenidor. Descriu breument, fent servir la documentació oficial del Docker, què és el que permet fer l'opció "-p". Indiqueu també què passa si no es fa servir l'opció "-p" per executar el contenidor. Quina és la pàgina web del Docker on està descrit el funcionament d'aquesta opció?

El que permet fer l'opció "-p" és fer que un port estigui disponible per serveis fora de Docker, o per a contenidors de Docker que no estan connectats a la xarxa del contenidor. Si no es fa servir l'opció "-p", aquesta no publicarà cap dels seus ports al món exterior. La pàgina web on està descrit aquesta funció és: <https://docs.docker.com/config/containers/container-networking/>

.....

**Pregunta** Per què, actualment, es fan servir els contenidors per executar els serveis associats a una aplicació més gran en contenidors diferents? Quines avantatges aporta?

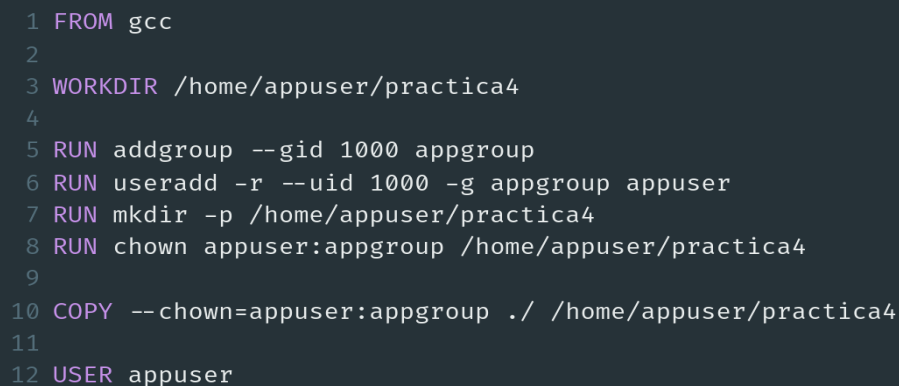
Perquè d'aquesta manera podem ajustar les llibreries, la versió que utilitzarem, l'escala de l'aplicació, etc. D'aquesta manera podem modificar diverses parts d'un programa amb major facilitat i seguretat.

Els avantatges que aporta és el fet de poder executar diferents serveis, modificar-los i desenvolupar-los, amb més seguretat, ja que el sistema operatiu és independent de l'aplicació i el contenidor no està fix a un dispositiu local.

## 3 Exercicis: pautes pel desenvolupament d'aplicacions

### 3.2 Etapa de desenvolupament: compilació i execució en un contenidor

**Exercici** Crear un contenidor que tingui el compilador (gcc). En executar el contenidor es farà un “bind mount” de forma que internament hi ha un director **/home/appuser/practica4** des del qual es pot accedir a un directori extern al Docker el qual contindrà el codi font i les dades de la pràctica 4. Provar de compilar i executar el codi des de l'interior del contenidor. Comproveu que els executables així com les dades associades que es llegeixen i s'escriuen en executar les aplicacions són fora del contenidor.



```
1 FROM gcc
2
3 WORKDIR /home/appuser/practica4
4
5 RUN addgroup --gid 1000 appgroup
6 RUN useradd -r --uid 1000 -g appgroup appuser
7 RUN mkdir -p /home/appuser/practica4
8 RUN chown appuser:appgroup /home/appuser/practica4
9
10 COPY --chown=appuser:appgroup ./ /home/appuser/practica4
11
12 USER appuser
```

Figure 1: Dockerfile Exercici 1

Per executar codi de la pràctica 4 en un contenidor, necessitem un fitxer Dockerfile, com la imatge de dalt, no hi ha una gran diferència amb el Dockerfile a l'experiment, l'únic canvi és canviar el **WORKDIR** en **Practica4**

Per compilar el nostre contenidor: **docker build -t practica4:v1 .**

I per executar el contenidor perquè es pugui accedir a un directori extern al Docker farem servir **docker run -ti -v \$(pwd):/home/appuser/practica4 --name practica4-exercici-1 -d practica4:v1**

És a dir **-v [directori absolut]** assigna els fitxers del directori actual de **pwd** al contenidor de **--name [nom de contenidor]** del directori **/home/appuser/practica4** i **-d [imatge]** utilitza aquesta imatge al backend.

D'aquesta forma, és adequat per penjar directoris de codi i fitxers de configuració. Es pot connectar a diversos contenidors.

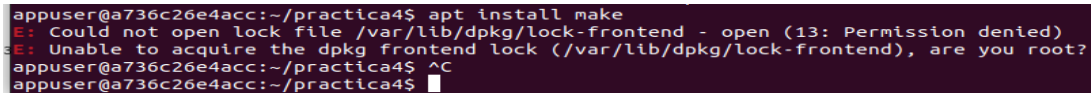
Un cop executem el contenidor, podem utilitzar **docker exec -it practica4-exercici-1 /bin/bash** per entrar al contenidor, 0 entra el contenidor directament en temps d'execució

```
docker run -ti --name practica4-exercici-1 -v $(pwd):/home/appuser/practica4  
practica4:v1 /bin/bash
```

.....

**Pregunta** Observar que per poder escriure al directori extern al Docker cal permisos per poder-ho fer. Com ho solucioneu?

Per poder escriure al directori extern del Docker, com podem veure en la següent imatge:



```
appuser@a736c26e4acc:~/practica4$ apt install make  
E: Could not open lock file /var/lib/dpkg/lock-frontent - open (13: Permission denied)  
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontent), are you root?  
appuser@a736c26e4acc:~/practica4$ ^C  
appuser@a736c26e4acc:~/practica4$
```

Figure 2: Docker permisos

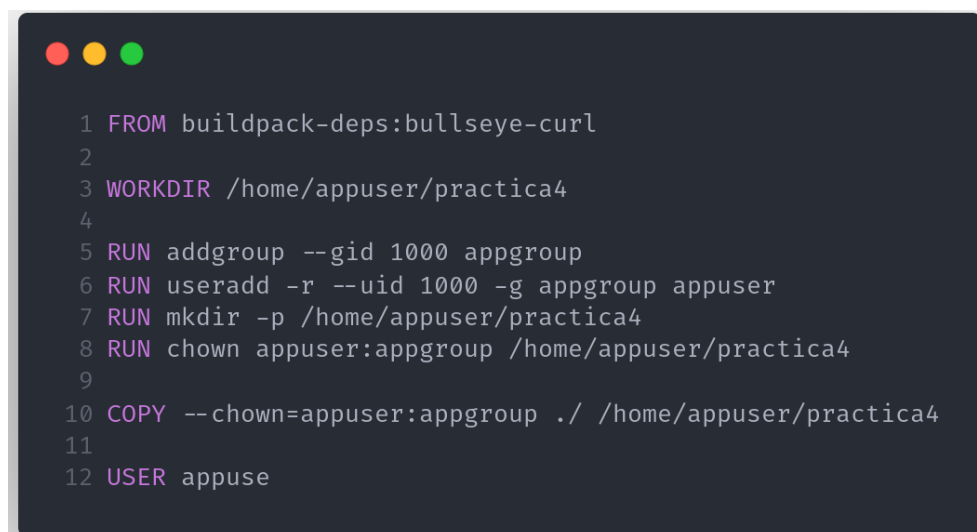
Per solucionar aquest problema, podem utilitzar la següent comanda: **ocker run -ti -u root --name practica4-exercici-1 -v \$(pwd):/home/appuser/practica4 practica4:v1 /bin/bash, -u [user]** Podem definir el tipus d'usuari que entra al contenidor, que pot ser **root** o l'usuari que hem creat al **dockerfile**.

Una altra manera és definir l'usuari com a **root** directament a Dockerfile, amb: **USER root** al figura1

.....

### 3.3 Etapa de producció: execució en un contenidor petit


**Exercici** L'objectiu és utilitzar una imatge que contingui el mínim necessari per poder executar l'aplicació (accedint a les dades externes amb el "bind mount"). Quin és el Dockerfile corresponent? Com heu arribat a trobar-lo?



```
1 FROM buildpack-deps:bullseye-curl  
2  
3 WORKDIR /home/appuser/practica4  
4  
5 RUN addgroup --gid 1000 appgroup  
6 RUN useradd -r --uid 1000 -g appgroup appuser  
7 RUN mkdir -p /home/appuser/practica4  
8 RUN chown appuser:appgroup /home/appuser/practica4  
9  
10 COPY --chown=appuser:appgroup ./ /home/appuser/practica4  
11  
12 USER appuse
```

Figure 3: Dockerfile Exercici 2

Veient [https://hub.docker.com/\\_/gcc](https://hub.docker.com/_/gcc), vam trobar que la capa superior de **gcc** és **buildpack-deps:bullseye-curl**, de manera que podem utilitzar directament aquesta capa per construir el nostre contenidor, la imatge següent ens permet veure que la mida de la imatge és de només **154 MB**



REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
small-size	latest	30e644658942	11 minutes ago	154MB
practica4	v1	95151a75b76a	About an hour ago	4.65GB
server2	latest	71687ffcdee7	11 hours ago	1.27GB
server1	latest	9a18efa61e7a	11 hours ago	1.27GB
statistics	latest	c0c52f37a13a	11 days ago	1.28GB
busybox	latest	2bd29714875d	2 weeks ago	1.24MB
gcc	latest	feaa519db663	2 weeks ago	1.27GB
buildpack-deps	bullseye-curl	8138e50ba783	2 weeks ago	154MB
hello-world	latest	feb5d9fea6a5	12 months ago	13.3kB

Figure 4: Size de imatges

### 3.4 Etapa de producció: execució en un contenidor fent servir volums

**Pregunta** Què és el que fan cadascuna de les instruccions anteriors?

- **docker volume create vol-practica4** Crea un volum nou anomenat **vol-practica4** pels contenidors
- **docker volume ls** Veure tots els volums creats per Docker
- **cd practica4** Canvia directori al **practica4**
- **docker container create --name temp -v vol-practica4:/data busybox** Crea un contenidor anomenat **temp** que utilitza l'imatge de **busybox** on munta el volum de **vol-practica4** i el directori del contenidor **/data**
- **docker cp -a . temp:/data** Copia les dades del directori actual al contenidor **temp** del directori **/data**
- **docker rm temp** Borra el contenidor **temp**, encara que s'hagi eliminat el contenidor, les seves dades encara existien

**Exercici** Comprovem que s'ha copiat tot correctament. Per això es demana executar el contenidor petit muntant el volum **vol-practica4** en el directori **/home/appuser/practica4**. Què hi ha en aquest directori? Quin és l'usuari propietari dels arxius? Es pot executar el codi que hi ha?

Utilitzem la següent comanda per executar el contenidor amb muntant el volum

**docker run -ti -u root --name practica4-exercici-3 -v vol-practica4:/home/appuser/practica4 small-size:latest**

En aquest directori hi ha :

```
drwxrwxrwx 2 appuser appgroup 4096 Oct 3 20:54 model
root@5feee6033af0:/home/appuser/practica4# ls
Dockerfile  README.md  data      hash.bin  mainRecc.c  mainSave  mainSave.o  meta.bin
Makefile    Readme.txt dataStructures mainRecc  mainRecc.o  mainSave.c  matrix.bin  model
root@5feee6033af0:/home/appuser/practica4#
```

I l'usuari propietari dels arxius són:

```
appuser@0d29c1286475:~/practica4$ ls -l
total 3112868
-rwxrwxrwx 1 appuser appgroup 607 Oct 3 18:02 Dockerfile
-rwxrwxrwx 1 appuser appgroup 2370 Sep 2 2021 Makefile
-rwxrwxrwx 1 appuser appgroup 282 Oct 3 18:48 README.md
-rwxrwxrwx 1 appuser appgroup 1075 Sep 1 2021 Readme.txt
drwxrwxrwx 2 appuser appgroup 4096 Oct 3 20:53 data
drwxrwxrwx 2 appuser appgroup 4096 Oct 3 20:53 dataStructures
-rwxrwxrwx 1 appuser appgroup 3556000 Oct 3 09:16 hash.bin
-rwxrwxrwx 1 appuser appgroup 27432 Oct 3 09:15 mainRecc
-rwxrwxrwx 1 appuser appgroup 4838 Sep 1 2021 mainRecc.c
-rwxrwxrwx 1 appuser appgroup 27256 Oct 3 09:15 mainSave
-rwxrwxrwx 1 appuser appgroup 2967 Sep 1 2021 mainSave.c
-rwxrwxrwx 1 appuser appgroup 3504 Oct 3 09:15 mainSave.o
-rwxrwxrwx 1 appuser appgroup 3183905592 Oct 3 09:16 matrix.bin
-rwxrwxrwx 1 appuser appgroup 8 Oct 3 09:16 meta.bin
drwxrwxrwx 2 appuser appgroup 4096 Oct 3 20:54 model
appuser@0d29c1286475:~/practica4$
```

I es pot executar el codi igualment

```
makefile  Readme.txt  dataStructures  mainRecc  mainRecc.o  mainSave.c  matrix.bin  model
root@b70a5cbe2b8c:/home/appuser/practica4# ./mainSave
root@b70a5cbe2b8c:/home/appuser/practica4# ./mainRecc 1 2207774
The number of movies seen by the user 2207774 is 57
root@b70a5cbe2b8c:/home/appuser/practica4#
```

## 4 Conclusió

En aquesta pràctica, la part més difícil va ser el Dockerfile, ja que no enteníem algunes coses, a part d'això, hem après les comandes bàsiques del Docker, tenir usuaris connectats a servidors, com utilitzar els volums.