Nom:
Cognoms:

ASIX_CIBER_M11_UF4_PAC1

Per fer aquesta activitat, es pot fer servir una màquina Ubuntu amb Docker instal·lat, o es pot utilitzar la web gratuïta de https://labs.play-with-docker.com/ (cal registrar-se).

TOTS els passos requereixen captures de pantalla demostrant l'execució.

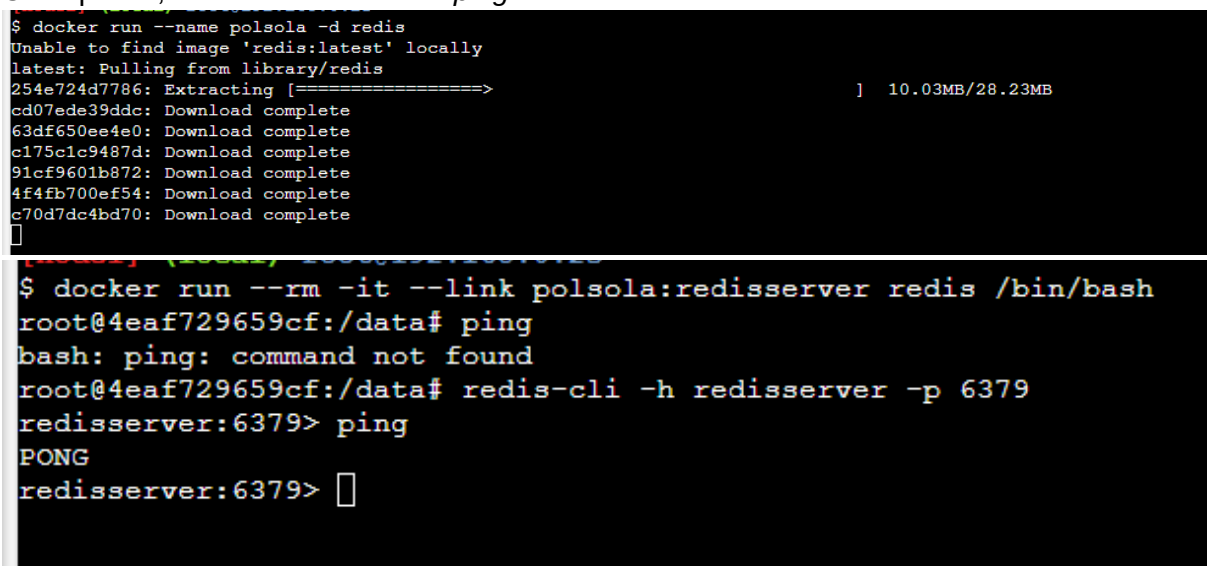
Exercici 1 - Virtualització amb Docker

1.1 (0,5 punts) Si fas servir Play With Docker, crea una instància nova i enganxa una captura de pantalla.
Crear un contenidor, i un altre que hi apunta amb un enllaç:

Recorda canviar "NomCognom" amb les teves dades!

docker run --name nomcognom -d redis
docker run --rm -it --link nomcognom:redisserver redis /bin/bash
redis-cli -h redisserver -p 6379

Un cop dins, executa la comanda "*ping*".



1.2. (0,5 punts) Escriu "*exit*" i prem enter dos cops per sortir altre cop al shell; realitza una captura de pantalla amb l'execució de "*docker ps*".

```
redisserver:6379> exit
root@4eaf729659cf:/data# exit
exit
[node1] (local) root@192.168.0.23 ~
$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS      NAMES
5fdfac3dd1d1   redis     "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   6379/tcp   polsola
[node1] (local) root@192.168.0.23 ~
```

1.3. (0,5 punts) Executa la comanda "*more /etc/issue*" sense connectar-te al contenidor.

```
[node1] (local) root@192.168.0.23 ~
$ docker exec polsola more /etc/issue
::::::::::::::
/etc/issue
::::::::::::::
Debian GNU/Linux 12 \n \l

[node1] (local) root@192.168.0.23 ~
```

1.4. (0,5 punts) Ara connecta't al terminal del contenidor i executa la mateixa comanda "*more /etc/issue*". Explica per què el resultat de la comanda és diferent.

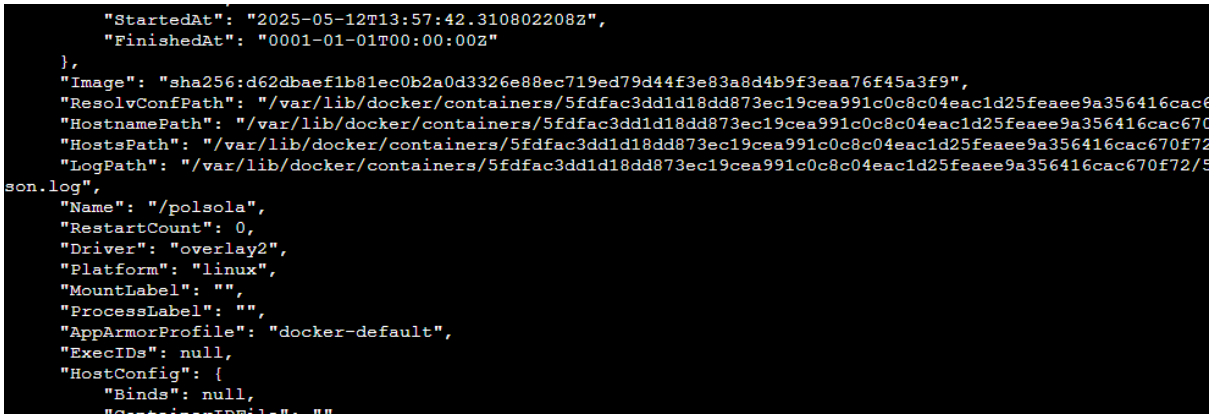
Per connectar-te al contenidor, pots fer servir aquesta comanda:

docker run --rm -it --link nomcognom:redisserver redis /bin/bash

```
Debian GNU/Linux 12 \n \l

(END)
```

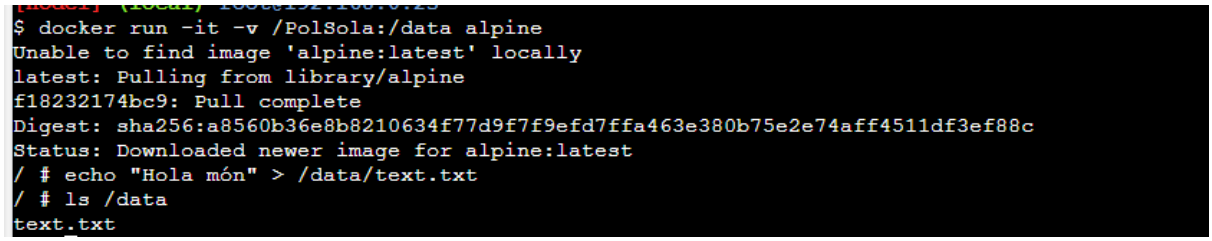
La diferencia que hi ha es que una s'obre el document perque estem en mode interactiu i amb l'altre solament el resultat

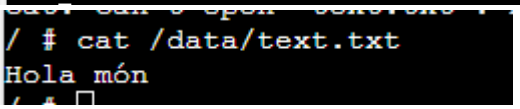
1.5. (0.5 punts) Executa la comanda "*docker inspect IdContenidor > nomFitxer.txt*" (posant les dades pertinents). Això generarà un fitxer de text amb molta informació del contenidor. Fes captura d'un fragment del text on surti el nom del contenidor "*NomCognom*".

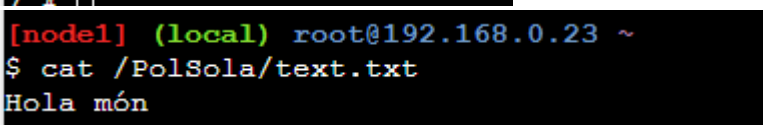


1.6. (1 punt) Ara afegirem volums al contenidor per tal d'aconseguir persistència d'algun directori o fitxers.

- Executa la comanda "*docker run -it -v /NomCognom:/data alpine*" per crear un contenidor bàsic amb Alpine.
- Dins del contenidor, crea un fitxer de text dins de la carpeta "*/data*", i escriu alguna cosa. Fes captura d'un "*ls*" sobre aquesta carpeta.
- Surt del contenidor amb la comanda "*exit*". Si tot ha anat bé, haurieu de tenir una carpeta a l'arrel del sistema "*/NomCognom*", i dins hauria d'estar el fitxer que heu creat dins del contenidor. Fes un "*cat*" sobre aquest fitxer.







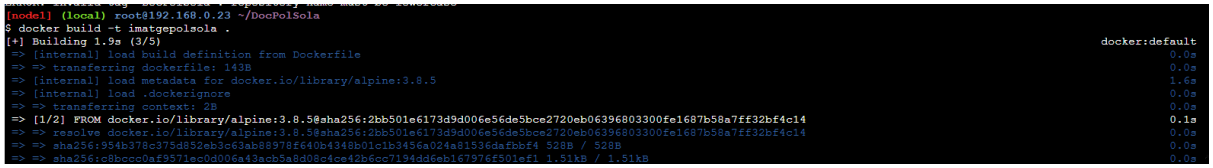
1.7 (0.5 punts) Ara crearem una imatge de Docker. Per fer-ho, haurem d'escriure un Dockerfile. Aquesta imatge partirà amb base a una altre (Alpine), en una versió concreta, en lloc de l'última. Crea una carpeta anomenada "DockNomCognom", i genera un arxiu de text anomenat "Dockerfile" amb el següent contingut:

```
# Instal·lem la versió 3.8.5 d'Alpine. En entorns operatius, convé
tenir una versió fixa que sabem que funciona, en lloc d'agafar
```

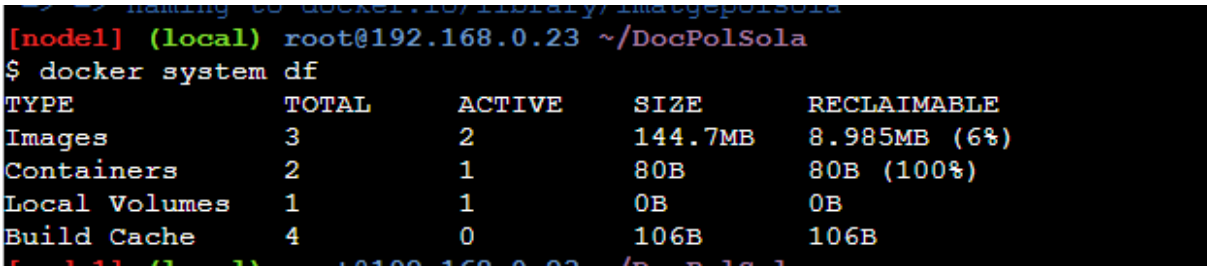
```
l'última directament.

FROM alpine:3.8.5

# Actualitzem el repositori d'Alpine i instalem Apache2, en una
mateixa línia. Al fer-ho en una sola comanda, només afegim una capa a
la imatge Docker.

RUN apk add --update apache2

# Indica quin port s'ha d'exposar

EXPOSE 80

# Indiquem quina comanda s'haurà d'executar en iniciar la imatge

CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Crea la imatge a partir d'aquest Dockerfile. Per fer-ho, executa la següent comanda:

docker build -t ImatgeNomCognom .



1.8. (0,5 punts) Captura de pantalla resultant d'executar la comanda "*docker system df*", que mostra l'espai ocupat per les imatges, els contenidors i els volums:



1.9. (0,5 punts) Captura de pantalla, resultat d'eliminar la imatge *docker amb "docker image rm -f redis*":

```
[node1] (local) root@192.168.0.23 ~/DocPolSola
$ docker image rm -f redis
Untagged: redis:latest
Untagged: redis@sha256:a4b90e7079b67c41bdf4ca6f9d87197079e4c1c3273b7f489a74f2687d85a05e
[node1] (local) root@192.168.0.23 ~/DocPolSola
```

Exercici 2 - Alta disponibilitat

Ara, volem que la nostra aplicació estigui sempre funcionant, mitjançant un clúster de Docker.

Per començar, executarem una imatge de Docker, definida en un fitxer .yml enlloc d'en una comanda. Crea un fitxer anomenat "*docker-compose.yml*" i enganxa el següent:

```yaml
services:
  web:
    image: ubuntu:20.04
    deploy:
      replicas: 4
      resources:
        limits:
          cpus: "0.4"
          memory: '40M'
      restart_policy:
        condition: on-failure
    ports:
      - "8081:80"
    networks:
      - xweb
networks:
  xweb:
```

- Això executarà un màxim de 4 instàncies de la imatge, en un servei anomenat "*web*", limitant cadascuna a un màxim del 40% de la CPU i 40mb de RAM.
- Si algun contenidor es cau, es reiniciarà
- Redireccionarà el port 80 de la imatge al port 8081 del host.

2.1. (1 punt) Inicia aquest contenidor. Per fer-ho primer crearem una swarm de Docker, amb la comanda "*docker swarm init*". Fes captura d'aquesta comanda.

```
$ docker swarm init --advertise-addr 192.168.0.22
Swarm initialized: current node (1itmmbtwi72bwz244fttijdk2) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3o24zlng28zp0b1277toq6agyxu28ysnrglez518lo3m4jy79y-er3vymazdgvo07rhrtz4vr6jk 192.168.0.22:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

Amb el swarm ja iniciat, ara iniciarem els nostres contenidors dins d'aquesta swarm:

docker stack deploy -c docker-compose.yml NomCognom

```
[node2] (local) root@192.168.0.22 ~
$ docker stack deploy -c docker-compose.yml PolSola
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network PolSola_xweb
Creating service PolSola_web
```

2.2 (0,5 punts) Fes una captura de pantalla del resultat de la comanda "*docker service ls*". S'ha de veure el contenidor que has creat.

```
$ docker service ls
ID             NAME          MODE         REPLICAS   IMAGE          PORTS
57hquv2xoss5   PolSola_web   replicated   0/4        ubuntu:20.04   *:8081->80/tcp
```

2.3 (1,75 punts) Ara, executa l'ordre per escalar el servei i augmentar el nombre de rèpliques de 4 a 6.

```
$ docker service scale PolSola_web=6
PolSola_web scaled to 6
overall progress: 0 out of 6 tasks
1/6:
2/6: running   [==================================================>]
3/6:
4/6:
5/6: starting  [==============================================>     ]
6/6:
```

2.4 (1,75 punts) Què és un clúster d'alta disponibilitat?

Un clúster d'alta disponibilitat és un conjunt de servidors (nodes) que treballen conjuntament per assegurar que una aplicació o servei estigui sempre operativa. Si un dels nodes falla, els altres poden assumir-ne la càrrega i garantir que el sistema continua funcionant. En el cas de Docker Swarm, això es gestiona amb rèpliques de serveis i polítiques de reinici automàtic en cas de fallada.