NETWORK & MULTIMEDIA LAB

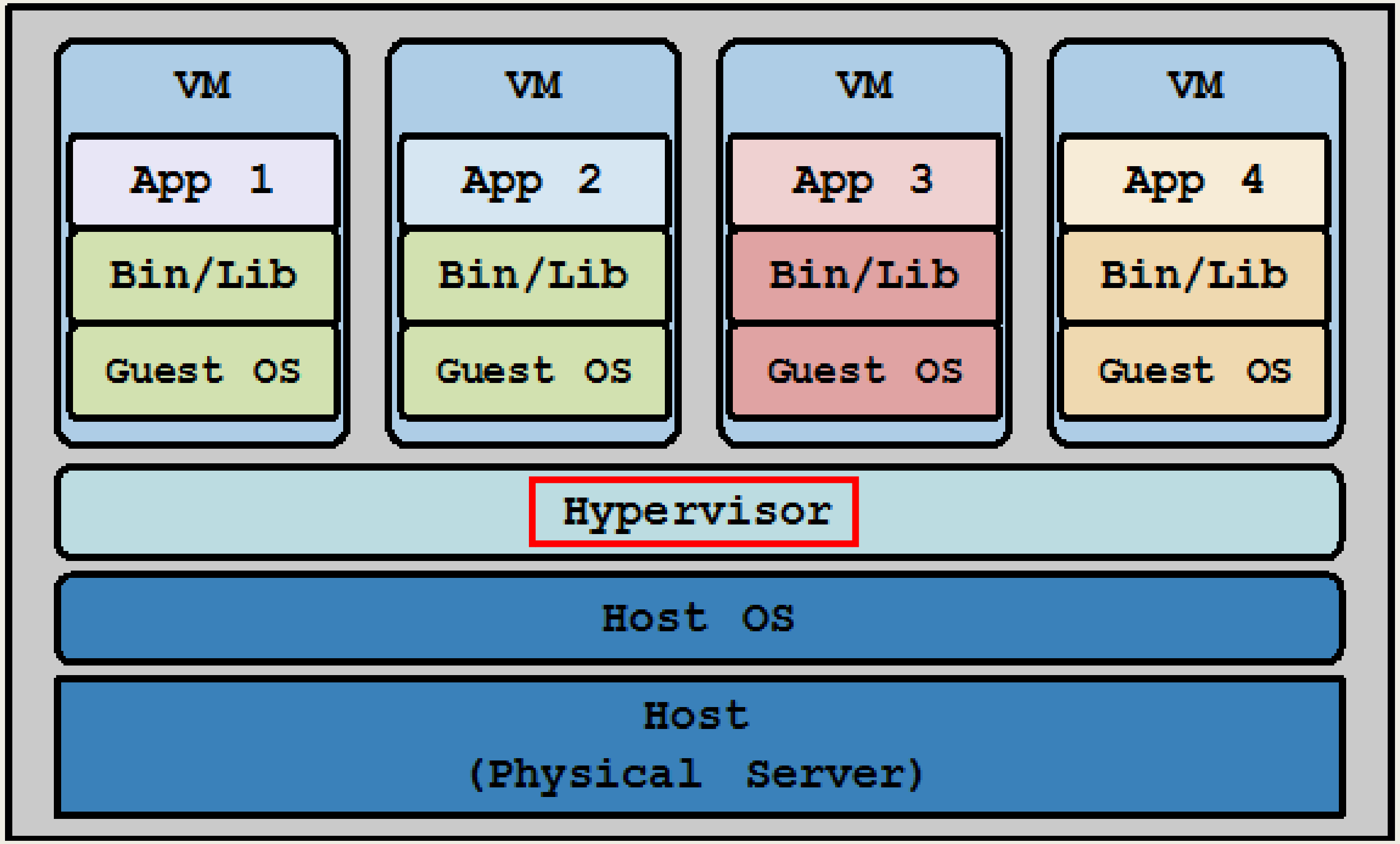
DOCKER

Spring 2022

Outline

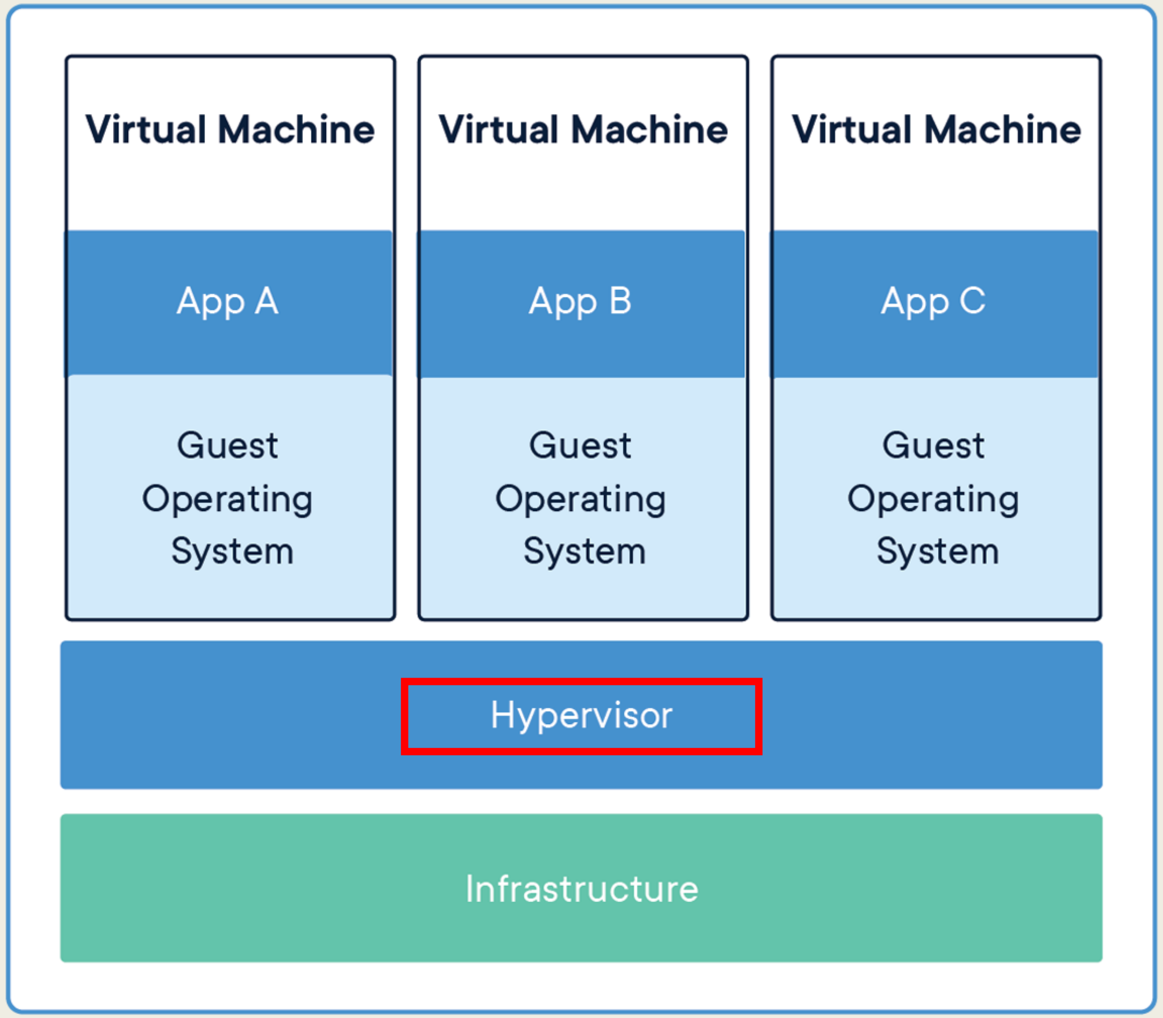
- Containers & Virtual Machines
- Docker
  - Portainer
  - Docker Compose
- Swarm
  - docker-swarm-visualizer

Containers & Virtual Machines



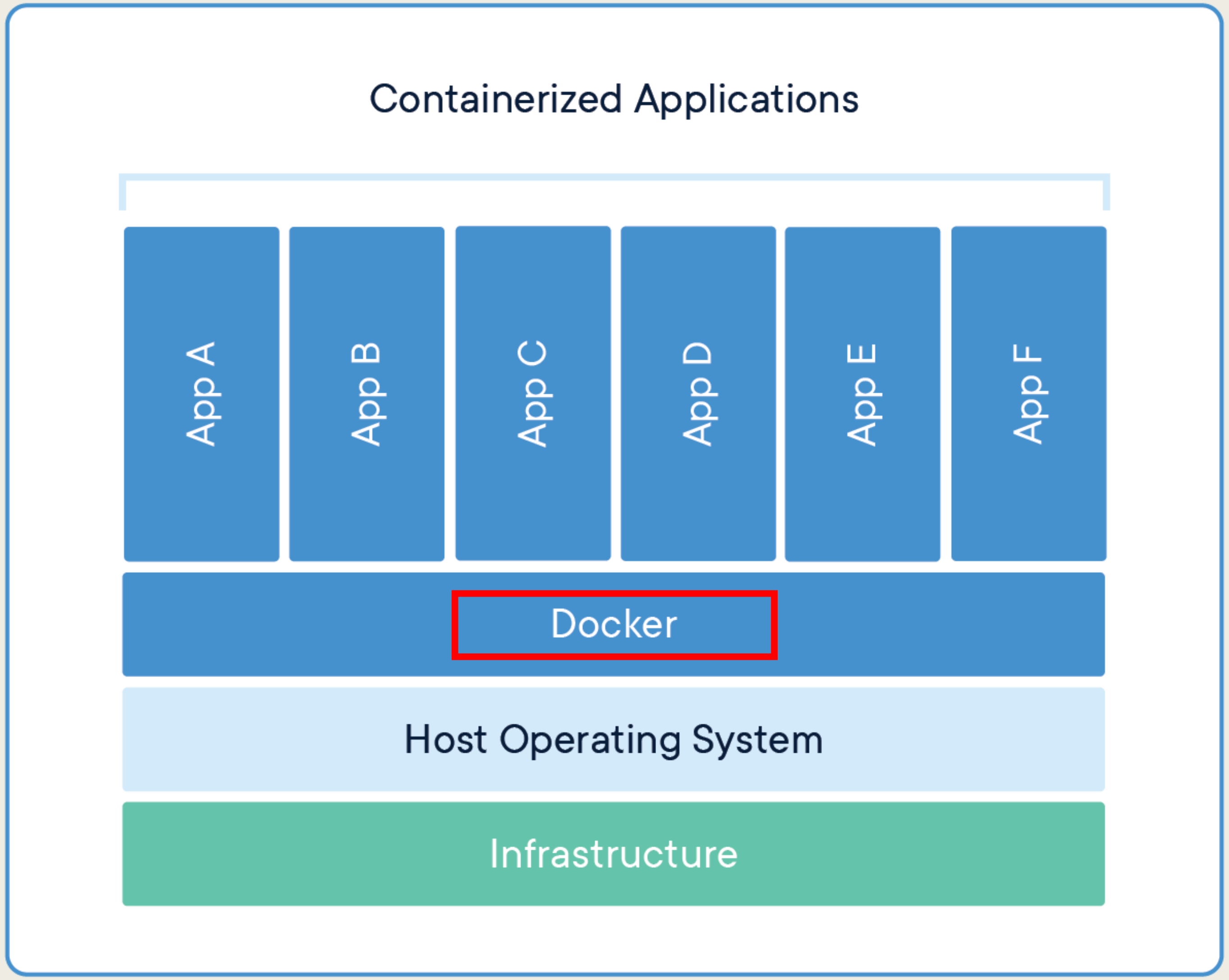
- Virtual Machines
  - Virtualize the hardware

Containers & Virtual Machines



- VMware ESXi
  - Vmware 開發的企業級 Hypervisor
  - 直接安裝在硬體上並且集成了重要的作業系統組件，如內核

Containers & Virtual Machines

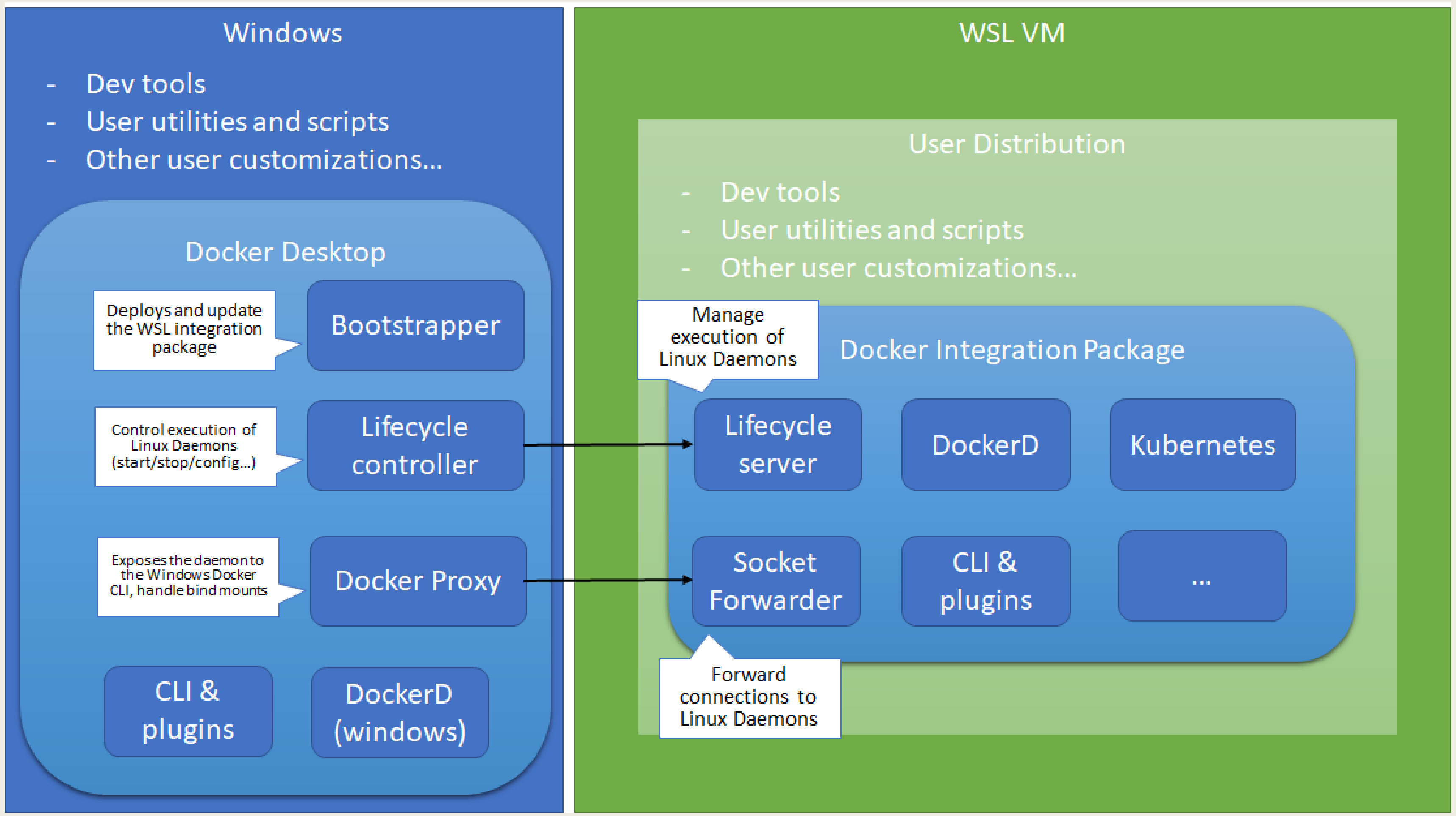


- Containers
  - Virtualize the operating system

Containers virtualize the operating system

- Linux containers are running on Linux
- Windows containers are running on Windows

Docker Desktop for Windows:


Windows
- Dev tools
- User utilities and scripts
- Other user customizations...
Docker Desktop
Deploys and update the WSL integration package
Bootstrapper
Control execution of Linux Daemons (start/stop/config...)
Lifecycle controller
Exposes the daemon to the Windows Docker CLI, handle bind mounts
Docker Proxy
CLI & plugins
DockerD (windows)
WSL VM
User Distribution
- Dev tools
- User utilities and scripts
- Other user customizations...
Manage execution of Linux Daemons
Docker Integration Package
Lifecycle server
DockerD
Kubernetes
Socket Forwarder
CLI & plugins
...
Forward connections to Linux Daemons

DOCKER

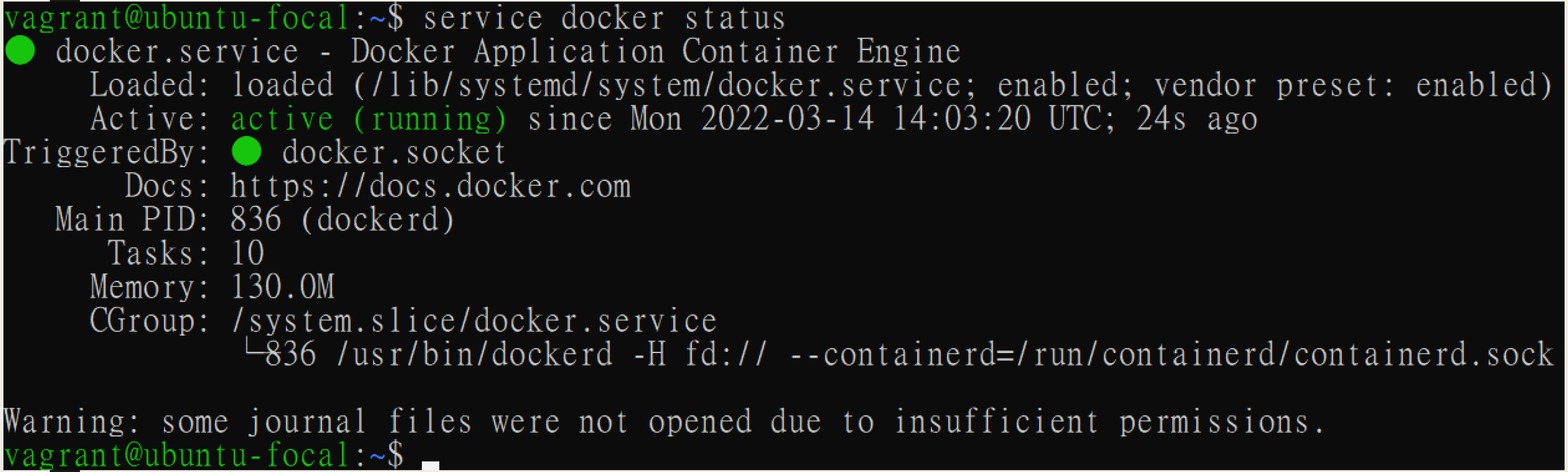
實驗環境

- Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.define "ubuntu" do |u|
    u.vm.box = "ubuntu/focal64"
    u.vm.network "forwarded_port", guest: 9443, host: 9443
    u.vm.network "forwarded_port", guest: 443, host: 443
    u.vm.network "forwarded_port", guest: 80, host: 80
    u.vm.network "forwarded_port", guest: 8080, host: 8080
    u.vm.network "private_network", ip: "192.168.50.2", auto_config:
false # Host-only
    #u.vm.network "public_network"   # Bridge
  end

  config.vm.define "ubuntu2" do |u2|
    u2.vm.box = "ubuntu/focal64"
    u2.vm.network "private_network", ip: "192.168.50.3", auto_config:
false # Host-only
    #u.vm.network "public_network"   # Bridge
  end
end
```

Docker 安裝

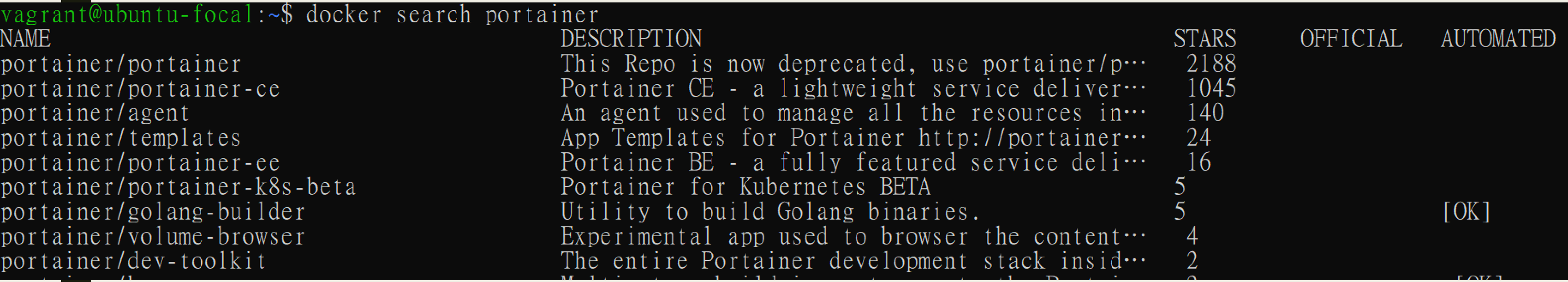
- sudo apt install docker.io
- sudo usermod -aG docker vagrant (重新登入後生效)
- service docker status



```
vagrant@ubuntu-focal:~$ service docker status
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2022-03-14 14:03:20 UTC; 24s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 836 (dockerd)
      Tasks: 10
     Memory: 130.0M
     CGroup: /system.slice/docker.service
             └─836 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Warning: some journal files were not opened due to insufficient permissions.
vagrant@ubuntu-focal:~$
```

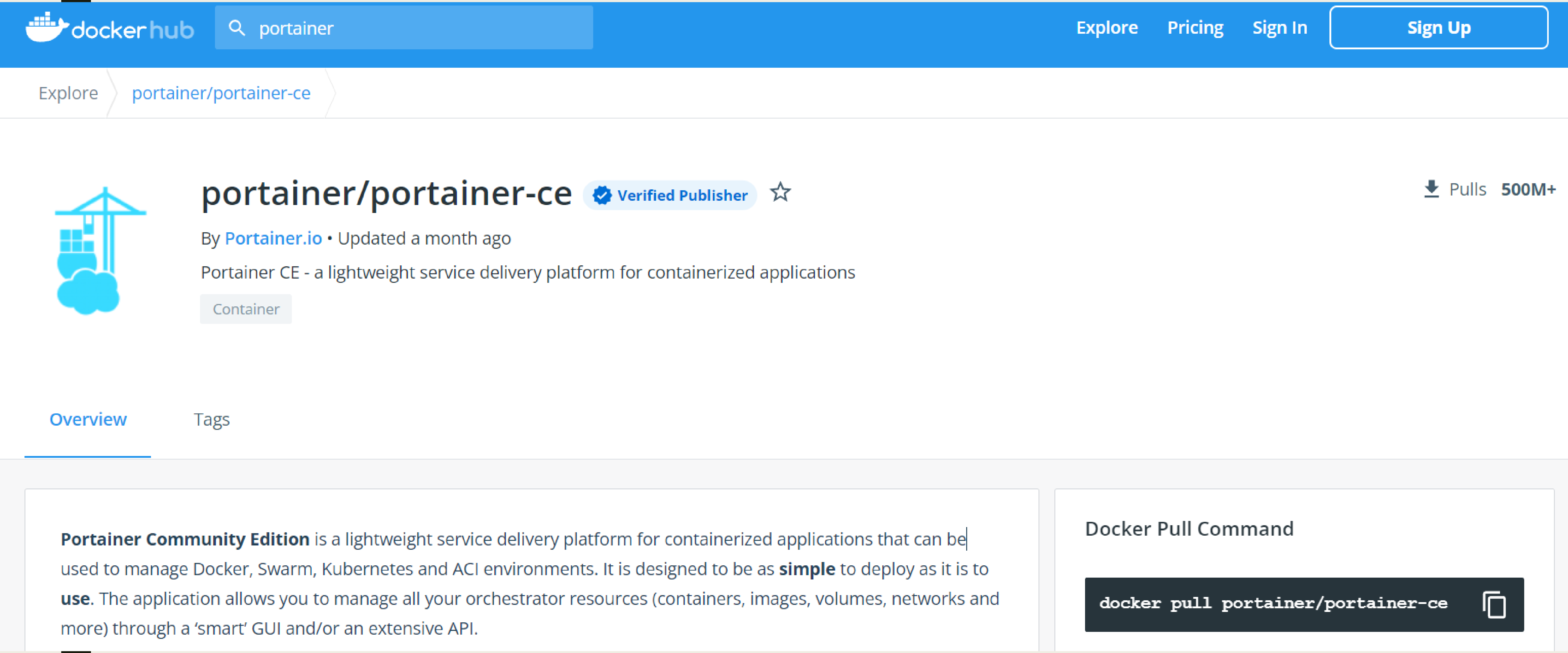
Portainer

- Portainer 是一個開源的 Docker GUI 管理工具
- Portainer 本身也是一個 Docker 容器
  - docker search portainer



```
vagrant@ubuntu-focal:~$ docker search portainer
NAME                             DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
portainer/portainer              This Repo is now deprecated, use portainer/p…   2188
portainer/portainer-ce           Portainer CE - a lightweight service deliver…   1045
portainer/agent                  An agent used to manage all the resources in…   140
portainer/templates              App Templates for Portainer http://portainer…   24
portainer/portainer-ee           Portainer BE - a fully featured service deli…   16
portainer/portainer-k8s-beta     Portainer for Kubernetes BETA                   5
portainer/golang-builder         Utility to build Golang binaries.               5                    [OK]
portainer/volume-browser         Experimental app used to browser the content…   4
portainer/dev-toolkit            The entire Portainer development stack insid…   2
```

Docker Hub



portainer/portainer-ce Verified Publisher

Pulls 500M+

By Portainer.io • Updated a month ago

Portainer CE - a lightweight service delivery platform for containerized applications

Container

Overview Tags

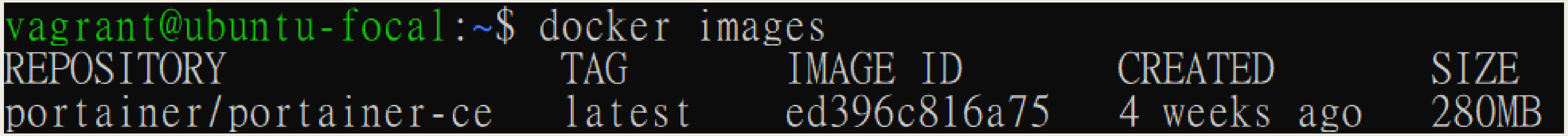
Portainer Community Edition is a lightweight service delivery platform for containerized applications that can be used to manage Docker, Swarm, Kubernetes and ACI environments. It is designed to be as **simple** to deploy as it is to **use**. The application allows you to manage all your orchestrator resources (containers, images, volumes, networks and more) through a 'smart' GUI and/or an extensive API.

Docker Pull Command

```
docker pull portainer/portainer-ce
```

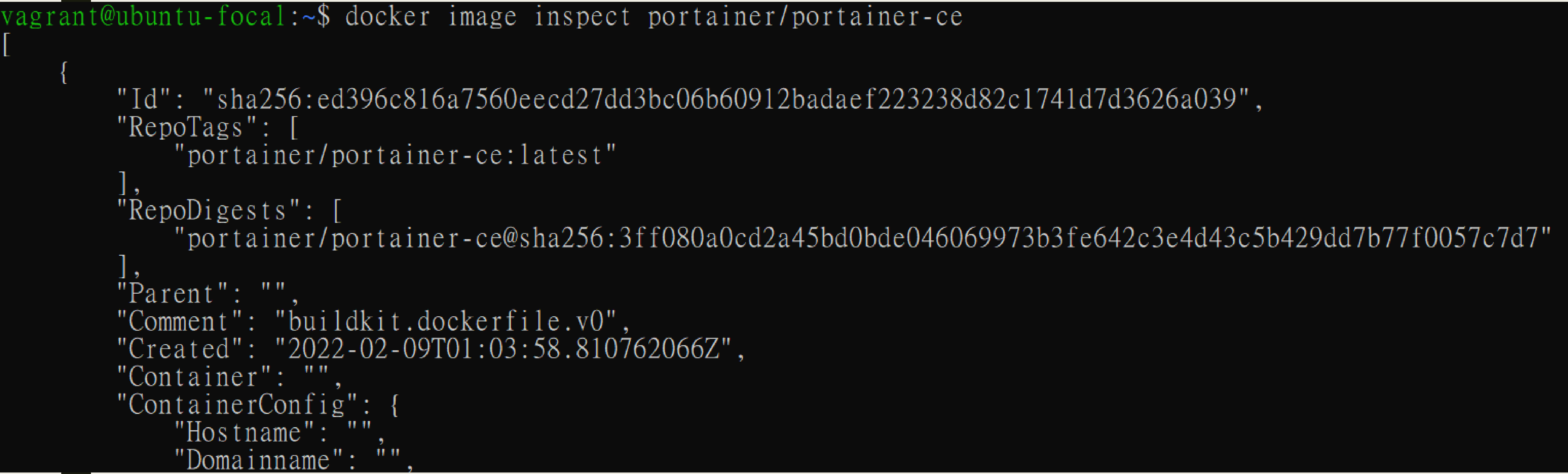
Portainer 安裝

- docker pull portainer/portainer-ce
- docker images



```
vagrant@ubuntu-focal:~$ docker images
REPOSITORY              TAG       IMAGE ID       CREATED        SIZE
portainer/portainer-ce  latest    ed396c816a75   4 weeks ago    280MB
```

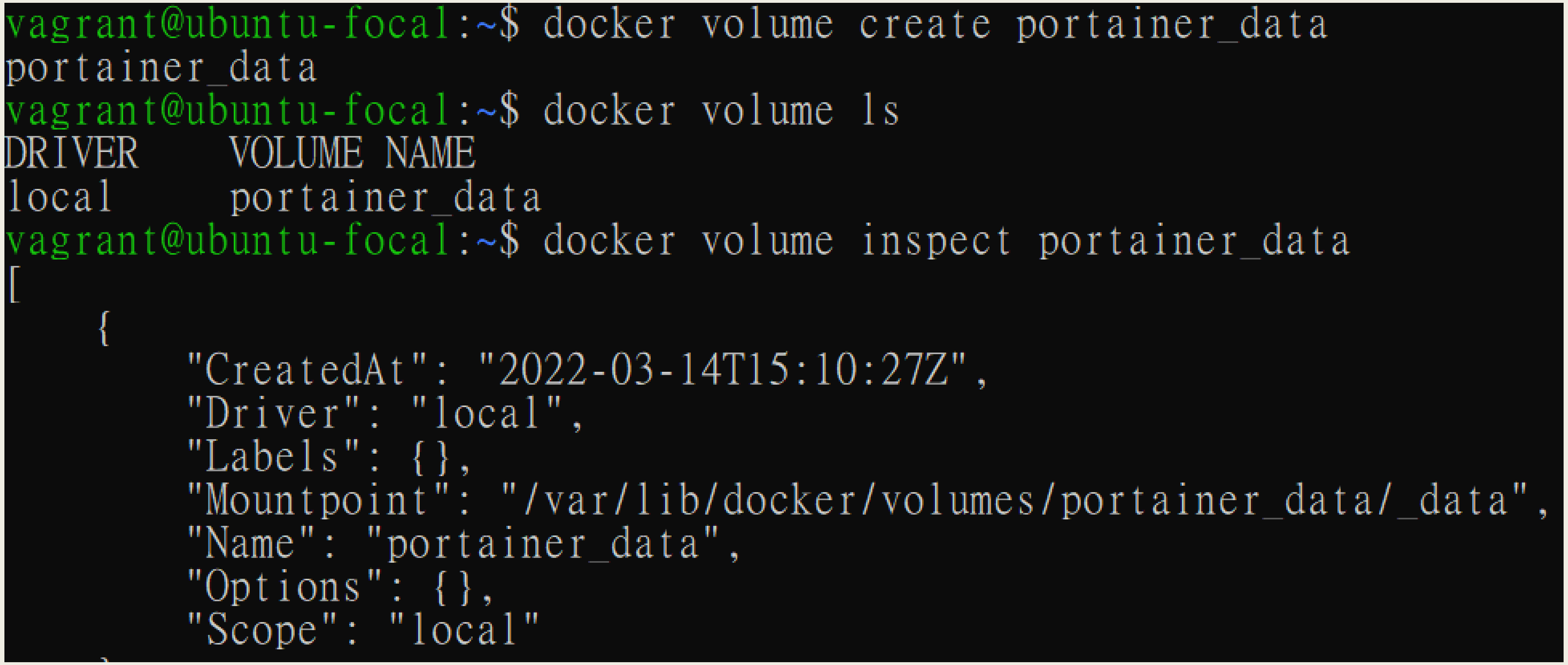
- docker image inspect portainer/portainer-ce



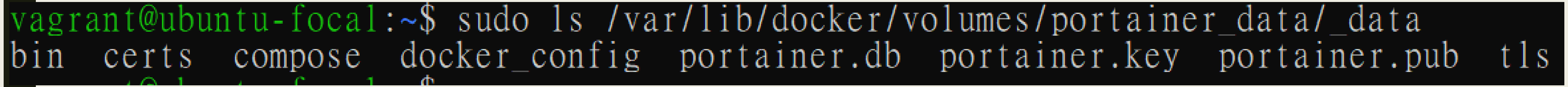
```
vagrant@ubuntu-focal:~$ docker image inspect portainer/portainer-ce
[
    {
        "Id": "sha256:ed396c816a7560eecd27dd3bc06b60912badaef223238d82c1741d7d3626a039",
        "RepoTags": [
            "portainer/portainer-ce:latest"
        ],
        "RepoDigests": [
            "portainer/portainer-ce@sha256:3ff080a0cd2a45bd0bde046069973b3fe642c3e4d43c5b429dd7b77f0057c7d7"
        ],
        "Parent": "",
        "Comment": "buildkit.dockerfile.v0",
        "Created": "2022-02-09T01:03:58.810762066Z",
        "Container": "",
        "ContainerConfig": {
            "Hostname": "",
            "Domainname": "",
```

Portainer 安裝

- docker volume create portainer_data
- docker volume ls
- docker volume inspect portainer_data



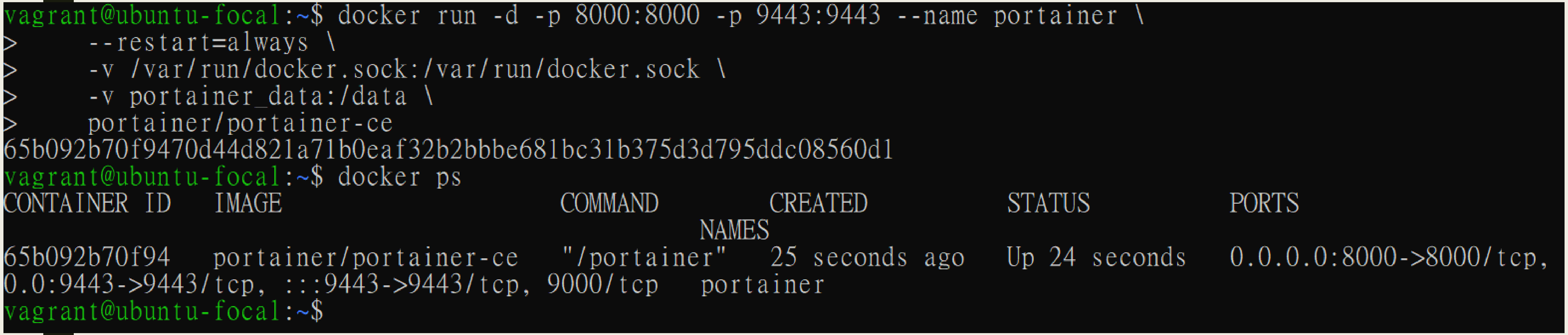
```
vagrant@ubuntu-focal:~$ docker volume create portainer_data
portainer_data
vagrant@ubuntu-focal:~$ docker volume ls
DRIVER    VOLUME NAME
local     portainer_data
vagrant@ubuntu-focal:~$ docker volume inspect portainer_data
[
    {
        "CreatedAt": "2022-03-14T15:10:27Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/portainer_data/_data",
        "Name": "portainer_data",
        "Options": {},
        "Scope": "local"
```



```
vagrant@ubuntu-focal:~$ sudo ls /var/lib/docker/volumes/portainer_data/_data
bin  certs  compose  docker_config  portainer.db  portainer.key  portainer.pub  tls
```

Portainer 啟動

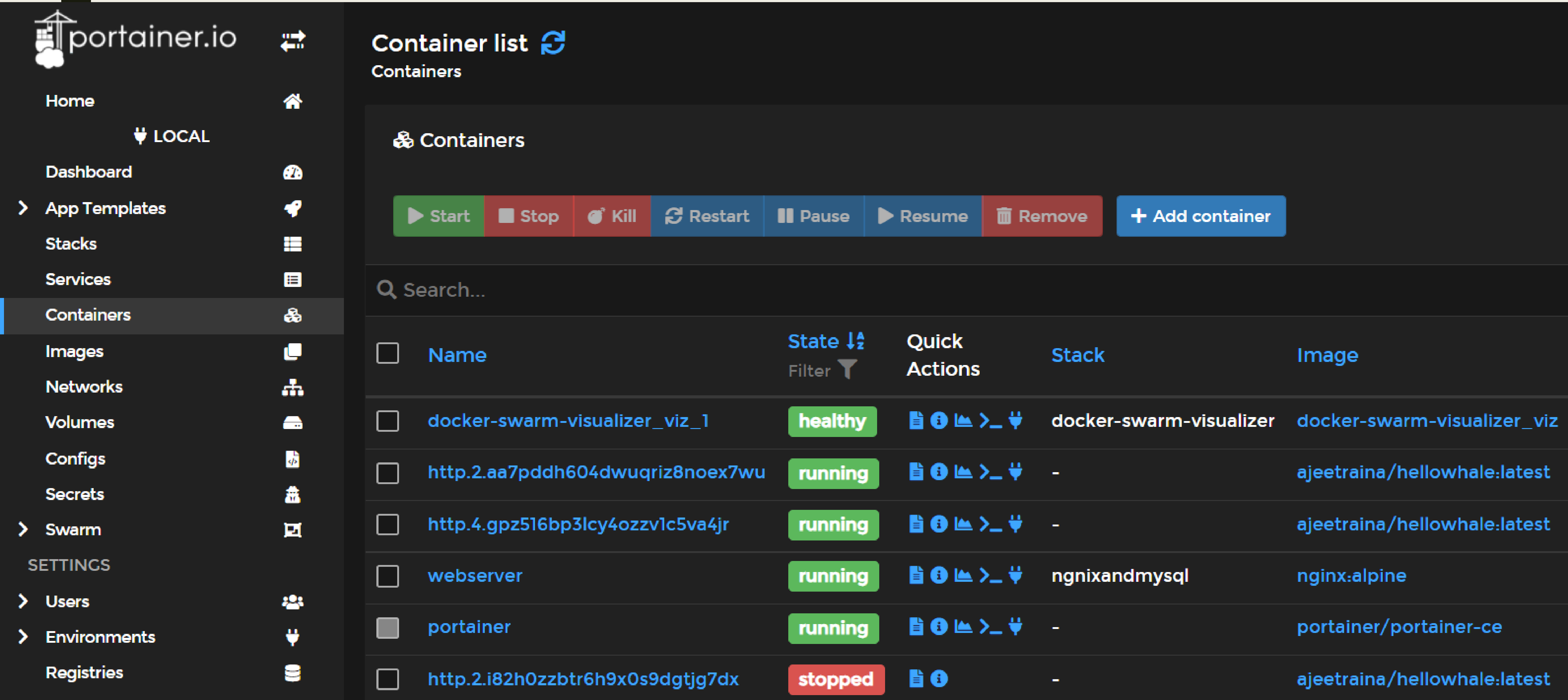
- docker run -d -p 8000:8000 -p 9443:9443 --name portainer \
  --restart=always \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v portainer_data:/data \
  portainer/portainer-ce
- docker ps (列出運行中的容器)



```
vagrant@ubuntu-focal:~$ docker run -d -p 8000:8000 -p 9443:9443 --name portainer \
>     --restart=always \
>     -v /var/run/docker.sock:/var/run/docker.sock \
>     -v portainer_data:/data \
>     portainer/portainer-ce
65b092b70f9470d44d821a71b0eaf32b2bbbe681bc31b375d3d795ddc08560d1
vagrant@ubuntu-focal:~$ docker ps
CONTAINER ID   IMAGE                    COMMAND         CREATED          STATUS          PORTS
                                   NAMES
65b092b70f94   portainer/portainer-ce   "/portainer"    25 seconds ago   Up 24 seconds   0.0.0.0:8000->8000/tcp,
0.0:9443->9443/tcp, :::9443->9443/tcp, 9000/tcp   portainer
vagrant@ubuntu-focal:~$
```

Portainer

- https://127.0.0.1:9443/



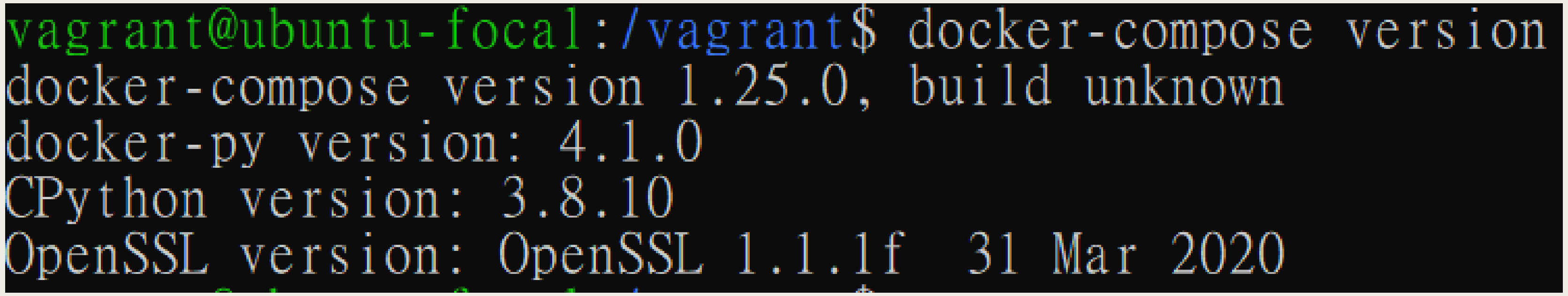
Container list
Containers

Containers

	Name	State Filter	Quick Actions	Stack	Image
☐	docker-swarm-visualizer_viz_1	healthy		docker-swarm-visualizer	docker-swarm-visualizer_viz
☐	http.2.aa7pddh604dwuqriz8noex7wu	running		-	ajeetraina/hellowhale:latest
☐	http.4.gpz516bp3lcy4ozzv1c5va4jr	running		-	ajeetraina/hellowhale:latest
☐	webserver	running		ngnixandmysql	nginx:alpine
☐	portainer	running		-	portainer/portainer-ce
☐	http.2.i82h0zzbtr6h9x0s9dgtjg7dx	stopped		-	ajeetraina/hellowhale:latest

Docker Compose

- Docker Compose 是用來運行多個 docker container 的工具，透過在YAML檔定義要運行的服務(service)，便能使用一個指令啟動配置檔中定義的服務。
- Docker Compose 安裝
  - sudo apt install docker-compose
  - docker-compose version



```
vagrant@ubuntu-focal:/vagrant$ docker-compose version
docker-compose version 1.25.0, build unknown
docker-py version: 4.1.0
CPython version: 3.8.10
OpenSSL version: OpenSSL 1.1.1f  31 Mar 2020
```

Docker Compose (ngnix and mysql)

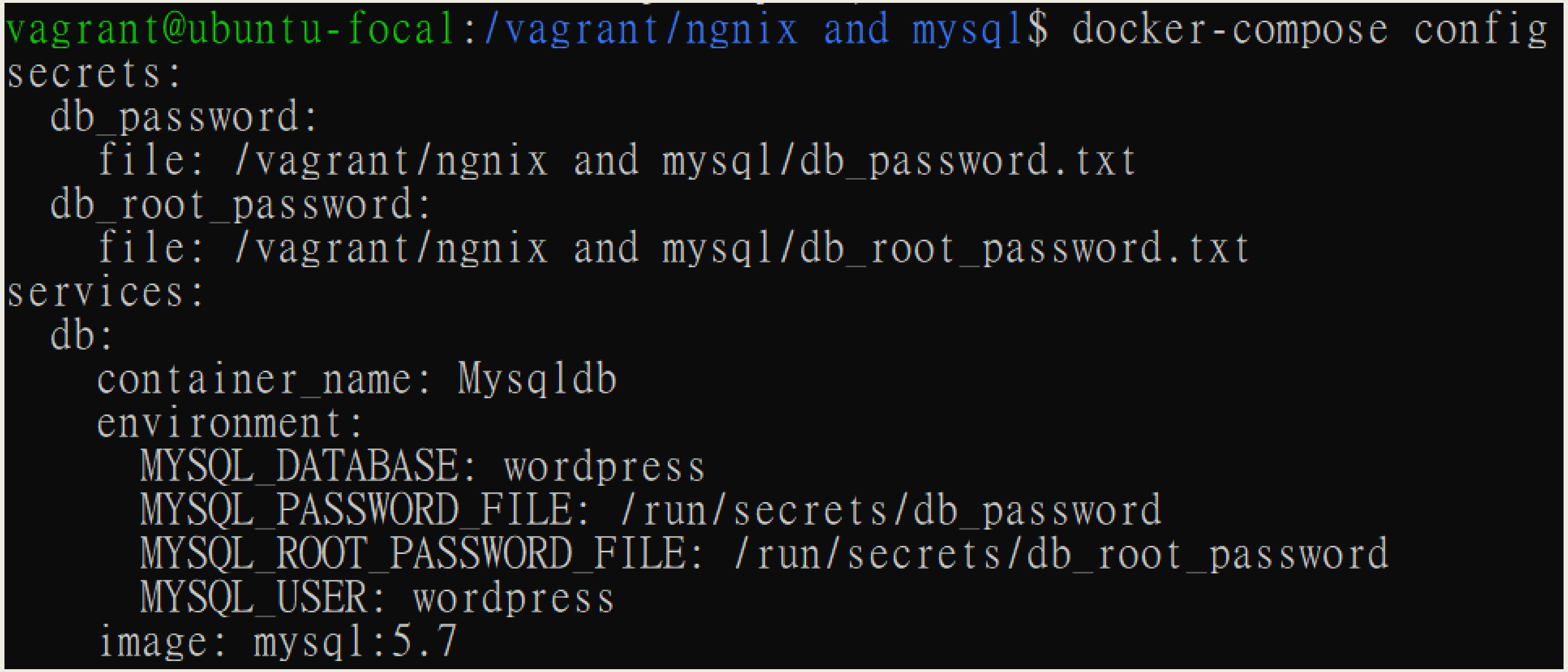
- Create docker-compose.yml

- openssl rand -base64 32 > db_password.txt
- openssl rand -base64 32 > db_root_password.txt

```
version: '3.1'
services:
  #Nginx Service
   webserver:
     image: nginx:alpine
     container_name: webserver
     restart: unless-stopped
     ports:
       - "80:80"
       - "443:443"
   #Mysql DB
   db:
     image: mysql:5.7
     container_name: Mysqldb
     restart: unless-stopped
     volumes:
       - db_data:/var/lib/mysql
     ports:
       - "3306:3306"
     environment:
       MYSQL_ROOT_PASSWORD_FILE: /run/secrets/db_root_password
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD_FILE: /run/secrets/db_password
     secrets:
       - db_root_password
       - db_password
secrets:
   db_password:
     file: db_password.txt
   db_root_password:
     file: db_root_password.txt

volumes:
    db_data:
```

Docker Compose (ngnix and mysql)

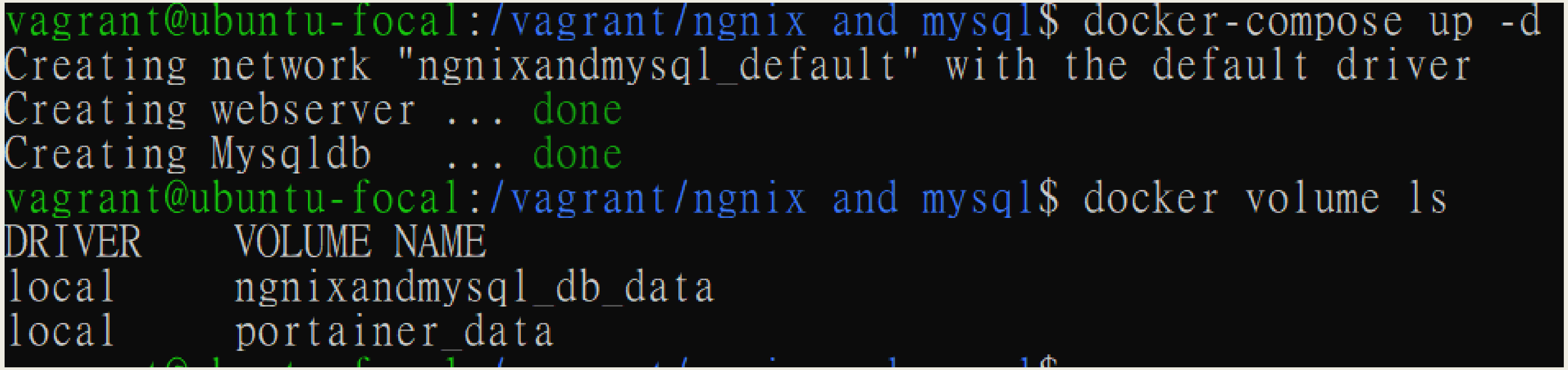
- docker-compose config (確認 config 格式正確)



```
vagrant@ubuntu-focal:/vagrant/ngnix and mysql$ docker-compose config
secrets:
  db_password:
    file: /vagrant/ngnix and mysql/db_password.txt
  db_root_password:
    file: /vagrant/ngnix and mysql/db_root_password.txt
services:
  db:
    container_name: Mysqldb
    environment:
      MYSQL_DATABASE: wordpress
      MYSQL_PASSWORD_FILE: /run/secrets/db_password
      MYSQL_ROOT_PASSWORD_FILE: /run/secrets/db_root_password
      MYSQL_USER: wordpress
    image: mysql:5.7
```

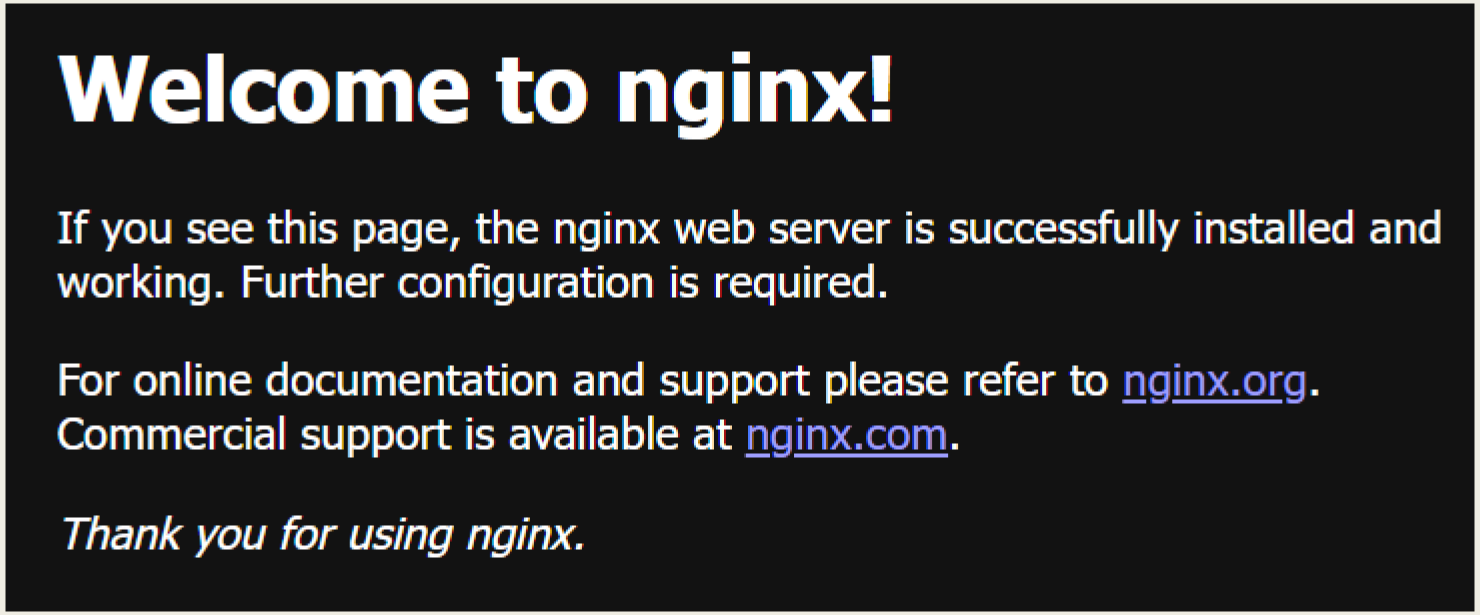
Docker Compose (ngnix and mysql)

- docker-compose up -d



```
vagrant@ubuntu-focal:/vagrant/ngnix and mysql$ docker-compose up -d
Creating network "ngnixandmysql_default" with the default driver
Creating webserver ... done
Creating Mysqldb   ... done
vagrant@ubuntu-focal:/vagrant/ngnix and mysql$ docker volume ls
DRIVER    VOLUME NAME
local     ngnixandmysql_db_data
local     portainer_data
```

Ngnix @ 127.0.0.1



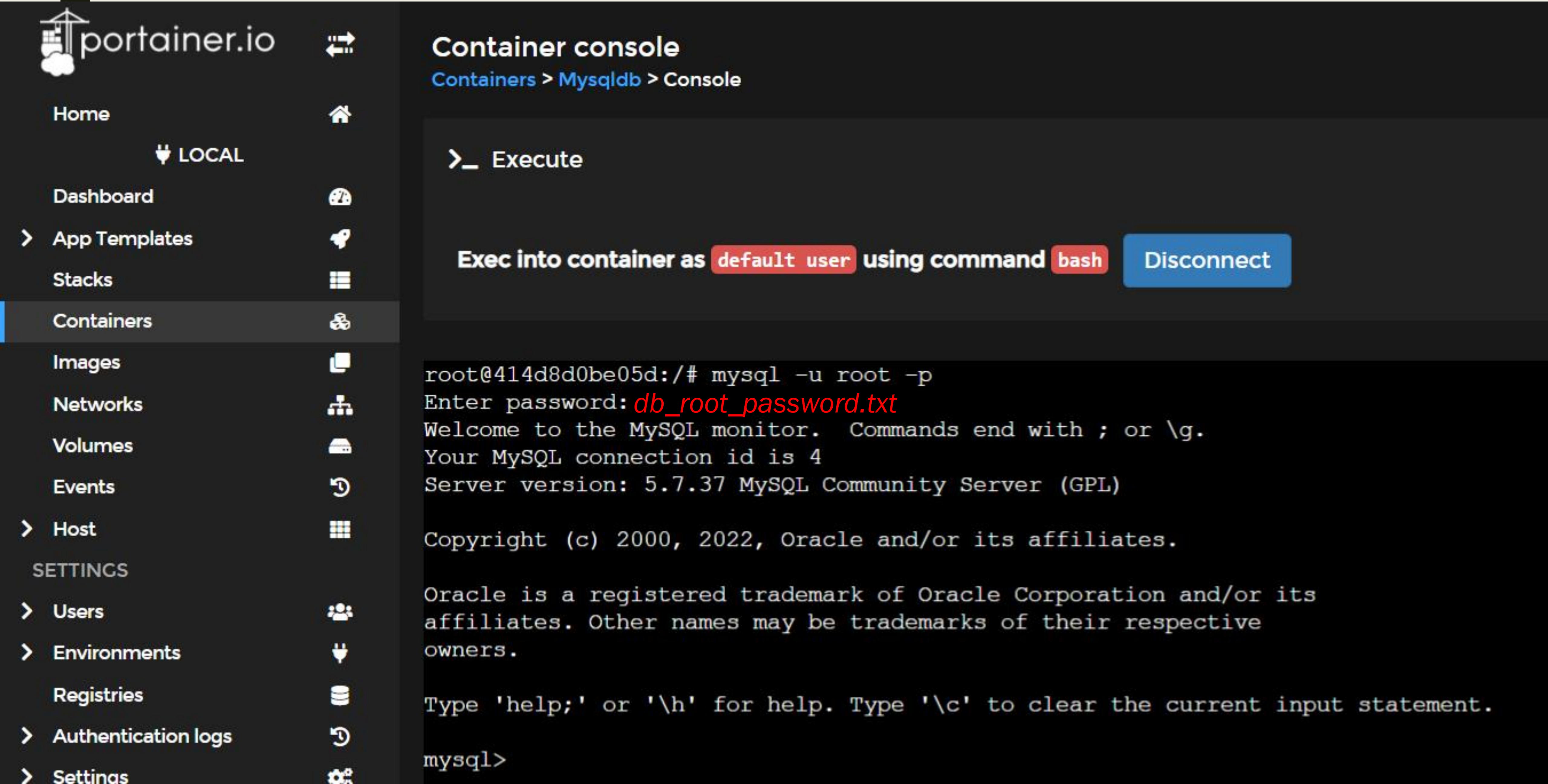
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Mysqldb

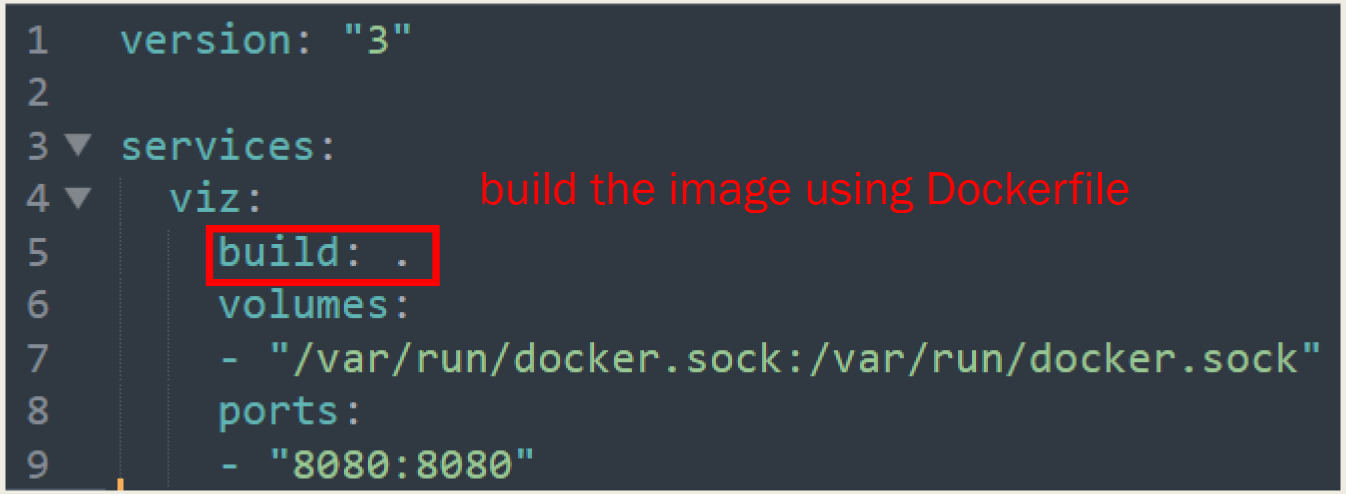


```
root@414d8d0be05d:/# mysql -u root -p
Enter password: db_root_password.txt
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.37 MySQL Community Server (GPL)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

SWARM

docker-swarm-visualizer

- git clone https://github.com/dockersamples/docker-swarm-visualizer
- cd docker-swarm-visualizer
- docker-compose up –d

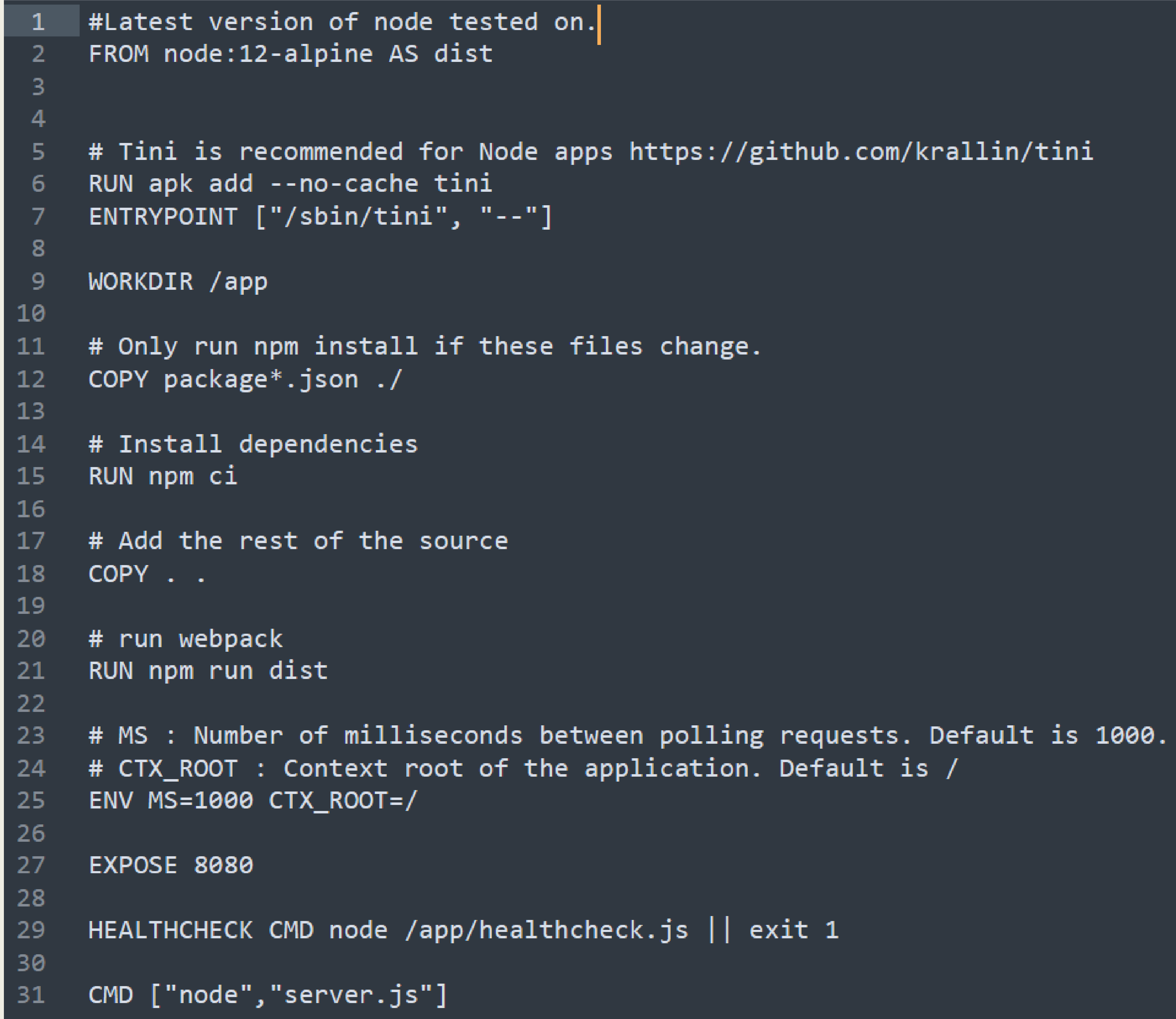


```
version: "3"

services:
  viz:
    build: .
    volumes:
    - "/var/run/docker.sock:/var/run/docker.sock"
    ports:
    - "8080:8080"
```

build the image using Dockerfile

Dockerfile

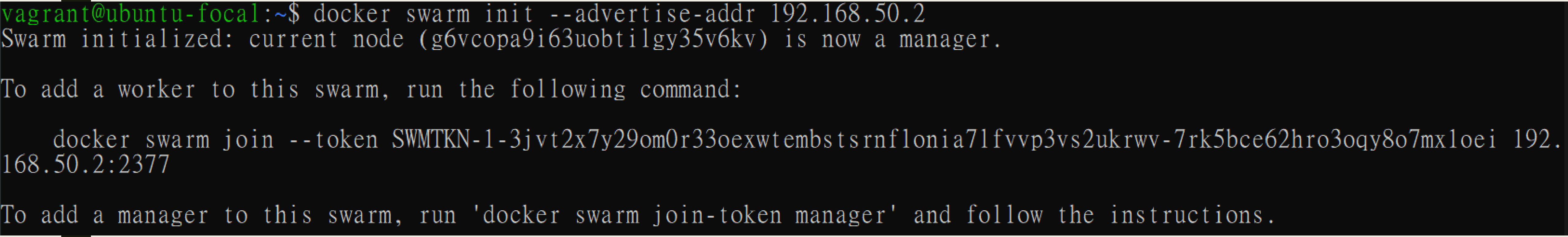
建構 docker images 的腳本



```
#Latest version of node tested on.
FROM node:12-alpine AS dist


# Tini is recommended for Node apps https://github.com/krallin/tini
RUN apk add --no-cache tini
ENTRYPOINT ["/sbin/tini", "--"]

WORKDIR /app

# Only run npm install if these files change.
COPY package*.json ./

# Install dependencies
RUN npm ci

# Add the rest of the source
COPY . .

# run webpack
RUN npm run dist

# MS : Number of milliseconds between polling requests. Default is 1000.
# CTX_ROOT : Context root of the application. Default is /
ENV MS=1000 CTX_ROOT=/

EXPOSE 8080

HEALTHCHECK CMD node /app/healthcheck.js || exit 1

CMD ["node","server.js"]
```

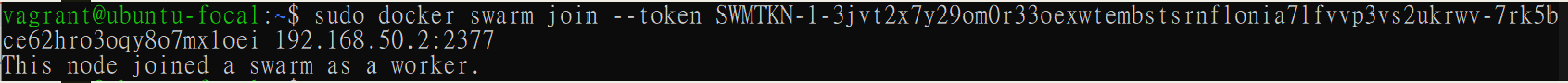
Add Manager & Worker

- docker swarm init --advertise-addr 192.168.50.2 (Add Manager on VM1)



```
vagrant@ubuntu-focal:~$ docker swarm init --advertise-addr 192.168.50.2
Swarm initialized: current node (g6vcopa9i63uobtilgy35v6kv) is now a manager.

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

    docker swarm join --token SWMTKN-1-3jvt2x7y29om0r33oexwtembstsrnflonia71fvvp3vs2ukrwv-7rk5bce62hro3oqy8o7mxloei 192.
168.50.2:2377

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

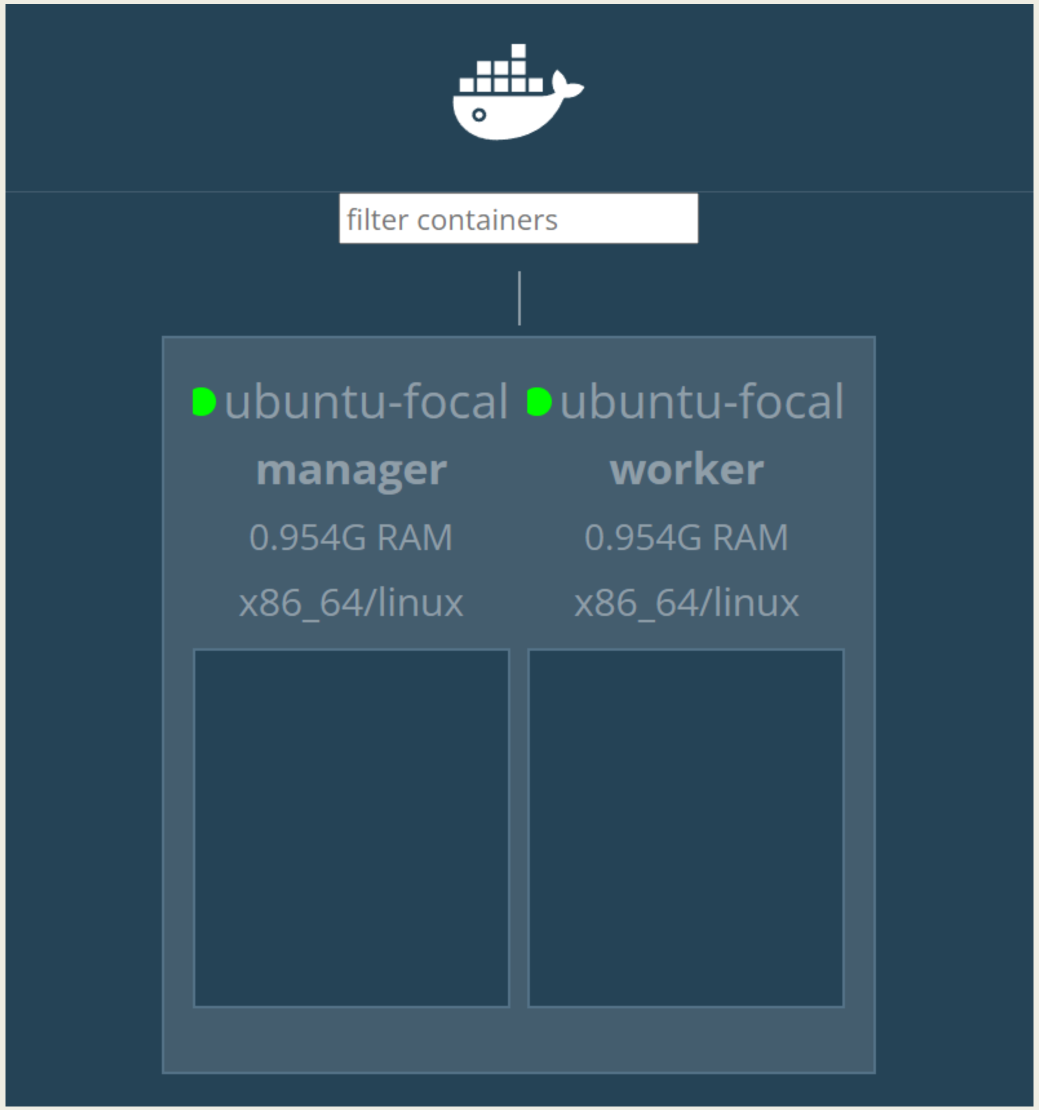
- (Add Worker on VM2)



```
vagrant@ubuntu-focal:~$ sudo docker swarm join --token SWMTKN-1-3jvt2x7y29om0r33oexwtembstsrnflonia71fvvp3vs2ukrwv-7rk5b
ce62hro3oqy8o7mxloei 192.168.50.2:2377
This node joined a swarm as a worker.
```

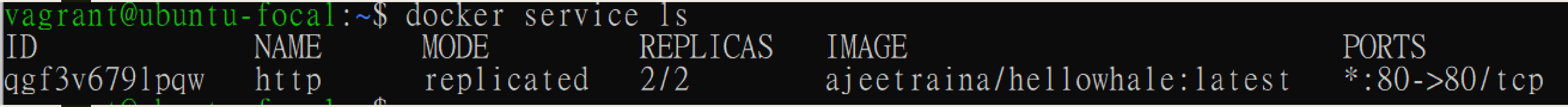
http://127.0.0.1:8080/

- Manager
  - 192.168.50.2
- Worker
  - 192.168.50.3



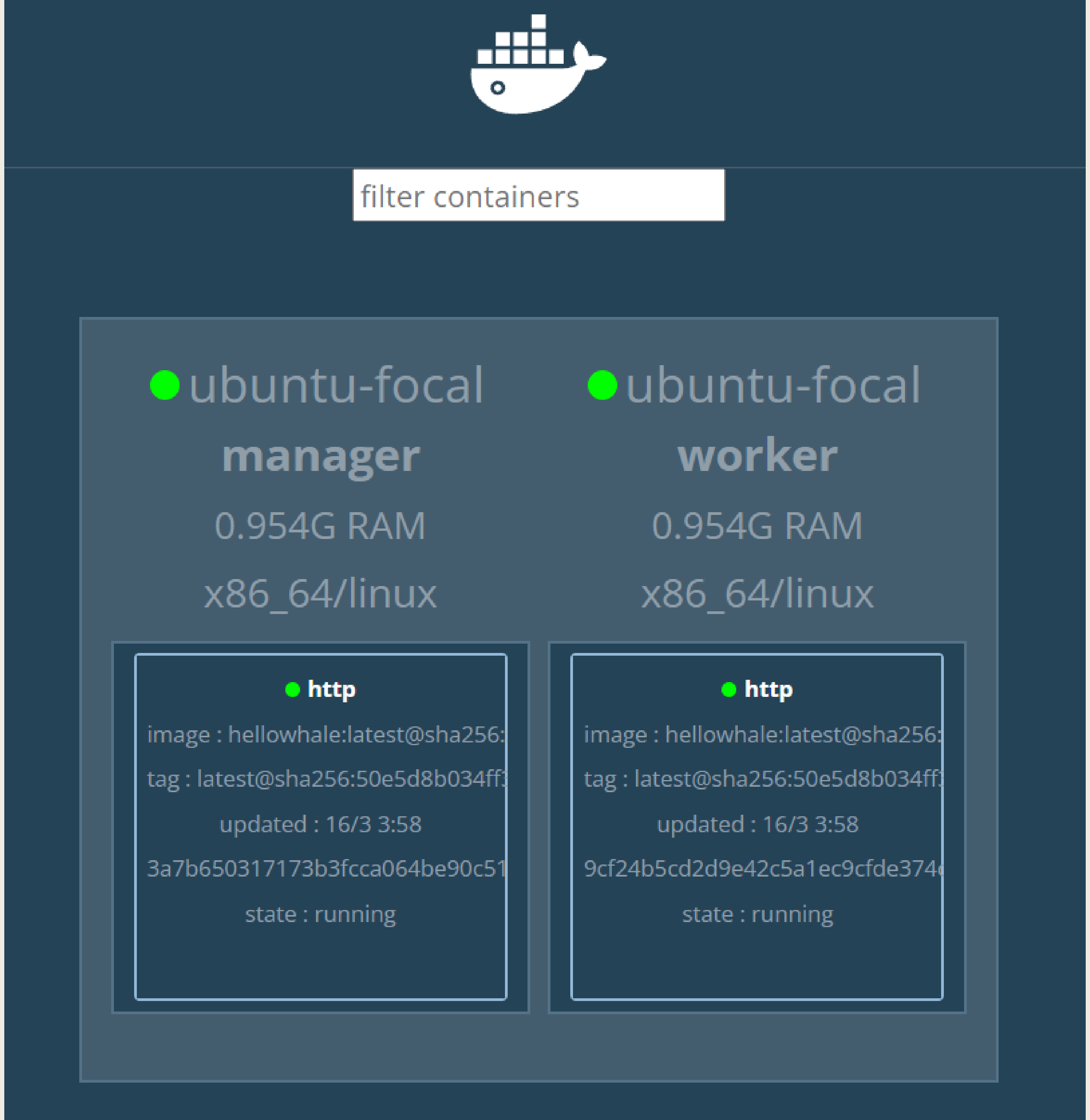
Create docker service

- docker network create -d overlay collabnet
- docker service create --name http --network collabnet --replicas 2 -p 80:80 ajeetraina/hellowhale
- docker service ls



```
vagrant@ubuntu-focal:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE                          PORTS
qgf3v6791pqw   http      replicated   2/2        ajeetraina/hellowhale:latest   *:80->80/tcp
```

- --replicas 2



● ubuntu-focal
manager
0.954G RAM
x86_64/linux

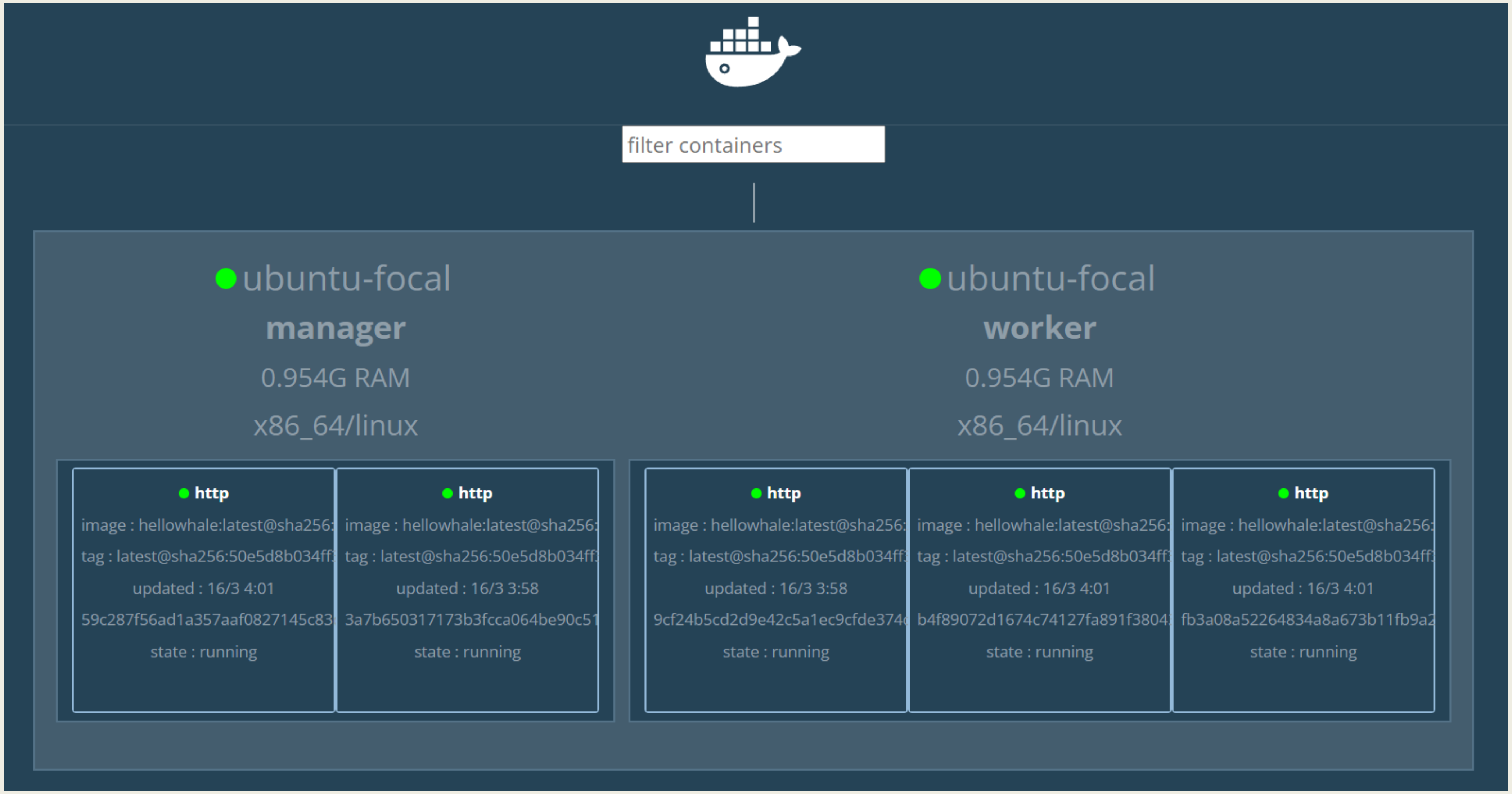
● http
image : hellowhale:latest@sha256:
tag : latest@sha256:50e5d8b034ff
updated : 16/3 3:58
3a7b650317173b3fcca064be90c51
state : running

● ubuntu-focal
worker
0.954G RAM
x86_64/linux

● http
image : hellowhale:latest@sha256:
tag : latest@sha256:50e5d8b034ff
updated : 16/3 3:58
9cf24b5cd2d9e42c5a1ec9cfde374
state : running

Scale our http service to be running across five containers.

- docker service scale http=5



filter containers

● ubuntu-focal
manager
0.954G RAM
x86_64/linux

● http
image : hellowhale:latest@sha256:
tag : latest@sha256:50e5d8b034ff
updated : 16/3 4:01
59c287f56ad1a357aaf0827145c83
state : running

● http
image : hellowhale:latest@sha256:
tag : latest@sha256:50e5d8b034ff
updated : 16/3 3:58
3a7b650317173b3fcca064be90c51
state : running

● ubuntu-focal
worker
0.954G RAM
x86_64/linux

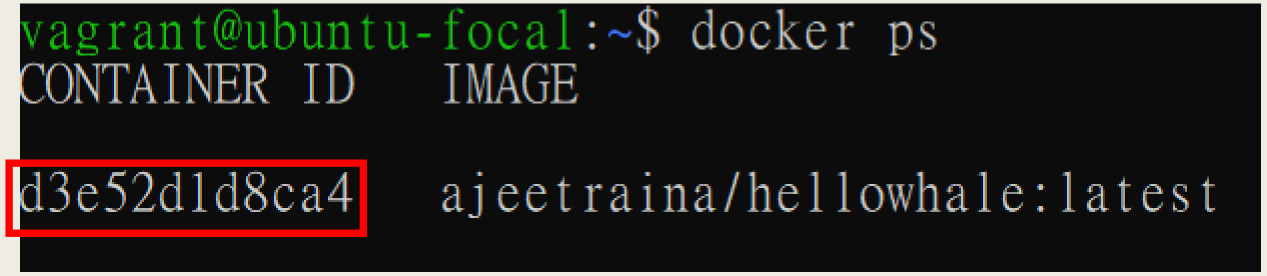
● http
image : hellowhale:latest@sha256:
tag : latest@sha256:50e5d8b034ff
updated : 16/3 3:58
9cf24b5cd2d9e42c5a1ec9cfde374
state : running

● http
image : hellowhale:latest@sha256:
tag : latest@sha256:50e5d8b034ff
updated : 16/3 4:01
b4f89072d1674c74127fa891f3804
state : running

● http
image : hellowhale:latest@sha256:
tag : latest@sha256:50e5d8b034ff
updated : 16/3 4:01
fb3a08a52264834a8a673b11fb9a2
state : running

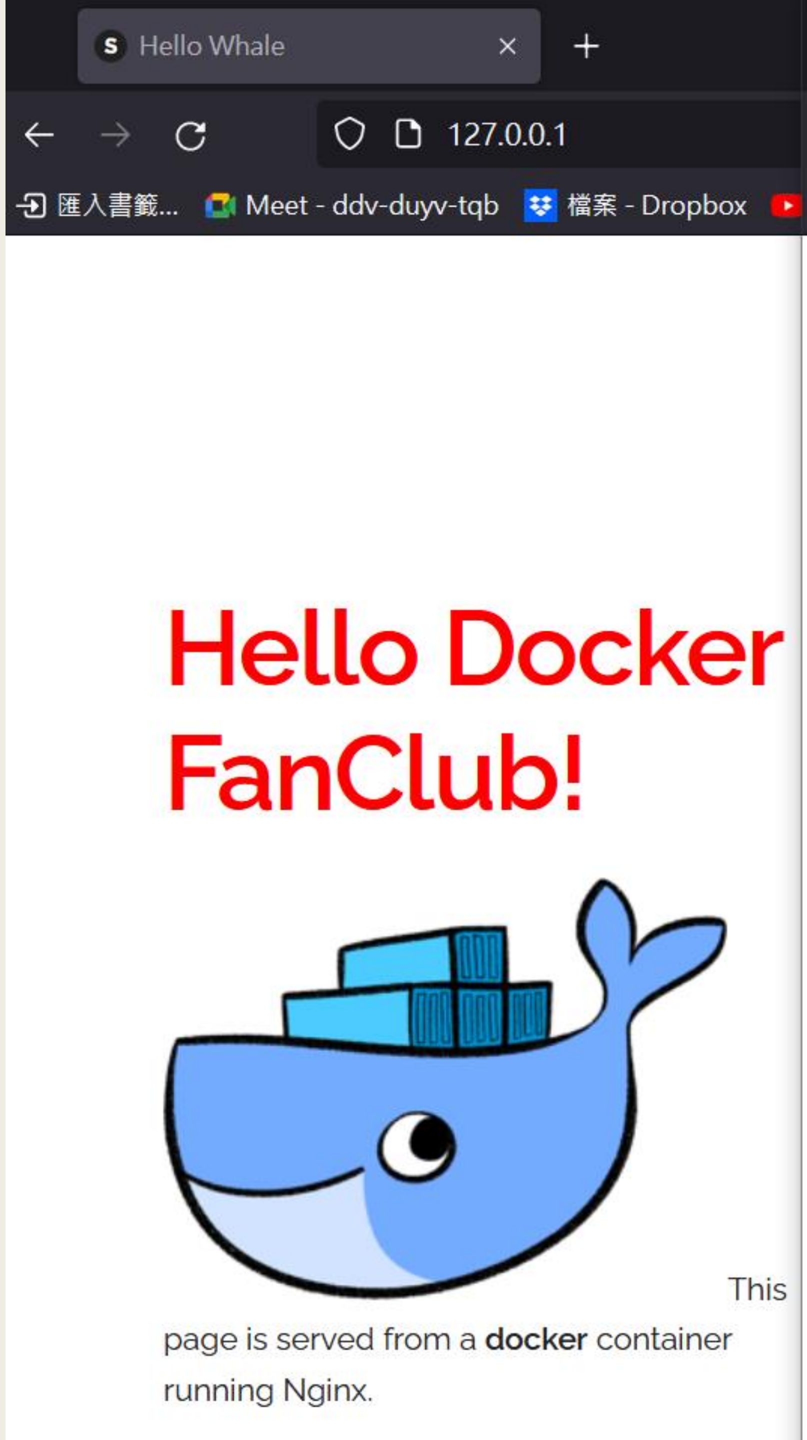
更改網頁內容，確認

- docker ps



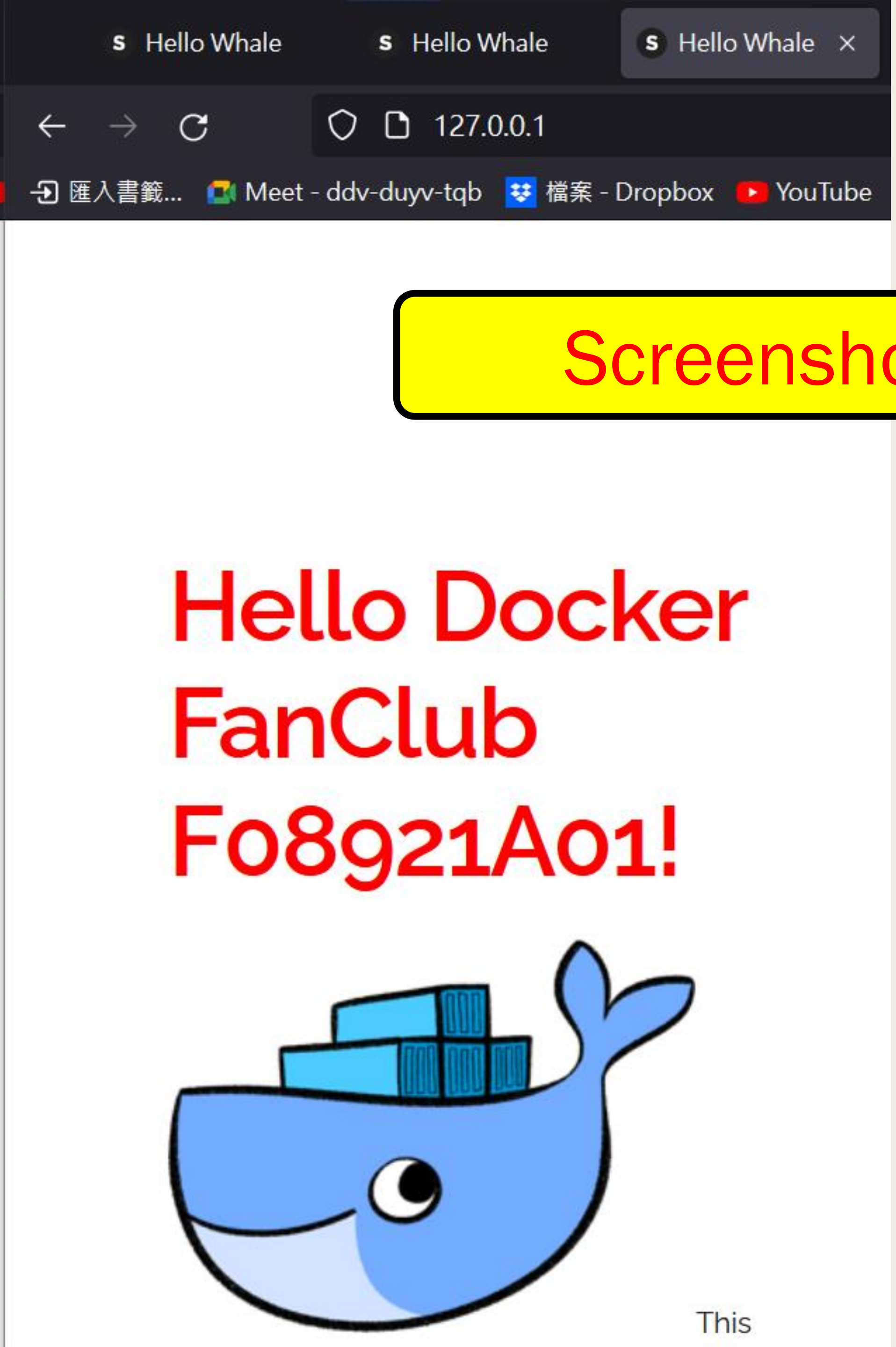
```
vagrant@ubuntu-focal:~$ docker ps
CONTAINER ID   IMAGE

d3e52d1d8ca4   ajeetraina/hellowhale:latest
```

- docker exec -it <container> bash
- vim /usr/share/nginx/html/index.html
- ^P^Q (detach from container)



Hello Docker FanClub!

This page is served from a **docker** container running Nginx.



Hello Docker FanClub F08921A01!

This

Screenshot-01

HW (5pt)

- 上傳 Screenshot-01