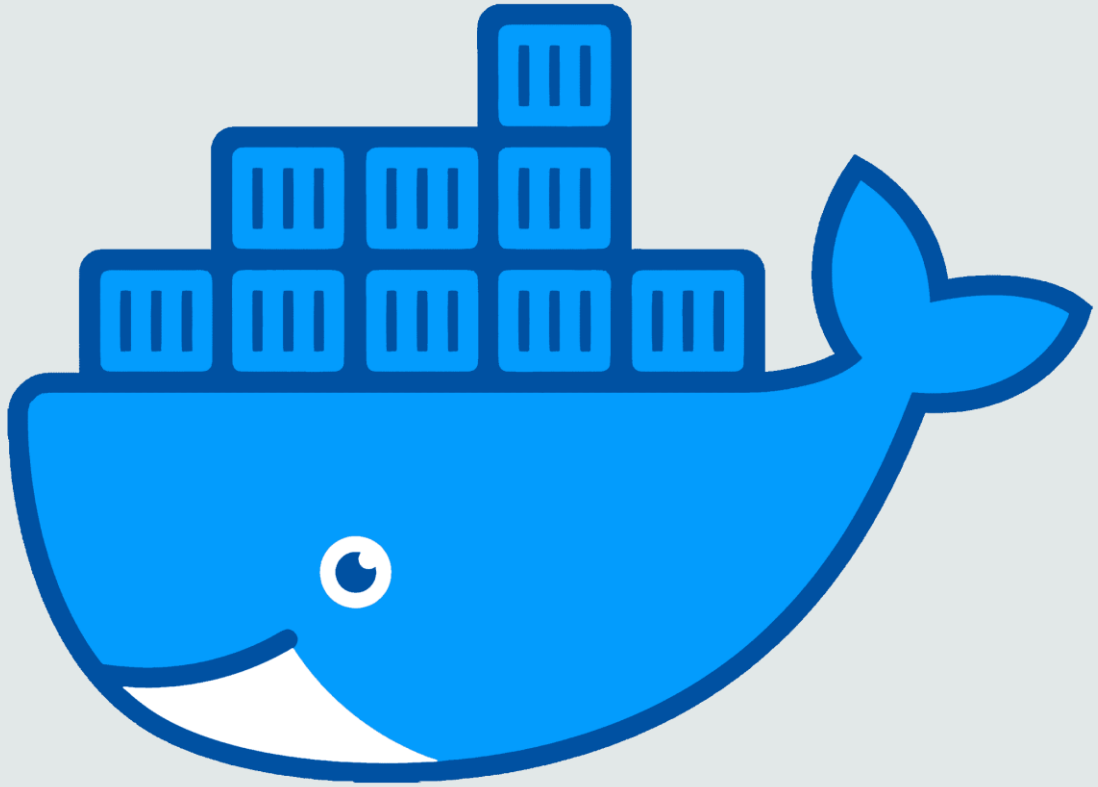
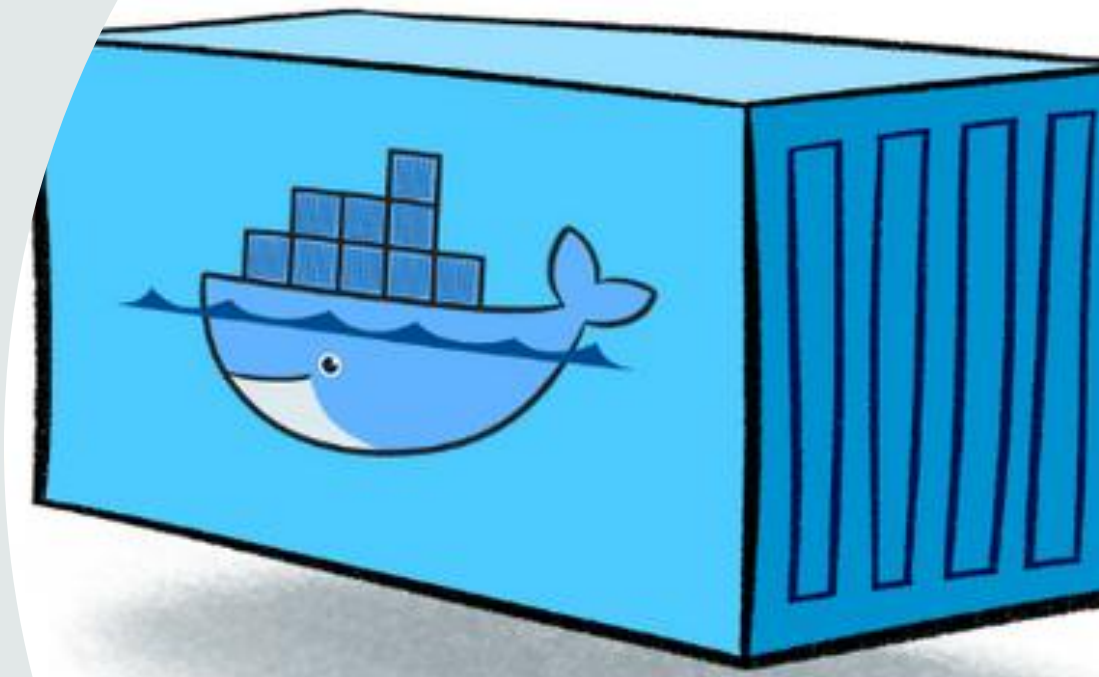
Docker



Docker 簡介

- Docker是一種開源的應用容器引擎，可輕鬆建立、部署和管理容器化應用程式。它透過將應用程式及其相依套件打包在一起，確保在不同環境中具有一致的運作效果。 Docker的使用使得開發和維運之間的協作更有效率。



Docker 用途

- 簡化環境設置流程
- 版本控制
- CI/CD

¯_(ツ)_/¯

It works

on my machine

Docker 誕生前

- 所有電腦的環境配置
- 版本控制
- 開發人員個人電腦測試
- 伺服器部屬

Docker 誕生後

容器都使用相同 **Docker Image**

Docker Image 版本控制

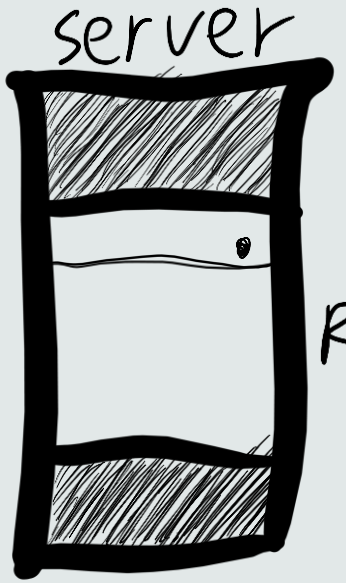
與伺服器相同環境的自動化測試

Docker 部署

環境配置

Download and install mysql.msi
redis.exe
Download project from github

```
curl -f xxoxoxx (install brew)
brew install mysql mycli redis git
git clone ……
```

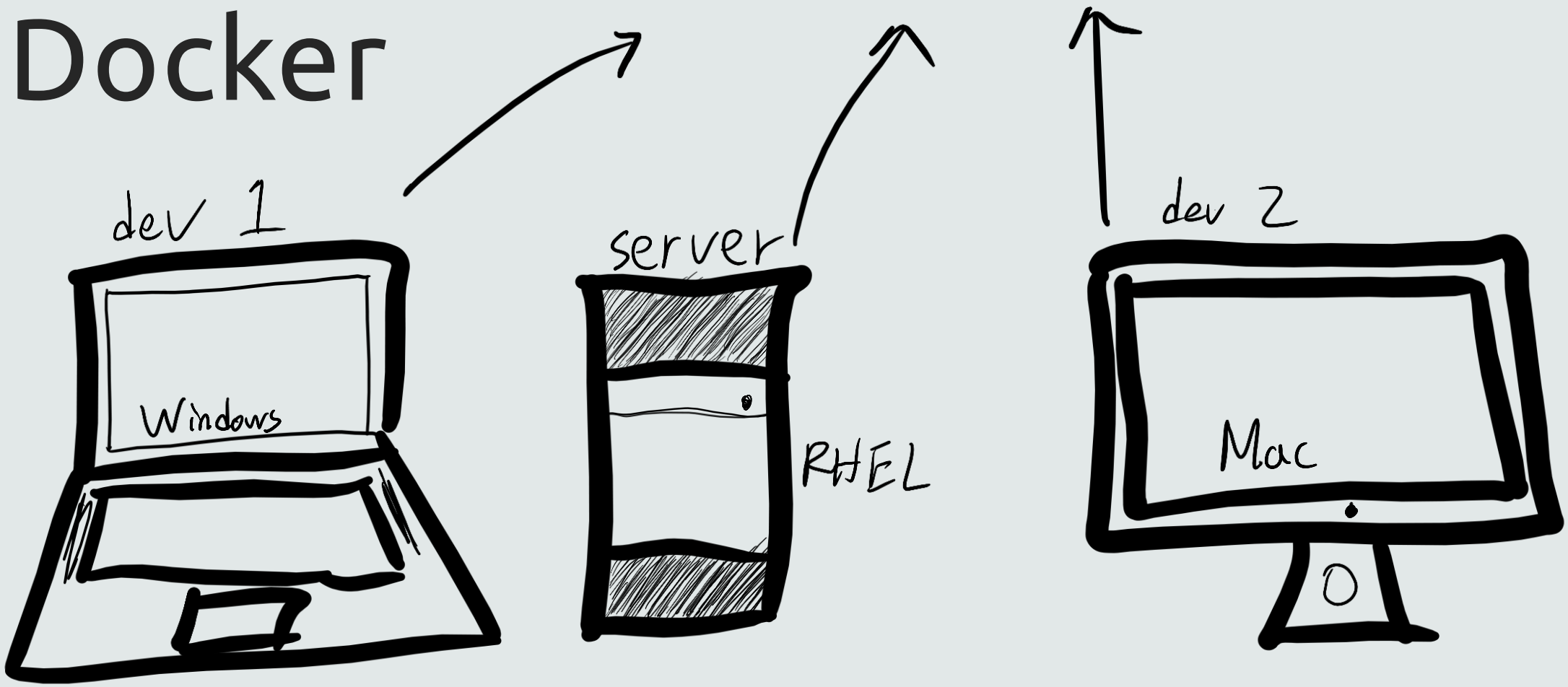


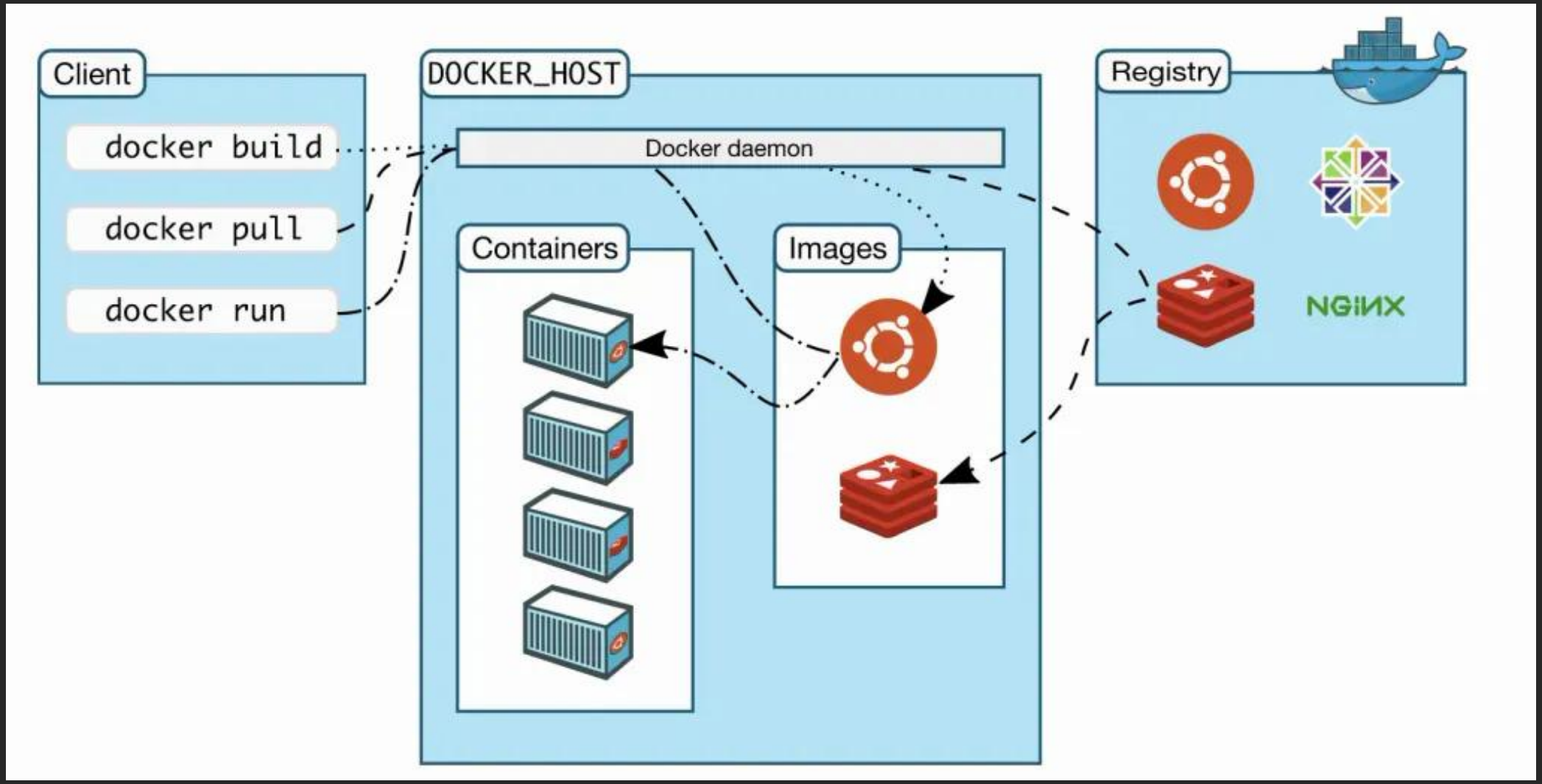

```
# yum install redis mysql-server
# systemctl enable -now mysqld
# git clone ……
```

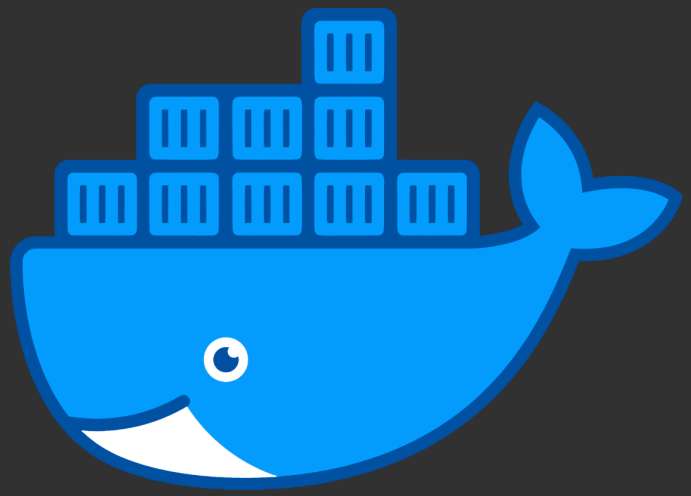
RHEL

使用 Docker

```
$ docker run npc/proj:3.1415
```




Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX



安裝與配置

安裝 WSL (optional)

安裝 Docker Desktop

- https://www.docker.com/products/docker-desktop/

Docker Run

[OPTIONS]

-d 或 --detach: 在背景運行容器。

-p 或 --publish: 將容器的端口映射到主機的端口，例如 -p 8080:80。

[OPTIONS]

-v (--volume): -v 主機上路徑:容器內路徑

--name: 指定容器名稱

[OPTIONS]

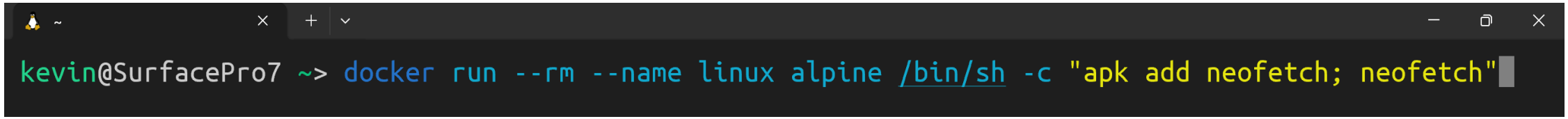
-e 或 --env: 環境變數

--rm: 在容器停止後自動刪除容器

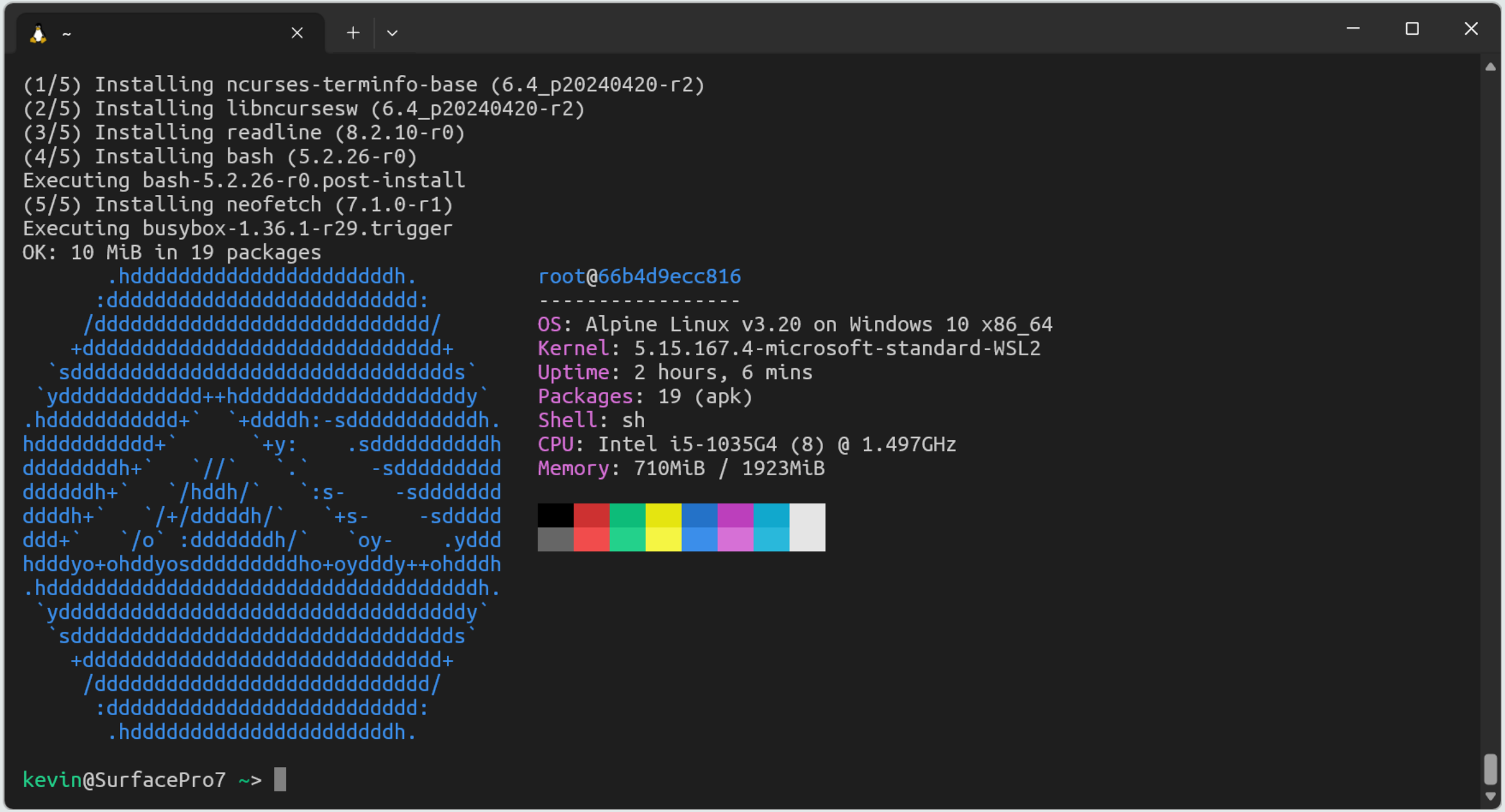
--help: 查看其他選項

實做：執行 Windows Terminal

- https://github.com/NTUT-NPC/2024-docker-course



- [ ] **docker run**: 執行 Docker 容器
- [ ] **--rm**: 停止後自動刪除容器
- [ ] **--name linux**: 指定容器的名稱為 linux
- [ ] **alpine**: 使用 Alpine Linux 鏡像
- [ ] **/bin/sh -c**: 在容器中執行命令 "apk add neofetch; neofetch"

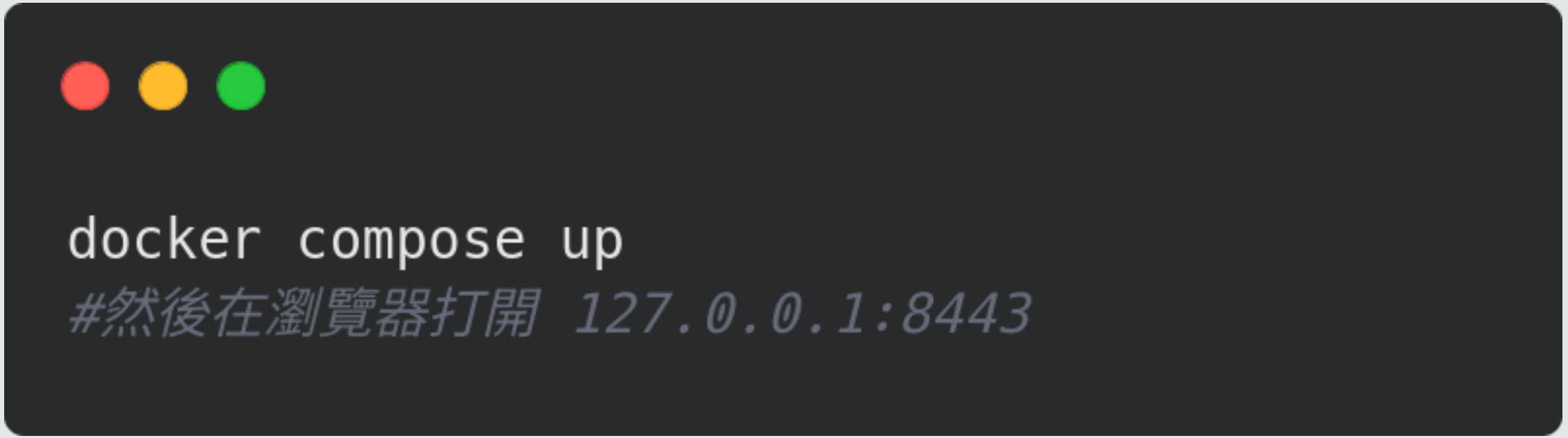

~
(1/5) Installing ncurses-terminfo-base (6.4_p20240420-r2)
(2/5) Installing libncursesw (6.4_p20240420-r2)
(3/5) Installing readline (8.2.10-r0)
(4/5) Installing bash (5.2.26-r0)
Executing bash-5.2.26-r0.post-install
(5/5) Installing neofetch (7.1.0-r1)
Executing busybox-1.36.1-r29.trigger
OK: 10 MiB in 19 packages
root@66b4d9ecc816
OS: Alpine Linux v3.20 on Windows 10 x86_64
Kernel: 5.15.167.4-microsoft-standard-WSL2
Uptime: 2 hours, 6 mins
Packages: 19 (apk)
Shell: sh
CPU: Intel i5-1035G4 (8) @ 1.497GHz
Memory: 710MiB / 1923MiB
kevin@SurfacePro7 ~>

Docker Compose

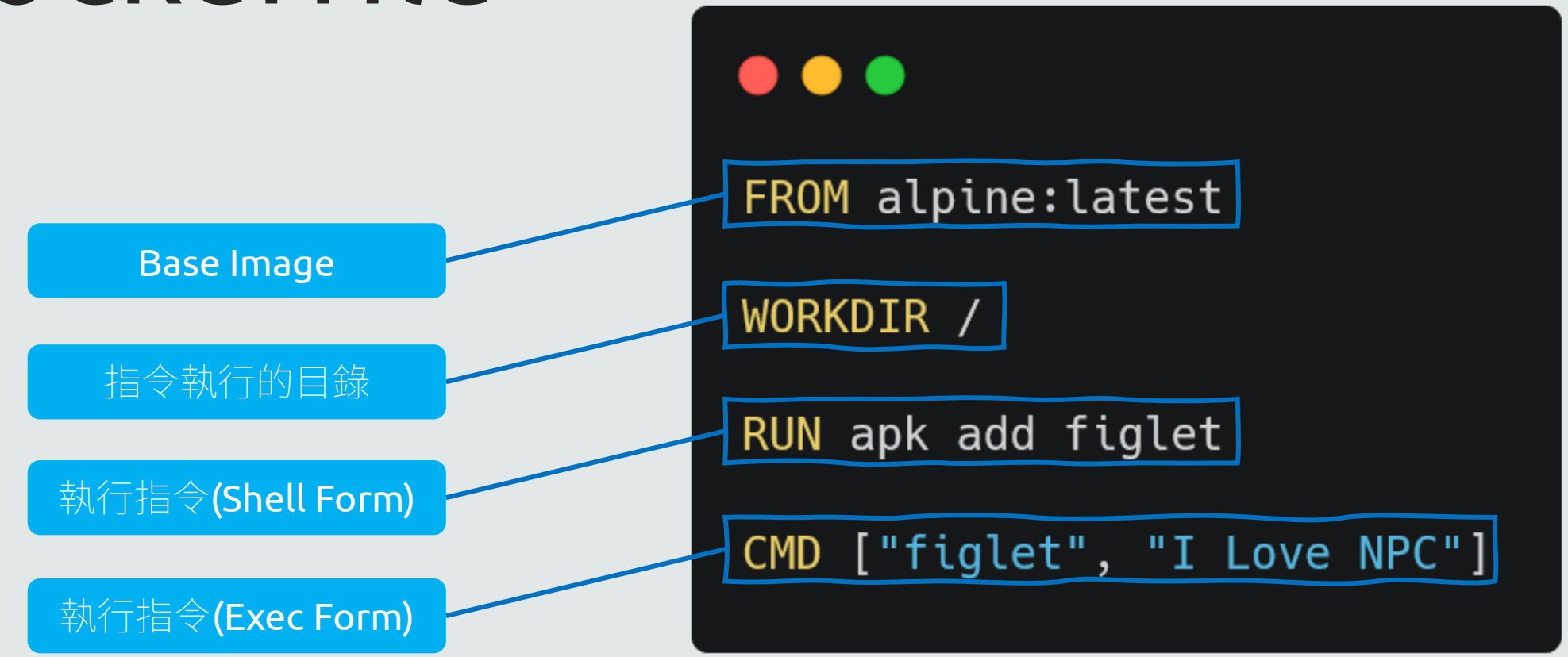
```
vscode-docker
├── compose.yml
└── config
    ├── data
    │   ├── ...
    │   └── ...
    ├── extensions
    │   └── extensions.json
    └── workspace
```

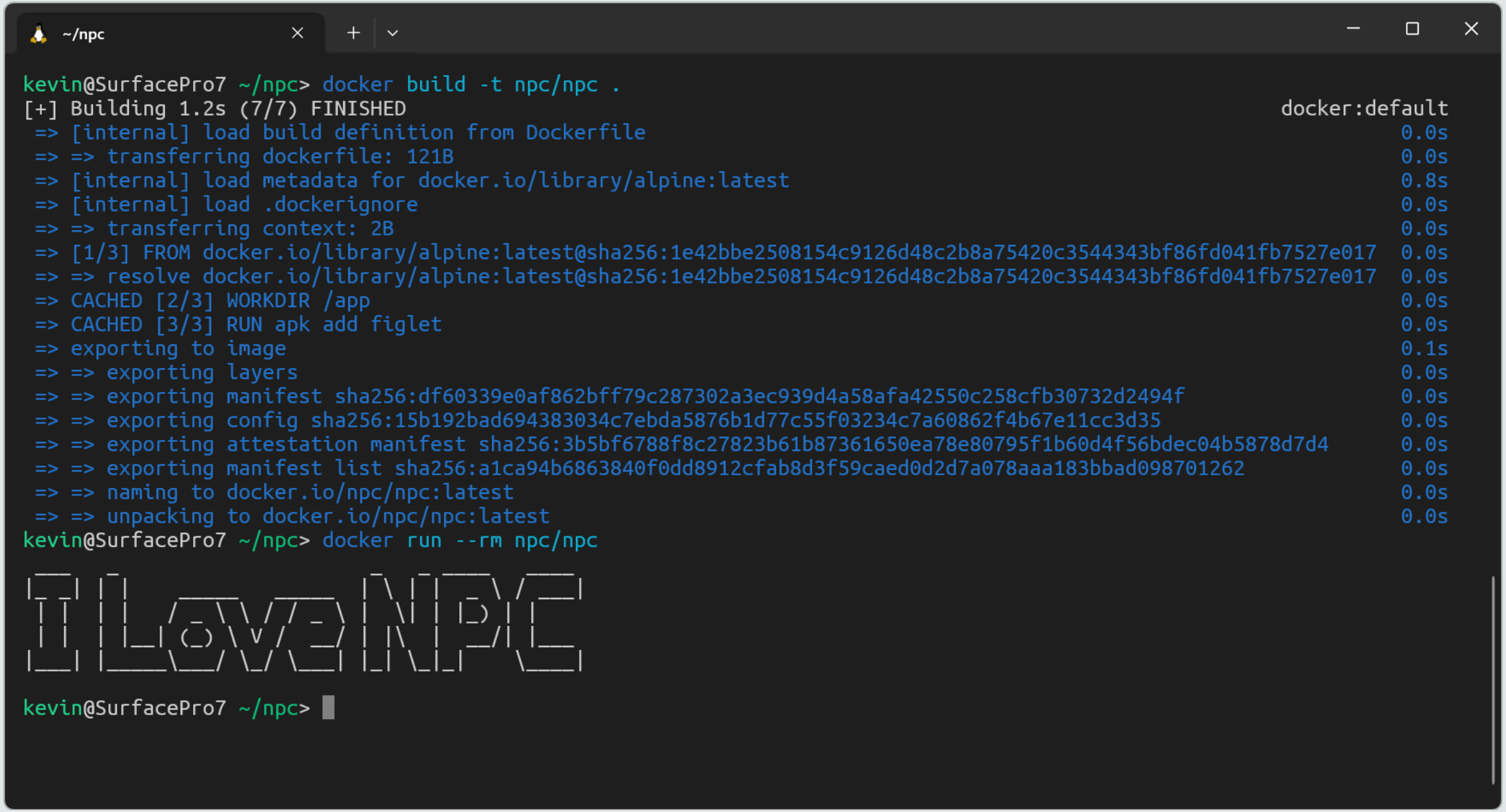
- https://github.com/linuxserver/docker-code-server

```yaml
# vscode-docker/compose.yml
services:
  code-server:
    image: lscr.io/linuxserver/code-server:latest
    container_name: code-server
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Asia/Taipei
      - SUDO_PASSWORD=root
    volumes:
      - ./config:/config
    ports:
      - 8443:8443
    restart: unless-stopped
```


docker compose up
#然後在瀏覽器打開 127.0.0.1:8443

Dockerfile


Base Image
指令執行的目錄
執行指令(Shell Form)
執行指令(Exec Form)
FROM alpine:latest
WORKDIR /
RUN apk add figlet
CMD ["figlet", "I Love NPC"]


~/npc
kevin@SurfacePro7 ~/npc> docker build -t npc/npc .
[+] Building 1.2s (7/7) FINISHED docker:default
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 121B 0.0s
=> [internal] load metadata for docker.io/library/alpine:latest 0.8s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/3] FROM docker.io/library/alpine:latest@sha256:1e42bbe2508154c9126d48c2b8a75420c3544343bf86fd041fb7527e017 0.0s
=> => resolve docker.io/library/alpine:latest@sha256:1e42bbe2508154c9126d48c2b8a75420c3544343bf86fd041fb7527e017 0.0s
=> CACHED [2/3] WORKDIR /app 0.0s
=> CACHED [3/3] RUN apk add figlet 0.0s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => exporting manifest sha256:df60339e0af862bff79c287302a3ec939d4a58afa42550c258cfb30732d2494f 0.0s
=> => exporting config sha256:15b192bad694383034c7ebda5876b1d77c55f03234c7a60862f4b67e11cc3d35 0.0s
=> => exporting attestation manifest sha256:3b5bf6788f8c27823b61b87361650ea78e80795f1b60d4f56bdec04b5878d7d4 0.0s
=> => exporting manifest list sha256:a1ca94b6863840f0dd8912cfab8d3f59caed0d2d7a078aaa183bbad098701262 0.0s
=> => naming to docker.io/npc/npc:latest 0.0s
=> => unpacking to docker.io/npc/npc:latest 0.0s
kevin@SurfacePro7 ~/npc> docker run --rm npc/npc
kevin@SurfacePro7 ~/npc>

Dockerfile

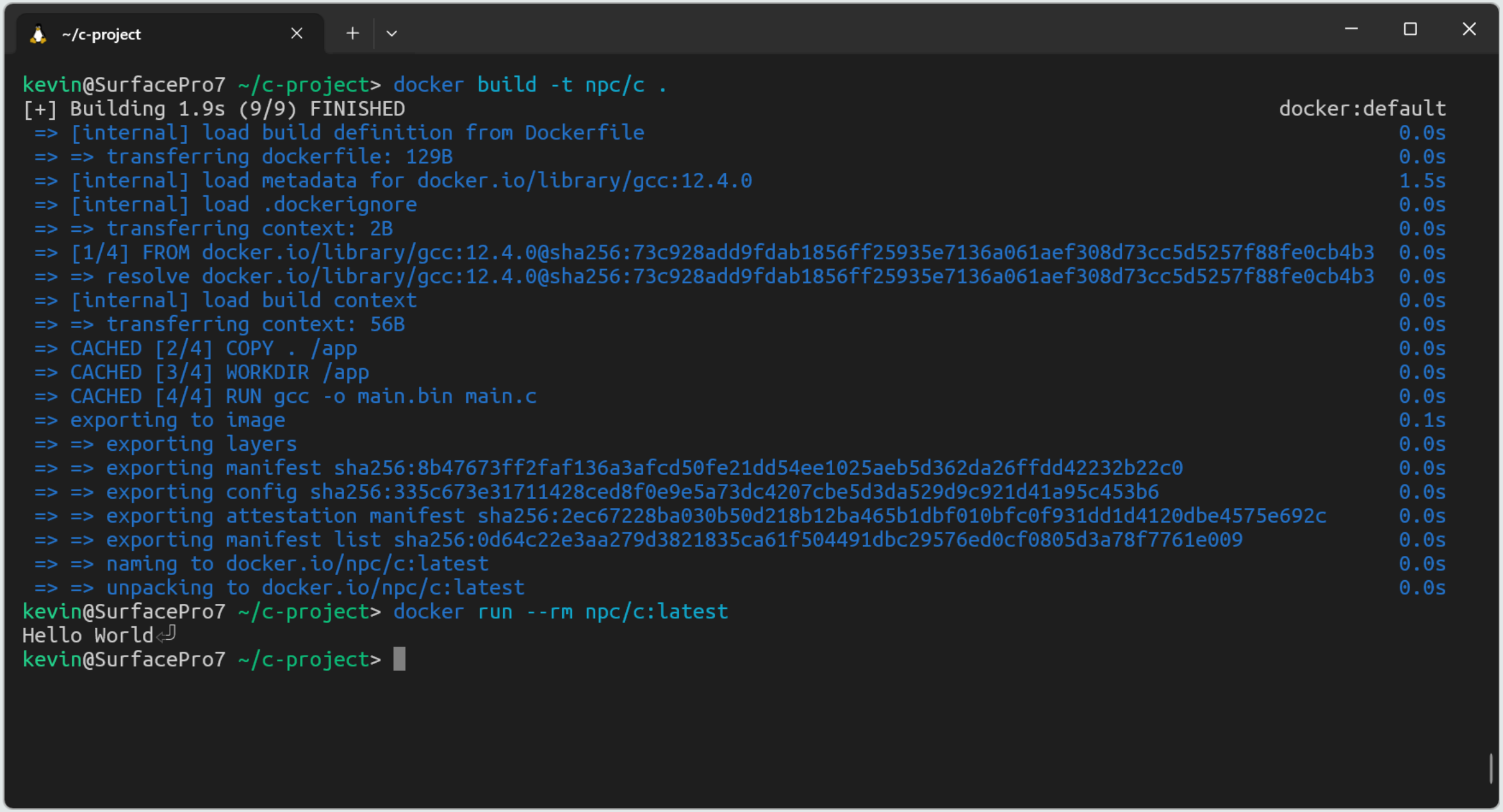
main.c

```c
#import<stdio.h>
int main(){
  printf("%s", "Hello World");
}
```

Dockerfile

```
FROM gcc:12.4.0

COPY . /app

WORKDIR /app

RUN gcc -o main.bin main.c

CMD ["./main.bin"]
```

```
gcc [-o outfile] [@file] infile
```


~/c-project
kevin@SurfacePro7 ~/c-project> docker build -t npc/c .
[+] Building 1.9s (9/9) FINISHED docker:default
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 129B 0.0s
=> [internal] load metadata for docker.io/library/gcc:12.4.0 1.5s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/4] FROM docker.io/library/gcc:12.4.0@sha256:73c928add9fdab1856ff25935e7136a061aef308d73cc5d5257f88fe0cb4b3 0.0s
=> => resolve docker.io/library/gcc:12.4.0@sha256:73c928add9fdab1856ff25935e7136a061aef308d73cc5d5257f88fe0cb4b3 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 56B 0.0s
=> CACHED [2/4] COPY . /app 0.0s
=> CACHED [3/4] WORKDIR /app 0.0s
=> CACHED [4/4] RUN gcc -o main.bin main.c 0.0s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => exporting manifest sha256:8b47673ff2faf136a3afcd50fe21dd54ee1025aeb5d362da26ffdd42232b22c0 0.0s
=> => exporting config sha256:335c673e31711428ced8f0e9e5a73dc4207cbe5d3da529d9c921d41a95c453b6 0.0s
=> => exporting attestation manifest sha256:2ec67228ba030b50d218b12ba465b1dbf010bfc0f931dd1d4120dbe4575e692c 0.0s
=> => exporting manifest list sha256:0d64c22e3aa279d3821835ca61f504491dbc29576ed0cf0805d3a78f7761e009 0.0s
=> => naming to docker.io/npc/c:latest 0.0s
=> => unpacking to docker.io/npc/c:latest 0.0s
kevin@SurfacePro7 ~/c-project> docker run --rm npc/c:latest
Hello World
kevin@SurfacePro7 ~/c-project>

```
#檔案結構
web_server/
├── Dockerfile
├── index.html
└── server.py
```

```
#Dockerfile
FROM python:3.9-slim
WORKDIR /app
COPY . .
EXPOSE 8000
CMD ["python", "server.py"]
```

```
#server.py
from http.server import SimpleHTTPRequestHandler as Handler, HTTPServer as Server

PORT = 8000
Server(("", PORT), Handler).serve_forever()
```

```
docker build -t web .
docker run -p 10000:8000 --rm web
#打開 http://127.0.0.1:10000
```

```yaml
#compose.yml
services:
 python-app:
    build: .
    ports:
      - "10000:8000"
```

```
docker compose up -d
```

刪除用不到的鏡像
```
docker image prune -a
```

刪除終止的容器
```
docker container prune
```

終止容器
```
docker stop <container_name>
docker stop <container_id>
```

列出正在執行的容器
```
docker ps
```

列出所有容器
```
docker ps -a
```

刪除容器
```
docker rm <container_name>
docker rm <container_id>
```

刪除鏡像
```
docker rmi <image_name>
```

回饋表單 ⬇️