# **Hướng dẫn sử dụng Docker**

## **I. Cài đặt Docker**

Truy cập vào đường dẫn <https://docs.docker.com/desktop/windows/install/> để cài đặt Docker cho máy tính.

Truy cập vào đường dẫn <https://docs.microsoft.com/en-us/windows/wsl/install-manual> hoàn thành hết tất cả các bước để cài đặt WSL.

Nếu bị lỗi thì do chưa cài đặt WSL và cần phải update phiên bản. Nếu vẫn còn lỗi thì truy cầm cáo AppData\...\Docker kéo xuống phía dưới kiểm tra xem "wslEngineEnabled": true

## **II. Image và Container**

Image và Container hoạt động khá là giống nhau nhưng khác ở chỗ là không có Container thì Image vẫn có thể hoạt động một cách bình thường. Còn Docker chạy thì phải phụ thuộc vào Image

## III. Hub Docker

Docker Hub là một dịch vụ do Docker cung cấp, cho phép tìm kiếm và chia sẻ các container images.

* Repositories: Push và pull container images.
* Teams & Organizations: Quản lý quyền truy cập vào private repositories của container images.
* Official Images: Pull sử dụng container images chất lượng cao của Docker.
* Publisher Images: Pull và sử dụng container images được cung cấp bởi vendors khác.
* Builds: Tự động tạo container images từ GitHub và Bitbucket. Push chúng lên Docker Hub.
* Webhooks: Kích hoạt các actions sau khi push thành công một repository lên Docker Hub với các dịch vụ khác.

Để sử dụng thì đầu tiên ta phải truy cập vào hub.docker.com tiến hành search và cài đặt platform mà mình muốn chạy chẳng hạn như node sẽ chạy “docker pull node” thì có thể chạy được images rồi

## **IV. Set up Dockerfile**

Dựa vào những thuộc tính

* FROM node:17-alpine
* WORKDIR /app (file parent nào)
* COPY . .
* RUN npm install
* EXPOSE 4000 (Port nào)
* CMD [“node”, “app.js”]

Sau đó chạy docker build -t myapp

## **V. dockerignore**

**dockerignore** sinh ra với mục đích báo cho docker biết để loại trừ những file này ra khỏi build context, tức **là** loại ra khỏi những file mà docker định dùng để build image.

Ta phải tạo file .dockerignore và trong file này để nội dung chẳng hạn như node\_modules, .env để nó sẽ loại khỏi những file này khỏi build context

## **VI. Starting and Stopping Container**

Cách 1: Sau khi mà chạy “**docker build -t myapp:v1 .”** thì chúng ta sẽ có được myapp đang chạy và chúng ta sẽ bấm vào myapp để run và đặt tên myapp\_run chẳng hạn như vậy

Cách 2: Chạy bằng command line

**docker images** kiểm tra hiện có bao nhiêu image

tiếp theo đó mình sẽ chọn image mình cần chạy bằng lệnh

**docker run --name myapp\_run -p 4000:4000 -d myapp** là để tạo ra image và run image đó luôn

**docker ps** là để kiểm tra hiện có image nào đang chạy

**docker stop myapp\_run** là kết thúc cái image đó

sau khi đã tạo image myapp\_run rồi thì ta có thể chạy nó lại mà k cần phải khởi tạo nữa

**docker start myapp\_run**

**docker ps -a** là để kiểm tra toàn bộ image trong container

**docker image rm myapp\_run** là để xóa image

**docker image rm myapp\_run -f** là để xóa image theo kiểu ép

docker container rm “Cột NA” để xóa image trong container đó

Tóm lại: Buill docker “**docker build -t myapp:v1 .”**

Tiếp theo: run docker **“docker run --name myapp\_run -p 4000:4000 myapp:v1”**

## **VII. Layer Caching**

Tận dụng tốc độ để build images

Vì khi ta build images cho 1 project lớn thì mất khá nhiều thời gian có thể mất tầm 30’ -> 1h hoặc hơn thế nữa. Thì khi ta thêm 2 dòng code console chẳng hạn và muốn sử dụng nó thì mình lại phải build lại hết toản bộ cái project điều này sẽ mất rất nhiều time và tốn tài nguyên. Solution để xử lý điều này là Layer Caching.

Khi ta build image, docker sẽ đọc từng dòng trong Dockerfile, nếu thấy layer nào ko thay đổi, đã có từ những lần build trước thì Docker sẽ tận dụng luôn chứ ko chạy lại nữa, do vậy nếu ta biết cách tổ chức Dockerfile, **đưa các thành phần ít thay đổi lên trên, thành phần hay thay đổi xuống dưới** thì sẽ tận dụng được tính năng tuyệt vời này và giảm đáng kể thời gian ngồi chờ build image.

**docker build -t myapp\_run**

Bởi vì file package.json thì ít bị thay đổi nhất. Vì khi cài npm mới thì mới thay đổi thôi nên mình sẽ layer caching file này

* FROM node:17-alpine
* WORKDIR /app (file parent nào)
* COPY package.json package-lock.json .
* RUN npm install
* COPY . . (Copy và chạy toàn bộ source code)
* EXPOSE 4000 (Port nào)
* CMD [“node”, “app.js”]

Từ đây ta có thể hiểu sau khi ta thay đổi source code lại lần nữa thì “COPY package.json package-lock.json .” sẽ làm trò không build lại 2 file này vì k có sự thay đổi.

Lứu ý: Phải đặt những file layer caching lên trên RUN npm install bởi vì nó ít bị thay đổi. Copy code phải đặt sau RUN npm install vì source code luôn luôn thay đổi.

## **VII. Volumes**

Volume trong Docker được dùng để chia sẻ dữ liệu cho container

* FROM node:17-alpine
* RUN npm install -g nodemon
* WORKDIR /app (file parent nào)
* COPY package.json package-lock.json .
* RUN npm install
* COPY . . (Copy và chạy toàn bộ source code)
* EXPOSE 4000 (Port nào)
* CMD [“node”, “app.js”] có thể thay đổi bằng ở dưới
* CMD [“npm”, “run”, “dev”]

*docker run --name myapp\_run -p 4000:4000 --rm -v C:\Code\docker-build\api:/app -v /app/node\_modules myapp:v1*

## **VIII. Docker Compose**

Docker compose là công cụ dùng để định nghĩa và run multi-container cho Docker application. Với compose bạn sử dụng file YAML để config các services cho application của bạn. Sau đó dùng command để create và run từ những config đó. Sử dụng cũng khá đơn giản chỉ với ba bước.

* Khai báo app’s environment trong Dockerfile.
* Khai báo các services cần thiết để chạy application trong file docker-compose.yml.
* Run docker-compose up để start và run app.

Ta phải build các ***.dockerfile*** và ***.dockerignore*** trong cái container để có thể run được ứng dụng.

Sau đó ta sẽ setup file ***docker-compose.yml*** ở global bên ngoài 2 container.

* **version**: chỉ ra phiên bản docker-compose đã sử dụng.
* **services**: thiết lập các services(containers) muốn cài đặt và chạy.
* **image**: chỉ ra image được sử dụng trong lúc tạo ra container.
* **build**: dùng để tạo container.
* **ports**: thiết lập ports chạy tại máy host và trong container.
* **restart**: tự động khởi chạy khi container bị tắt.

**docker-compose up** là để run một loạt container

**docker-compose down --rmi all -v** là để stop một loạt container

## **IX. Share Image Docker Hub**

Truy cập vào docker hub tạo 1 repository

Ta sẽ đứng ở folder container chứa server chẳng hạn như folder api

Sau đó run dòng lệnh “docker build -t tienkim9920/docker-first .”

Ta sẽ phải “docker login”

Tiếp theo sẽ push lên docker hub “docker push tienkim9920/docker-first”

Để xóa “docker image rm tienkim9920/docker-first”