**Phần 9: Docker File**

**A.1 Dockerfile** sử dụng DSL (Ngôn ngữ dành riêng cho miền) và chứa các hướng dẫn để tạo hình ảnh Docker. Dockerfile sẽ xác định các quy trình để tạo hình ảnh nhanh chóng. Khi tạo ứng dụng của bạn, bạn nên tạo Dockerfile theo thứ tự vì daemon Docker chạy tất cả các hướng dẫn từ trên xuống dưới.

**A.2 Các giai đoạn tạo Docker Image từ Dockerfile**

Sau đây là các giai đoạn tạo hình ảnh docker từ Dockerfile:

1. Tạo một tệp có tên là Dockerfile.
2. Thêm hướng dẫn vào Dockerfile.
3. Xây dựng Dockerfile để tạo một hình ảnh.
4. Chạy hình ảnh để tạo vùng chứa.

**A.3 Cấu trúc của docker file**

**FROM**

Câu lệnh FROM xác định hình ảnh nào cần tải xuống và bắt đầu từ đó. Đây phải là lệnh đầu tiên trong Dockerfile của bạn. Một Dockerfile có thể có nhiều câu lệnh FROM, điều đó có nghĩa là Dockerfile tạo ra nhiều hơn một hình ảnh.

**FROM <ImageName>**

**COPY**

Lệnh sao chép được sử dụng để sao chép tệp/thư mục vào hình ảnh trong khi xây dựng hình ảnh

**COPY <Source> <Destination>**

**ADD**

Nếu chúng ta xác định thêm một số tệp, câu lệnh ADD sẽ được sử dụng. Về cơ bản, nó cung cấp hướng dẫn sao chép các tệp, thư mục mới hoặc URL tệp từ xa và sau đó thêm chúng vào hệ thống tệp của hình ảnh.

**ADD <URL>**

**RUN**

Câu lệnh RUN xác định việc chạy một lệnh thông qua shell, đợi nó hoàn thành và lưu kết quả. Nó cho biết tiến trình nào sẽ chạy bên trong container vào thời gian chạy.

**RUN < Command + ARGS>**

**CMD**

CMD chỉ định toàn bộ lệnh để chạy. Chúng ta có thể nói CMD là đối số mặc định được truyền vào ENTRYPOINT. Mục đích chính của lệnh CMD là khởi chạy phần mềm được yêu cầu trong vùng chứa

**CMD [command + args]**

**ENTRYPOINT**

Một vùng chứa sẽ hoạt động như một tệp thực thi được định cấu hình bởi ENTRYPOINT. Khi bạn khởi động vùng chứa Docker, một lệnh hoặc tập lệnh có tên ENTRYPOINT sẽ được thực thi.

Không thể ghi đè nó. Sự khác biệt duy nhất giữa CMD và ENTRYPOINT là CMD có thể bị ghi đè còn ENTRYPOINT thì không.

**ENTRYPOINT [command + args]**

**ARG**

Lệnh ARG xác định một biến mà người dùng có thể chuyển vào thời điểm xây dựng cho trình xây dựng bằng lệnh docker build bằng cách sử dụng cờ --build-arg <varname>=<value>

**ARG <name>[=<default value>]**

**ENV**

Câu lệnh ENV đặt các biến môi trường cả trong quá trình xây dựng và khi chạy kết quả. Nó có thể được sử dụng trong Dockerfile và bất kỳ tập lệnh nào mà nó gọi. Nó có thể được sử dụng trong Dockerfile cũng như bất kỳ tập lệnh nào mà Dockerfile gọi.

**ENV <key>=<value> ...**

**EXPOSE**

Câu lệnh EXPOSE ánh xạ một cổng vào vùng chứa. Các cổng có thể là TCP hoặc UDP nhưng theo mặc định là TCP.

**EXPOSE <port> [<port>/<protocol>...]**

**LABEL**

Lệnh LABEL thêm siêu dữ liệu vào hình ảnh

**LABEL <key>=<value> <key>=<value> <key>=<value> ...**

**ONBUILD**

Lệnh ONBUILD thêm vào hình ảnh một lệnh kích hoạt sẽ được thực thi sau đó, khi hình ảnh được sử dụng làm cơ sở cho một bản dựng khác

**ONBUILD <INSTRUCTION>**

**SHELL**

Lệnh SHELL cho phép ghi đè shell mặc định được sử dụng cho dạng lệnh shell

**SHELL ["executable", "parameters"]**

**USER**

Lệnh USER đặt tên người dùng (hoặc UID) và tùy chọn nhóm người dùng (hoặc GID) để sử dụng làm người dùng và nhóm mặc định cho phần còn lại của giai đoạn hiện tại. Người dùng được chỉ định sẽ được sử dụng cho các lệnh RUN và trong thời gian chạy, chạy các lệnh ENTRYPOINT và CMD có liên quan

**USER <user>[:<group>] or USER <UID>[:<GID>]**

**VOLUME**

Câu lệnh VOLUME xác định các khối được chia sẻ hoặc các khối tạm thời tùy thuộc vào việc bạn có một hay hai đối số

**VOLUME ["/data"]**

**WORKDIR**

Lệnh WORKDIR thiết lập thư mục làm việc cho mọi lệnh RUN, CMD, ENTRYPOINT, COPY và ADD theo sau nó trong Dockerfile

# Nếu WORKDIR không tồn tại, nó sẽ được tạo ngay cả khi nó không được sử dụng trong bất kỳ lệnh Dockerfile nào tiếp theo.

**WORKDIR /path/to/workdir**