**Docker**

**Docker là gì và Tại sao cần sử dụng?**

**Docker** là một nền tảng mã nguồn mở giúp đóng gói ứng dụng và các thành phần phụ thuộc của nó thành các đơn vị có thể chạy được, gọi là **container**. Container là các môi trường chạy độc lập, có thể được triển khai trên nhiều hệ thống khác nhau một cách nhất quán và hiệu quả.

Docker cung cấp một cách đơn giản và hiệu quả để xây dựng, chia sẻ và triển khai ứng dụng. Nó giúp giảm thiểu sự phức tạp trong quá trình triển khai ứng dụng, tối ưu hóa tài nguyên và tăng tính di động của ứng dụng.

**Lợi ích chính của Docker**

* **Dễ sử dụng**: Có thể được sử dụng bởi bất kỳ ai, từ lập trình viên đến quản trị viên hệ thống
* **Nhanh chóng**: Container tương đối nhỏ gọn và có thể được khởi chạy trong vài giây
* **Tính di động**: Container có thể được triển khai trên bất kỳ nền tảng nào hỗ trợ Docker
* **Khả năng mở rộng**: Container có thể được kết nối với nhau để tạo thành các ứng dụng phức tạp

**Kiến trúc và Thành phần Docker**

**Docker Engine**

Docker hoạt động trên kiến trúc client-server với các thành phần chính:

* **Docker CLI**: Giao diện dòng lệnh để tương tác với Docker
* **Docker API**: Giao diện cho các ứng dụng tương tác với Docker Engine
* **Docker Daemon**: Chạy trên máy chủ và thực hiện các công việc chính như build, chạy và quản lý container

**Docker Images**

Docker Images là các bản thiết kế cho container, định nghĩa tất cả những gì một ứng dụng cần để chạy. Một image sau khi được tạo ra sẽ không thể thay đổi (immutable).

**Docker Containers**

Containers là các instance đang chạy của Docker images, đóng gói một ứng dụng và tất cả các thành phần phụ thuộc của nó.

**Cài đặt Docker**

**Trên Ubuntu/Linux**

*# Cập nhật package list*

sudo apt-get update

*# Cài đặt các dependencies*

sudo apt-get install ca-certificates curl gnupg

*# Thêm Docker repository*

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

A screenshot of a computer

AI-generated content may be incorrect.

*# Cài đặt Docker*

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io

A screenshot of a computer program

AI-generated content may be incorrect.

*# Khởi động Docker*

sudo systemctl start docker

sudo systemctl enable docker

A screenshot of a computer

AI-generated content may be incorrect.

*# Kiểm tra cài đặt*

sudo docker run hello-world

A screenshot of a computer

AI-generated content may be incorrect.

**Trên Windows**

1. Tải Docker Desktop từ trang chính thức của Docker
2. Chạy file cài đặt và làm theo hướng dẫn
3. Kích hoạt tính năng Hyper-V trong quá trình cài đặt
4. Khởi động lại máy tính sau khi cài đặt hoàn tất

**Trên macOS**

Tải Docker Desktop từ trang chính thức và cài đặt như các ứng dụng thông thường[4](https://xuanthulab.net/gioi-thieu-ve-docker-lam-quen-voi-docker-tao-container.html)

**Các Lệnh Docker Cơ bản**

**Kiểm tra phiên bản và thông tin hệ thống**

bash

*# Kiểm tra phiên bản Docker*

docker –version

A computer screen with white text

AI-generated content may be incorrect.

*# Xem thông tin chi tiết về Docker*

docker info

A screenshot of a computer

AI-generated content may be incorrect.

**Làm việc với Images**

*# Tải image từ Docker Hub*

docker pull <image-name>

*# Liệt kê tất cả images*

docker images

*# Xóa image*

docker rmi <image-id>

*# Tìm kiếm image trên Docker Hub*

docker search <image-name>

**Làm việc với Containers**

*# Chạy container từ image*

docker run <image-name>

*# Chạy container ở chế độ background*

docker run -d <image-name>

*# Chạy container với port mapping*

docker run -p <host-port>:<container-port> <image-name>

*# Liệt kê containers đang chạy*

docker ps

*# Liệt kê tất cả containers*

docker ps -a

*# Dừng container*

docker stop <container-id>

*# Xóa container*

docker rm <container-id>

*# Truy cập vào container đang chạy*

docker exec -it <container-id> bash

**Dockerfile - Tạo Images Tùy chỉnh**

**Cấu trúc Dockerfile**

Dockerfile là file text chứa các chỉ thị để build một Docker image[5](https://docs.docker.com/get-started/docker-concepts/building-images/writing-a-dockerfile/). Các chỉ thị phổ biến:

# Chỉ định base image

FROM ubuntu:20.04

# Thiết lập thư mục làm việc

WORKDIR /app

# Copy files từ host vào container

COPY . .

# Chạy lệnh trong quá trình build

RUN apt-get update && apt-get install -y python3

# Thiết lập biến môi trường

ENV PYTHON\_VERSION=3.9

# Mở port

EXPOSE 8080

# Chạy lệnh khi container start

CMD ["python3", "app.py"]

**Ví dụ Dockerfile cho ứng dụng Node.js**

FROM node:18-alpine

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 3000

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

**Build image từ Dockerfile**

*# Build image*

docker build -t my-app .

*# Build với tag cụ thể*

docker build -t my-app:v1.0 .

**Docker Compose - Quản lý Multi-Container**

Docker Compose là công cụ giúp định nghĩa và chạy các ứng dụng multi-container bằng file YAML.

**Cấu trúc docker-compose.yml**

version: '3.8'

services:

web:

build: .

ports:

- "3000:3000"

environment:

- NODE\_ENV=production

depends\_on:

- db

db:

image: postgres:13

environment:

POSTGRES\_DB: myapp

POSTGRES\_USER: user

POSTGRES\_PASSWORD: password

volumes:

- postgres\_data:/var/lib/postgresql/data

volumes:

postgres\_data:

**Lệnh Docker Compose cơ bản**

bash

*# Khởi động tất cả services*

docker-compose up

*# Khởi động ở background*

docker-compose up -d

*# Dừng và xóa containers*

docker-compose down

*# Xem logs*

docker-compose logs

*# Build lại services*

docker-compose build

*# Khởi động lại service cụ thể*

docker-compose restart web

**Docker Volumes - Quản lý Dữ liệu**

Docker Volumes là cơ chế lưu trữ dữ liệu bên ngoài containers, đảm bảo dữ liệu không bị mất khi container bị xóa.

**Các loại volumes**

1. **Named Volumes**: Được quản lý hoàn toàn bởi Docker
2. **Bind Mounts**: Mount trực tiếp thư mục từ host
3. **Anonymous Volumes**: Volumes tạm thời không có tên

**Làm việc với Volumes**

*# Tạo volume*

docker volume create my-volume

*# Liệt kê volumes*

docker volume ls

*# Kiểm tra thông tin volume*

docker volume inspect my-volume

*# Chạy container với volume*

docker run -v my-volume:/data nginx

*# Bind mount*

docker run -v /host/path:/container/path nginx

*# Xóa volume*

docker volume rm my-volume

*# Xóa tất cả volumes không sử dụng*

docker volume prune

**Docker Networking**

Docker cung cấp nhiều loại mạng để containers có thể giao tiếp với nhau[8](https://www.geeksforgeeks.org/basics-of-docker-networking/):

**Các loại network drivers**

* **Bridge**: Mạng mặc định cho containers standalone
* **Host**: Container sử dụng trực tiếp network của host
* **None**: Container không có network interface
* **Overlay**: Cho Docker Swarm và multi-host networking

**Lệnh networking cơ bản**

*# Liệt kê networks*

docker network ls

*# Tạo network*

docker network create my-network

*# Kiểm tra thông tin network*

docker network inspect my-network

*# Kết nối container với network*

docker network connect my-network container-name

*# Ngắt kết nối*

docker network disconnect my-network container-name

*# Xóa network*

docker network rm my-network

**Bảo mật Docker**

**Best Practices bảo mật**

1. **Sử dụng base images minimal và trusted**

FROM alpine:3.18.3 # Thay vì ubuntu:latest

1. **Không chạy container với quyền root**

RUN useradd -m appuser

USER appuser

1. **Quét vulnerabilities trong images**

docker scout cves <image-name>

1. **Sử dụng .dockerignore**

node\_modules/

.git/

\*.env

.DS\_Store

1. **Giới hạn capabilities**

docker run --cap-drop ALL --cap-add NET\_BIND\_SERVICE my-app

1. **Cập nhật thường xuyên**

*# Rebuild images thường xuyên*

docker build --no-cache -t my-app .

**Triển khai Production**

**Chuẩn bị cho Production**

1. **Tạo production Dockerfile**

# Multi-stage build

FROM node:18-alpine AS builder

WORKDIR /app

COPY package\*.json ./

RUN npm ci --only=production

FROM node:18-alpine AS production

WORKDIR /app

COPY --from=builder /app/node\_modules ./node\_modules

COPY . .

USER node

EXPOSE 3000

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

1. **Production docker-compose.yml**

version: '3.8'

services:

app:

build:

context: .

dockerfile: Dockerfile.production

restart: always

environment:

- NODE\_ENV=production

deploy:

resources:

limits:

memory: 512M

cpus: "0.5"

1. **CI/CD Pipeline cơ bản**

# GitHub Actions example

name: Deploy to Production

on:

push:

branches: [main]

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- name: Build and push Docker image

run: |

docker build -t my-app:latest .

docker tag my-app:latest registry.com/my-app:latest

docker push registry.com/my-app:latest

- name: Deploy to server

run: |

ssh user@server "docker pull registry.com/my-app:latest"

ssh user@server "docker-compose up -d"

**Bài tập Thực hành**

**Bài tập 1: Container cơ bản**

1. Chạy container nginx với port mapping 8080:80
2. Tạo container MySQL với password và port tùy chỉnh
3. Thực hành mount volume để lưu trữ dữ liệu

1. Chạy container nginx với port mapping 8080:80

Pull image nginx

A screenshot of a computer

AI-generated content may be incorrect.

 Chạy container với port mapping

A screenshot of a computer program

AI-generated content may be incorrect.

2. Tạo container MySQL với password và port tùy chỉnh

Chạy container MySQL:

docker run -d --name test-mysql -e MYSQL\_ROOT\_PASSWORD=strong\_password -p 3307:3306 mysql

* -e MYSQL\_ROOT\_PASSWORD=strong\_password: Thiết lập password cho user root
* -p 3307:3306: Map port 3307 của host với port 3306 mặc định của MySQL

A screenshot of a computer program

AI-generated content may be incorrect.

Kiểm tra port mapping:

docker port test-mysql

Kết nối tới MySQL từ host:

mysql --host=127.0.0.1 --port=3307 -u root -pA screenshot of a computer

AI-generated content may be incorrect.

3. Thực hành mount volume để lưu trữ dữ liệu

**Tạo named volume và chạy container với volume mount**

A screenshot of a computer program

AI-generated content may be incorrect.

Liệt kê volume và kiểm tra volume

A screenshot of a computer

AI-generated content may be incorrect.

**Bài tập 2: Dockerfile**

1. Tạo Dockerfile cho ứng dụng Python Flask đơn giản
2. Build image và chạy container
3. Optimize Dockerfile với multi-stage builds

**1. Tạo Dockerfile cho ứng dụng Python Flask đơn giản**

**Bước 1: Tạo cấu trúc thư mục**

mkdir flask-docker

cd flask-docker

**Bước 2: Tạo file Flask app**

**Tạo file app.py:**

**from** flask **import** Flask

app = Flask(\_\_name\_\_)

@app.route('/')

**def** hello\_world():

**return** '<h1>Hello from Flask & Docker!</h1>'

@app.route('/health')

**def** health\_check():

**return** {'status': 'healthy', 'message': 'Flask app is running'}

**if** \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True, host='0.0.0.0', port=5000)

**Bước 3: Tạo file requirements.txt**

Flask==3.0.0

**Bước 4: Tạo Dockerfile đơn giản**

# Sử dụng Python official image

FROM python:3.11-slim

# Thiết lập working directory

WORKDIR /app

# Copy requirements file

COPY requirements.txt .

# Cài đặt dependencies

RUN pip install --no-cache-dir -r requirements.txt

# Copy application code

COPY . .

# Expose port

EXPOSE 5000

# Chạy application

CMD ["python", "app.py"]

**2. Build image và chạy container**

**Build Docker image**

docker build -t flask-app .

A screenshot of a computer screen

AI-generated content may be incorrect.

**Kiểm tra image đã tạo:**

docker images

**Chạy container**

docker run -d --name my-flask-app -p 5000:5000 flask-app

A screenshot of a computer

AI-generated content may be incorrect.

**Kiểm tra container đang chạy:**

docker ps

**Test ứng dụng:**

curl http://localhost:5000

curl <http://localhost:5000/health>

A screenshot of a computer program

AI-generated content may be incorrect.

**Xem logs của container:**

docker logs my-flask-app

A screenshot of a computer

AI-generated content may be incorrect.

**Bài tập 3: Docker Compose**

1. Tạo stack WordPress + MySQL bằng Docker Compose
2. Thêm phpMyAdmin để quản lý database
3. Cấu hình volumes cho persistent data

Tạo file docker-compose.yml với nội dung sau:

version: '3.9'

services:

db:

image: mysql:5.7

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: password

MYSQL\_DATABASE: wordpress

MYSQL\_USER: wordpress

MYSQL\_PASSWORD: wordpress

volumes:

- db\_data:/var/lib/mysql

networks:

- wpsite

phpmyadmin:

depends\_on:

- db

image: phpmyadmin/phpmyadmin

restart: always

ports:

- '8080:80'

environment:

PMA\_HOST: db

MYSQL\_ROOT\_PASSWORD: password

networks:

- wpsite

wordpress:

depends\_on:

- db

image: wordpress:latest

restart: always

ports:

- '8000:80'

environment:

WORDPRESS\_DB\_HOST: db:3306

WORDPRESS\_DB\_USER: wordpress

WORDPRESS\_DB\_PASSWORD: wordpress

WORDPRESS\_DB\_NAME: wordpress

volumes:

- ./wp-app:/var/www/html

networks:

- wpsite

networks:

wpsite:

volumes:

db\_data:

Chạy lệnh: *docker-compose up -d*

A screenshot of a computer

AI-generated content may be incorrect.  
Truy cập các dịch vụ

* **WordPress:** [http://localhost:8000](http://localhost:8000/)

A screenshot of a computer

AI-generated content may be incorrect.

* **phpMyAdmin:** [http://localhost:8080](http://localhost:8080/)
  + Đăng nhập bằng user: root, password: password hoặc user: wordpress, password: wordpress.
  + Do xung đột cổng 8080 ở bài tập 1, chuyển sang port 8081 sửa trong file docker-compose.yml

A screenshot of a computer

AI-generated content may be incorrect.

Một số thao tác liên quan

A screenshot of a computer

AI-generated content may be incorrect.

**Bài tập 4: Networking**

1. Tạo custom network
2. Chạy nhiều containers trong cùng network
3. Test connectivity giữa các containers

Tạo một mạng bridge mới tên là my-custom-network:

docker network create my-custom-network

A computer screen shot of a computer program

AI-generated content may be incorrect.

 chạy 2 container (nginx và alpine) cùng nằm trong network này

A screenshot of a computer program

AI-generated content may be incorrect.

Kết nối vào container web2:

docker exec -it web2 sh

 Ping tới container web1 bằng tên:

ping web1

A computer screen shot of a program

AI-generated content may be incorrect.

**Tài nguyên Học tập**

1. **Documentation chính thức**: [docs.docker.com](https://docs.docker.com/)
2. **Docker Hub**: [hub.docker.com](https://hub.docker.com/) - Repository các images
3. **Tutorials interactives**: [docker.com/101-tutorial](https://www.docker.com/101-tutorial/)
4. **Video tutorials**: Nhiều channel YouTube như "TechWorld with Nana", "Hỏi Dân IT"

Docker là công nghệ quan trọng trong DevOps hiện đại. Việc thành thạo Docker sẽ giúp bạn dễ dàng triển khai, scale và quản lý ứng dụng một cách hiệu quả. Hãy bắt đầu với các khái niệm cơ bản và từ từ tiến tới các tính năng nâng cao như orchestration với Docker Swarm hoặc Kubernetes.