

# BÀI 6:

# BUILD DOCKER IMAGE (P2)



## Mục đích:

- Giảm kích thước Docker Image
- Cải thiện tốc độ build
- Tăng cường bảo mật và hiệu suất:
  - Loại bỏ các công cụ, thư viện, hoặc mã nguồn không cần thiết khỏi image, giảm bloat.
- Cải thiện tính linh hoạt trong quá trình triển khai:
  - Sử dụng các công cụ như Multi-Stage Build, ARG và ENV để tạo ra các Docker Image có khả năng cấu hình và mở rộng linh hoạt.
- Tối ưu hóa tài nguyên hệ thống

# Multi-Stage Build

1.1

- Multi-Stage Build là kỹ thuật chia quá trình build Docker Image thành nhiều giai đoạn (stages) để giảm kích thước và cải thiện hiệu suất.

```
# Stage 1: Build
FROM golang:1.19 AS builder
WORKDIR /app
COPY ..
RUN go build -o main .
```

```
# Stage 2: Runtime
FROM alpine:3.17
WORKDIR /app
COPY --from=builder /app/main .
CMD ["./main"]
```

### ARG (Build Arguments)

- Dùng để truyền giá trị vào quá trình build Docker Image.
- Cú pháp:  
**ARG VARIABLE\_NAME**

```
ARG VERSION=1.0.0
```

### ENV

- Dùng để cấu hình container trong runtime.
- Cú pháp:  
**ENV KEY VALUE**

```
ENV APP_ENV production  
ENV PORT 8080
```

### So sánh ENV và ARG:

Đặc điểm	ENV	ARG
Thời điểm sử dụng	Runtime (khi container chạy)	Build-time (khi tạo Docker Image)
Phạm vi	Toàn bộ container	Chỉ trong quá trình build

- Nhằm giúp tối ưu quá trình build nhanh hơn.
- Được ENABLED mặc định trong docker.

## Nguyên lý hoạt động:

- Mỗi command trong dockerfile là một layer
- Giữa 2 lần build, nếu thành phần của command(layer) có sự thay đổi, docker sẽ build lại layer đó và các layer sau nó.

## Áp dụng:

- Giảm số lượng layer, gom các lệnh có điểm chung thành một line (1 layer)
- Tách các phần ít thay đổi để tạo thành layer có thể cached.

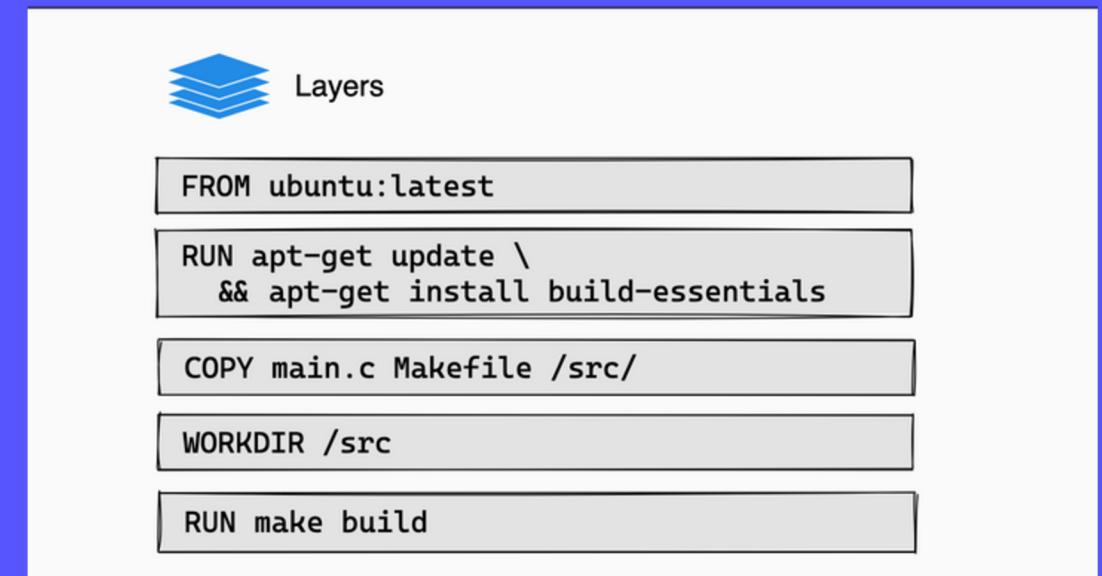
# Build cache

1.3

## Nguyên lý hoạt động:

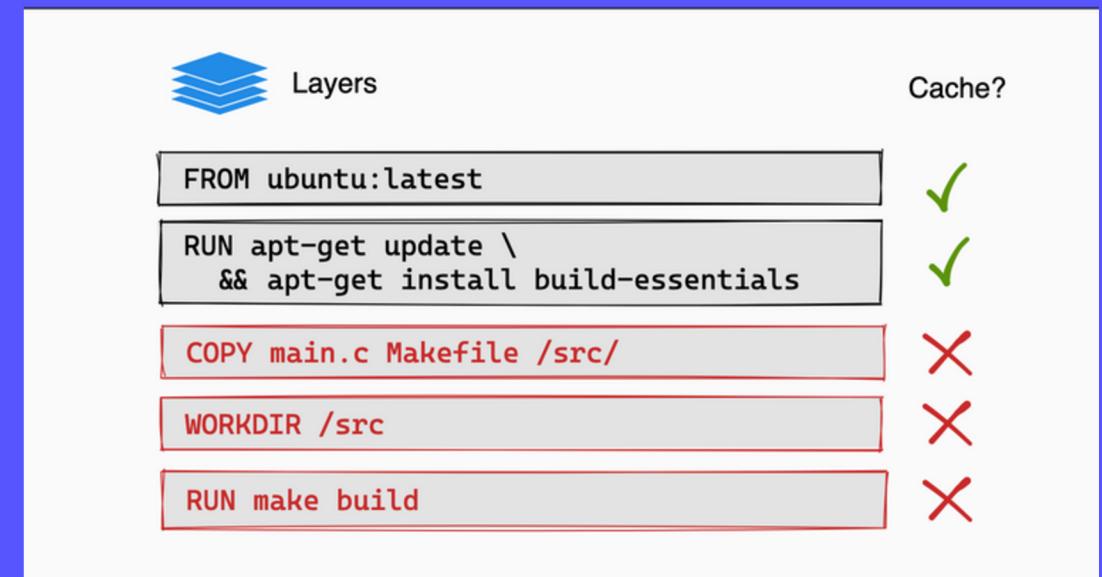
build #1

```
FROM ubuntu:latest  
  
RUN apt-get update && apt-get install -y build-essentials  
COPY main.c Makefile /src/  
WORKDIR /src/  
RUN make build
```



update file main.c -> build #2

```
FROM ubuntu:latest  
  
RUN apt-get update && apt-get install -y build-essentials  
COPY main.c Makefile /src/  
WORKDIR /src/  
RUN make build
```



# Thực hành kết hợp ARG, ENV

2

## Ví dụ Dockerize Golang app

```
//step1: Dockerfile

# Stage 1: Build
FROM golang:1.19 AS builder
ARG VERSION
ENV APP_VERSION=$VERSION
WORKDIR /app
COPY .
RUN go build -o main .
```

```
# Stage 2: Runtime
FROM alpine:3.17
ENV APP_ENV=production
WORKDIR /app
COPY --from=builder /app/main .
CMD ["./main"]
```

//step2: Run build command

```
docker build --build-arg VERSION=1.0.0 -t app:v1 .
docker run -p 3000:3000 app:v2
```

Backend Team

THANK  
YOU

