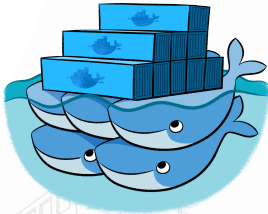
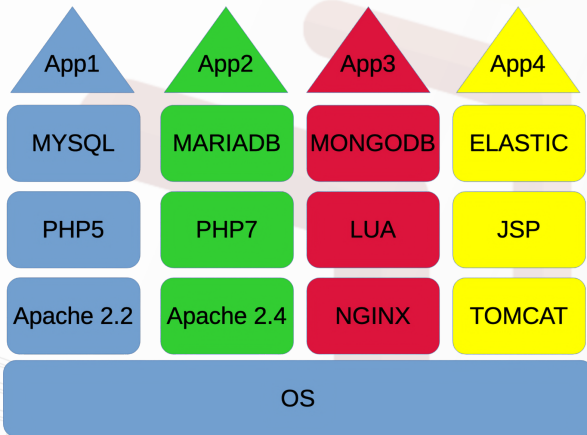


- Introduction to Docker
 - What is Docker?
- Workshop
 - Hello Docker
 - Docker Images
 - Docker Container
 - Docker Volume
 - Docker Network
 - Docker Build



Docker

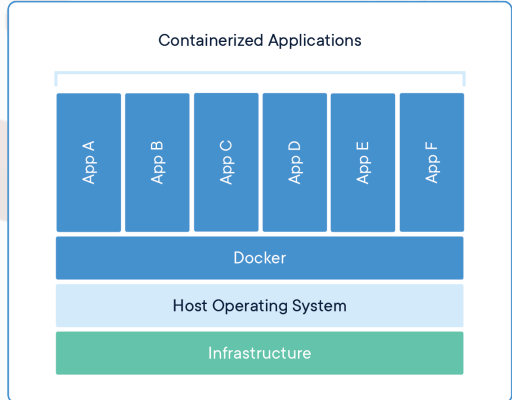
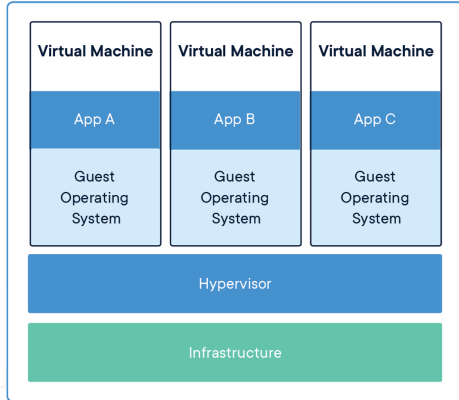
A container technology platform that allows you to package applications and other components into a unit called a "container" which can run on any server.



You still doing that?

New Generation Developers

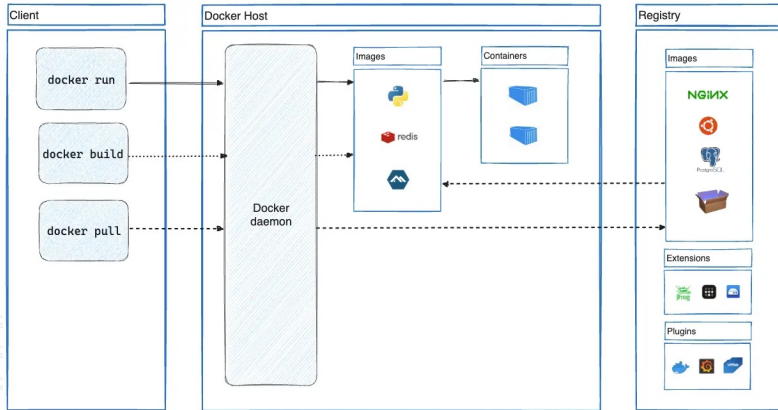
Introduction to Docker • What is Docker?



Virtual Machines vs Containers ®

Docker Architecture

Introduction to Docker • What is Docker?



Docker Architecture

```
$ docker info
```

คำสั่งหมวดแรกที่จะแนะนำให้รู้จักคือ docker image ซึ่งใช้สำหรับแสดงรายการ และ จัดการ images ที่มีอยู่ในเครื่อง

```
$ docker image ls
```

```
$ docker search "minimal image"
```

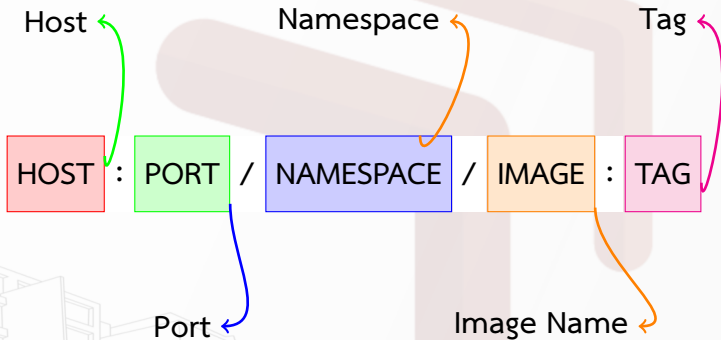
```
$ docker image pull busybox
```

```
$ docker image ls
```

```
$ docker image ls --no-trunc
```

Docker Image Name

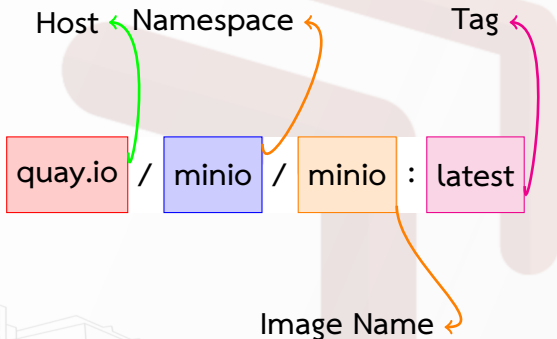
โครงสร้างของชื่อ Docker Image ประกอบไปด้วยส่วนต่าง ๆ ดังนี้



ส่วนประกอบของ Docker Image Name

Docker Image Name

ตัวอย่างของ Docker Image Name

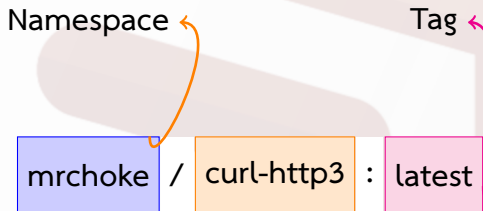


Docker Image ของ `quay.io/minio/minio:latest`

ตัวอย่างของ Docker Image Name

Namespace / Image Name : Tag

mrchoke / curl-http3 : latest



Docker Image ของ mrchoke/curl-http3:latest

```
$ docker image pull docker.io/library/alpine:3.16
```

มาเปลี่ยนชื่อ และ tag image กันด้วยคำสั่ง docker image tag กัน

```
$ docker image tag alpine alpine:abc
```

```
$ docker image tag alpine mrchoke/alpine:abc
```

```
$ docker image tag alpine:3.16 mrchoke:3.16
```

มาลบ image กันด้วยคำสั่ง docker image rm กัน

```
$ docker image rm alpine:abc
```

```
$ docker rmi mrchoke/alpine:abc
```

```
$ docker image rm 452e7292acee 036a068d7d6b
```

```
$ docker rmi $(docker images -q) # dangerous
```

```
$ docker rmi -f $(docker images -q) # dangerous
```

จัดการกับ container ด้วยคำสั่ง docker container

```
$ docker container ls
```

```
$ docker run -d alpine sleep infinity
```

```
$ docker container ls
```

```
$ docker run -d alpine sleep 1
```

```
$ docker container ls
```

```
$ docker container ls -a
```

```
$ docker container ls -a --format '{{ .Names }} {{ .Status }}
```

สร้าง container ด้วยคำสั่ง docker container run

```
$ docker container run alpine echo 'Hello, World!'
```

```
$ docker container run --rm alpine echo 'Hello, World!'
```

```
$ docker container run --rm alpine sh -c 'read name; echo Hi, $name'
```

```
$ docker container run --rm -i alpine sh -c 'read name; echo Hi, $name'
```

สร้าง container ด้วยคำสั่ง docker container run กันต่อ

```
$ docker container run --rm alpine /bin/sh
```

```
$ docker container run --rm -i alpine /bin/sh
```

```
$ docker container run --rm -it alpine /bin/sh
```

สร้าง container ด้วยคำสั่ง docker container run กันต่อ

```
$ docker container run -d alpine sleep infinity
```

```
$ docker container run -d --name con1 alpine sleep infinity
```

```
$ docker container rename con1 con2
```

```
$ docker container rename 0db24957ed64 con3
```

จัดการ container ด้วยคำสั่ง docker container [start, stop, restart]

\$ docker container ls

\$ docker container stop con2

\$ docker container ls -a

\$ docker container start con2

\$ docker container restart con2

ตรวจสอบ container ด้วยคำสั่ง docker container inspect

```
$ docker container ls
```

```
$ docker container inspect con2
```

```
$ docker container inspect --format '{{ .NetworkSettings.IPAddress }}' con2
```

```
$ docker container inspect --format '{{ .Config.Cmd }}' con2
```

ลบ container ด้วยคำสั่ง docker container rm

\$ docker container ls

\$ docker container rm con2

\$ docker container stop con2

\$ docker container rm con2

\$ docker run -d --name neverdie alpine sleep infinity

\$ docker container rm neverdie

\$ docker container rm -f neverdie

ลบ container ด้วยคำสั่ง docker container rm กันต่อ
ถ้าจะลบทั้งหมดที่กำลังทำงาน และไม่ทำงาน ?

```
$ docker container ls -a
```

dangerous

```
$ docker container rm -f $(docker container ls -aq)
```

Container กับ Environment

```
$ docker container run --rm alpine env
```

```
$ docker container run --rm -e MYVAR=HelloWorld alpine env
```

```
$ echo MYVAR=HelloWorld > .env
```

```
$ docker container run --rm --env-file .env alpine env
```

```
$ docker container run --rm alpine echo $MYVAR
```

```
$ docker container run --rm alpine echo \
```

```
$ docker container run --rm -e TZ='Asia/Bangkok' debian:12-slim date
```

ทำงานกับ port ของ container

```
$ docker container run -d --name www nginx:alpine
```

```
$ docker container ls
```

```
$ docker container rm -f www
```

```
$ docker container run -d -p '8080:80' --name www nginx:alpine
```

```
$ curl localhost:8080
```

```
$ docker container port www
```

Monitor logs ของ container ด้วยคำสั่ง docker container logs

```
$ docker container run -d -p '8080:80' --name www nginx:alpine
```

```
$ docker container logs www
```

```
$ docker container logs -f www
```

```
$ docker container logs -f --tail 5 www
```

ท่อง container ด้วยคำสั่ง docker container exec

```
$ docker container run -d -p '8080:80' --name www nginx:alpine
```

```
$ docker container exec -it www sh
```

```
$ cd /usr/share/nginx/html
```

```
$ vi index.html
```

Copy file ระหว่าง Container กับ Host ด้วยคำสั่ง docker container cp

```
$ docker container run -d -p '8080:80' --name www nginx:alpine
```

```
$ docker container cp www:/usr/share/nginx/html/index.html .
```

```
# edit index.html
```

```
$ docker container cp index.html www:/usr/share/nginx/html/index.html
```

สร้าง Docker Image ด้วยคำสั่ง docker container commit

```
$ docker container commit www mynginx:limited
```

```
$ docker image ls
```

```
$ docker container run --rm -p '8081:80' --name www2 mynginx:limited
```

```
$ curl localhost:8081
```

Run container พร้อมกับ mount local volume โดยให้ copy ไฟล์จาก container มาไว้ที่ local directory แล้วแก้ไขตามต้องการ

```
$ docker container run -d -p '8080:80' --name www nginx:alpine
```

```
$ docker container cp www:/usr/share/nginx/html/index.html .
```

```
$ docker container cp www:/etc/nginx/conf.d/default.conf .
```

```
$ docker container rm -f www
```

```
$ mkdir www && mv index.html www
```

เมื่อแก้ไขเสร็จแล้วให้ run พร้อมกับ mount volume

```
$ docker container run -d -p '8080:80' \  
--name www \  
-v ${PWD}/www:/app \  
-v ${PWD}/default.conf:/etc/nginx/conf.d/default.conf:ro \  
nginx:alpine
```

```
$ curl localhost:8080
```

```
$ docker container inspect www
```

```
$ docker container inspect --format '{{ .Mounts }}' www
```

Docker Container Run with Working Directory

ตั้ง Working Directory ให้กับ container ด้วย -w หรือ --workdir

```
$ docker container run -d -p '8080:80' \  
--name www \  
-v ${PWD}/www:/app \  
-v ${PWD}/default.conf:/etc/nginx/conf.d/default.conf:ro \  
-w /app \  
nginx:alpine
```

```
$ docker container inspect --format '{{ .Config.WorkingDir }}' www
```

```
$ docker container exec -it www sh
```

```
$ pwd
```

Docker Volume คือ การแชร์ข้อมูลระหว่าง Host กับ Container และ Container กับ Container

```
$ docker volume ls
```

```
$ docker volume create vol1
```

```
$ docker volume ls
```

```
$ docker volume inspect vol1
```

```
$ docker container run --rm -v vol1:/data alpine touch /data/hello
```

```
$ docker container run --rm -v vol1:/data debian:12-slim ls -l /data
```

Docker Volume แบบ ระบุ mount point

```
$ docker volume create web -o type=none -o device=${PWD}/www -o o=bind
```

```
$ docker volume ls
```

```
$ docker volume inspect web
```

```
$ docker container run -d -p '8080:80' \  
--name www \  
-v web:/app \  
-v ${PWD}/default.conf:/etc/nginx/conf.d/default.conf:ro \  
-w /app \  
nginx:alpine
```

ลบ volume ด้วยคำสั่ง docker volume rm

\$ docker volume ls

\$ docker volume rm web

Docker Network เป็นการเชื่อมต่อระหว่าง Container หรือ Container กับ Host และ ยังสามารถเชื่อมต่อกับ Network อื่นๆ ได้อีกด้วย

```
$ docker network ls
```

```
$ docker network create db
```

```
$ docker network ls
```

```
$ docker network inspect db
```

Docker Network เป็นการเชื่อมต่อระหว่าง Container หรือ Container กับ Host และ ยังสามารถเชื่อมต่อกับ Network อื่นๆ ได้อีกด้วย

```
$ docker network ls
```

```
$ docker network create db
```

```
$ docker network inspect db
```

```
$ docker container run -d --name host1 --network db alpine sleep infinity
```

```
$ docker container run -d --name host2 alpine sleep infinity
```

Docker Network เป็นการเชื่อมต่อระหว่าง Container หรือ Container กับ Host และ ยังสามารถเชื่อมต่อกับ Network อื่นๆ ได้อีกด้วย

```
$ docker container exec -it host2 getent hosts host1
```

```
$ docker network connect db host2
```

```
$ docker container exec -it host2 getent hosts host1
```

```
$ docker container exec -it host2 ping host1
```

ตัวอย่างการเชื่อมต่อ Network ระหว่าง PostgreSQL และ PgAdmin

```
$ docker container run -d --name pgsql \  
-e POSTGRES_PASSWORD=1234 \  
-e POSTGRES_USER=postgres \  
-e POSTGRES_DB=postgres \  
postgres:alpine
```

```
$ docker container run -d --name pgadmin \  
-e PGADMIN_DEFAULT_EMAIL=admin@abc.com \  
-e PGADMIN_DEFAULT_PASSWORD=1234 \  
-p 5050:80 \  
elestio/pgadmin
```

ตัวอย่างการเชื่อมต่อ Network ระหว่าง PostgreSQL และ PgAdmin

```
$ docker container ls
```

```
# เปิด http://localhost:5050 ลองเชื่อมต่อ PgAdmin ไปยัง Host: postgresql
```

```
$ docker network connect postgresql
```

```
$ docker network connect pgadmin
```

```
# ลองเชื่อมต่อ PgAdmin ไปยัง Host: postgresql อีกครั้ง
```

```
$ docker container rm -f postgresql pgadmin
```

```
$ docker network rm db
```

Download Docker Build Materials



<https://github.com/mrchoke/docker-build-super-ai-workshop/>

ตัวอย่างการสร้าง Docker Image ด้วย Dockerfile

FROM alpine

CMD ["echo", "Hello Docker!"]

\$ docker build -t hello .

\$ docker run hello

ตัวอย่างการสร้าง Docker Image ด้วย Dockerfile

FROM alpine

```
$ docker build -t curl .
```

```
RUN apk add --no-cache curl
```

```
$ docker run curl
```

```
CMD ["curl", "https://www.google.com"]
```

ตัวอย่างการสร้าง Docker Image ด้วย Dockerfile Dockerfile.entrpoint

FROM alpine

RUN apk add --no-cache curl

ENTRYPOINT ["curl"]

\$ docker build -t entrpoint .

\$ docker run entrpoint

\$ docker run entrpoint https://www.google.com

\$ docker run entrpoint -s https://www.google.com

ตัวอย่างการสร้าง Vite Project Image

```
$ docker run --rm \  
-v $PWD:/app \  
-w /app \  
node:20-alpine \  
npm create vite@latest vue -- --template vue
```

```
$ docker build -t vue .
```

```
$ docker run --rm -p 3000:3000 vue
```

```
FROM node:20-alpine
```

```
WORKDIR /app
```

```
COPY . .
```

```
RUN npm install && npm run build
```

```
EXPOSE 3000
```

```
CMD ["npm", "run", "preview", "--", \  
"--port", "3000", "--host", "0.0.0.0"]
```

ตัวอย่างการสร้าง Python Project Image

```
from fastapi import FastAPI
```

```
app = FastAPI()
```

```
@app.get("/")
```

```
def read_root():
```

```
    return {"message": "I love FastAPI"}
```

```
FROM python:3.12-alpine
```

```
COPY . .
```

```
RUN pip install -r requirements.txt
```

```
EXPOSE 8000
```

```
CMD ["uvicorn", "main:app", \  
    "--host", "0.0.0.0", "--port", "8000"]
```

ตัวอย่างการสร้าง Golang Project Image

```
// main.go  
package main
```

```
import "fmt"
```

```
func main() {  
    fmt.Println("hello world")  
}
```

```
FROM golang  
WORKDIR /app
```

```
COPY main.go .
```

```
ENV CGO_ENABLED 0  
ENV GOOS linux  
ENV GOARCH ${TARGETARCH}
```

```
RUN go build -o hello main.go
```

```
CMD ["/app/hello"]
```

```
$ docker build -t hello .
```

```
$ docker run --rm hello
```

```
$ docker image ls
```

ตัวอย่างการสร้าง Docker Image แบบ Multi-stage Dockerfile.multi

```
FROM golang AS builder
```

```
FROM scratch
```

```
$ docker build -t hello:2 . -f Dockerfile.multi
```

```
WORKDIR /app
```

```
COPY --from=builder /app/hello /hello
```

```
$ docker run --rm hello:2
```

```
COPY main.go .
```

```
CMD ["/hello"]
```

```
$ docker image ls
```

```
ENV CGO_ENABLED 0
```

```
ENV GOOS linux
```

```
ENV GOARCH ${TARGETARCH}
```

```
RUN go build -o hello main.go
```

เราสามารถสร้าง Docker Image และเผยแพร่ไปยัง Docker Hub ได้

```
$ docker login
```

```
$ docker tag hello:2 mrchoke/hello:latest
```

```
$ docker push mrchoke/hello:latest
```


ตัวอย่างการสร้าง Docker Image สำหรับหลาย Architecture

docker buildx version

docker buildx ls

docker buildx create --name builder

docker buildx inspect --bootstrap builder

docker buildx use builder

```
$ docker buildx build \  
--platform linux/arm64,linux/amd64 \  
-t mrchoke/hello:multiarch \  
--push .
```