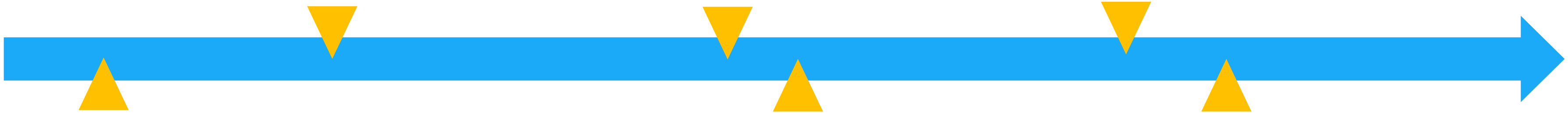
# Introduction to Docker

# Docker and Container Overview

docker

↱ Google ใช้

# History of Docker



**2004**
Solaris Containers / Zones technology introduced

**2008**
Linux containers (LXC 1.0) introduced

**2013**
Solomon Hykes starts Docker as an internal project within dotCloud

**Mar 2013**
Docker released to open source

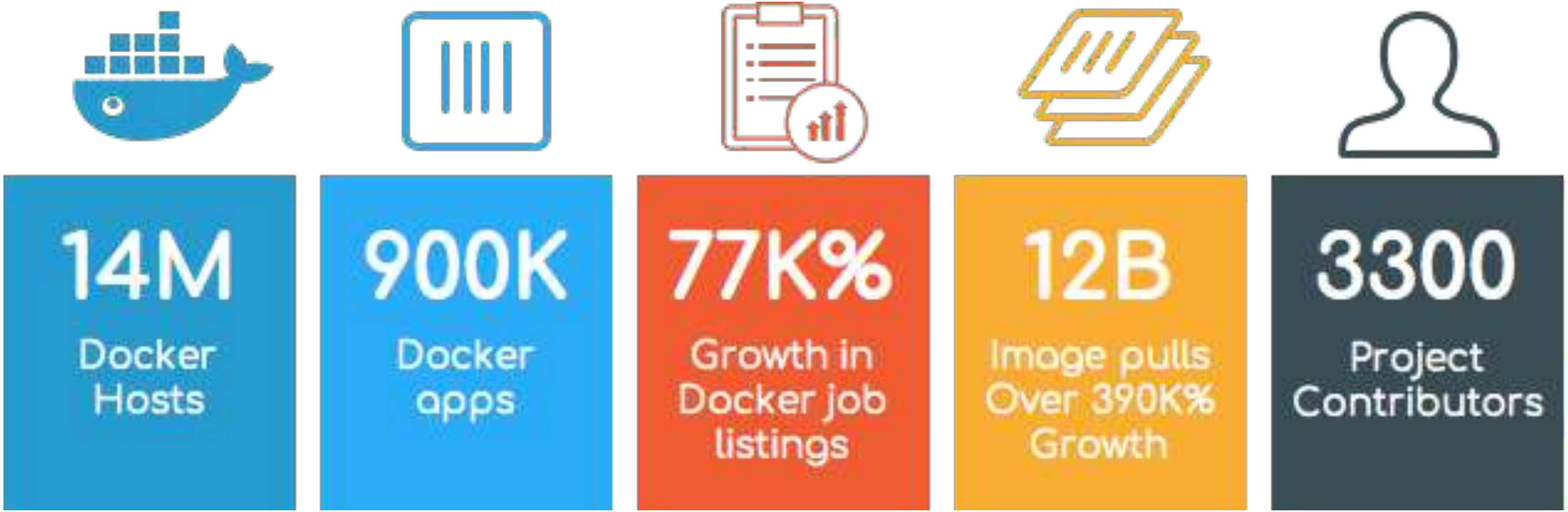
**Feb 2016**
Docker introduces first commercial product – now called Docker Enterprise Edition

**Today**
Open source community includes:

- 3,300+ contributors
- 43,000+ stars
- 12,000+ forks

# Incredible adoption in just 4 years

เป็นสิ่งที่โตเร็วอันดับ 2-3 ของโลก



14M
Docker Hosts

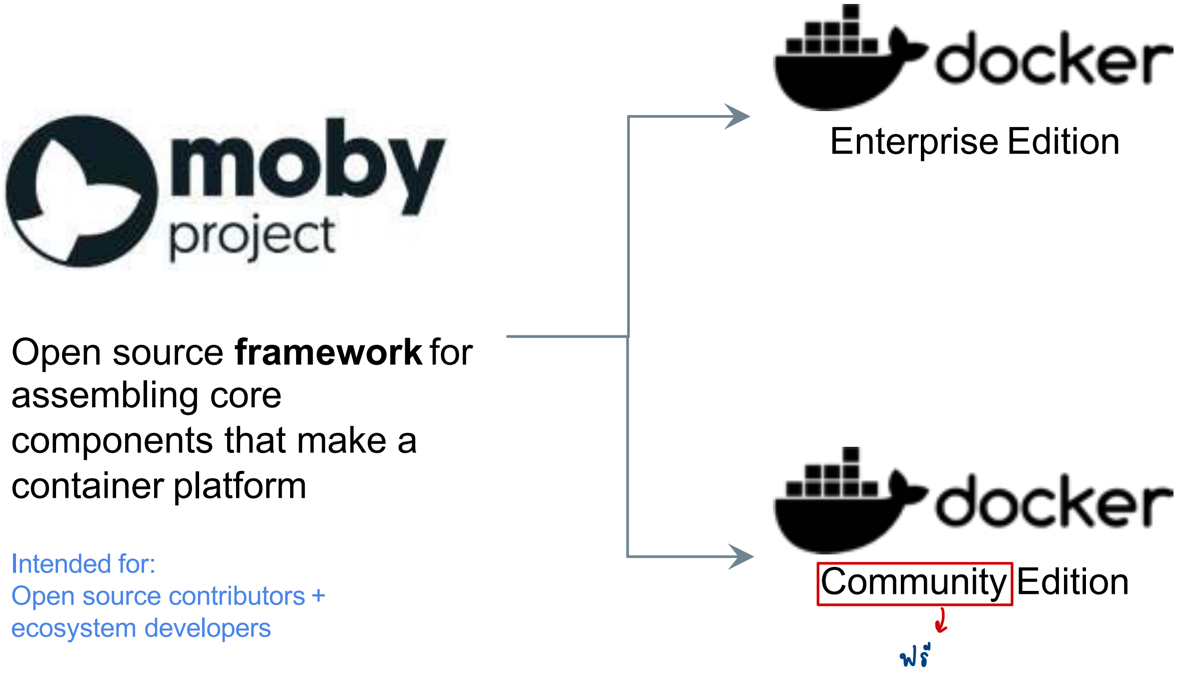
900K
Docker apps

77K%
Growth in Docker job listings

12B
Image pulls Over 390K% Growth

3300
Project Contributors

# The Docker Family Tree



Open source **framework** for assembling core components that make a container platform

Intended for:
Open source contributors + ecosystem developers

Subscription-based, commercially supported **products** for delivering a secure software supply chain

Intended for:
Production deployments + Enterprise customers

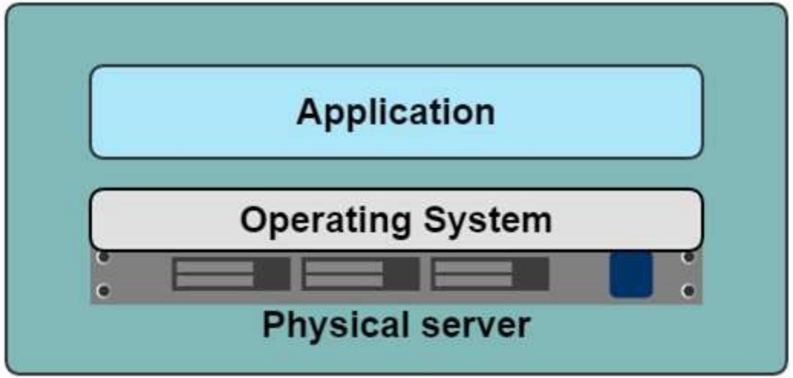
Free, community-supported **product** for delivering a container solution

Intended for:
Software dev & test

# A History Lesson

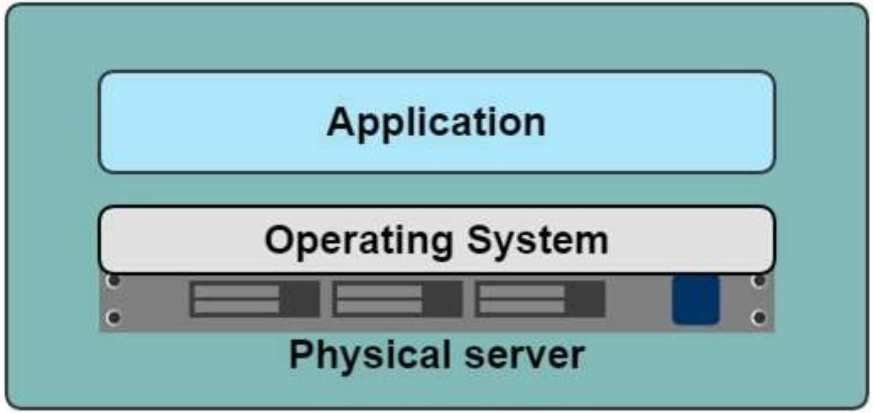
## In the Dark Ages

**One application on one physical server**



# Historical limitations of application deployment

- Slow deployment times
- Huge costs
- Wasted resources
- Difficult to scale
- Difficult to migrate
- Vendor lock in



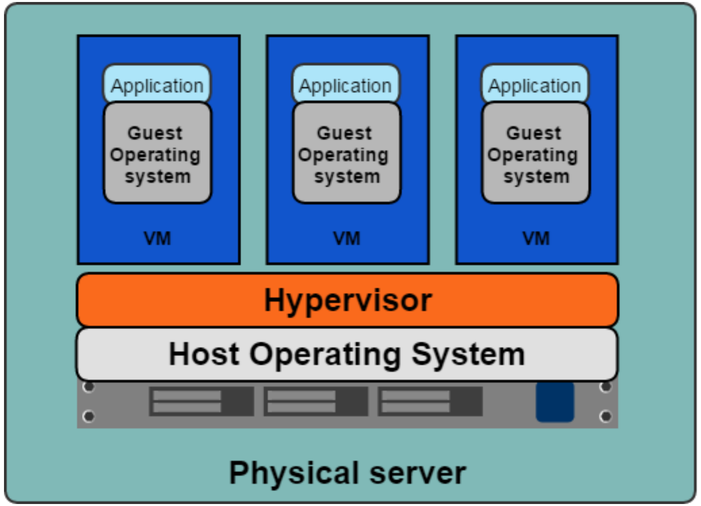
# A History Lesson

ยุคของ VM

## Hypervisor-based Virtualization

- One physical server can contain multiple applications
- Each application runs in a virtual machine (VM)

ทำให้รัน OS หลายตัวได้



# Benefits of VMs

- Better resource pooling
  - One physical machine divided into multiple virtual machines
- Easier to scale
- VMs in the cloud
  - Rapid elasticity
  - Pay as you go model

Microsoft Azure

amazon web services™

vmware®

# Limitations of VMs

แต่ละ VM ใช้ทรัพยากรแยกกัน

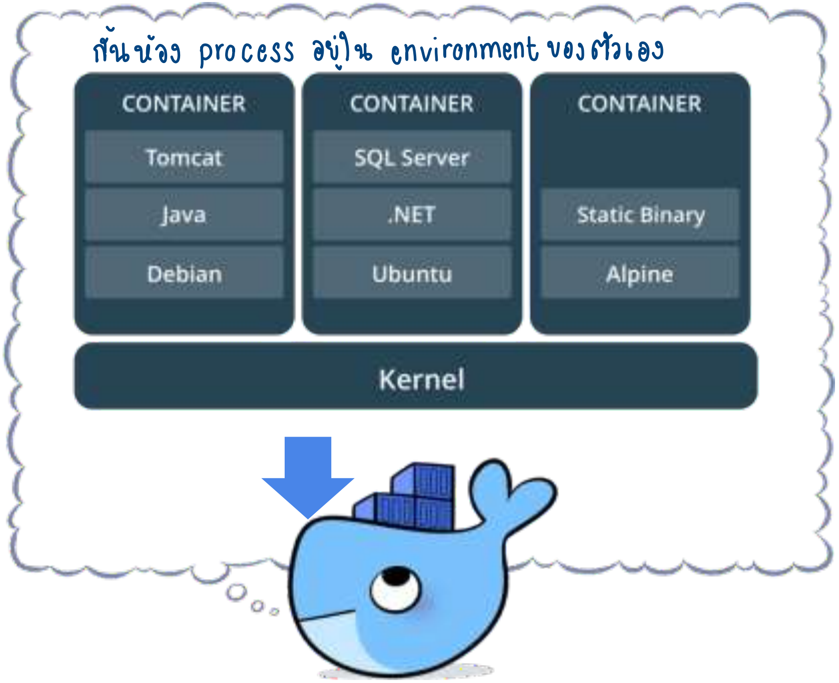
→ ข้อเสีย VM ขยาย Disk ยาก ให้ 2 GB แล้วเต็ม ลงทั้งทำใหม่ทั้งตัว

- Each VM stills requires
  - CPU allocation
  - Storage
  - RAM
  - An entire guest operating system
- The more VMs you run, the more resources you need
- Guest OS means wasted resources
- Application portability not guaranteed

# What is a container?

การแยกprocess

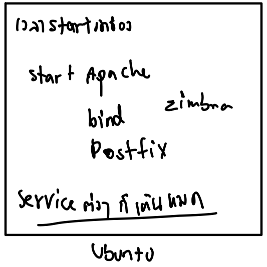
แนวความคิด container

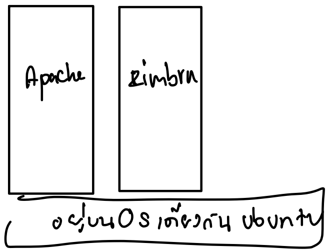


กั้นห้อง process อยู่ใน environment ของตัวเอง

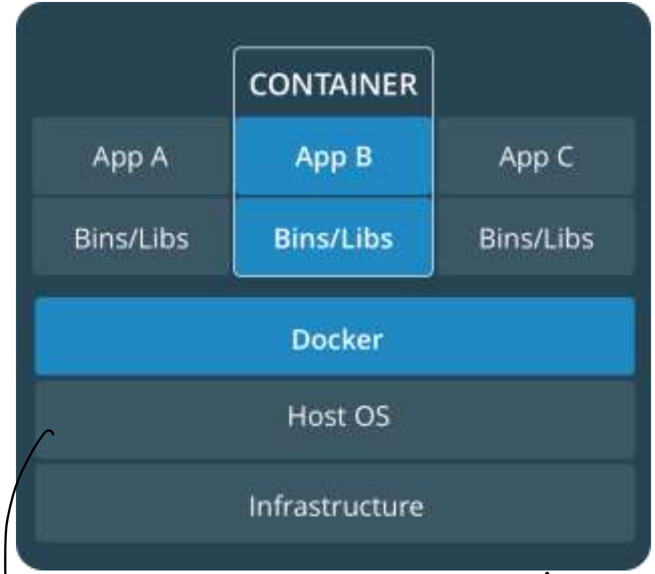
- Standardized packaging for software and dependencies
- Isolate apps from each other
- Share the same OS kernel
- Works with all major Linux and Windows Server

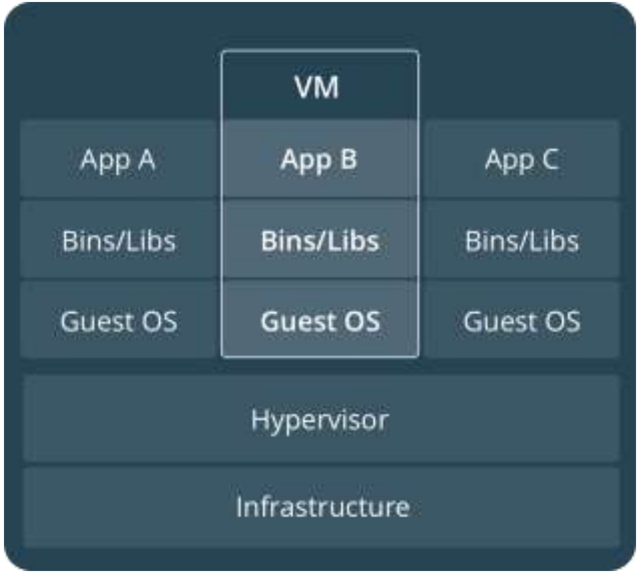
Patition processให้อยู่ในห้องของตนเอง





# Comparing Containers and VMs





ใช้ Host ได้ร่วมกัน / Container จะ support ตัว App (โปรแกรมต่างๆ)
โปรแกรมที่จะรัน ต้องรองรับ Host OS

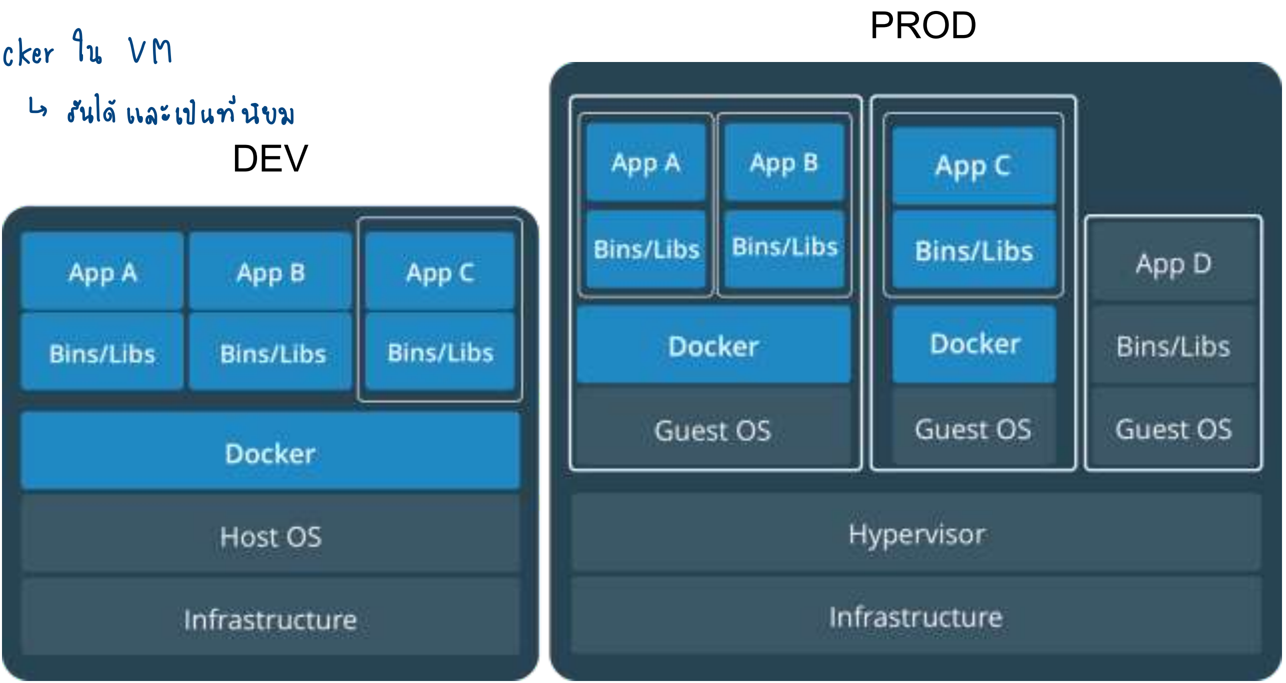
Containers are an app level construct

VMs are an infrastructure level construct to turn one machine into many servers

# Containers and VMs together

รัน Docker ใน VM

↳ รันได้ และเปนท่นิยม



DEV zone

Production zone

Containers and VMs together provide a tremendous amount of flexibility for IT to optimally deploy and manage apps.

# Key Benefits of Docker Containers

เร็ว

## Speed

- No OS to boot = applications online in seconds

ย้ายไปย้ายมาได้

## Portability

- Less dependencies between process layers = ability to move between infrastructure

ใช้ทรัพยากรนอย

## Efficiency

- Less OS overhead
- Improved VM density

# Container Solutions & Landscape

# Docker Basics ทรัพยากรของ Docker

**Image** → File ที่เก็บของกับพวกทั้งงาน โปรแกรม

The basis of a Docker container. The content at rest.

**Container** (environment)

The image when it is ‘running.’ The standard unit for app service

**Engine** ตัวที่จะรันใน container

The software that executes commands for containers. Networking and volumes are part of Engine. Can be clustered together.

**Registry** ที่เอาไว้เก็บ image

Stores, distributes and manages Docker images

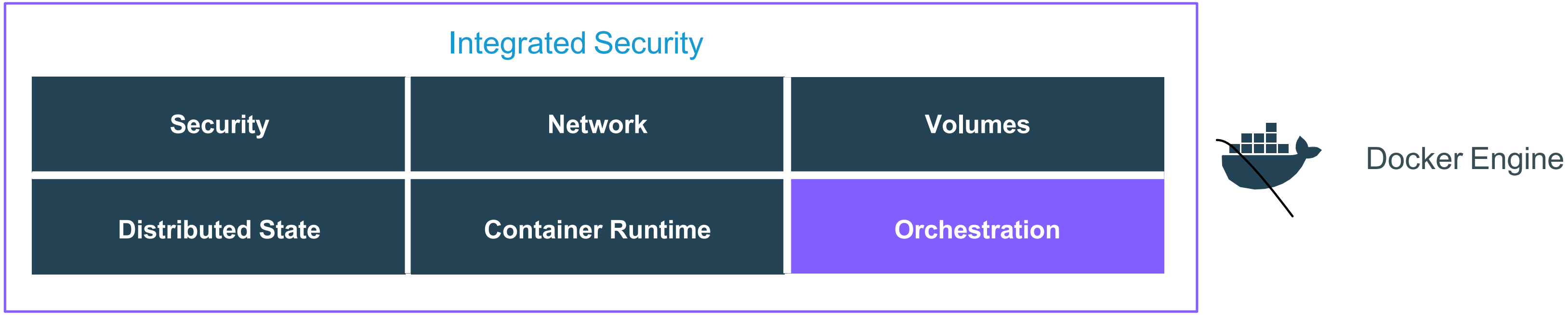
**Control Plane** บริหาร จัดการ

Management plane for container and cluster orchestration

# Foundation: Docker Engine

↳ จัดการ disk , network

บริหารการทำงาน

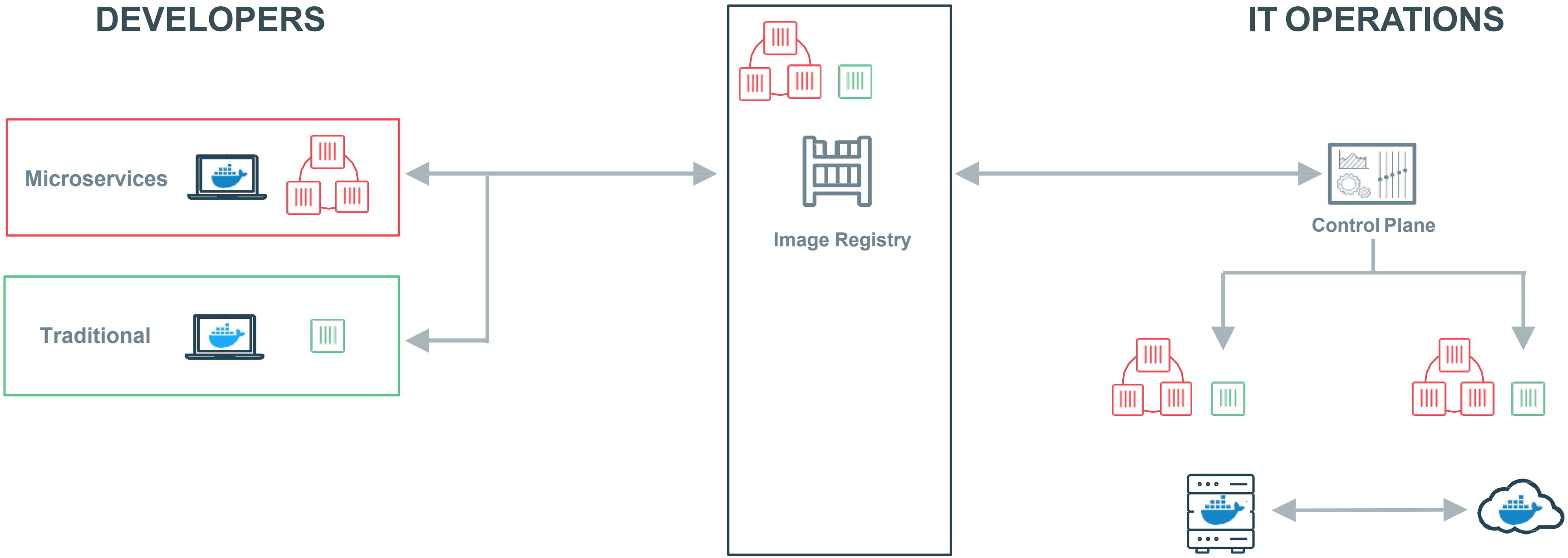


Integrated Security

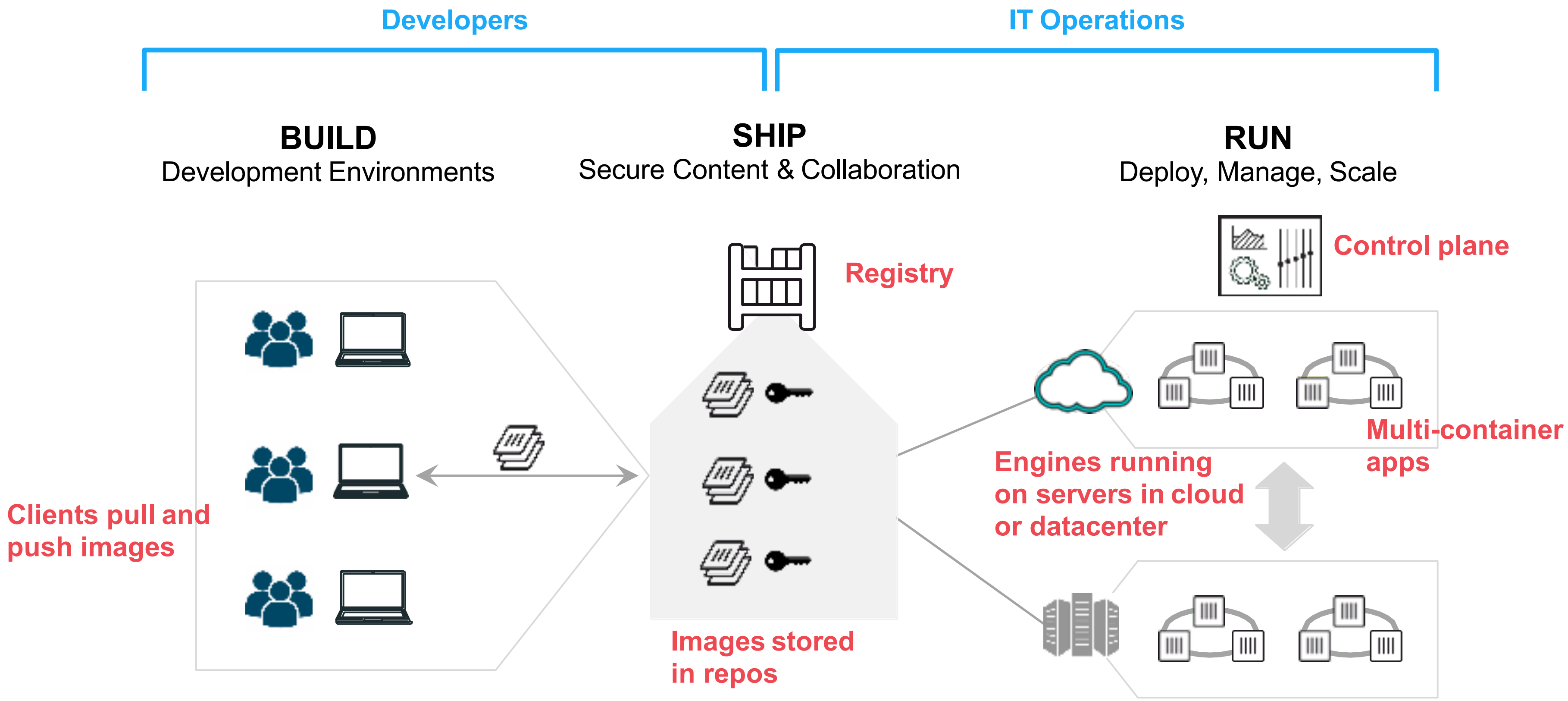
| | | |
|---|---|---|
| Security | Network | Volumes |
| Distributed State | Container Runtime | Orchestration |

Docker Engine

# Building a Software Supply Chain



# Containers as a Service



**Developers**

**IT Operations**

**BUILD**
Development Environments

**SHIP**
Secure Content & Collaboration

**RUN**
Deploy, Manage, Scale

Registry

Control plane

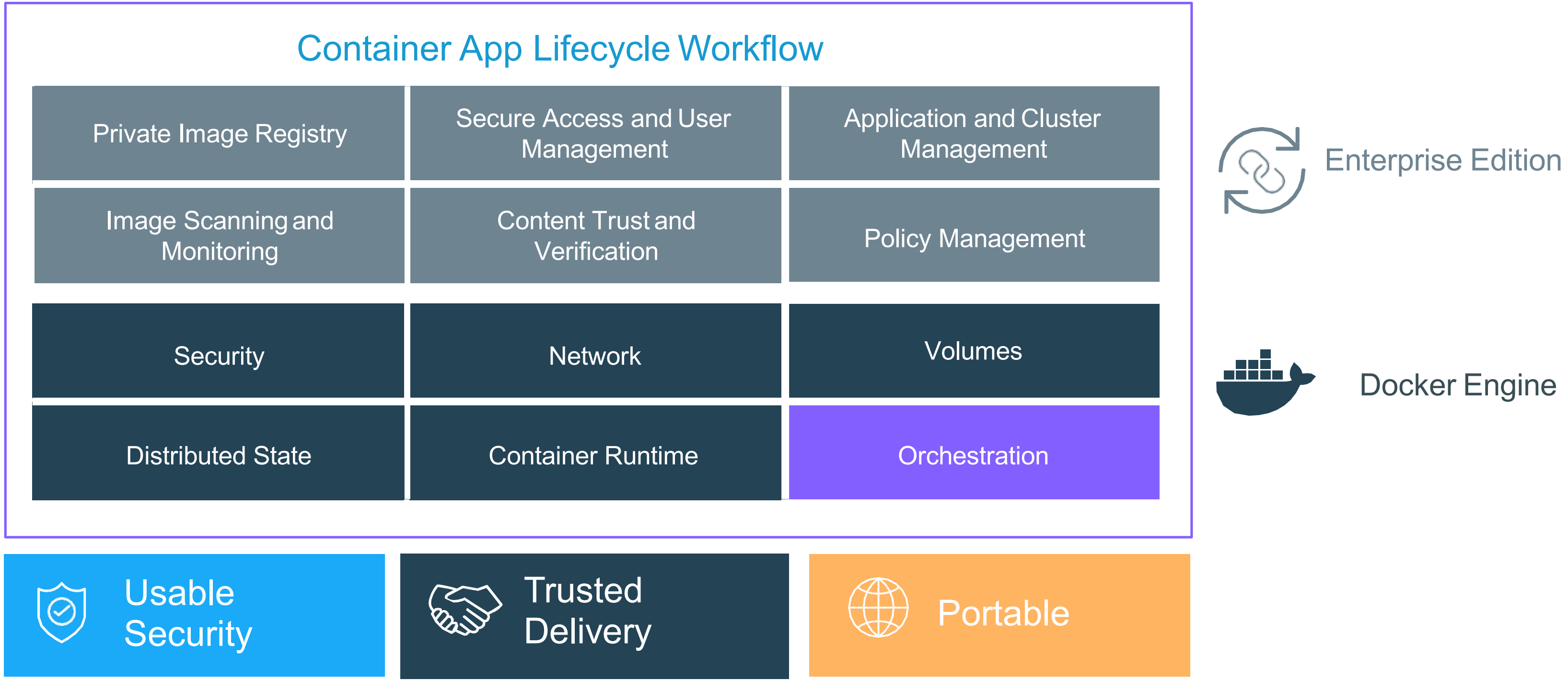
Clients pull and push images

Images stored in repos

Engines running on servers in cloud or datacenter

Multi-container apps

# Building a Secure Supply Chain



Container App Lifecycle Workflow

Private Image Registry

Secure Access and User Management

Application and Cluster Management

Image Scanning and Monitoring

Content Trust and Verification

Policy Management

Security

Network

Volumes

Distributed State

Container Runtime

Orchestration

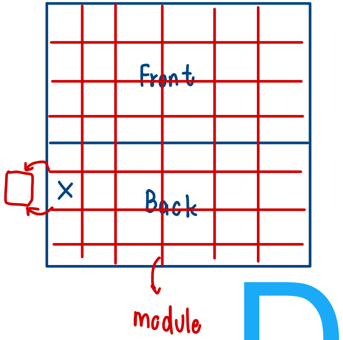
Enterprise Edition

Docker Engine

Usable Security

Trusted Delivery

Portable



มีปัญหา 1 รก

2. แบ่งการทำงานลำบาก

ข้อดี ช่วยกันทำได้

เวลา fail ก็ fail เฉพาะก้อนนั้นเท่านั้น

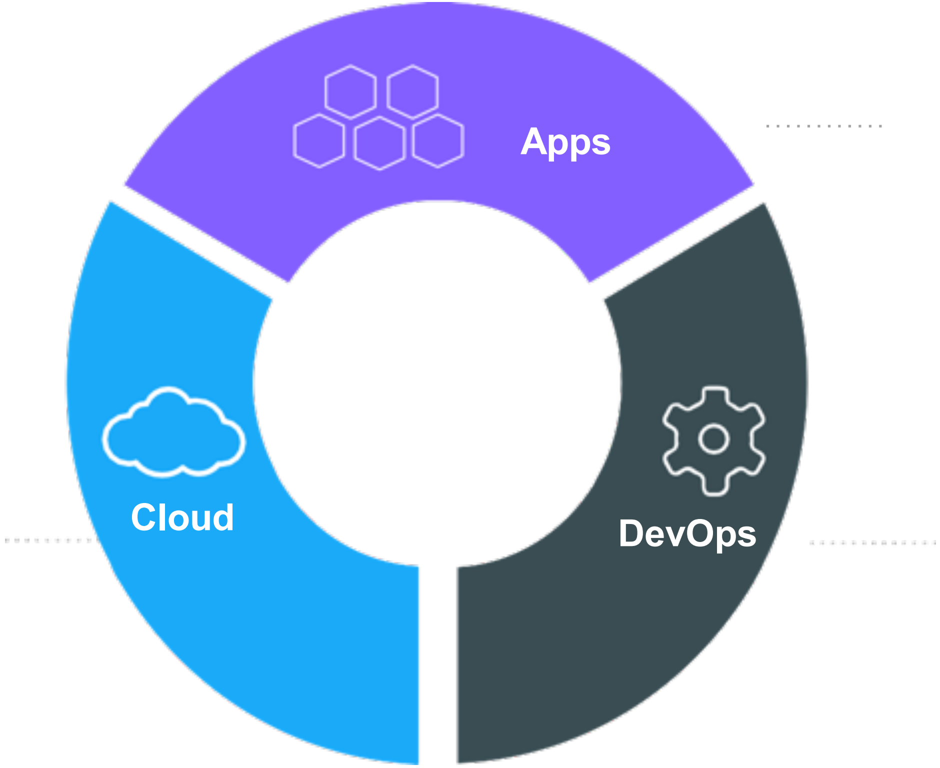
ข้อเสีย ใช้ทรัพยากรเยอะ

# Docker and Microservices

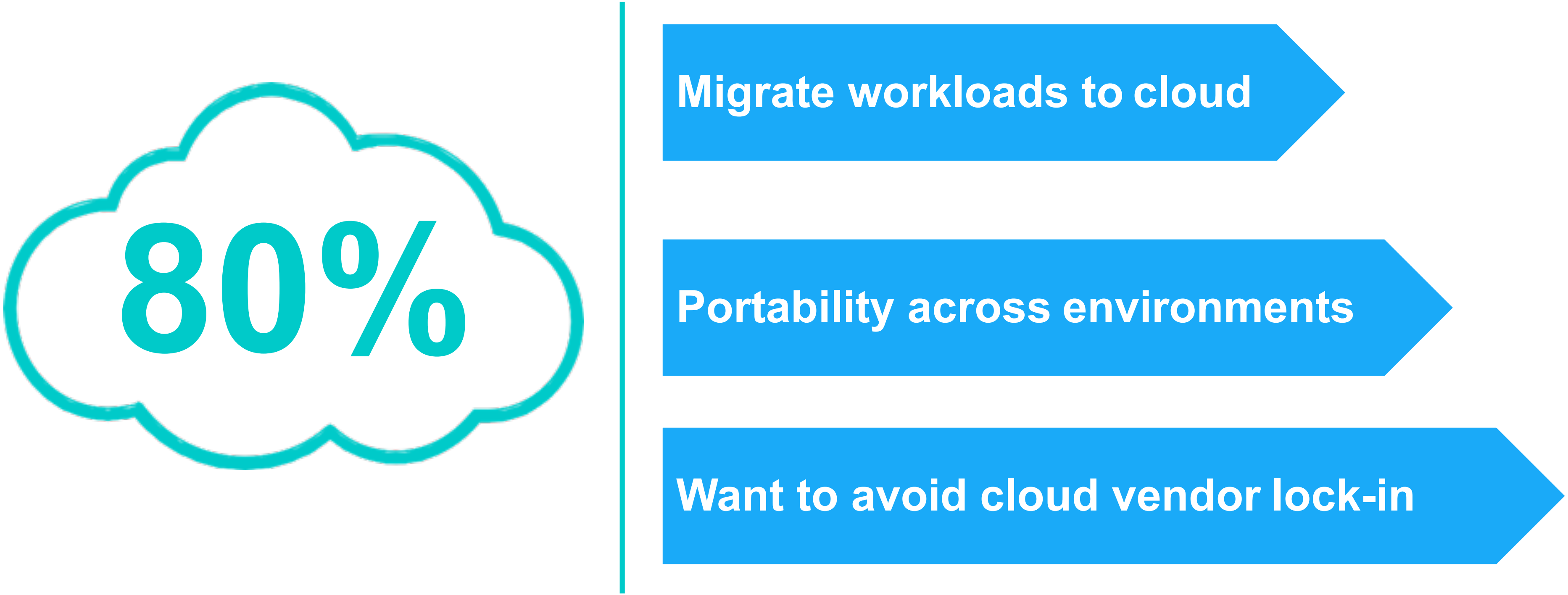
↳ ซอย module เปน services แล้ว deploy แยกจากกัน

แยกสัดส่วนชัดเจน

# The IT Landscape is Changing



# Movement in the cloud

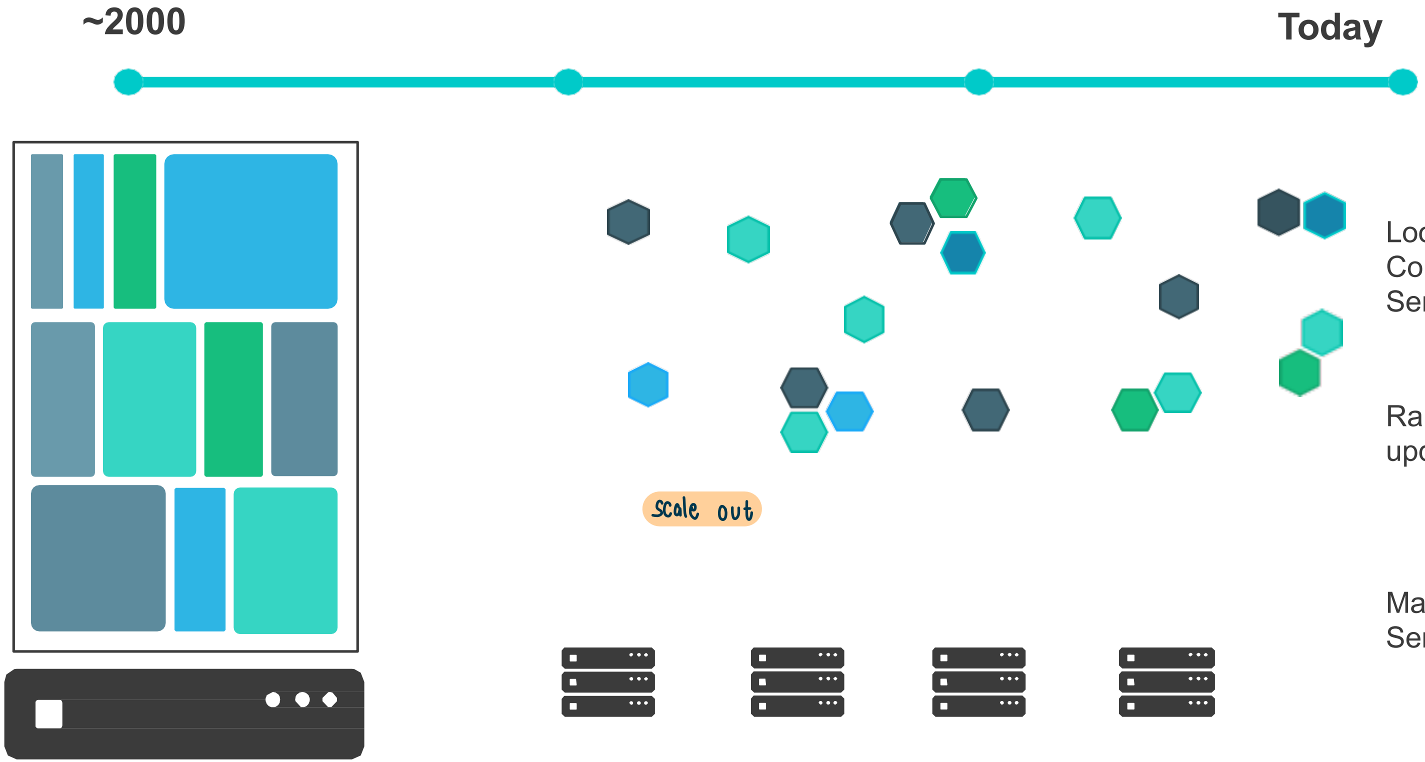


80%

- Migrate workloads to cloud
- Portability across environments
- Want to avoid cloud vendor lock-in

State of App development Survey: Q1 - 2016

# Applications are transforming



~2000

Today

Monolithic

Slow changing

Big Servers

Loosely Coupled Services

Rapidly updated

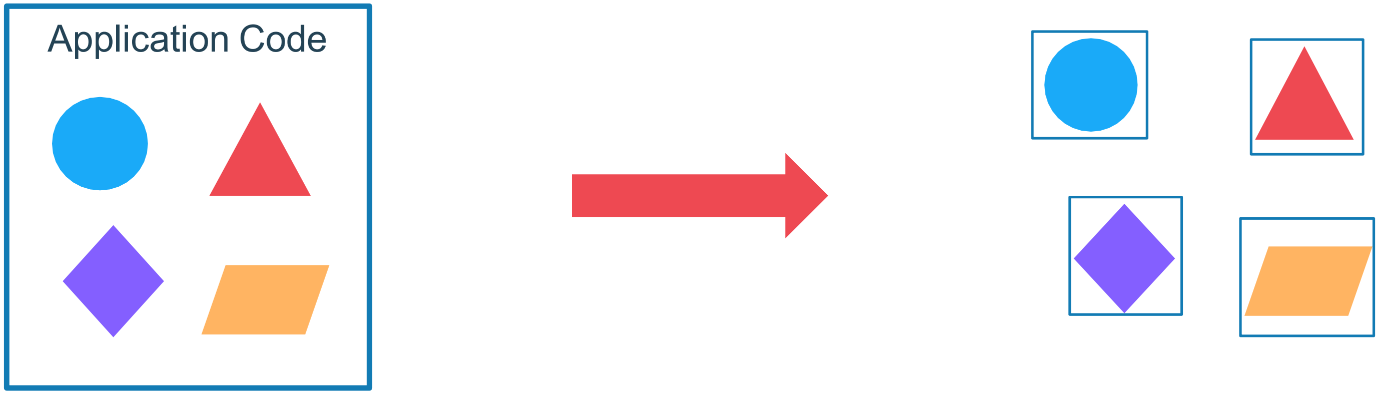
Many Small Servers or devices

Scale out

หลักการ scale up เติมเครื่องเข้าไป

# Application Modernization

รันแบบนี้ด้วย containe



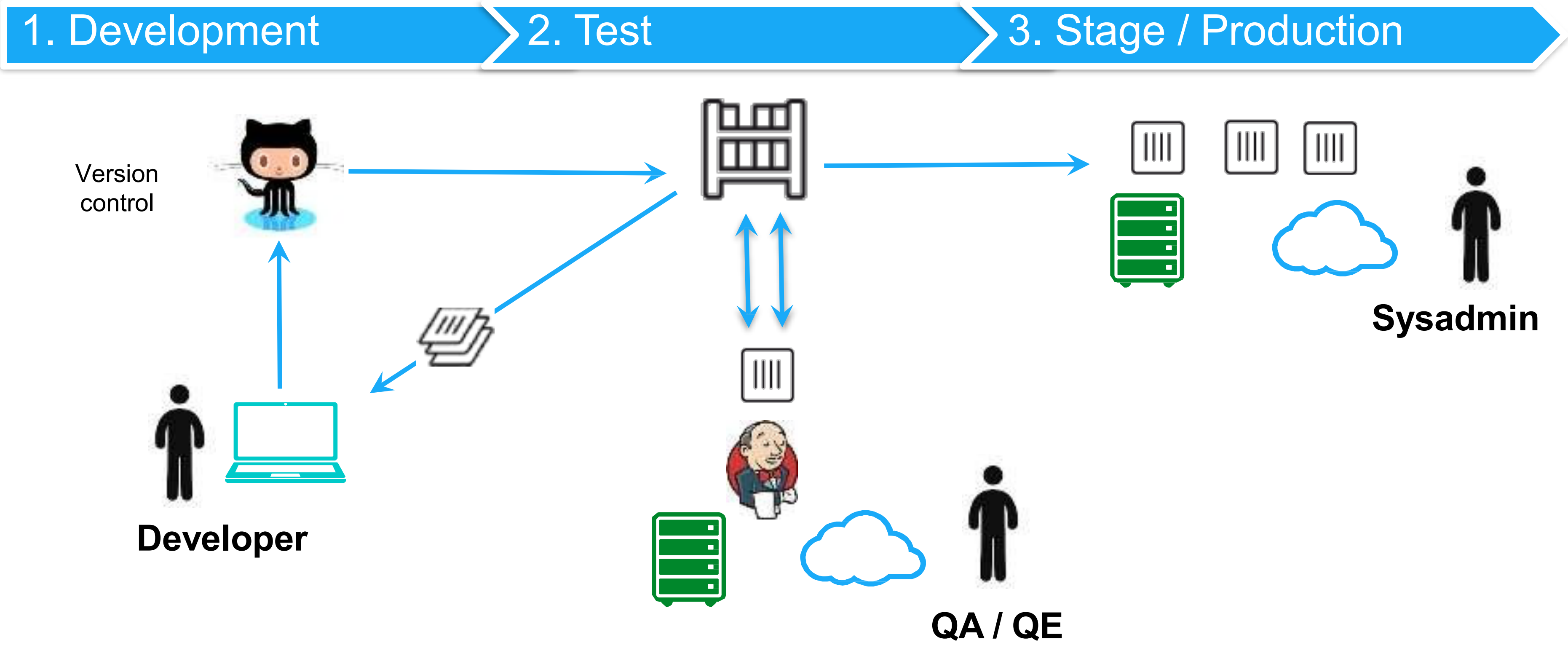
**Developer Issues:**

- Minor code changes require full re-compile and re-test
- Application becomes single point of failure
- Application is difficult to scale

**Microservices**: Break application into separate operations

**12-Factor Apps**: Make the app independently scalable, stateless, highly available by design

# Continuous Integration and Delivery



# The Myth of Bi-Modal IT

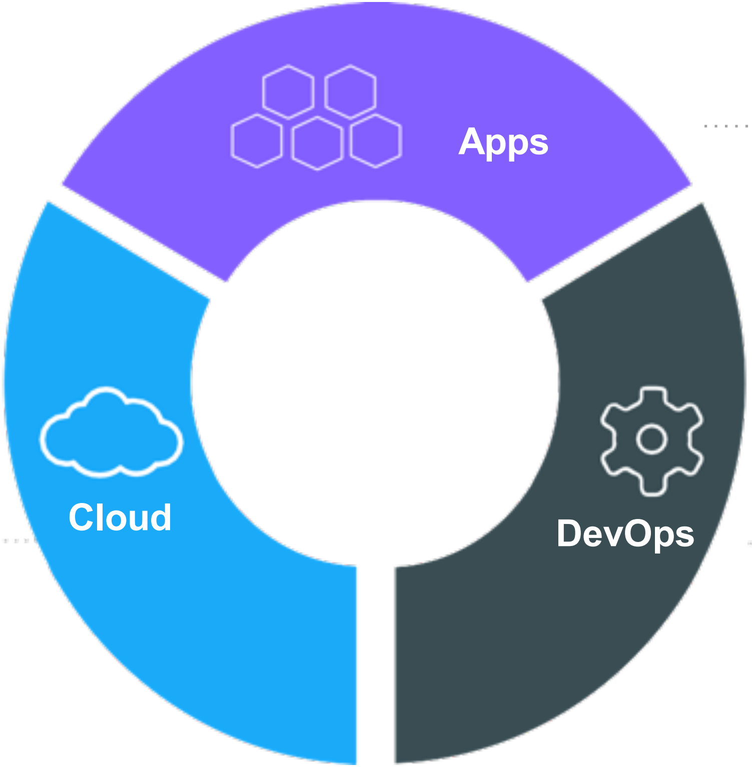
| | MICROSERVICES | TRADITIONAL APPS |
|---|---|---|
| Cloud or New Infrastructure | You are either here.. | ▨ |
| Old Infrastructure | ▨ | …or here |

# Enabling a Journey

| | MICROSERVICES | AGILE TRADITIONAL APPS | TRADITIONAL APPS |
|---|---|---|---|
| Cloud or New Infrastructure | | | |
| Old Infrastructure | | | |

...that is past AND future proof

# Docker Aligns to Multiple IT Initiatives



**3 out 4**

Top initiatives are app modernization

**80%**

Looking at Docker for Cloud Strategy

**44%**

Looking to adopt DevOps with Docker

State of App development Survey: Q1 2016

# Docker Is in the Enterprise

## Service Provider

- AT&T
- Sprint
- verizon

## Healthcare & Science

- AMGEN
- HudsonAlpha Institute for Biotechnology
- MERCK
- OPTUM

## Financial Services

- Goldman Sachs
- Northern Trust
- Societe Generale
- VISA
- ADP

## Tech

- Juniper Networks
- RSA
- TIBCO
- splunk>
- GE

## Insurance

- Anthem
- Liberty Mutual Insurance
- MetLife
- Mutual of Omaha

## Public Sector

- CDC
- Cornell University Founded A.D. 1865
- GSA
- Indiana University

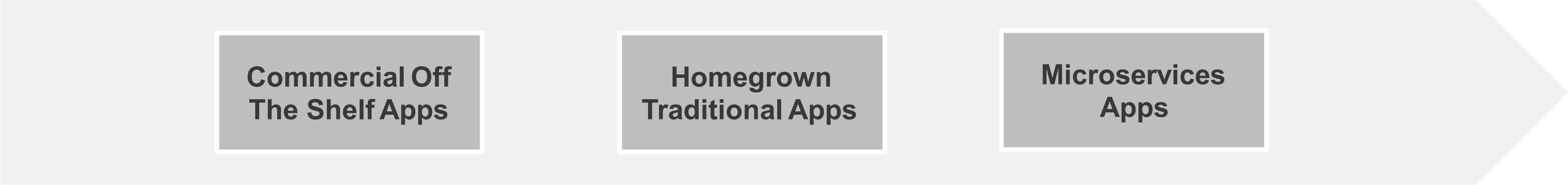
# Docker delivers agility, security and cost savings

Hardened containers deliver new levels of security to monoliths on the transition to microservices

Transform monoliths to secure and agile DevOps environments

Reduce maintenance costs by 10X for legacy, commercial and new apps

# Docker delivers agility, resiliency, portability security and cost savings for all applications



Commercial Off The Shelf Apps

Homegrown Traditional Apps

Microservices Apps

**13X**

More software releases

**65%**

Reduction in developer onboarding time

**~47%**

Reduction in VMs, OS licensing and Server costs

**Eliminate**

“works on my machine” issues

**62%**

Report reduction in MTTR

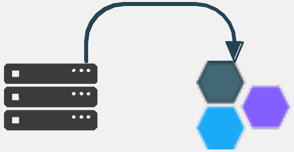
**10X**

Cost reduction in maintaining existing applications

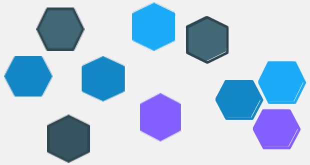
# One platform and one journey for all applications

1 **Traditional apps in containers**
Gain portability, efficiency and security

2 **Transform to Microservices**
Look for shared services to transform



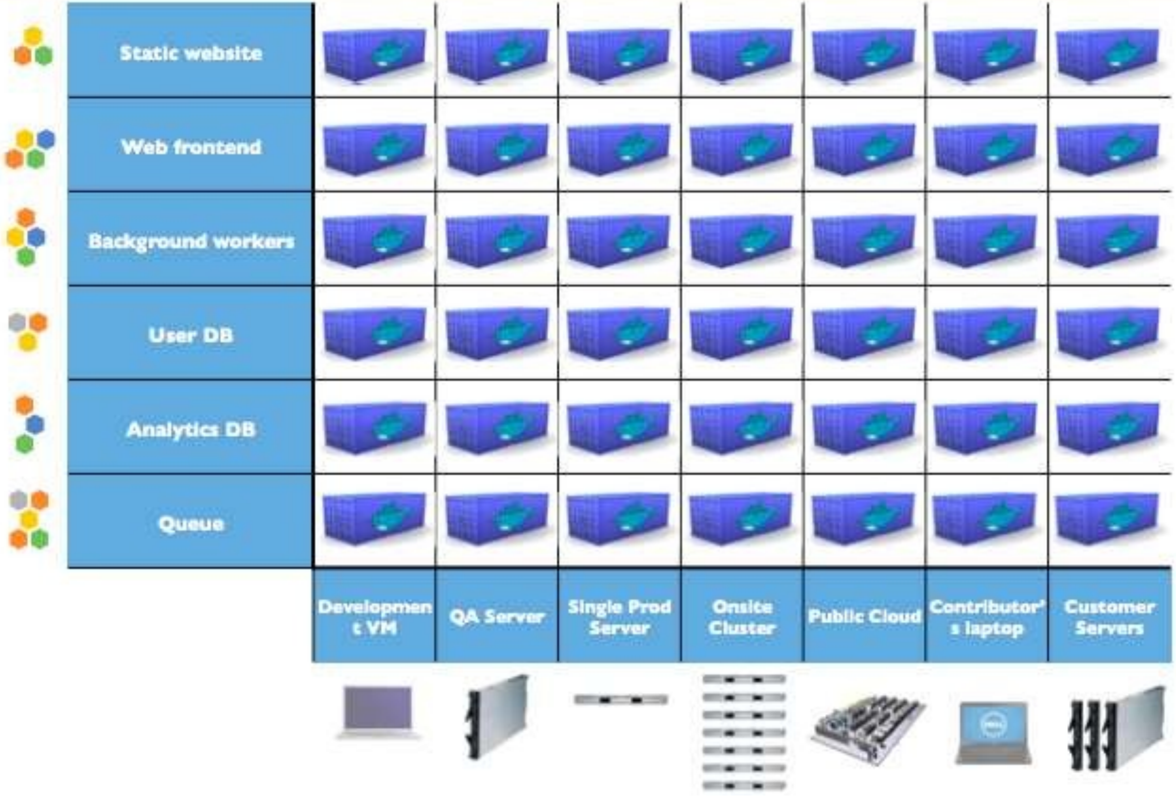
3 **Accelerate New Applications**
Greenfield innovation



# Multiple Stacks, Multiple Stages = Complexity

| | Development VM | QA Server | Single Prod Server | Onsite Cluster | Public Cloud | Contributor's laptop | Customer Servers |
|---|---|---|---|---|---|---|---|
| Static website | ? | ? | ? | ? | ? | ? | ? |
| Web frontend | ? | ? | ? | ? | ? | ? | ? |
| Background workers | ? | ? | ? | ? | ? | ? | ? |
| User DB | ? | ? | ? | ? | ? | ? | ? |
| Analytics DB | ? | ? | ? | ? | ? | ? | ? |
| Queue | ? | ? | ? | ? | ? | ? | ? |

# Solving the deployment matrix



Static website

Web frontend

Background workers

User DB

Analytics DB

Queue

Development VM

QA Server

Single Prod Server

Onsite Cluster

Public Cloud

Contributor's laptop

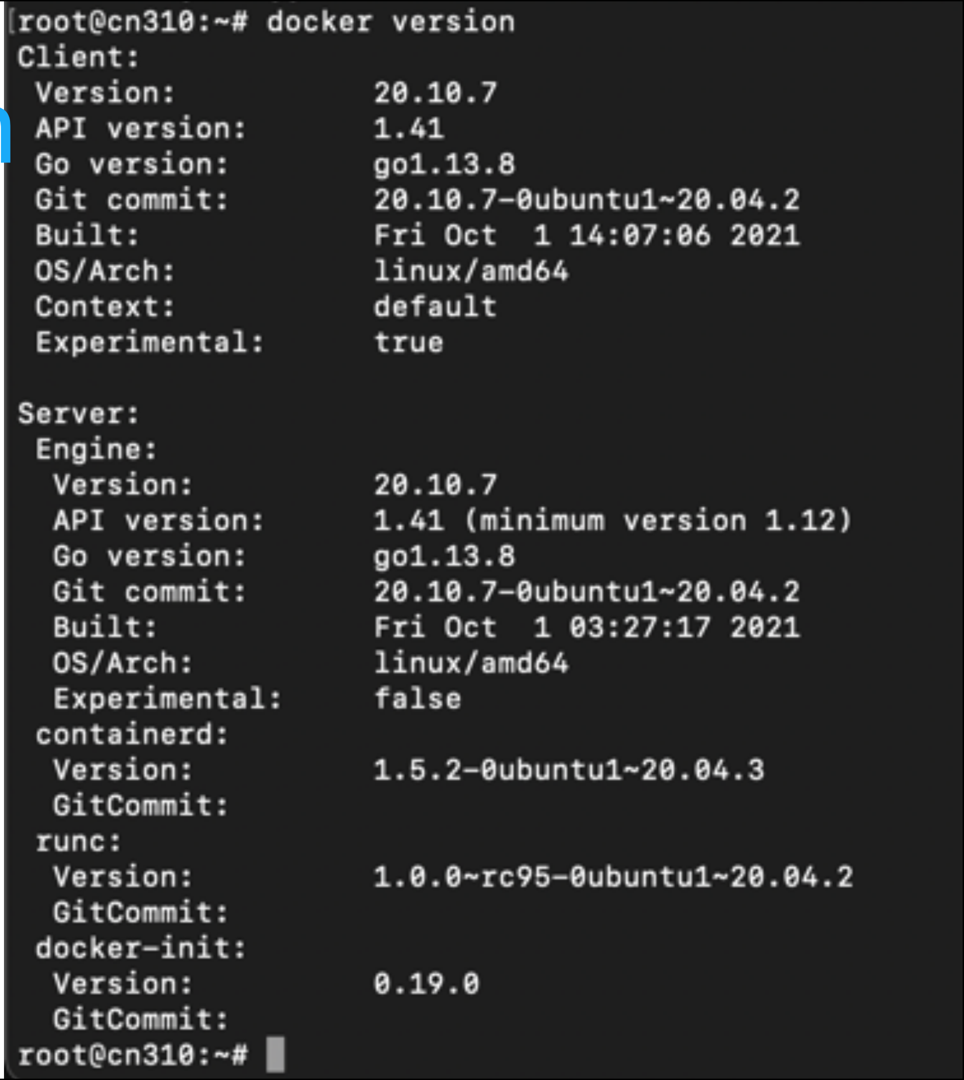
Customer Servers

# Docker101

# Docker Installation

#sudo apt install –y docker.io

#sudo docker version



```
root@cn310:~# docker version
Client:
 Version:           20.10.7
 API version:       1.41
 Go version:        go1.13.8
 Git commit:        20.10.7-0ubuntu1~20.04.2
 Built:             Fri Oct  1 14:07:06 2021
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server:
 Engine:
  Version:          20.10.7
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.13.8
  Git commit:       20.10.7-0ubuntu1~20.04.2
  Built:            Fri Oct  1 03:27:17 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.5.2-0ubuntu1~20.04.3
  GitCommit:
 runc:
  Version:          1.0.0~rc95-0ubuntu1~20.04.2
  GitCommit:
 docker-init:
  Version:          0.19.0
  GitCommit:
root@cn310:~#
```

# Docker Run

#sudo su –

#docker search ubuntu

**#docker pull ubuntu** ดาวน์โหลด

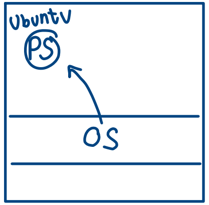
ให้ปริ้นมาให้

**#docker run ubuntu /bin/echo "Welcome to the Docker World!”** คำสั่งในการรัน Ubuntu

ตัวเราเข้าไปอยู่ใน container นั้นด้วย

**#docker run -it ubuntu /bin/bash**

↳ Option "interactive"



## Container’s Console

โชว์ชื่อ docker

```
root@0c80f908e41e:/# uname -a
Linux 0c80f908e41e 5.4.0-26-generic #30-Ubuntu SMP Mon Apr 20 16:58:30 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
root@0c80f908e41e:/# exit
exit
```

ออกจาก docker

# Docker Installation

Control + p + q   จะเด้งออกมาให้ แต่ docker ยังอยู่

```
# docker run -it ubuntu /bin/bash
root@3883a5e11c57:/# # Ctrl+p, Ctrl+q
root@cn310:~#

# show docker process
root@cn310:~# docker ps    ตรวจสอบ
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3883a5e11c57 ubuntu "/bin/bash" 19 seconds ago Up 18 seconds supakit
# connect to container's session

                                    กลับเข้าไปใหม่
root@cn310:~# docker attach 3883a5e11c57

root@3883a5e11c57:/#
# shutdown container's process from Host's console

root@cn310:~# docker kill 3883a5e11c57

3883a5e11c57
root@cn310:~# docker ps
```

# Docker images

```
root@cn310:~# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
ubuntu latest 7e0aa2d69a15 2 weeks ago 72.7MB
# start a Container and install nginx
```

อยากให้ docker install nginx

```
root@cn310:~# docker run ubuntu /bin/bash -c "apt-get update; apt-get -y install nginx;"
root@cn310:~# docker ps -a | head -2

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
555bbffdd1aa ubuntu "/bin/bash -c 'apt-g…" 36 seconds ago Exited (0) 14 seconds ago wonderful_nightingale
# add the image
root@cn310:~# docker commit 555bbffdd1aa  cn310/ubuntu-nginx

sha256:8f1fbe417eb2f1260495629f350c75324368d9bce9c61262158813987f085273
root@cn310:~# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
cn310/ubuntu-nginx latest 8f1fbe417eb2 16 seconds ago 160MB
ubuntu latest 7e0aa2d69a15 2 weeks ago 72.7MB
```

ให้คนข้างนอกมา access กันได้

# Docker mapping port to container

**# map the port of Host and the port of Container with [-p xxx:xxx]**

รันแล้วยังค้างอยู่

→ port

```
root@cn310:~# docker run -t -d -p 8081:80 cn310/ubuntu-nginx /usr/sbin/nginx -g "daemon off;"
```

↳ daemon

ค่าสั่ง start

```
fdb3a02ff0140cb9aefff34b7ee740855d2949450b3f7c068cf4a693f3f5e96b
root@cn310:~# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
fdb3a02ff014 cn310/ubuntu-nginx "/usr/sbin/nginx -g …" 8 seconds ago Up 8 seconds 0.0.0.0:8081->80/tcp priceless_pascal
```

**# create a test page**

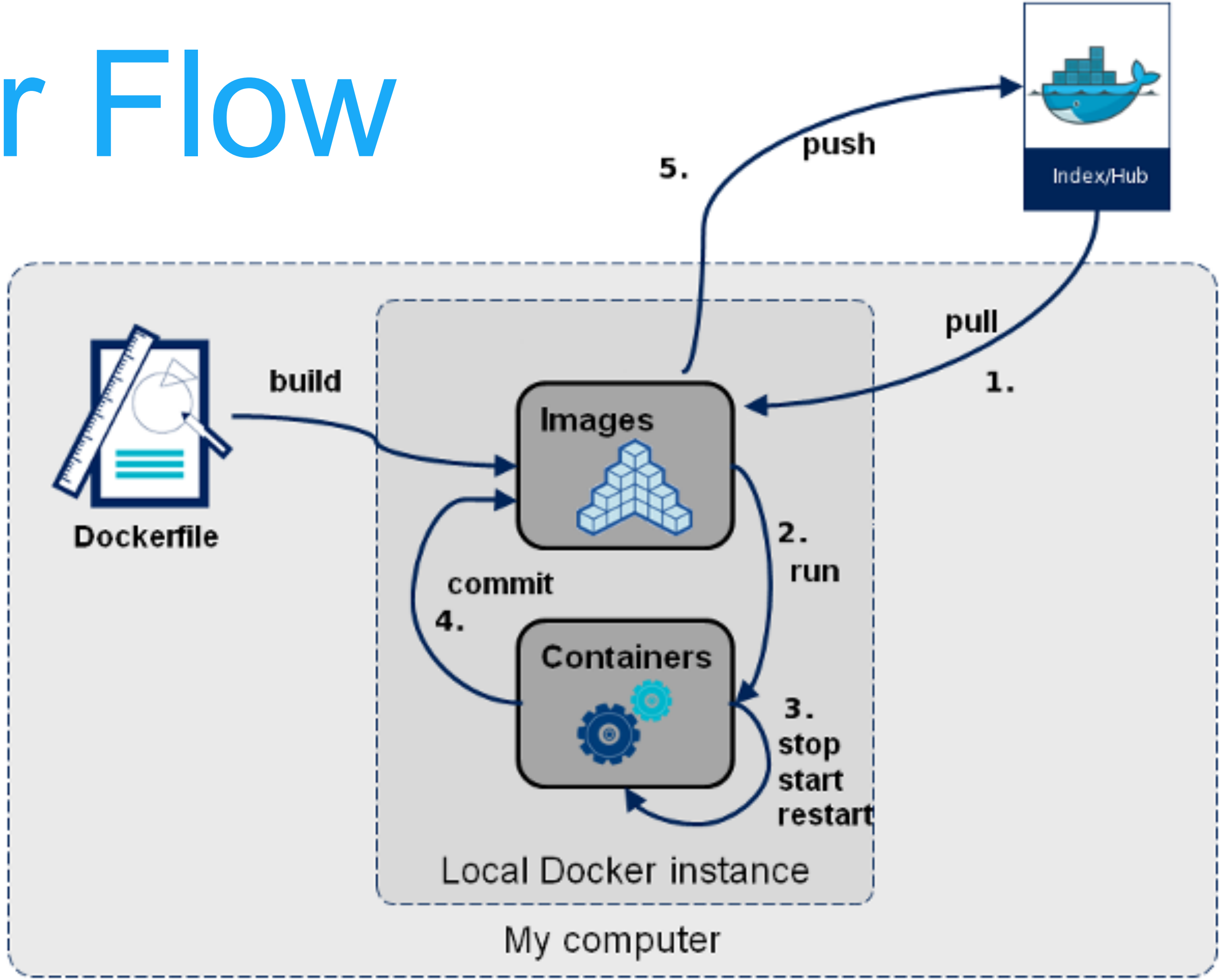
```
root@cn310:~# docker exec fdb3a02ff014 /bin/bash -c 'echo "Nginx on Docker Container" > /var/www/html/index.html’
```

**# verify it works normally**

curl เรียก brownser

```
root@cn310:~# curl localhost:8081

Nginx on Docker Container
```

# Docker Flow


Index/Hub
5.
push
pull
1.
build
Dockerfile
Images
2.
run
commit
4.
Containers
3.
stop
start
restart
Local Docker instance
My computer

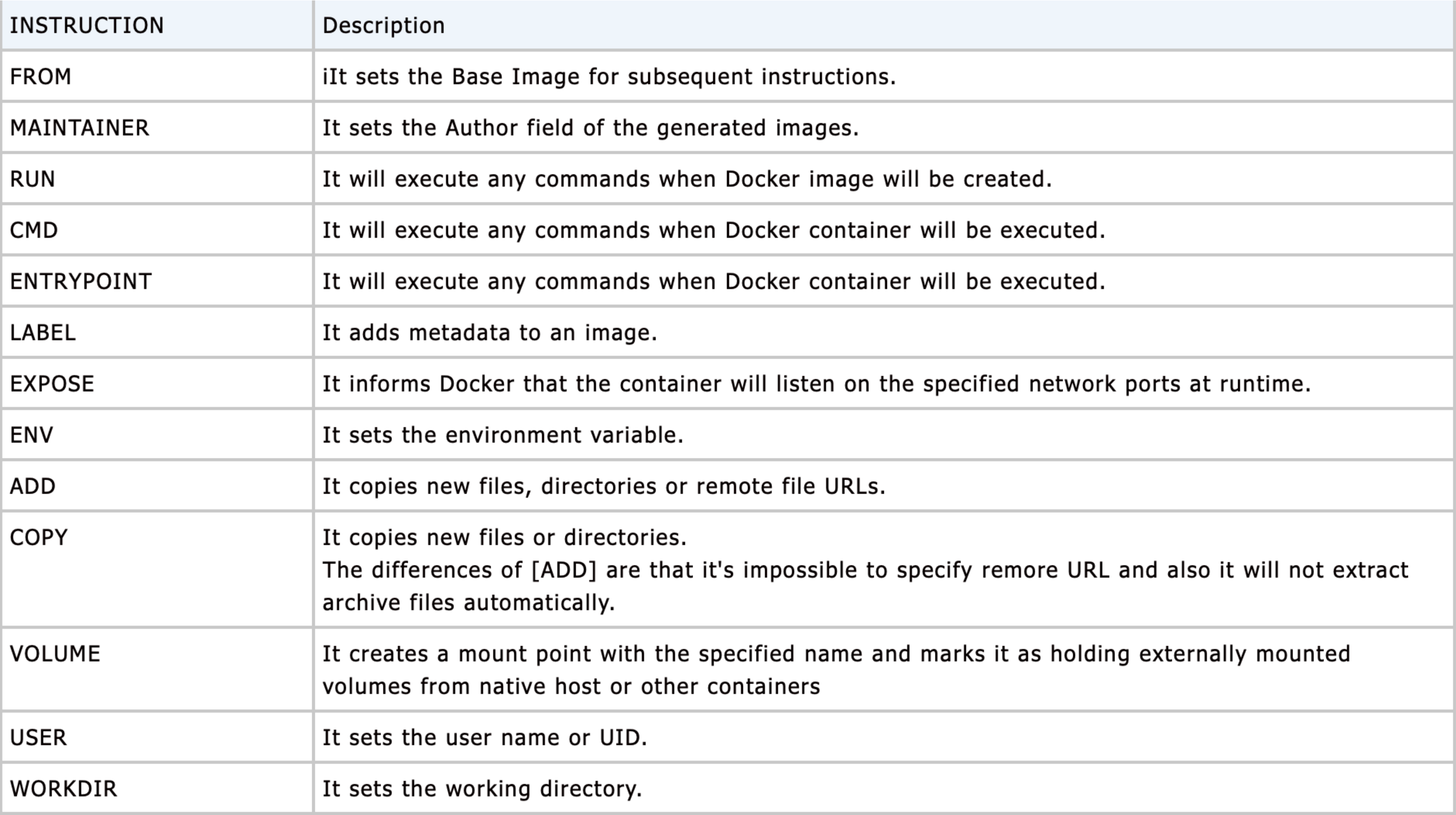
# Dockerfile

```
root@cn310:~# vi Dockerfile
# create new

FROM ubuntu
MAINTAINER CN31 <root@cn310.info>
RUN apt-get update
RUN apt-get -y install tzdata
RUN apt-get -y install apache2
RUN echo "Dockerfile Test on Apache2" > /var/www/html/index.html
EXPOSE 80
CMD ["/usr/sbin/apachectl", "-D", "FOREGROUND"]

# build image ⇒ docker build -t [image name]:[tag] .
root@cn310:~# docker build -t cn310/ubuntu-apache2:latest   ./
....
Successfully built 84bcc150feb9
Successfully tagged cn310/ubuntu-apache2:latest
```

# Dockerfile

```
root@cn310:~# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE
cn310/ubuntu-apache2 latest 84bcc150feb9 3 minutes ago 216MB
cn310/ubuntu-nginx latest 8f1fbe417eb2 14 minutes ago 160MB
ubuntu latest 7e0aa2d69a15 2 weeks ago 72.7MB

# run container

root@cn310:~# docker run -d -p 8081:80  cn310/ubuntu-apache2

91f835c52b0f2b6f71b7deb0591e0bef5cdd70c5fb0e817f69d905c9a0ae83cc
root@dlp:~# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 91f835c52b0f srv.world/ubuntu-apache2 "/usr/sbin/apachectl…" 8
seconds ago Up 7 seconds 0.0.0.0:8081->80/tcp ecstatic_pare
# verify accesses

root@dlp:~# curl localhost:8081
Dockerfile Test on Apache2
```

# Dockerfile



| INSTRUCTION | Description |
|---|---|
| FROM | iIt sets the Base Image for subsequent instructions. |
| MAINTAINER | It sets the Author field of the generated images. |
| RUN | It will execute any commands when Docker image will be created. |
| CMD | It will execute any commands when Docker container will be executed. |
| ENTRYPOINT | It will execute any commands when Docker container will be executed. |
| LABEL | It adds metadata to an image. |
| EXPOSE | It informs Docker that the container will listen on the specified network ports at runtime. |
| ENV | It sets the environment variable. |
| ADD | It copies new files, directories or remote file URLs. |
| COPY | It copies new files or directories.<br>The differences of [ADD] are that it's impossible to specify remore URL and also it will not extract archive files automatically. |
| VOLUME | It creates a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers |
| USER | It sets the user name or UID. |
| WORKDIR | It sets the working directory. |

# Docker Compose

To Install Docker Compose, it's easy to configure and run multiple containers as a Docker application.

```
root@cn310:~#apt -y install docker-compose
root@cn310:~# vi Dockerfile

FROM ubuntu
MAINTAINER CN310 <root@cn310.info>
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update
RUN apt-get -y install apache2
EXPOSE 80
CMD ["/usr/sbin/apachectl", "-D", "FOREGROUND"]
```

# Docker Compose

```
# define application configuration
root@cn310:~# vi docker-compose.yml
version: '3'
services:
  db:
    image: mariadb
    volumes:
        - /var/lib/docker/disk01:/var/lib/mysql
    environment:
       MYSQL_ROOT_PASSWORD: password
       MYSQL_USER: hirsute
       MYSQL_PASSWORD: password
       MYSQL_DATABASE: hirsute_db
    ports:
     - "3306:3306"
  web:
       build: .
       ports:
          - "8082:80"
       volumes:
          - /var/lib/docker/disk02:/var/www/html
```

# Docker Compose

```
root@cn310:~# docker-compose up -d
root@cn310:~# docker ps
root@cn310:~# apt install –y mariadb-client-core-10.3

root@cn310:~# mysql -h 127.0.0.1 -u root -p -e "show variables like 'hostname';"
root@cn310:~# mysql -h 127.0.0.1 -u hirsute -p -e "show databases;"
root@cn310:~# echo "Hello Docker Compose World"  >  /var/lib/docker/disk02/index.html
root@cn310:~#  docker-compose ps
root@cn310:~# curl 127.0.0.1:8082

Hello Docker Compose World

root@cn310:~# docker-compose exec web /bin/bash
root@cn310:~# docker-compose stop
```

# Assignment4

- Use docker-compose to install Wordpress
- Create docker-compose.yml  file that start your wordpress and sperate mysql
- Bring up wordpress in a web browser