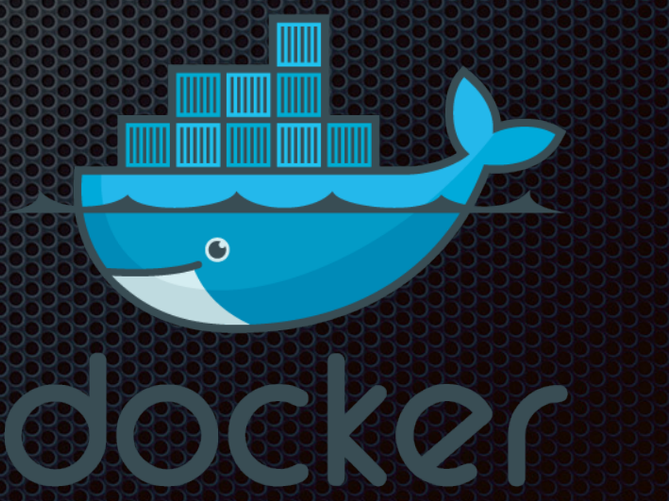
Workshop
Introduction to Docker

Krerk Piromsopa, Ph. D.
Department of Computer Engineering
Chulalongkorn University



Some slides from docker.io

Goal

- What/Why/How?
- Docker BASIC
- Docker Compose
- Docker Swam

What/Why Docker?

The Challenge

Multiplicity of Stacks

Static website

ginx 1.5 + modsecurity + openssl + bootstrap 2

User DB

postgresql + pgv8 + v8

Queue

Redis + redis-sentinel

Analytics DB

hadoop + hive + thrift + OpenJDK

Background workers

on 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs

Web frontend

Ruby + Rails + sass + Unicorn

API endpoint

Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client

Do services and apps interact appropriately?

Multiplicity of hardware environments

Development VM

QA server

Customer Data Center

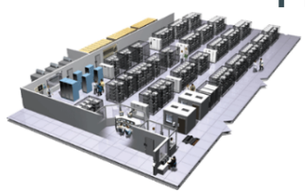
docker

Public Cloud

Disaster recovery

Production Servers

Production Cluster



Contributor's laptop

Can I migrate smoothly and quickly?

The Matrix From Hell

	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	?	?	?	?	?	?	?

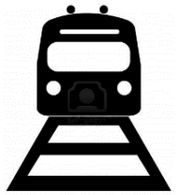
docker

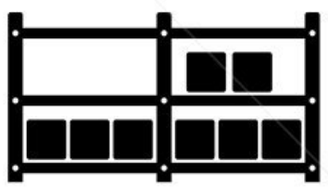
Cargo Transport Pre-1960

Multiplicity of Goods

Do I worry about how goods interact (e.g. coffee beans next to spices)

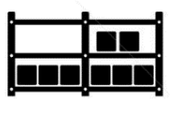
Multipilicity of methods for transporting/storing





Can I transport quickly and smoothly (e.g. from boat to train to truck)

Also a matrix from hell

	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
							

Solution: Intermodal Shipping Container

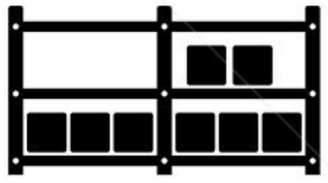
Multiplicity of Goods

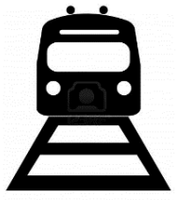
Do I worry about how goods interact (e.g. coffee beans next to spices)

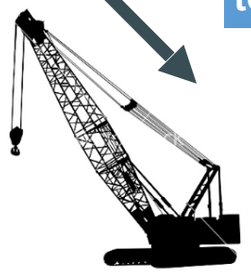
A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.

…in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another

Multiplicity of methods for transporting/ storing







Can I transport quickly and smoothly (e.g. from boat to train to truck)

Docker is a shipping container system for code

Multiplicity of Stacks

Static website | User DB | Web frontend | Queue | Analytics DB

Do services and apps interact appropriately?

An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…

…that can be manipulated using standard operations and run consistently on virtually any hardware platform

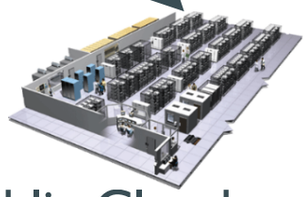
Multiplicity of hardware environments

Can I migrate smoothly and quickly

Developme nt VM

QA server

Customer Data Center

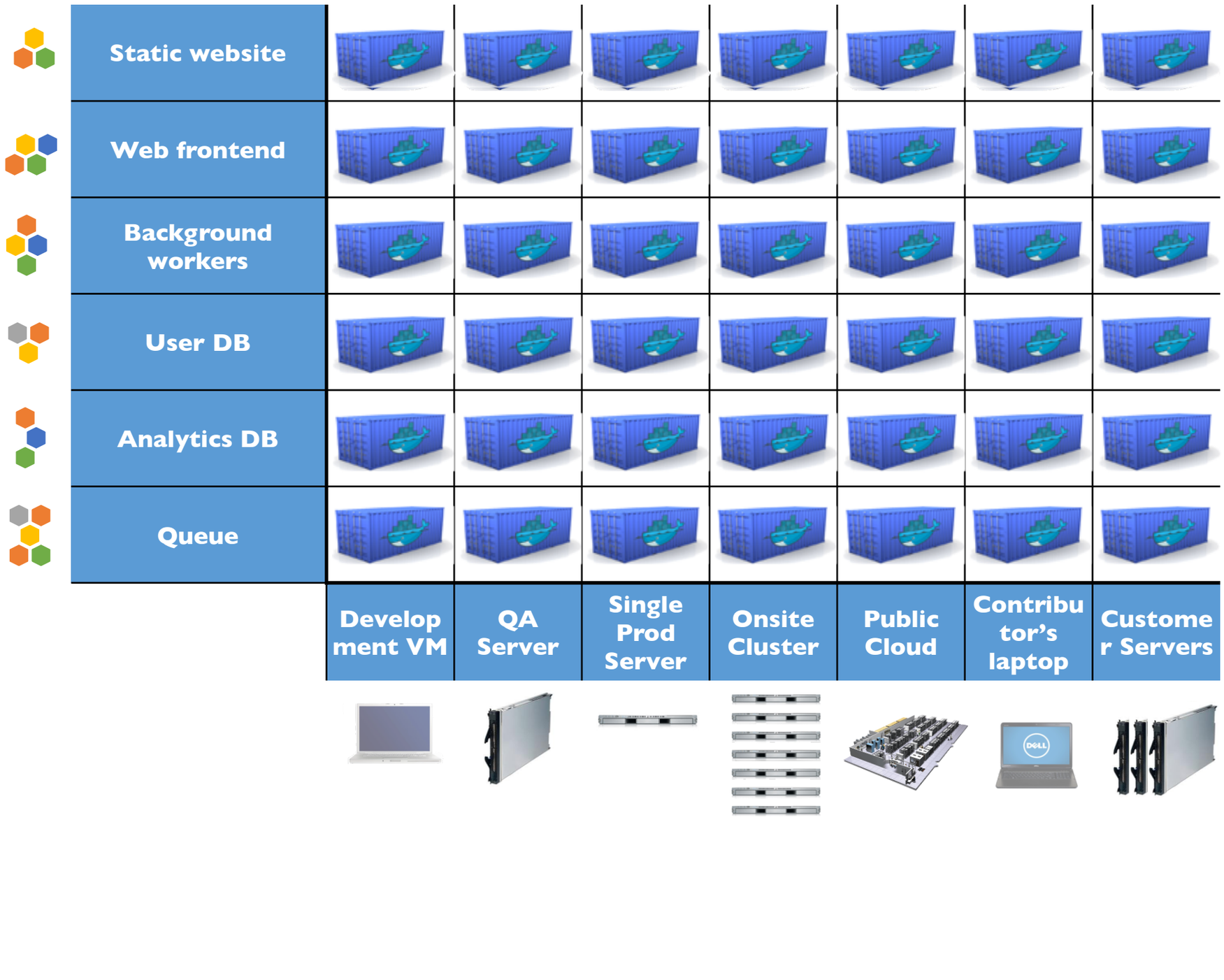


Public Cloud

Production Cluster

Contributor's laptop

Docker eliminates the matrix from Hell



Why Developers Care

- Build once…(finally) run anywhere*
  - A clean, safe, hygienic and portable runtime environment for your app.
  - No worries about missing dependencies, packages and other pain points during subsequent deployments.
  - Run each app in its own isolated container, so you can run various versions of libraries and other dependencies for each app without worrying
  - Automate testing, integration, packaging…anything you can script
  - Reduce/eliminate concerns about compatibility on different platforms, either your own or your customers.
  - Cheap, zero-penalty containers to deploy services? A VM without the overhead of a VM? Instant replay and reset of image snapshots? That's the power of Docker

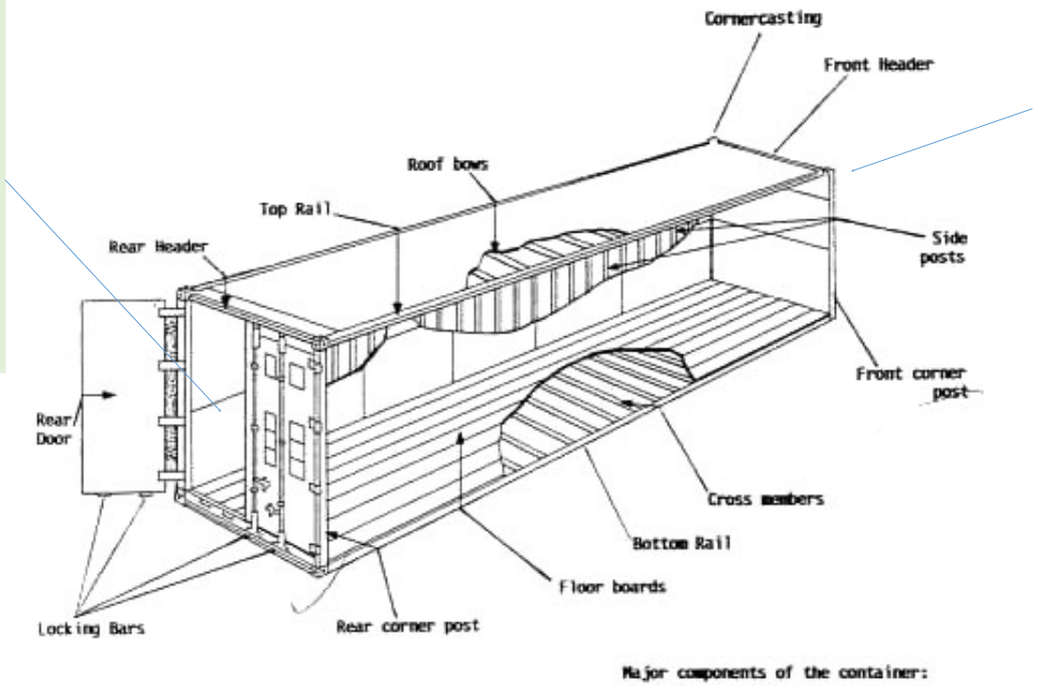
* With the 0.7 release, we support any x86 server running a modern Linux kernel (3.2+ generally. 2.6.32+ for RHEL 6.5+, Fedora, & related)

Why Devops Cares?

- Configure once…run anything
  - Make the entire lifecycle more efficient, consistent, and repeatable
  - Increase the quality of code produced by developers.
  - Eliminate inconsistencies between development, test, production, and customer environments
  - Support segregation of duties
  - Significantly improves the speed and reliability of continuous deployment and continuous integration systems
  - Because the containers are so lightweight, address significant performance, costs, deployment, and portability issues normally associated with VMs

Why it works—separation of concerns

- Dan the Developer
  - Worries about what's "inside" the container
    - His code
    - His Libraries
    - His Package Manager
    - His Apps
    - His Data
  - All Linux servers look the same



Major components of the container:

- Oscar the Ops Guy
  - Worries about what's "outside" the container
    - Logging
    - Remote access
    - Monitoring
    - Network config
  - All containers start, stop, copy, attach, migrate, etc. the same way

docker

More technical explanation

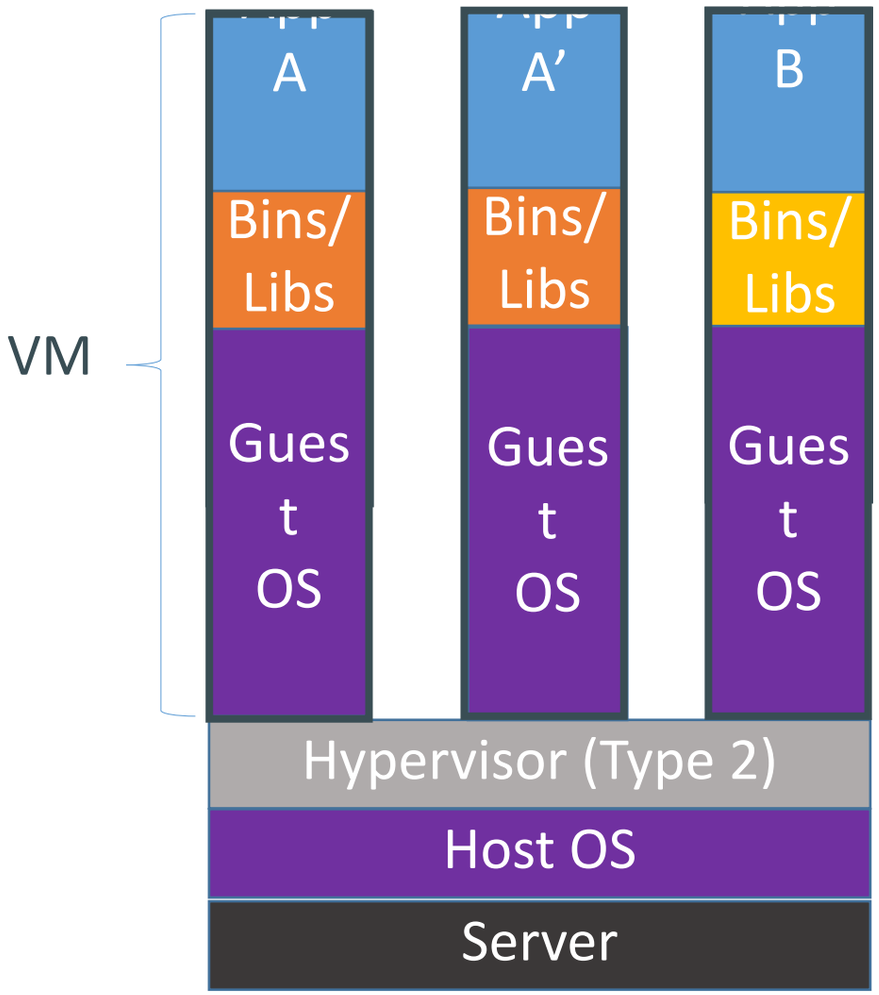
WHY

- Run everywhere
  - Regardless of kernel version (2.6.32+)
  - Regardless of host distro
  - Physical or virtual, cloud or not
  - Container and host architecture must match*
- Run anything
  - If it can run on the host, it can run in the container
  - i.e. if it can run on a Linux kernel, it can run

WHAT

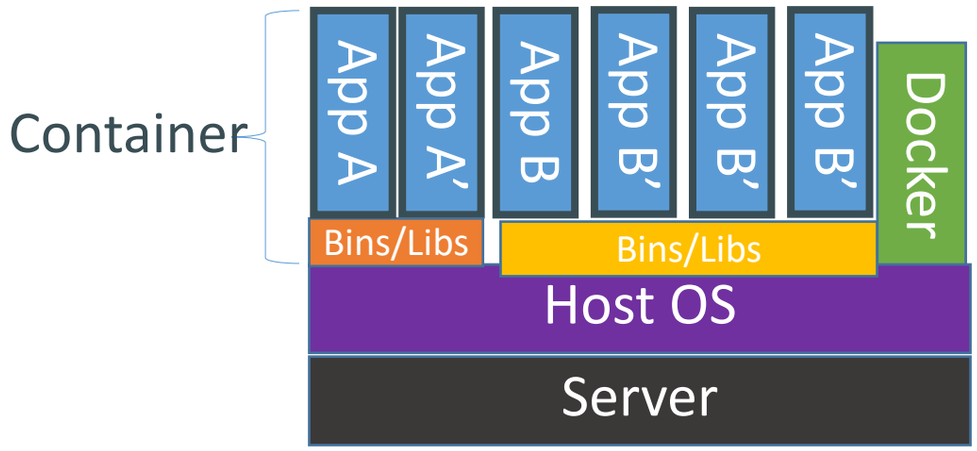
- High Level—It's a lightweight VM
  - Own process space
  - Own network interface
  - Can run stuff as root
  - Can have its own /sbin/init (different from host)
  - <<machine container>>
- Low Level—It's chroot on steroids
  - Can also *not* have its own /sbin/init
  - Container=isolated processes
  - Share kernel with host
  - No device emulation (neither HVM nor PV) from host)
  - <<application container>>

Containers vs. VMs



Containers are isolated, but share OS and, where appropriate, bins/libraries

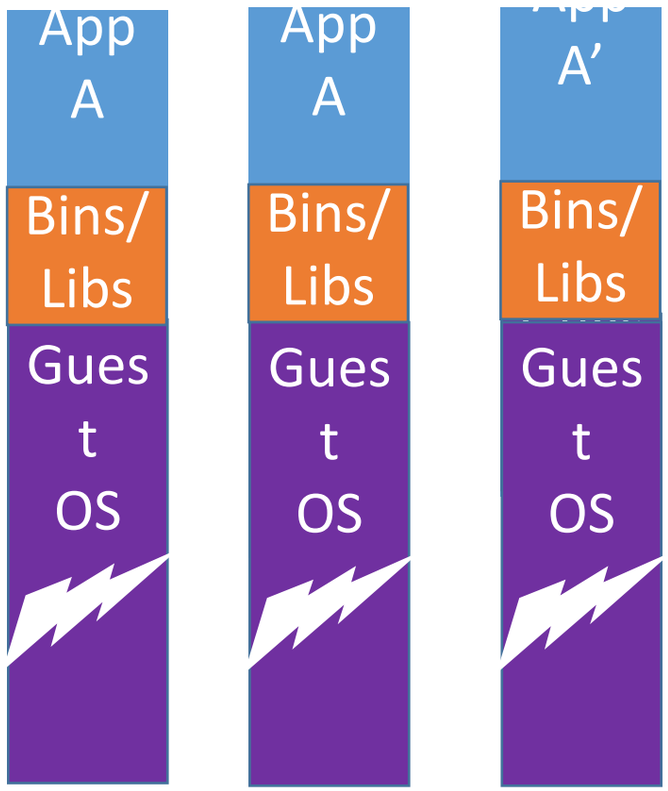
…result is significantly faster deployment, much less overhead, easier migration, faster restart



docker

Why are Docker containers lightweight?

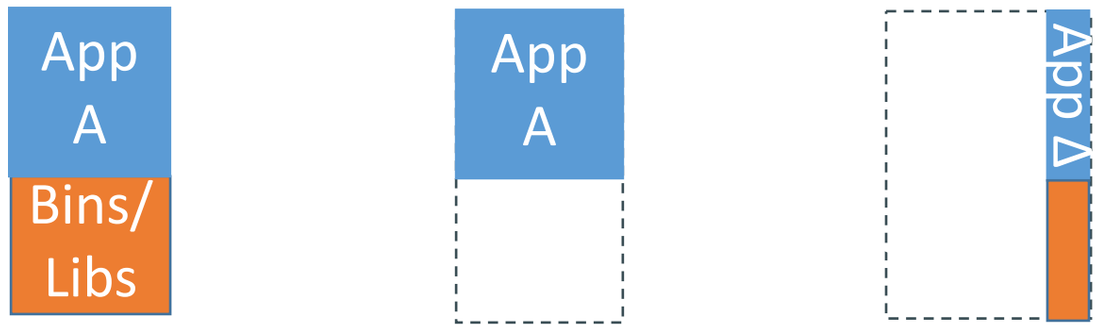
VMs



VMs

Every app, every copy of an app, and every slight modification of the app requires a new virtual server

Containers



Original App

(No OS to take up space, resources, or require restart)

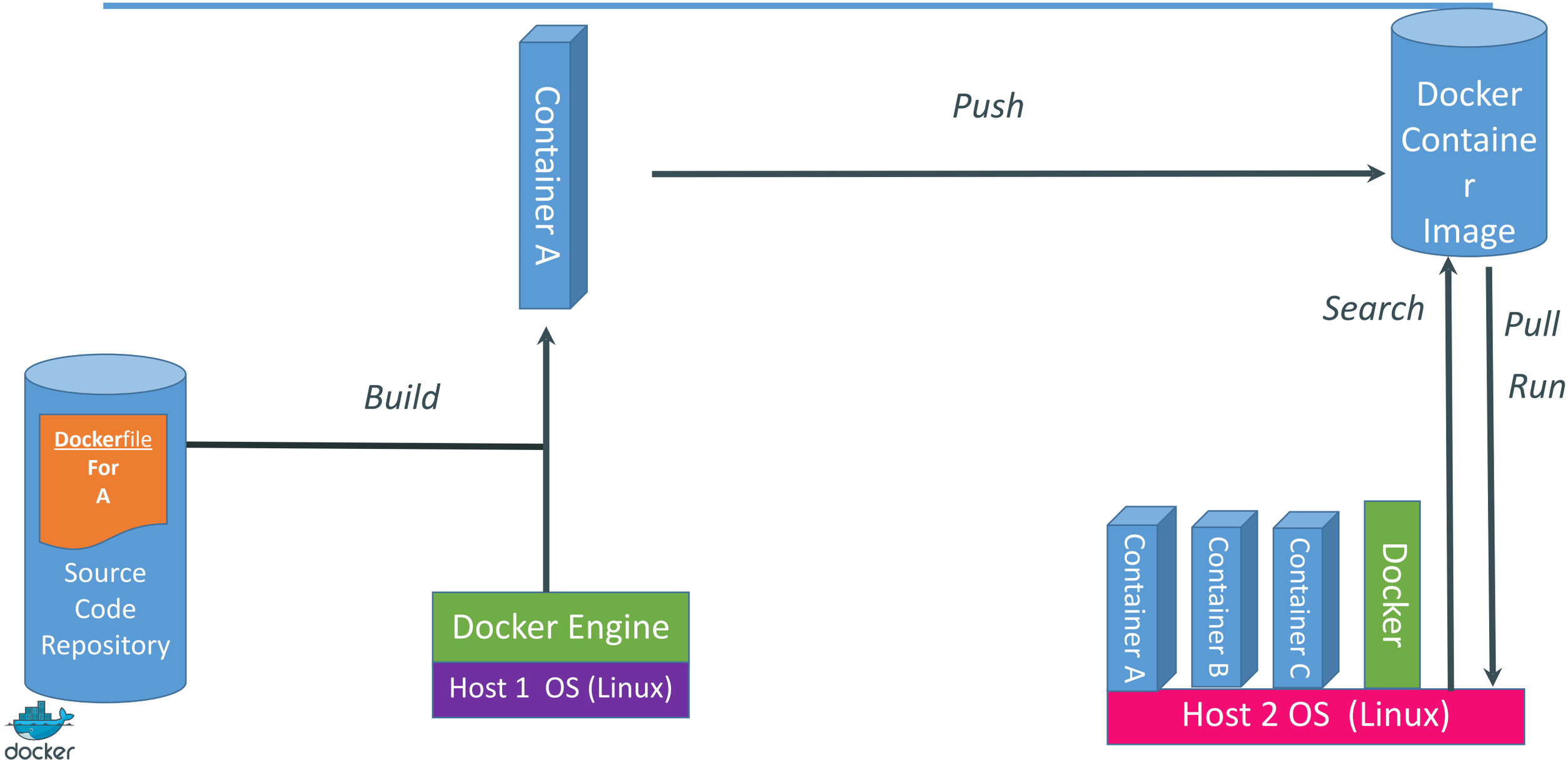
Copy of App

No OS. Can Share bins/libs

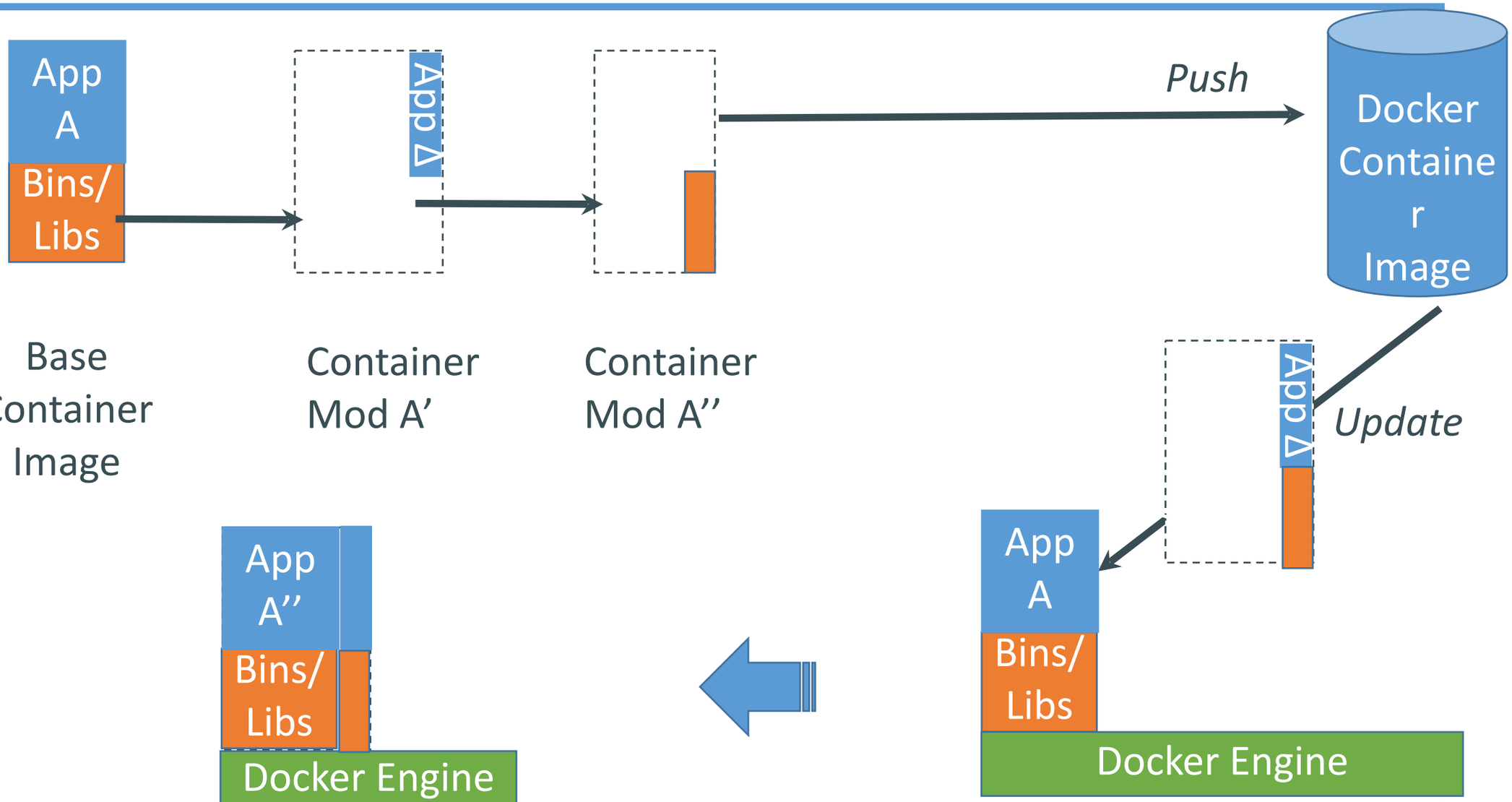
Modified App

Copy on write capabilities allow us to only save the diffs Between container A and container A'

What are the basics of the Docker system?

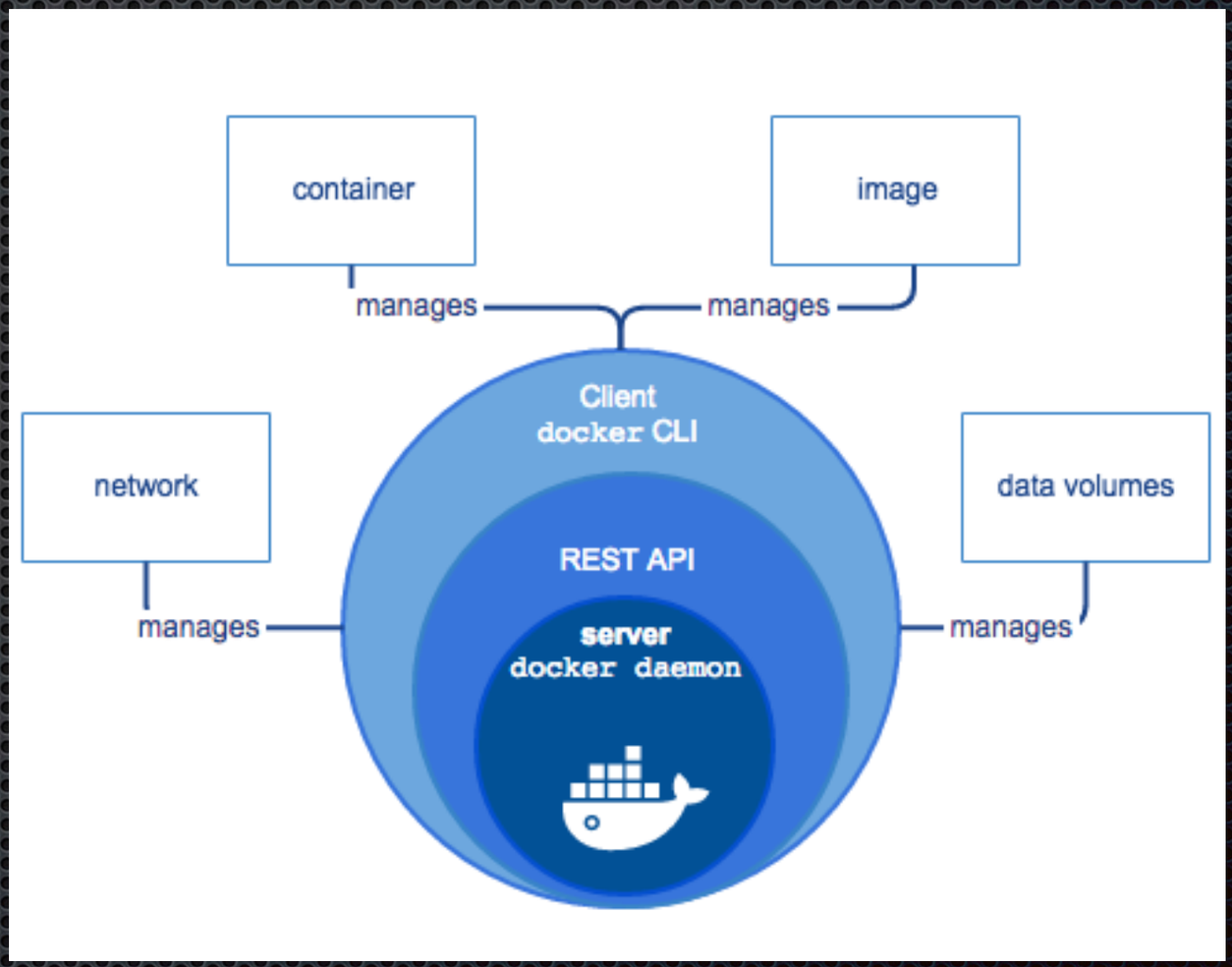


Changes and Updates



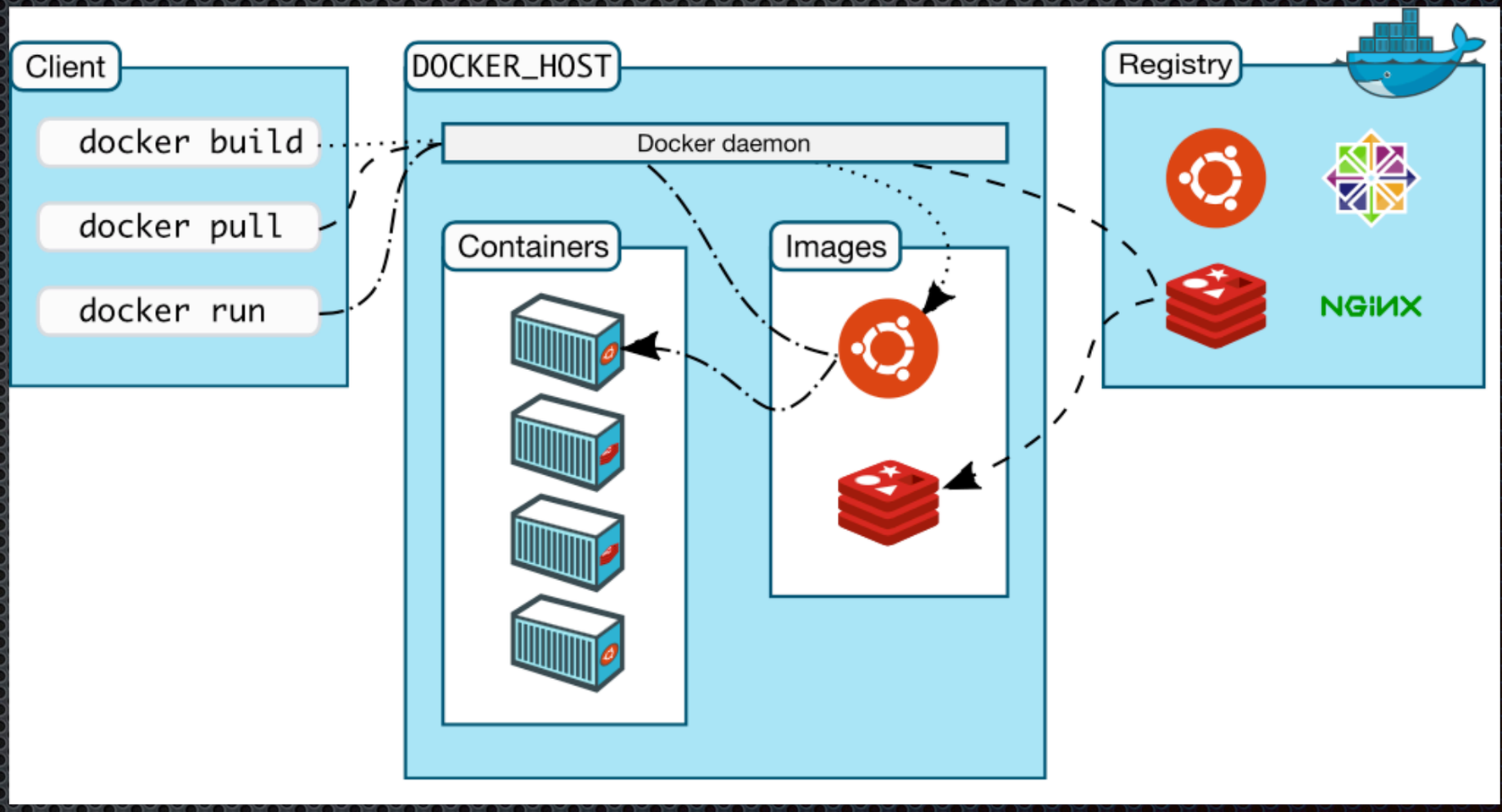
Docker Basic

- Docker abstracts:
  - Storage
  - Network
  - Management (start/stop)



Docker Architecture

- daemon
- client
- Inside
  - images
  - containers
  - registry
  - services



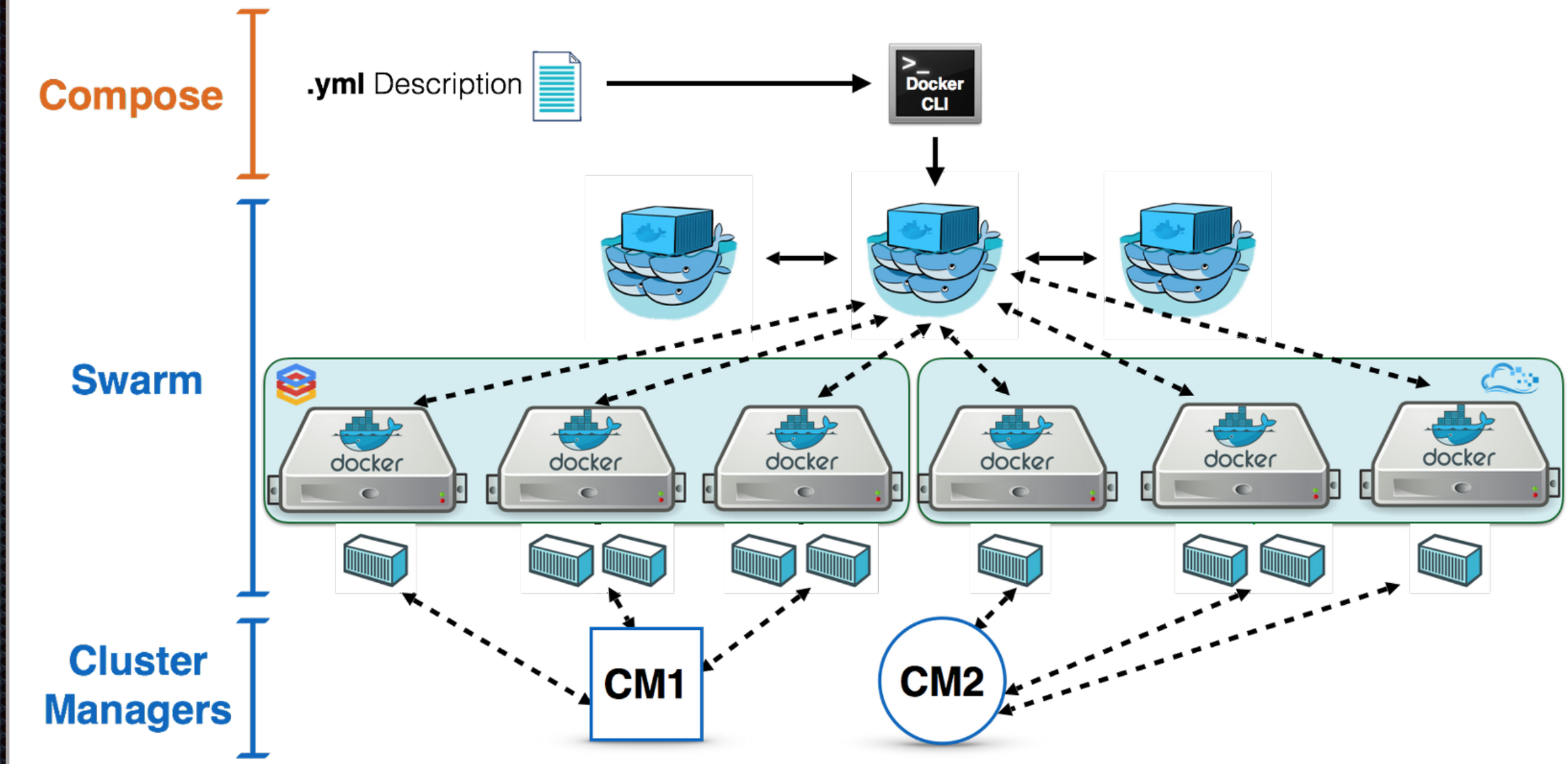
Docker Compose

- Compose is a tool for defining and running multi-container Docker applications.
- Use a compose file to configure services.
- Deploy (create and start) with a single command

Swarm Mode

- Cluster management or orchestration features for Docker Engine.
- Managers and Workers
- Services
  - Definitions of tasks to execute on managers and workers
- Ingres load balancing (we will revisit this later)
- PublishedPort

Anatomy of Swarm



Give it a try.

Prerequisite

- docker client, docker engine (or docker toolbox)
- pull
  phpmyadmin/phpmyadmin
  mariadb:10.1
  php:7.3-alpine

Scenario 1: command line

- Run a PHP script using docker engine.
- docker pull php:7.3-alpine
- docker run \
  -v $PWD:/usr/src/myapp \
  -w /usr/src/myapp \
  php:7.3-alpine \
  php HelloWorld.php

HelloWorld.php

```
<?php
echo "PHP ".phpversion()."\n";
echo "Hello, World\n”;
?>
```

Scenario 2 : service

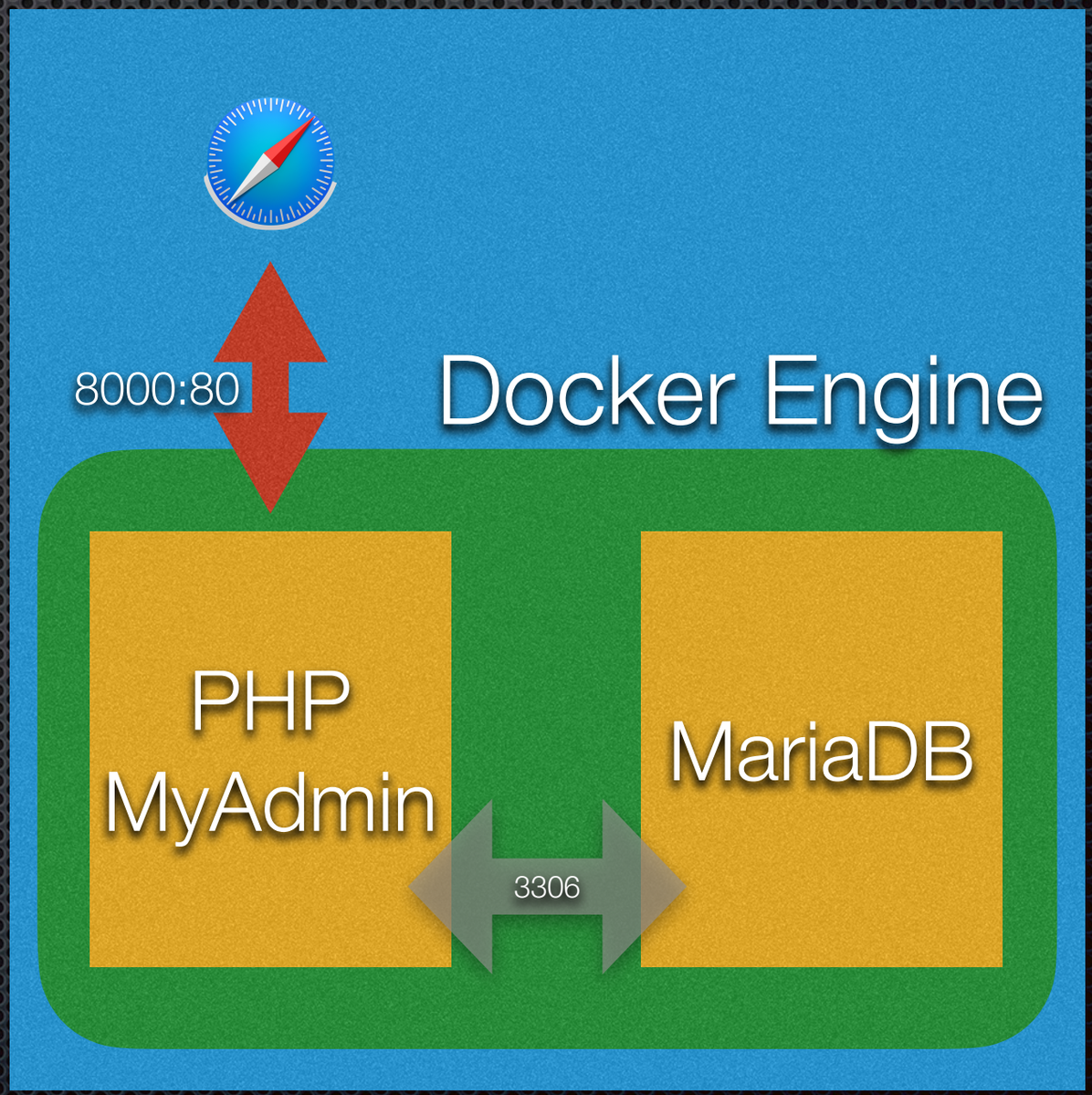
MariaDB (mariadb:10.1)

```
docker run --name ex-app-db \
-e MYSQL_ROOT_PASSWORD=exroot \
-d mariadb:10.1
```

phpmyadmin/phpmyadmin

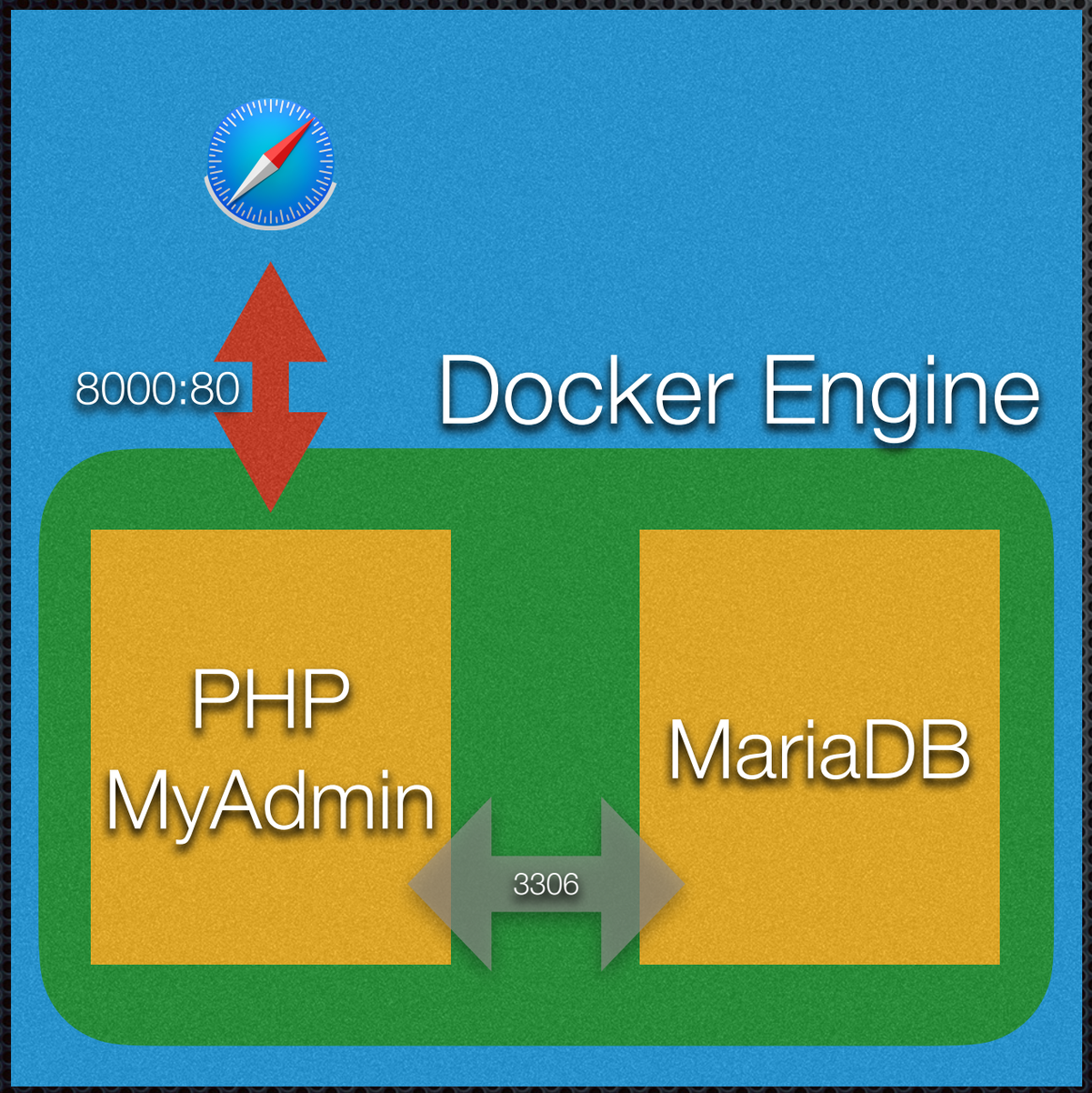
```
docker run --name ex-app-myadmin \
 --link ex-app-db:db \
-d -p 8000:80 \
phpmyadmin/phpmyadmin
```

browse to http://localhost:8000



Use docker compose

Host (Mac OSX)



docker-compose.yaml

```
version: "3.0"
services:
   db:
     image: mariadb:10.1
     container_name: ex-app-db
     volumes:
       - "C:\Users\Test\db":/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: exroot
phpmyadmin:
   links:
     - db
   image: phpmyadmin/phpmyadmin
   container_name: ex-app-myadmin
   restart: always
   ports:
    - 8000:80
```

docker-compose up -d

Scenario 3 : Docker Swarm

index.php

```
Hello from ..“
<?php
echo  $_SERVER['SERVER_ADDR'];
?>
”..
```

- Work as a group (2-4 persons)
- Make a swarm of dockers.
- Deploy 4 instances of php application.
  (Let’s ignore the database for now)

TODO

Dockerfile

```
FROM php:apache
MATAINER Krerk Piromsopa, Ph.D. <krerk.p@chula.ac.th>

COPY index.php /var/www/html
```

- First try to create the simple php app as a standalone service with Dockerfile and docker-compose.
  - docker-compose build
  - docker-compose up
- Try connecting to it from your localhost.

docker-compose.yaml

```
version: "3.0"
services:
    myapp:
        build: .
        image: 127.0.0.1:5000/appdemo
        restart: always
        ports:
                - 60080:80
```

Steps

- To start a manager
  - docker swarm init
- To join a master (run on workers)
  - docker swarm join —token [token from master] [host:port]
- See your cluster
  - docker node ls

Publish image to registry

- Create a local registry to publish to..
  (we will remove later)
  - docker service create \
    —name registry \
    —publish published=5000,target=5000 \
    registry:2
  - #test with
    curl http://localhost:5000/v2/
- Push
  - docker-compose push

Deploy a service/stack

- Deploy with
  - docker stack deploy —compose-file docker-compose.yaml \ mydemo
- Use docker service command to see your service
  - docker stack services mydemo
- Scale with
  - docker service scale <SERVICE-ID>=<NUMBER-OF-TASKS>

More

- Delete stack with
  - docker stack rm mydemo
- Delete service with
  - docker service rm [servicename]
  - eg. docker service rm registry
- Drain a node (to remove)
  - docker node update --availability drain [worker]
- Leave the swarm
  - docker swarm leave

Scenario 4: DB App (Optional)

- Deploy
  - 1 instance of mariadb
  - 4 instances of phpmyadmin on docker swarm

Scenario 5: Docker Build

- Rather than mounting volume and to installation on the fly, let's build / pack our container.
- We will build a working php app (adding index.php in to an image)
- Use "docker build -t test/myapp ." to build an image

"index.php"

```
Hello from ..“
<?php
echo  $_SERVER['SERVER_ADDR'];
?>
”..
```

Dockerfile

```
FROM php:7.3-apache-buster

Label MAINTAINER=“Krerk Piromsopa”

COPY index.php /var/www/html

RUN chmod 777 /var/www/html/index.php
```

Notes

- docker vs. docker-compose
  - docker to manage single container
  - docker-compose to manage multiple containers
- docker service vs. docker stack
  - docker service to manage a single service
  - docker stack to manage group of services

Notes

- It is also possible to specify the number of replicas in the compose file.
- Now, we can also use kubernetes to manage docker swarm. (not in this workshop).
  - This will use kubernetes deployment yaml.

References

- https://docs.docker.com/engine/swarm/stack-deploy/
- https://docs.docker.com/compose/compose-file/