Introduction to Docker

Roparat Sukapirom

Skooldio

/me

- Roparat Sukapirom (Pap)
- CP30
- Software Engineer at Skooldio
- Former Technical Specialist (3rd Level Support) at Thomson Reuters (Thailand) Ltd.

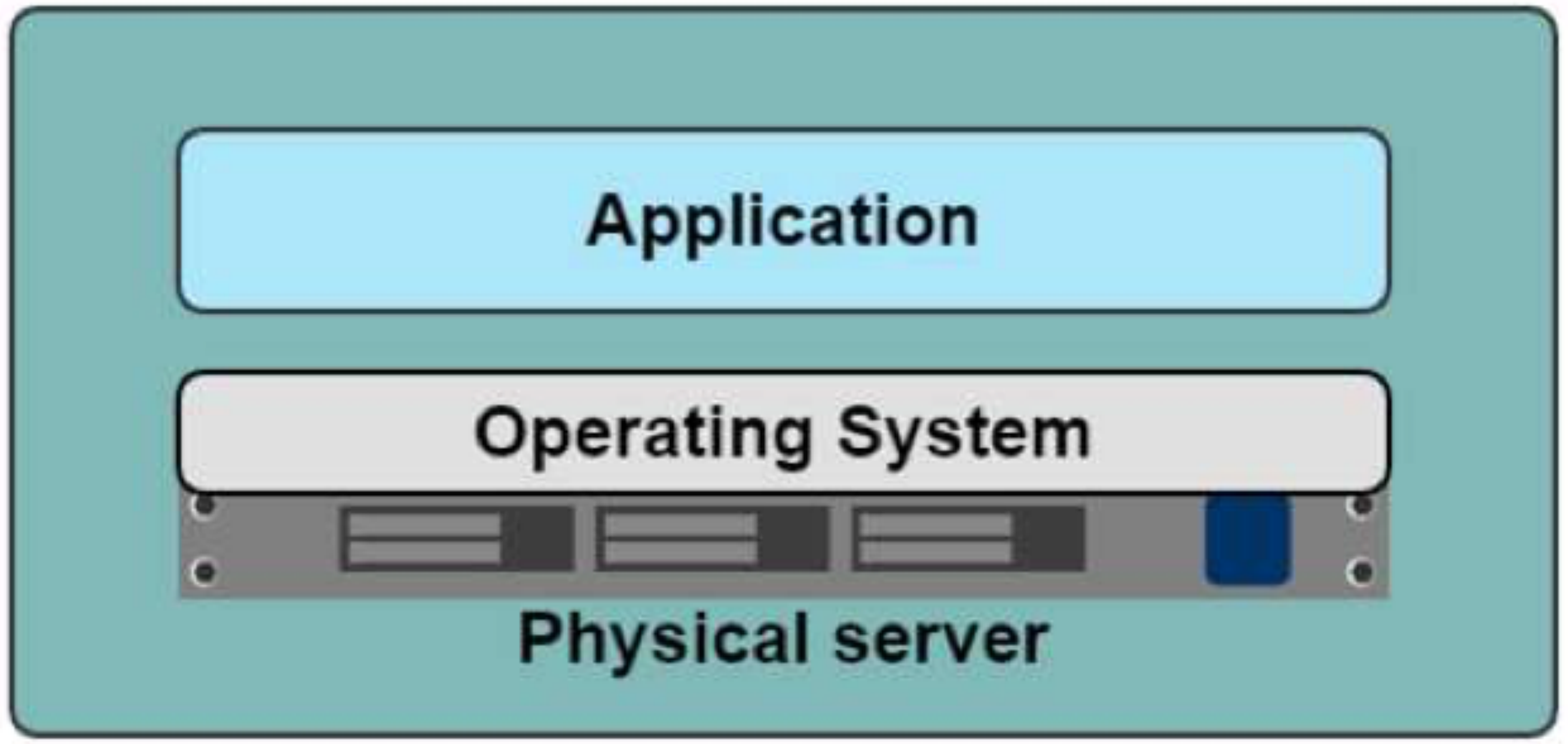
Why Docker Existed?

Ancient History

A History Lesson

In the Dark Ages

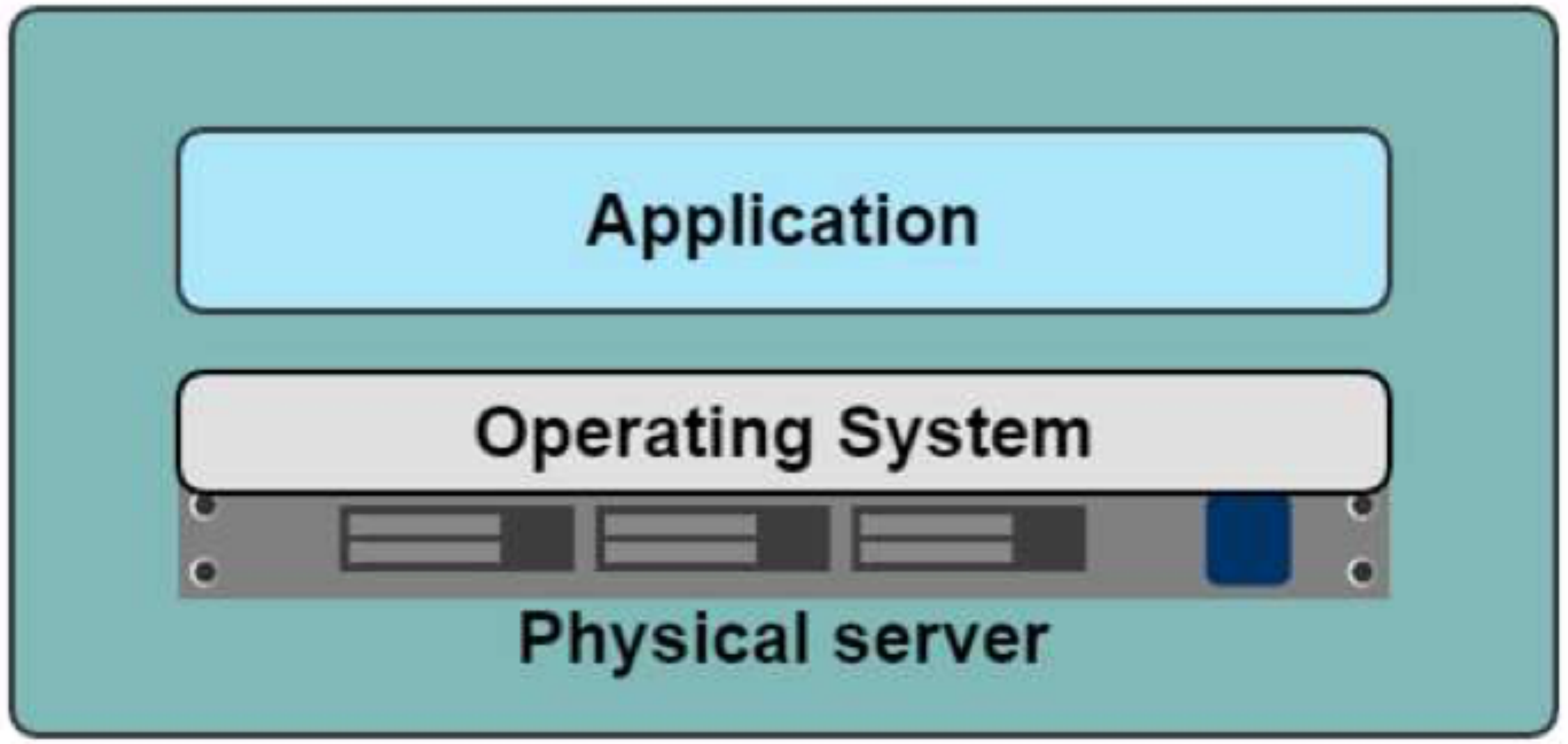
One application on one physical server



https://www.slideshare.net/mobile/Docker/introduction-to-docker-2017

Historical limitations of application deployment

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

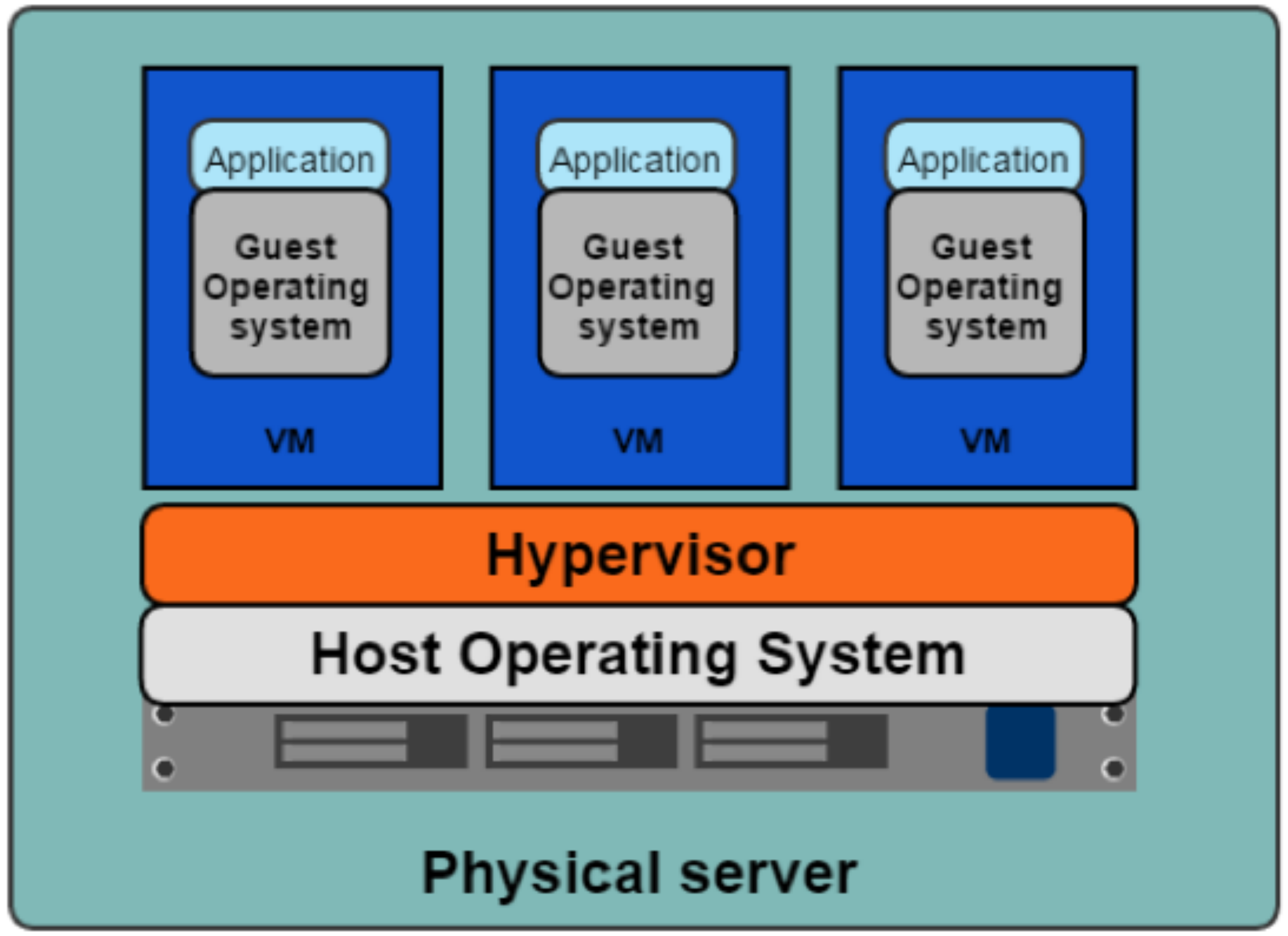


https://www.slideshare.net/mobile/Docker/introduction-to-docker-2017

A History Lesson

Hypervisor-based Virtualization

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



https://www.slideshare.net/mobile/Docker/introduction-to-docker-2017

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

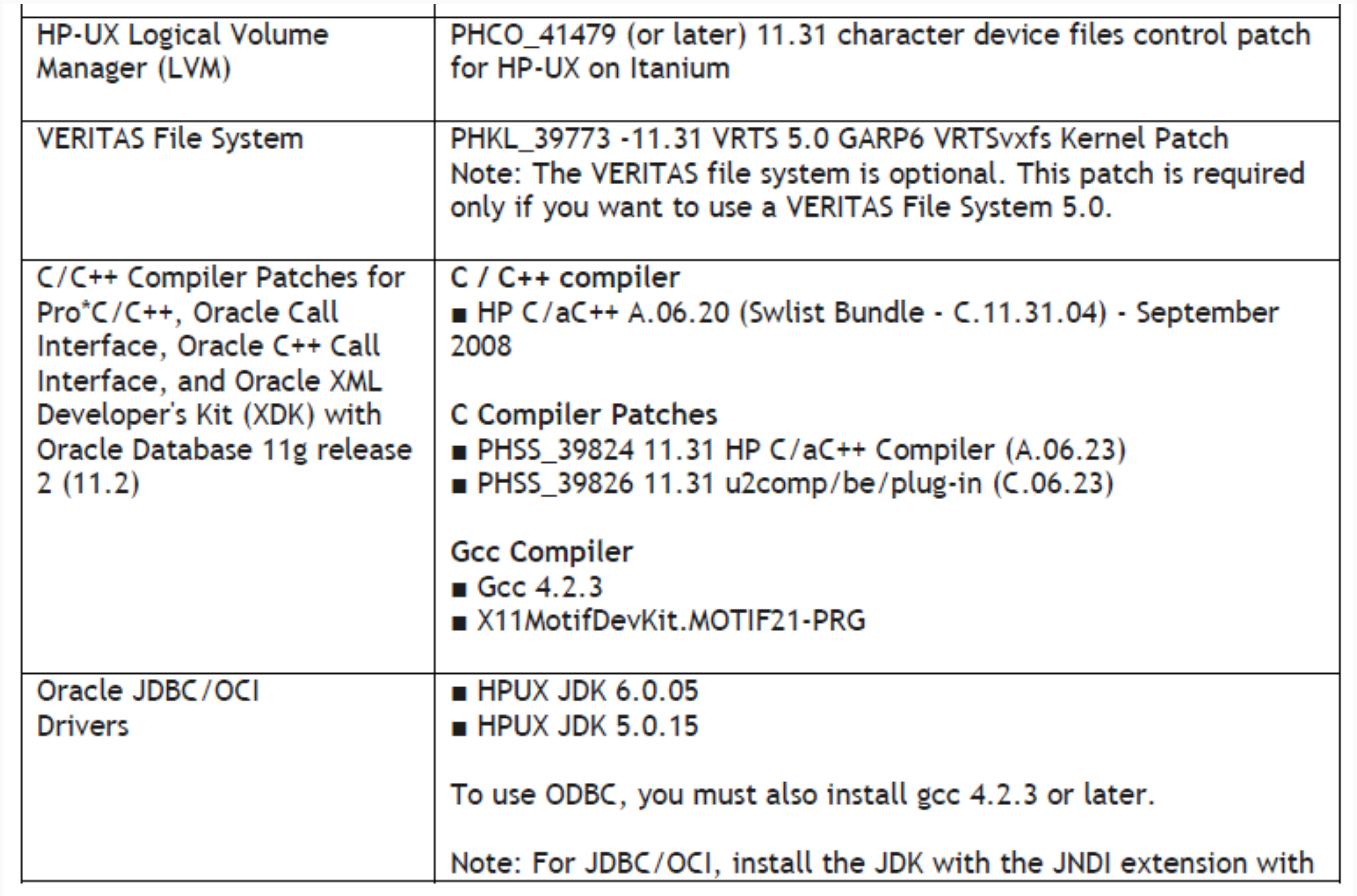
Microsoft Azure

amazon web services™

vmware®

Limitation of VM

- Each VM requires dedicate resource
  - CPU
  - RAM
  - Storage
  - Guest OS Installation (takes many GBs, overhead CPU and RAM)
- Application dependent on OS environment
- Service Oriented Architecture/Micro-services
  - VM has far more overhead
  - VM Boot Up Time (when scaling)

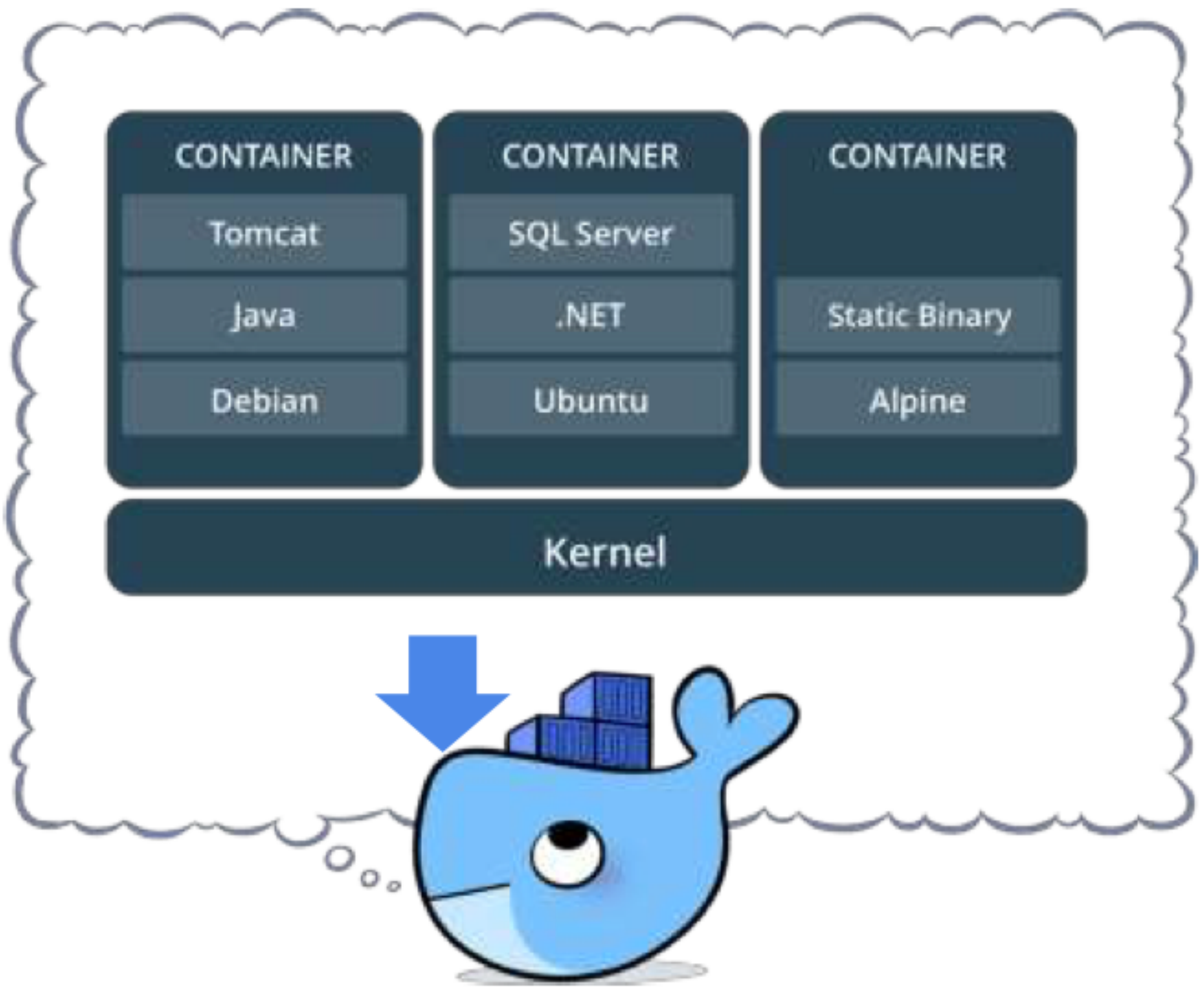


HP-UX Logical Volume Manager (LVM)	PHCO_41479 (or later) 11.31 character device files control patch for HP-UX on Itanium
VERITAS File System	PHKL_39773 -11.31 VRTS 5.0 GARP6 VRTSvxfs Kernel Patch Note: The VERITAS file system is optional. This patch is required only if you want to use a VERITAS File System 5.0.
C/C++ Compiler Patches for Pro*C/C++, Oracle Call Interface, Oracle C++ Call Interface, and Oracle XML Developer's Kit (XDK) with Oracle Database 11g release 2 (11.2)	C / C++ compiler ■ HP C/aC++ A.06.20 (Swlist Bundle - C.11.31.04) - September 2008 C Compiler Patches ■ PHSS_39824 11.31 HP C/aC++ Compiler (A.06.23) ■ PHSS_39826 11.31 u2comp/be/plug-in (C.06.23) Gcc Compiler ■ Gcc 4.2.3 ■ X11MotifDevKit.MOTIF21-PRG
Oracle JDBC/OCI Drivers	■ HPUX JDK 6.0.05 ■ HPUX JDK 5.0.15 To use ODBC, you must also install gcc 4.2.3 or later. Note: For JDBC/OCI, install the JDK with the JNDI extension with

http://bishwanathdas.blogspot.com/2015/07/oracle-11g-release-2-112040-database.html

What would happen if we virtualize environment?

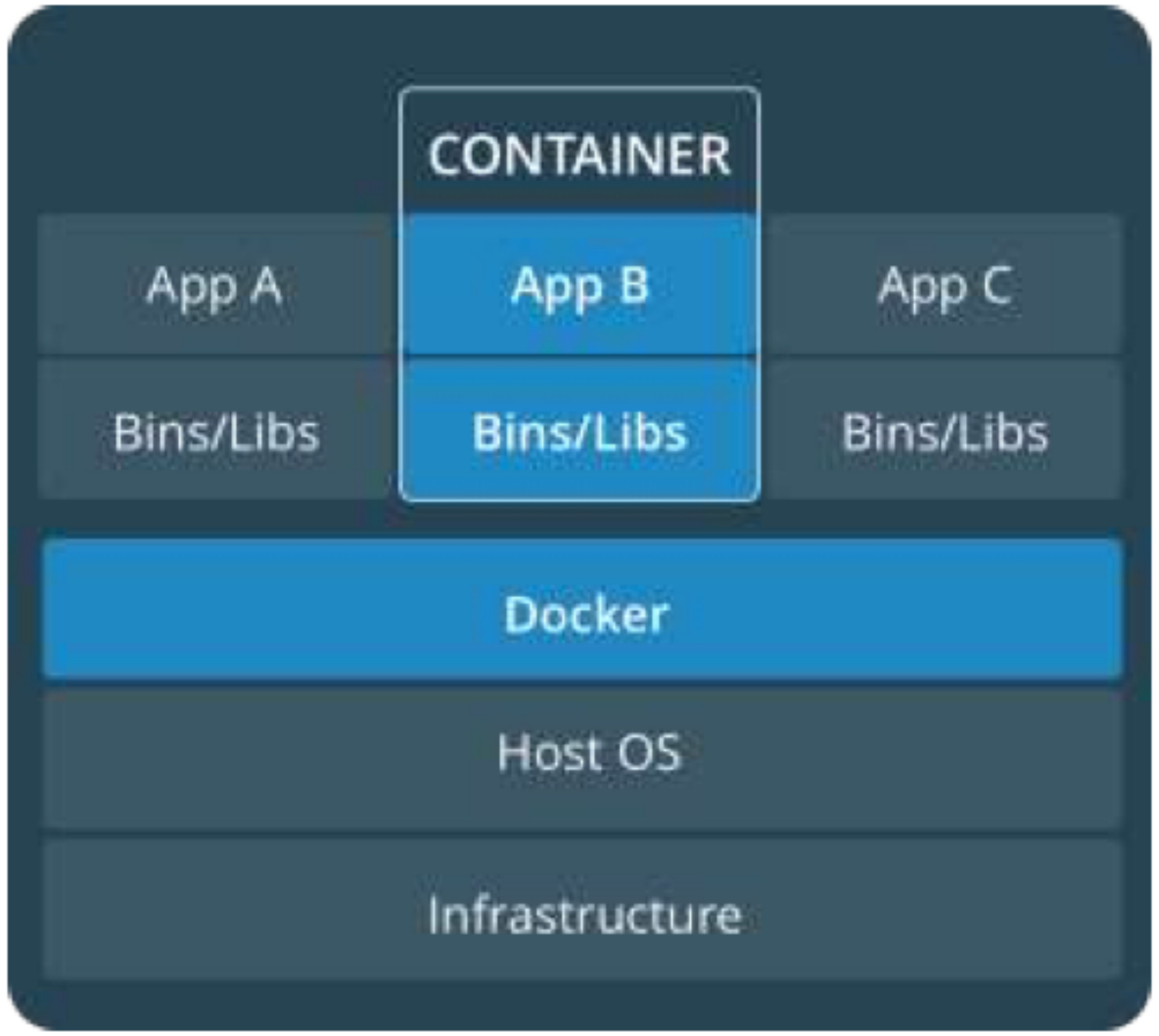
What is a container?



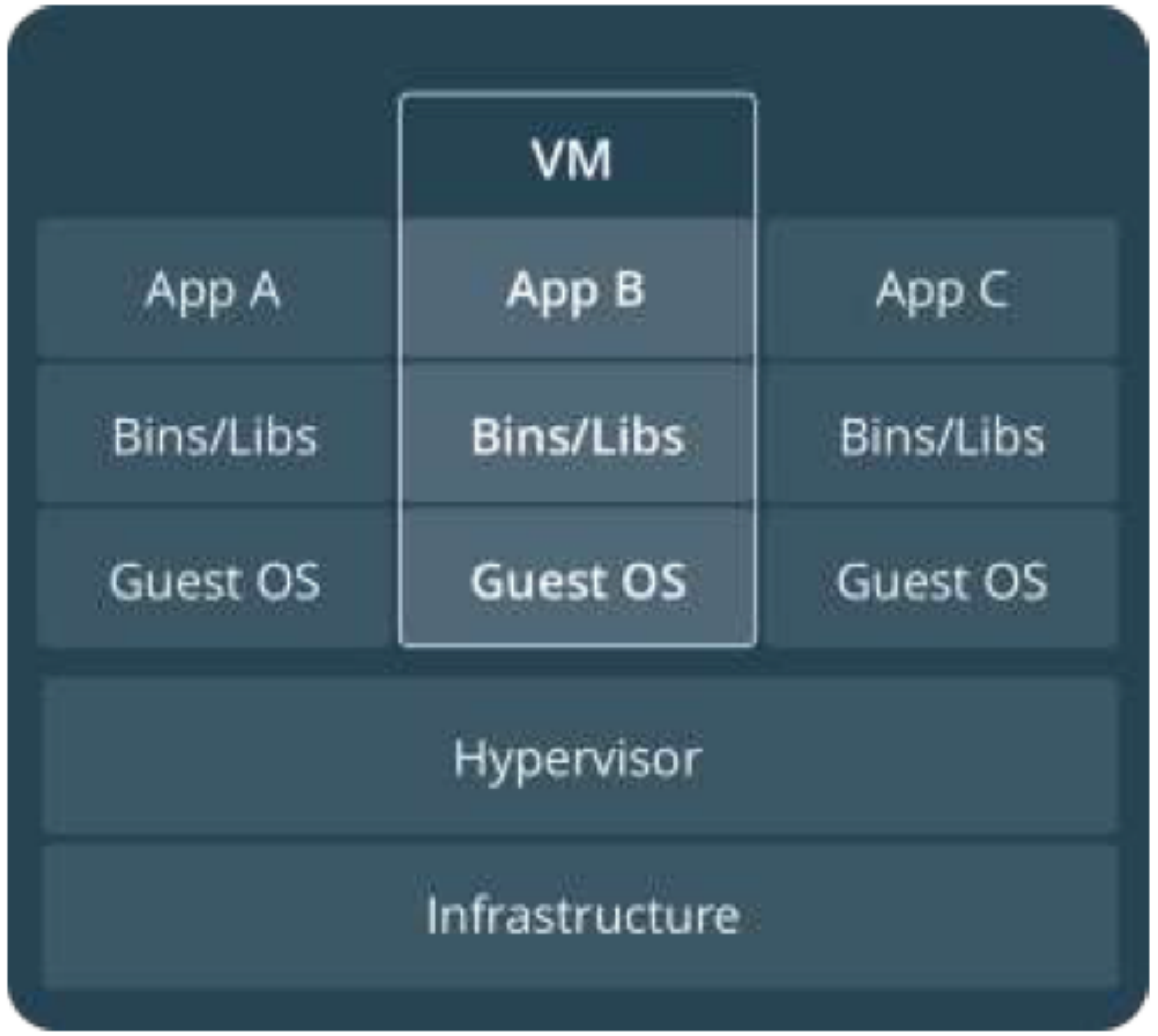
- Standardized packaging for software and dependencies
- Isolate apps from each other
- Share the same OS kernel
- Works with all major Linux and Windows Server

https://www.slideshare.net/mobile/Docker/introduction-to-docker-2017

Comparing Containers and VMs



Containers are an app level construct



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

https://www.slideshare.net/mobile/Docker/introduction-to-docker-2017

What is Docker?

Docker

- Docker is a Popular Container Engine
- Docker is a platform
    - For developers and sysadmins
    - Develop, deploy, and run applications with containers.

Key Benefits of Docker Containers

Speed

- No OS to boot = applications online in seconds

Portability

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

Efficiency

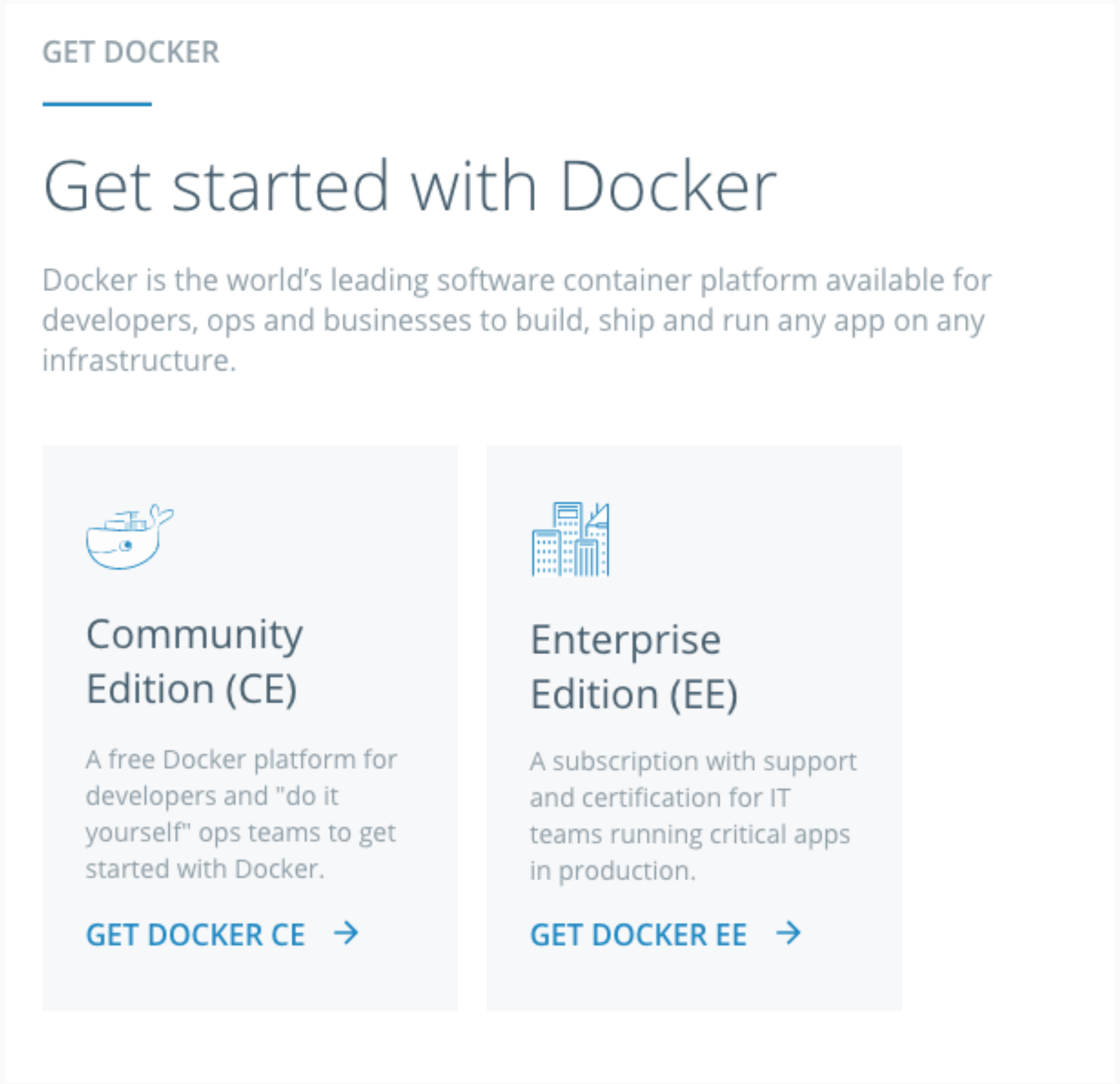
- Less OS overhead
- Improved VM density

How to use Docker?

Installation

https://store.docker.com

Search for Docker Community Edition (CE)



GET DOCKER

Get started with Docker

Docker is the world's leading software container platform available for developers, ops and businesses to build, ship and run any app on any infrastructure.

Community Edition (CE)

A free Docker platform for developers and "do it yourself" ops teams to get started with Docker.

GET DOCKER CE →

Enterprise Edition (EE)

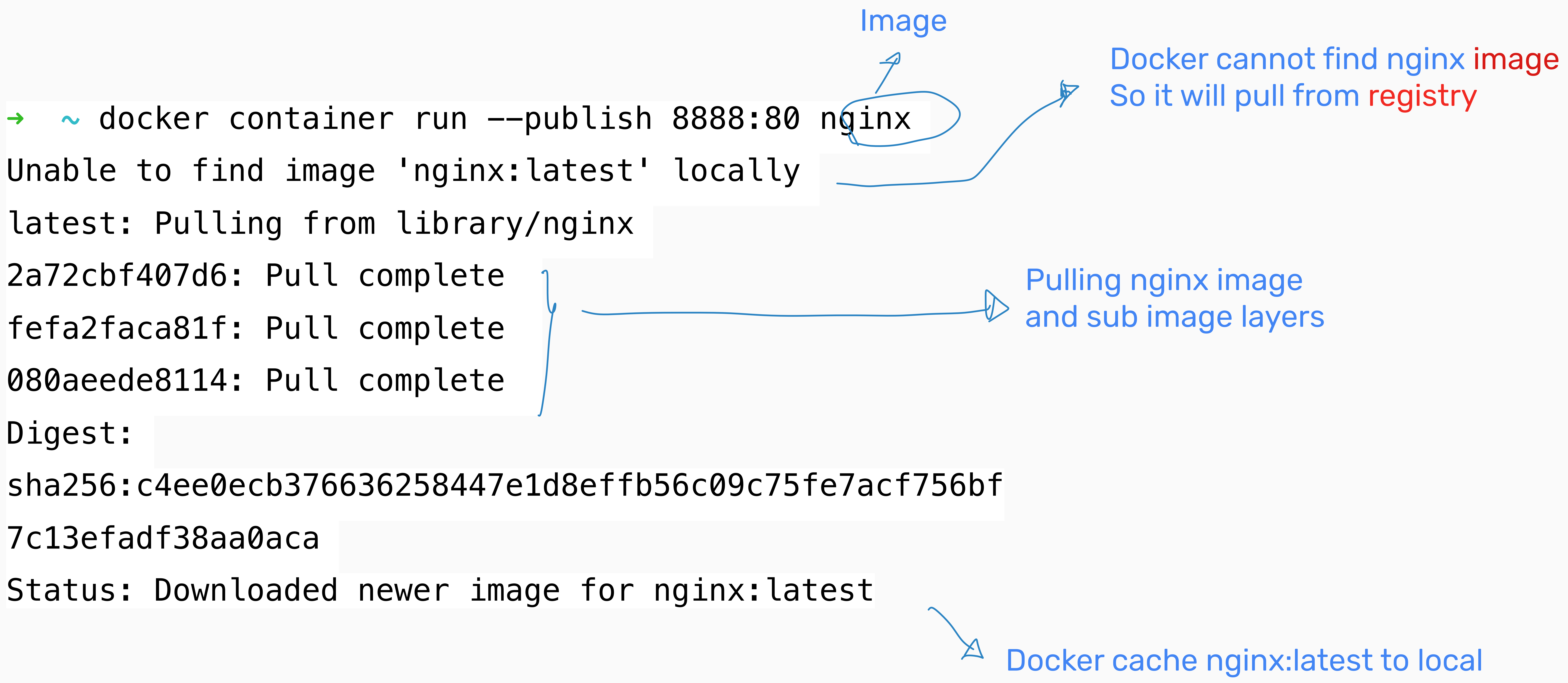
A subscription with support and certification for IT teams running critical apps in production.

GET DOCKER EE →

Basic: start nginx server

```
> docker container run --publish 8888:80 nginx
```

What happen here?



```
➜  ~ docker container run --publish 8888:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
2a72cbf407d6: Pull complete
fefa2faca81f: Pull complete
080aeede8114: Pull complete
Digest:
sha256:c4ee0ecb376636258447e1d8effb56c09c75fe7acf756bf
7c13efadf38aa0aca
Status: Downloaded newer image for nginx:latest
```

Image

Docker cannot find nginx image
So it will pull from registry

Pulling nginx image
and sub image layers

Docker cache nginx:latest to local

Docker Concept

Docker Basics

Image

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

Container

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

Engine

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

Registry

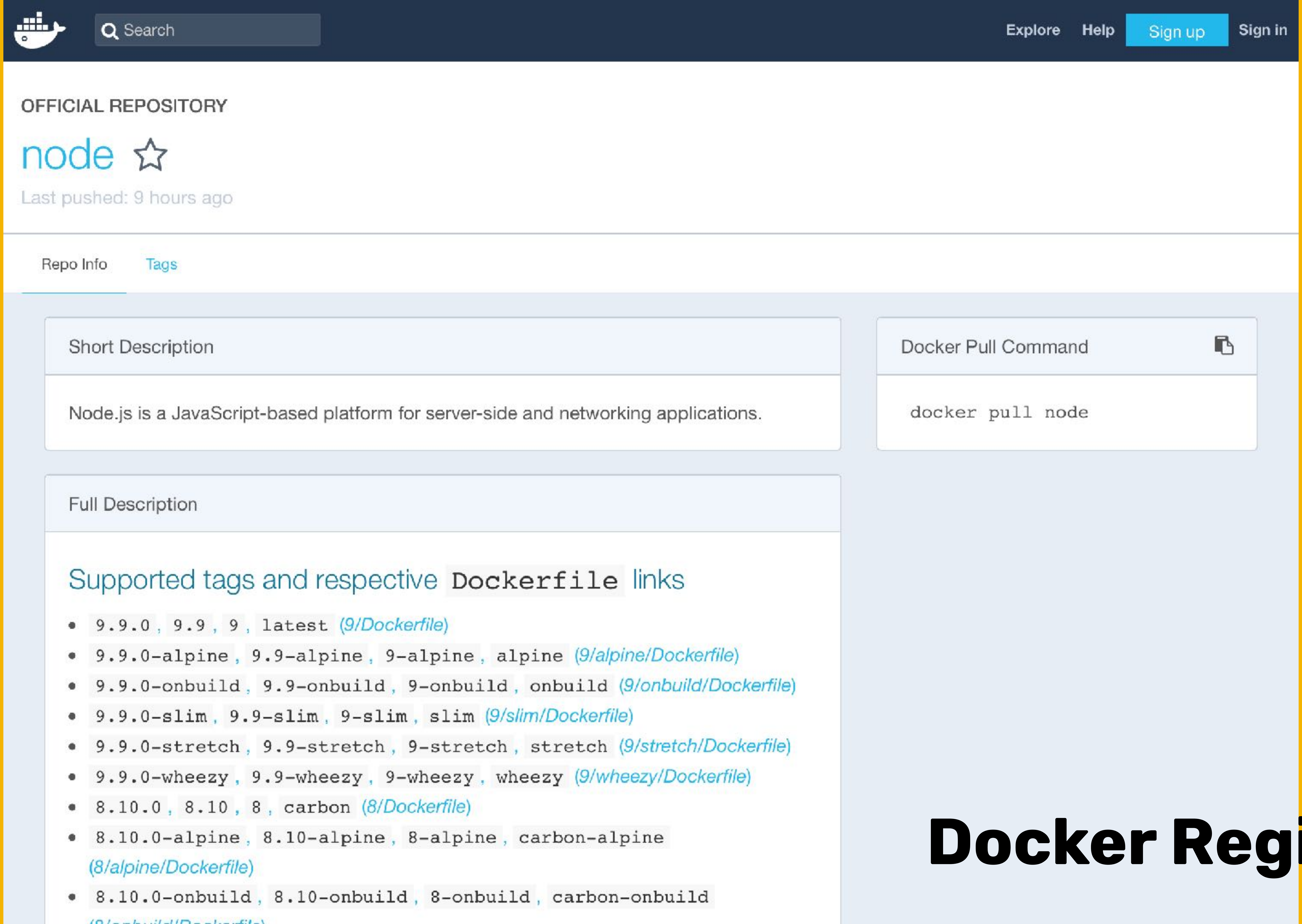
Stores, distributes and manages Docker images

Control Plane

Management plane for container and cluster orchestration

Image

- **Read-only** data that **contain Application and its dependencies**
- Blueprints for **Container**
- Eg. node, mysql, nginx, apache, python
- Hosted on **Docker Repository**
- Built from **Dockerfile**
- Layered
- <image name>, <image name>:<tag>, <repo>/<image name>:<tag>
  - node, node:8, node:8.10, node:8.10.1-alpine



OFFICIAL REPOSITORY

node

Last pushed: 9 hours ago

Repo Info Tags

Short Description

Node.js is a JavaScript-based platform for server-side and networking applications.

Docker Pull Command

```
docker pull node
```

Full Description

Supported tags and respective `Dockerfile` links

- `9.9.0`, `9.9`, `9`, `latest` (9/Dockerfile)
- `9.9.0-alpine`, `9.9-alpine`, `9-alpine`, `alpine` (9/alpine/Dockerfile)
- `9.9.0-onbuild`, `9.9-onbuild`, `9-onbuild`, `onbuild` (9/onbuild/Dockerfile)
- `9.9.0-slim`, `9.9-slim`, `9-slim`, `slim` (9/slim/Dockerfile)
- `9.9.0-stretch`, `9.9-stretch`, `9-stretch`, `stretch` (9/stretch/Dockerfile)
- `9.9.0-wheezy`, `9.9-wheezy`, `9-wheezy`, `wheezy` (9/wheezy/Dockerfile)
- `8.10.0`, `8.10`, `8`, `carbon` (8/Dockerfile)
- `8.10.0-alpine`, `8.10-alpine`, `8-alpine`, `carbon-alpine` (8/alpine/Dockerfile)
- `8.10.0-onbuild`, `8.10-onbuild`, `8-onbuild`, `carbon-onbuild`

Docker Registry

Container

- Running Application
- Read-Write Layer of Docker Image
- Have its own storage (volume), network
- Eg. Use MySQL image, we created container
    - Data persisted in docker container
    - If we use same container we can retrieve data in the database
    - Until we remove it

Image vs Container

Image

- **Read-Only** layers
- Contain application and dependencies
- Created from **Dockerfile**
- Use to create **Container(s)**

Container

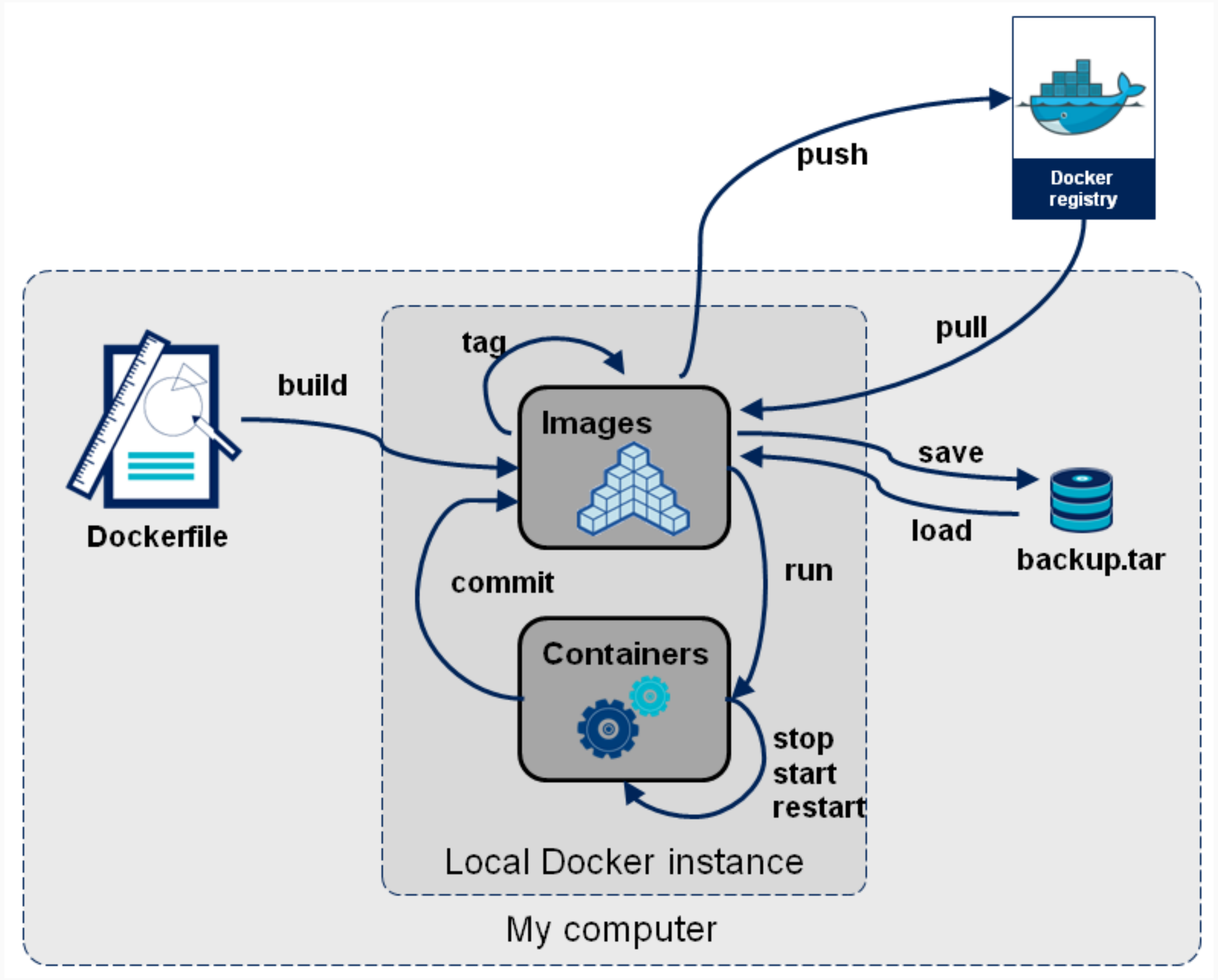
- **Read-Write** layers
- Running Application
- Created from **an Image**
- Use to deploy application/create service ...
- Have its own volume, network

Docker Daemon

- Actual docker process running on machine
  - Build Image
  - Run and manage containers
- Use docker cli to control it

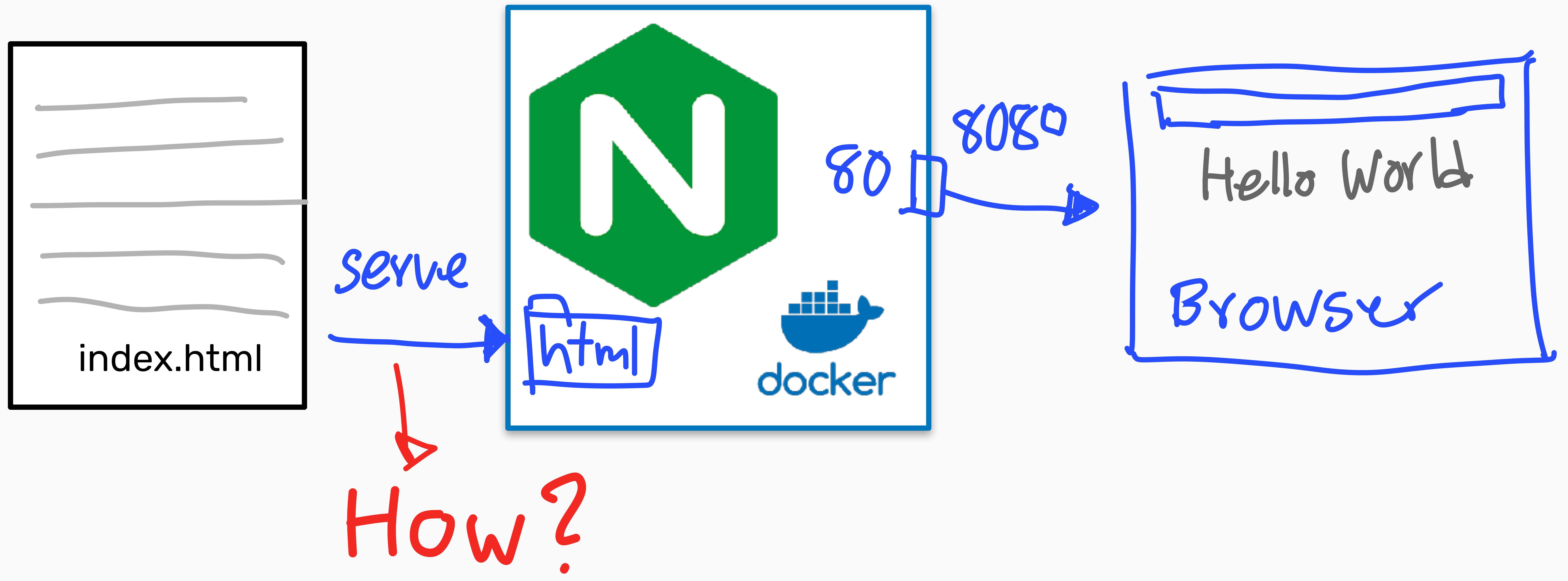
Docker Registry/Repository

- Server that host Docker Image
- Can be public (hub.docker.com)
- Can be private (AWS ECS Registry, Google Cloud Container Registry)
- Docker daemon will pull image from registry if it does not exist in machine
- Can have image on local machine without publish to registry


push
Docker registry
pull
tag
build
Images
save
load
backup.tar
Dockerfile
commit
run
Containers
stop
start
restart
Local Docker instance
My computer

Life Cycle

Demo: Serving Static Page Using NGINX



Demo: Serving Static Page Using NGINX

Container Name

Remove when stop

Expose port 80 to host's 8080

Image Name

```
docker run --name my-nginx --rm \
-p8080:80 \
-v `pwd`/public:/usr/share/nginx/html:ro nginx
```

Host

container

mode

Using host's ./public to container's /usr/share/nginx/html (read only)

Demo: Running MySQL (Environments)

```
➜  ~ docker run mysql:5.7
error: database is uninitialized and password option is not specified
  You need to specify one of MYSQL_ROOT_PASSWORD,
MYSQL_ALLOW_EMPTY_PASSWORD and MYSQL_RANDOM_ROOT_PASSWORD
```

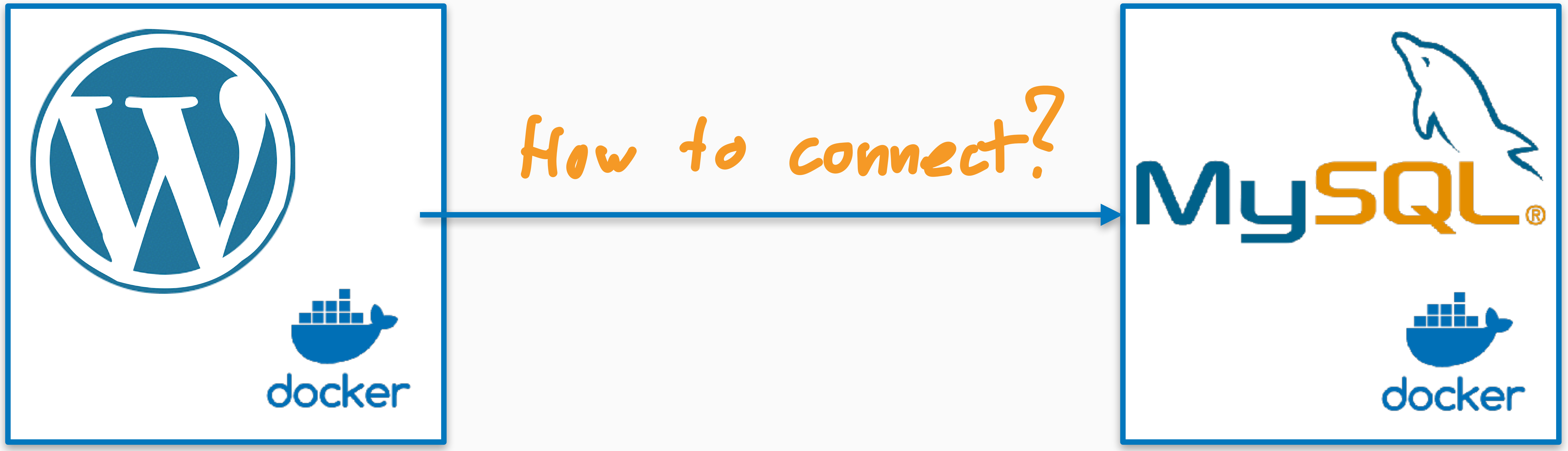
How to pass option?

Running MySQL

Passing Environment Variables

```
➜  ~ docker run --rm --name my-sql-example \
-e MYSQL_ALLOW_EMPTY_PASSWORD=1 \
-p3306:3306 \
mysql:5.7
```

Demo: Install Wordpress



Demo: Running Wordpress

```
→ docker run --name my-wordpress --rm \
-v `pwd`/public:/var/www/html \
-p 8081:80 \
--link my-sql-example:mysql \
wordpress:4.9.4-apache
```

1. Link to same network as my-sql-example

2. Connection to my-sql-example via "mysql" host

What we know so far...

- docker (container) run
  - --publish / -p
  - --volume / -v
  - --environment / -e
  - --link
  - --name
  - --rm
  - --detach / -d
- docker container ls (--all)
- docker stop/start/restart <container name>
- docker logs <container>

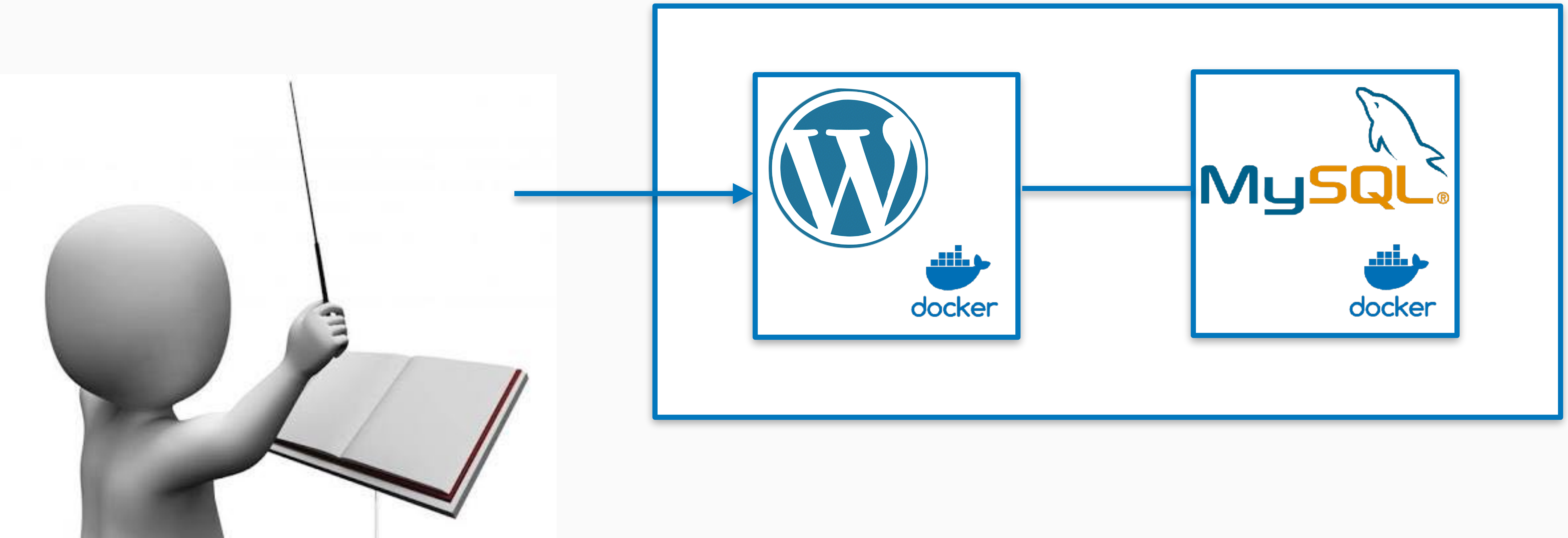
Now I know concept.
How can I really use it?

How to run multiple connected-containers?

- You have Node.js API, you want to connect to MySQL, you want to make the network go through Nginx
- It is possible to use docker-cli to run and create network. But it will consist of many command
- Not include configurations, update strategies, etc...

Container Orchestration

- Combine multiple containers as a service/application



docker-compose

- Write how to configure docker containers in docker-compose.yml file
- Then run `docker-compose up`
- Done!

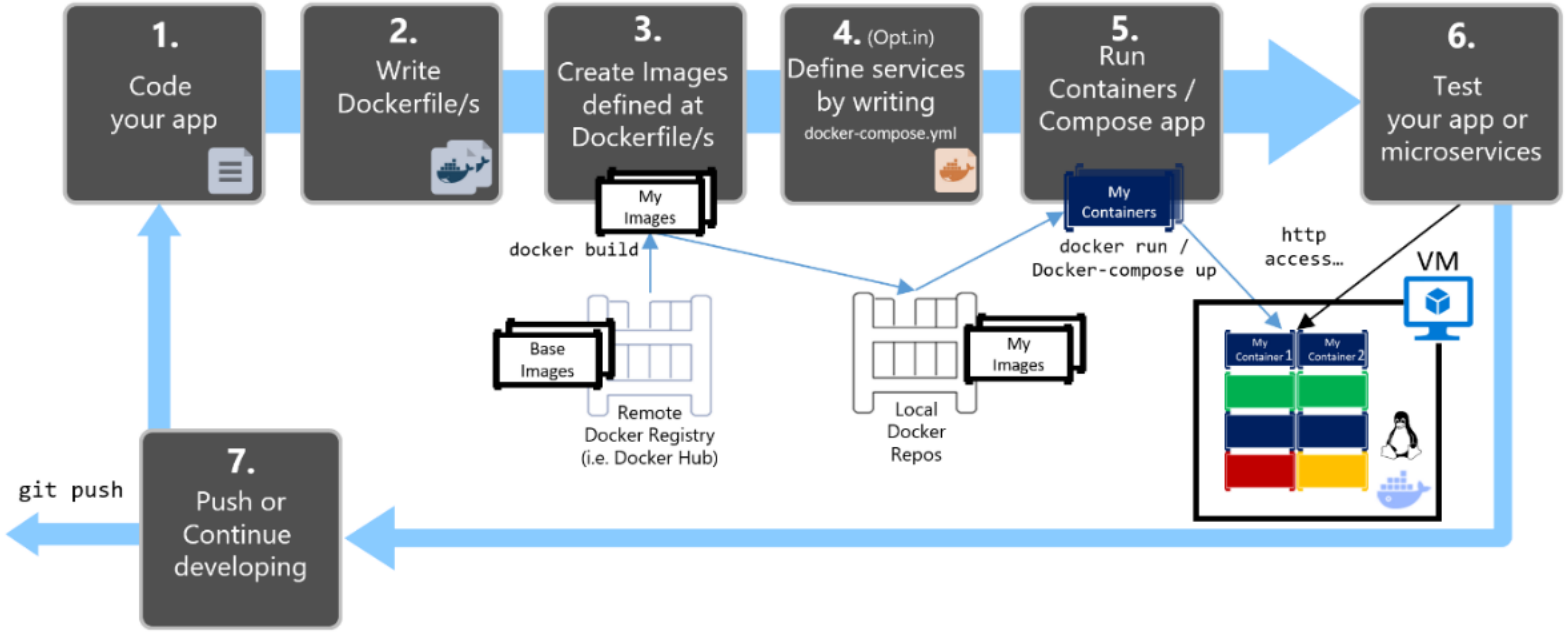
```
version: '3'

services:
 api:
   build:
     context: ./
     dockerfile: Dockerfile
   command: yarn start
   volumes:
     - .:/usr/app/
     - /usr/app/node_modules
   ports:
         -  "5000:3000"
   depends_on:
     - mysqldb
 mysqldb:
   image: mysql:5.7.17
   command: jkmysqld --character-set-server=utf8 --collation-
server=utf8_unicode_ci
   ports:
         -  "3206:3306"
   environment:
     MYSQL_ROOT_PASSWORD: abcdef
     MYSQL_USER: api-user
     MYSQL_PASSWORD: abcdef
     MYSQL_DATABASE: server
```

In api, configure MySQL host to ‘mysqldb’

docker-compose

- Single machine
- Very Simple Configuration
- Suite for simple application
    - Eg. Testing, Local Development
- You may have to configure other component: eg. Cloud Provider's Load Balancers

Inner-Loop development workflow for Docker apps



1. Code your app
2. Write Dockerfile/s
3. Create Images defined at Dockerfile/s
4. (Opt.in) Define services by writing docker-compose.yml
5. Run Containers / Compose app
6. Test your app or microservices
7. Push or Continue developing
My Images
docker build
Base Images
Remote Docker Registry (i.e. Docker Hub)
Local Docker Repos
My Images
My Containers
docker run / Docker-compose up
http access…
VM
My Container 1
My Container 2
git push

https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/docker-application-development-process/docker-app-development-workflow

Dockerfile

- Build Your Own Docker Image(s)
- Package your code
- docker build -t <image name>:<image version> .

Dockerfile

```
FROM node:8.10-alpine

MAINTAINER Roparat Sukapirom <roparat@skooldio.com>

COPY ./package.json /code/
COPY ./yarn.lock /code/yarn.lock

RUN cd /code && yarn install

COPY ./build/ /code/

WORKDIR /code

RUN ls -al

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

What my Dockerfile based on

My custom command for this Image

Node has yarn, so it can run
Otherwise, write install command

Set working directory. Eg. Where is my starting path for application

Run this command when container is created

Docker build

- Docker Build Separate Image for Each steps in Dockerfile
- So when rebuilding, it can use cached images
- In example, package.json may not change much, so we move it first
    - Package installation require some time, so we cache image of installed packaged
- We can tag built image, so we can push to repository later

Now I have my own Image, how can I configure my application?

- Mount configuration file
  - -v ./config.json:/usr/app/config.json
- Passing as ENV
  - -e NODE_ENV=production

Real Life Production

Simple Application

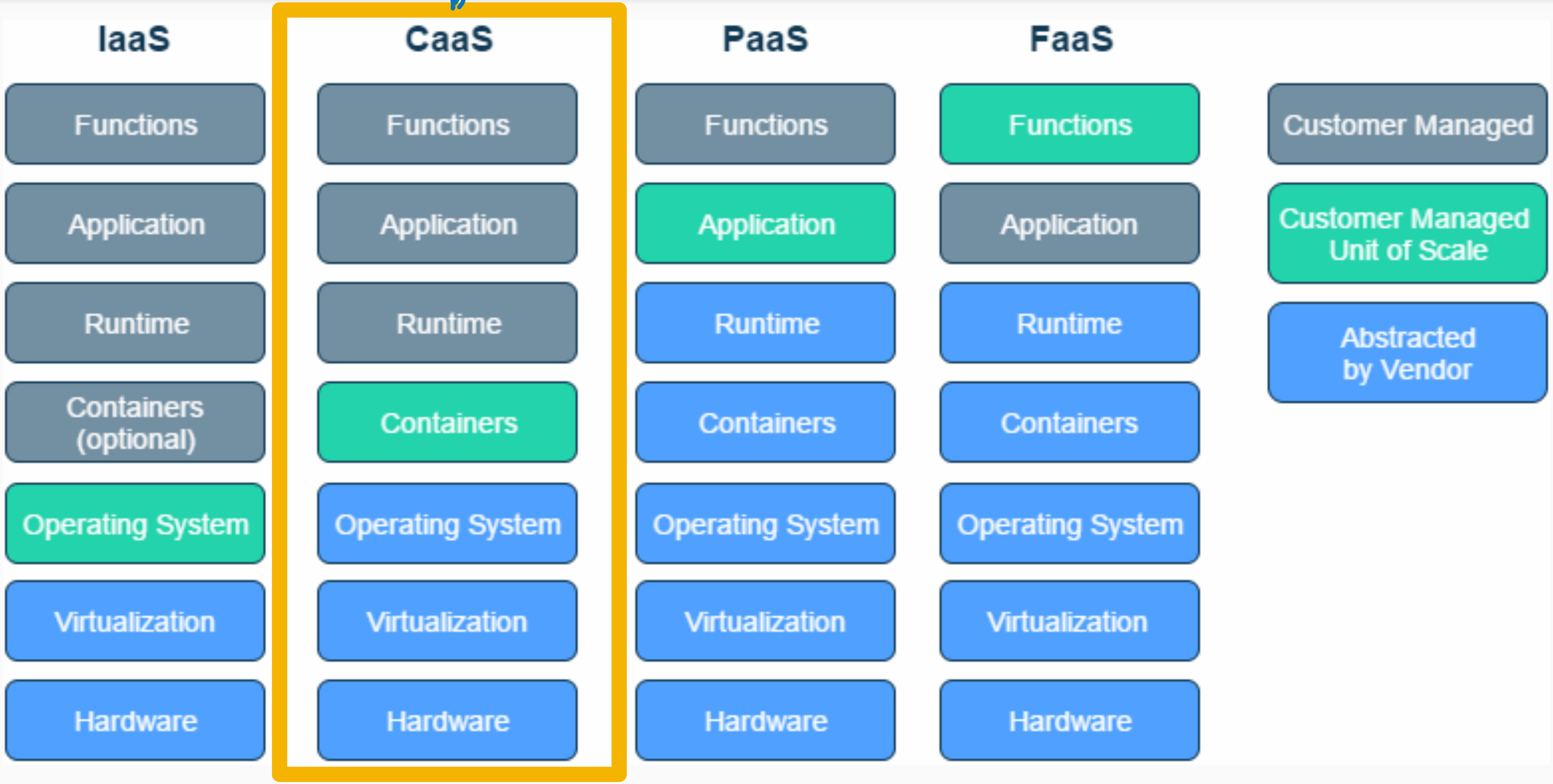
- You can just use docker-compose
  - Can also remote deploy

What if I want to use docker on multiple machines (clusters)

- Mostly Two Choices
  - Docker Swarm
  - Kubernetes
  - AWS ECR
- If you want to know one, learn Kubernetes!

___ as a Service

container as a service



IaaS	CaaS	PaaS	FaaS	
⬜ Functions	⬜ Functions	⬜ Functions	🟩 Functions	⬜ Customer Managed
⬜ Application	⬜ Application	🟩 Application	⬜ Application	🟩 Customer Managed Unit of Scale
⬜ Runtime	⬜ Runtime	🟦 Runtime	🟦 Runtime	🟦 Abstracted by Vendor
⬜ Containers (optional)	🟩 Containers	🟦 Containers	🟦 Containers	
🟩 Operating System	🟦 Operating System	🟦 Operating System	🟦 Operating System	
🟦 Virtualization	🟦 Virtualization	🟦 Virtualization	🟦 Virtualization	
🟦 Hardware	🟦 Hardware	🟦 Hardware	🟦 Hardware	

Q&A

Skooldio

Thank You

We are hiring!

people@skooldio.com
FB: Skooldio