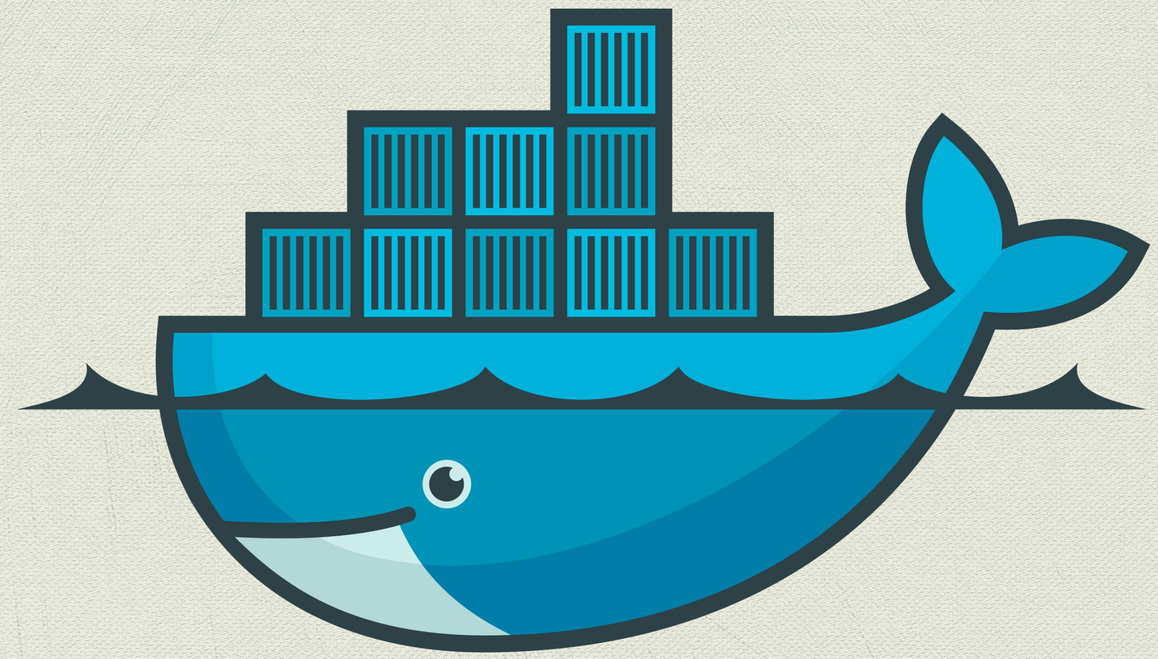


docker

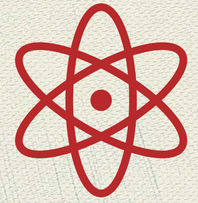
Quality Excellence

Ekapol W.

Key take away

Get to know Docker Anatomy

How to create Docker Image and Container



Build up Container Network

Daily Life Scenarios

It is working on my machine.

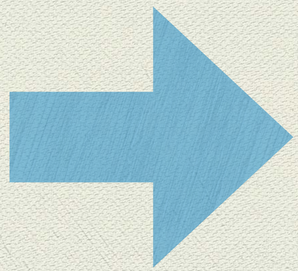
Develop, SIT, UAT and Production are not the same.

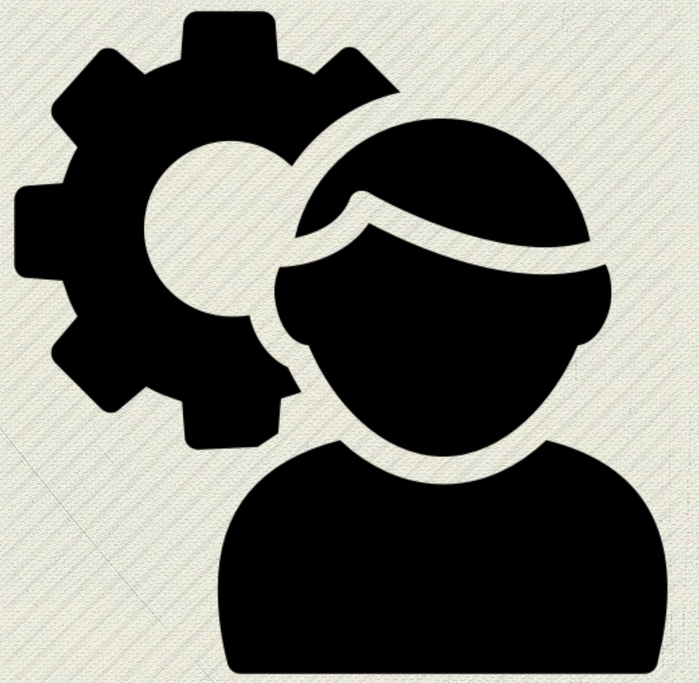
Software is broken every time change environment.

Traditional Software Transition

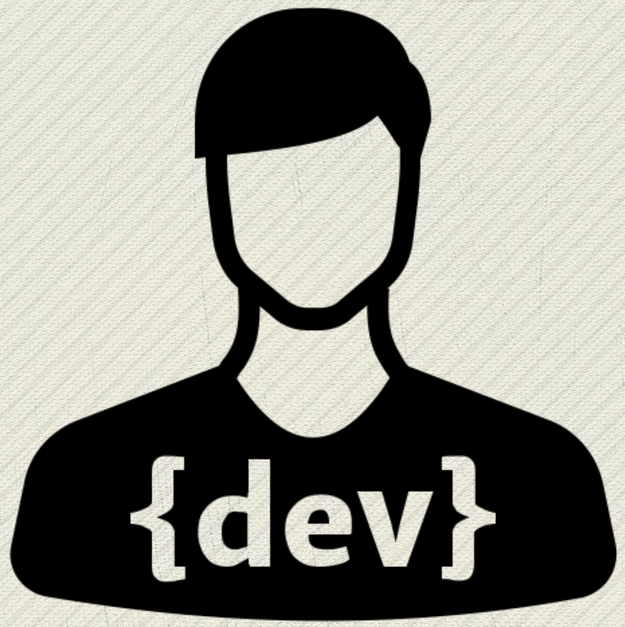
Software .ear .war .jar

Installation Instruction

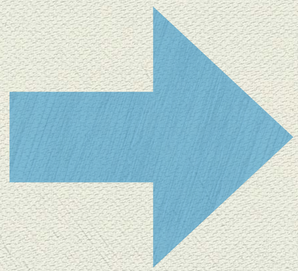


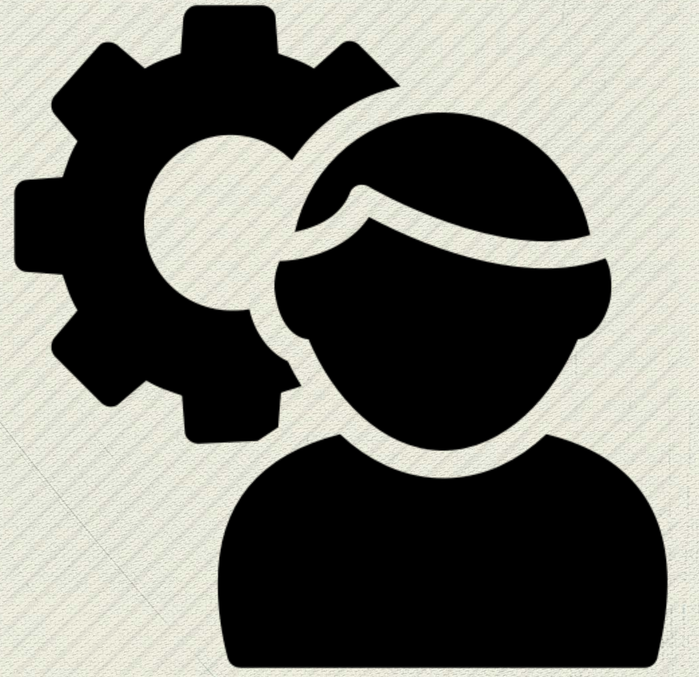


Modern Software Transition

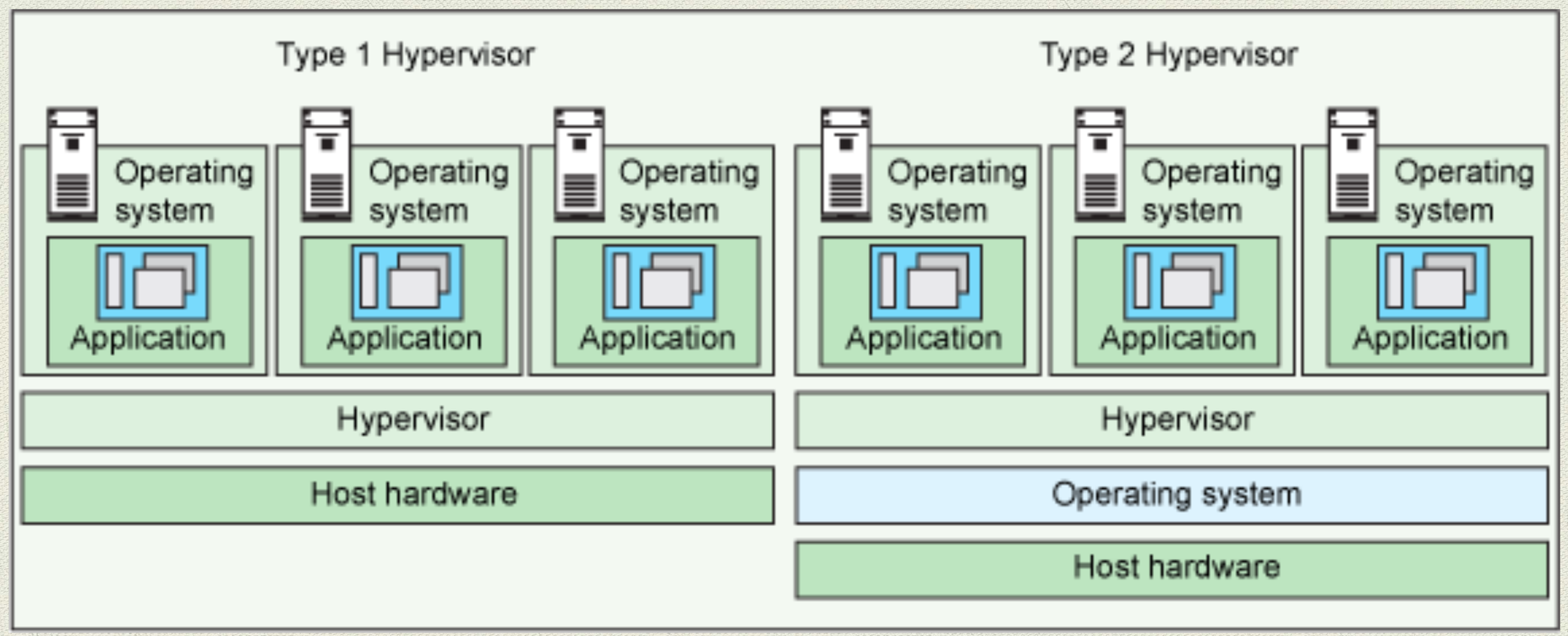


Images





Virtual Machine



Type 1 Hypervisors: *Xen, VMware ESXi, and Microsoft Hyper-*

Type 2 Hypervisors: *VMware Workstation, VMware Player, Oracle VirtualBox*

What is Container ?

Application

Runtime

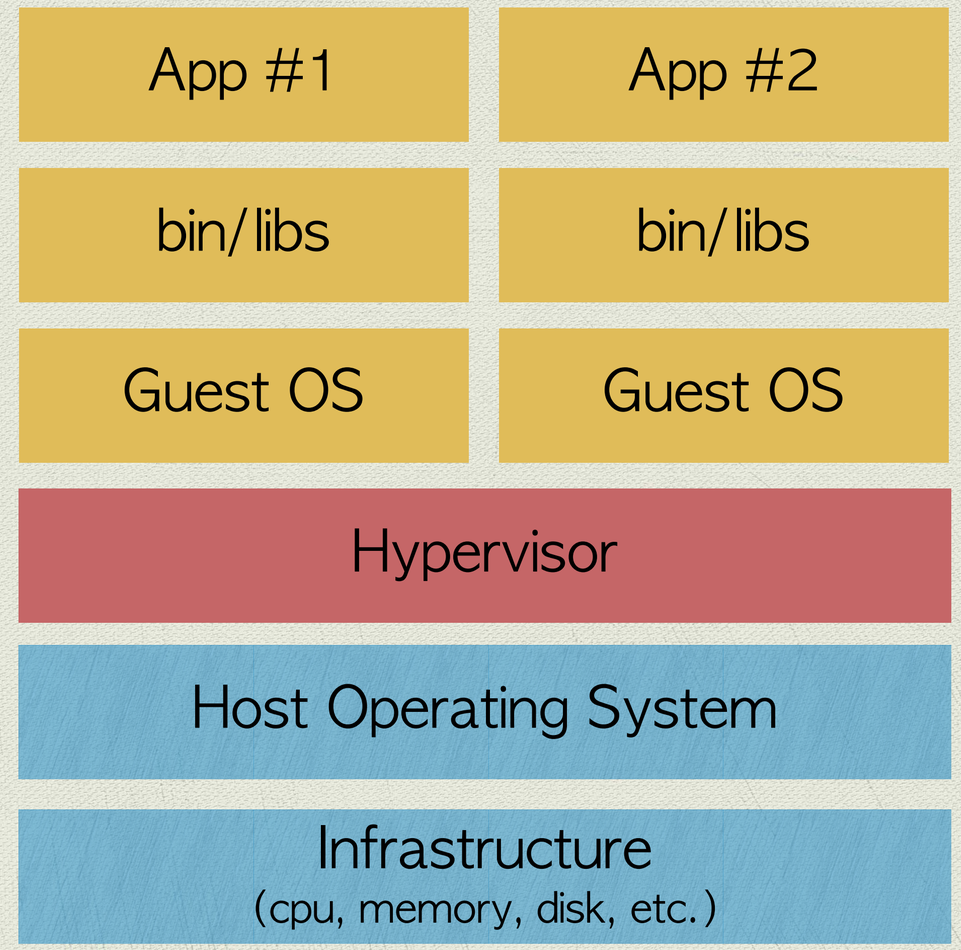
System tools

System Libraries

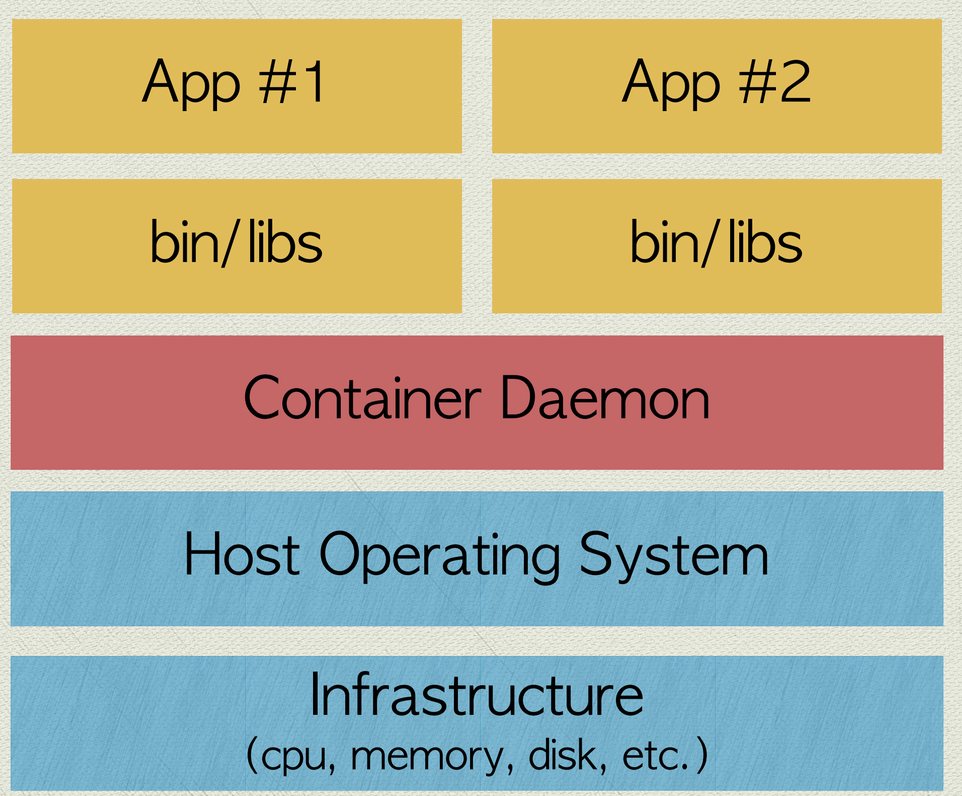
Setting & Configuration

Package Software into Standardized Units for Development, Shipment and Deployment

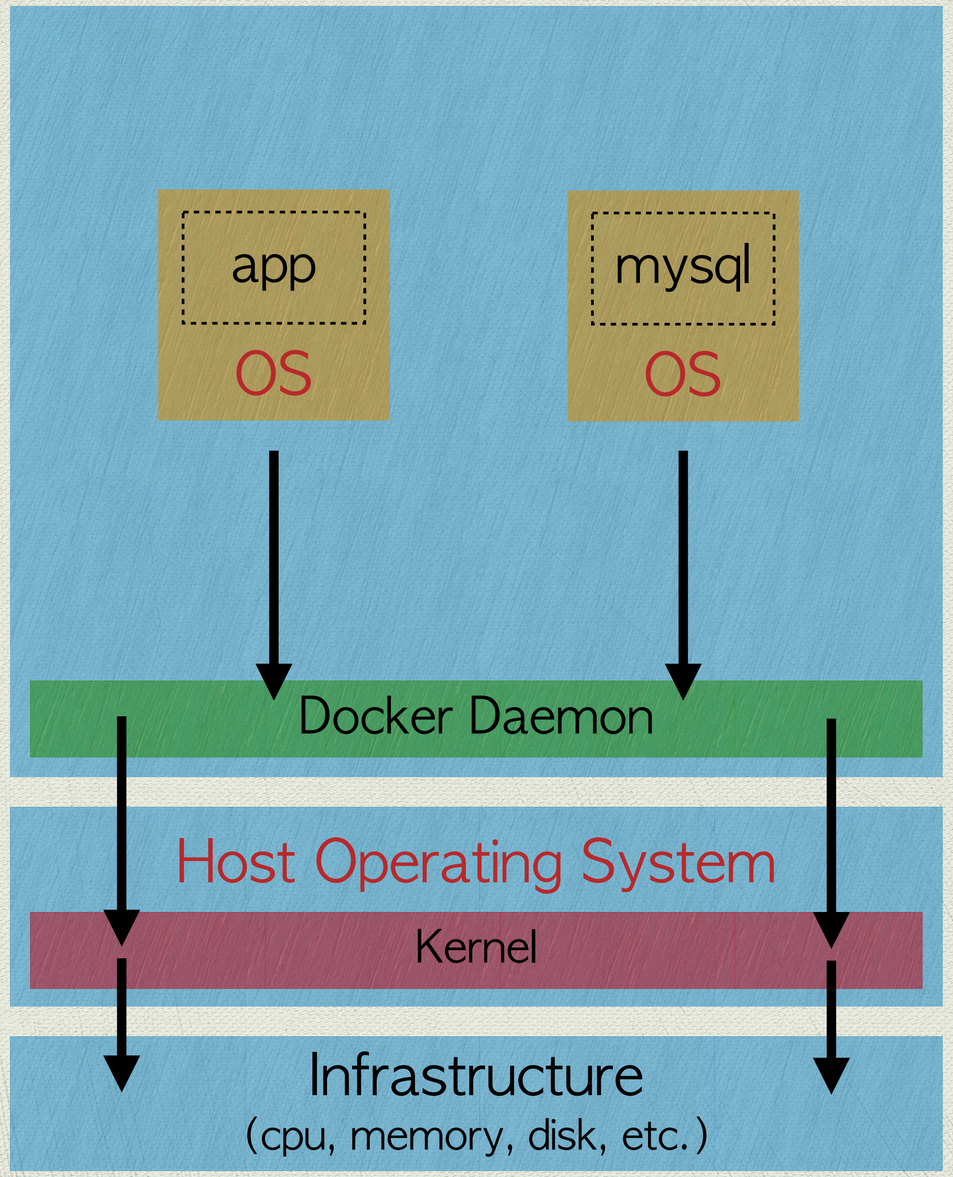
Virtual Machines



Containers



Containers virtualise the operating system instead of hardware.



Every OS must use the same Kernel.

Linux

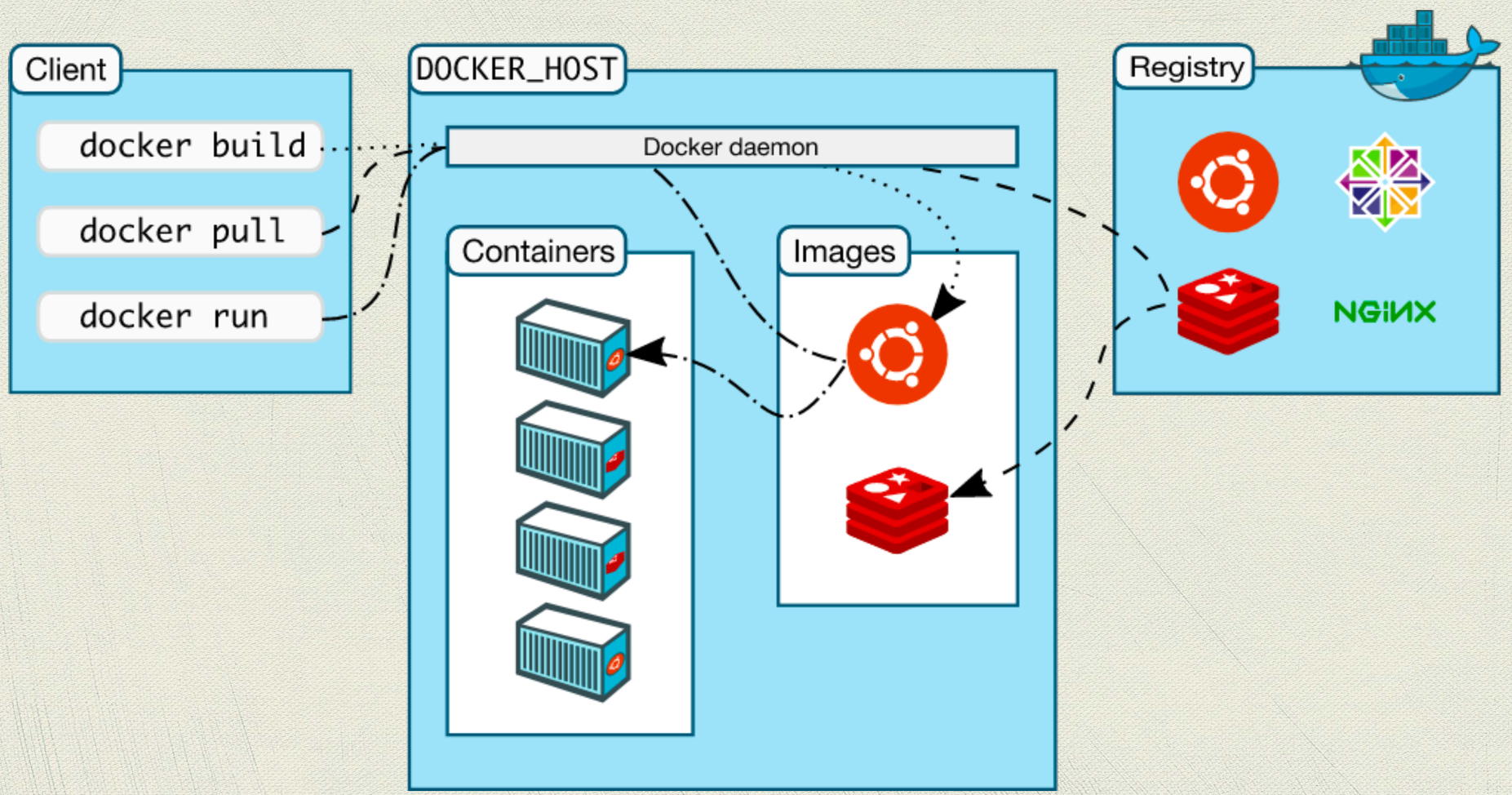
- Debian / Ubuntu
- Redhat
- Alpine

Windows Server

- Microsoft Server Core
- Microsoft Nano Server

How about Linux container on Windows ?

Docker Architecture



https://docs.docker.com/engine/docker-overview/

Installing Docker Desktop

https://www.docker.com/products/docker-desktop

Download


```
Ekapols-MBP:~ ekapolwongnapapan$ docker --version
Docker version 19.03.2, build 6a30dfc
Ekapols-MBP:~ ekapolwongnapapan$
```

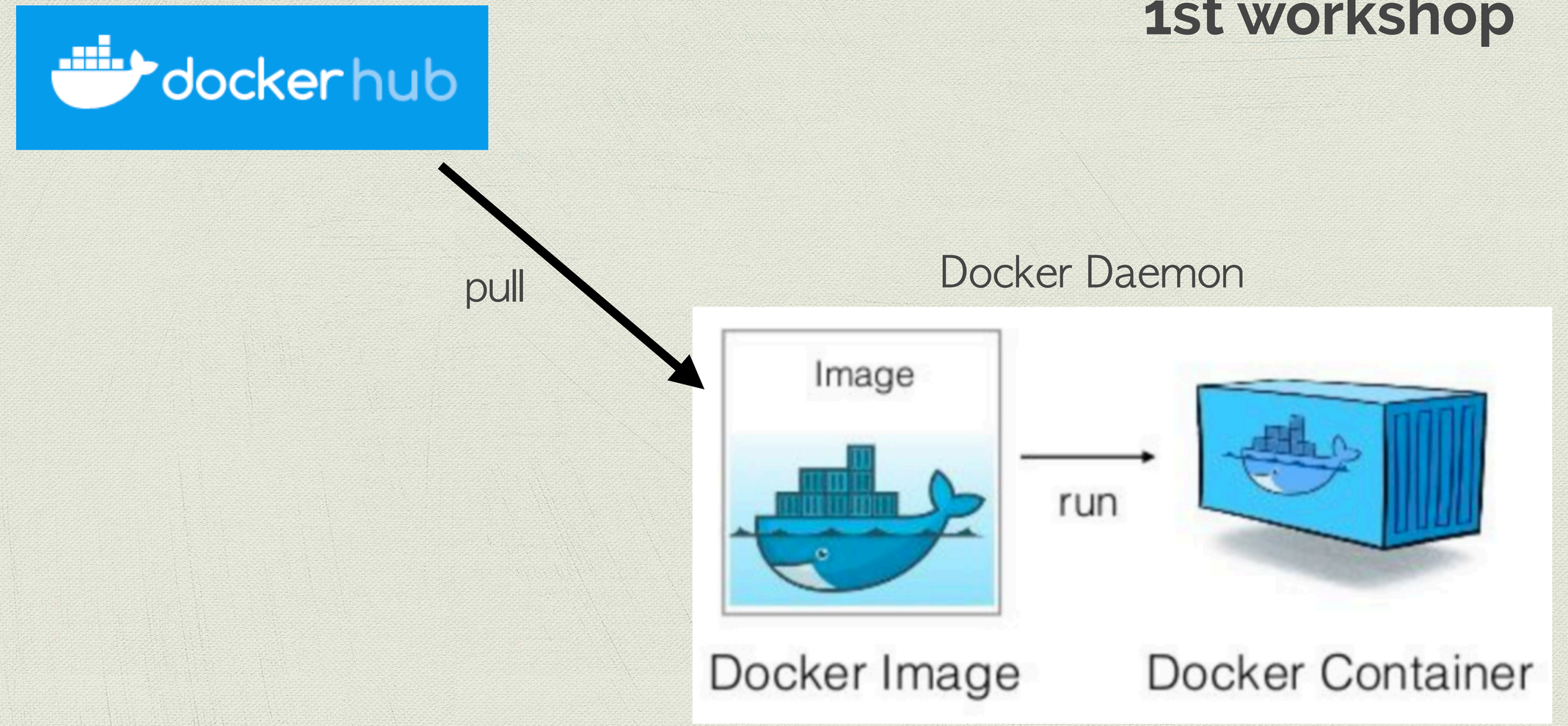
Validate Docker Installation

```
Ekapols-MBP:~ ekapolwongnapapan$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:4df8ca8a7e309c256d60d7971ea14c27672fc0d10c5f303856d7bc48f8cc17ff
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

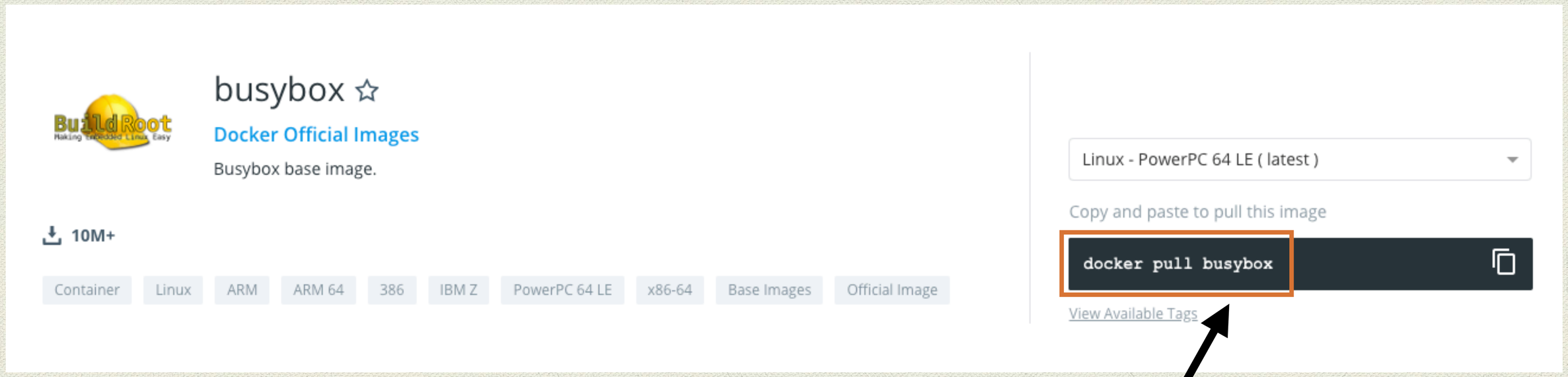
```
Ekapols-MBP:~ ekapolwongnapapan$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              fce289e99eb9        10 months ago       1.84kB
Ekapols-MBP:~ ekapolwongnapapan$
```

The image for hello-world

1st workshop



Docker Hub



Run this command to pull the image.

https://hub.docker.com/

Pull images into local repository

```
Ekapols-MBP:~ ekapolwongnapapan$ docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
0f8c40e1270f: Pull complete
Digest: sha256:1303dbf110c57f3edf68d9f5a16c082ec06c4cf7604831669faf2c712260b5a0
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
Ekapols-MBP:~ ekapolwongnapapan$
```

```
Ekapols-MBP:~ ekapolwongnapapan$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
busybox             latest              020584afccce        3 weeks ago         1.22MB
hello-world         latest              fce289e99eb9        10 months ago       1.84kB
Ekapols-MBP:~ ekapolwongnapapan$
```

Run container based on the image

```
Ekapols-MBP:~ ekapolwongnapapan$ docker run busybox
Ekapols-MBP:~ ekapolwongnapapan$
```

```
Ekapols-MBP:~ ekapolwongnapapan$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
906df8c13a36        busybox             "sh"                3 minutes ago       Exited (0) 3 minutes ago                        ecstatic_nightingale
Ekapols-MBP:~ ekapolwongnapapan$
```

- Create a container from image.
- Run the command in the container.
- Exit the container.

Run container with command

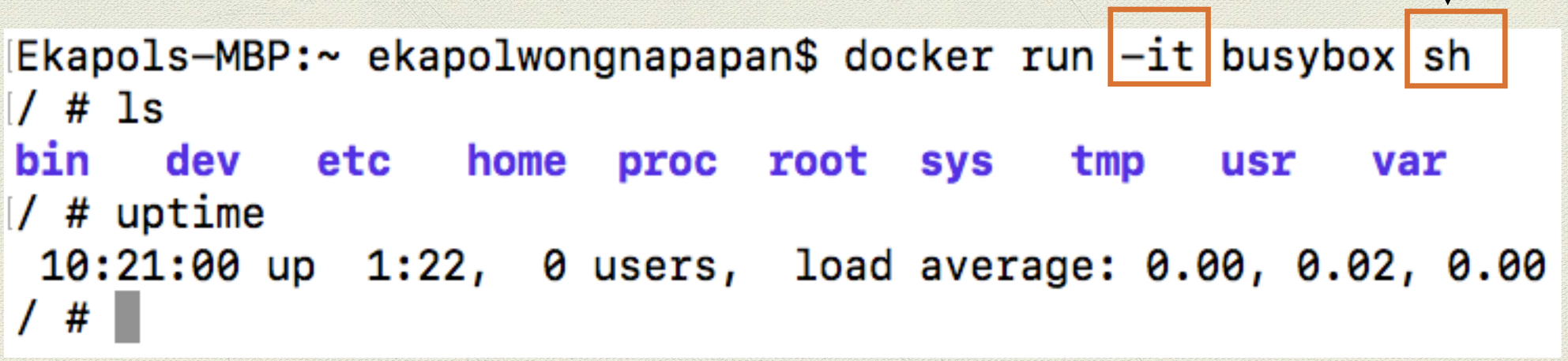
```
Ekapols-MBP:~ ekapolwongnapapan$ docker run busybox echo "hello from busybox"
hello from busybox
```

```
Ekapols-MBP:~ ekapolwongnapapan$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS               NAMES
5f1ba64694d8        busybox             "echo 'hello from bu…"   20 minutes ago      Exited (0) 20 minutes ago                       inspiring_haslett
906df8c13a36        busybox             "sh"                     28 minutes ago      Exited (0) 28 minutes ago                       ecstatic_nightingale
```

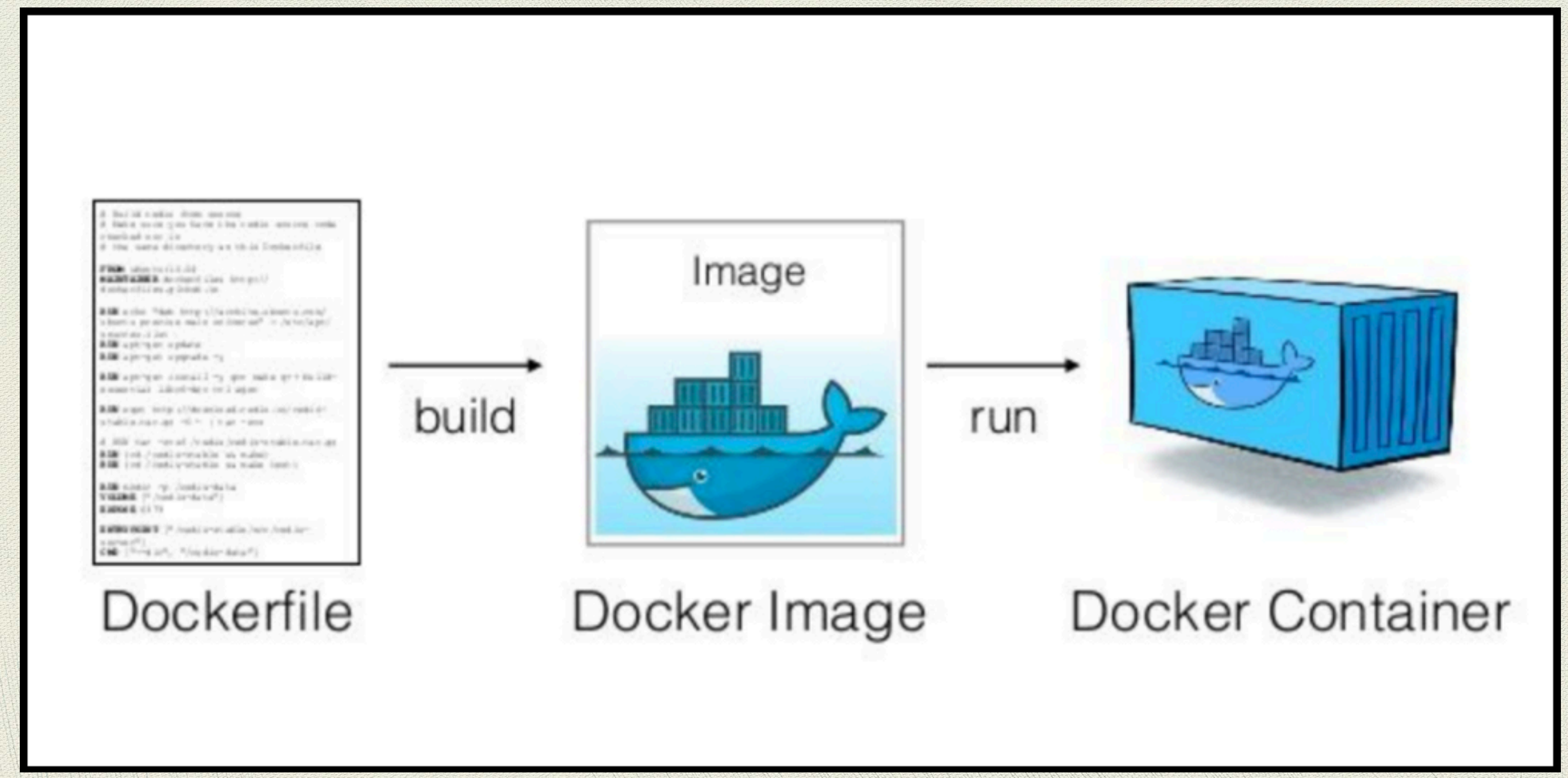
- Create a container from image.
- Run the command in the container.
- Exit the container.

Run container with interactive tty

Enter Bourne shell



Create image and container



Dockerfile

1. Write Dockerfile

```
FROM busybox:latest
CMD ["echo", "hello from busybox"]
```

2. Build image

```
docker build -t ekapolw/hellobusy .
```

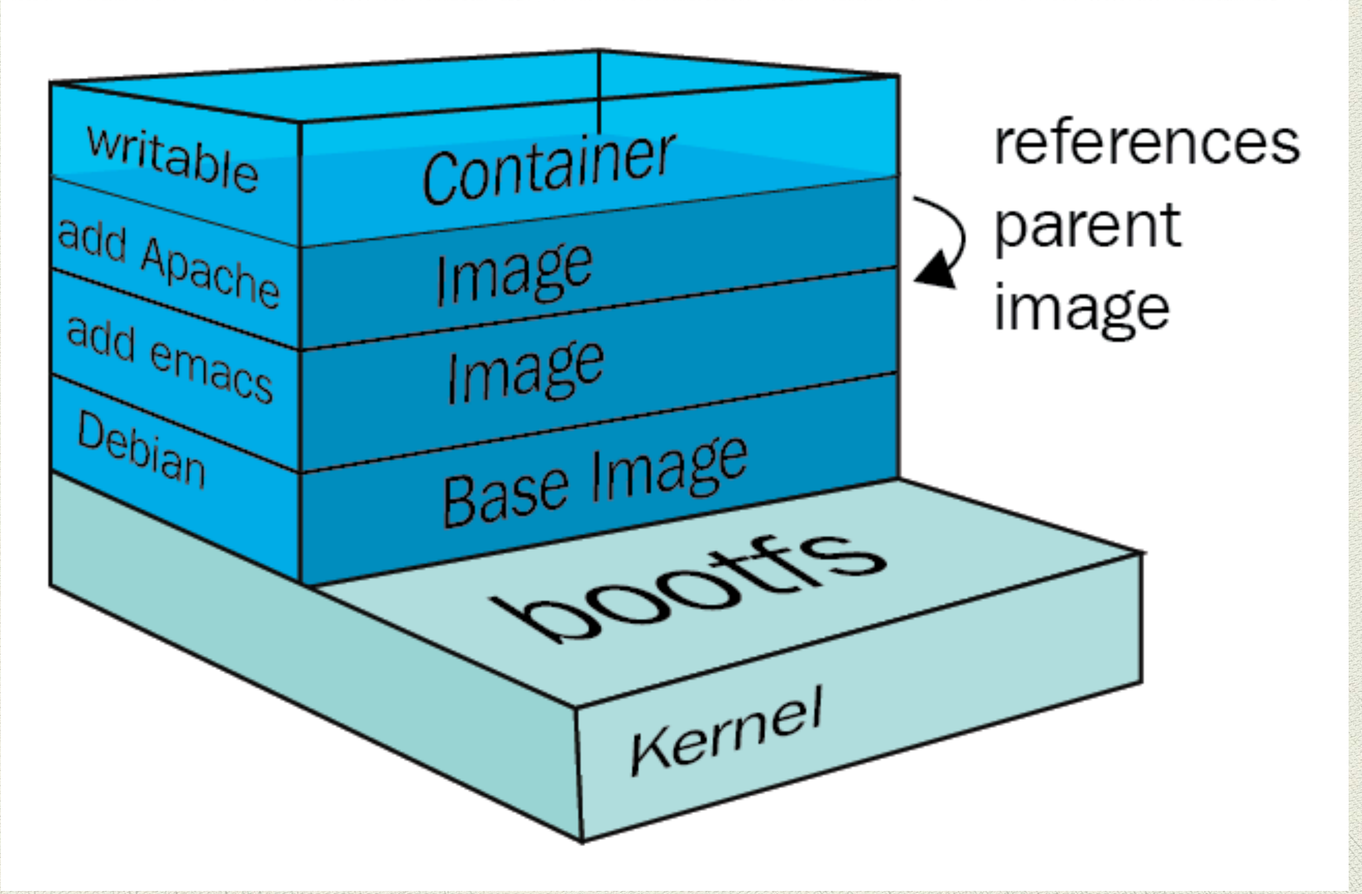
3. List image

```
ekapolwongnapapan@Ekapols-MacBook-Pro online_workshop % docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
ekapolw/hellobusy           latest              4687063799ec        2 minutes ago       1.22MB
```

4. Run Container

```
ekapolwongnapapan@Ekapols-MacBook-Pro online_workshop % docker run ekapolw/hellobusy
hello from busybox
```

Image Structure



Images

An image is a read-only template with instructions for creating a Docker container.

Base Image (example)

scratch ☆

Docker Official Images

an explicitly empty image, especially for building images "FROM scratch"

debian ☆

Docker Official Images

Debian is a Linux distribution that's composed entirely of free and open-source software.

busybox ☆

Docker Official Images

Busybox base image.

alpine ☆

Docker Official Images

A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!

ubuntu ☆

Docker Official Images

Ubuntu is a Debian-based Linux operating system based on free software.

centos ☆

Docker Official Images

The official build of CentOS.

openjdk ☆

Docker Official Images

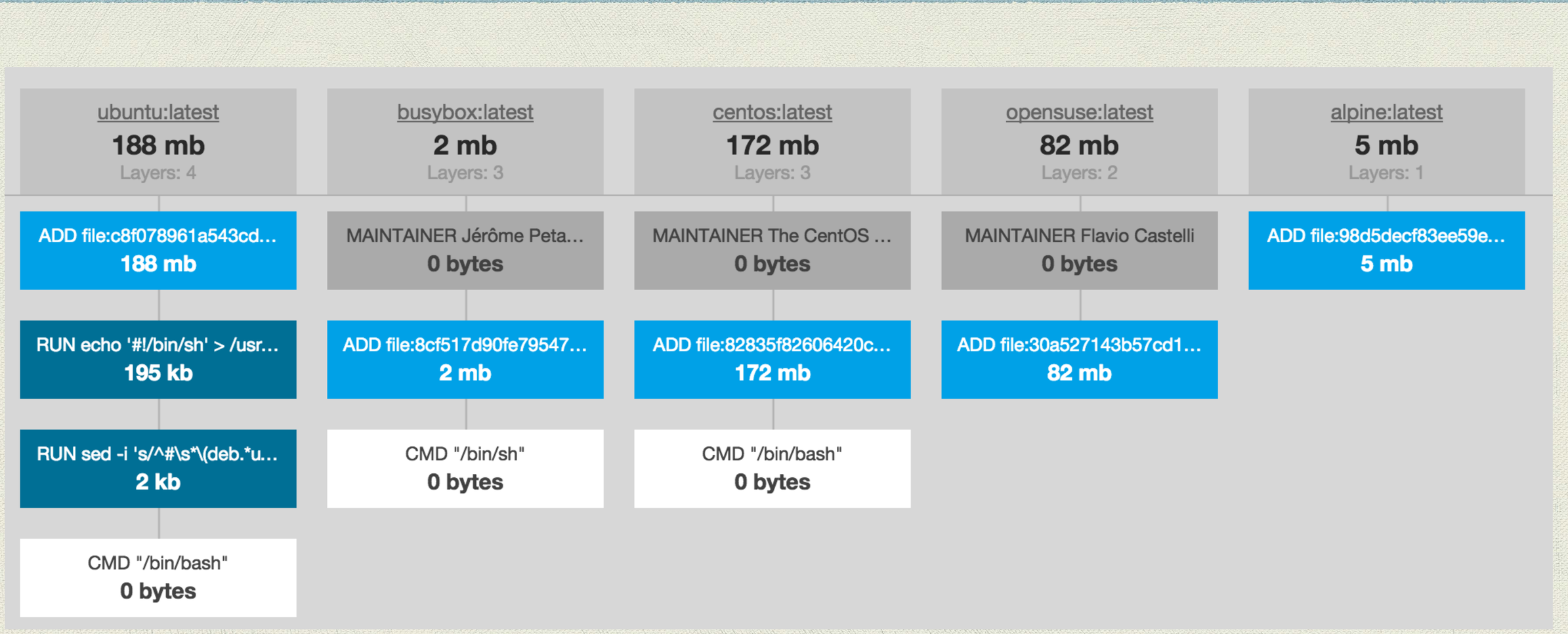
OpenJDK is an open-source implementation of the Java Platform, Standard Edition

python ☆

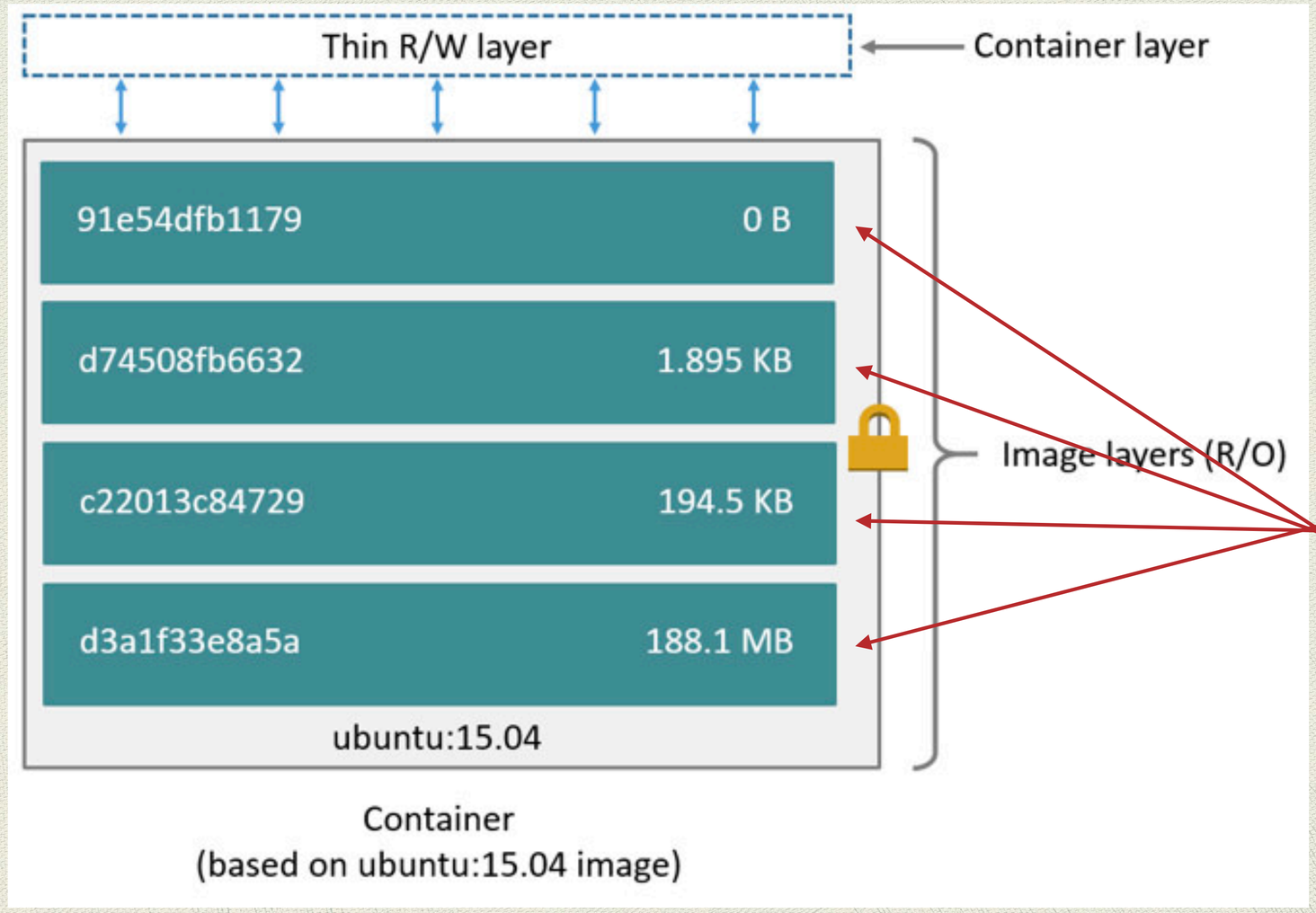
Docker Official Images

Python is an interpreted, interactive, object-oriented, open-source programming language.

Base Image Size



Container



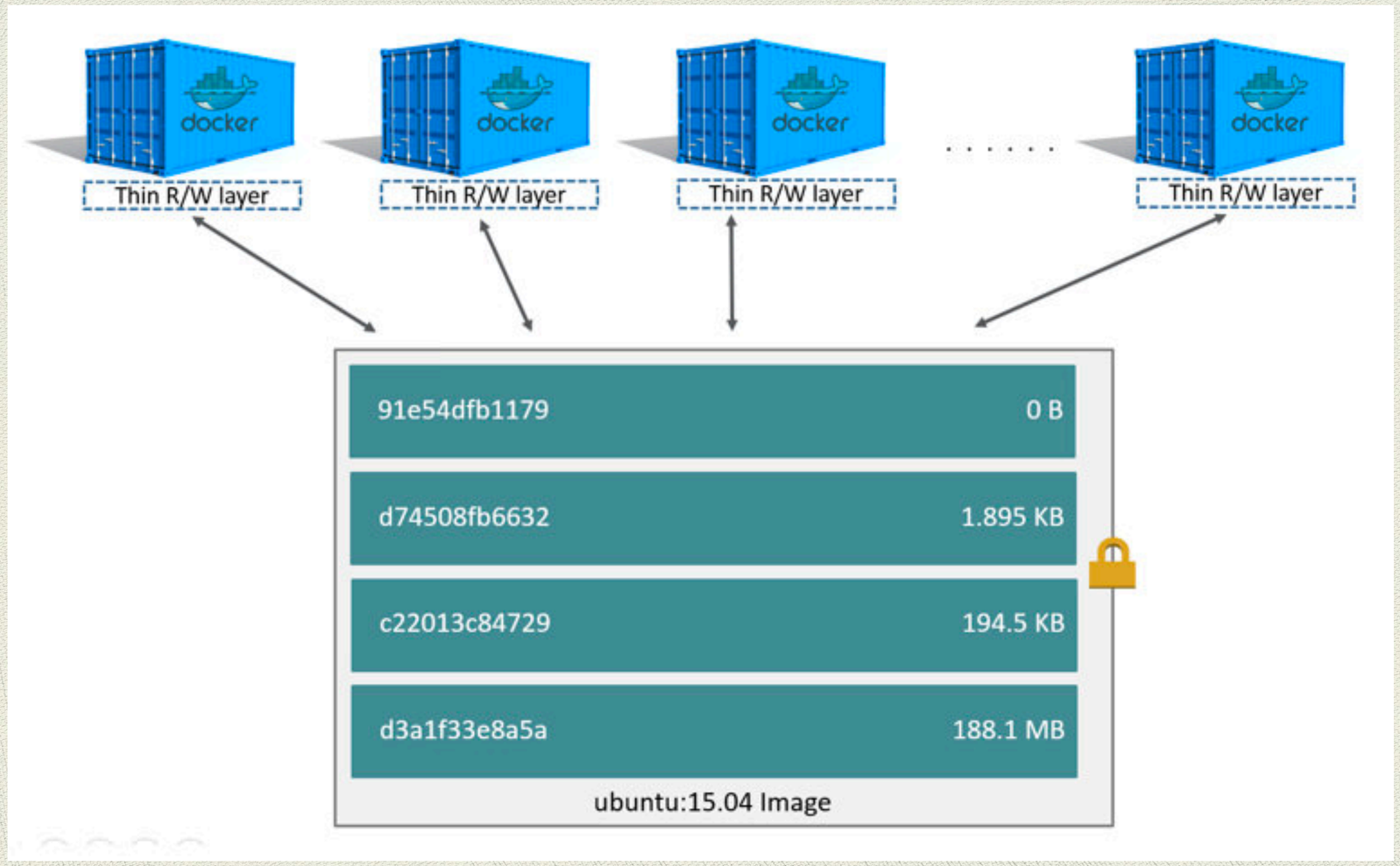
Containers

A container is a runnable instance of an image.

Dockerfile

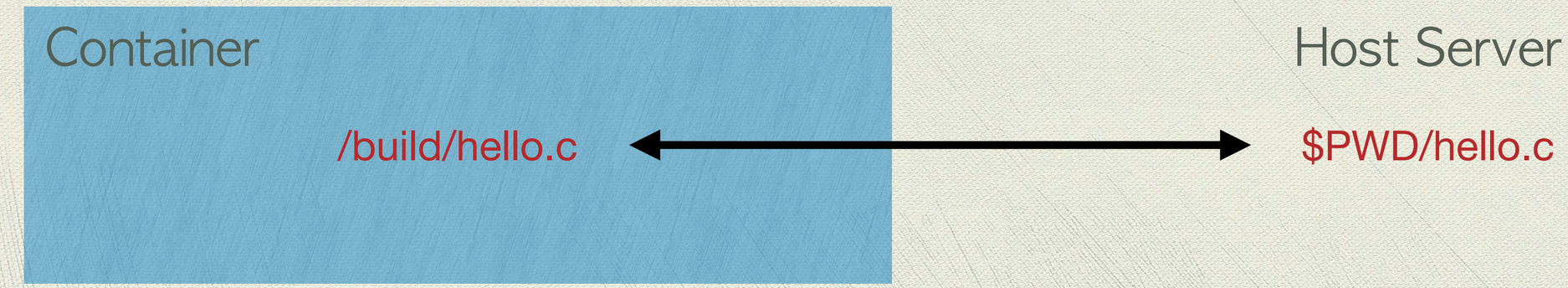
```
FROM ubuntu:15.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

Multiple Containers



Hello my edition (2nd workshop)

- Copy hello.c from https://github.com/docker-library/hello-world/ into local directory.
- Docker Desktop for Windows or Mac is using Linux VM, so we need to build hello binary in Linux, not Windows or Mac.
- In local directory where place hello.c, run “docker run --rm -it -v $PWD:/build ubuntu:16.04”
- In container, apt-get update && apt-get install build-essential
- In container, cd /build
- In container, gcc -o hello -static -nostartfiles hello.c



Dockerfile

A Dockerfile is a simple text file that contains a list of commands that Docker client calls while creating an image.

Dockerfile

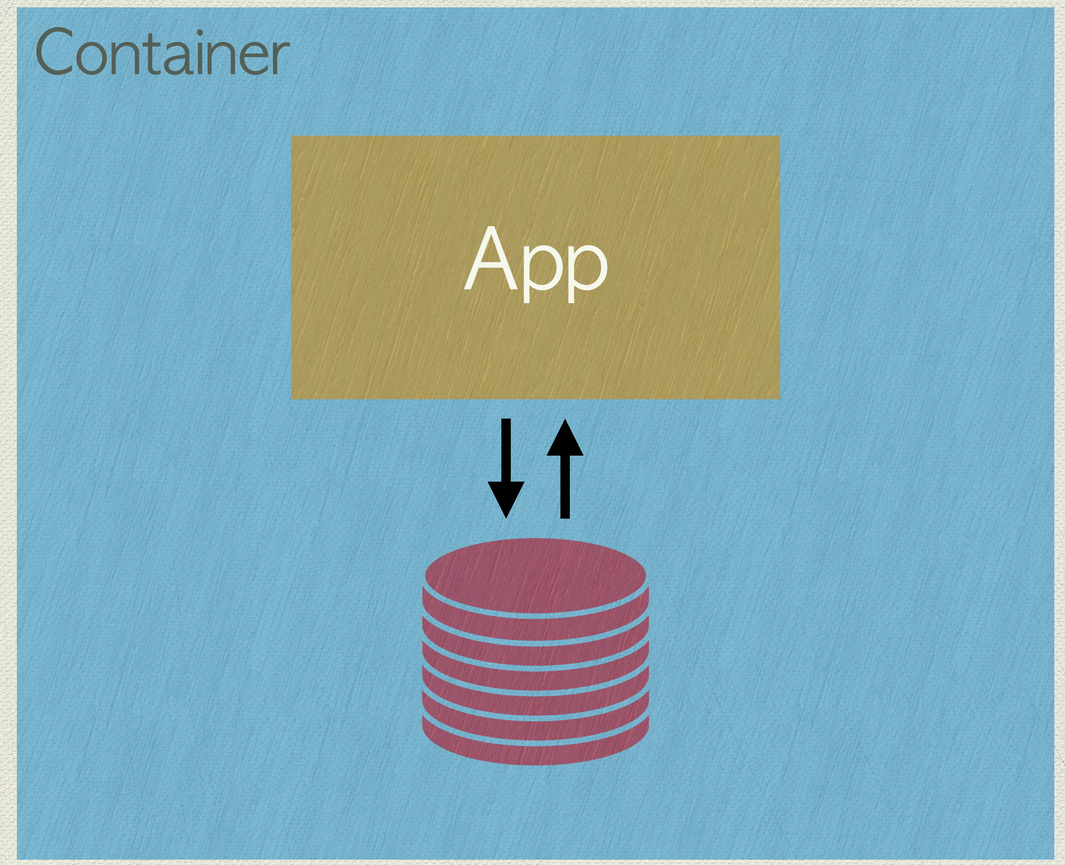
```
FROM scratch
ADD hello /
CMD ["/hello"]
```

docker build --tag hello .

Image

* See how to build hello in next page

Persistence ?

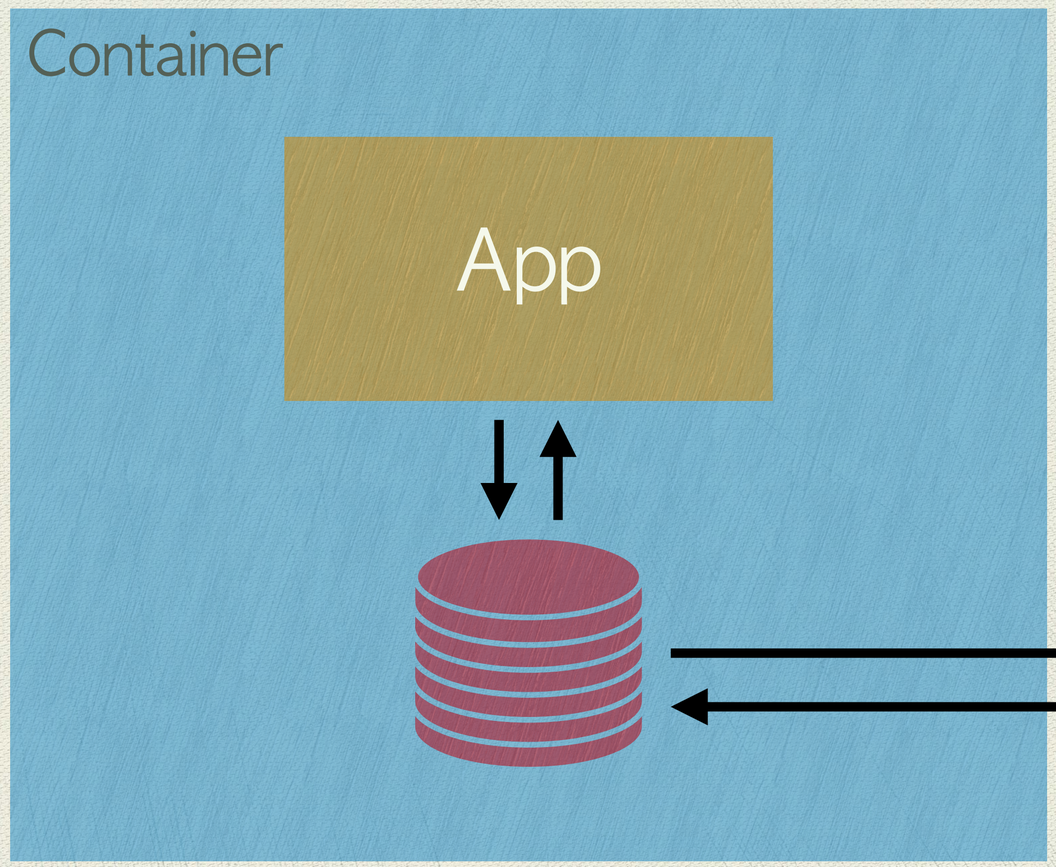


When stop container, data could not be accessed.

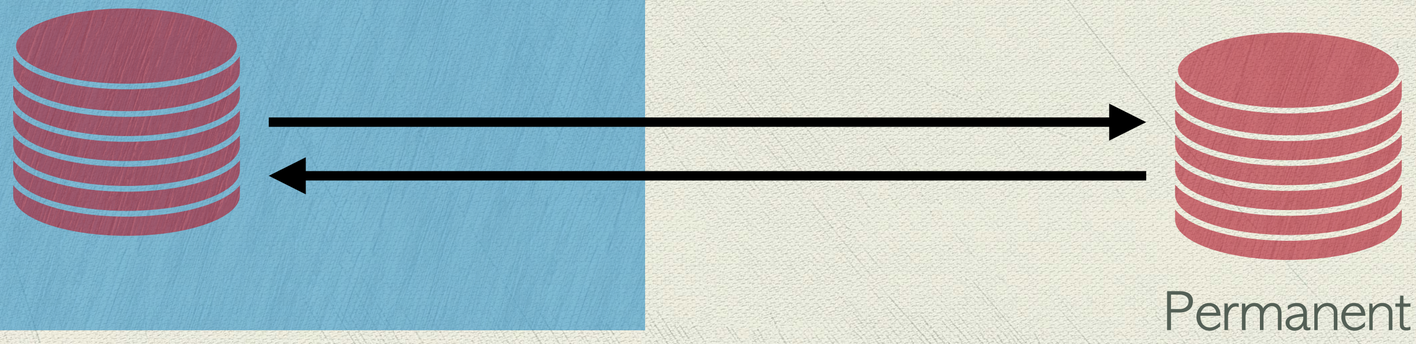
When delete container, data are deleted.

Volume

To keep data around when container is removed.
To share data between the host filesystem and the Docker container.
To share data among Docker containers.



docker run --rm -it -v $PWD:/build ubuntu:16.04



Develop Mode (3rd workshop)

Create and Run PHP Web Page

hello.php

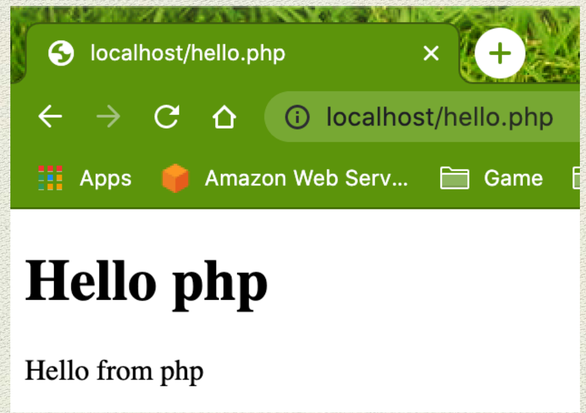
workshop_3 > src > hello.php

```php
<!DOCTYPE html>
<html>
<body>
<h1>Hello php</h1>
<?php echo "Hello from php"; ?>
</body>
</html>
```

Dockerfile

workshop_3 > Dockerfile > ...

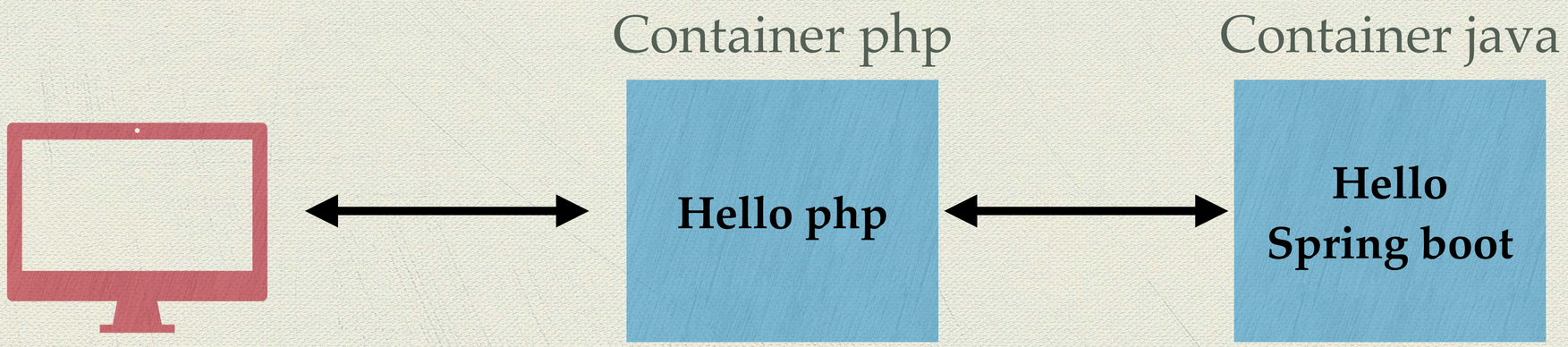
```
FROM php:7.2-apache
COPY src/ /var/www/html/
```

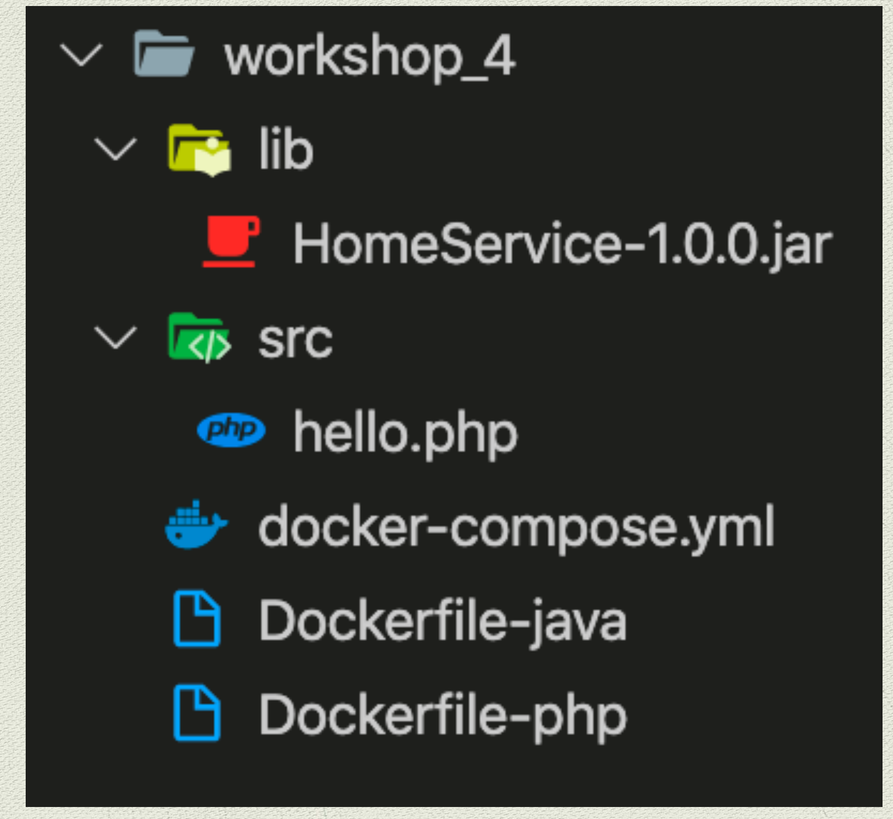



```
docker build -t ekapolw/hellp-php .
```

```
docker run -d -p 80:80 --name my-apache-php-app -v "$PWD":/var/www/html ekapolw/hellp-php
```

Hello Network (4th workshop)





workshop_4 > Dockerfile-java

```
1  FROM openjdk:8-jdk-alpine
2  VOLUME /tmp
3  ARG JAR_FILE=*.jar
4  COPY lib/${JAR_FILE} HomeService.jar
5  ENTRYPOINT [ "java", "-jar", "/HomeService.jar" ]
```

workshop_4 > Dockerfile-php

```
1  FROM php:7.2-apache
2  COPY src/ /var/www/html/
```

workshop_4 > docker-compose.yml

```yaml
version: "3"
services:

    java-container:
        build:
            context: .
            dockerfile: Dockerfile-java
        container_name: java-container
        ports:
         - 8080:8080
        networks:
         - workshop-network

    php-container:
        build:
            context: .
            dockerfile: Dockerfile-php
        container_name: php-container
        ports:
         - 80:80
        networks:
         - workshop-network
        depends_on:
         - java-container

networks:
    workshop-network:
        driver: bridge
```

Build container network

docker-compose.yml

```
docker-compose up -d
```

```
docker-compose down
```

Network Drivers

Bridge	Default network driver. Multiple containers are communicating with the same docker host.
Host	This driver removes the network isolation between docker containers and docker host.
Overlay	This driver enables swarm services to communicate with each other. Containers on different Docker Hosts could communicate.
None	This driver disables all the networking.
macvlan	This driver assigns mac address to containers to make them look like physical devices.

Save image into tar

```
Ekapols-MacBook-Pro:CreateImage ekapolwongnapapan$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello               latest              60cac4cae542        47 minutes ago      2.75kB
ubuntu              16.04               56bab49eef2e        14 hours ago        123MB
busybox             latest              020584afccce        3 weeks ago         1.22MB
Ekapols-MacBook-Pro:CreateImage ekapolwongnapapan$ docker save -o hello.tar hello
Ekapols-MacBook-Pro:CreateImage ekapolwongnapapan$ ls
Dockerfile      hello           hello.c         hello.tar
Ekapols-MacBook-Pro:CreateImage ekapolwongnapapan$
```

Load tar into image

```
Ekapols-MacBook-Pro:CreateImage ekapolwongnapapan$ ls
Dockerfile      hello           hello.c         hello.tar
Ekapols-MacBook-Pro:CreateImage ekapolwongnapapan$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              16.04               56bab49eef2e        14 hours ago        123MB
busybox             latest              020584afccce        3 weeks ago         1.22MB
Ekapols-MacBook-Pro:CreateImage ekapolwongnapapan$ docker load -i hello.tar
c93b9ce75bcd: Loading layer [==================================================>]  4.608kB/4.608kB
Loaded image: hello:latest
Ekapols-MacBook-Pro:CreateImage ekapolwongnapapan$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello               latest              60cac4cae542        52 minutes ago      2.75kB
ubuntu              16.04               56bab49eef2e        14 hours ago        123MB
busybox             latest              020584afccce        3 weeks ago         1.22MB
Ekapols-MacBook-Pro:CreateImage ekapolwongnapapan$
```

Clean Up Containers

```
Ekapols-MBP:~ ekapolwongnapapan$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
0cbeb8e612b4        busybox             "sh"                     14 seconds ago      Exited (0) 6 seconds ago
58621879e96a        busybox             "echo 'Hello from bu…"   30 seconds ago      Exited (0) 29 seconds ago
e46f8958a6d9        busybox             "sh"                     59 seconds ago      Exited (0) 58 seconds ago
```

```
Ekapols-MBP:~ ekapolwongnapapan$ docker container rm 0cbeb8e612b4
0cbeb8e612b4
```

Container id

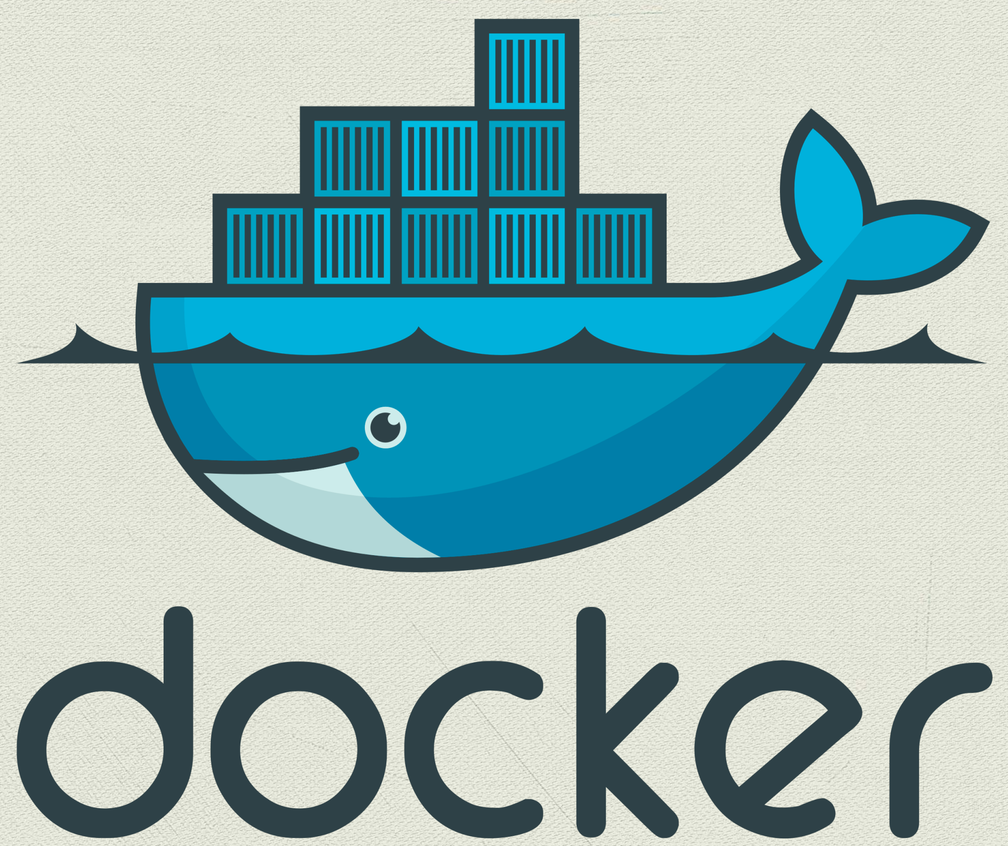
```
Ekapols-MBP:~ ekapolwongnapapan$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
58621879e96a        busybox             "echo 'Hello from bu…"   About a minute ago  Exited (0) About a minute ago
e46f8958a6d9        busybox             "sh"                     About a minute ago  Exited (0) About a minute ago
```

```
Ekapols-MBP:~ ekapolwongnapapan$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
58621879e96a5ad452dd71f3bb53cf33a64fc54e94e9836dc7d5a004a6dd424e
e46f8958a6d979647da12313e1c50db67e04d158591be05e70ce47f713f5c791

Total reclaimed space: 0B
```

Remove all
not running containers

```
Ekapols-MBP:~ ekapolwongnapapan$ docker ps -a
CONTAINER ID        IMAGE               COMMAND
Ekapols-MBP:~ ekapolwongnapapan$
```

Thanks
Enjoy Docker



Create TechnicalLabService image

Create image for TechnicalLabService.jar (Spring Boot)

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ARG JAR_FILE=*.jar
COPY ${JAR_FILE} TechnicalLabService.jar
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/TechnicalLabService.jar"]
```

```
docker build -t springio/technical-lab-service .
```

```
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
springio/technical-lab-service      latest              e5eff8a3e7a6        8 minutes ago       122MB
```

Run TechnicalLabService Container

```
docker run -p 8090:8090 springio/technical-lab-service
```

Host port

Container port

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.9.RELEASE)

2019-11-27 15:55:33.523  INFO 1 --- [           main] c.t.t.TechnicalLabService                : Starting TechnicalLabService v0.0.1-SNAPSHOT on 00a7b4050a8d with PID 1 (/Technic
alLabService.jar started by root in /)
2019-11-27 15:55:33.533  INFO 1 --- [           main] c.t.t.TechnicalLabService                : No active profile set, falling back to default profiles: default
2019-11-27 15:55:37.077  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8090 (http)
2019-11-27 15:55:37.194  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2019-11-27 15:55:37.195  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.26]
2019-11-27 15:55:37.454  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2019-11-27 15:55:37.455  INFO 1 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 3672 ms
2019-11-27 15:55:38.115  INFO 1 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2019-11-27 15:55:38.781  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8090 (http) with context path ''
2019-11-27 15:55:38.792  INFO 1 --- [           main] c.t.t.TechnicalLabService                : Started TechnicalLabService in 6.772 seconds (JVM running for 7.988)
```