Container

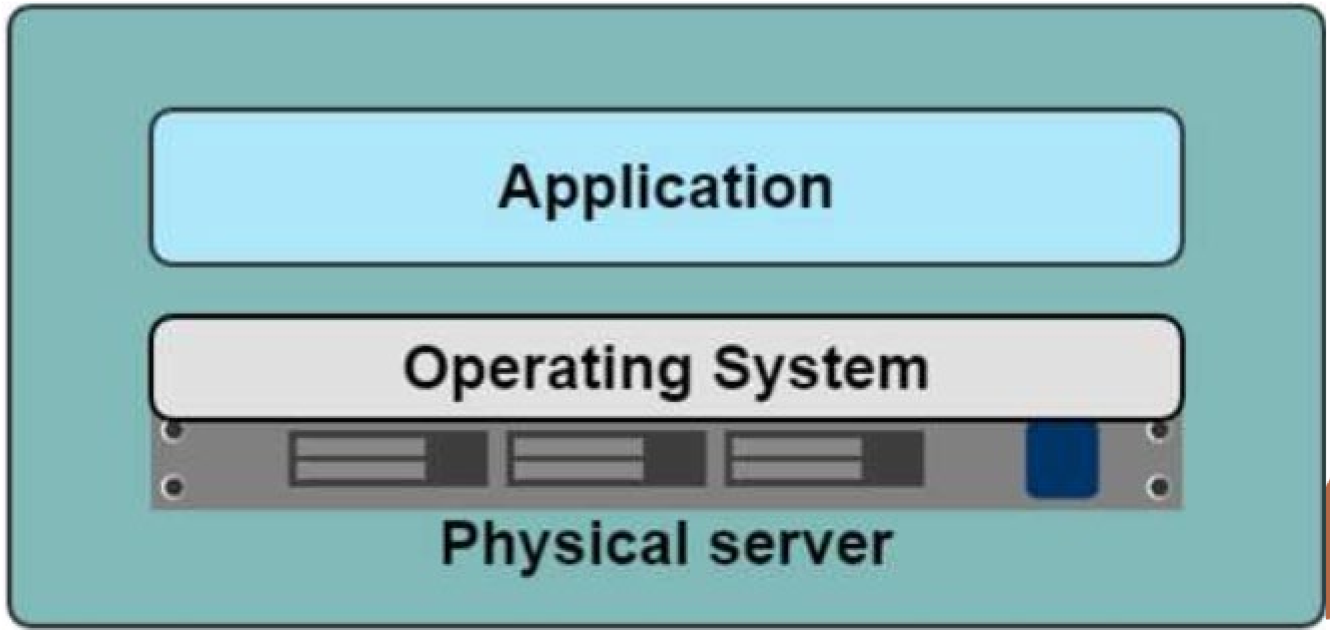
SE234 Advance Software Development

CAMT
College of Arts, Media and Technology
Chiang Mai University

Q1. What is Virtual Machine

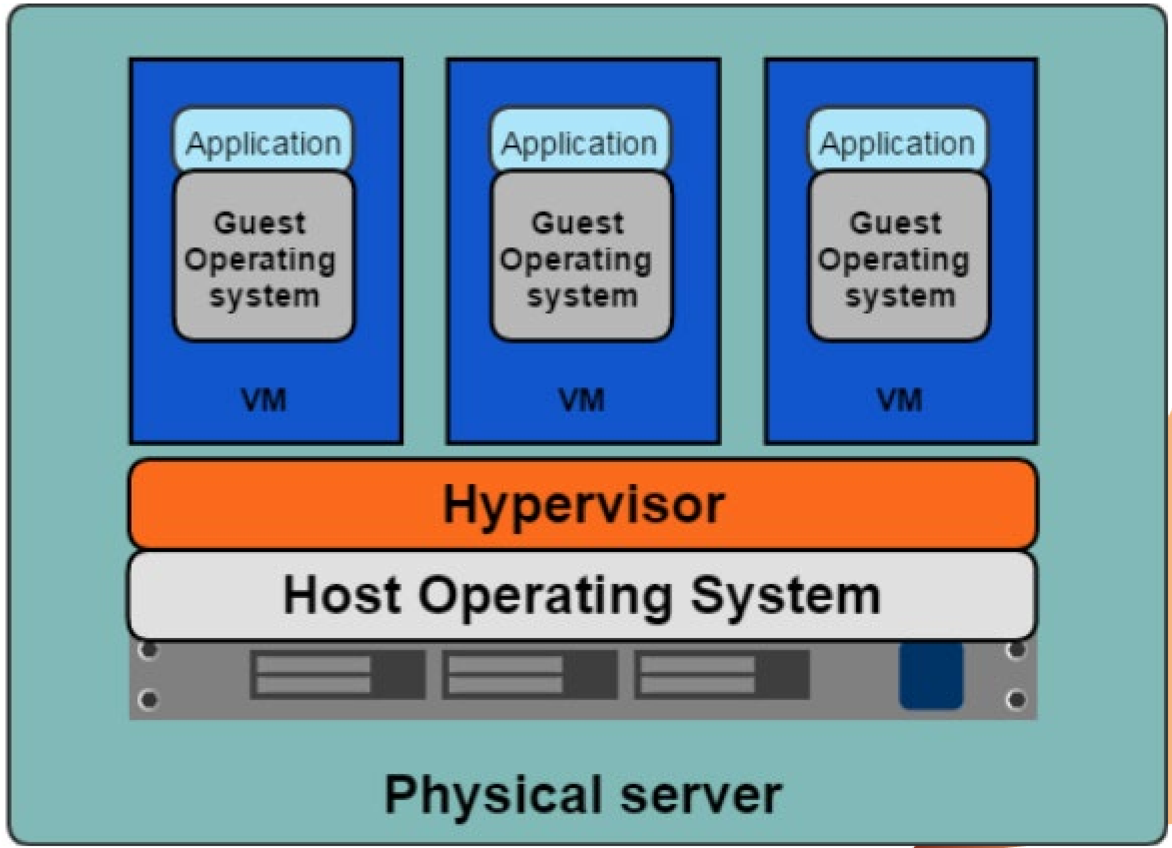
What is container?

- The legacy system limitation
  - Slow deployment times
  - Huge costs
  - Wasted resources
  - Difficult to scale
  - Difficult to migrate
  - Vendor lock in



What we have

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



Benefit 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®

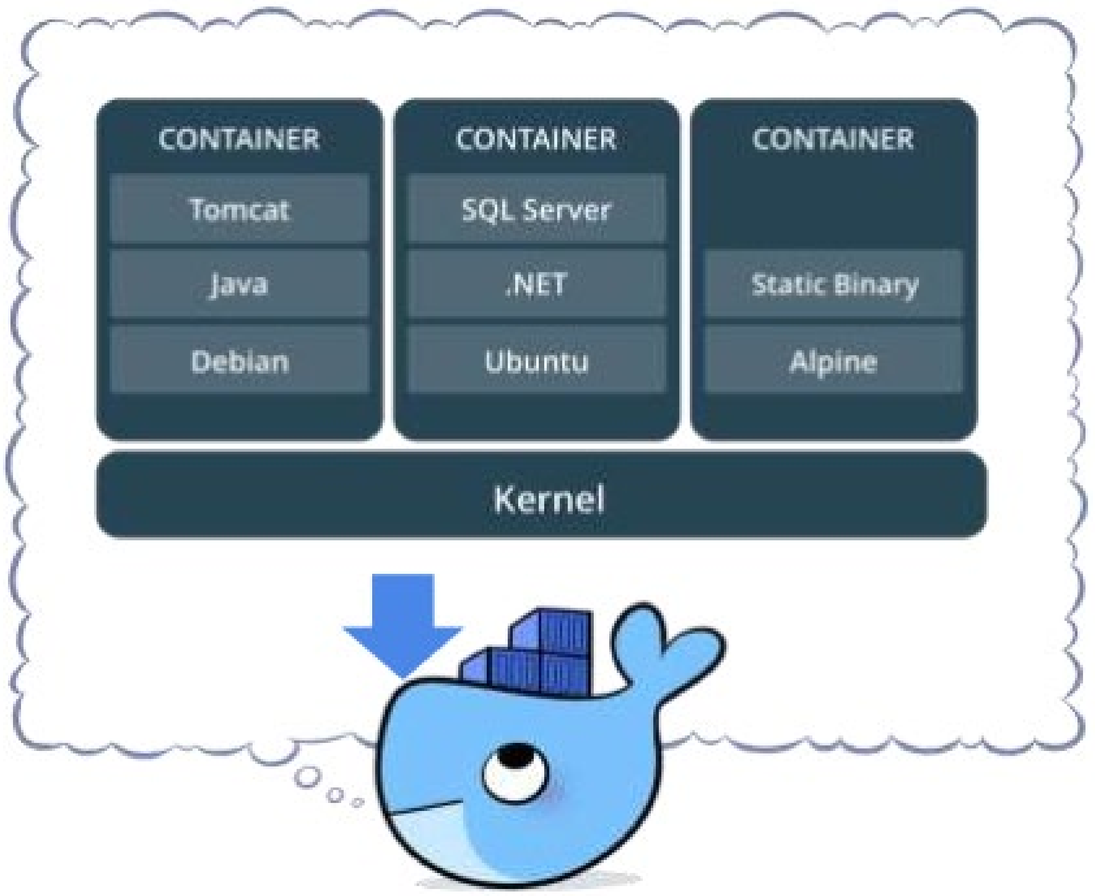
CAMT
College of Arts, Media and Technology
Chiang Mai University

Limitation of VMs

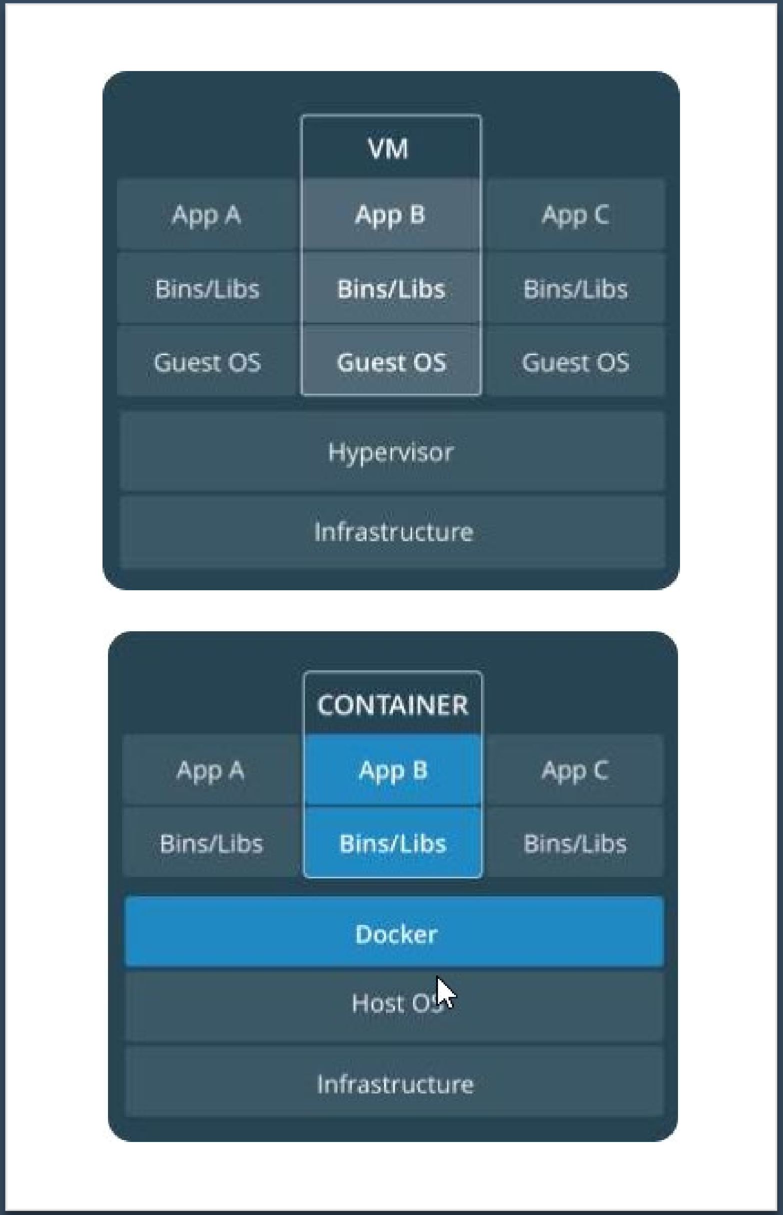
- Each VM stills requires
  - CPU allocation
  - Storage
  - RAM
- An entire guest operating system
- The more VMs you run, the more resources you need
- Guest OS means wasted resources
- Application portability not guaranteed

What is a Container

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

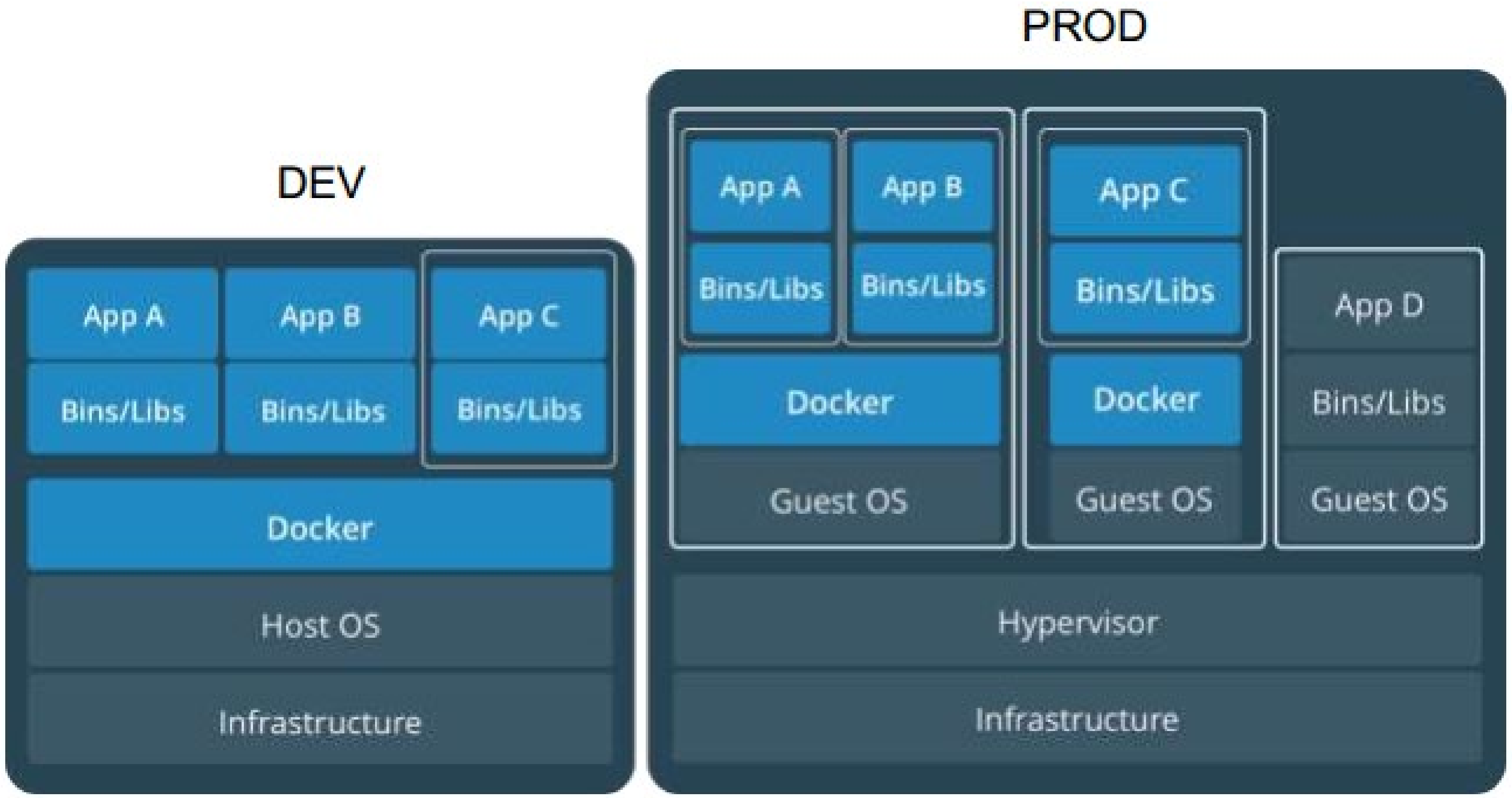


Comparing Containers and VMs



- Containers are app level construct
- VMs are infrastructure level construct to turn on machine into many servers

Containers and VMs together



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

Key Benefits of Docker Containers

Speed

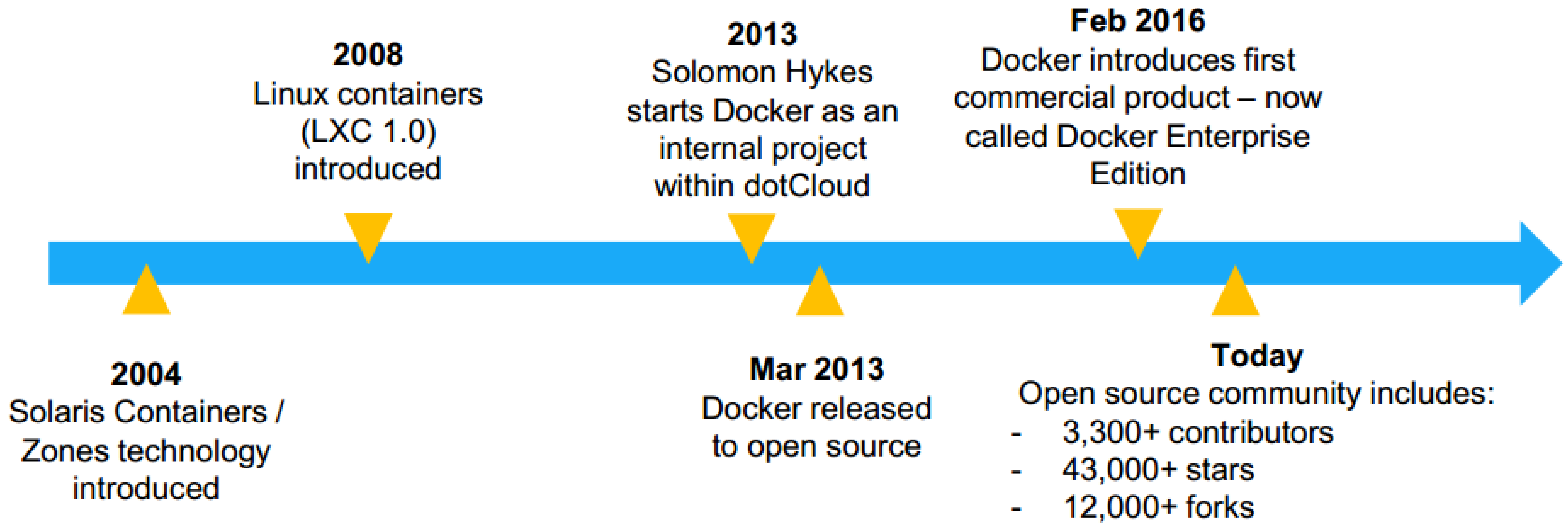
- No OS to boot = applications online in seconds

Portability

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

Efficiency

- Less OS overhead
- Improved VM density



2004
Solaris Containers / Zones technology introduced

2008
Linux containers (LXC 1.0) introduced

2013
Solomon Hykes starts Docker as an internal project within dotCloud

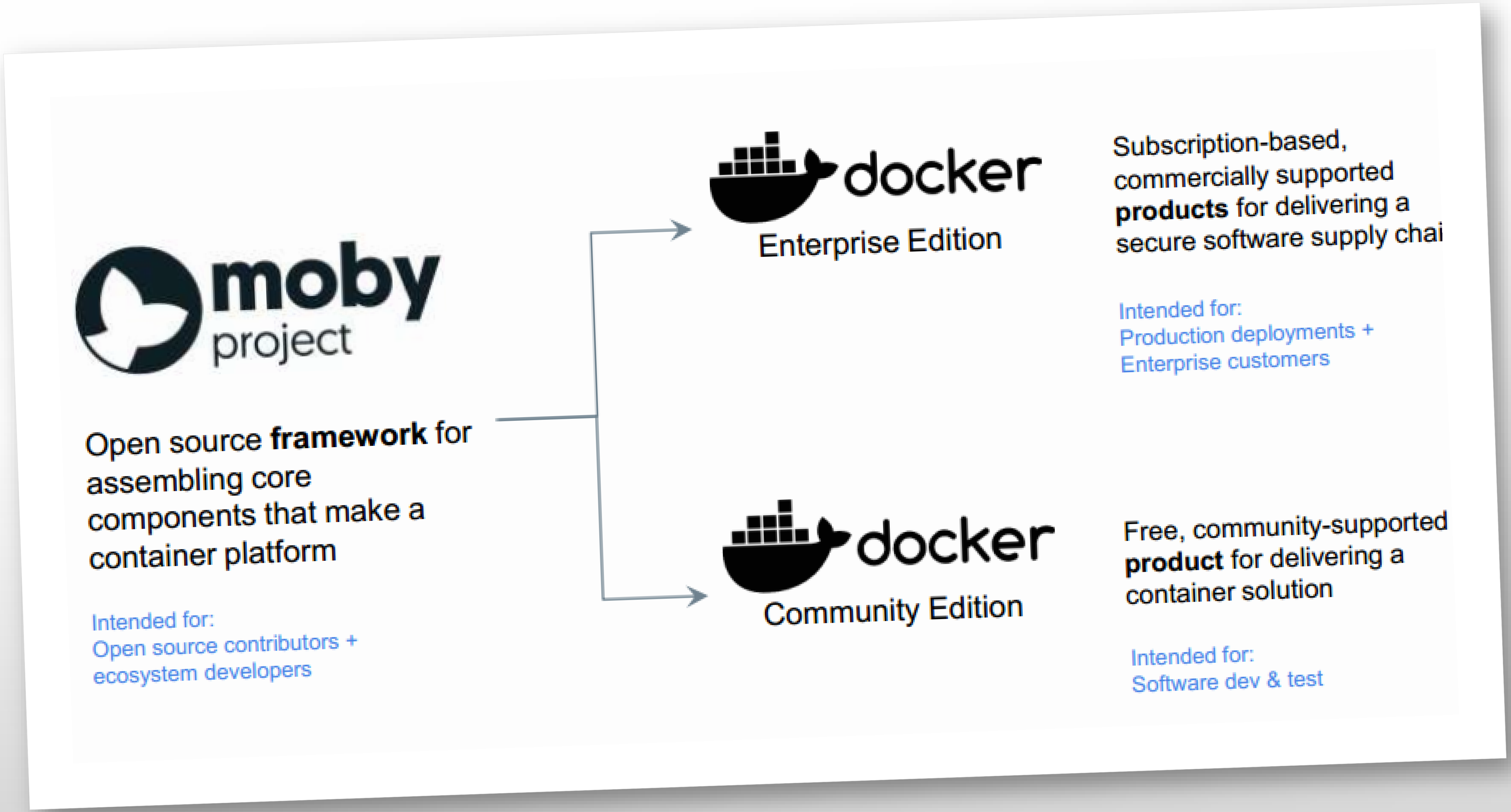
Mar 2013
Docker released to open source

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

Docker

History



moby project

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

Intended for:
Open source contributors + ecosystem developers

docker Enterprise Edition

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

Intended for:
Production deployments + Enterprise customers

docker Community Edition

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

Intended for:
Software dev & test

Docker Basics

- Image
    - The basis of 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

Docker Basics

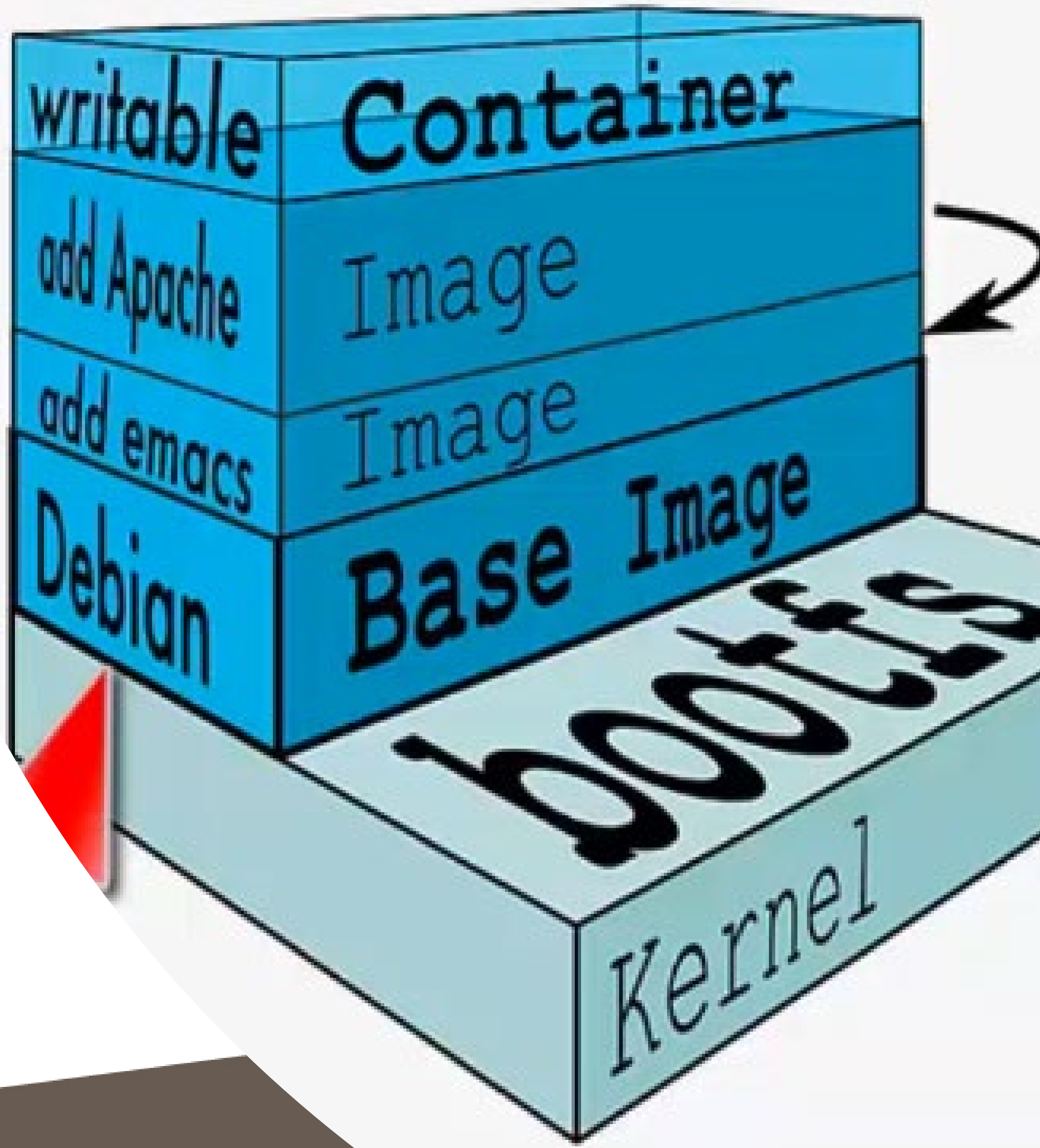
- Registry
  - Store, distributes and manages Docker images
- Control Pane
  - Management plane for container and cluster orchestration

Docker image

- Docker images are read-only templates from which Docker containers are launched.
- Each image consists of a series of layers.
- Every image starts from a base image.
    - E.g ubuntu, Apache.
- Docker images are then built from these base images using a simple, descriptive set of steps we call instructions.
- Each instruction creates a new **layer** in our image.

Docker image

- A Layer is just another image
- Docker uses a copy on write system

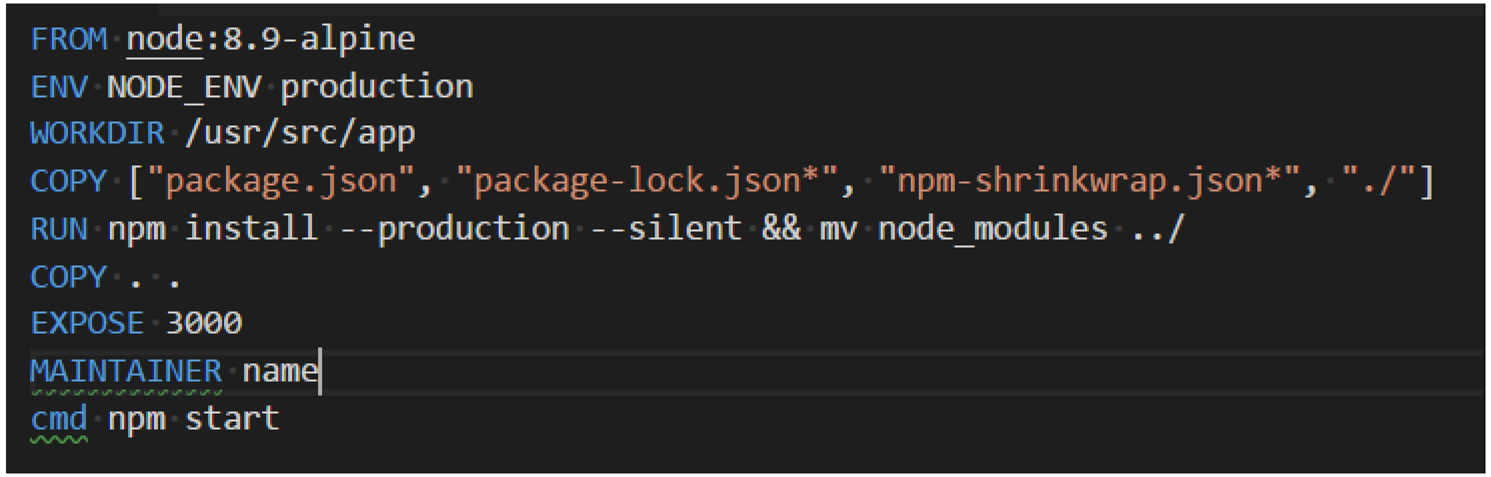


Docker image - instructions

- Instructions example:
  - *Run* a command.
  - *Add* a file or directory.
  - Create an environment variable.
  - What process to run when launching a container from this image.
- Dockerfile

CAMT
College of Arts, Media and Technology
Chiang Mai University

Dockerfile example



```
FROM node:8.9-alpine
ENV NODE_ENV production
WORKDIR /usr/src/app
COPY ["package.json", "package-lock.json*", "npm-shrinkwrap.json*", "./"]
RUN npm install --production --silent && mv node_modules ../
COPY . .
EXPOSE 3000
MAINTAINER name
cmd npm start
```

Docker Container

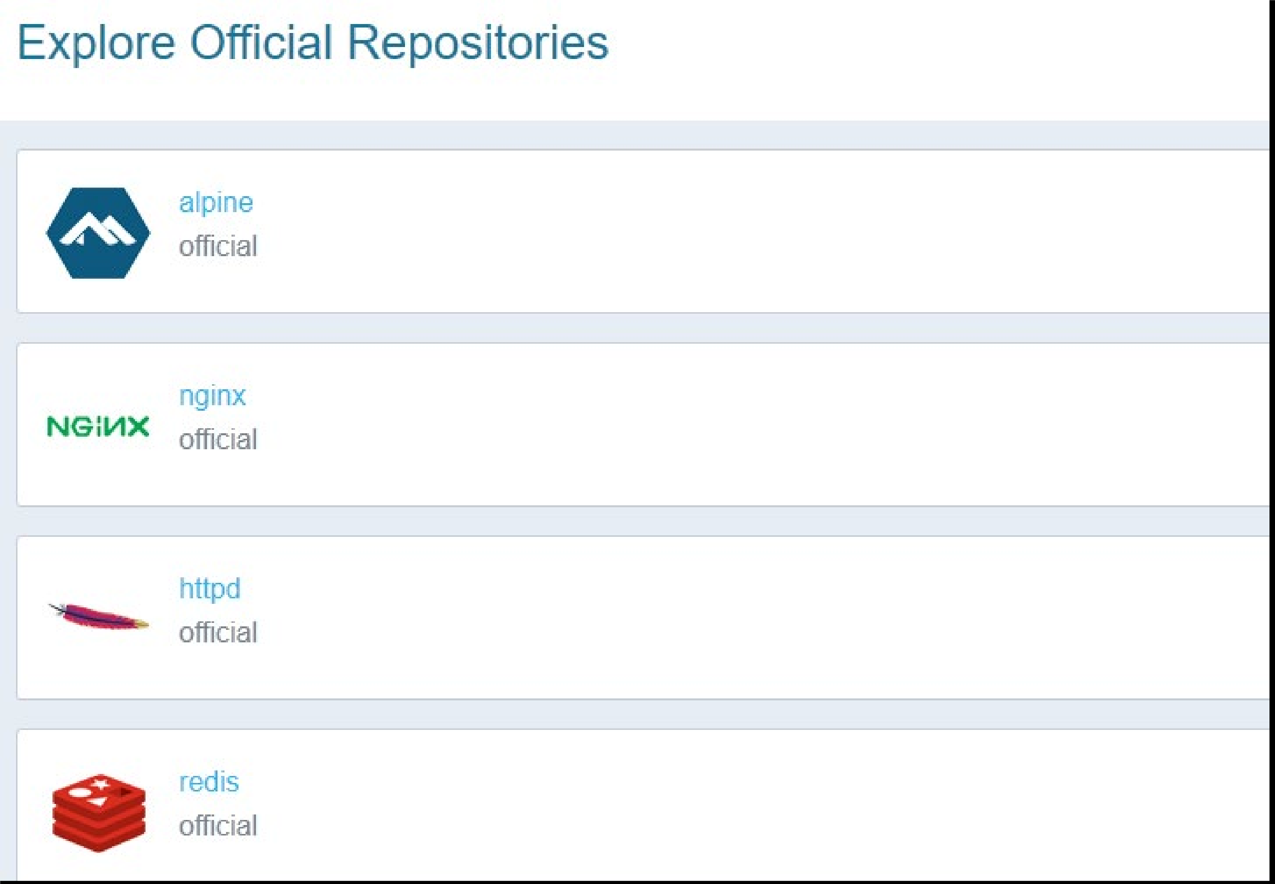
- Container is built from an image
- A container consists of
  - operating system
  - user-added files
  - meta-data
- That image tells Docker what the container holds, what process to run when the container is launched, and a variety of other configuration data

Docker Container

- The Docker image is read-only.
- When Docker runs a container from an image, it adds a read-write layer on top of the image (using a union file system as we saw earlier) in which your application can then run
    - Writeable layer
    - All changes are made at the writeable layer
- The other configurations can be set when running the docker

Docker Engine

- The Computer with the Docker_Machine runs
- The docker_machine run the Docker daemon
- All Containers run on the Docker Engine

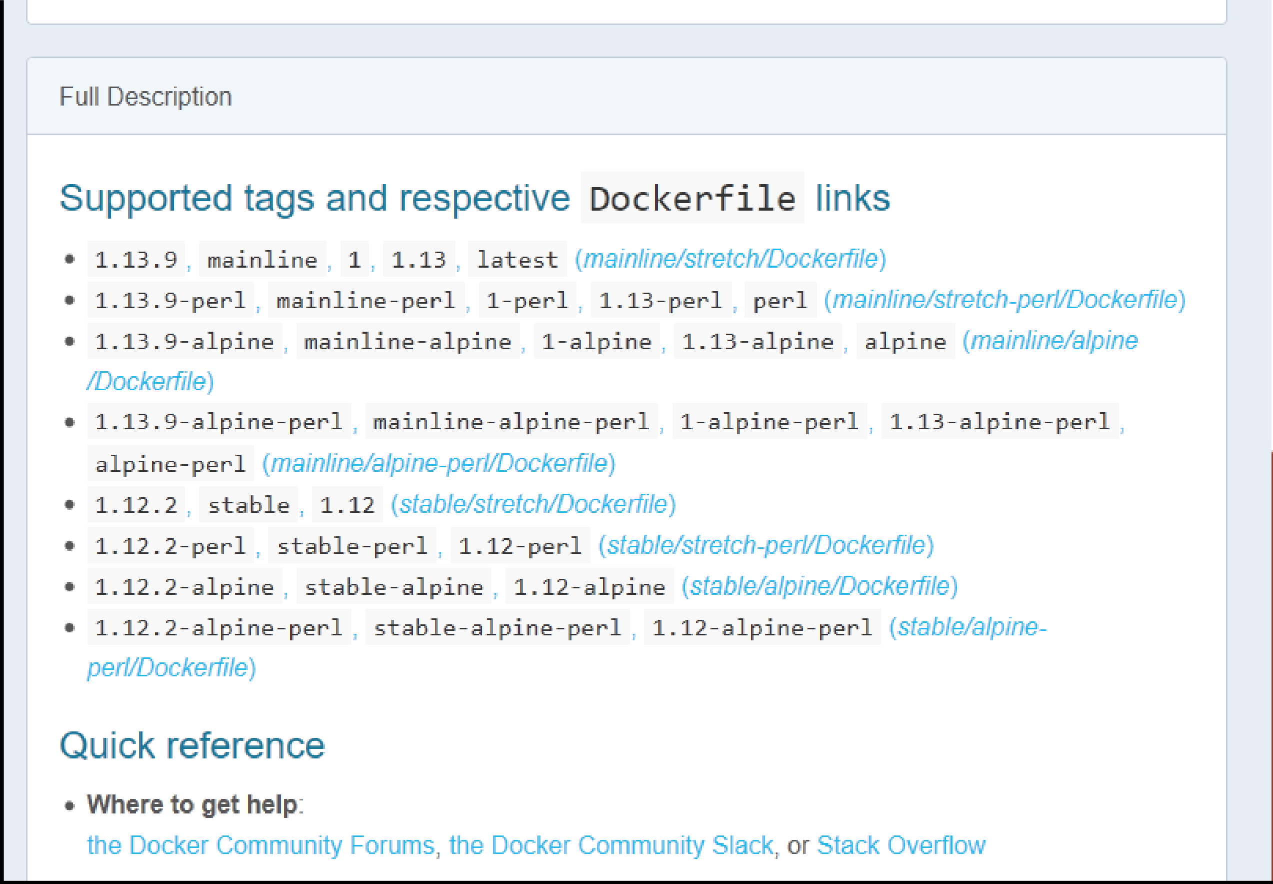

Explore Official Repositories

alpine
official

nginx
official

httpd
official

redis
official


Full Description

Supported tags and respective `Dockerfile` links

- `1.13.9`, `mainline`, `1`, `1.13`, `latest` (*mainline/stretch/Dockerfile*)
- `1.13.9-perl`, `mainline-perl`, `1-perl`, `1.13-perl`, `perl` (*mainline/stretch-perl/Dockerfile*)
- `1.13.9-alpine`, `mainline-alpine`, `1-alpine`, `1.13-alpine`, `alpine` (*mainline/alpine/Dockerfile*)
- `1.13.9-alpine-perl`, `mainline-alpine-perl`, `1-alpine-perl`, `1.13-alpine-perl`, `alpine-perl` (*mainline/alpine-perl/Dockerfile*)
- `1.12.2`, `stable`, `1.12` (*stable/stretch/Dockerfile*)
- `1.12.2-perl`, `stable-perl`, `1.12-perl` (*stable/stretch-perl/Dockerfile*)
- `1.12.2-alpine`, `stable-alpine`, `1.12-alpine` (*stable/alpine/Dockerfile*)
- `1.12.2-alpine-perl`, `stable-alpine-perl`, `1.12-alpine-perl` (*stable/alpine-perl/Dockerfile*)

Quick reference

- **Where to get help**:
  the Docker Community Forums, the Docker Community Slack, or Stack Overflow

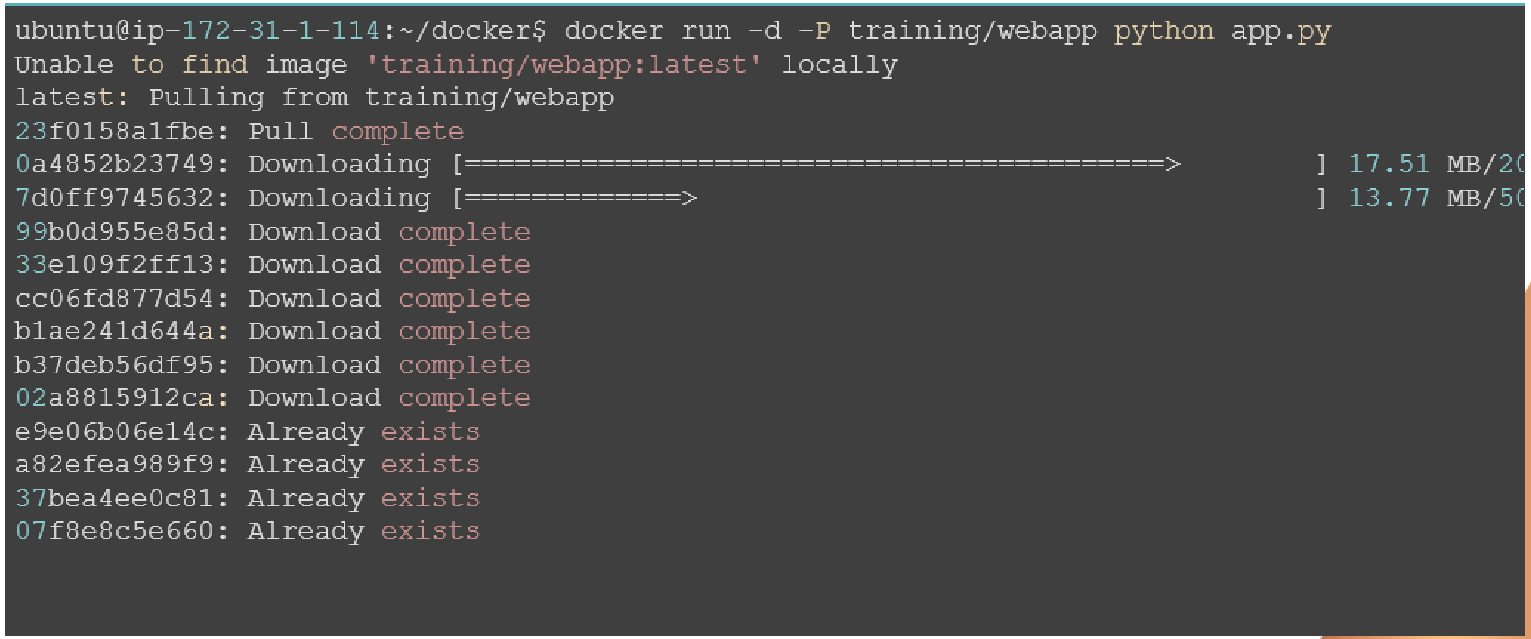
Registry

Where we can find the images

CAMT
College of Arts, Media and Technology
Chiang Mai University

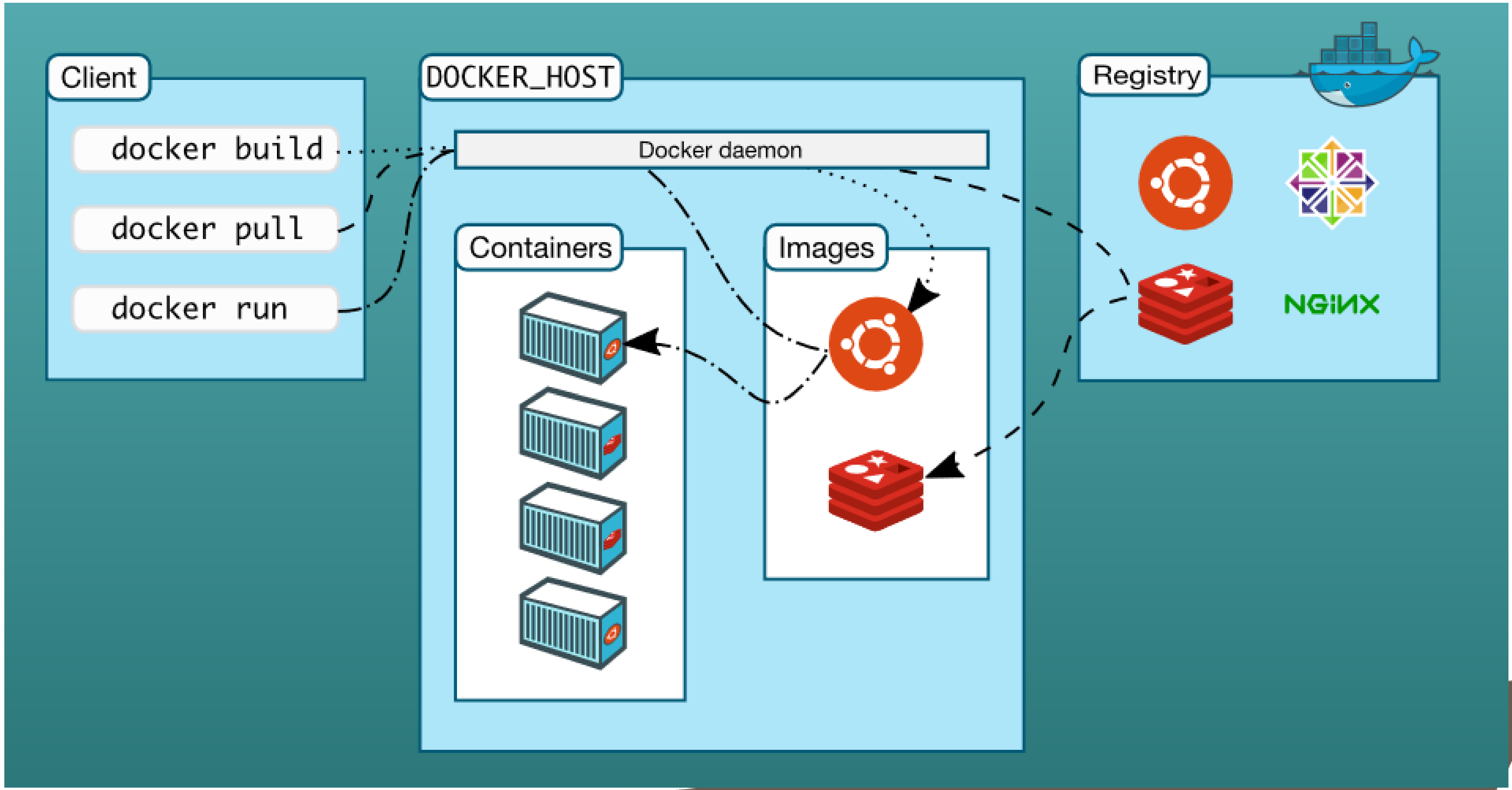
Docker Images Registry

- Images are stored locally, but can be pulled from an Image registry
- The Registry is a server that stores and lets you distribute Docker images

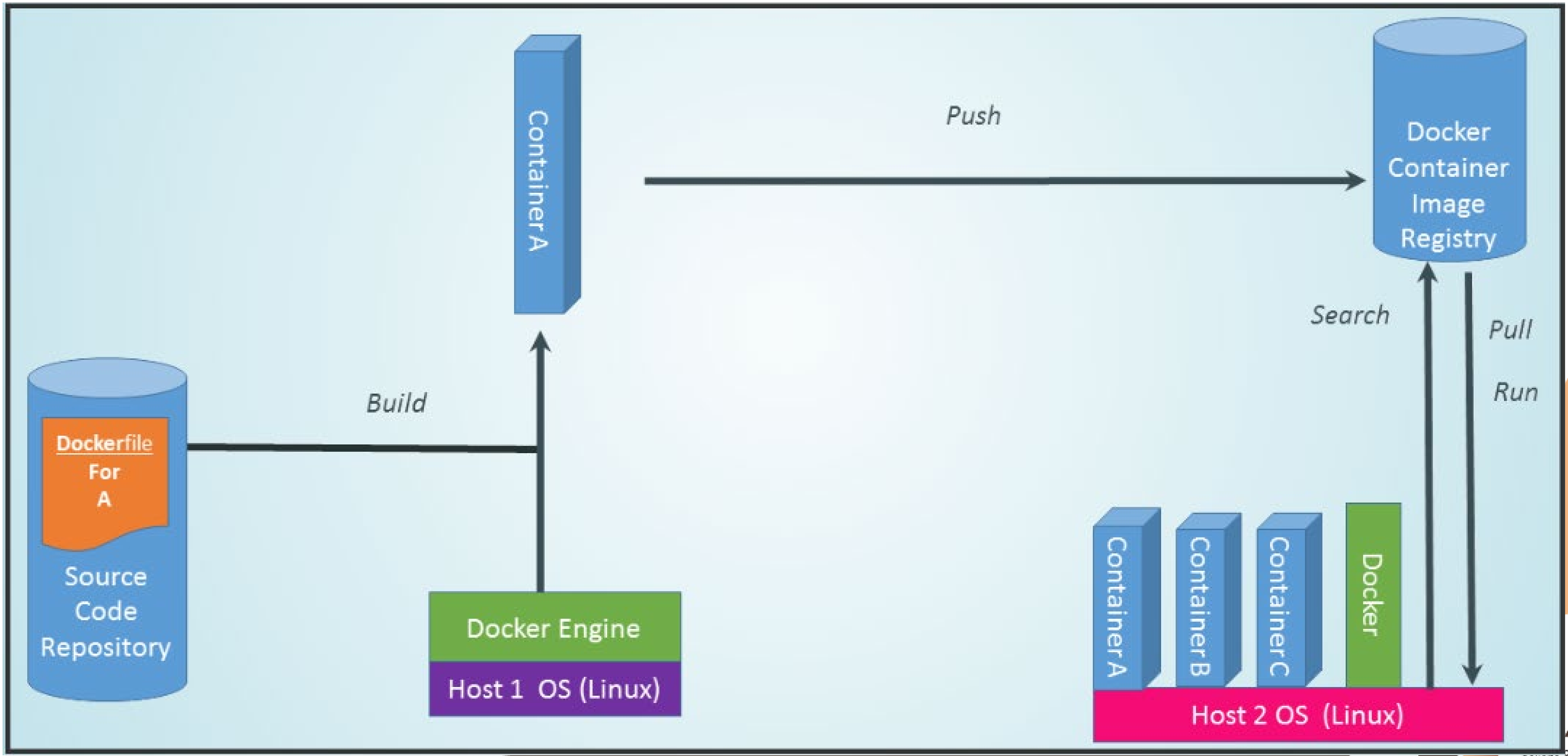


```
ubuntu@ip-172-31-1-114:~/docker$ docker run -d -P training/webapp python app.py
Unable to find image 'training/webapp:latest' locally
latest: Pulling from training/webapp
23f0158a1fbe: Pull complete
0a4852b23749: Downloading [===========================================>        ] 17.51 MB/20
7d0ff9745632: Downloading [=============>                                      ] 13.77 MB/50
99b0d955e85d: Download complete
33e109f2ff13: Download complete
cc06fd877d54: Download complete
b1ae241d644a: Download complete
b37deb56df95: Download complete
02a8815912ca: Download complete
e9e06b06e14c: Already exists
a82efea989f9: Already exists
37bea4ee0c81: Already exists
07f8e8c5e660: Already exists
```

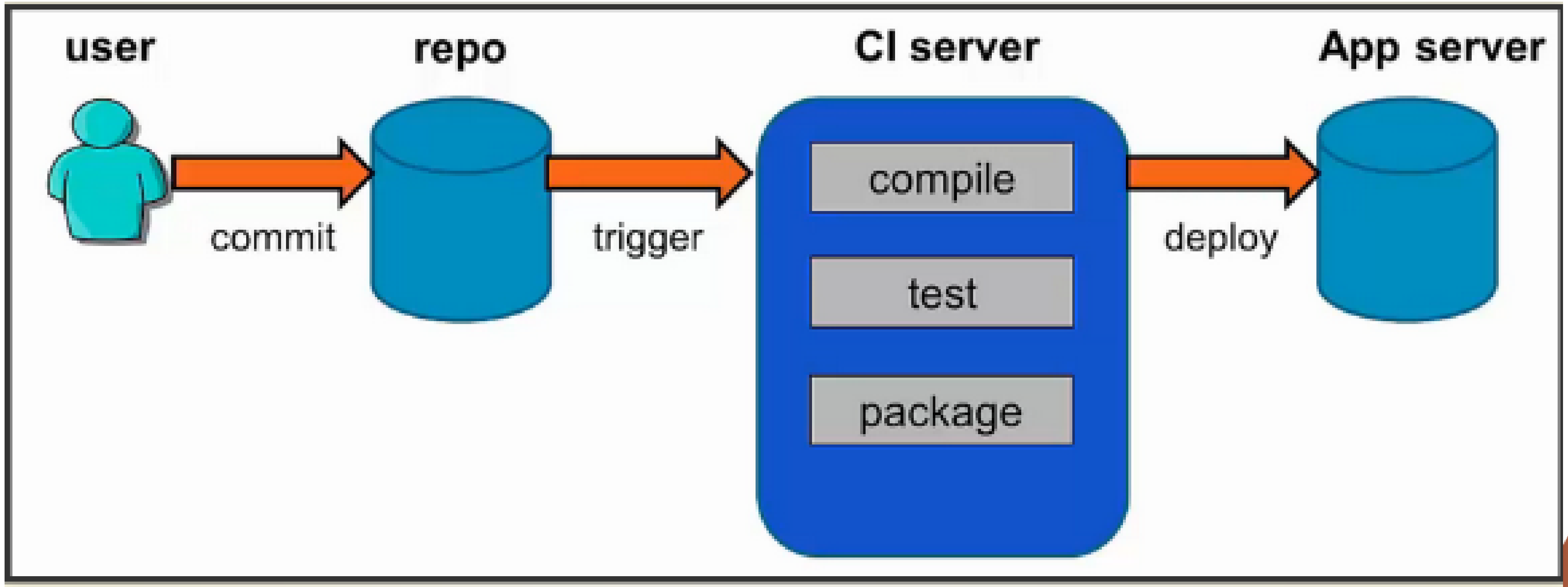
Docker architectures



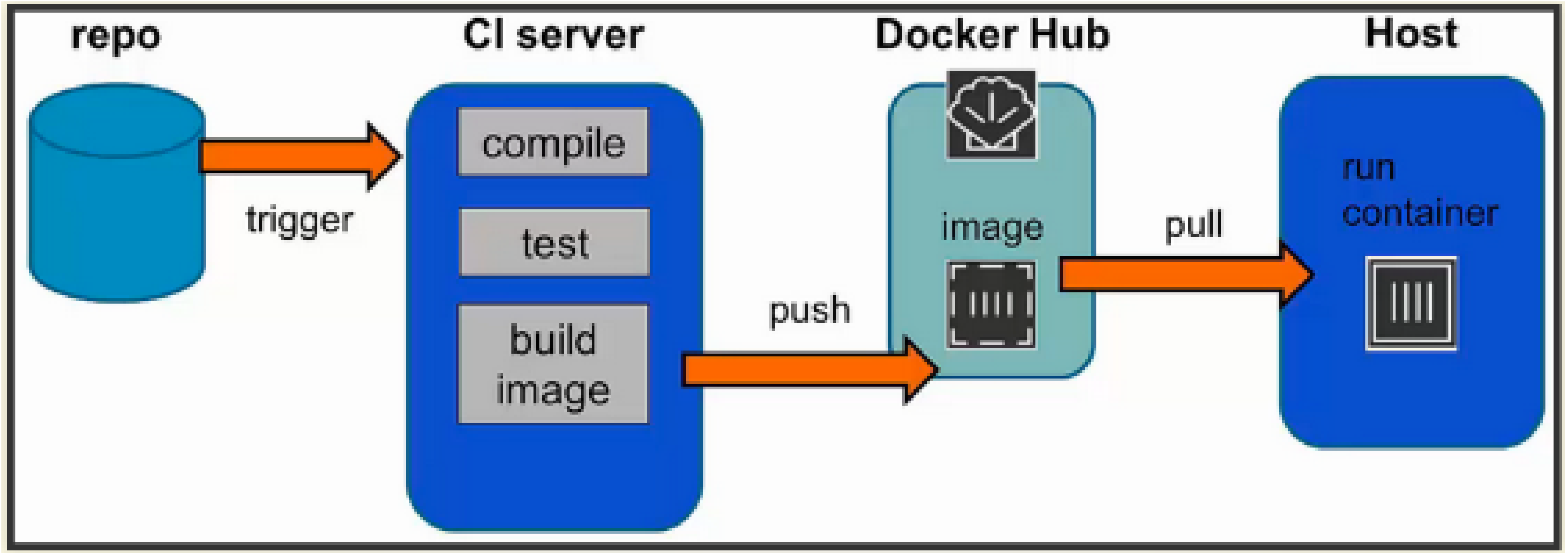
Basic Docker system



Traditional Continuous Integration



Using Docker in CI



Configuration files

- Help to config dockers in a different situation
- Dockerfile
- docker-compose

Q & A

There are no stupid questions, but questions to be easily misunderstood.