Introduction to Docker

Tejas Parikh (t.parikh@northeastern.edu)
Spring 2018

CSYE 6225
Northeastern University
https://spring2018.csye6225.com

Motivation for Containerization

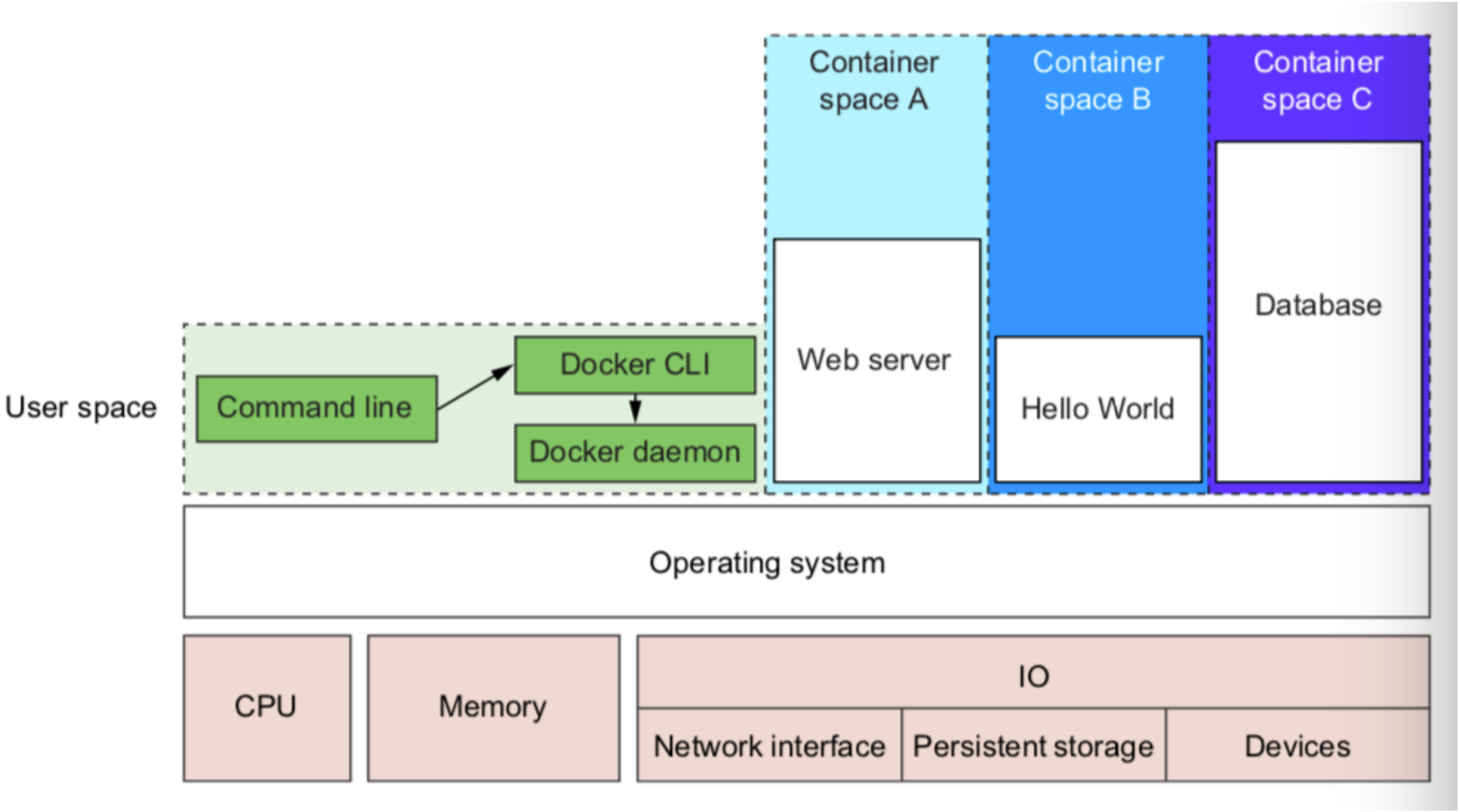
- Ease of Deployment - Write once run anywhere.
- Isolation – Reduce interference between applications.
- Efficiency – Better resource utilization.
- Better packaging – Bundling application software and required OS filesystems together in a single standardized image format.
- Using packaged artifacts to test and deliver the exact same artifact to all systems in all environments.

Containers != Virtualization

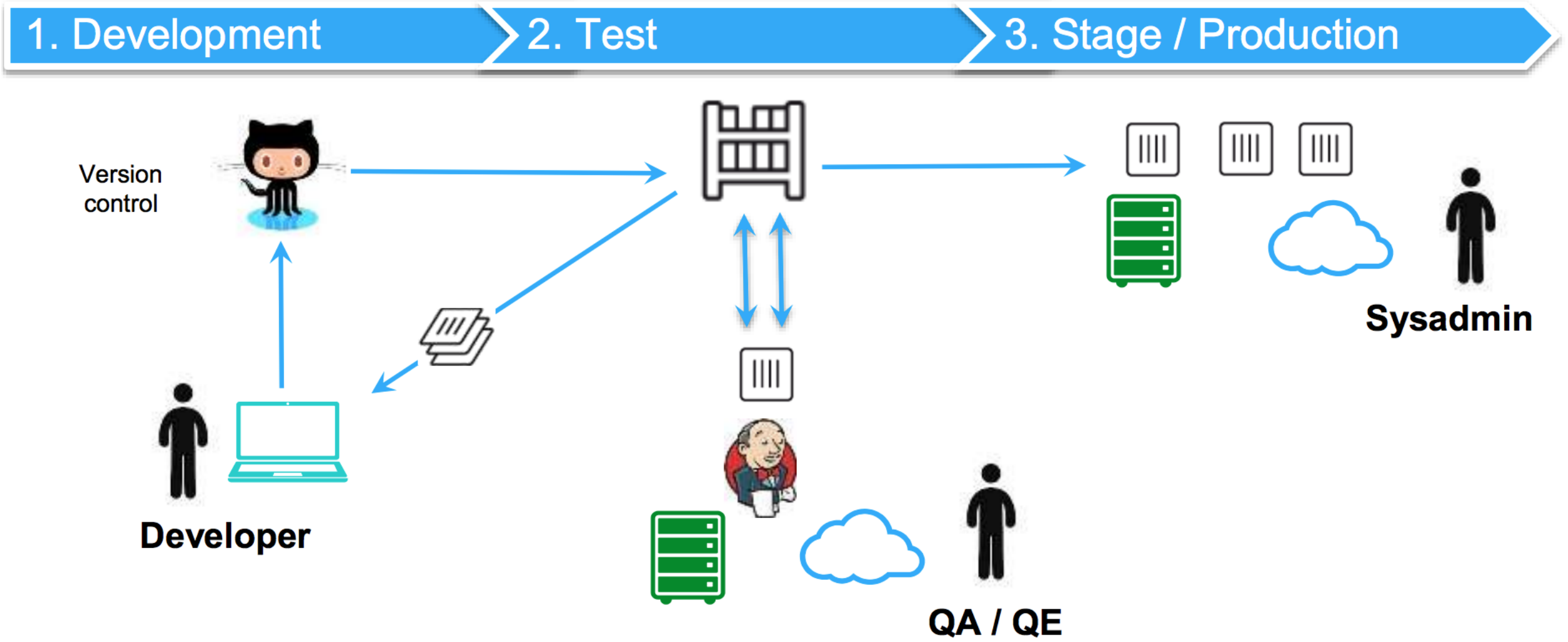
- Docker containers are not virtual machines but a very lightweight wrappers around a single Unix process.

Containers are Lightweight

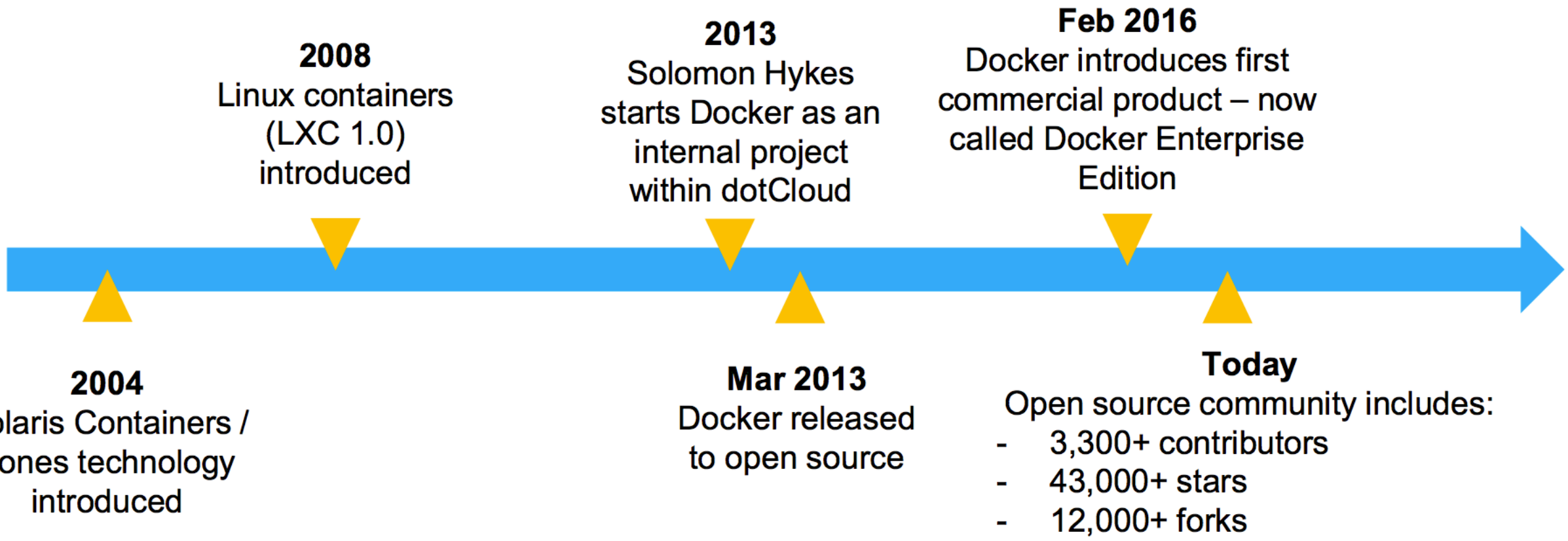
Containers takes very little space.


Container space A
Container space B
Container space C
Database
Web server
Hello World
User space
Command line
Docker CLI
Docker daemon
Operating system
CPU
Memory
IO
Network interface
Persistent storage
Devices

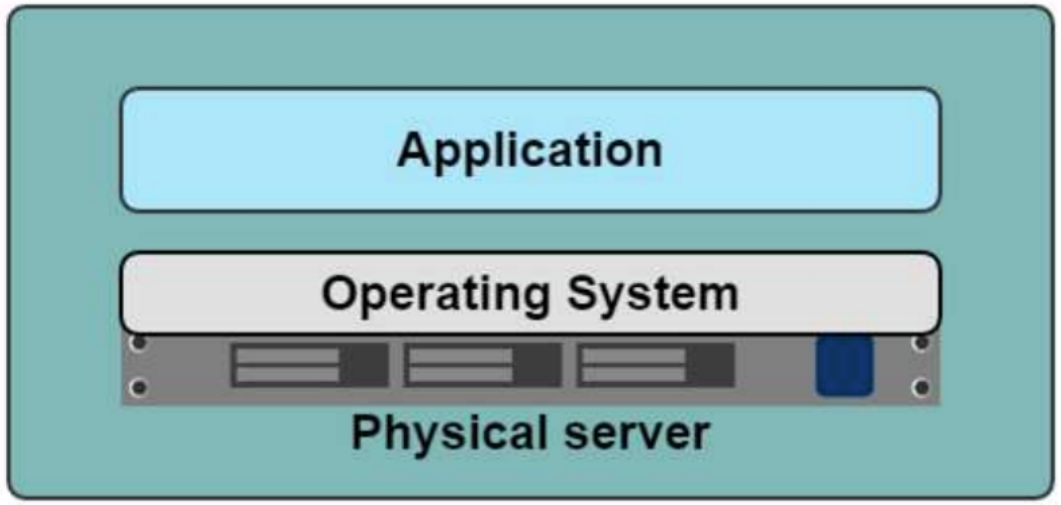
Continuous Integration and Delivery



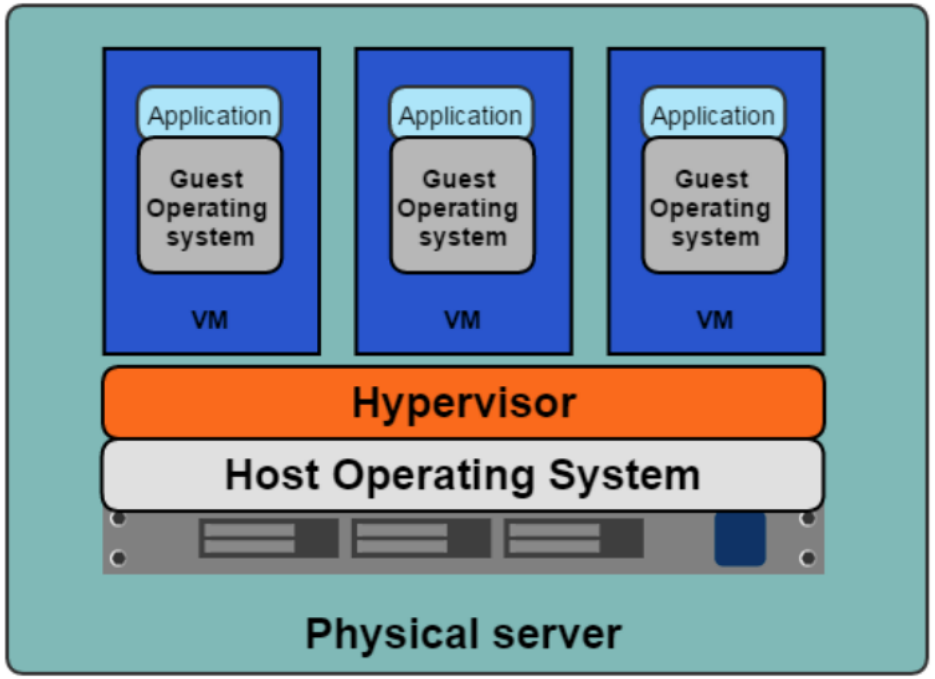
History of Docker



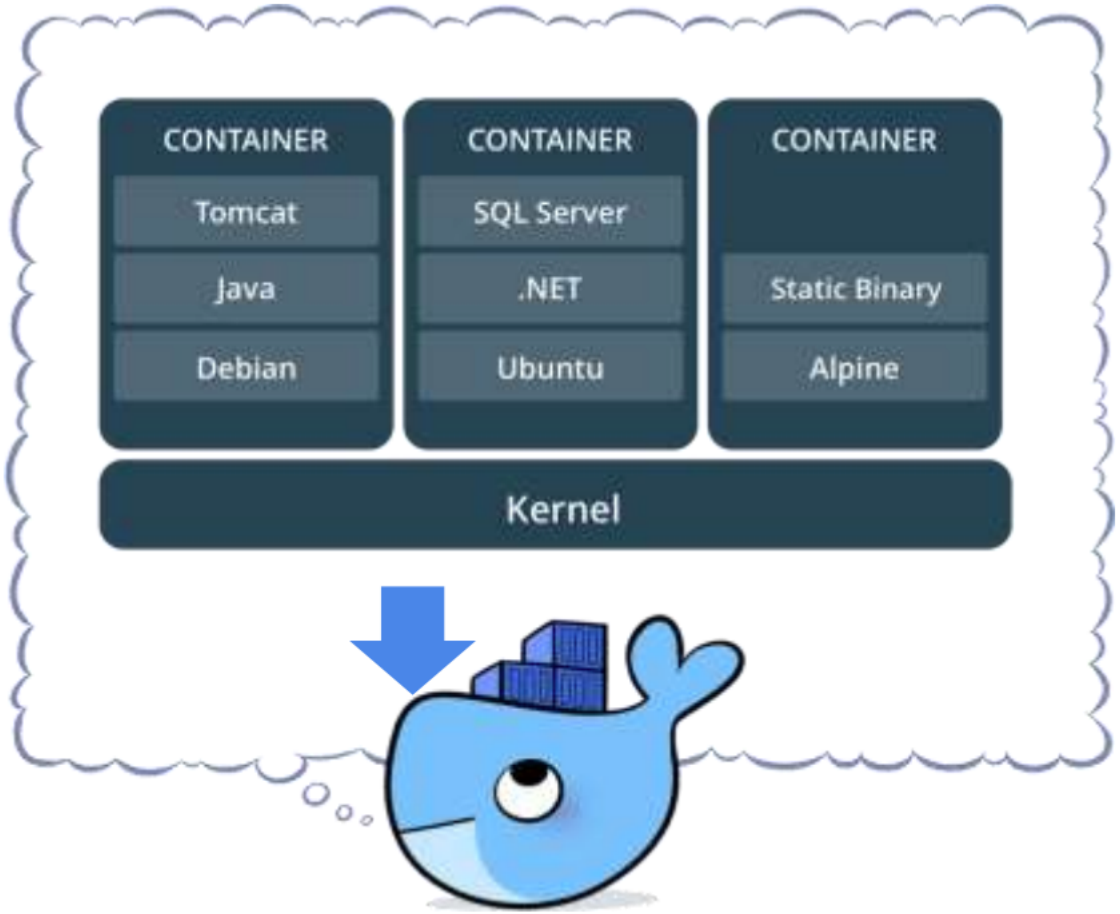
One Application on One Physical Server



Virtual Machines

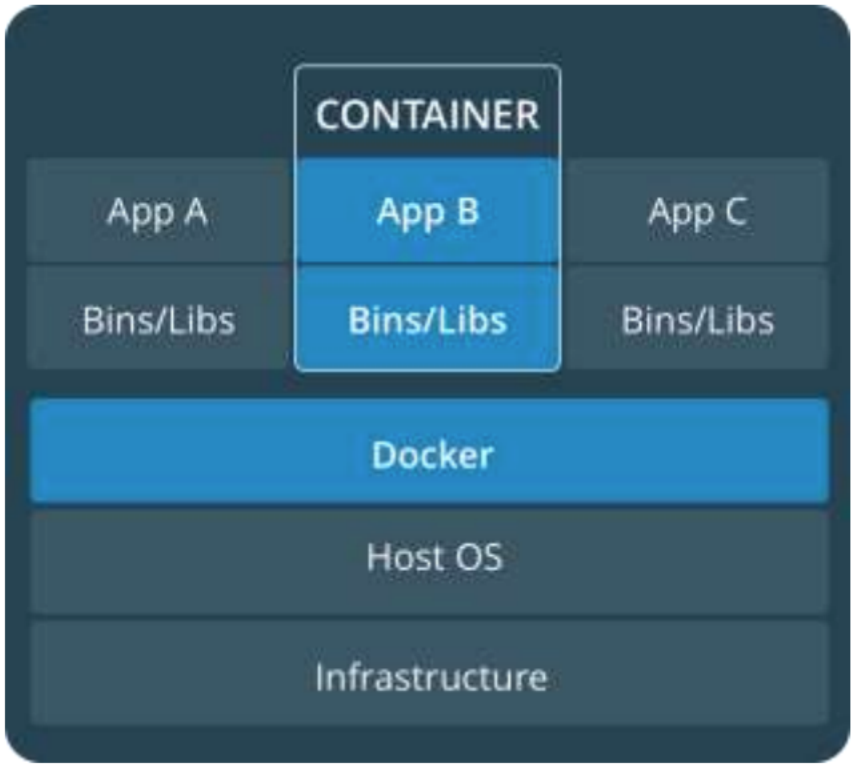


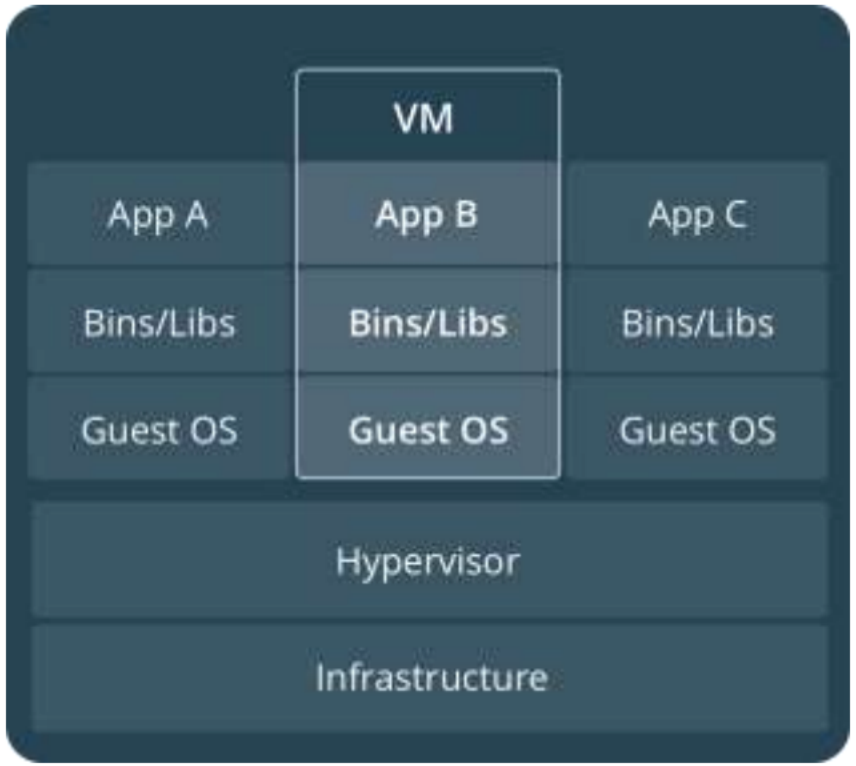
What is a container?



- 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

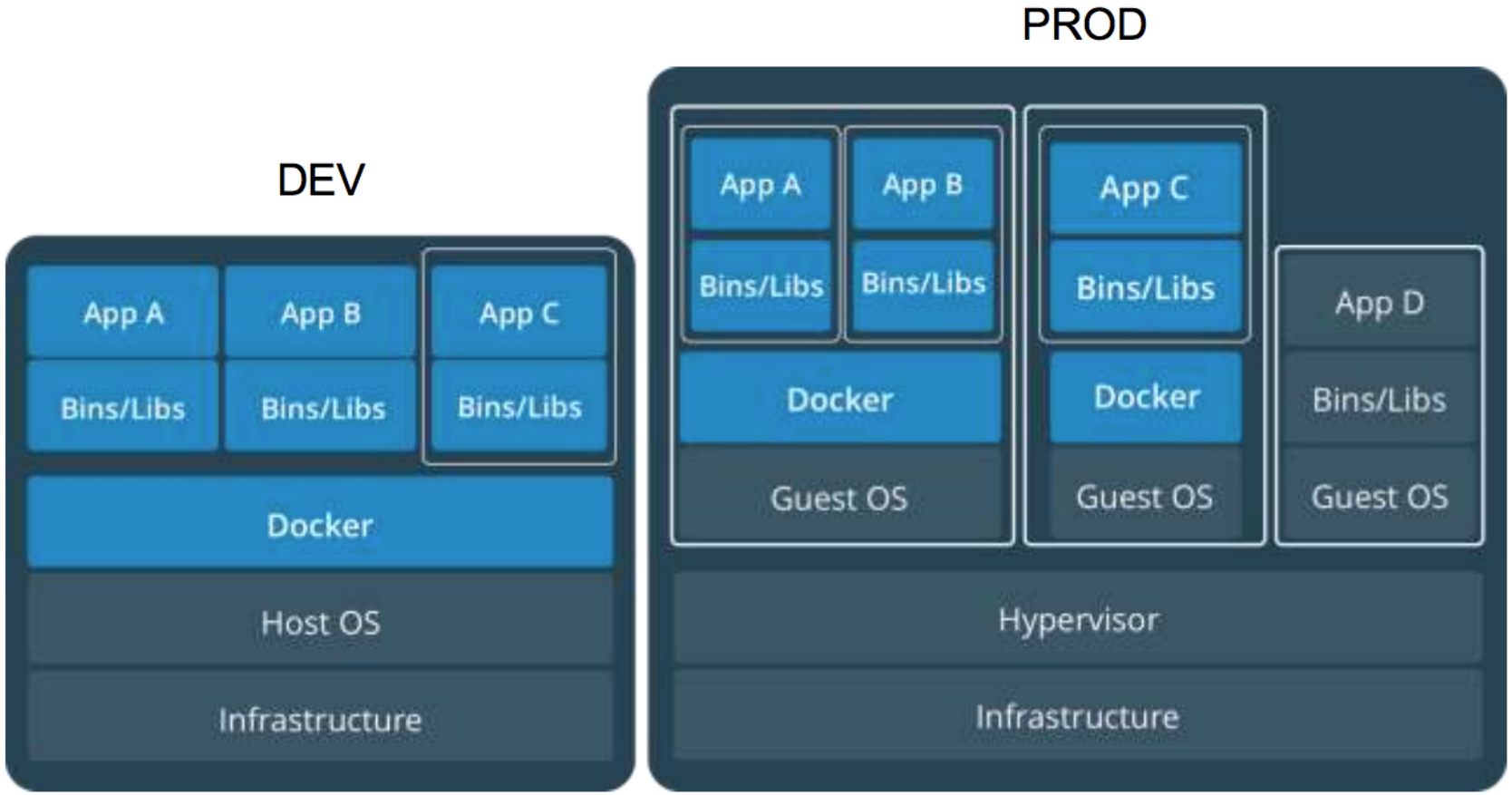




Containers are an app level construct

VMs are an infrastructure level construct to turn one 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 process layers = ability to move between infrastructure

Efficiency

- Less OS overhead
- Improved VM density

Docker Architecture

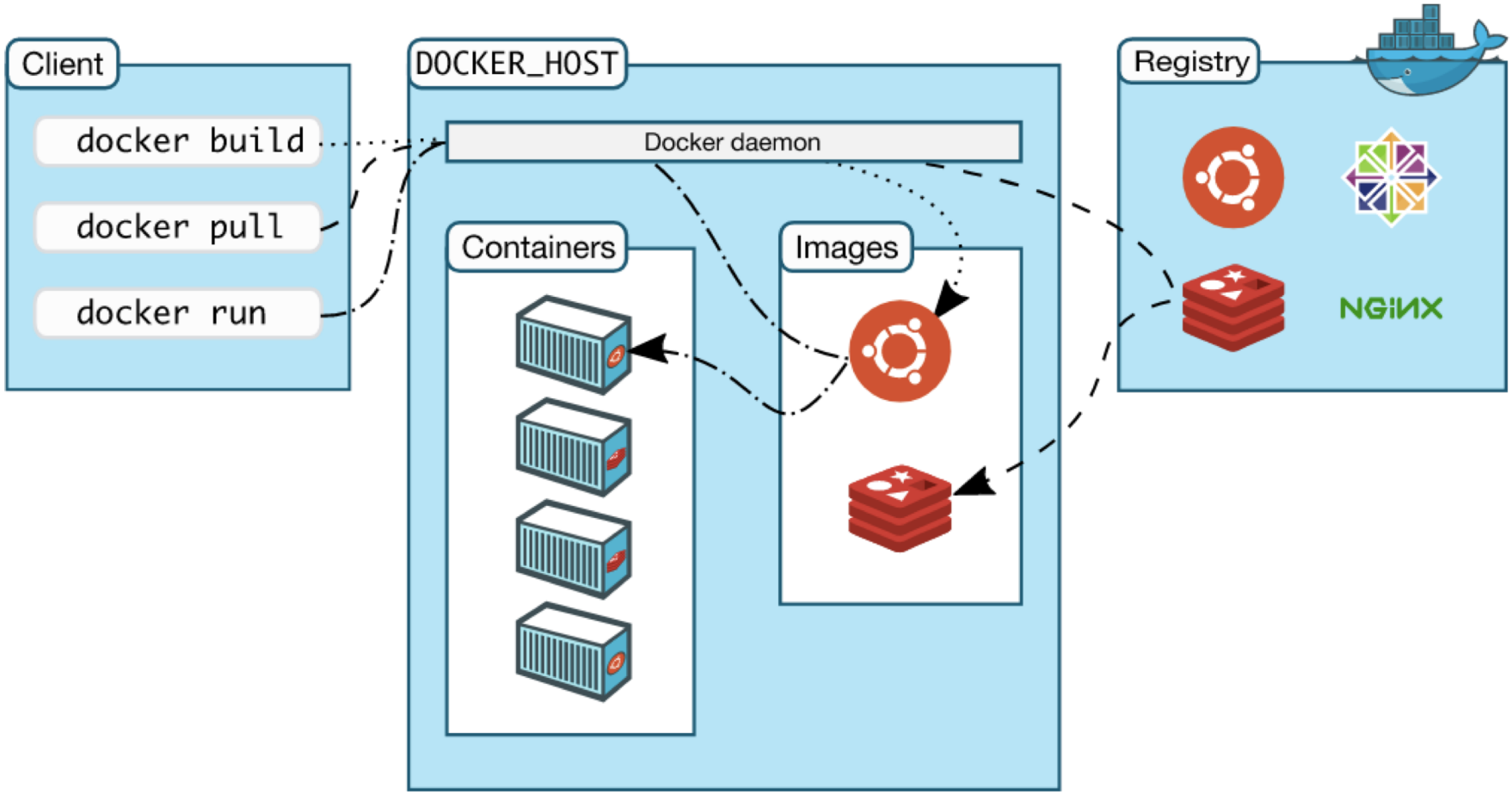
- Docker is a powerful technology, and that often means something that comes with a high level of complexity.
- Fundamental architecture of Docker is a simple client/server model, with only one executable that acts as both components, depending on how you invoke the *docker* command.
- There is a third component to Docker called registry which stores Docker images and metadata about those images.
- The server does the work of running and managing the containers.
- The client is used to tell the server what to do.

Working with Docker

Docker Vocabulary

- Docker Client – The docker command used to control most of the Docker workflow and to talk to remote Docker servers.
- Docker Server – The docker command run in the daemon mode.
- Docker Images – Docker images consist of one or more filesystem layers and some important metadata that represent all the files required to run a Dockerized application. A single docker image can be copied to numerous hosts. A container will typically have both a name and a tag. The tag is used to identify a particular release of an image.
- Docker Container - A Docker container is a Linux container that has be instantiated from a Docker image. A specific container can only exist once; however, you can easily create multiple containers from the same image.

High Level Architecture



Installing Docker

https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/#install-docker-ce

Configure Docker to start on boot

https://docs.docker.com/engine/installation/linux/linux-postinstall/#configure-docker-to-start-on-boot

Docker Command

- Docker command is used for managing docker containers
- Containers usually encapsulate a single process. **pid = 1**.
- Though they can be used as a full VM a container lifecycle ends when the process with pid = 1 ends.

Pull an image or a repository from a registry

https://docs.docker.com/engine/reference/commandline/pull/

Remove one or more images

https://docs.docker.com/engine/reference/commandline/rmi/

Run Docker Container

docker run -it --rm tomcat:8.0

You can test it by visiting http://container-ip:8080 in a browser

docker run -it --rm -p 8888:8080 tomcat:8.0

You can go to http://localhost:8888 or http://host-ip:8888 in a browser.

Fetch the logs of a container

https://docs.docker.com/engine/reference/commandline/logs/

Docker Commands

https://docs.docker.com/engine/reference/commandline/docker/#child-commands

Limit a container's resources

https://docs.docker.com/engine/admin/resource_constraints/

Additional Resources

https://spring2018.csye6225.com/