Docker & Containers

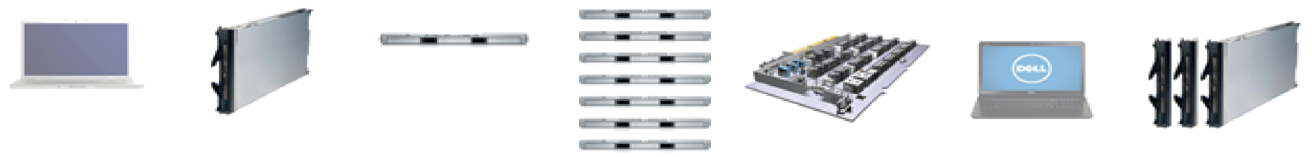
Vikram Krishnamurthy

Agenda

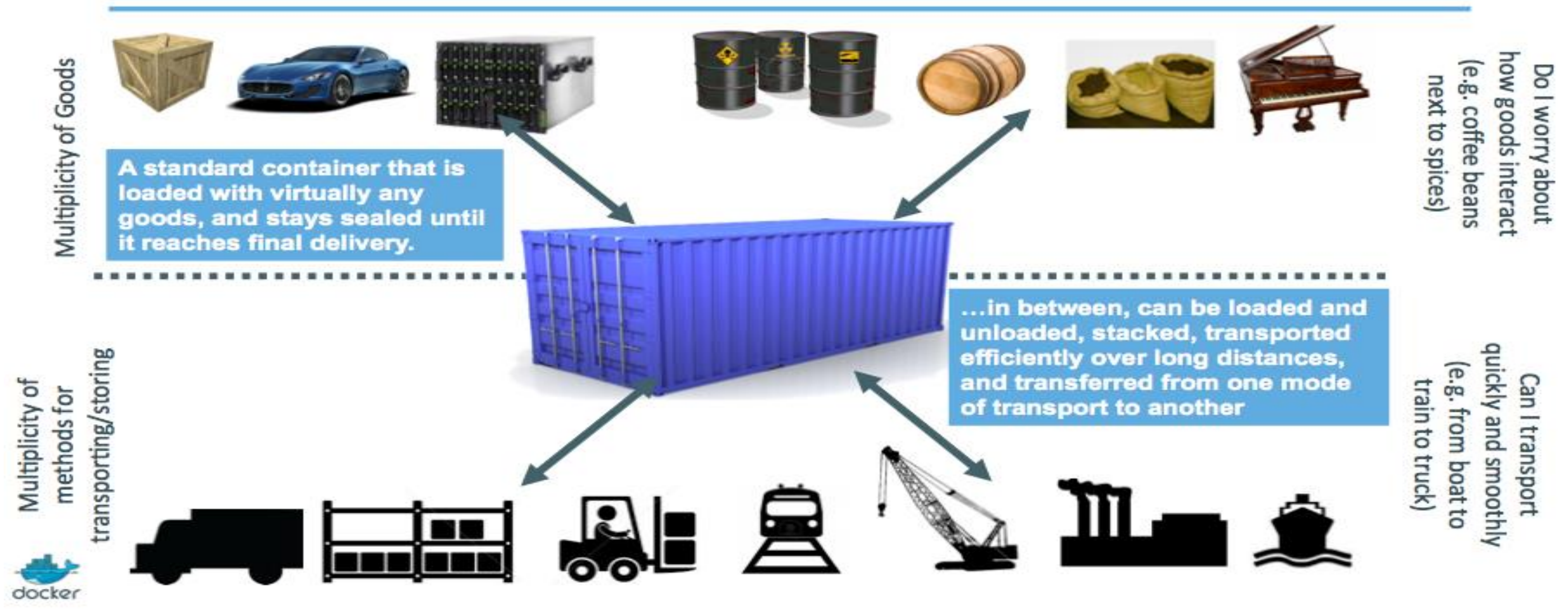
- ***History – cgroups, namespaces, overlayfs***
- ***What is/are Docker/Containers, what problems do they solve?***
- ***How do Containers differ from Hypervisor based virtualization?***
- ***What difference does Docker bring to Containers?***
- ***How does Docker work fundamentally?***

The Challenge – The Matrix From Hell

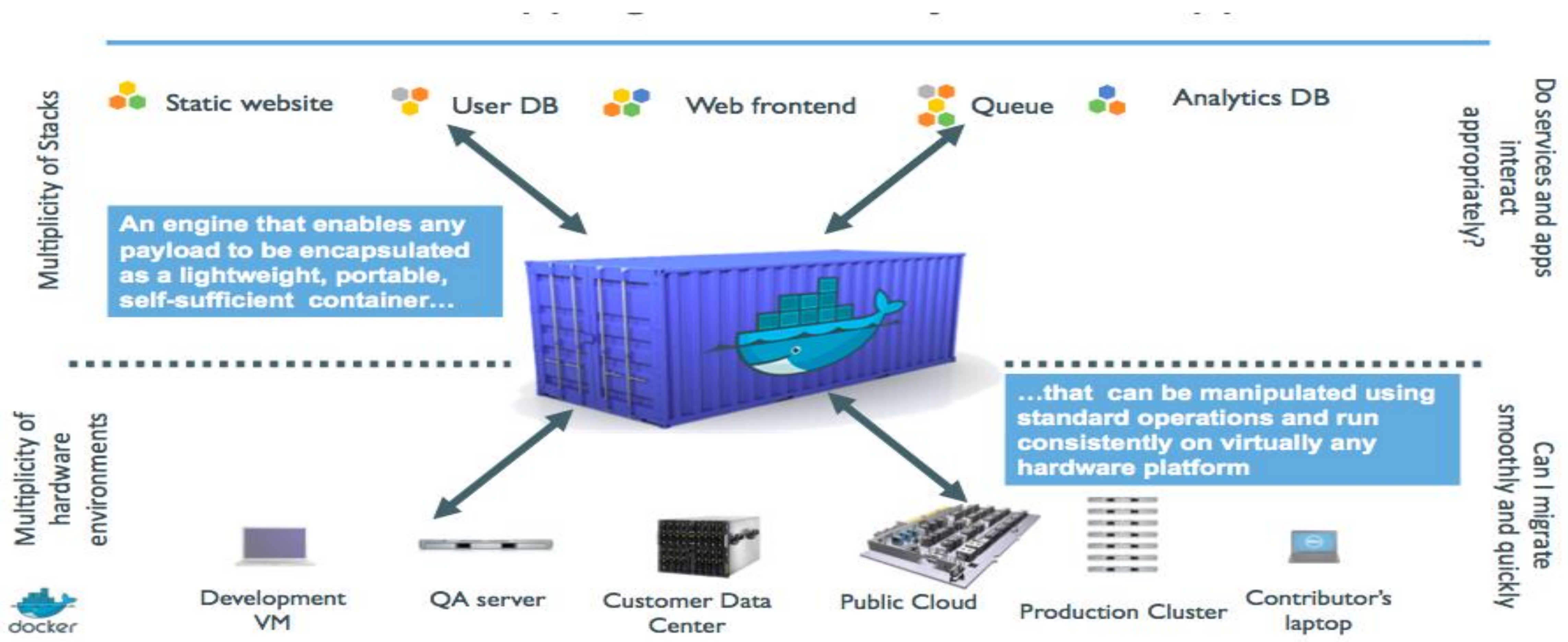
	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?



Help From Elsewhere - Shipping Containers


Multiplicity of Goods
A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.
Do I worry about how goods interact (e.g. coffee beans next to spices)
…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

Applied to IT World - Application Containers

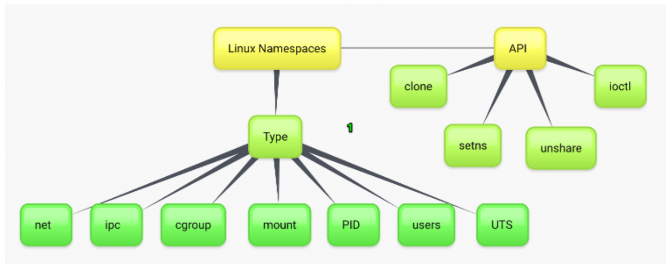

Multiplicity of Stacks
Static website
User DB
Web frontend
Queue
Analytics DB
Do services and apps interact appropriately?
An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…
…that can be manipulated using standard operations and run consistently on virtually any hardware platform
Multiplicity of hardware environments
Can I migrate smoothly and quickly
docker
Development VM
QA server
Customer Data Center
Public Cloud
Production Cluster
Contributor's laptop

Brief History of Docker

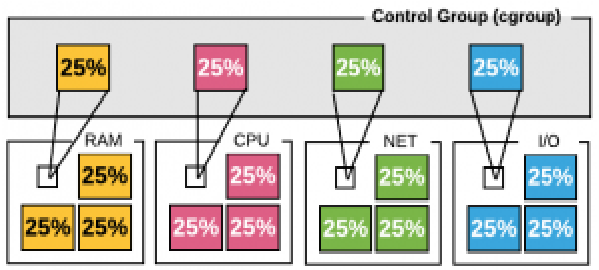
- Docker is a set of Platform as a Service products that use OS level virtalization to deliver software in packages called “Containers”
- Solomon Hykes started Docker project as an internal project within dotCloud, a Platform as a Service company. Later, Docker Inc. was launched by Solomon Hykes and Sebastien Pahl in 2011
- It all started as ‘chroot’ system call in 1979, and followed an evolution path through FreeBSD Jails, Linux Vserver, Solaris Containers, Open Virtuozzo, Process Containers, LXC and finally the Docker ecosystem

Some Concepts behind Docker/Containers

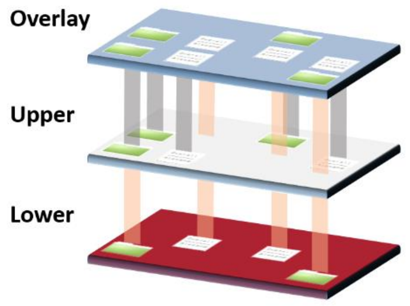
- **Namespaces** are kernel enforced user space views. They are mechanisms to abstract, isolate, and limit the visibility that a group of processes has over various system entities such as process trees, network interfaces, user IDs, and filesystem mounts.
  - Namespace Types: Process ID, Mount, UTS, IPC, User, Network, Cgroups – virtualizes content of /proc/self/cgroup file.



- **Cgroups** limit the amount of resources a process can consume (CPU, memory, network bandwidth, etc.). They were introduced in Linux kernel 2.6.24
  - Applying cgroups on namespaces allows isolation of processes into containers within a system, where resources can be managed distinctly.



- **OverlayFS**: Union file systems are a creative solution to allow a virtual merge of multiple folders, while keeping their actual contents separate. The Overlay file system (OverlayFS) is one example of these, though it is more of a mounting mechanism than a file system.
- OverlayFS allows you to overlay the contents (both files and directories) of one directory onto another
- *Lower* directory is Read only, *Upper* directory is RW



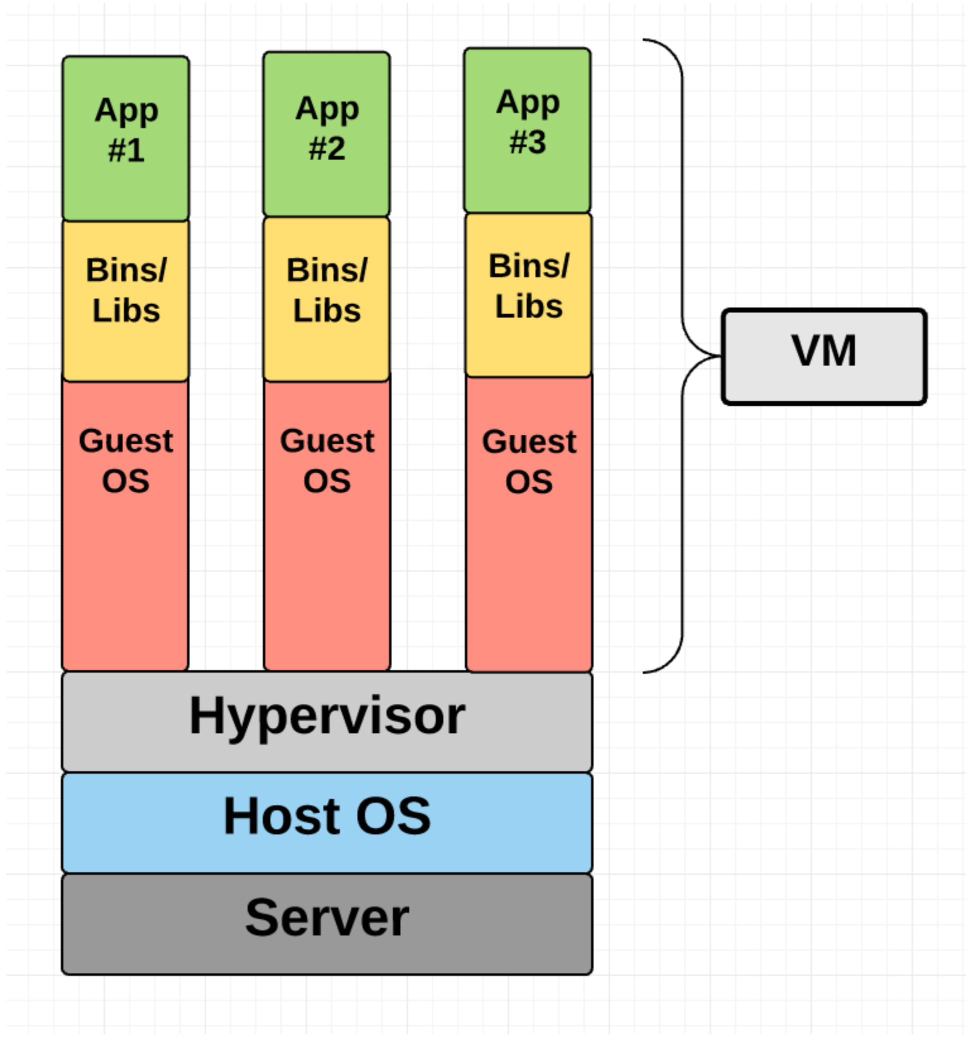
Processes vs VMs vs Containers

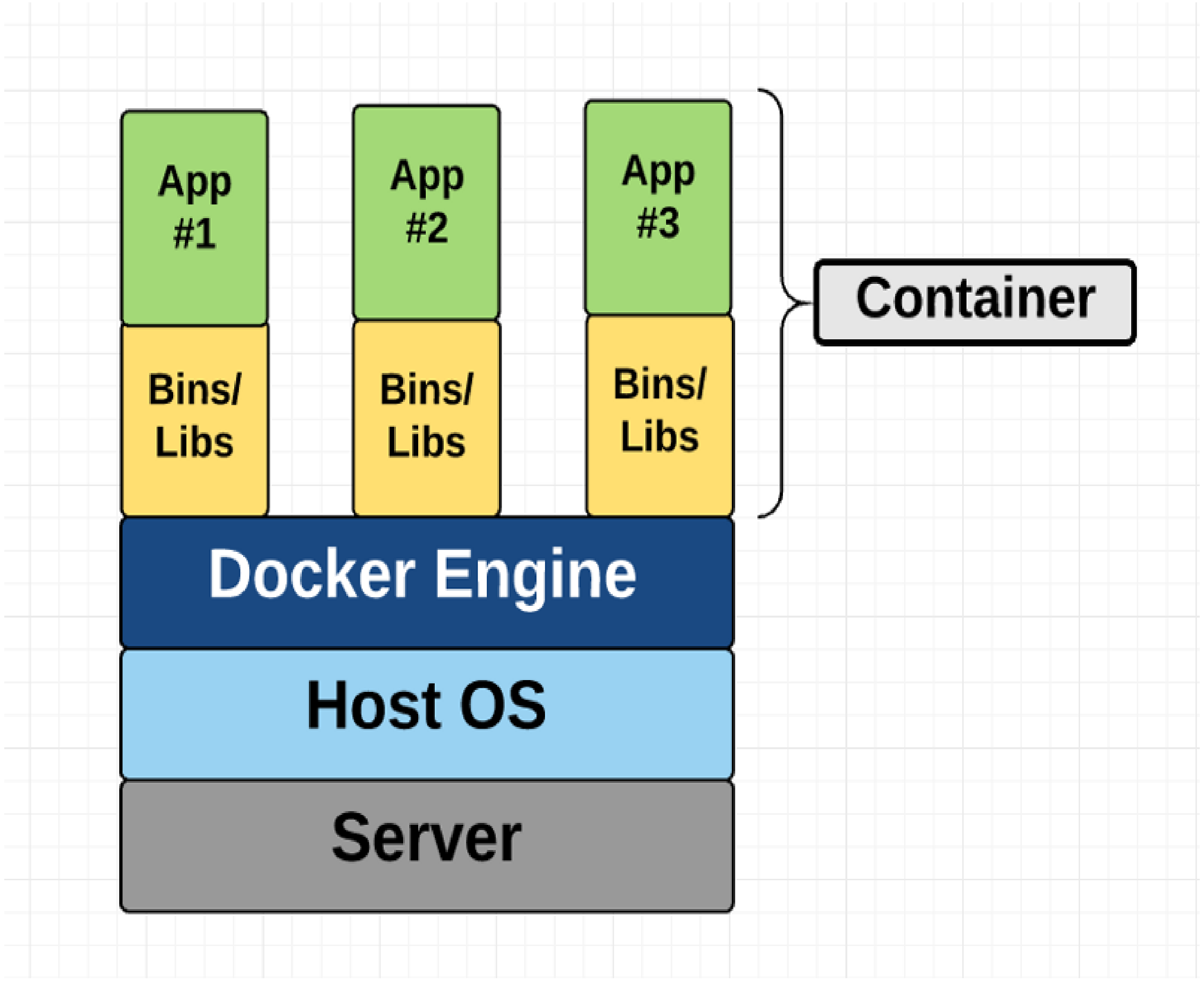
Process
- Isolate address space
- No isolation for files or networks
- Lightweight

Virtual Machine
- Isolate address space
- isolate files and networks
- Heavyweight

Container
- Isolate address space
- isolate files and networks
- Lightweight

Containers vs Virtual Machines?





Hypervisors virtualize hardware, Containers virtualize OS!!

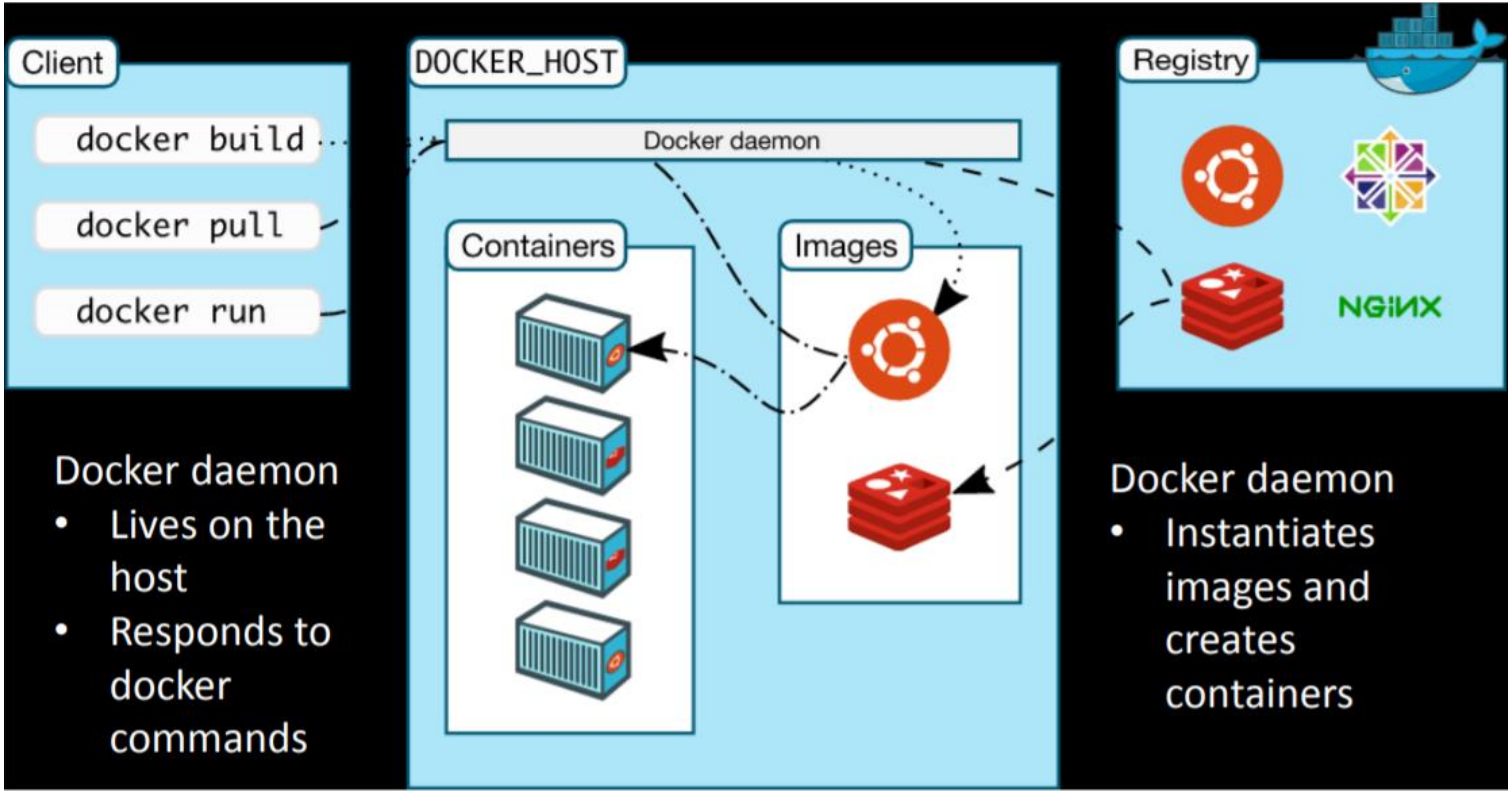
Why Should We Care about Containers?

- Build once, run *almost* anywhere
- A clean, safe, portable runtime environment for the app.
- Eliminate worries about missing dependencies, packages or compatibility between different platforms.
- Run each app in its own isolated container, so various versions of libraries and app can be run without conflicts.
- Automate testing, integration, packaging
- A VM without the overhead of a VM.

Docker – Important Terminology

- Image: Persisted snapshot that can be «run»
- Container: Live running instance of a Docker «image»
- Dockerfile: A text document with commands to build Docker "image".
- Docker Client : The utility that runs docker commands – *docker run, docker ps, docker build* etc.
- Docker Daemon/Engine: The server part that talks to the kernel, makes the system calls to create, operate and manage containers.

Docker Basic Architecture



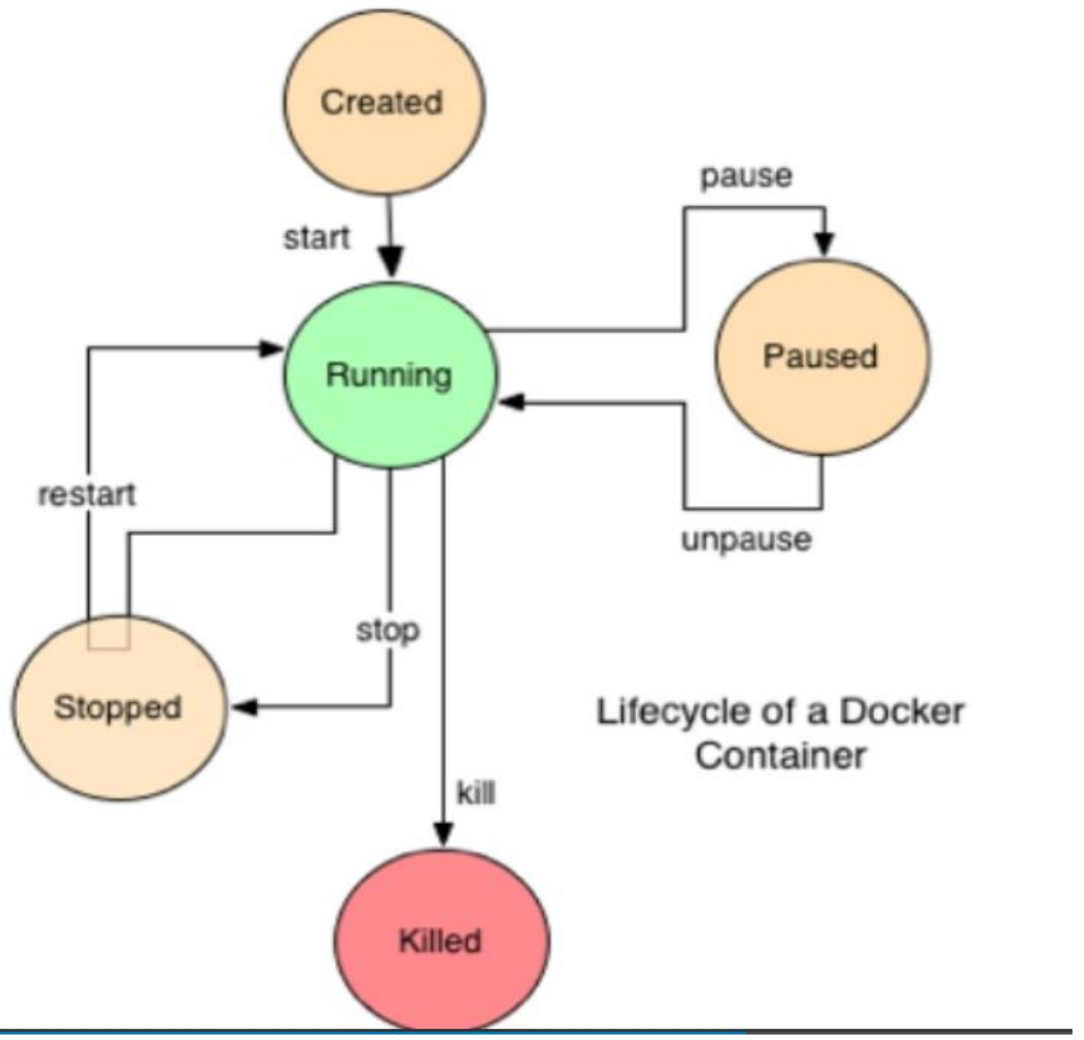
Docker daemon

- Lives on the host
- Responds to docker commands

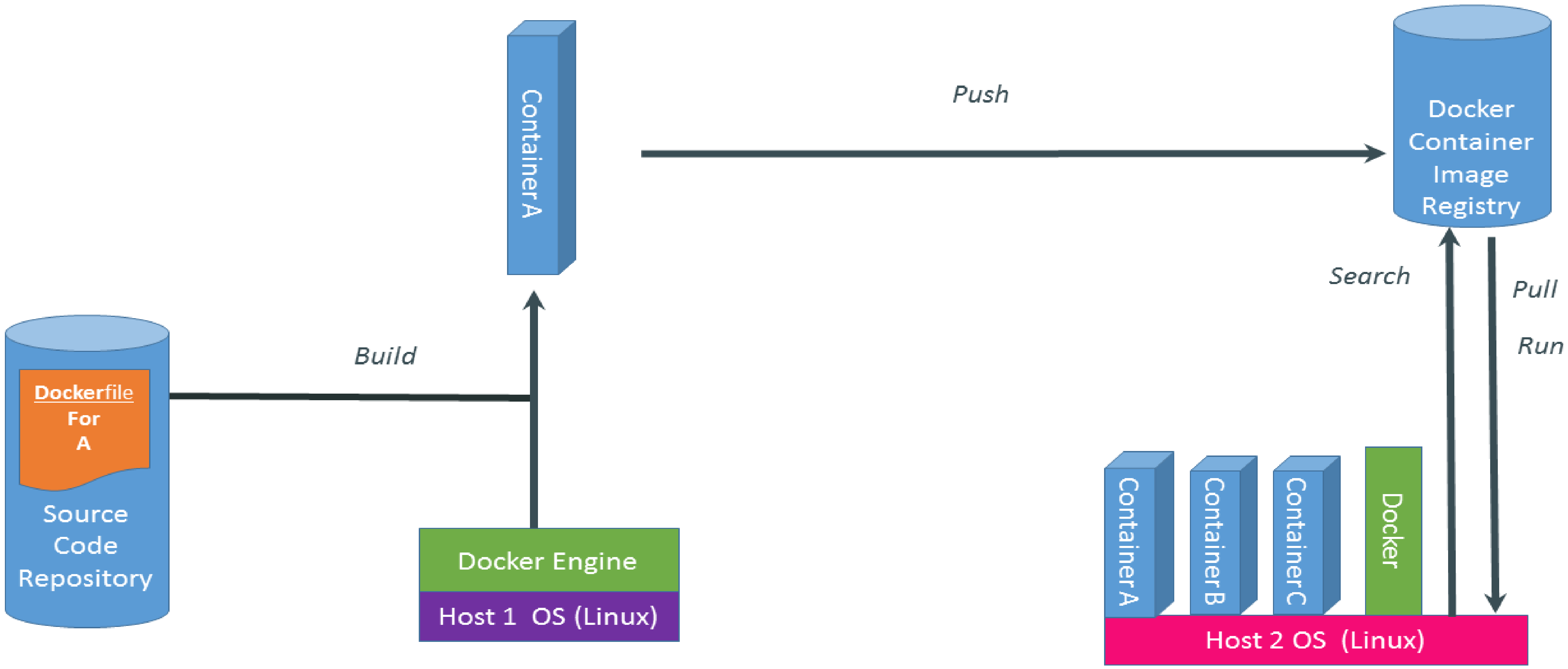
Docker daemon

- Instantiates images and creates containers

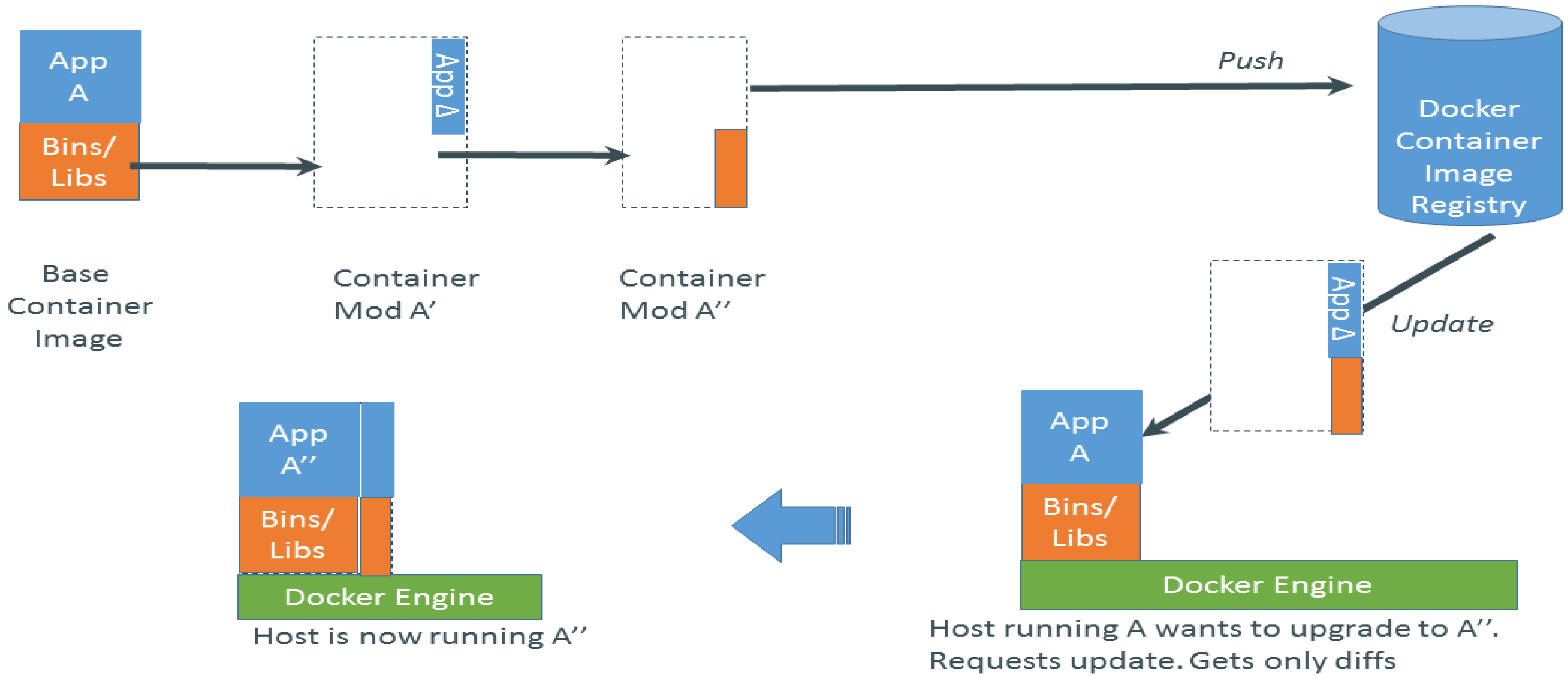
Docker Container Life Cycle



Docker Workflow - Basics



Docker Workflow – App Updates / Changes



Host is now running A''

Host running A wants to upgrade to A''. Requests update. Gets only diffs

Thank You.