Docker & Containers – A Practitioner’s Perspective

Nanda Kishore
Micro Focus
September 2018

Agenda

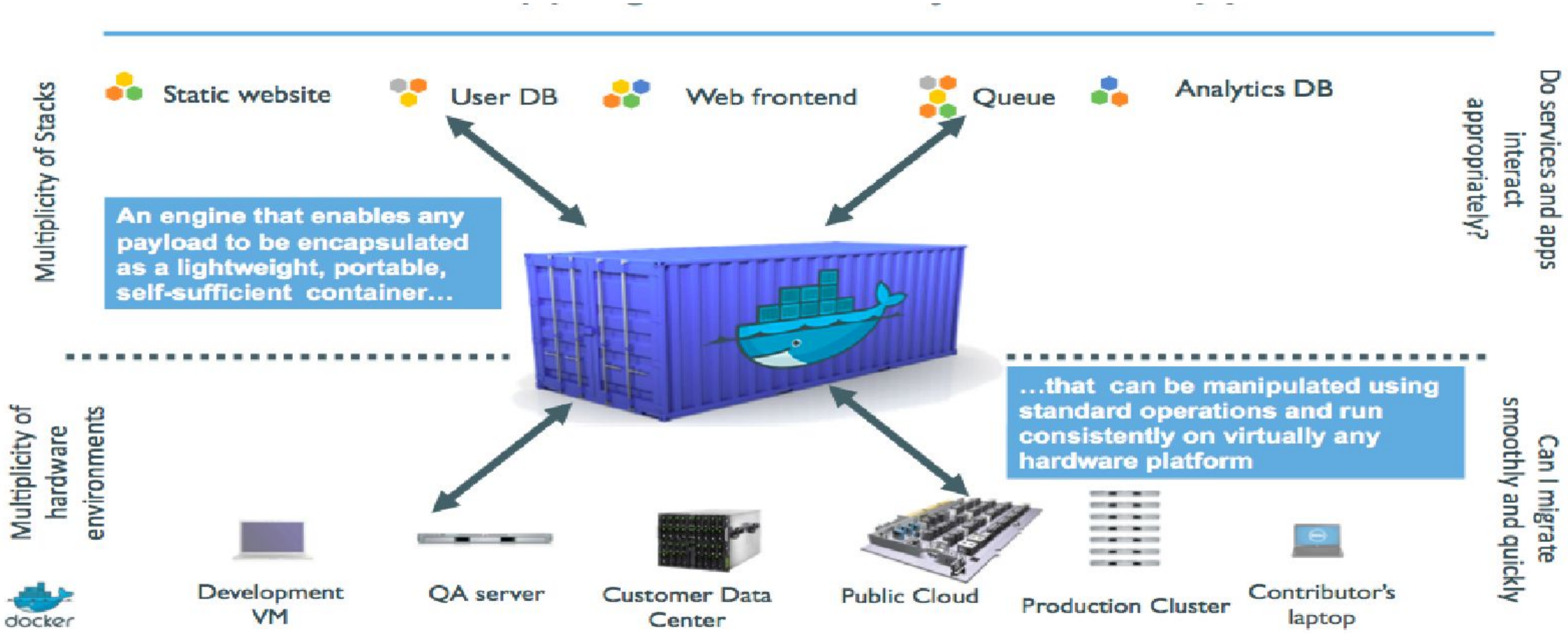
- ***What is/are Docker/Containers, what problem they solve?***
- ***How containers differ from Hypervisor based virtualization?***
- ***What difference does Docker bring to Containers?***
- ***How Docker works fundamentally?***
- ***Demo Time!!***

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

Applied to IT World - Application Containers



Containers vs Processes vs VMs

Process

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

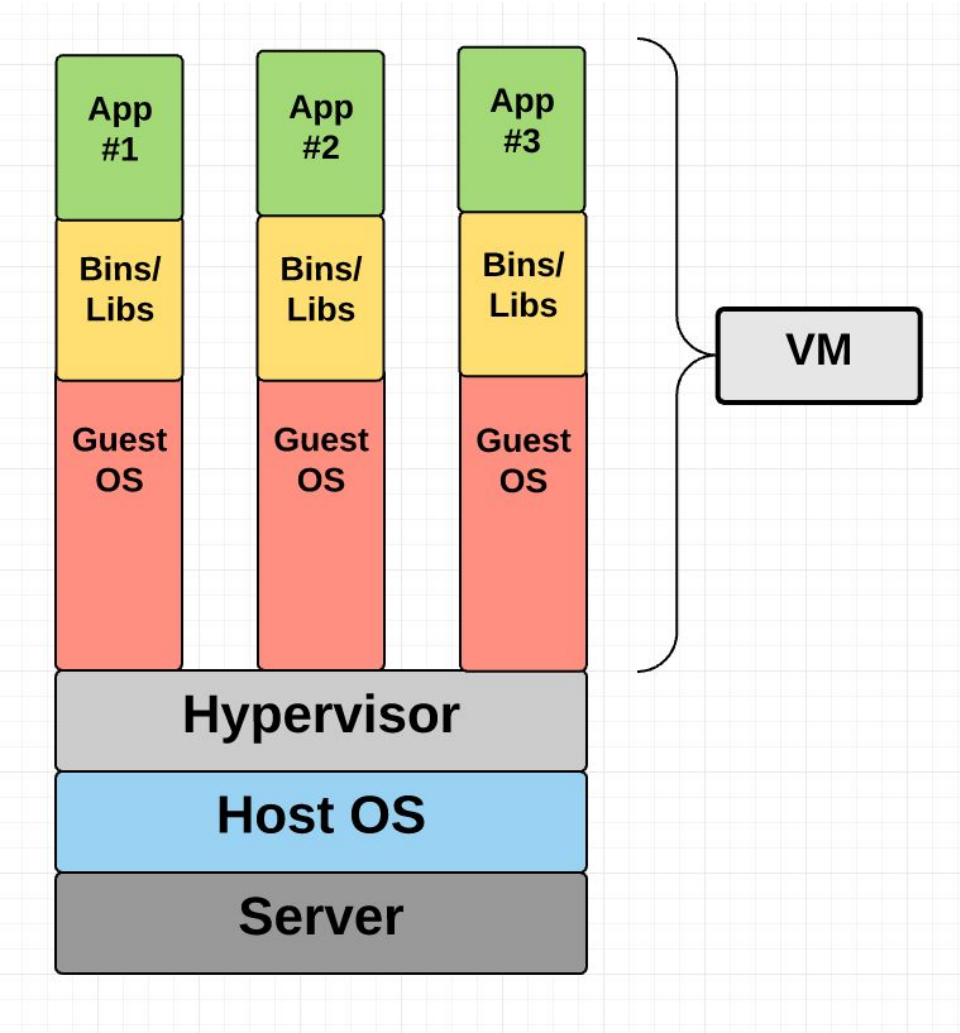
Virtual Machine

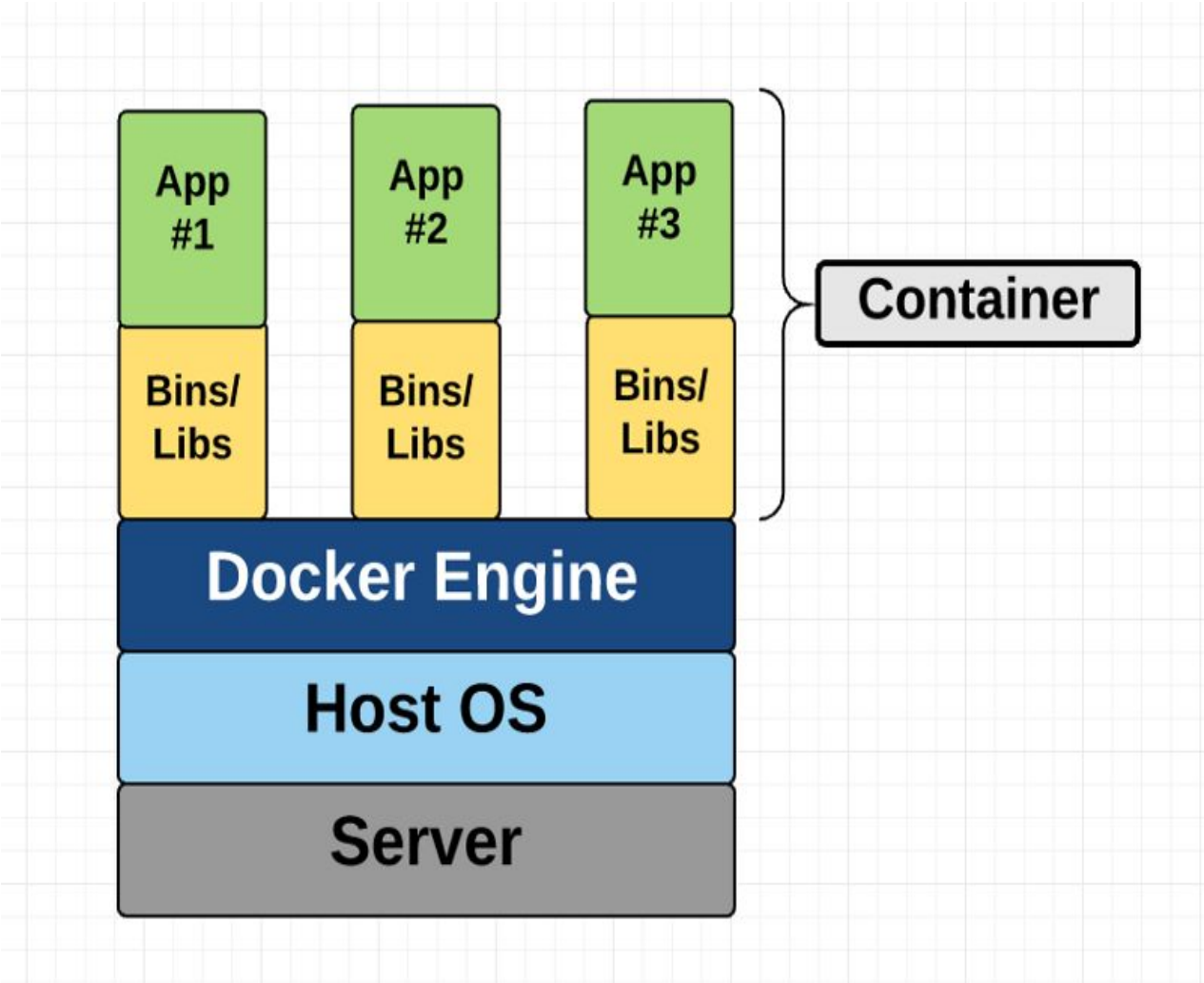
- Isolate address space
- isolate files and networks
- Heavyweight

Container

- Isolate address space
- isolate files and networks
- Lightweight

Containers vs Virtual Machines?





Hypervisors virtualize hardware, Containers virtualize OS!!

Why Developers Should Care for 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.

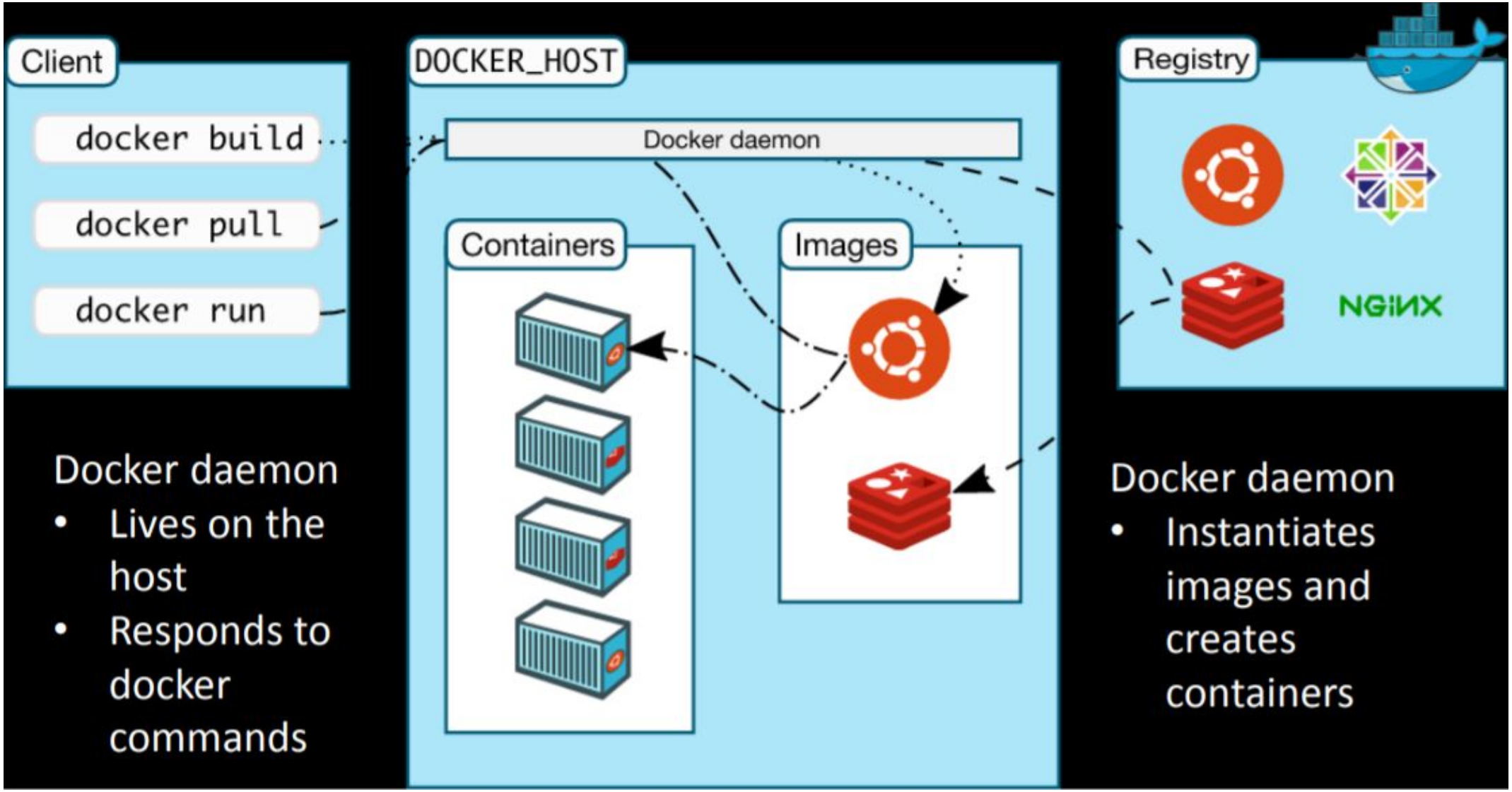
Containers – Docker's Value Proposition

- Before Docker
  - Have to be hand crafted manually
  - Shipped as individual packages/scripts: deb, rpm, gem, jar...
  - Not many re-usable components, APIs, tools.
- Docker Containers
  - Can be assembled automatically from "*Dockerfile*"s
  - Can be shipped with all their dependencies as standard format "Docker image"s
  - Images can be versioned
  - Break image into layers and only ship layers that have changed
  - Command line and API interfaces, ecosystem of standard tools.

What is Docker - Disambiguation

- ***Docker Project***: An Open Platform to Build, Ship, and Run Distributed Applications as Containers
- ***Docker Inc***: The company managing the open source project and building enterprise grade toolset around it

Docker Basic Architecture



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.

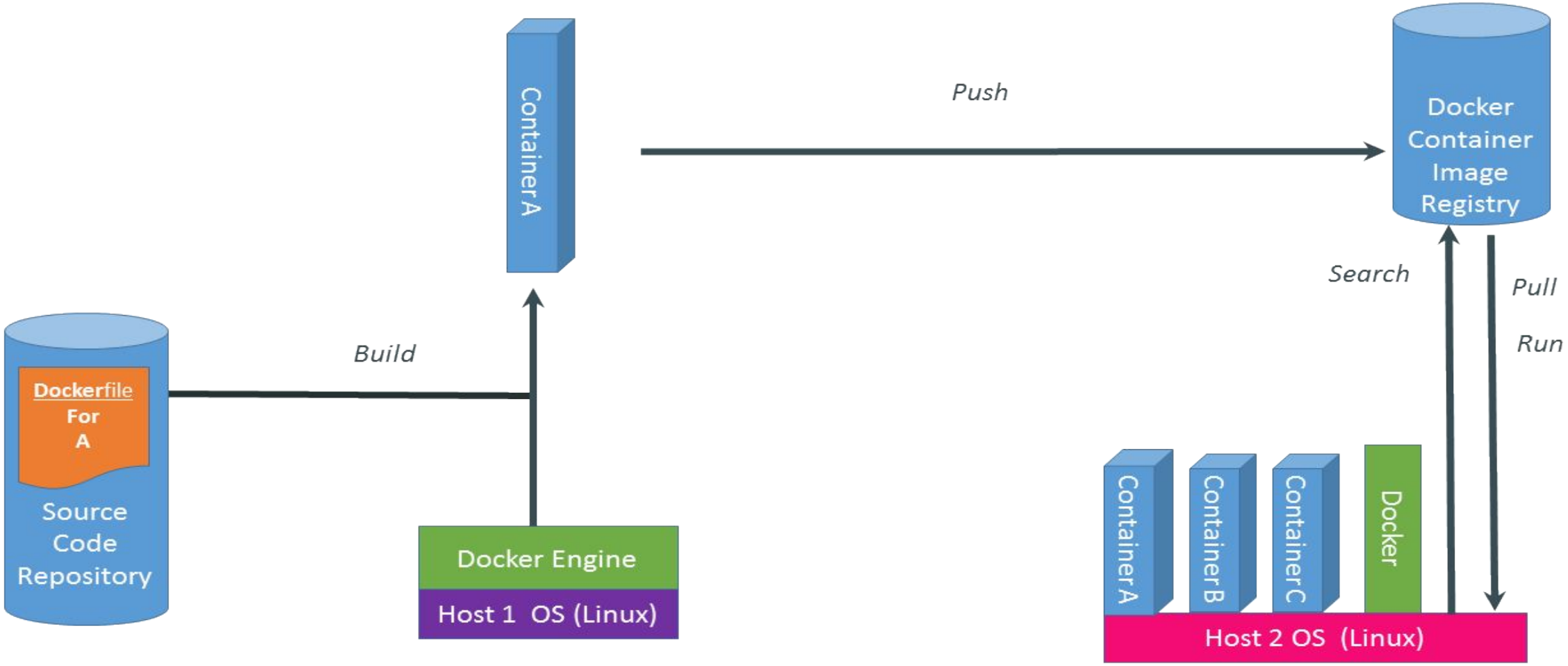
Docker Containers – Linux and Windows

- Similarities
  - Both are application containers, run natively, do not depend on hypervisors or virtual machines.
  - Both administered through Docker CLI/APIs
  - They provide the same portability and modularity features on both operating systems.

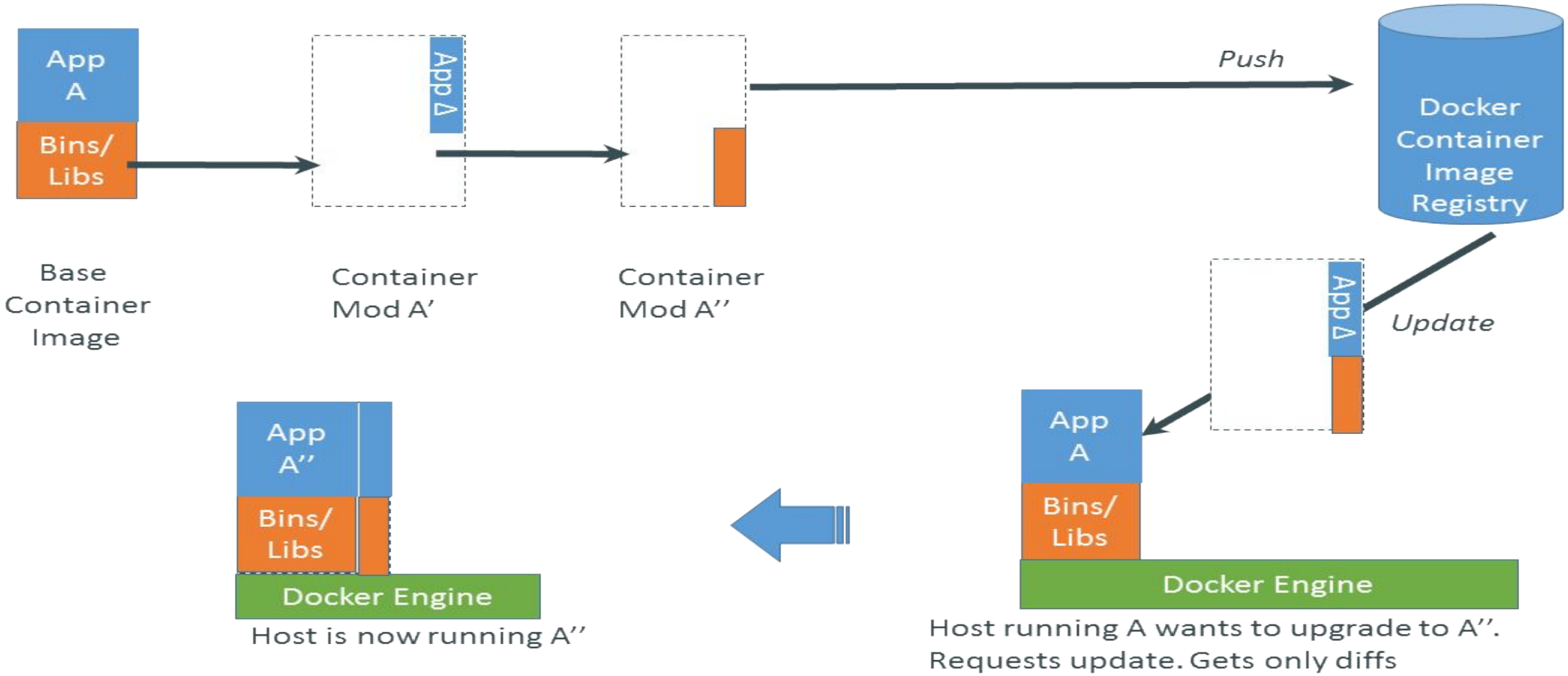
Docker Containers – Linux and Windows

- Differences
  - Docker supports only Windows Server 2016 and Windows 10 now.
  - But Docker can run on any type of modern Linux-based operating system.
  - Most container orchestration systems used for Docker on Linux are not supported on Windows.

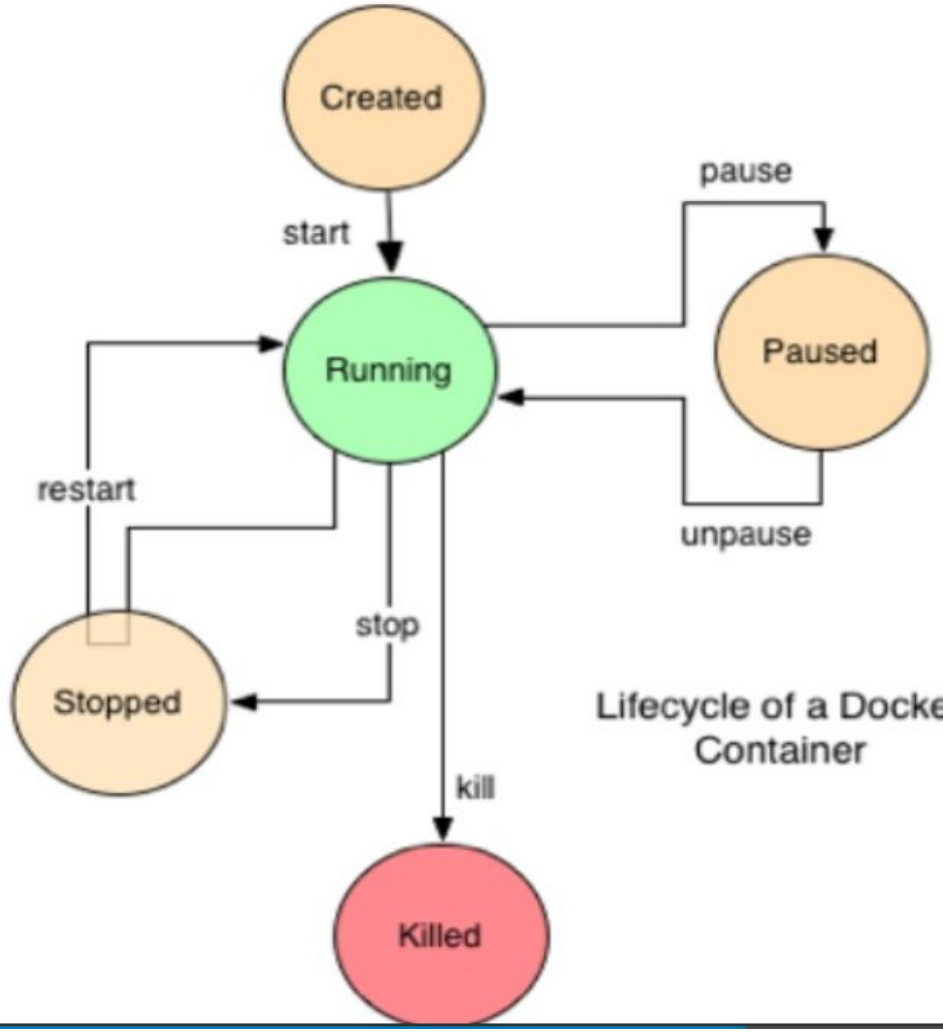
Docker Workflow - Basics



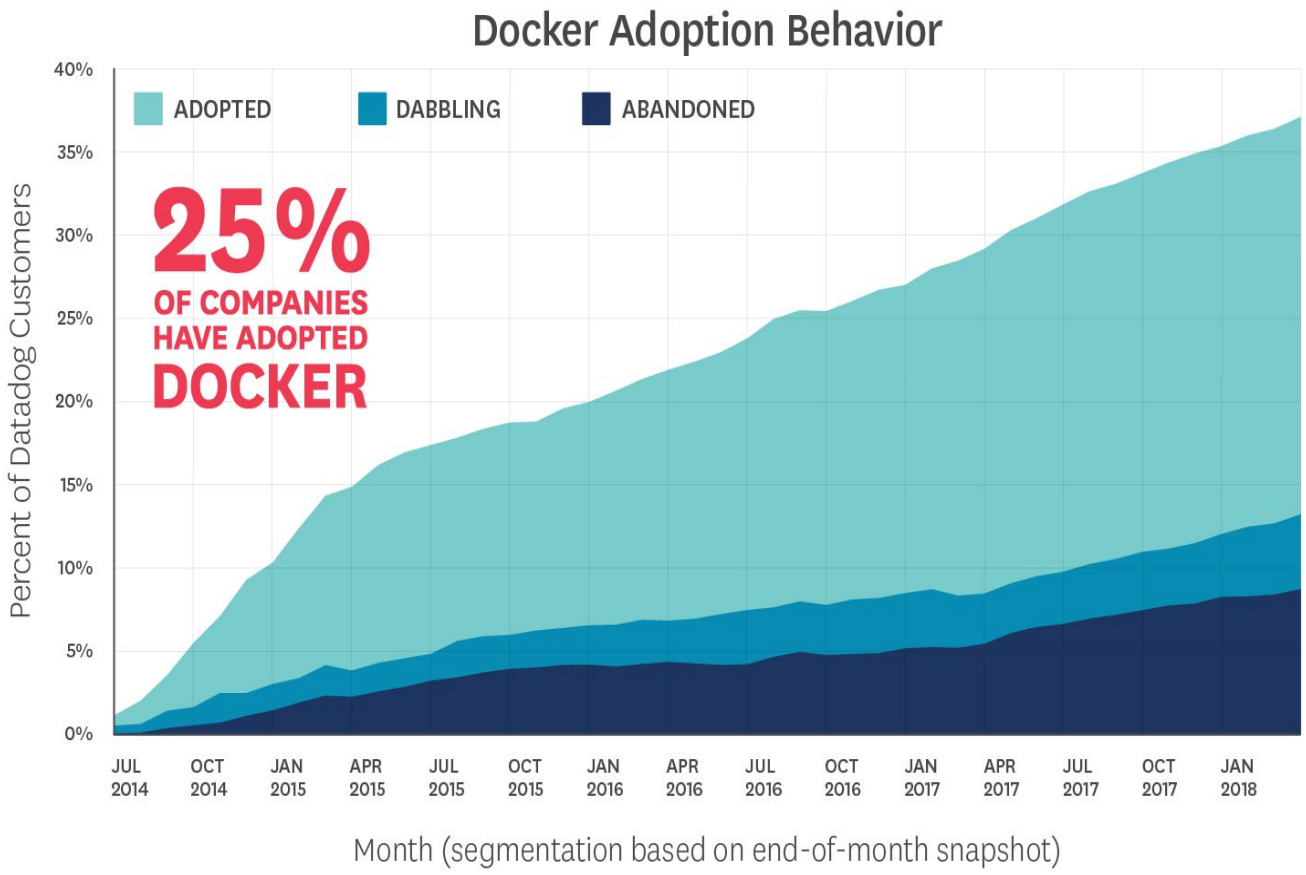
Docker Workflow – App Updates / Changes



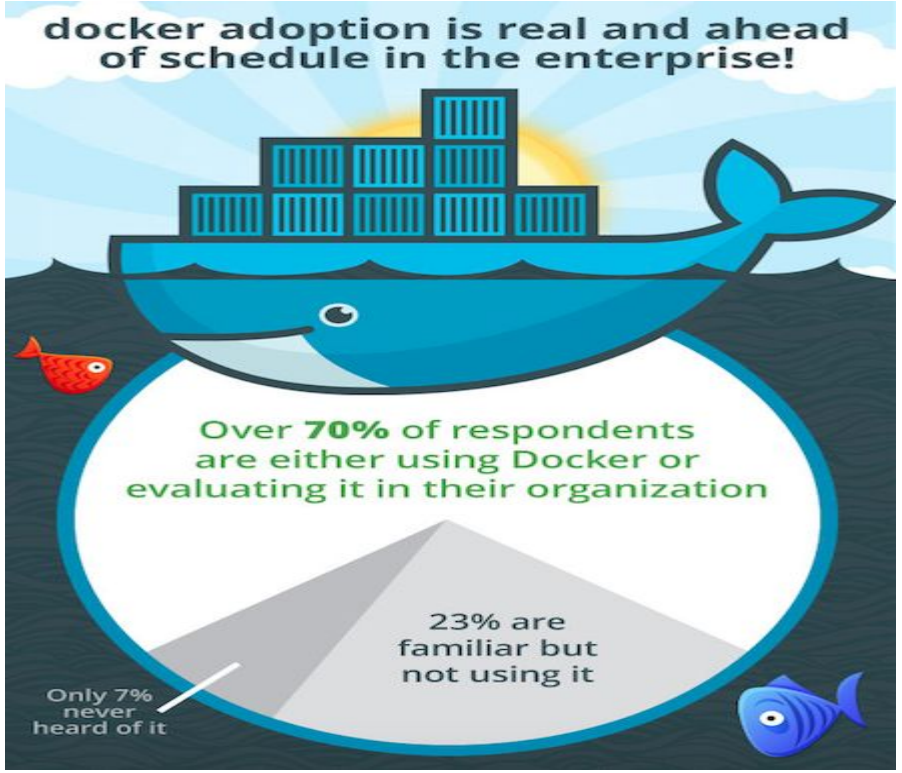
Docker Container Life Cycle



Docker – User Adoption



Source: Datadog



Docker Basic Examples

- docker run hello-world, docker pull and docker images
- docker run ubuntu bash
- docker run -idt ubuntu bash , install vi and docker commit
- docker exec –it «container_id»
- docker run --name web -d -p 3000:80 nginx
- docker run --name web1 -d -p 3100:80 --mount type=volume,source=nginx-vol,destination=/usr/share/nginx/html nginx
- docker run --name web2 -d -p 3200:80 -v /root/Public:/usr/share/nginx/html nginx

Thank You.