

docker

Docker, Containers, and the  
Future of Application Delivery

# In the four months since we launched

- >50,000 pulls
- >4,000 github stars
- >100 significant contributors
- >150 projects built on top of docker
  - UIs, mini-PaaS, Remote Desktop....
- 1000's of Dockerized applications
  - Memcached, Redis, Node.js...
- Integration in Jenkins, Travis, Chef, Puppet, Vagrant and OpenStack
- Meetups arranged around the world...with organizations like Ebay, Uber, Mozilla, Cloudflare, and Rackspace presenting on their use of Docker

 **David Rousselie** @drousselie 2d  
Docker community is expanding. Really the most exciting project lately.  
[blog.docker.io/2013/07/docker...](http://blog.docker.io/2013/07/docker...)  
Details

 **Phil Whelan** @philwhln 2d  
"Awesome projects from the Docker community | Docker Blog"  
[bit.ly/16yC72C](http://bit.ly/16yC72C)  
Details

 **Luc Perkins** @lucperkins 2d  
Somehow I get this weird feeling that I haven't even begun to grasp the implications of @getdocker  
Details

 **John Fink** @adr 3d  
there are probably a million of these, but this one is mine: generic LAMP stack for @getdocker.  
[index.docker.io/u/jbfink/lamps...](http://index.docker.io/u/jbfink/lamps...)  
Details

 **Phil Plante** @pplante 23d  
woot! our new @getdocker cluster is performing way better than expected, and is 5x faster than our cloud setup.  
Details

 **Ben Bleything** @bleything 5d  
you guys, @getdocker. holy shit.  
Details

 **omo** @omo2009 6d  
[blog.docker.io/2013/07/docker...](http://blog.docker.io/2013/07/docker...)  
DockerのなかでXを動かす話。コンテナ作ってから apt-get とか無茶しやがって...

 **Jake Dahn** @jakedahn 6d  
every time i use @getdocker it just gets more mind-glowingly amazing  
Details

 **Sandeep** @machbio 23d  
One of the most Kick-ass Project at this Moment. credits to @progrium and #docker.io  
Details

 **Damian Gryski** @dgryski 3d  
@i\_x\_s All the cool kids are moving towards @getdocker.  
Conversation

 **Fenn** @fennb 24d  
Docker (& LXC in general) could be the most important step in virtualization since hypervisors. Impressive stuff. docker.io  
Details

Why all the excitement?

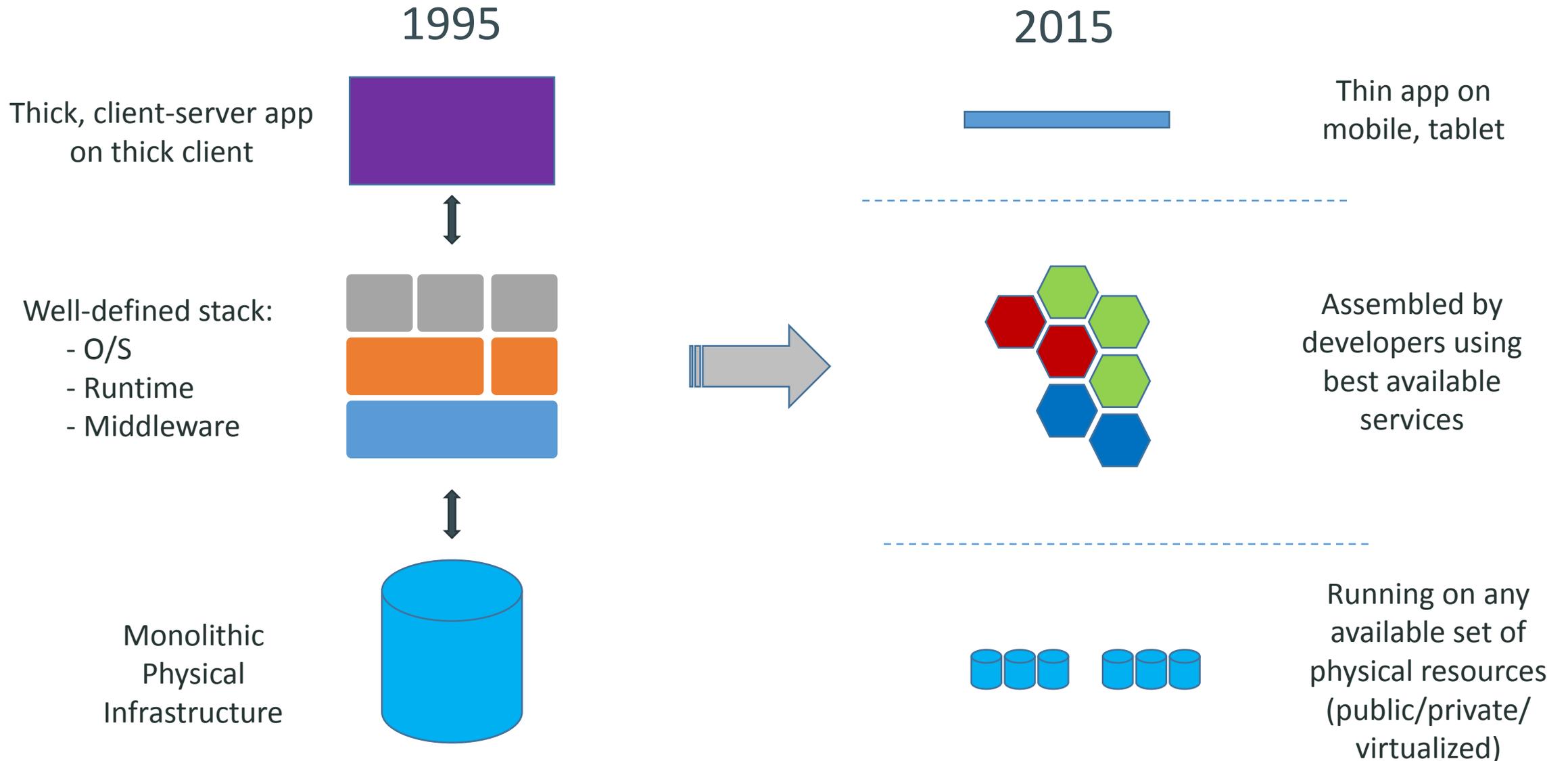
# Contents

---

- The challenge
- The solution
- Why Docker and Containers Matter?
- How They Work?
- Alternative/Complementary Approaches



# Market View: Evolution of IT



# Challenges

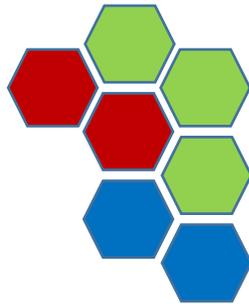
---

2015

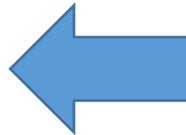
Thin app on  
mobile, tablet



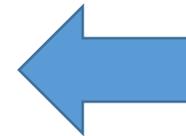
Assembled by  
developers using  
best available  
services



How to ensure services  
interact  
consistently, avoid  
dependency hell



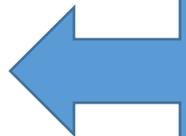
How to avoid n X n  
different configs



Running on any  
available set of  
physical resources  
(public/private/  
virtualized)



How to migrate & scale  
quickly, ensure  
compatibility



# The Challenge

Multiplicity of Stacks

 Static website  
nginx 1.5 + modsecurity + openssl + bootstrap 2

 User DB  
postgresql + pgv8 + v8

 Queue  
Redis + redis-sentinel

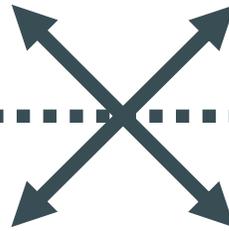
 Analytics DB  
hadoop + hive + thrift + OpenJDK

 Background workers  
Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs

 Web frontend  
Ruby + Rails + sass + Unicorn

 API endpoint  
Python 2.7 + Flask + pyredis + celery + pycopg + postgresql-client

Do services and apps interact appropriately?



Multiplicity of hardware environments

 Development VM

 QA server

Customer Data Center



Public Cloud

Disaster recovery

Production Servers

Production Cluster



Contributor's laptop



Can I migrate smoothly and quickly?



# Results in N X N compatibility nightmare

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



A useful analogy...

# Cargo Transport Pre-1960

Multiplicity of Goods



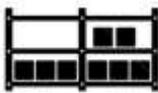
Do I worry about how goods interact (e.g. coffee beans next to spices)

Multiplicity of methods for transporting/storing



Can I transport quickly and smoothly (e.g. from boat to train to truck)

# Also an NxN Matrix

	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
							

# Contents

---

- The challenge
- The solution
- Why Docker and Containers Matter?
- How They Work?
- Alternative/Complementary Approaches



# Solution: Intermodal Shipping Container

Multiplicity of Goods



A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.

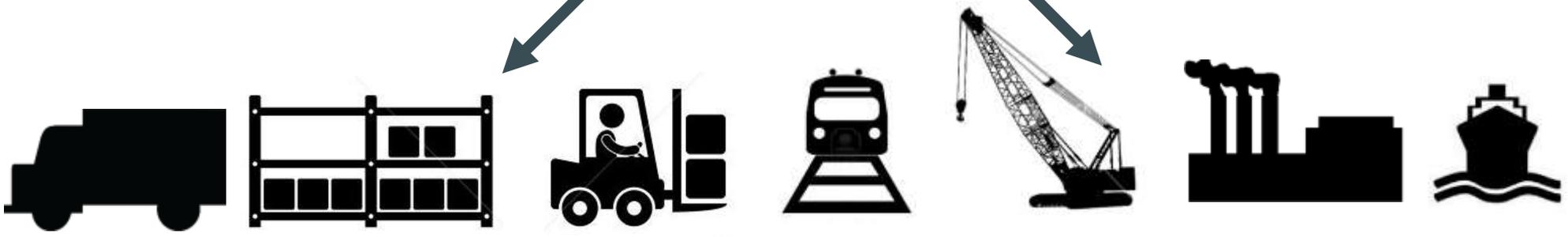


...in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another

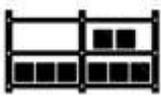
Do I worry about how goods interact (e.g. coffee beans next to spices)

Can I transport quickly and smoothly (e.g. from boat to train to truck)

Multiplicity of methods for transporting/storing



# This eliminated the NXN problem...

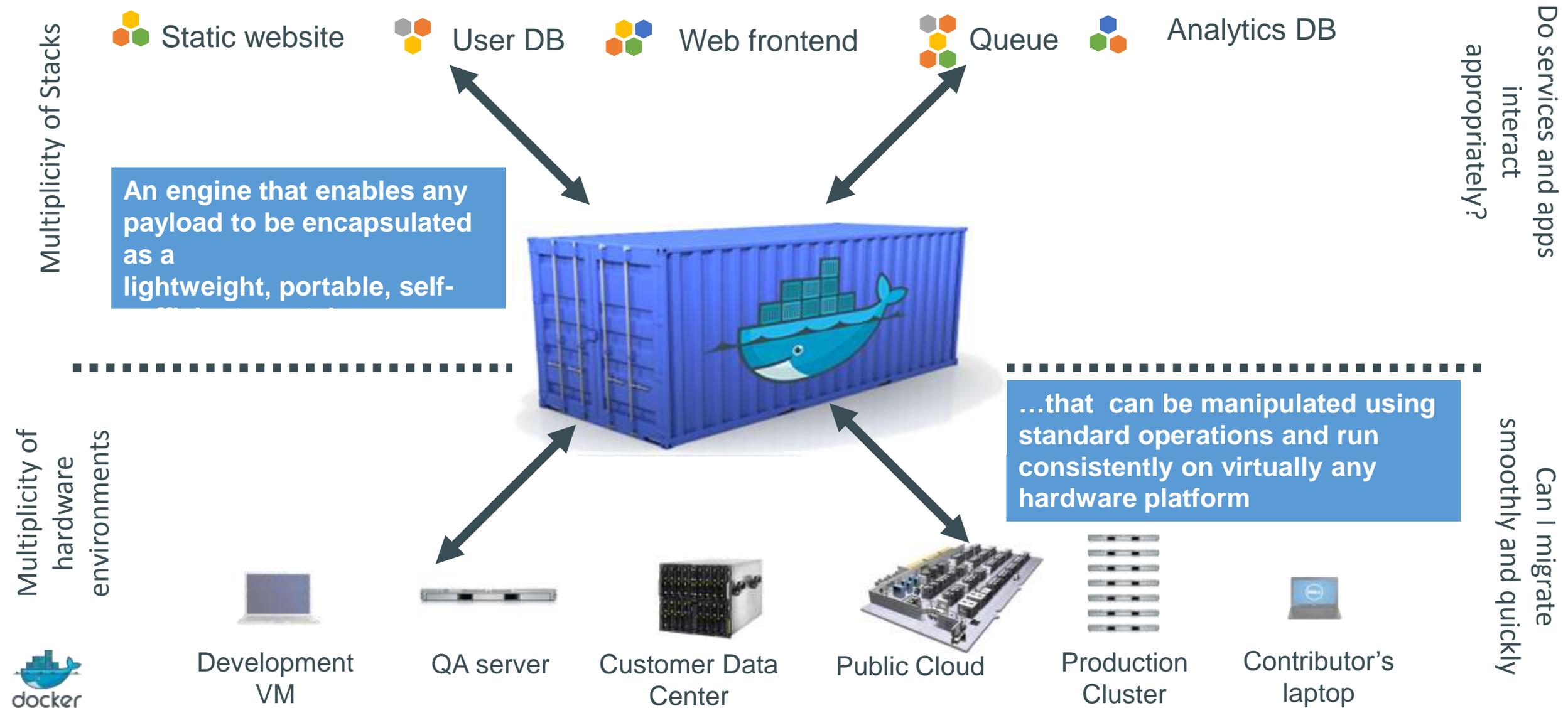
# and spawned an Intermodal Shipping Container Ecosystem

---

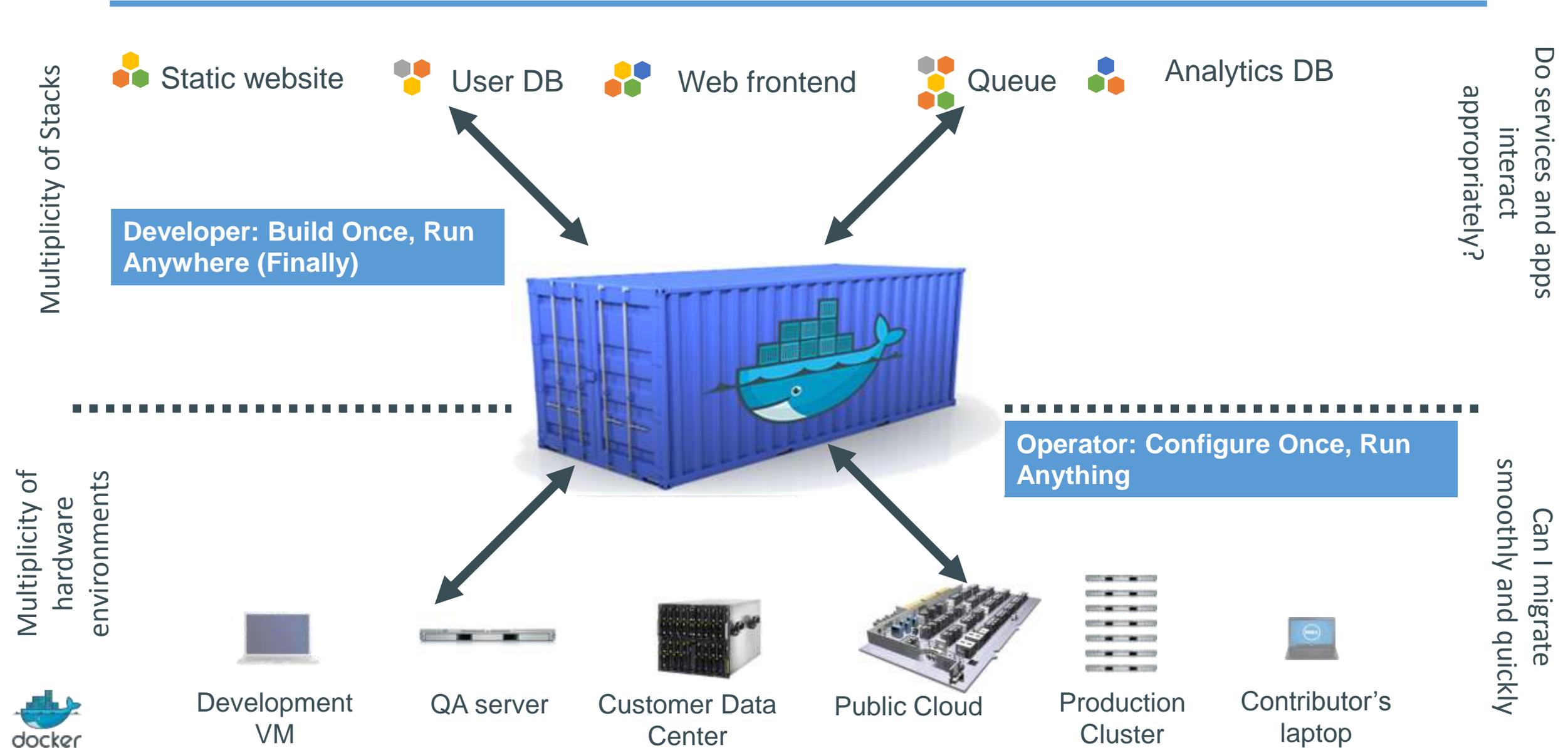


- 90% of all cargo now shipped in a standard container
- Order of magnitude reduction in cost and time to load and unload ships
- Massive reduction in losses due to theft or damage
- Huge reduction in freight cost as percent of final goods (from >25% to <3%)
- massive globalizations
- 5000 ships deliver 200M containers per year

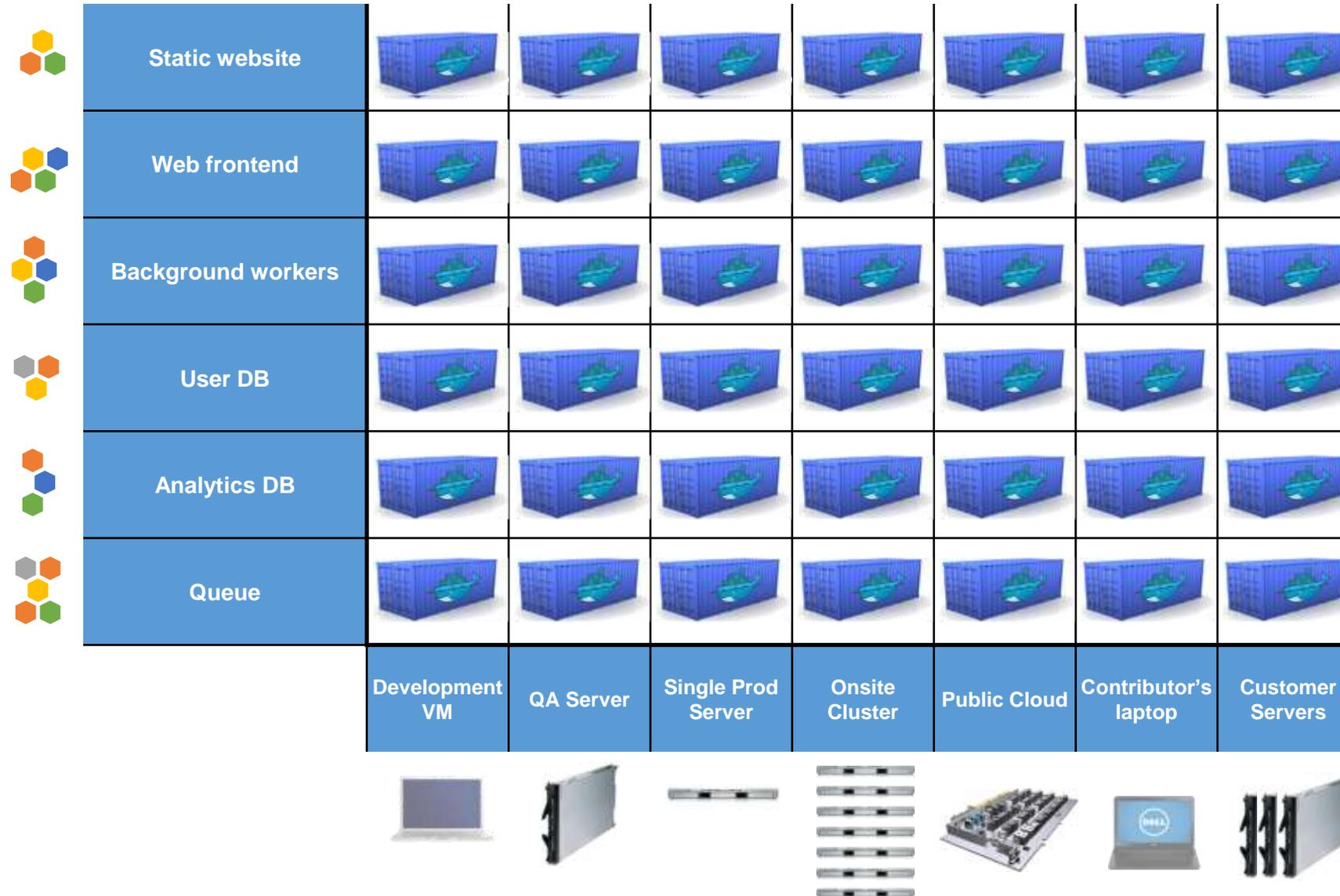
# Docker is a shipping container system for code



# Or...put more simply



# Docker solves the NXN problem



# Contents

---

- The challenge
- The solution
- Why Docker and Containers Matter?
- How They Work?
- Alternative/Complementary Approaches



# Why containers matter

	Physical Containers	Docker
Content Agnostic	The same container can hold almost any type of cargo	Can encapsulate any payload and its dependencies
Hardware Agnostic	Standard shape and interface allow same container to move from ship to train to semi-truck to warehouse to crane without being modified or opened	Using operating system primitives (e.g. LXC) can run consistently on virtually any hardware—VMs, bare metal, openstack, public IAAS, etc.—without modification
Content Isolation and Interaction	No worry about anvils crushing bananas. Containers can be stacked and shipped together	Resource, network, and content isolation. Avoids dependency hell
Automation	Standard interfaces make it easy to automate loading, unloading, moving, etc.	Standard operations to run, start, stop, commit, search, etc. Perfect for devops: CI, CD, autoscaling, hybrid clouds
Highly efficient	No opening or modification, quick to move between waypoints	Lightweight, virtually no perf or start-up penalty, quick to move and manipulate
Separation of duties	Shipper worries about inside of box, carrier worries about outside of box	Developer worries about code. Ops worries about infrastructure.

# Why Developers Care

---

- **Build once...run anywhere**
  - A clean, safe, hygienic and portable runtime environment for your app.
  - No worries about missing dependencies, packages and other pain points during subsequent deployments.
  - Run each app in its own isolated container, so you can run various versions of libraries and other dependencies for each app without worrying
  - Automate testing, integration, packaging...anything you can script
  - Reduce/eliminate concerns about compatibility on different platforms, either your own or your customers.
  - Cheap, zero-penalty containers to deploy services? A VM without the overhead of a VM? Instant replay and reset of image snapshots? That's the power of Docker

# Why Developers Care

---

“Docker interests me because it allows simple environment isolation and repeatability. I can create a run-time environment once, package it up, then run it again on any other machine. Furthermore, everything that runs in that environment is isolated from the underlying host (much like a virtual machine). And best of all, everything is fast and simple.”

-Gregory Szorc, Mozilla Foundation

<http://gregoryszorc.com/blog/2013/05/19/using-docker-to-build-firefox/>

# Why Devops Cares?

---

- **Configure once...run anything**
  - Make the entire lifecycle more efficient, consistent, and repeatable
  - Increase the quality of code produced by developers.
  - Eliminate inconsistencies between development, test, production, and customer environments
  - Support segregation of duties
  - Significantly improves the speed and reliability of continuous deployment and continuous integration systems
  - Because the containers are so lightweight, address significant performance, costs, deployment, and portability issues normally associated with VMs

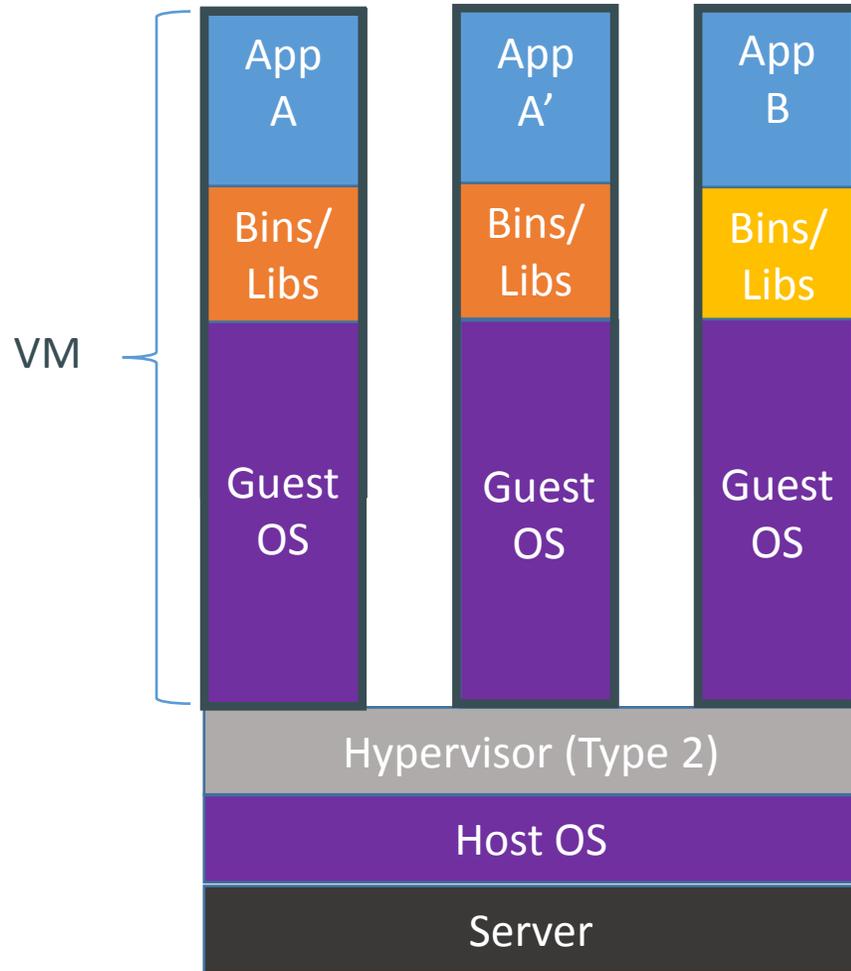
# Contents

---

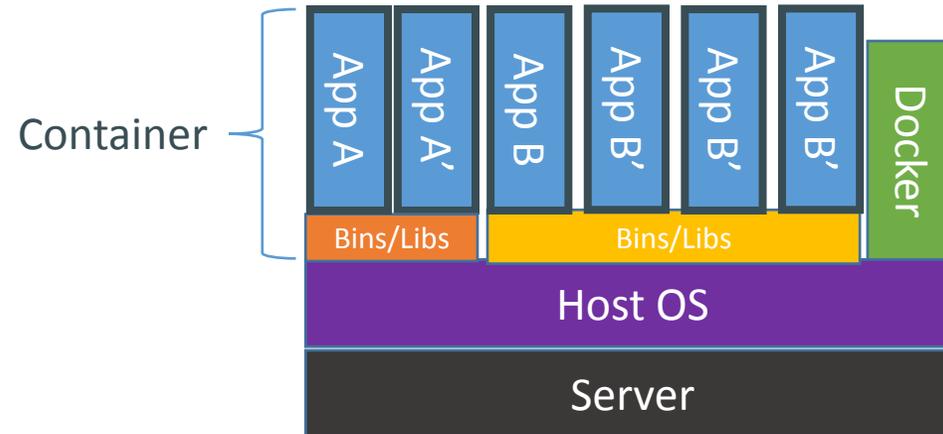
- The challenge
- The solution
- Why Docker and Containers Matter?
- How They Work
- Alternative/Complementary Approaches



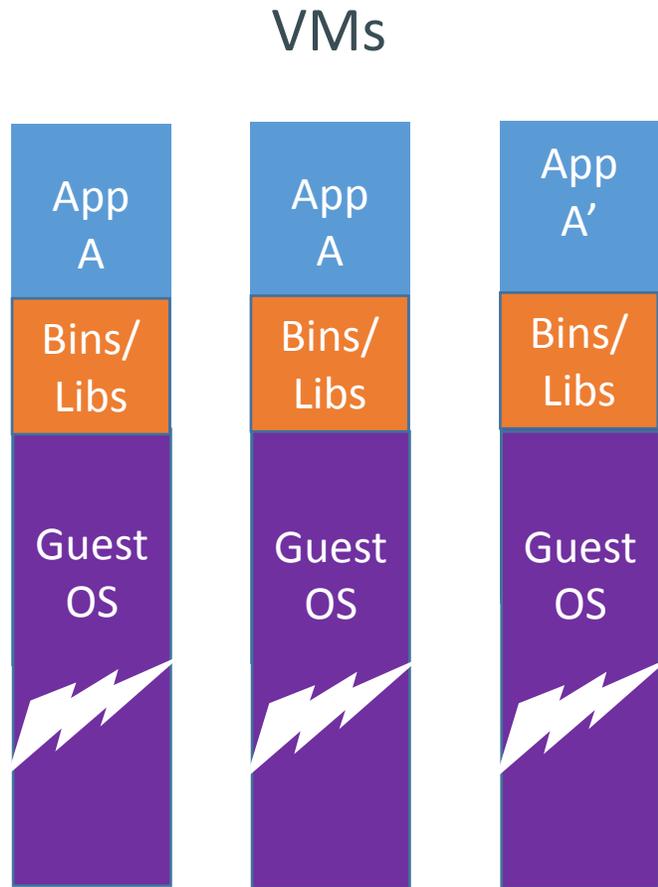
# Containers vs. VMs



Containers are isolated, but share OS and, where appropriate, bins/libraries



# Why are Docker containers lightweight?

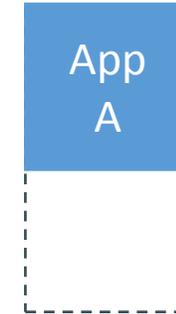


VMs

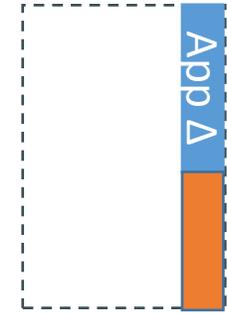
Every app, every copy of an app, and every slight modification of the app requires a new virtual server



Original App  
(No OS to take up space, resources, or require restart)

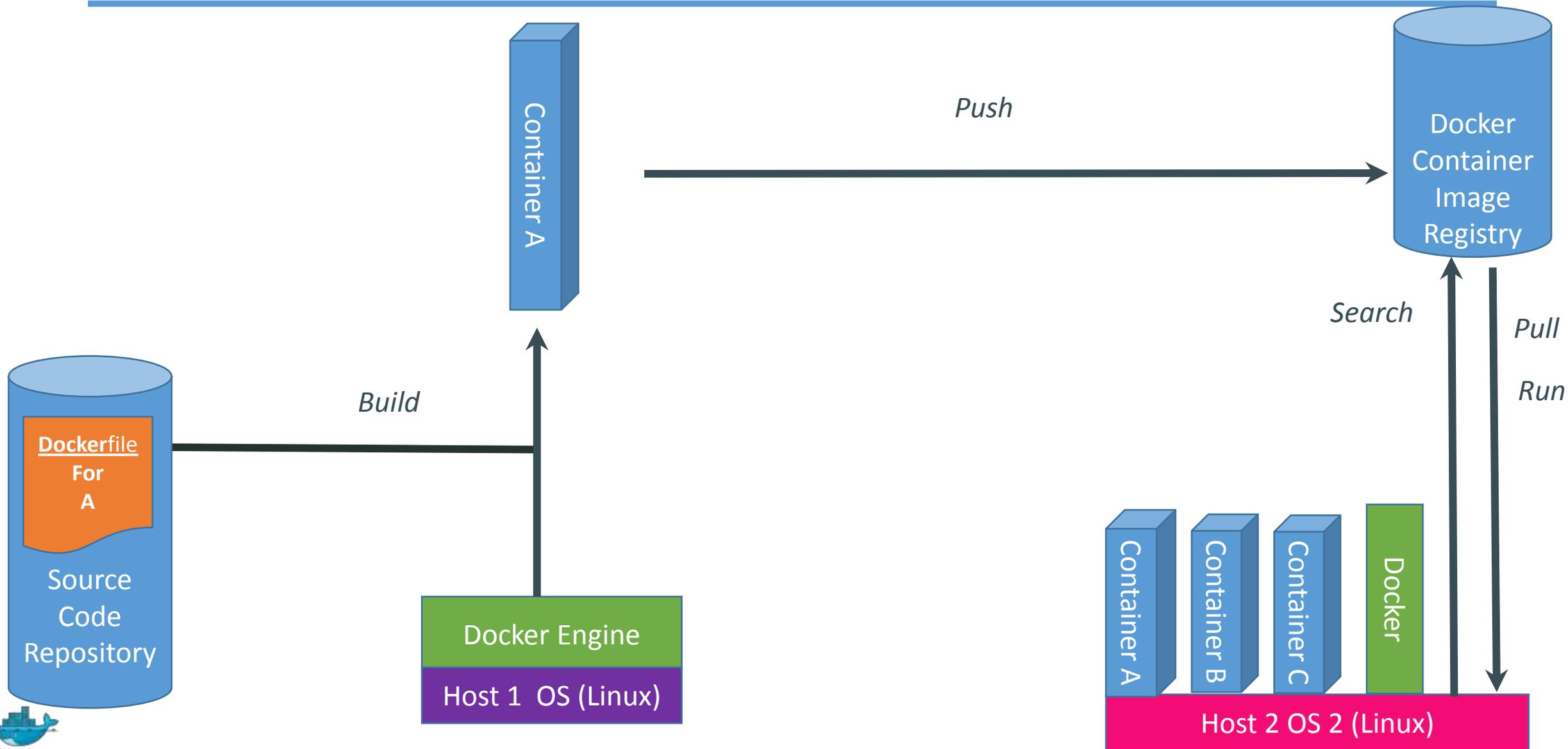


Copy of App  
No OS. Can Share bins/libs

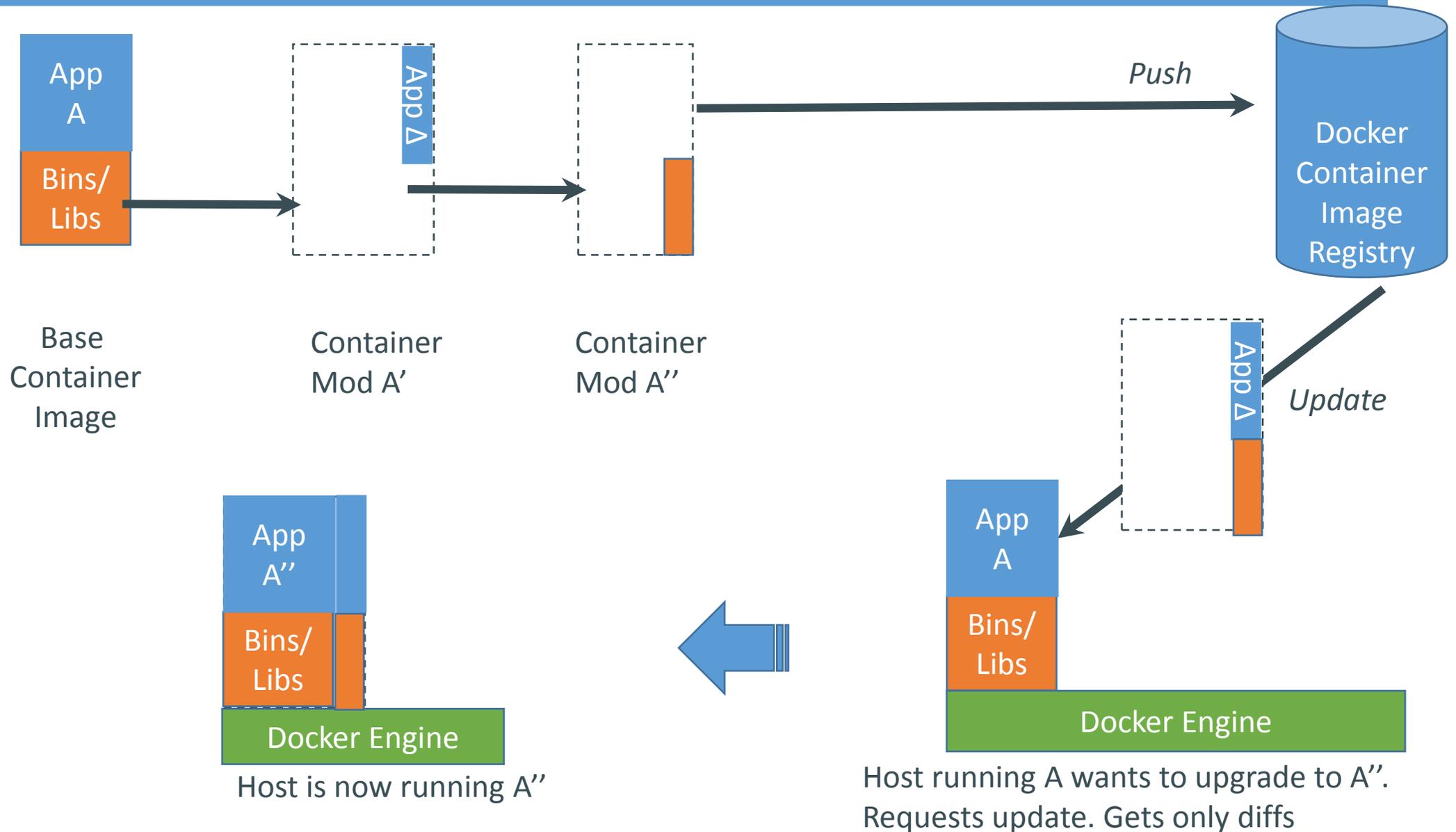


Modified App  
Union file system allows us to only save the diffs Between container A and container A'

# What are the basics of the Docker system?



# Changes and Updates



# Contents

---

- The challenge
- The solution
- Why Docker and Containers Matter?
- How They Work
- Alternative/Complementary Approaches



# Alternatives/Complementary Approaches

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

- Policy  
Reduce Rows
- Configuration Management  
Reduce Columns
- Traditional HW  
Virtualization
- Packaging Automation

# Alternative 1: Impose Consistent Dev Environment

Reduce # rows via policy

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

## Description:

- Try to impose a consistent development environment

## Challenges:

- Goes against 20 years of development trends
- Can't predict what will be needed for next app
- Doesn't work outside confines of the enterprise (e.g. at customer sites)

# Alternative 2: Configuration Mgt/Automation

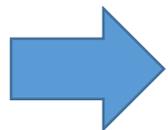
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?
	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers

## Description:

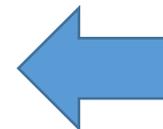
- Automate creation of consistent runtime environment for different machines

## Challenges:

- Chef/Puppet etc. are extremely useful for creating more consistent machine configuration
- But...has to be redone for each new application or version
- Brittle
- Doesn't work easily outside confines of the enterprise (e.g. at customer sites)



Reduce # Columns via Chef/Puppet/etc.



# Alternative 3: Hardware Virtualization

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

## Description:

- Create a virtual server for each app

## Challenges:

- HW Virtualization great for many uses cases (e.g. server consolidation)
- But.. heavyweight/expensive/slow
- Need different VM for different hypervisor environments
- Has to be completely redone for each new application or version
- Not good for scale out, hybrid clouds, massive clustering, iterative development

# Alternative 4: Package Automation

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

## Description:

- Automate creation of different VMs for different

## Challenges:

- A great solution for certain distribution challenges, but...
- VMs are still heavyweight/expensive
- Has to be completely redone for each new application or version
- Better idea: combine containers plus automation

# Use Cases—From Our Community

---

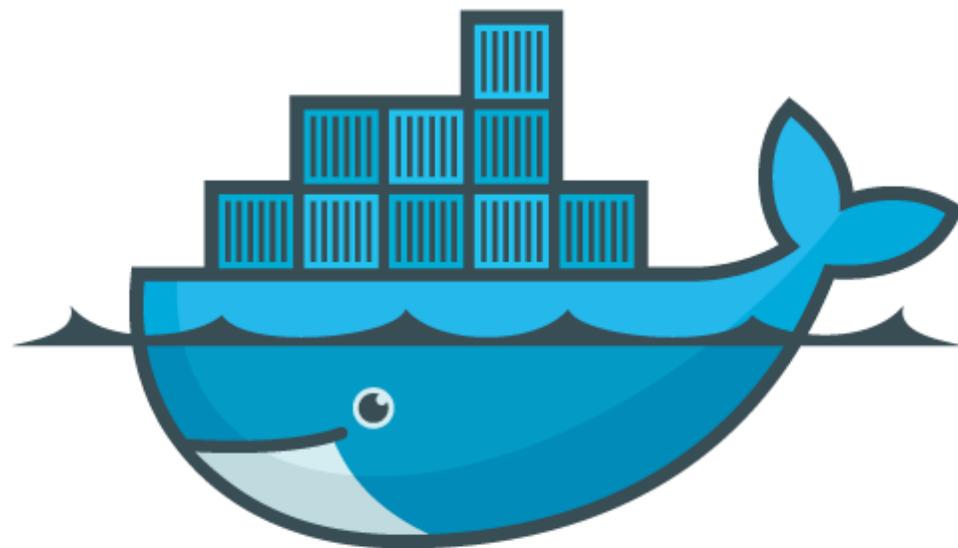
Use Case	Examples	Link
Build your own PaaS	Dokku - Docker powered mini-Heroku. The smallest PaaS implementation you've ever seen	<a href="http://bit.ly/191Tgsx">http://bit.ly/191Tgsx</a>
Web Based Environment for Instruction	JiffyLab – web based environment for the instruction, or lightweight use of, Python and UNIX shell	<a href="http://bit.ly/12oaj2K">http://bit.ly/12oaj2K</a>
Easy Application Deployment	Deploy Java Apps With Docker = Awesome	<a href="http://bit.ly/11BCvву">http://bit.ly/11BCvву</a>
	Running Drupal on Docker	<a href="http://bit.ly/15MJS6B">http://bit.ly/15MJS6B</a>
	Installing Redis on Docker	<a href="http://bit.ly/16EWOKh">http://bit.ly/16EWOKh</a>
Create Secure Sandboxes	Docker makes creating secure sandboxes easier than ever	<a href="http://bit.ly/13mZGJH">http://bit.ly/13mZGJH</a>
Create your own SaaS	Memcached as a Service	<a href="http://bit.ly/11nL8vh">http://bit.ly/11nL8vh</a>
Automated Application Deployment	Push-button Deployment with Docker	<a href="http://bit.ly/1bTKZTo">http://bit.ly/1bTKZTo</a>
Continuous Integration and Deployment	Next Generation Continuous Integration & Deployment with dotCloud's Docker and Strider	<a href="http://bit.ly/ZwTfoу">http://bit.ly/ZwTfoу</a>
Lightweight Desktop Virtualization	<a href="http://bit.ly/14RYL6x">Docker Desktop: Your Desktop Over SSH Running Inside Of A Docker Container</a>	<a href="http://bit.ly/14RYL6x">http://bit.ly/14RYL6x</a>



# Want to learn more?

---

- San Francisco Meetup: July 30
- Boston Openstack Meetup: August 14
- New York Meetup: Aug 21
- Austin Openstack Meetup: Sept 12
- Boston Docker Meetup: Sept 23
- [www.docker.io](http://www.docker.io)



docker

[www.docker.io](http://www.docker.io)