The Future of Distributed Applications
Agenda

• The Future of Applications: Distributed Apps
• The Matrix from Hell
• Docker & Containers: how they work
• Docker Overview and Docker Ecosystem
• Current Usage and Adoption
• Creating an Open Environment
• Docker Roadmap & Business Model
The One Thing You Need to Know About the Future of Applications?

Developers are content creators
What Happens When You Separate the Act of Creation from Concerns about Production & Distributions?
### Apps Have Fundamentally Changed

<table>
<thead>
<tr>
<th>~2000</th>
<th>2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long lived</td>
<td>Development is iterative and constant</td>
</tr>
<tr>
<td>Monolithic and built on a single stack</td>
<td>Built from loosely coupled components</td>
</tr>
<tr>
<td>Deployed to a single server</td>
<td>Deployed to a multitude of servers</td>
</tr>
</tbody>
</table>
The Problem in 2014: Distributed Applications

- **Static website**
  - nginx 1.5 + modsecurity + openssl + bootstrap 2

- **Background workers**
  - Python 3.0 + celery + pyredis + curl + ffmpeg + libopencv + nodejs + phantomjs

- **Web frontend**
  - Ruby + Rails + sass + Unicorn

- **User DB**
  - postgresql + pgv8 + v8

- **Queue**
  - Redis + redis-sentinel

- **Analytics DB**
  - hadoop + hive + thrift + OpenJDK

- **API endpoint**
  - Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client

---

**Development VM**

**QA server**

**Customer Data Center**

**Production Servers**

**Production Cluster**

**Public Cloud**

**Disaster recovery**

**Contributor’s laptop**

---

**Multiplicity of Stacks**

**Multiplicity of hardware environments**

---

Do services and apps interact appropriately?

Can I migrate smoothly and quickly?
|----------------|----------------|-----------|--------------------|----------------|--------------|----------------------|-----------------|
An Inspiration... and some really ancient history: Cargo Transport Pre-1960

Multiplicity of Goods

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/storage.
Solution: Intermodal Shipping Container Ecosystem

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 Goods

Multiplicity of methods for transporting/storing
This 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 globalization
- 5000 ships deliver 200M containers per year
Let’s create an **ecosystem for distributed applications**

- Static website
- User DB
- Web frontend
- Queue
- Analytics DB

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 Stacks

- Development VM
- QA server
- Customer Data Center
- Public Cloud
- Production Cluster
- Contributor’s laptop

Do services and apps interact appropriately?

Can I migrate smoothly and quickly?
And eliminate the matrix from Hell

<table>
<thead>
<tr>
<th>Static website</th>
<th>Web frontend</th>
<th>Background workers</th>
<th>User DB</th>
<th>Analytics DB</th>
<th>Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development VM</td>
<td>QA Server</td>
<td>Single Prod Server</td>
<td>Onsite Cluster</td>
<td>Public Cloud</td>
<td>Contributor’s laptop</td>
</tr>
</tbody>
</table>

And eliminate the matrix from Hell
...and enable the future of Distributed Applications
The future of distributed apps: Six Easy Steps

Create lightweight container

Make containers standard, interoperable, easy to use

Make them easy to update, layered, ultra-lightweight

Create an ecosystem

 Enable a multi-Docker app model

Create a platform for managing it all
Step One: Create a lightweight container (vs. VMs)

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

...result is significantly faster deployment, much less overhead, easier migration, faster restart
Step 2: Make the containers easy to use, standardized, interoperable, automatable

- Container technology has been around for a while (c.f. LXC, Solaris Zones, BSD Jails)
  - But, their use was largely limited to specialized organizations, with special tools & training. Containers were not portable
- Analogy: Shipping containers are not just steel boxes.
  - They are steel boxes that are a standard size, and have hooks and hole in all the same places, and have an ecosystem of ships, trains, trucks, and cranes.
- With Docker, low level containers get the following:
  - Ease of use, tooling
  - Re-usable components
  - Ability to run on any Linux server today: physical, virtual, VM, cloud, OpenStack, +++
    - (Stay tuned for other O/S’s)
  - Ability to move between any of the above in a matter of seconds-no modification or delay
  - Ability to share containerized components
  - Interoperability with all existing devops tools
  - Self contained environment—no dependency hell
  - Tools for how containers work together: linking, nesting, discovery, orchestration, ++
- “Containerization” is really “Dockerization”
Step 3: Make containers super lightweight, stackable

**VMs**

- **App A**
  - Bins/Libs
  - Guest OS

- **App A**
  - Bins/Libs
  - Guest OS

- **App A’**
  - Bins/Libs
  - Guest OS

**Containers**

- **App A**
  - Bins/Libs

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

**Copy of App**
No OS. Can share bins/libs

**Modified App**
Copy on write allows us to only save the diffs between container A and container A’

Every app, every copy of an app, and every slight modification of the app requires a new virtual server.
A set of slides from IBM: Comparing Docker to VMs

Cloudy Performance: Steady State Packing

Cloudy Performance: Serial VM Boot

Cloudy Performance: Serial VM Boot

Cloudy Performance: Serial VM Reboot

Cloudy Performance: Snapshot VM To Image

In Summary

- Near bare metal performance in the guest
- Fast operations in the Cloud
  - Often capped by Cloud management framework
- Reduced resource consumption (CPU, MEM) on the compute node – greater density
- Out of the box smaller image footprint
Step 4: Build an Ecosystem

Community
- 600+ Contributors
- 250+ Meetups on Docker
- 13M Downloads
- 12K Projects on GitHub

Support
- Enterprise Support
- Robust Documentation
- Implementation, Integration, Training
- Network of Partners

The Docker Platform
- Docker Engine
- Docker Hub
- Build, Ship, and Run

Content
- Official Repos & 30K Dockerized Apps

Users
- ebay
- Spotify
- GILT
- New Relic

Partners
- Rackspace
- IBM
- AWS
- Chef
- SaltStack
- Puppet

---

Step 4: Build an Ecosystem

Official Repos & 14K+ Dockerized Apps

Community
- 600+ Contributors
- 250+ Meetups on Docker
- 13M Downloads
- 12K Projects on GitHub

Support
- Enterprise Support
- Robust Documentation
- Implementation, Integration, Training
- Network of Partners

The Docker Platform
- Docker Engine
- Docker Hub
- Build, Ship, and Run

Content
- Official Repos & 30K Dockerized Apps

Users
- ebay
- Spotify
- GILT
- New Relic

Partners
- Rackspace
- IBM
- AWS
- Chef
- SaltStack
- Puppet
Partner Ecosystem

Service Providers

Operating Systems

Configuration Management

Big Data

Dev Tools

Service Discovery

Official Repositories

Orchestration

System Integrators
Step 5: Build a Framework for Multi-Docker Apps

**Multi-Container**

- Single Logical Service
- Tomcat Container ➔ MySQL Container
- Host

**Multi-Cluster**

- Single Logical Service
- Tomcat Container ➔ MySQL Container
- Host

**Multi-Site**

- Single Logical Service
- Tomcat Container ➔ MySQL Container
- Host

- On-Premise
  - Host
  - Cluster A ➔ Cluster B
  - Host

- Public Cloud
  - Host
Step 6: Build an open platform

An Open Platform to **Build**, **Ship**, and **Run** Distributed Applications
An Open Platform…

**Engine**
open source software at the heart of the Docker platform

**Hub**
cloud-based platform services for distributed applications
An Open Platform…

Any App
+ 30K apps
+ 12K projects

Engine
open source software at the heart of the Docker platform

Hub
cloud-based platform services for distributed applications

Any infrastructure
- Physical
- Virtual cloud
...to Build, Ship, and Run
Distributed Applications

- User DB
- Web Frontend
- Background Workers
- Analytics DB
- API Endpoint
- Development VM
- QA Server
- Public Cloud
- Customer Data Center
- Production Servers
- Production Cluster
- Contributor's Laptop
- Disaster Recovery
Who is using Docker?

E-Commerce
- eBay
- GILT
- Groupon

Media
- Disney
- Spotify
- Riot Games

Life Sciences
- Illumina
- Cambridge Healthcare

Intelligence

Vertical Search
- Yelp
- OpenTable
- autocom

IT SaaS
- Atlassian
- New Relic
- AppDynamics

IaaS
- Google Cloud Platform
- Rackspace
- SoftLayer

PaaS
- Red Hat
- Yandex

...and thousands of others!
Learn More

• All Videos and Talks from DockerCon: http://bit.ly/1AnvHFz
• Demo of DockerHub: see Ben Golub keynote from DockerCon, http://bit.ly/1xHqUL8 demo starts at 29:00
• Sign up for a DockerHub account: https://hub.docker.com/
• Security & Isolation: see above, also blog.docker.io
• Use Cases: https://docker.com/resources/usecases/
• Docker project: www.docker.com/
• Follow Docker on Twitter: twitter.com/docker
• Take the Docker interactive tutorial: https://docker.com/tryit/
• Join Docker on IRC: botbot.me/freenode/docker/
• Go to the Docker repository on GitHub: github.com/docker/docker/
• Go to a meetup: https://www.docker.com/community/meetups/