CONTAINERS, CONTAINERISATION, AND CONTAINER ORCHESTRATION
THE WHAT, THE WHY, AND THE HOW

PHILIP NORMAN (@philipnrnmn)
SUNIL SHAH (@ssk2)
BACKGROUND & MOTIVATION
Operating-system-level virtualisation is a server virtualisation method in which the kernel of an operating system allows the existence of multiple isolated user-space instances, instead of just one. Such instances, which are sometimes called containers, software containers, virtualisation engines (VEs) or jails (FreeBSD jail or chroot jail), may look and feel like a real server from the point of view of its owners and users.

— Wikipedia article on Software Containers
WHO USES THEM?

Background & Motivation
WHO USES THEM?

Background & Motivation
WHY USE THEM?

Background & Motivation

- Efficient (cheaper than virtualisation)
- Guaranteed to be identical
- Reproducible
- Isolated

Google’s doing it (GIFEE):

Google and Containers

*Everything* at Google runs in a container.

Internal usage:
- Resource isolation and predictability
- Quality of Services
  - batch vs. latency sensitive serving
- Overcommitment (not for GCE)
- Resource Accounting

We start over 2 billion containers per week.

(from Containers At Scale by Joe Beda)
WHY NOT USE THEM?

You’re sharing physical resources: container apps can suffer from ‘noisy neighbours’
WHAT IS A CONTAINER?
Containers have two high level features:

1. Operating system level virtualisation (which allows applications to be isolated)
2. Dependency management

The application typically shouldn’t (and doesn’t) know that it is running within a container.
In this talk, we’ll refer to isolation as

*The ability of two applications to run together without the operations of one interfering with those of another.*
Alice and Bob are developers working at the same company.
What is a container?

**AN EXAMPLE**

They each want to deploy their own application to a production server.
What is a container?

AN EXAMPLE

Alice trusts Bob but is nervous about running her application on the same server.
What happens if Bob’s application is buggy and uses more memory than it should?
One example of container resource isolation is enforcing memory constraints.
What is a container?

AN EXAMPLE

If App B attempts to use more memory than it should, it will simply get terminated. App A remains safe and Alice doesn’t need to worry.
There are many different facets of isolation provided by operating system level virtualisation, not all of which are implemented by every container tool.

Some examples:

- Resources (CPU / memory)
- Disk quotas
- Network
- I/O
- Filesystem
Operating system level virtualisation is different to host level virtualisation.

The underlying primitives that provide isolation functionality must be built into the kernel.

What is a container?
**OPERATING SYSTEM LEVEL ISOLATION**

**What is a container?**

- **Virtual Machine–Based Application Deployment**
  - Isolate apps by running multiple VMs per physical server; still need to manage each guest OS!

- **Container–Based Application Deployment**
  - Isolate apps using features of the host OS, such as Linux cgroups.
The second feature of modern container systems is their ability to manage dependencies.

Containers run in isolation with their own view of the filesystem.
What is a container?

**DEPENDENCY MANAGEMENT**

**Virtual Machines**

- Application
- Dependencies
- Guest OS
- VM
- Hypervisor

**Container w/ dependencies**

- Application
- Dependencies
- Container
- Container Engine
- Host OS
- Linux Host OS

**Container w/ out dependencies**

- Application
- Linux cgroup
- Dependencies
- Host OS
- Linux Host OS
What is a container?

**DEPENDENCY MANAGEMENT**

- **App A**
  - Python 2.7
  - Container

- **App B**
  - Python 3.4
  - Container

**Container Engine**

**Host OS**
CONTAINERS IN REAL LIFE
**Containers in Real Life**

**SOME HISTORY**

- **1979**: UNIX chroot
  - Solves dependency control

- **2000**: BSD jails
  - Improves security

- **2005**: Solaris zones
  - Adds system resource controls

- **2008**: Linux LXC
  - Containers on Linux

- **2013**: Docker released
  - Composable, configurable LXC

- **2015**: OCI announced
  - Industry standards for containers
Let’s talk a little bit about the most popular container format and runtime: Docker (since practically this is what most tools operate with)
RUNNING CONTAINERS AT SCALE
Container orchestration exploits the benefits of containers by using them to manage structured applications across a large pool of computing resources.
Container orchestration helps abstract the complexity of working in a distributed environment.

- deploy 5 servers
- deploy 1 database
Container orchestration provides an API so the datacenter operator can automate monitoring and scaling.

- deploy 5 servers
- deploy 1 database
- restart any server when unhealthy
- page me when database unhealthy
Popular tools (non-exhaustive):

- **CloudFoundry** (Pivotal)
- **Datacenter Operating System** (Mesosphere)
- **Docker Swarm**
- **Fleet** (CoreOS)
- **Kubernetes** (Google)
DATACENTER OPERATING SYSTEM (DC/OS)

DC/OS is an open source “batteries included” container orchestration system that is built upon the Apache Mesos project.

Apache Mesos is:
- Designed for efficient resource utilisation (bin packing of containers running on agents)
- Highly scalable (10,000+ machines)
- Production proven (runs much of Twitter, Siri and others)
- Open source
DC/OS also provides:

- An easy installer
- A user friendly web interface
- A powerful command line tool
- A package registry to install popular distributed systems and applications
- Advanced operator features to help manage applications at scale
- Various APIs to allow you to programmatically manage your cluster

We’ll see DC/OS in action during our demo.
Container Orchestration

MESOS ARCHITECTURE

Mesos Master Quorum

Mesos Master
(Leader)

Mesos Master
(Standby)

Mesos Master
(Standby)

Mesos Master
(Standby)

Framework A
Scheduler

Framework B
Scheduler

Agent 1

Agent n

Framework A
Executor

Framework B
Executor

Resource offer

Launch task(s)

Resource offer

Launch task(s)
git clone https://github.com/philipnrnmn/os101-demo.git
cd os101-demo/appA
docker build -t philipnrnmn/os101-demo:appA .
# run locally
docker run -p 80:5000 philipnrnmn/os101-demo:appA
docker push -p 80:5000 philipnrnmn/os101-demo:appA

*replace `philipnrnmn/os101-demo` with your own DockerHub repository*
Navigate over to a DC/OS cluster!

Services > Services > Run a Service
Thank you! Please come and chat to us!

Check out our websites:
- mesosphere.com
- dcos.io

Learn more about container formats:
- Docker.io
- OCI

And view slides at:
https://mesosphere.github.io/presentations