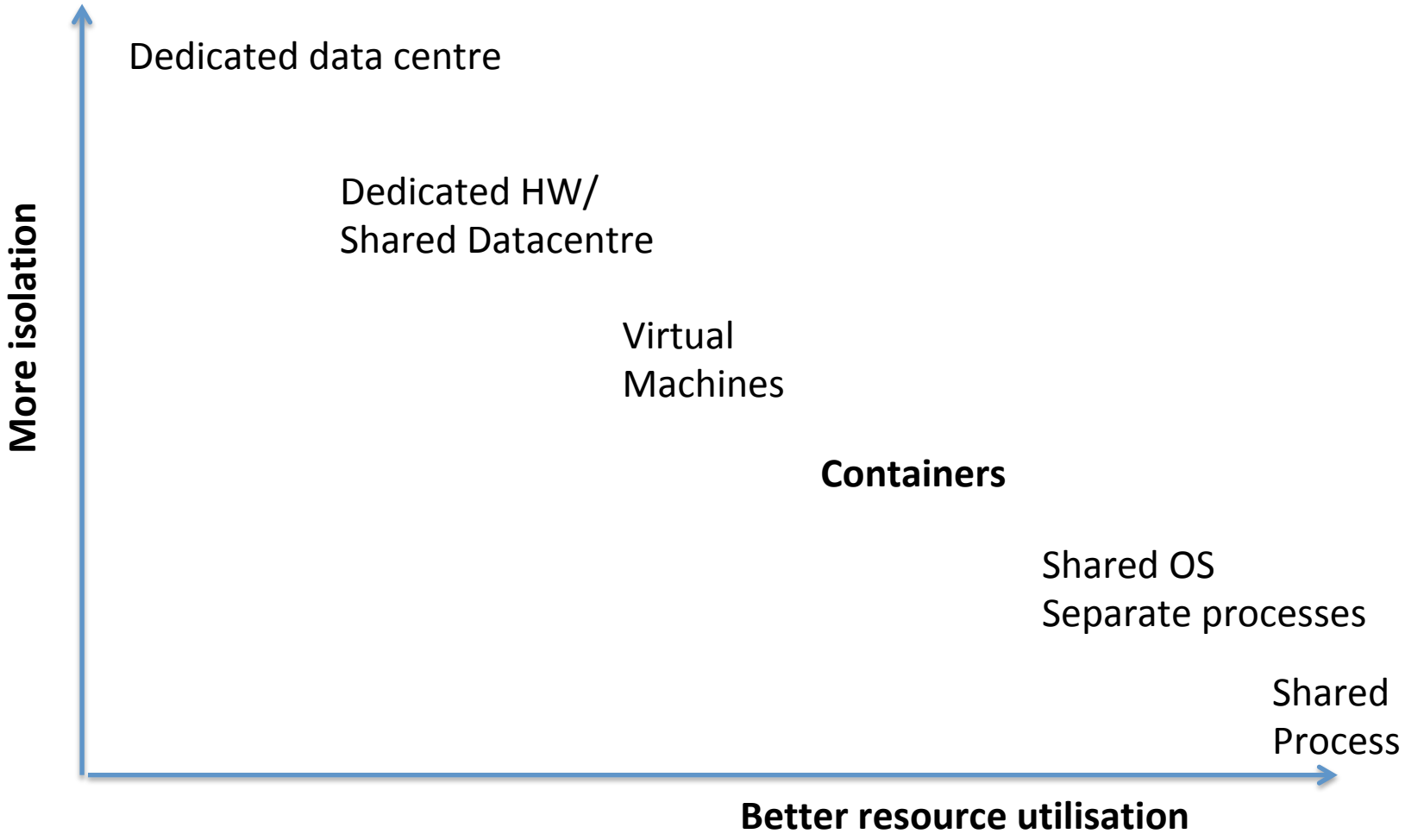
Containers, Cloud Native Computing, DevOps

Oxford University
Software Engineering
Programme
January 2018

Contents

- Containers
- History and Approach
- Docker
- Docker ecosystem
- PaaS in a container model
- Futures

Sharing of resources vs Isolation



Lightweight Virtualization history

- zSystems Virtual Servers from late 1990s
  - (the mainframe really did do everything first)
- Solaris Containers
- AIX Workload Partitions
- FreeBSD Jail
- …

What is a Container?

- A lightweight virtual server
  - Running within an Operating System
  - Providing various levels of isolation and control
  - E.g. Disk isolation and control
  - Network isolation
  - CPU and memory controls

Containers at Google

- Every GMail session is a container
  - Try doing an export and then searching your email ☺
- "Everything runs in a container"
- **2 billion** containers launched a week
- Borg
  - **Any** Google developer can instantiate their code in **10,000 instances** any time they want
  - Takes about 5 minutes to start that many
  - Never exactly 10,000 because of failures

Linux Containers (LXC)

- Virtualization inside the Linux Operating System
  - Not the only Linux option, but the most popular
- Allows virtualization including CPU, memory, disk
- Simple and effective

A sample of LXC

- apt-get install lxc
- lxc-create -t ubuntu -n cn-01
- lxc-start –n cn-01
- lxc-console –n cn-01
- lxc-freeze –n cn-01

cgroups

- Control of resources by process:
  - blkio — this subsystem sets limits block devices such as physical drives
  - cpu - access to the CPU.
  - cpuacct — this reports on CPU usage
  - cpuset — this controls usage by CPUs in a multicore
  - devices — this denies or grants access to devices
  - freezer — suspends and resumes tasks
  - memory — controls and reports on memory usage
  - net_cls — tags network packets with ids for control
  - net_prio — priority of network traffic per interface.
  - ns — the namespace subsystem.

Alternatives to LXC

- xhyve – OS/X
- bhyve – BSD containers
- rkt – CoreOS Rocket (aiming at Docker)

libcontainer and the Open Container Foundation

- A standardised interface into the container layer
  - Part of runC the open runtime from Docker
  - A key basis of the Open Container Foundation

Cloud Native

MAY
28

Cloud Native

Together with Sanjiva and the rest of the WSO2 architecture team, I've been thinking a lot about what it means for applications and middleware to work well in a cloud environment - on top of an Infrastructure-as-a-Service such as Amazon EC2, Eucalyptus, or Ubuntu Enterprise Cloud.
One of our team - Lavi - has a great analogy. Think of a 6-lane freeway/motorway/autobahn as the infrastructure. Before the autobahn existed there were forms of transport optimized first to dirt tracks and then to simple tarmac roads. The horse-drawn cart is optimized to a dirt track. On an autobahn it works - but it doesn't go any faster than on a dirt track. A Ford Model T can go faster, but it can't go safely at autobahn speeds: even if it could accelerate to 100mph it won't steer well enough at that speed or brake quickly enough.

Similarly, existing applications taken and run in a cloud environment may not fully utilize that environment. Even if systems can be clustered they may not be able to dynamically change the cluster size (elasticity). Its not just acceleration, but braking as well! We believe there are a set of these technical attributes that software needs to take account of to work well in a cloud environment. In other words - what do middleware and applications have to do to be ***Cloud Native***.

Cloud Native Computing Foundation

- A new definition of “Cloud Native”
  - Container Packaged
  - Dynamically Managed
  - Micro-Service oriented

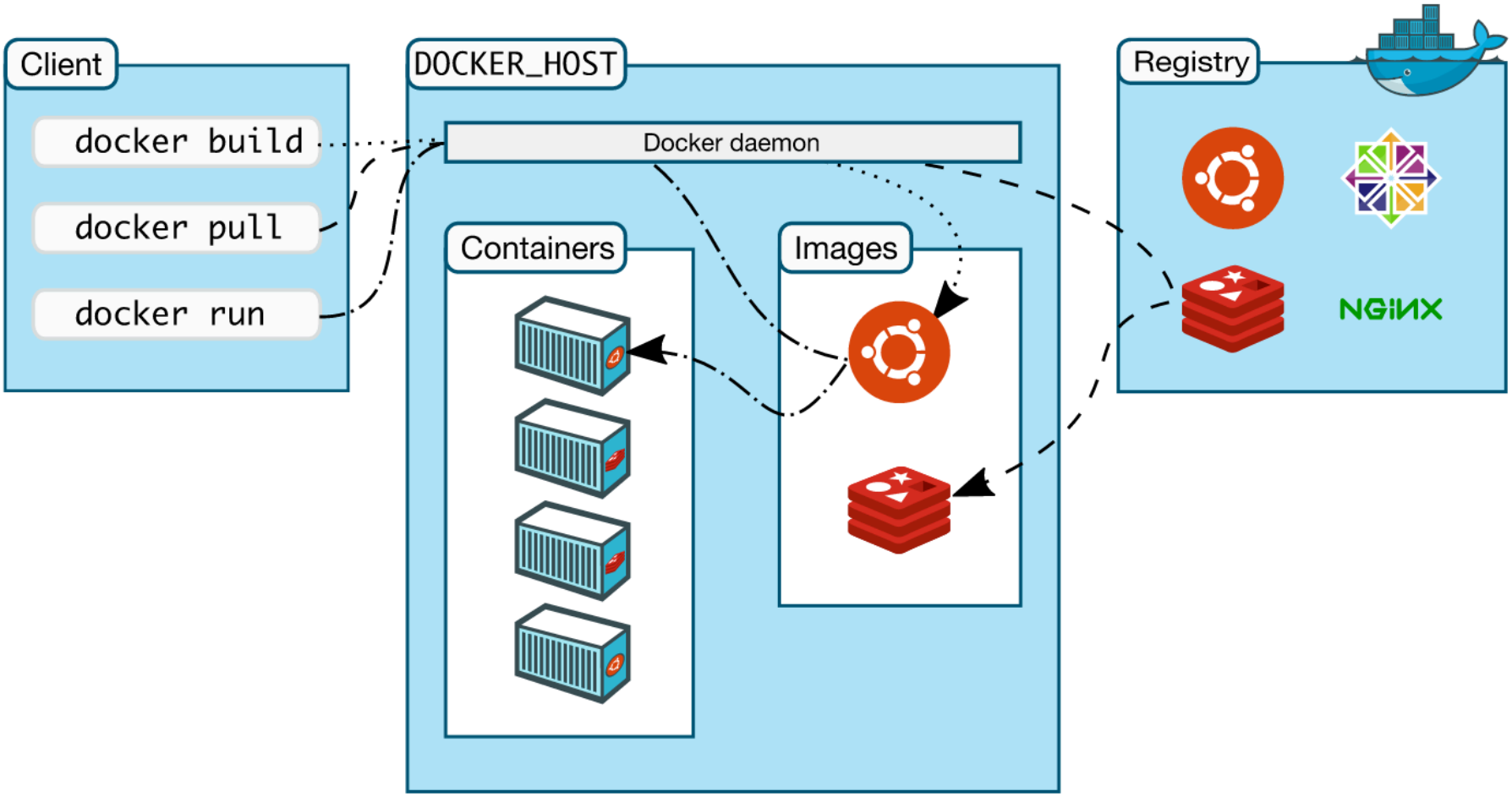
Docker on top of LXC

- Docker adds several things to LXC and containerization:
  - Copy on write filesystem
    - Layered images and the ability to extend machines easily
  - Simple textual config file
  - Portable deployment across machines
    - Creating an ecosystem of images
  - Application centric
    - Each VM is a process (roughly speaking)
  - Plus others (auto-build, etc)

Why Docker?

- The *ecosystem* has created a *network effect*
- Metcalfe's Law states
  - the value of a telecommunications network is proportional to the square of the number of connected users of the system
- There is surely a corollary for ecosystems

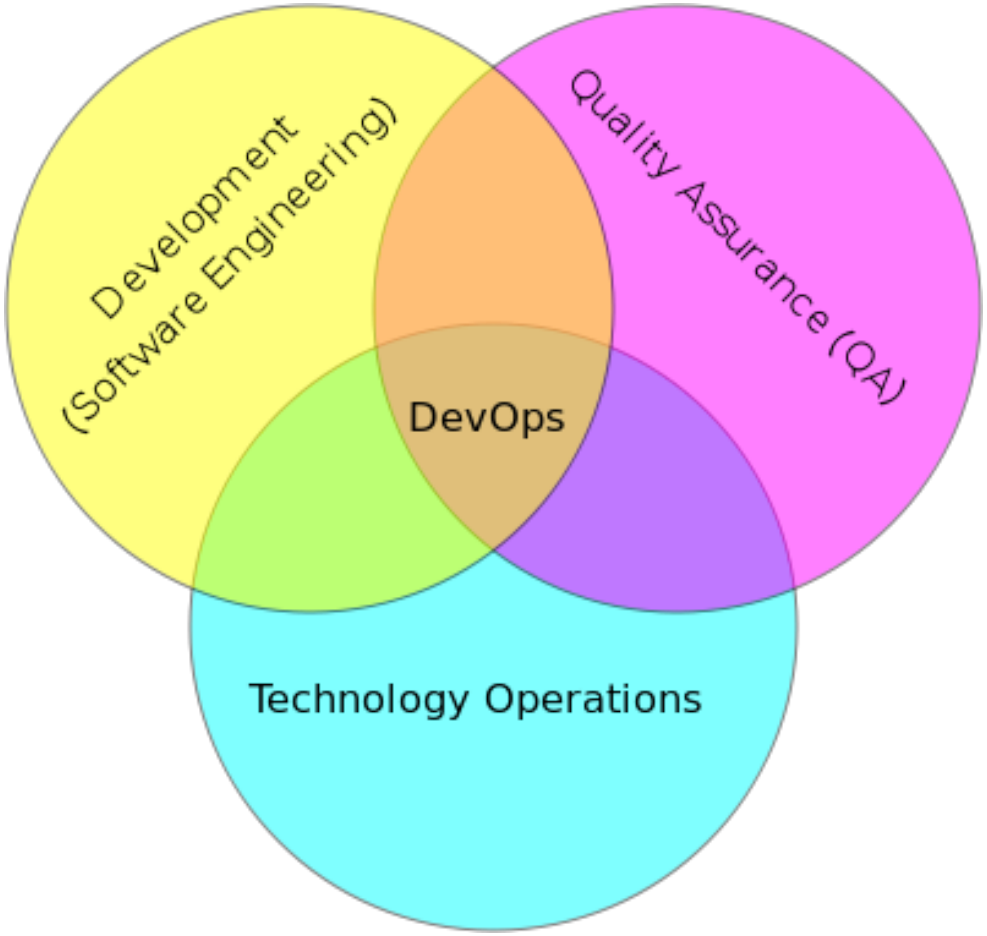
How does Docker work?



Some simple Docker commands

- apt-get install docker.io
- docker pull ubuntu
- docker run –t –i ubuntu /bin/bash
- docker ps
- docker commit funky_freo image
- docker push image

DevOps



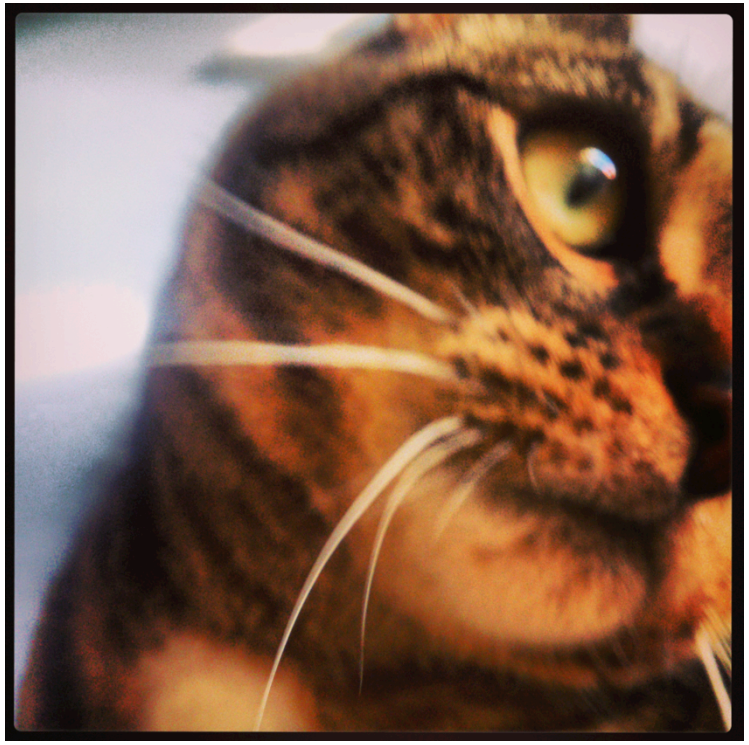
DevOps

- DevOps is the codification of the interface between Development and Operations
  - Agile
  - Repeatable
  - Collaborative
  - Versioned
  - Automated

Cloud and DevOps

- It could be argued strongly that the rise of DevOps is tied to the rise of Cloud
  - Clear requirement for automated, repeatable configuration and deployment
  - Reducing the hardware provisioning time has highlighted the challenges

Kittens vs Cattle
(An unpleasant but effective analogy)

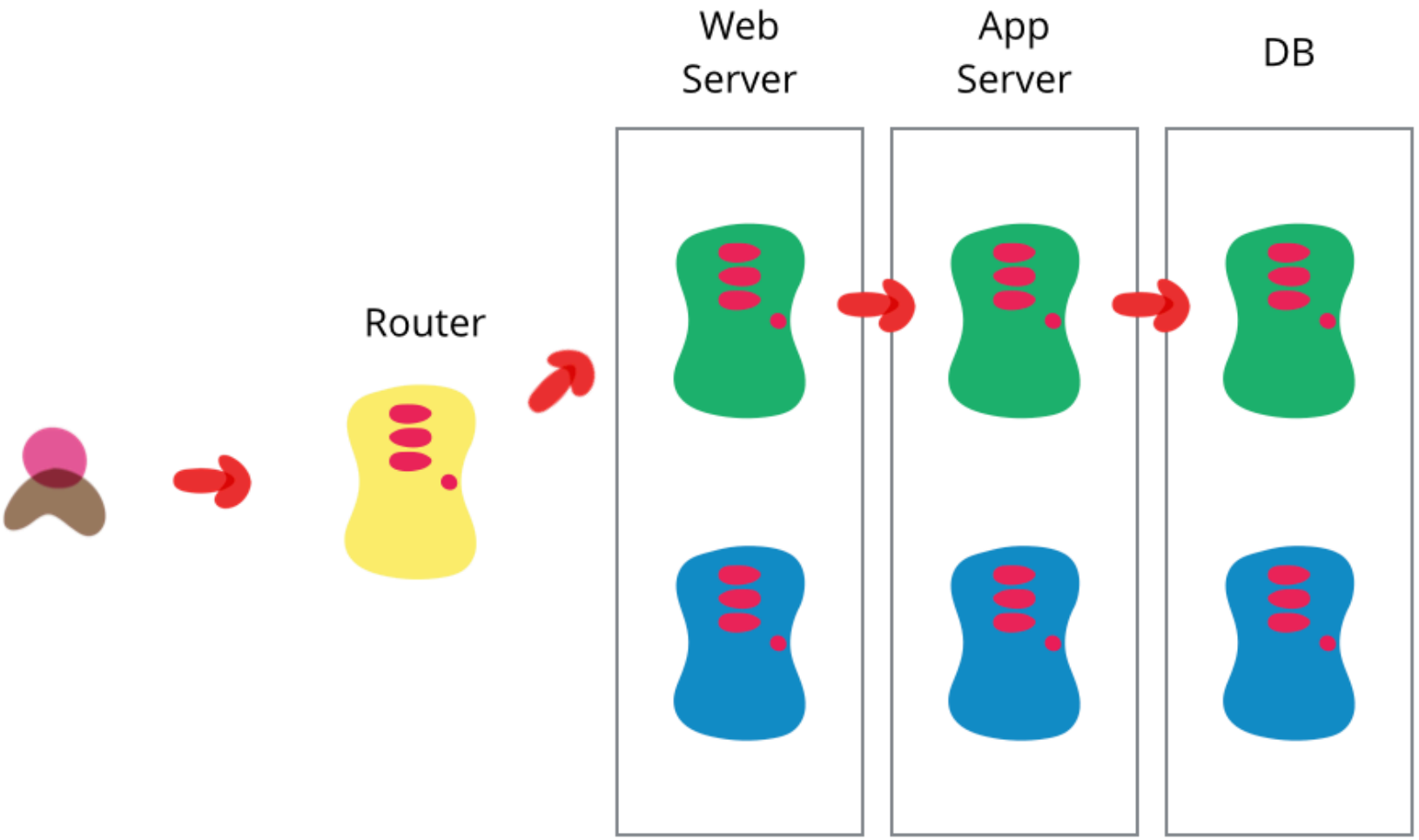


Immutable objects

- String in Java
- You can't change a String, only create a new one

- Applies to devops:
  - Never change a running server
  - Only create a new one that is better
  - Track the changes in a version control model

Blue Green Deployment



http://martinfowler.com/bliki/BlueGreenDeployment.html

DevOps tools

- Puppet, Chef
  - Automated configuration and deployment tools
  - Allow complex infrastructures to be re-configured automatically
- Vagrant
  - Create VMs instantly
- Plus many many more!

DevOps and Docker

- Docker is a key DevOps tool
- Speeds up the creation of repeatable deployments
- Consistency between development, test and production
- Versioned repository
- Works with Chef, Puppet, etc

Challenges with Docker and Solutions

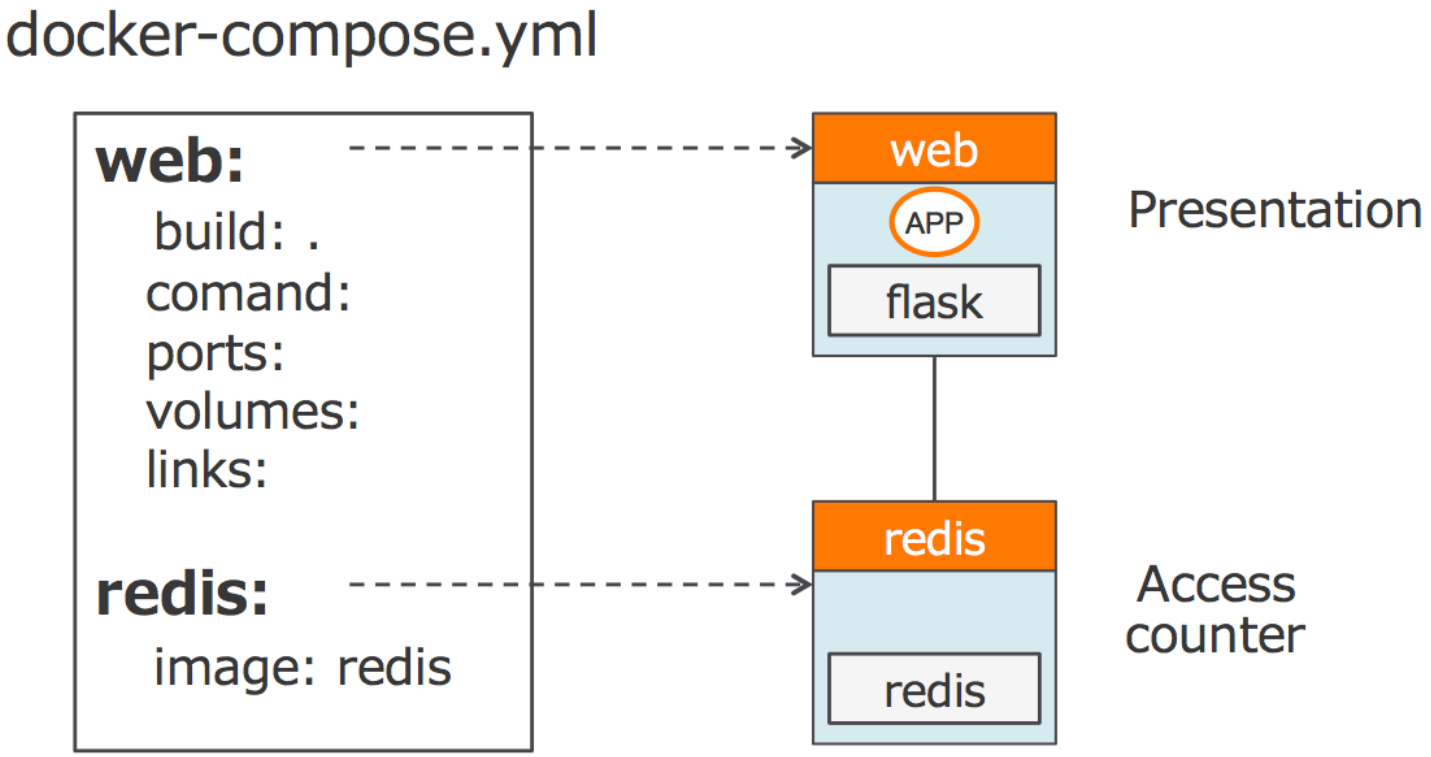
- Networking
  - It is very complex to connect different containers, even on a single machine
    - Weave Networks
    - SocketPlane (bought by Docker)
- Clustering
  - Docker Swarm
  - Google Kubernetes
  - CoreOS
  - Apache Mesos
- Lack of mutable file system
  - Flocker

Docker Machine

- A useful tool for creating and managing docker host machines
- Works with swarm
- Adapters for AWS, DigitalOcean, VirtualBox, vmWare, etc

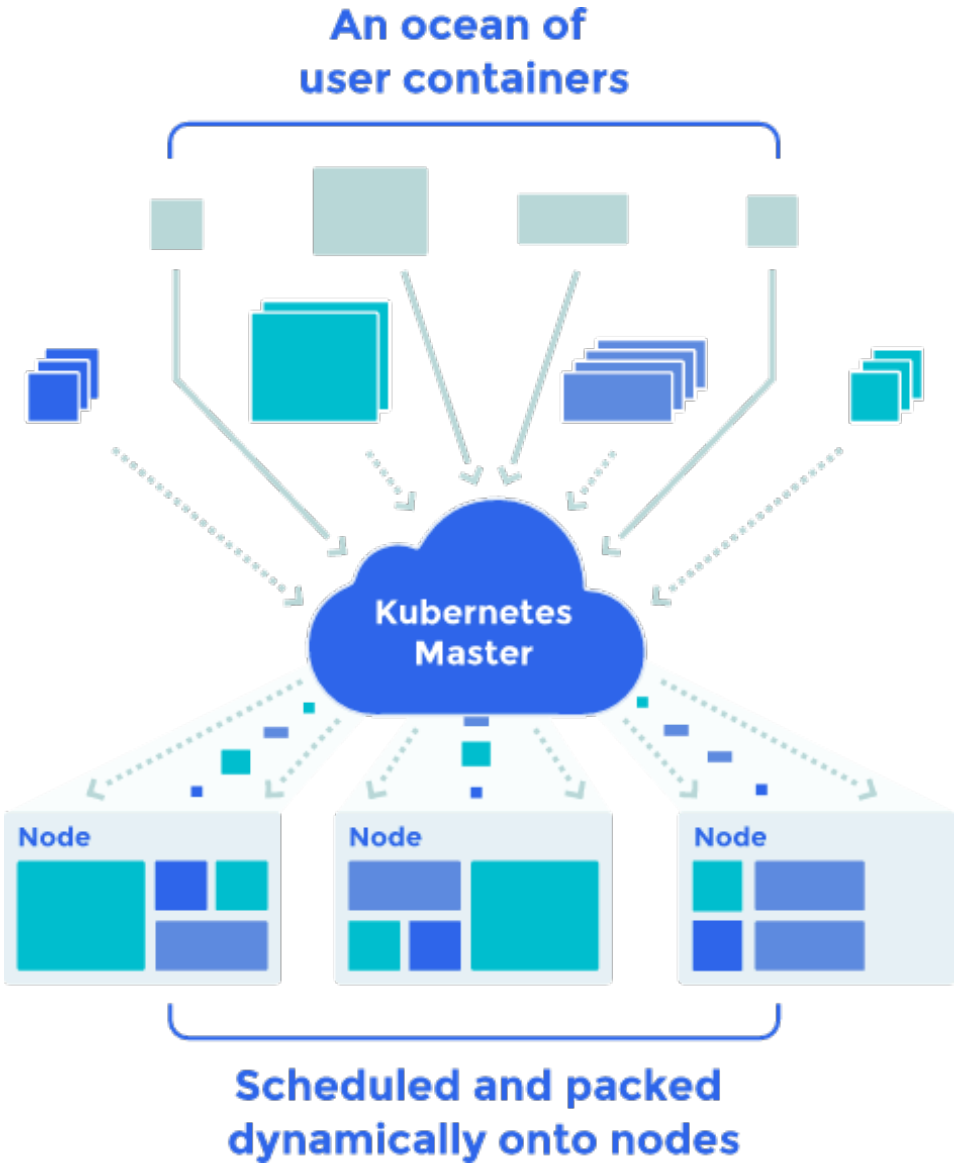
Docker Compose

- How to create a set of containers that work together



Kubernetes

- Open Source cluster management of containers
- From Google, but separate from the Borg project
- Now donated to the CNCF



Docker ecosystem

The Twelve-Factor App

Introduction

In the modern era, software is commonly delivered as a service: called *web apps*, or *software-as-a-service*. The twelve-factor app is a methodology for building software-as-a-service apps that:

- Use **declarative** formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;
- Are suitable for **deployment** on modern **cloud platforms**, obviating the need for servers and systems administration;
- **Minimize divergence** between development and production, enabling **continuous deployment** for maximum agility;
- And can **scale up** without significant changes to tooling, architecture, or development practices.

The twelve-factor methodology can be applied to apps written in any programming language, and which use any combination of backing services (database, queue, memory cache, etc).

http://12factor.net

12 Factor Apps

- Codebase
  - One codebase in revision control, many deploys
- Dependencies
  - Explicity define and declare
- Config
  - Store config in the environment
- Backing Services
  - Treat as attached resources
- Build, Release, Run
  - Strictly separate
- Processes
  - Execute the app as stateless processes
- Port Binding
  - Export services via port binding
- Concurrency
  - Scale out via processes
- Disposability
  - Fast startup and graceful shutdown
- Dev/Prod Parity
  - Keep dev/staging/prod as similar as possible
- Logs
  - Treat logs as event streams
- Admin Processes
  - Run admin/mgmt tasks as one-off processes

Summary

- Docker and the Container model
  - Lightweight virtualization and repeatability
  - Blue Green deployment
  - “Warehouse Scale” computing