

# Cloud Computing and Big Data

## Containers and the evolution of cloud native

Oxford University  
Software Engineering  
Programme  
July 2018



© Paul Fremantle 2015. This work is licensed under a Creative Commons  
Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

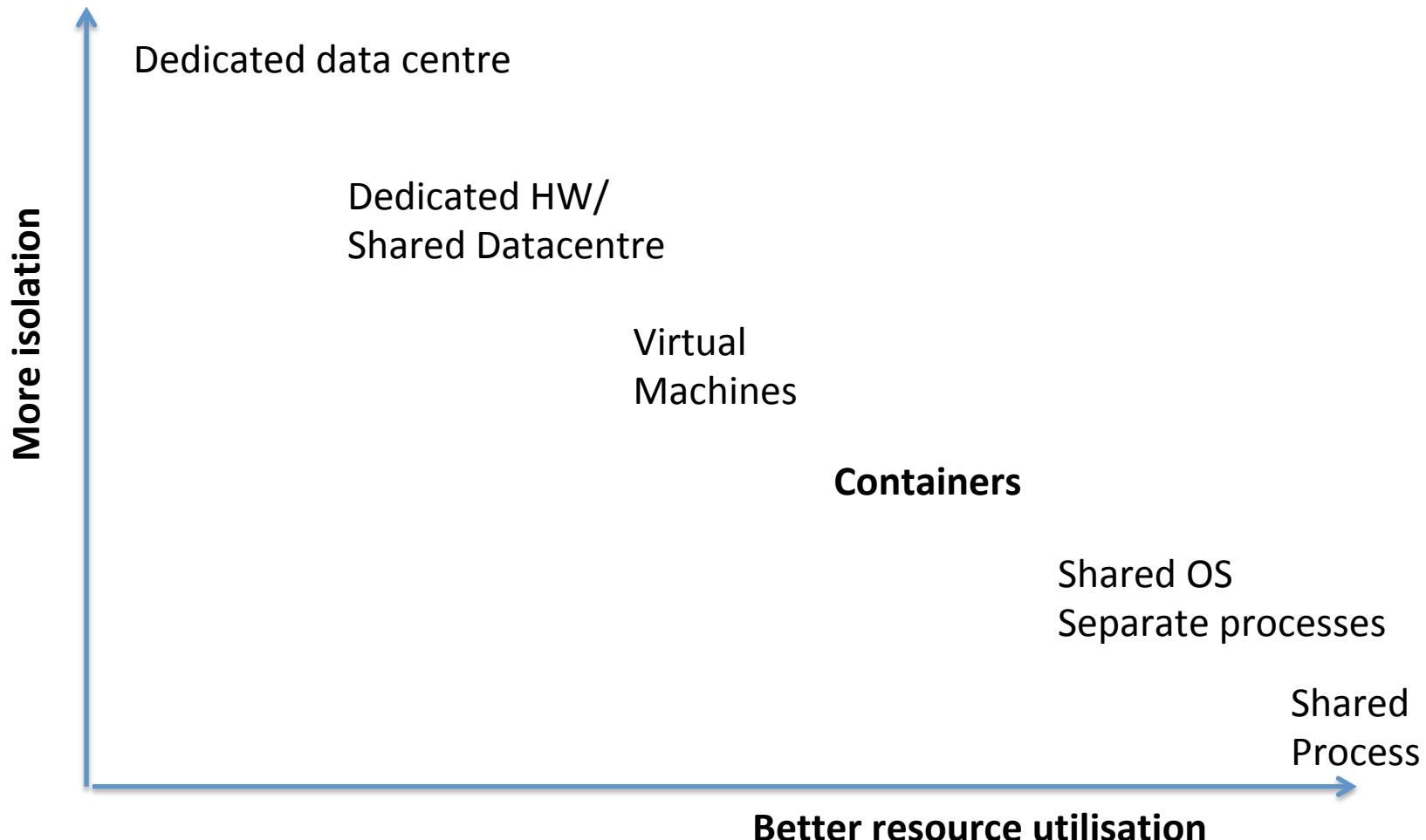
# Contents

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



© Paul Fremantle 2015. This work is licensed under a Creative Commons  
Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

# 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



# 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.



# 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 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)

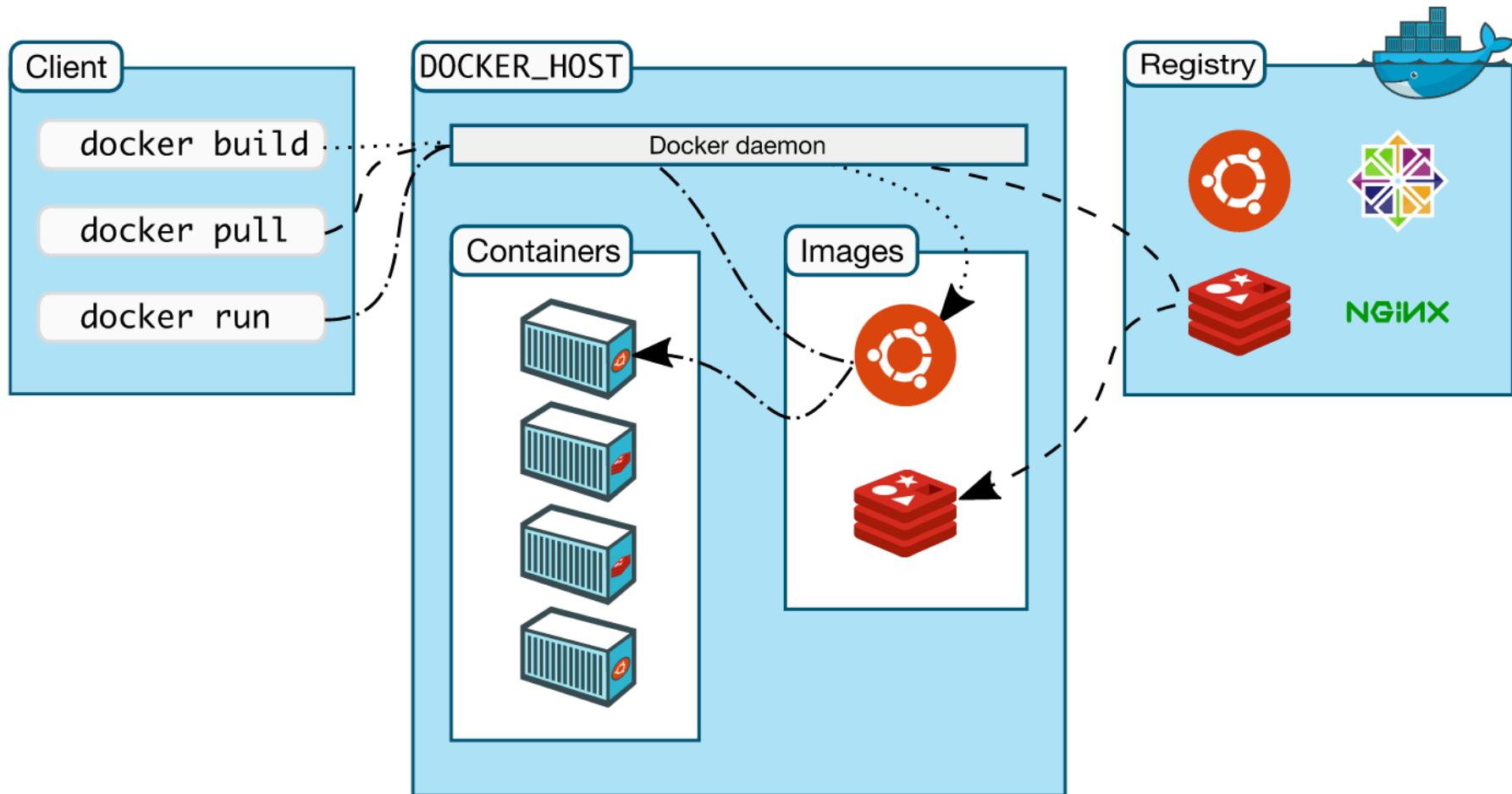


# Why Docker?

- 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?



# Dockerfile

```
FROM alpine
RUN apk --update add python py-pip && \
    pip install --upgrade pip && \
    mkdir -p /home/root/python && \
    pip install kafka && \
    pip install httpplib2

COPY tflrepub.py /home/root/python/

WORKDIR /home/root/python/

ENTRYPOINT python tflrepub.py
```



# 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`



# Docker Compose

- A way of configuring multiple Docker containers
  - Solves security issues
    - Shouldn't put secrets in Dockerfile or Docker image
  - Manages dependencies between containers



# docker-compose.yml

```
version: '2'

services:
  zookeeper:
    build:
      context: .
      dockerfile: Dockerfile-zookeeper
    ports:
      - "2181:2181"
  kafka:
    build:
      context: .
      dockerfile: Dockerfile-kafka
    ports:
      - "9092:9092"
  networks:
    default:
      aliases:
        - kafka.freo.me
  depends_on:
    - zookeeper
```



# Docker Machine

- Manages docker servers
  - e.g. VirtualBox, Amazon, DigitalOcean
  - Let's you start/stop and configure Docker to talk to the remote server



# Quick demo



© Paul Fremantle 2015. This work is licensed under a Creative Commons  
Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

# Cloud Orchestration

- What does an Operating System do?
  - Manages processes
  - Co-ordinates the processes access to resources
    - CPUs
    - Memory
    - Disk
    - Devices
  - Fairness and priority between processes



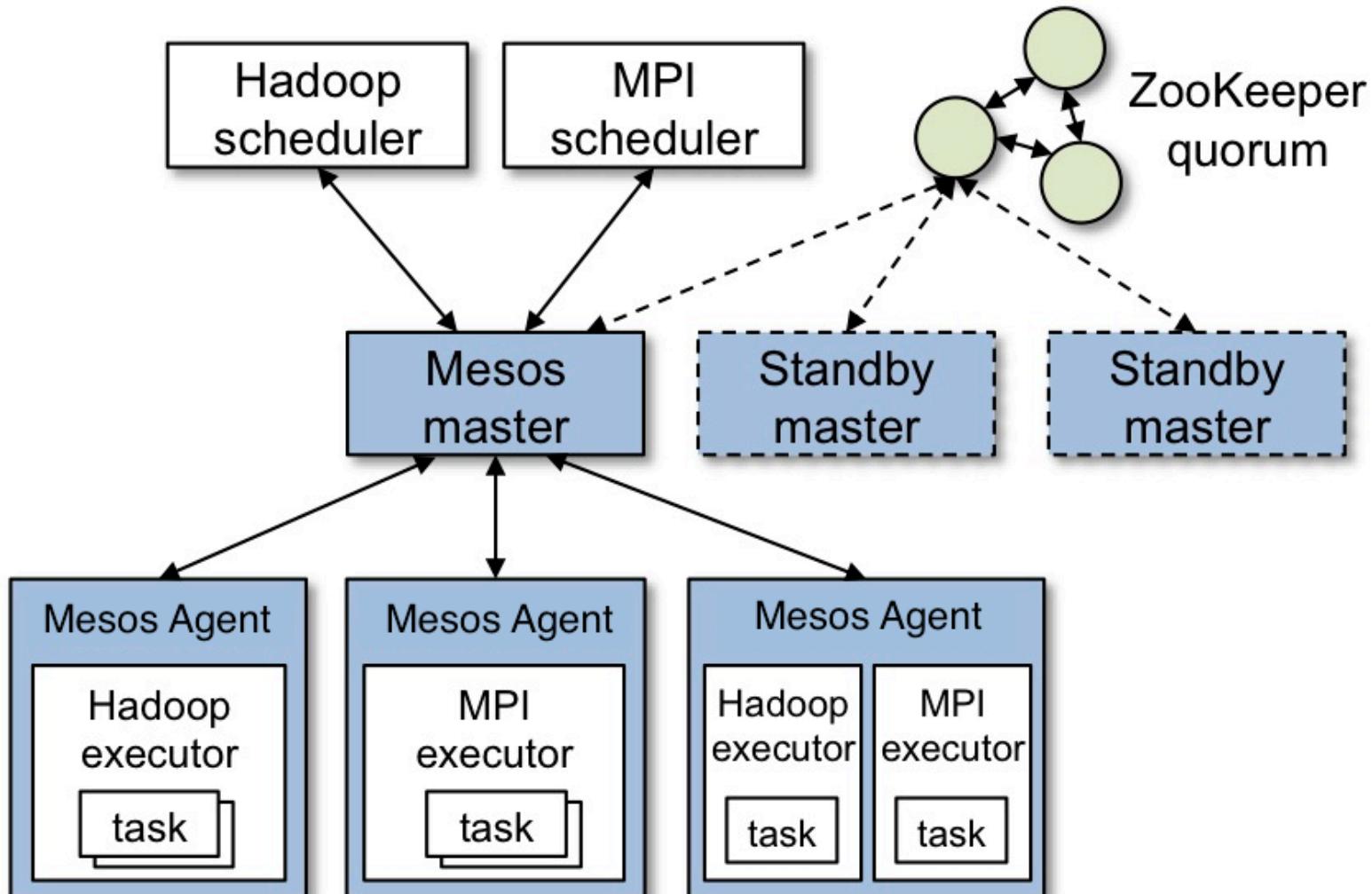
# Datacenter Operating System

aka Container Orchestration

- Manages the placement of containers
  - Access to resources
  - Configuration and networking
  - Moves containers
  - Load balances across containers
- Effectively creating a single OS across a cloud
  - Containers vs Processes



# Apache Mesos



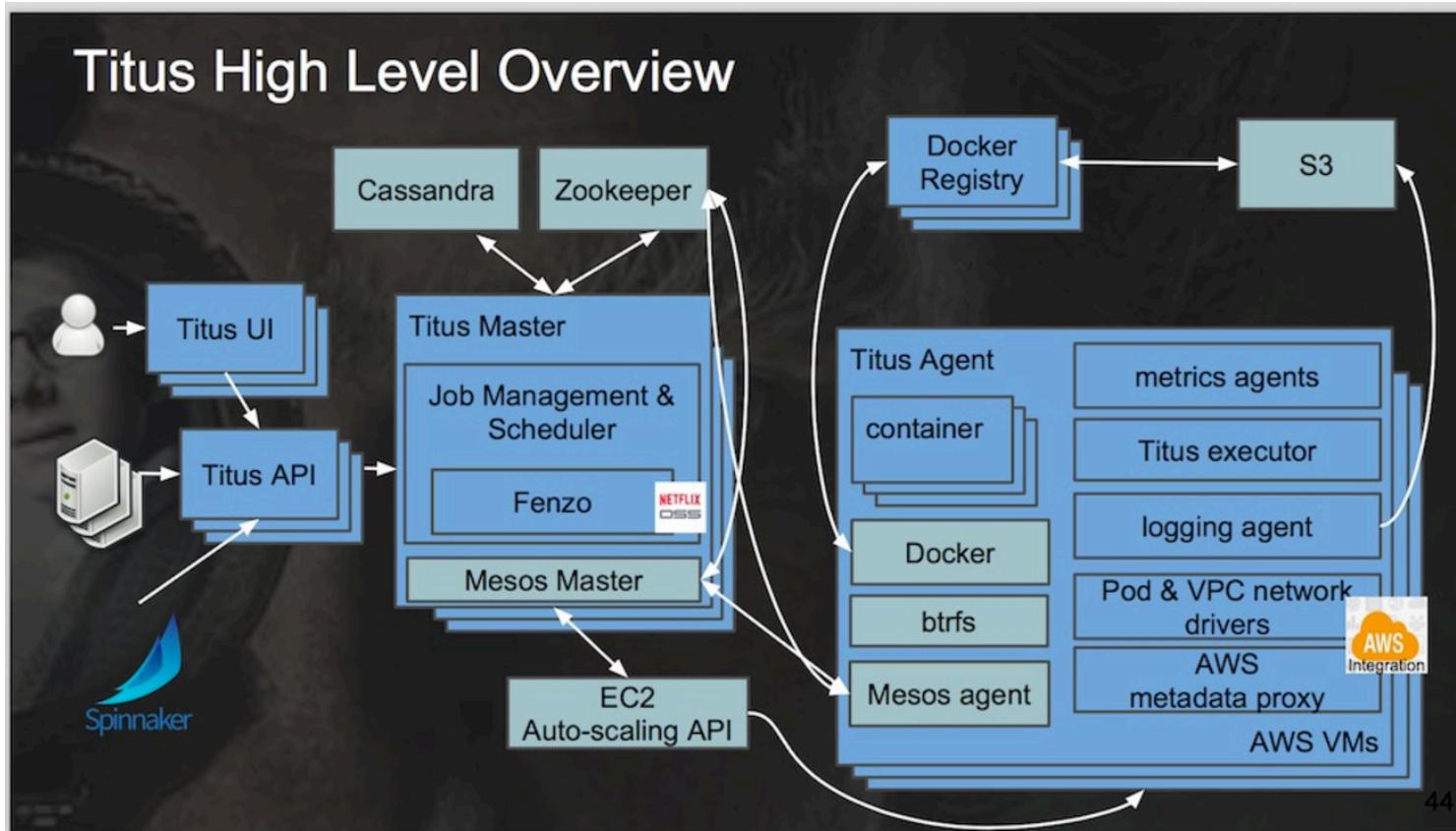
# Netflix Titus

Running on 5000 AWS instances (m4.4xlarge and r3.8xlarge)

Three regions

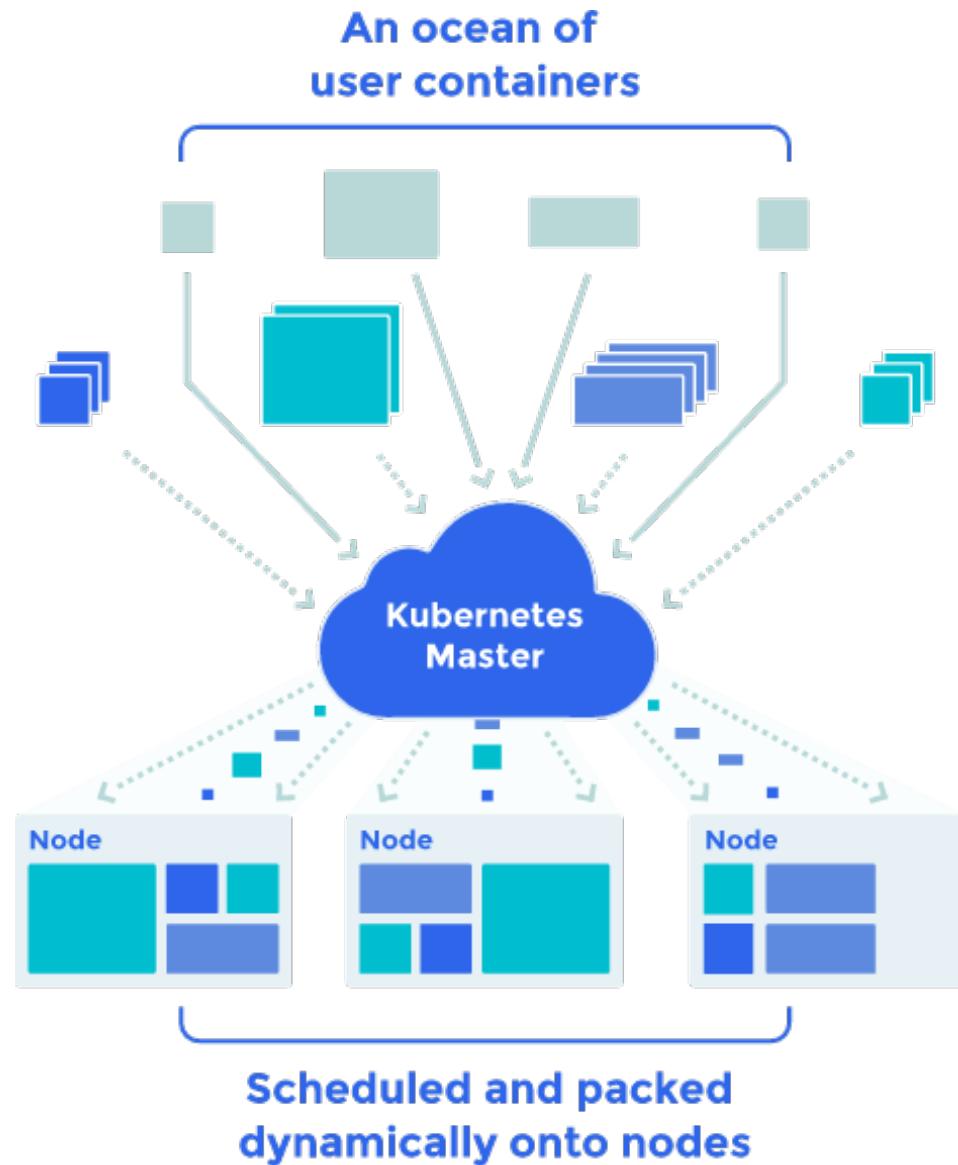
10,000 containers running at any time

1,000,000 containers launched

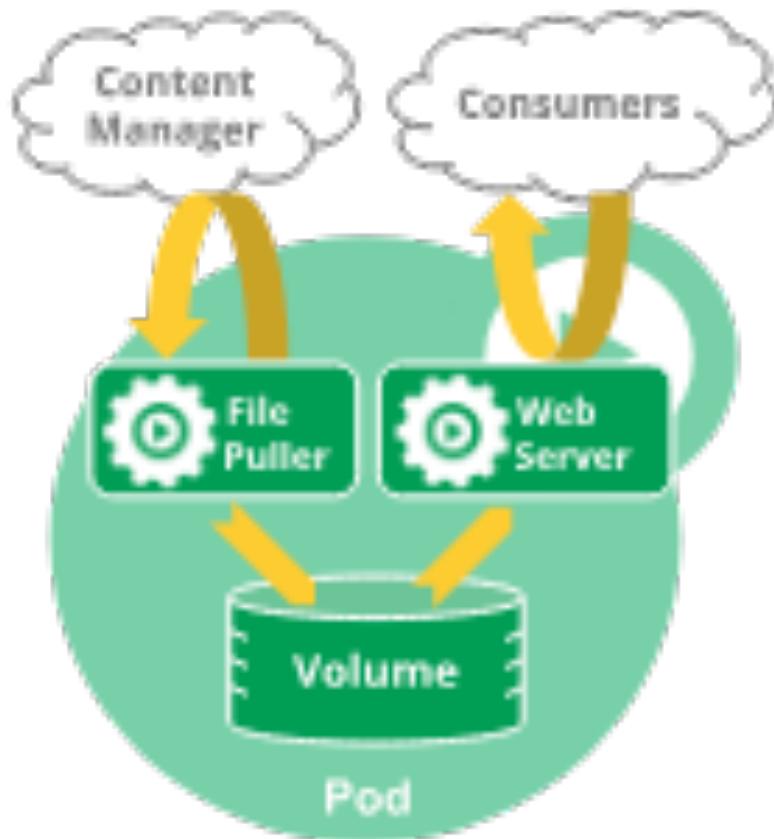


# Kubernetes

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



# Pods



A Pod encapsulates an application container (or, in some cases, multiple containers), storage resources, a unique network IP, and options that govern how the container(s) should run.

A Pod represents a unit of deployment: a single instance of an application in Kubernetes, which might consist of either a single container or a small number of containers that are tightly coupled and that share resources.



# Services

- An abstract exposure of pods
- Pods die and are recreated, replicated

“A Kubernetes Service is an abstraction which defines a logical set of Pods and a policy by which to access them”



# Volumes

- A persistent virtual disk that belongs to a Pod
- Shares data between containers
- Lives longer than a container, but no longer than the pod



# Namespaces

- A virtual cluster
- Names must be unique inside namespaces, can be the same across different namespaces



© Paul Fremantle 2015. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

# Kubernetes Operations (kops)

[build](#) passing [go report](#) A [godoc](#) [reference](#)

The easiest way to get a production grade Kubernetes cluster up and running.

## What is kops?

We like to think of it as `kubectl` for clusters.

`kops` helps you create, destroy, upgrade and maintain production-grade, highly available, Kubernetes clusters from the command line. AWS (Amazon Web Services) is currently officially supported, with GCE and VMware vSphere in alpha and other platforms planned.

## Can I see it in action?

```
AutoscalingGroup/nodes.example.nivenly.com
  MinSize      2
  MaxSize      2
  Subnets      [name:us-west-2a.example.nivenly.com]
  Tags         {k8s.io/role/node: 1, Name: nodes.example.nivenly.com, KubernetesCluster: example.nivenly.com}
  LaunchConfiguration name:nodes.example.nivenly.com

Cluster configuration has been created.

Suggestions:
* list clusters with: kops get cluster
* edit this cluster with: kops edit cluster example.nivenly.com
* edit your node instance group: kops edit ig --name=example.nivenly.com nodes
* edit your master instance group: kops edit ig --name=example.nivenly.com master-us-west-2a

Finally configure your cluster with: kops update cluster example.nivenly.com --yes
bash-3.2$ kops edit cluster $NAME
```



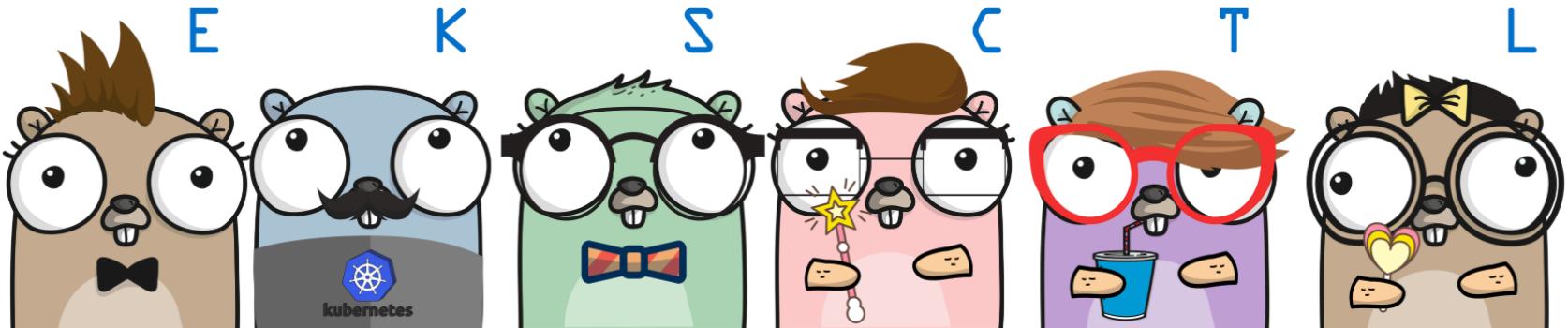
© Paul Fremantle 2015. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

# eksctl - a CLI for Amazon EKS

circleci passing

`eksctl` is a simple CLI tool for creating clusters on EKS - Amazon's new managed Kubernetes service for EC2. It is written in Go, and based on Amazon's official CloudFormation templates.

You can create a cluster in minutes with just one command – `eksctl create cluster` !



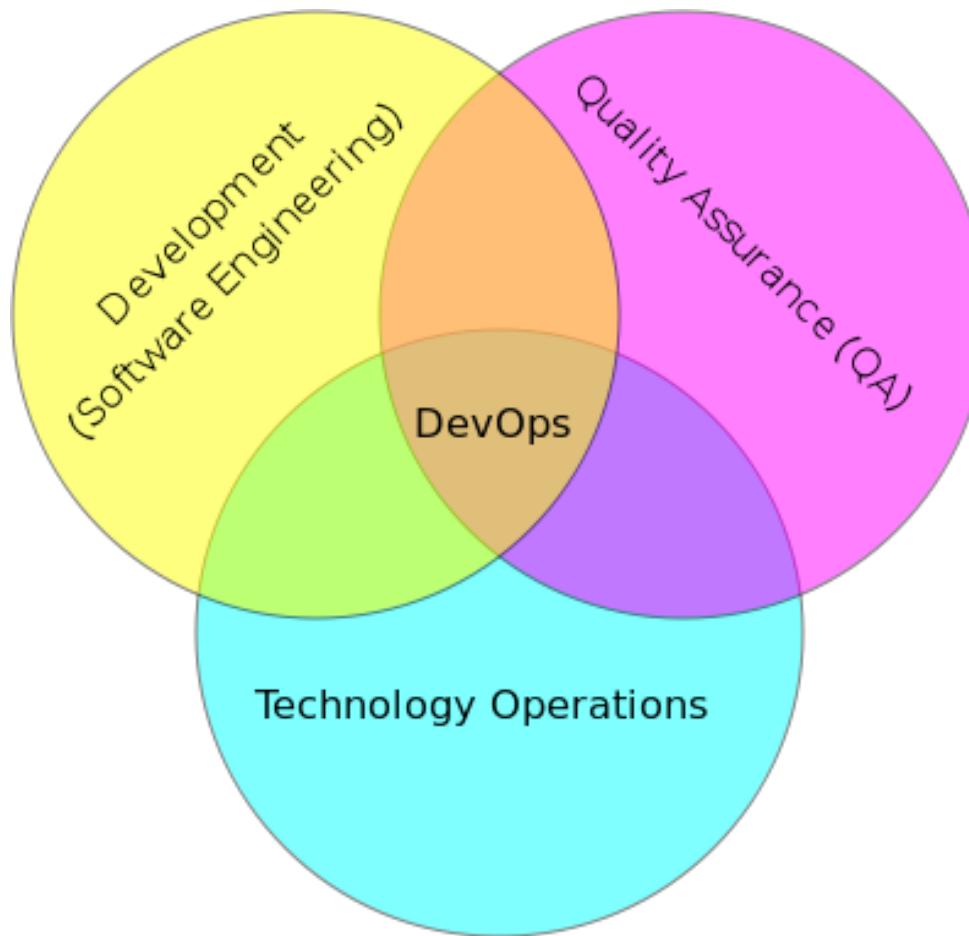
## Usage

To download the latest release, run:

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/download/latest_release/eksctl_$(uname -s)_arm64"
sudo mv /tmp/eksctl /usr/local/bin
```

Alternatively, macOS users can use [Homebrew](#):

# DevOps



# DevOps

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



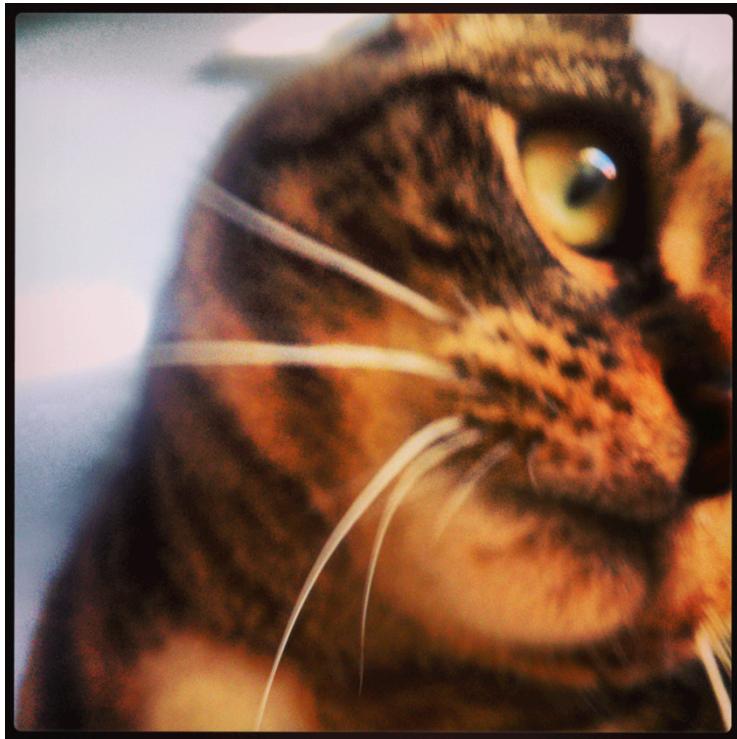
# Cloud and DevOps

- 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



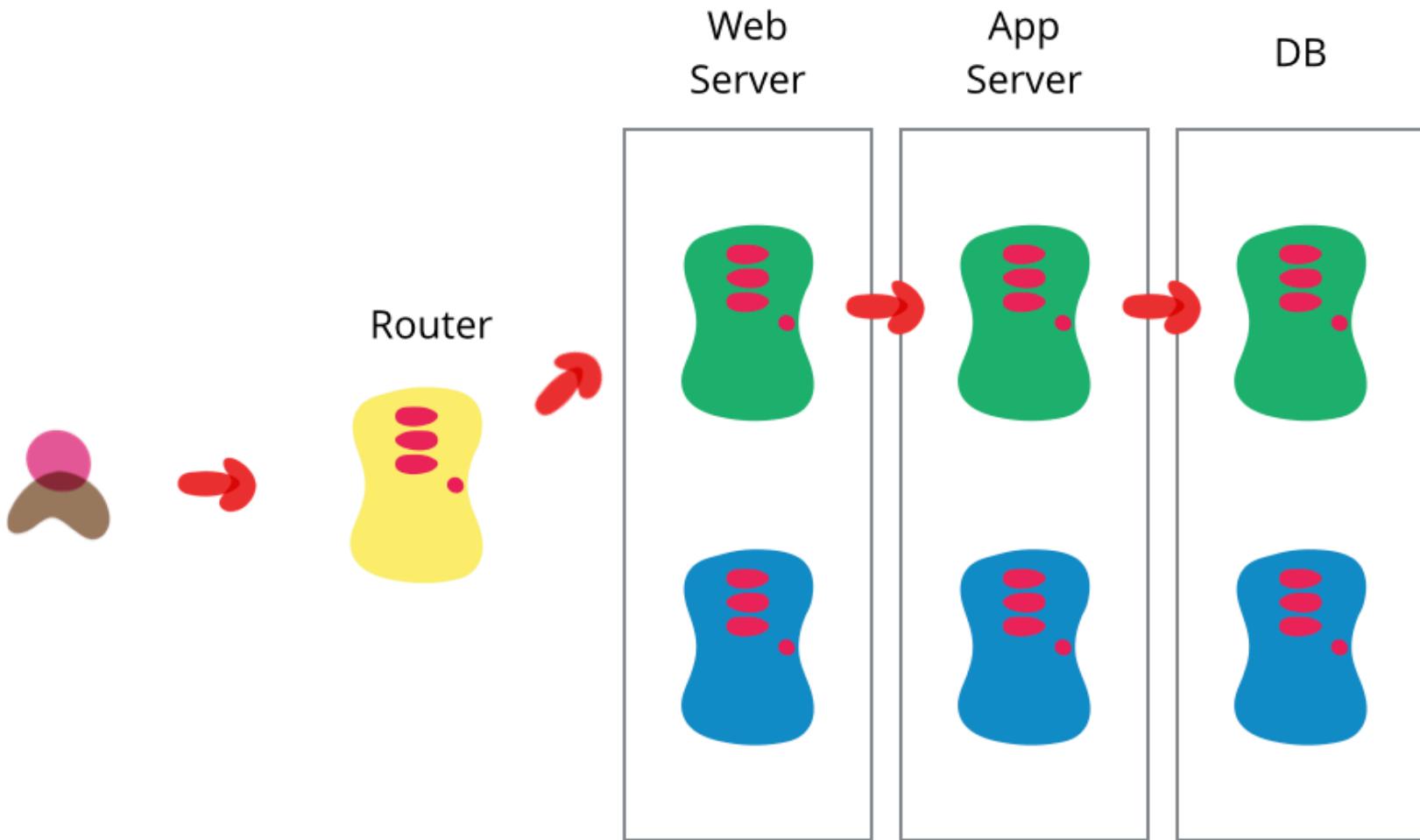
# Kittens vs Cattle

## (An unpleasant but effective analogy)



© Paul Fremantle 2015. This work is licensed under a Creative Commons  
Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

# Blue Green Deployment



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



Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

# 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



What can be described and observed, can be automated and operated



<https://www.weave.works/blog/gitops-git-push-all-the-things>



© Paul Fremantle 2015. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

# GitOps

- Infrastructure as Code
- Terraform + Deployment + Containers + Build
- Everything is in Git
  - Any change to the infrastructure is a Pull Request



# Summary

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



© Paul Fremantle 2015. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>