

Container Orchestration: Which Conductor?



Hewlett Packard
Enterprise

Mike Bright, @mjbright



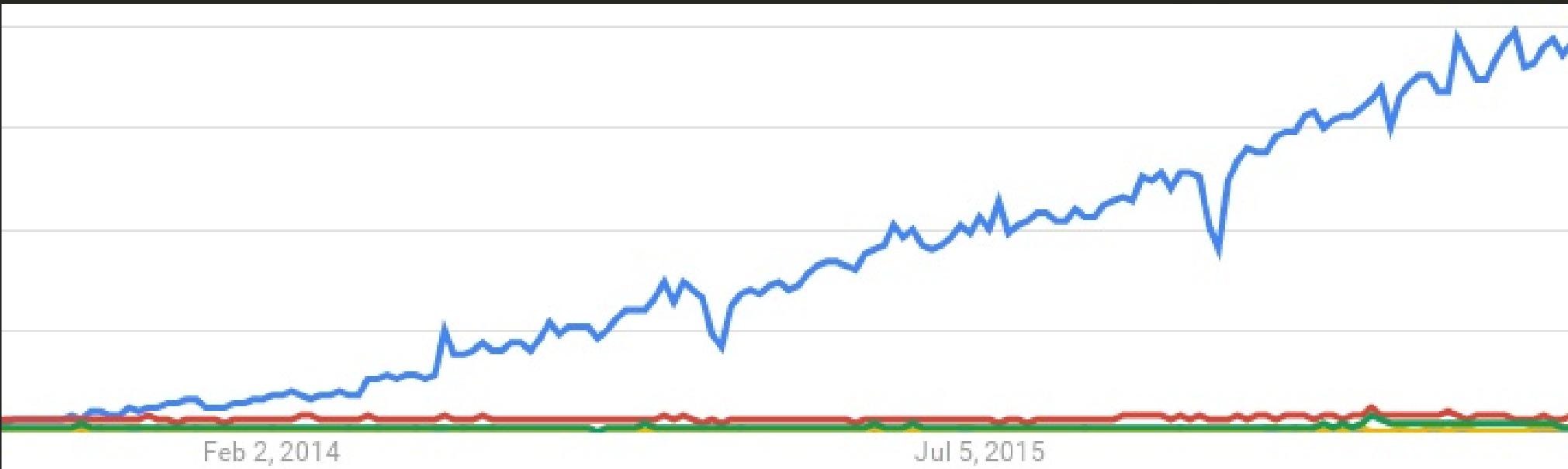
Haikel Guemar, @hguemar



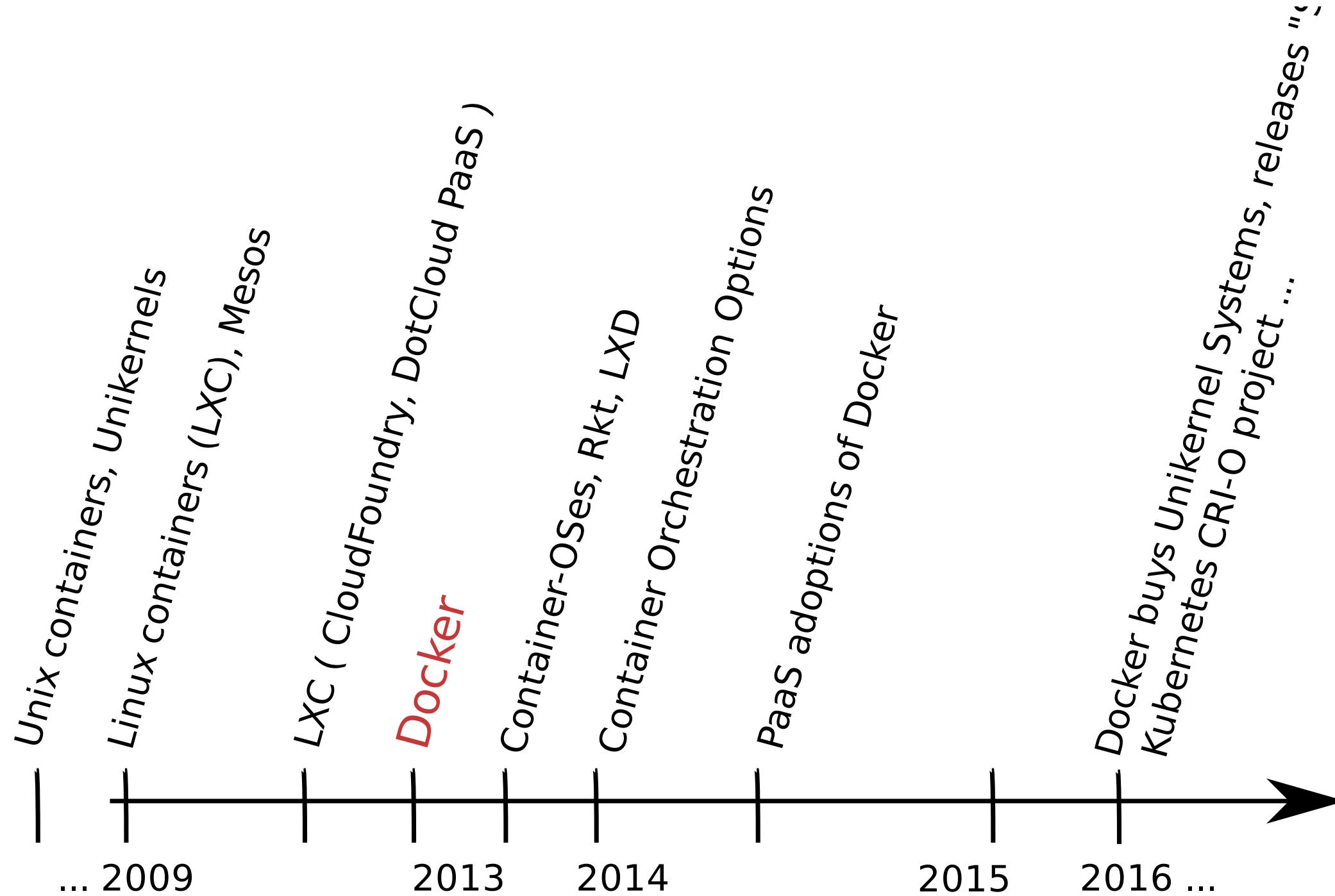
Mario Loriedo, @mariolet

First ...

A little bit of history



So let's first look at recent container history ...



@hguemar @mjbright @mariolet

History Container-OSes

Many vendors are developing Container-OSes, small OS (mainly Linux-based) to be the basis for container engine hosts whether they be bare-metal or virtual **host machines**.

They're small, with fast startup, use few resources and have a small attack surface and often "*atomic*" software updates.

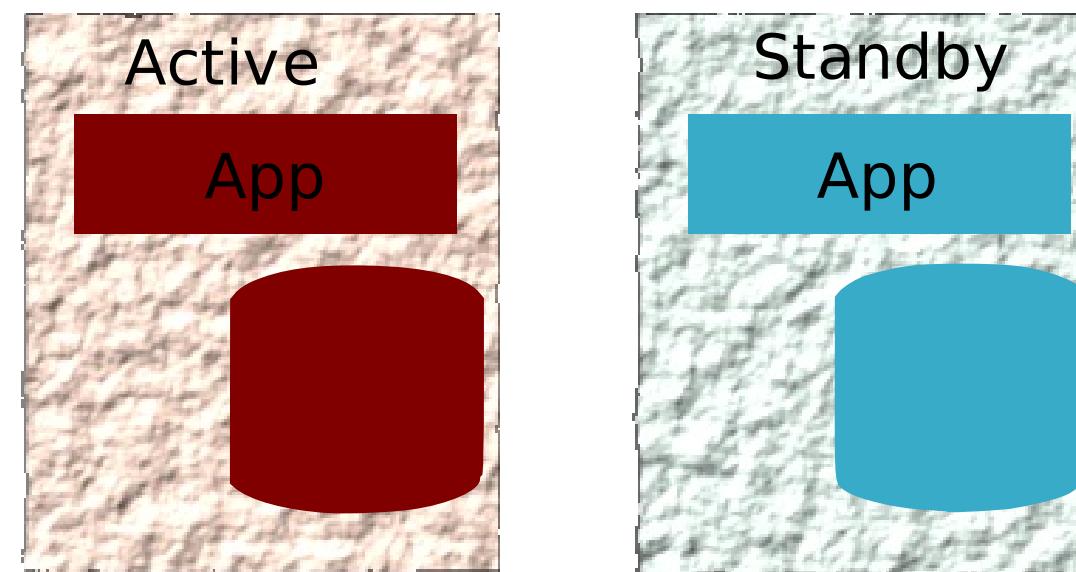
| OS | Vendor |
|--------------------|------------------|
| CoreOS Linux | - (CoreOS) |
| Project Atomic | - (RedHat) |
| RancherOS | - (Rancher Labs) |
| Photon | - (VMWare) |
| Nano Server OS | - (Microsoft) |
| Ubuntu Snappy Core | - (Canonical) |

...Unikernels

μ -Services

From monoliths to μ -services

Remember when **high availability** meant this ...?



Servers running **monolithic applications** in **Active-Standby** modes, as 1+1, N+1, or N+M or split across 3 tiers.

Scaling meant to "**scale up**" by adding CPU, RAM, disk.
But there's a limit to this ... then you have to "**scale out**"

@hguemar @mjbright @mariolet

μ - services

From monoliths to μ - services

Then came μ -services ..

As the industry moved to virtualized micro-services this allowed to obtain greater efficiencies (higher utilisation of resources) and the redesign of applications allows to scale out and achieve high availability.

Containers facilitate this move, allowing faster scaling and even greater efficiencies with less redundancy (no OS to reproduce).

How containers help?

Container solutions such as Docker go beyond the isolation capabilities of LXC by providing simple to use tools to enable packaging of apps with their dependencies allowing portable applications between systems.

Containers are lightweight

Versioned images containing all dependancies can be shared

Containers allow to use the same application binaries on development, test and production systems whether that be on a laptop, server or in the cloud.

It's a no brainer for developers, who can build and share their own images

μ -
services

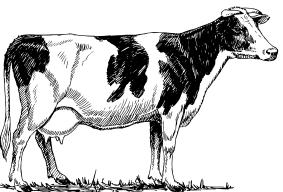
From monoliths to μ -services

But 1000's of instances are **unmanageable** ...

We can't take care of our



so we have to treat them like



that's cloud native !

@hguemar @mjbright @mariolet

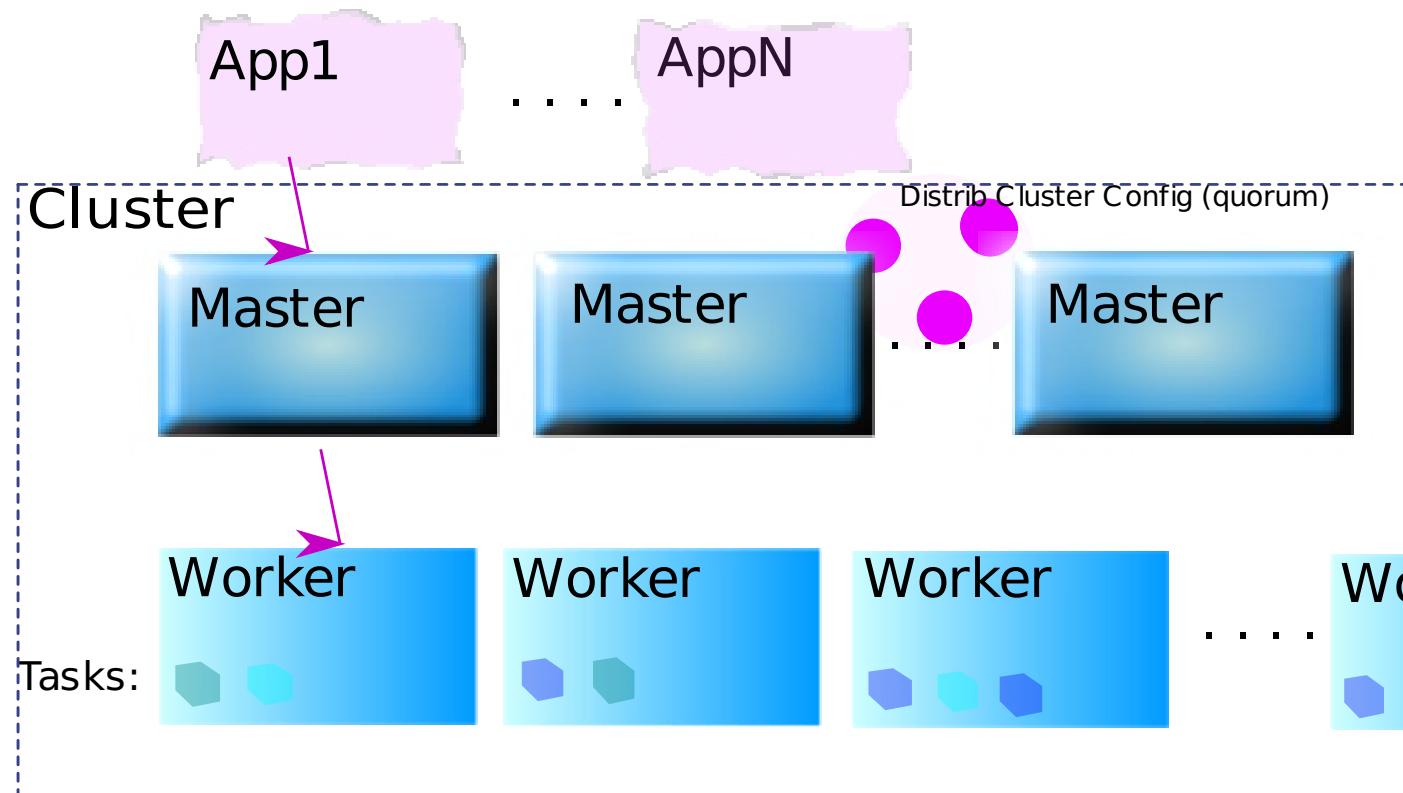
So we need container orchestration



Orchestration

What was Container Orchestration again?

- Architecture - Composition & Stitching
- Workflows & Policies to
 - Scale in/out (maybe automatically)
 - Place workloads for
 - load balancing, fault tolerance, resource optimization
 - Adapt to faults



Orchestration. Getting to "*Desired State*"

To manage 100's, 1000's, 10,000's of instances we need to express "*desired state*" rather than "*do this*".

| | Imperative | Declarative |
|---------------------|--|-------------------------------------|
| Tell system | Do this <i>"start a new instance"</i> | desired state <i>"3 x mysql"</i> |
| Intelligence | Operator | Orchestration Engine |
| Flexibility | Best | Least |

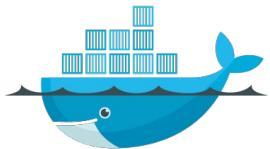
It is no longer feasible for an operator to

- know the resources available (e.g. SSD/HDD, GPU, ...)
- react to failure, know when to scale ...

@hguemar @mjbright @mariolet

Choice is great - when you know
what you want ...

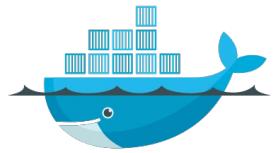
Orchestration The Big 3 - Main Orchestration Choices



- Docker Swarm ("Swarm Mode")
- Apache Mesos
- Kubernetes



Orchestration The Big 3 - Main Orchestration Choices



- Docker Swarm ("Swarm Mode")



- Apache Mesos
- Kubernetes



... more Choices ...

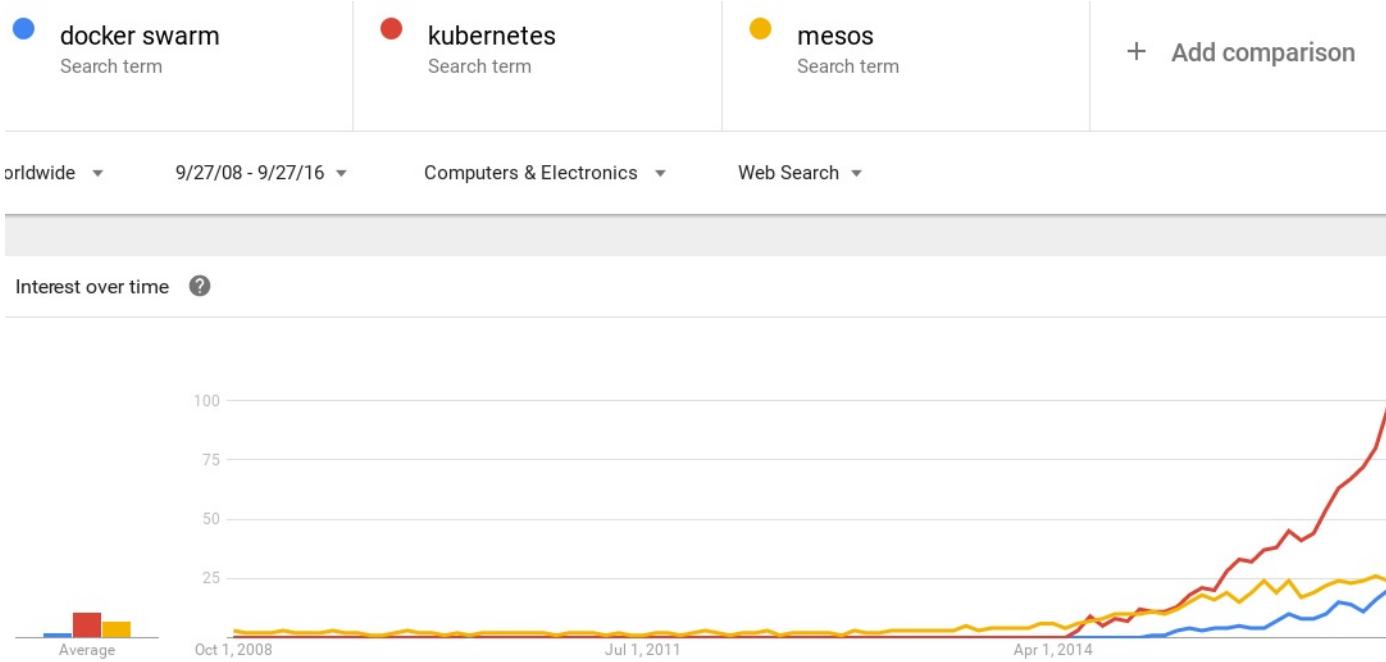
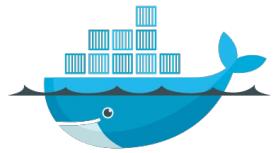


- Rancher (Rancher Labs)
- Fleet (CoreOS)
- Nomad (HashiCorp)
- Kontena



@hguerrero OpenStack Right @unriiolet
OpenStack Magnum

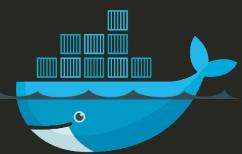
Orchestration The Big 3 - What does Google Trends say?

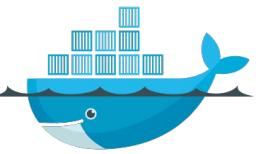


Clearly Kubernetes has a lead in *Google "search trends"*

But we can expect "*Docker Swarm*" to make quick progress thanks to the new "*swarm mode*"

Docker Swarm





Docker Swarm

Dec
2014

... **Docker Swarm** is announced

Orchestration using Docker Compose

Jun 2016 ... **Swarm Toolkit** released

OpenSource Orchestration Toolkit

Jun 2016 ... **Swarm Mode** announced

Orchestration integrated into Docker Engine

Docker 1.12 is the first release to integrate "**Swarm Mode**" The original *Docker Swarm* is maintained for legacy use.

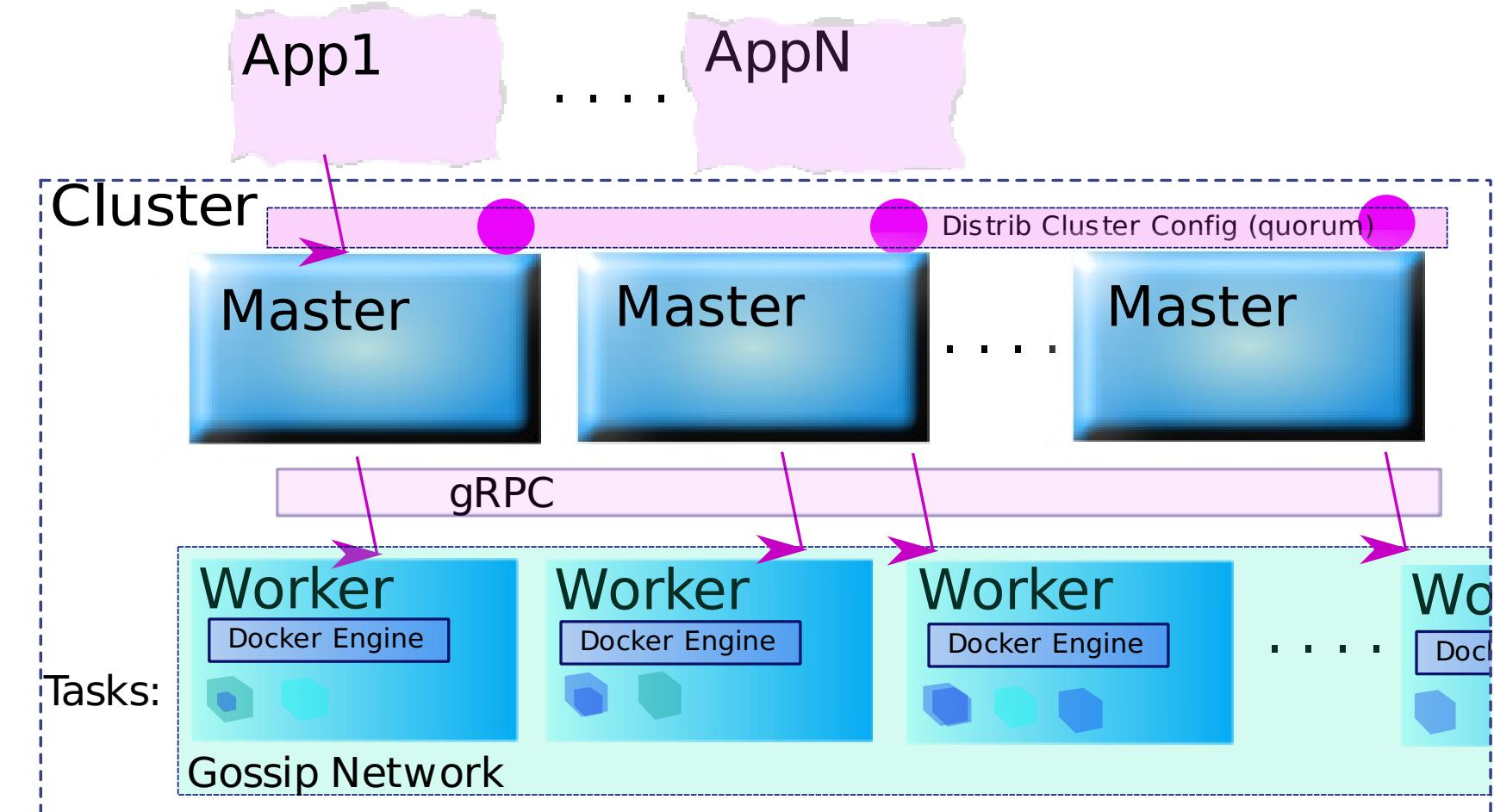
"Swarm Mode" is a revolution bringing:

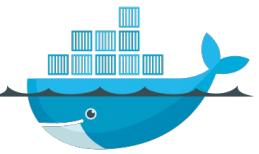
- Orchestration directly in the Docker Engine
- Advanced networking features
 - mesh network, vxlan
- Load balancing
- Service Discovery
- *Docker* traditional ease of use



Docker Swarm

Architecture





Docker Swarm

Using Docker "Swarm Mode"

Create a new swarm by creating the master node:

```
$ docker swarm init --advertise-addr 192.168.2.100
Swarm initialized: current node (dxn1zf6l61qsb1josjja83)
```

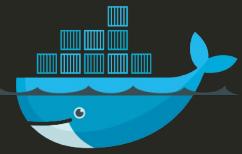
Join a new Worker node to the swarm:

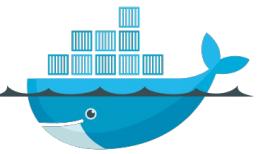
```
$ docker swarm join --token TOKEN 192.168.2.100:2377
```

Join a new Master node to the swarm:

```
$ docker swarm join-token manager
```

Docker Swarm Demo





Docker Swarm

Docker Swarm Demo

- Creation of a 3 node cluster
- Run a service on the cluster and scale it to 3 replicas
- Make a rolling update of the service
- Drain a node of the cluster

Kubernetes



From the Greek: "Steersman, helmsman, sailing master"



Google created based on extensive experience running containers internally ~ billions of containers a year

Started Oct 2014, reached v1.0 in July 2015, now at v1.4

Kubernetes
Managed by the Cloud Native Computing Foundation
<https://cncf.io/>

Commercial offerings from CoreOS (Tectonic) and Canonical

kubernetes.io



Integrated in:

- GKE (Google Container Engine)
- OpenStack above Kubernetes
 - Stackanetes (CoreOS, uses Tectonic)
 - Self healing OpenStack demo
 - Mirantis (OpenStack CI/CD based on Kolla)
- Various PaaS:
 - RedHat OpenShift CP
 - HPE Stackato v.40
 - Deis
 - Microsoft Azure (Azure Container Service)

Kubernetes

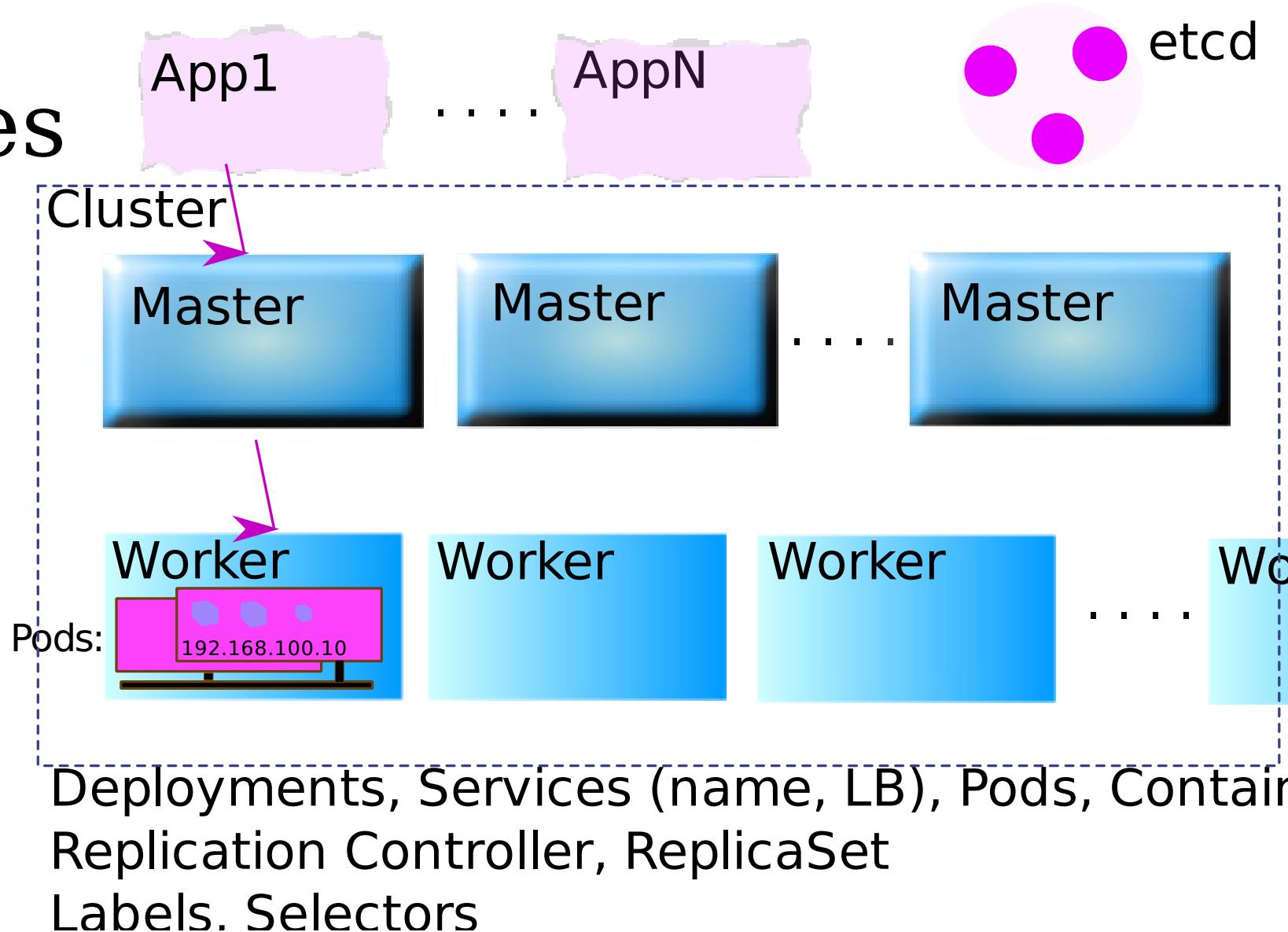
Many tools and add-ons available

- Dashboard
- Prometheus (monitoring)
- Heapster (monitoring)
- Helm (package manager)



Architecture

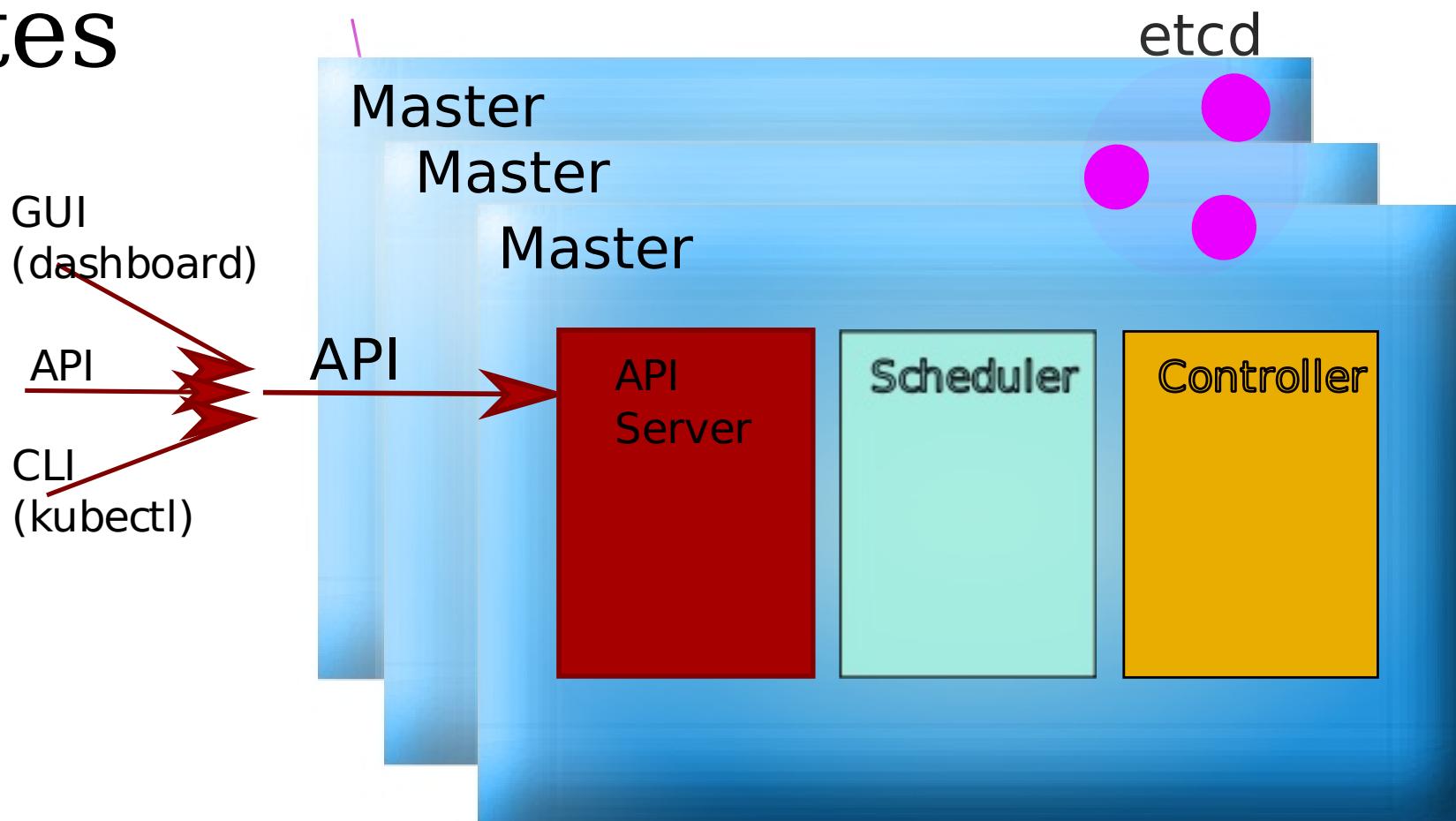
Kubernetes





Architecture - Master Nodes

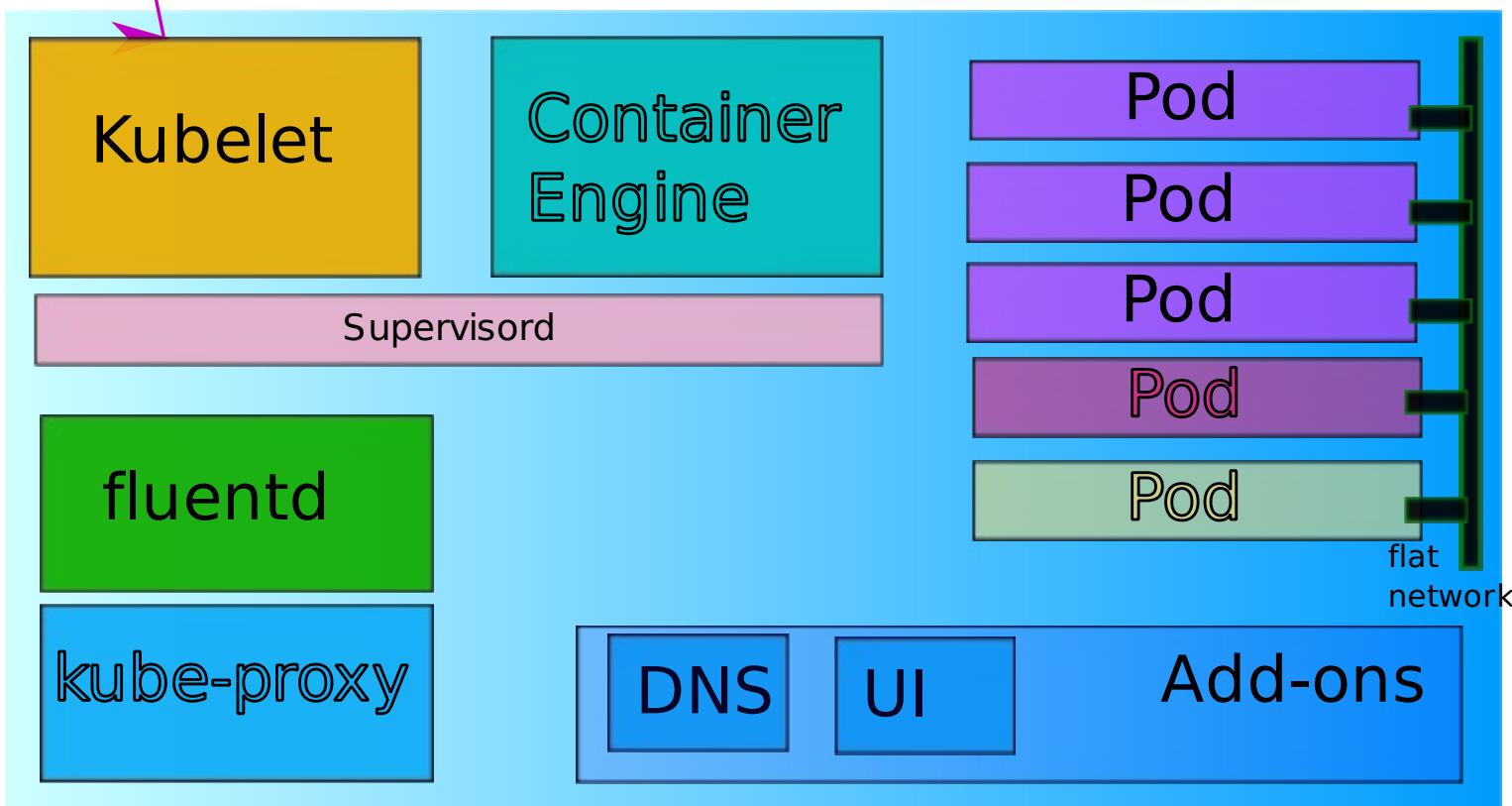
Kubernetes





Architecture - Worker Nodes

Kubernetes

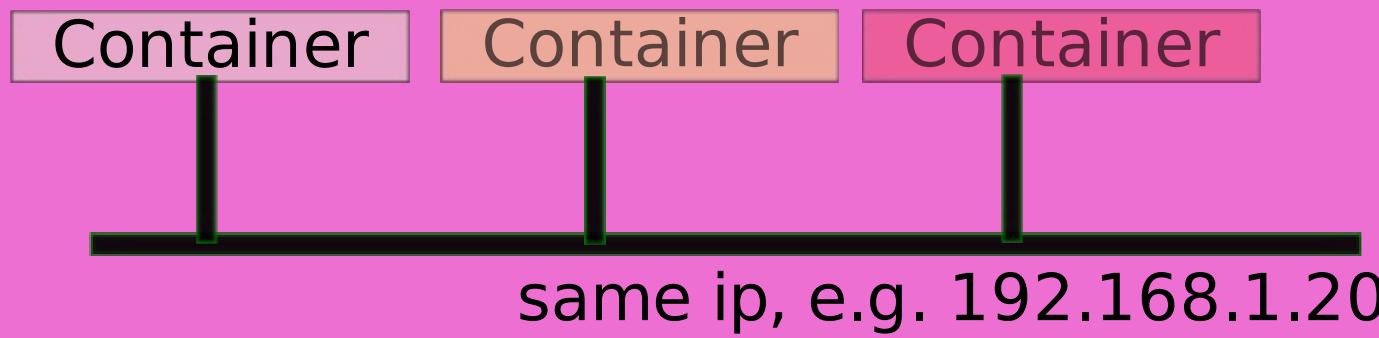




Architecture - Pods

Kubernetes

Containers share some namespaces:
- PID, IPC, network , time sharing



A pod houses one or more containers



Kubernetes Concepts

Kubernetes

- Pod
 - smallest unit: 1 or more containers on the same IP
- Replication Controller & ReplicaSet (NG)
 - Ensures that a given number of pods run
- Deployment
 - Declaratively manage ReplicaSets, rolling upgrade
- Service
 - addressable IP, load-balancing
- Labels
 - Key/value pairs which can be attached to objects
- Selectors
 - Used to select a group of objects by label
- Secrets
 - Small amounts of sensitive info (certs., keys, ..)
- ConfigMaps
 - Dynamically loadable config (k-v pairs, or json)
- DaemonSets
 - Ensure a pod runs on all (or subset of) nodes
- Jobs
 - Short-lived pods, e.g. for batch jobs

Kubernetes Demo



- **Minikube [single node cluster]**
 - **Kubernetes Tutorial**
 - **GCP: Kubernetes Tutorial**

Apache Mesos





Apache Mesos

The most proven orchestrator today, exists since 2009.
Can scale to ~ 10,000 nodes.

Mesos is used in production by:

- Uber
- Twitter
- Paypal
- Hubspot
- Airbnb
- eBay
- Groupon
- Netflix

Mesos has Containerizers (Mesos, Docker) to isolate tasks.

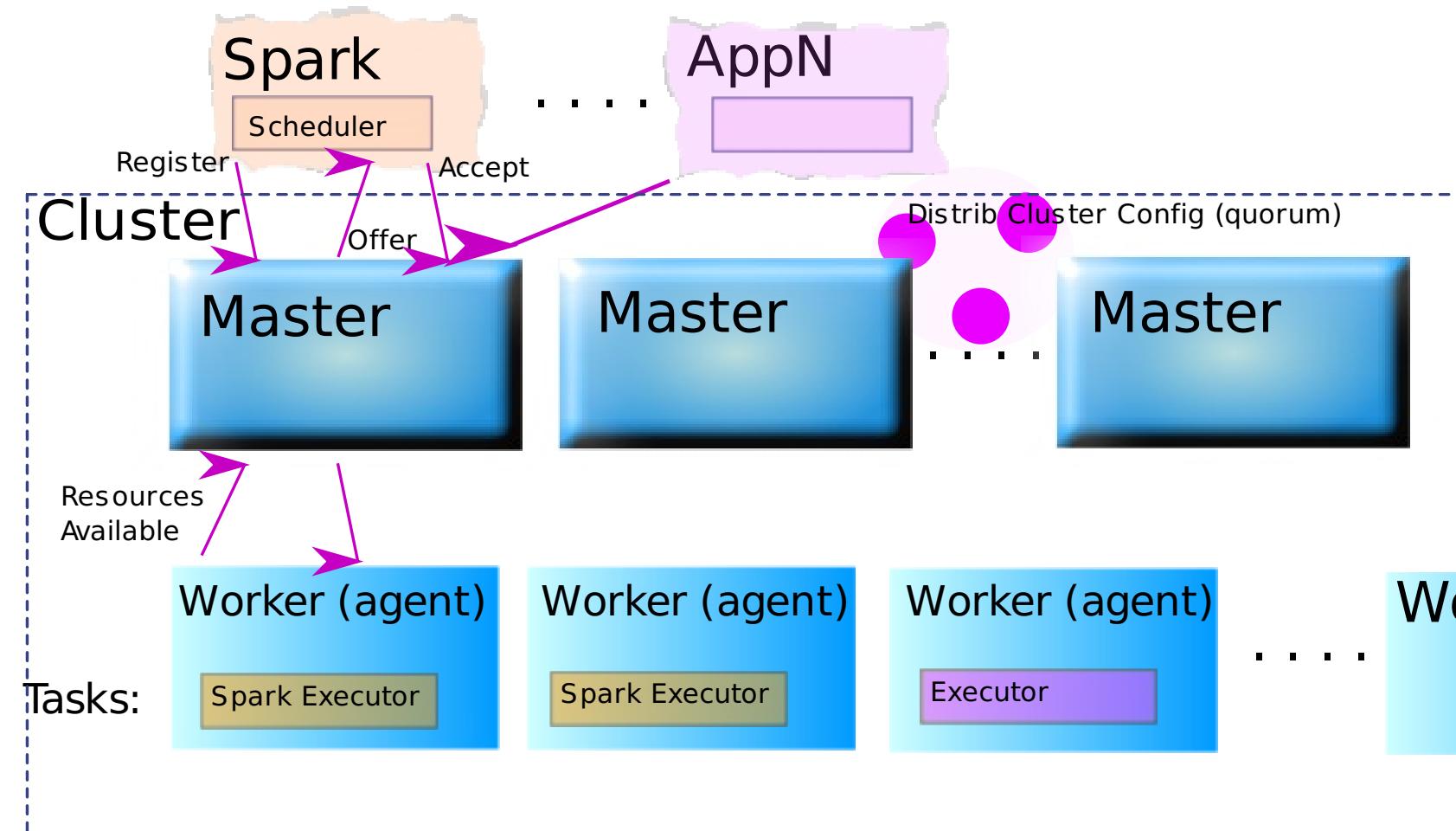
Unified Containerizer supports Docker and appc image formats.

Mesosphere sells the **DC/OS** product based on **Mesos**. DC/OS is also an open source project.

mesos.apache.org

Apache Mesos

Architecture



A Mesos application provides its own scheduler and a task executor.
mesos.apache.org



Apache Mesos

Mesos is used in conjunction with Frameworks such as

- For *long running* tasks:
 - **Marathon** (Mesosphere), Aurora or Singularity
- For job orchestration:
 - Chronos "cron", Jenkins
- For Big Data Processing:
 - Hadoop, Spark, Storm
 - Cassandra, ElasticSearch, ...

Apache Mesos Demo



So isn't it time we told you what to
choose?

... let's just compare them ...

What's common

Docker Swarm, Kubernetes, Apache Mesos are all creating rich Orchestration stacks with integrated container runtimes.

They're moving incredibly quickly ...

They are adding features such as networking capabilities, load balancing, services, labels.

They have a more 'declarative' approach allowing to abstract the data center as a single compute resource.

They support or are looking to support different runtime engines (*)

What advantages does Docker "Swarm Mode"

Simple to use (despite underlying complexity)

All-in-one container engine plus orchestration

Uses Docker API and familiar docker commands

Advanced networking

- mesh networking
- Load Balancing and Service Discovery

Replication

DDC (*Docker Data Center*) commercial product,
supported by HPE

@hguemar @mjbright @mariolet

What advantages Kubernetes

Rich conceptual model, based on Google experience

Pods as groupings of containers

Labels and Selectors (for all components)

Networking - Load Balancing and Service Discovery

Replication

Large ecosystem with several commercial offerings:

- Core OS Tectonic, Canonical
- PaaS (OpenShift Container Platform, Stackato)

and Public Cloud Support

- Google GKE, Azure ACS

What advantages Apache Mesos

Most mature

Battle tested by many service providers

Scales to 10,000 nodes

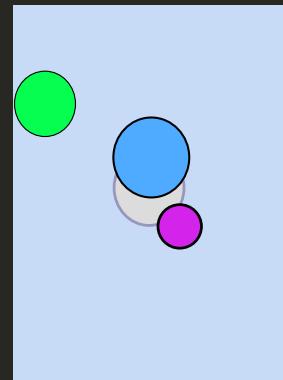
DataCenter OS - appears as 1 resource

Not just containers

Many frameworks available

(Steeper learning curve)

Thank you - Questions?



Latest slides available at:

<https://containerorchestration.github.io/ContainerOrchestration/>

@hguemar @mjbright @mariolet

Resources

@hguemar @mjbright @mariolet

Resources Books

| Publisher | Title | Author |
|-----------|-----------------------------|-----------------------------|
| O'Reilly | Docker Cookbook | Sébastien Goasguen |
| O'Reilly | Docker Up & Running | Karl Matthias, Sean P. Kane |
| O'Reilly | Using Docker [Early Access] | Adrian Mouat |
| O'Reilly | Kubernetes Up & Running | Kelsey Hightower |
| Manning | [MEAP] CoreOS in Action | Matt Bailey |
| Manning | [MEAP] Kubernetes in Action | Marko Lukša |

@hguemar @mjbright @mariolet

Resources Articles/Organisms

Cloud Native Computing Foundation - Kubernetes,
Prometheus <https://cncf.io/>

"Kubernetes the Hard Way, Kelsey Hightower" -
<https://github.com/kelseyhightower/kubernetes-the-hard-way>

"Kubernetes User Guide, Walkthrough" -
<http://kubernetes.io/docs/user-guide/walkthrough/>

@hguemar @mjbright @mariolet

Resources Videos

- June 2016 - Container Orchestration Wars, Karl Isenberg, Mesosphere
- Mar 2016 - Container Orchestration with Kubernetes, Docker Swarm & Mesos-Marathon - Adrian Mouat, Container Solutions
- Jan 2016 - Docker, Kubernetes, and Mesos: Compared.,,Adrian Otto, Southern California Linux Expo

Repos

@hguemar @mjbright @mariolet



Documentation

Kubernetes

- **Getting started guides**
 - [Creating a Kubernetes Cluster](#)
 - port Kubernetes to a new environment
 - in [Getting Started from Scratch](#)
- **User documentation**
 - to run programs on an existing Kubernetes cluster
 - [Kubernetes User Guide: Managing Applications](#)
 - the [Kubectl Command Line Interface](#) is a detailed reference on the `kubectl` CLI
 - [User FAQ](#)



Documentation - 2

Kubernetes

- **Cluster administrator documentation**
 - for people who want to create a Kubernetes cluster and administer it
 - in the [Kubernetes Cluster Admin Guide](#)
- **Developer and API documentation**
 - to write programs using the Kubernetes API, write plugins or extensions, or modify core code
 - [Kubernetes Developer Guide](#)
 - [notes on the API](#)
 - [API object documentation](#), a detailed description of all fields found in the core API objects
- **Walkthroughs and examples**
 - hands-on introduction and example config files
 - in the [user guide](#)
 - in the [docs/examples directory](#)
- **Contributions from the Kubernetes community**
 - in the [docs/contrib directory](#)



Documentation 3

Kubernetes

- **Design documentation and design proposals**
 - to understand the design of Kubernetes, and feature proposals
 - [Kubernetes Design Overview](#) and the [docs/design directory](#)
 - [docs/proposals directory](#)
- **Wiki/FAQ**
 - the [wiki](#)
 - [troubleshooting guide](#)

Community, discussion, contribution, and support

Consider joining the [Cloud Native Computing Foundation](#). For details about who's involved and how Kubernetes plays a role, read [their announcement](#).