Musings on Mesos: Docker, Kubernetes, and Beyond

Timothy St. Clair
@timothysc
Mesos committer and PMC member

11/19/2014

Audience Poll - Mesos

- I have ...

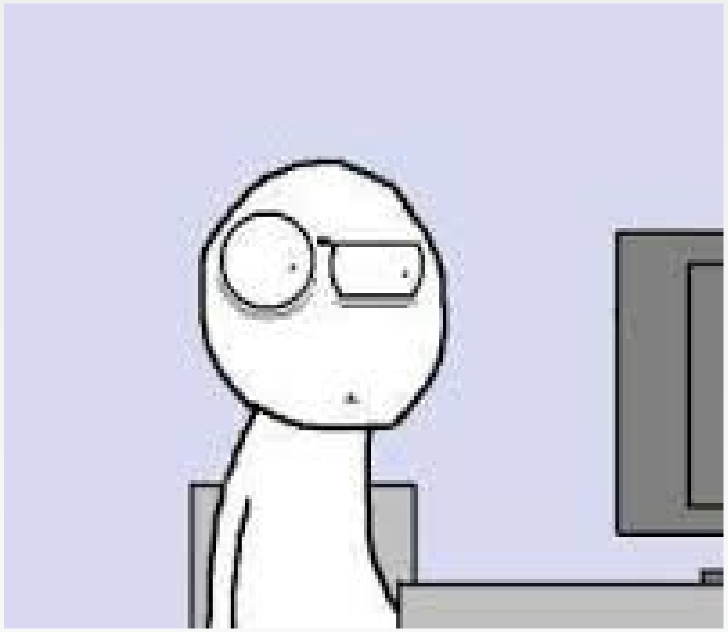
1. Only heard talks about Mesos, nothing more.
2. Played with Mesos on a sample cluster, initial POC.
3. Have a Mesos cluster in a production environment
4. DUDE.!.! I wrote a custom Mesos framework in Haskell before breakfast Bro ...

Quick Overview – Mesos 101 Condensed

At its core, Mesos is a **focused, scalable, two-phased meta-scheduler** that provides **primitives** to express a wide variety of scheduling patterns and use cases. Solutions are **written atop** of Mesos as **frameworks.** Frameworks are targeted for a particular use case, and maintain domain specific information. By remaining focused at its core, Mesos is not architecturally encumbered by domain specific problems that often exist in other monolithic schedulers.

The Response I Always Hope For

Typical Response



101 Unpacking: Meta-Scheduling?

At its core, Mesos is a ***focused, scalable, meta-scheduler*** *that provides* ***primitives*** *to express a wide variety of scheduling patterns and use cases.*

- It means that Mesos enables other distributed applications to define their own scheduling policies, with an core algorithm (DRF) to share resources across those applications.
- Mesos can run on 1-O(10^4) nodes.

So, it is a scheduler... that allows distributed applications to share resources in a cluster.

101 Unpacking: frameworks?

Solutions are ***written atop*** *of Mesos as* ***frameworks****, and are targeted for a particular use case.*

- Framework = distributed application = scheduler
- Framework could be anything from batch, application, or micro-service scheduling.

101 Unpacking: Architecturally Encumbered?

Mesos is not architecturally encumbered by domain specific problems that often exist within other monolithic schedulers.

- Mesos does not subsume domain specific problems, that is a frameworks job.
  - Service scheduling
    - Fault tolerance
    - Load balancing
    - Discovery
  - Batch Scheduling
    - Quotas (user/group)
    - Limits

101 Motivation: New Reality

- New applications need to be:
    - Fault tolerant (Withstand failure)
    - Scalable (Not crumble under it's own weight)
    - Elastic (Can grow and shrink based on demand)
    - Multi-tenent (It can't have it's own dedicated cluster)
        - Must play nice with the other kids.
- So what does that really mean?

101 Motivation: Distributed Application

- “There's Just No Getting Around It: You're Building a Distributed System” Mark Cavage
  - **queue.acm.org/detail.cfm?id=2482856**
- Key takeaways on architecture:
  - Decompose the business applications into discrete services on the boundaries of fault domains, scaling, and data workload.
  - Make as many things as possible stateless
  - When dealing with state, deeply understand CAP, latency, throughput, and durability requirements.

“Without practical experience working on successful—and failed—systems, most engineers take a "hopefully it works" approach and attempt to string together off-the-shelf software, whether open source or commercial, and often are unsuccessful at building a resilient, performant system. In reality, building a distributed system requires a methodical approach to requirements along the boundaries of failure domains, latency, throughput, durability, consistency, and desired SLAs for the business application at all aspects of the application.”

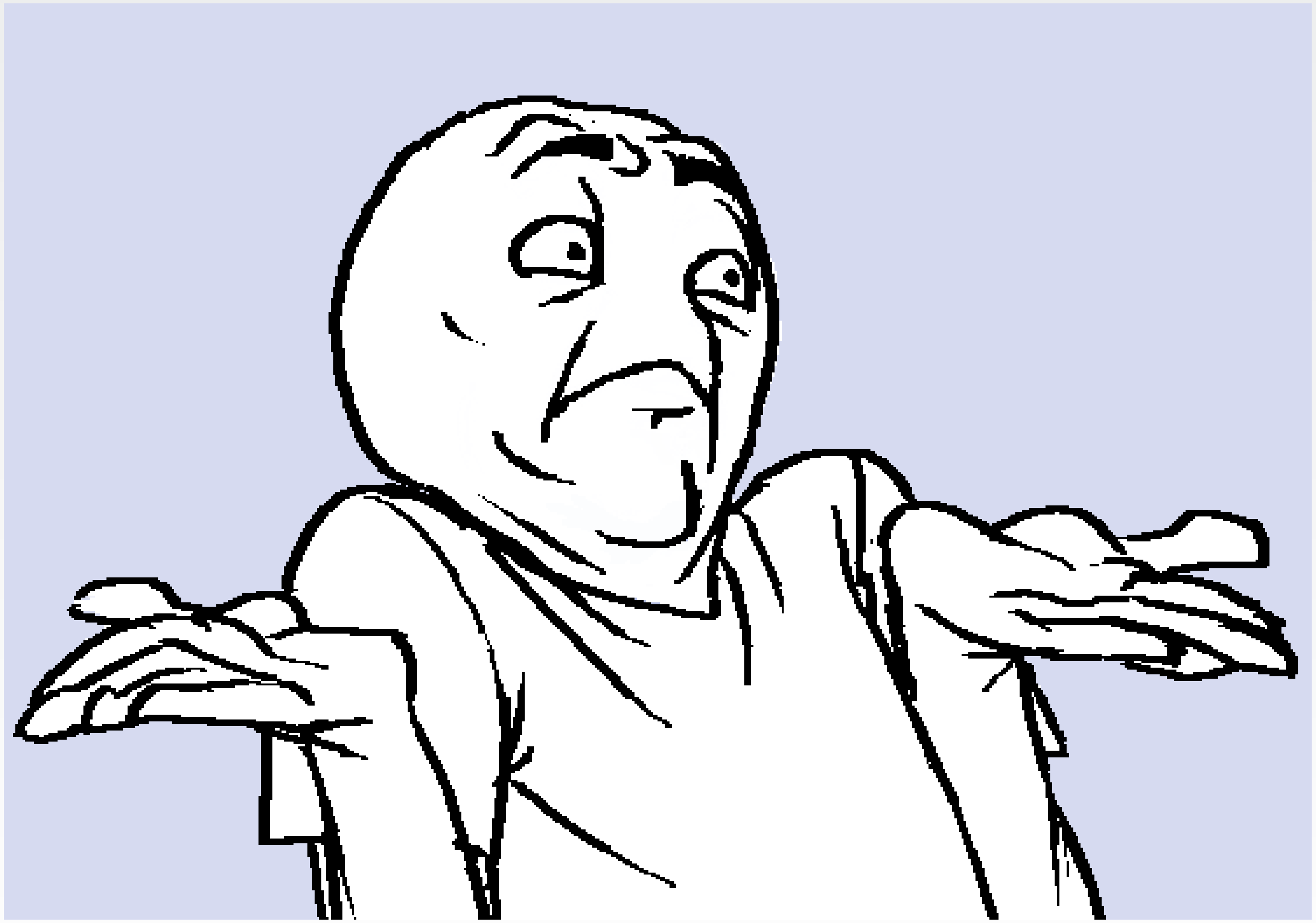
101 Summary : Distributed Systems Kernel Cluster Manager (google-ism)

- Mesos is built using the same principles as the Linux kernel, only at a different level of abstraction.

“We wanted people to be able to program for the data-center just like they program for their laptop” ~ Ben Hindman

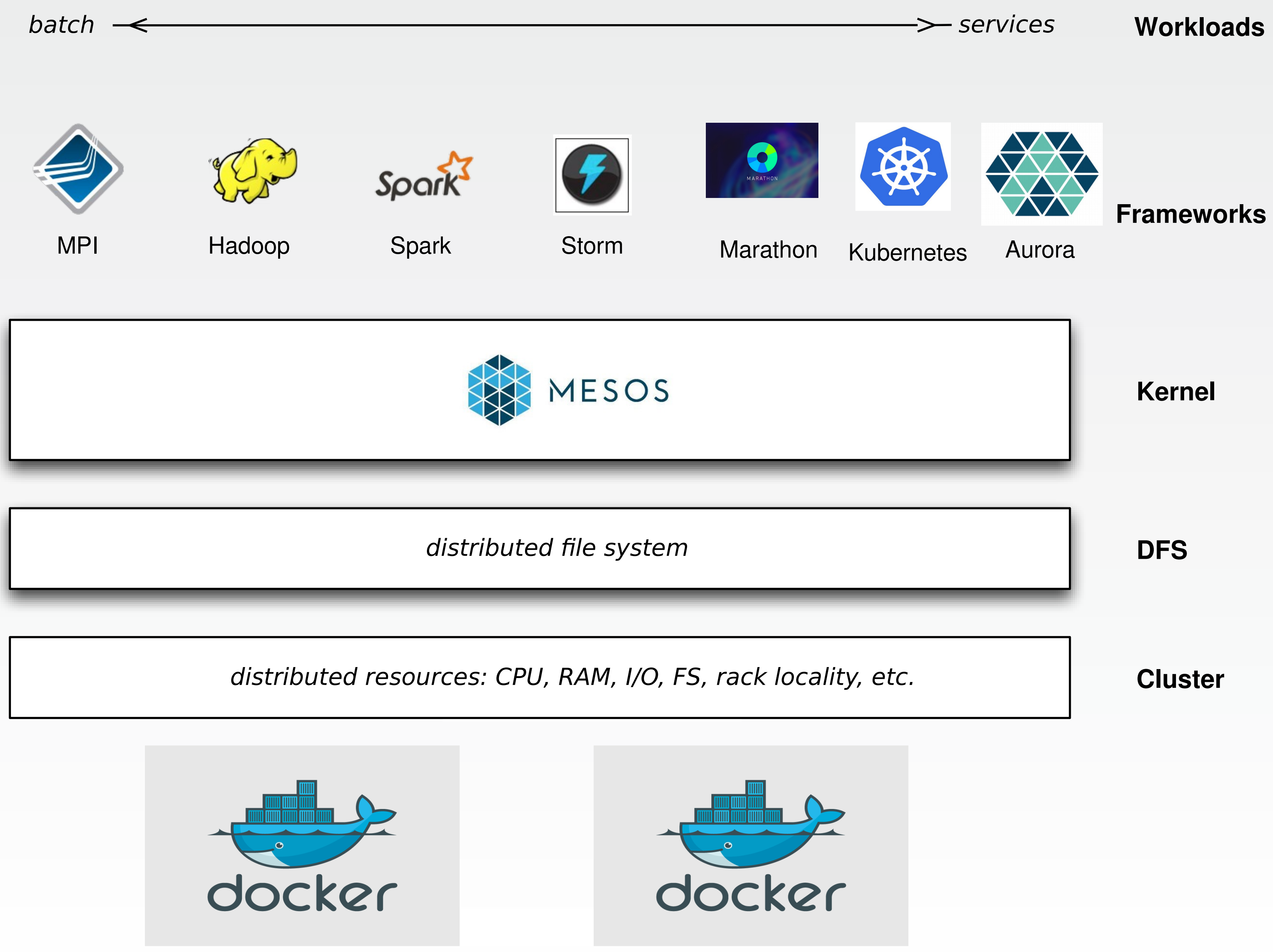
- Computer : Data-center
- Kernel : Mesos
- Application : Distributed application
- Operating System : (Mesos+Frameworks+ Ecosystem)

So Having a Kernel for Distributed Systems Makes a Lot of Sense.?.? ¯_(ツ)_/¯



Mesos @ 50K

http://mesos.apache.org/documentation/latest/mesos-frameworks/



Audience Poll - Docker

- I have ...

1. Only heard talks/news about Docker.
2. Played with Docker and use it on a single host.
3. Have Docker running in a production cluster.
4. DUDE.!.! I've ran 300 Million containers last month.

Docker Docker Docker

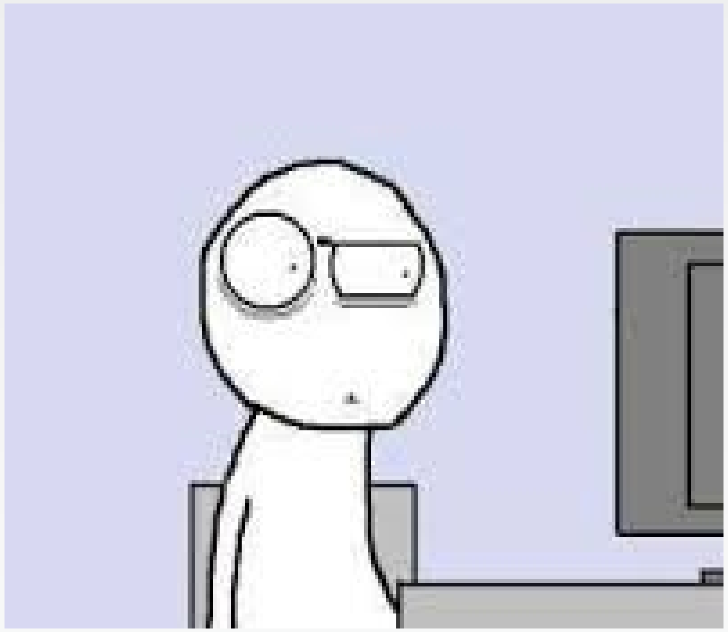
- Combination of technologies
  - cgroups + namespaces + image format + lifecycle management + application virtualization
- Git like semantics around image building
  - pull, commit, push ...
- Application virtualization
  - Native Speeds
  - >> Density the virtual machines

The 'Holy Grail' of Clustering

- Provide maximum application density per-machine (currently in the hundreds).
  - >> Then virtualization
- Improved 'application namespace' isolation across your cluster
  - No longer need to ensure stacks | sub-stacks are rolled onto your cluster.
  - Enables multiple versions across a cluster easily
- Combined with clustering, it can obviate traditional 'deployment problems'

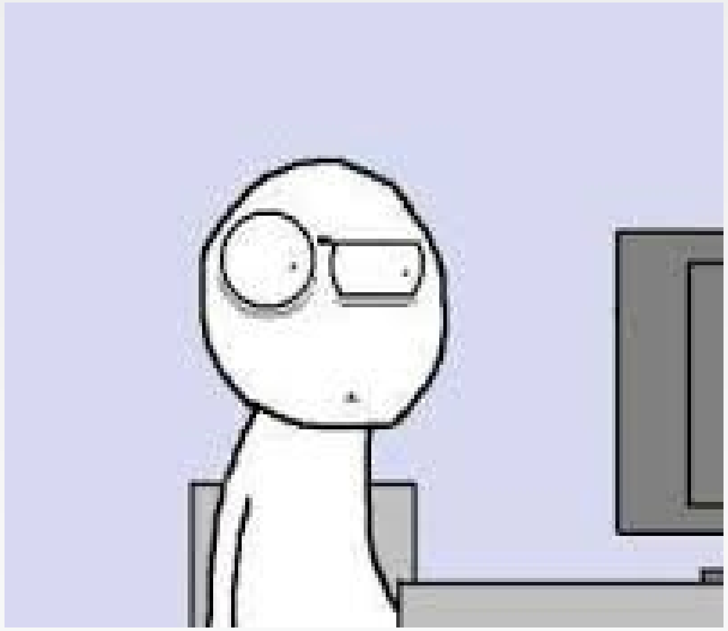
Typical Response

Reality



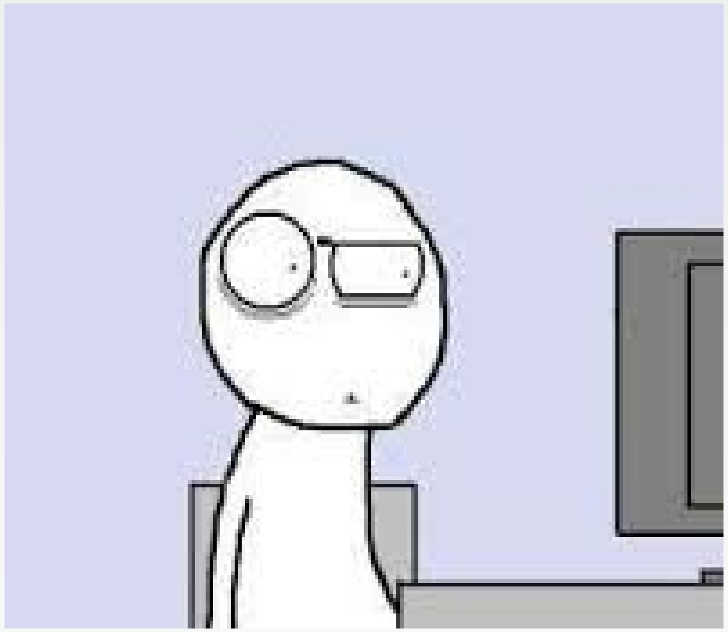
Docker 90% of the things, but that last 10% of things may require super-privileged containers, where the container has full access to the system and that gets kind of funky because then you could have dependency leakage... and potentially a host of other versioning issues. And uhhhhhhh..

Reality



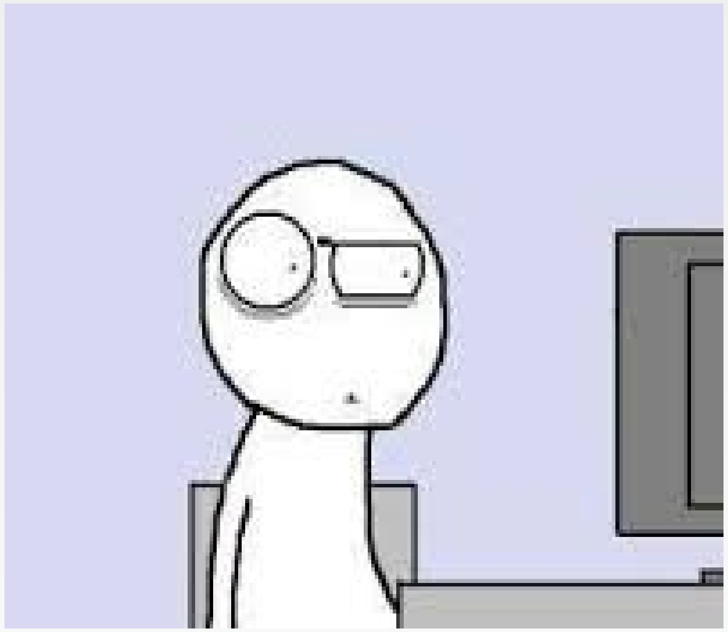
- The last 10%
  - Applications that want or need to load custom kernel drivers.
  - Low level system applications (namespaces)
  - Cluster managers, such as Mesos which leverage namespaces.

Reality



- Networking is thorny, with many solutions O(10^2)
  - Pipework
  - Flannel
  - Weave
  - vSwitch
- What happens when density is on the order 1000's.
- What happens if you wanted to have a density on the order of 10,000+.

10,000+ .?.?



- What is the average utilization of your data center?
- Twitter & Google run between 20-30% (Quasar paper)
- “ 640K ought to be enough for anybody.” - B.Gates.

Whatever

(Apache Infra) “What if.... There Were Official Apache Repositories on Docker-Hub.”

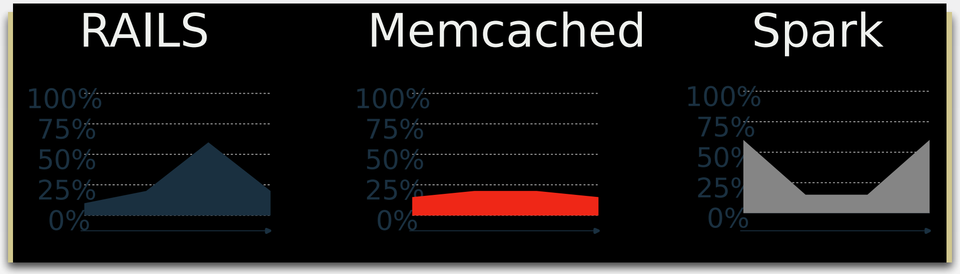
Mesos and Docker Integration

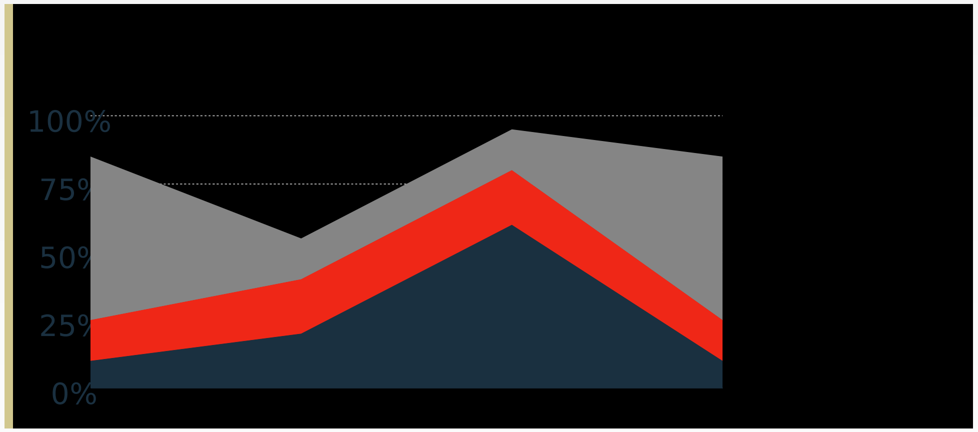
"If a Docker application is a Lego brick, Kubernetes would be like a kit for building the Millennium Falcon and the Mesos cluster would be like a whole Star Wars universe made of Legos." **~ Solomon**

- 1st classed in 0.20
  - A lot of feedback and updated 0.21 (soon-ish)
- Service Scheduling
  - Marathon + HA-Proxy + Docker – 10^4
- Kubernetes (WIP)
- Batch
  - Spark (https://issues.apache.org/jira/browse/SPARK-2691)
  - Chronos

Mesos and Docker Use Cases

- Elastic applications in a shared multi-tenant environment



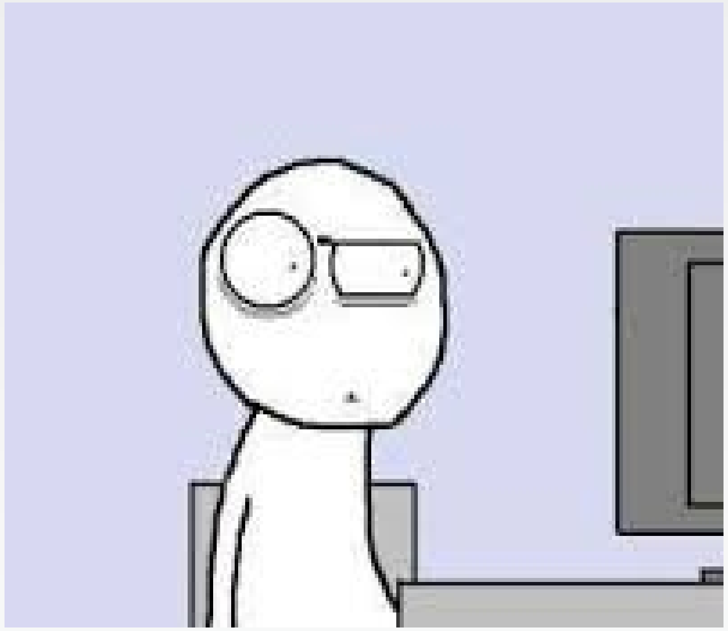


Graphic Courtesy of Paco Nathan

Can I Write My Own Framework That Uses Docker?

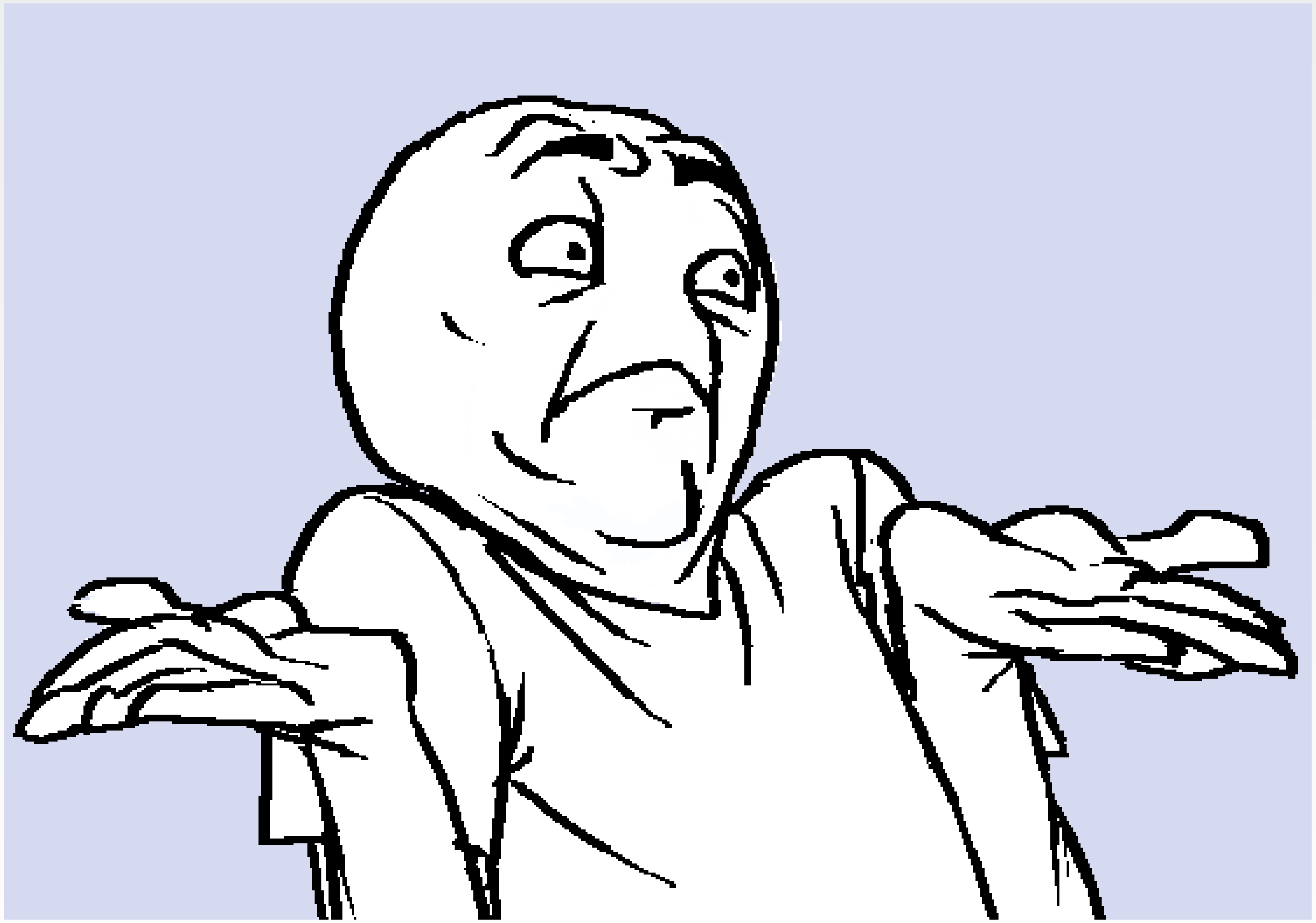
- YES!
  - Mesos has 1st class Docker support, by providing an API DockerInfo {}
- BUT...
  - Writing a framework in development is relatively easy
  - Writing an application that is highly available, scales well, can be difficult.
- HOPE
  - Creation of higher order libraries (libc) to help make it easier to create frameworks.

Can you run Mesos from a Docker Container?



- Current State – sort of.
  - mesos-master
    - Using bridged networking you can easily run a HA configuration
  - mesos-slave
    - mesos requires low level system access which makes it difficult to put in a container and maintain full feature parity
    - TBD with super privileged containers

How are we doing? ¯_(ツ)_/¯



Audience Poll - Kubernetes

- I have ...

1. Only heard talks/news about kubernetes.
2. Played with kubernetes and use it on a single host.
3. Have kubernetes running on GCE.

Kubernetes (WIP)

- Kubernetes is a system for managing containerized applications across multiple hosts, providing basic mechanisms for deployment, maintenance, and scaling of applications. Its APIs are intended to serve as the foundation for an open ecosystem of tools, automation systems, and higher-level API layers.
- Kubernetes establishes robust declarative primitives for maintaining the desired state requested by the user. These primitives are the main value added by Kubernetes.
- pre-production beta.

Kubernetes

- Pods are the atom of scheduling, and are a group of containers that are scheduled onto the same host.
- Pods facilitate data sharing and communication.
  - Shared mount point
  - Shared network namespace/IP and port space
  - Higher order abstraction to the low level interface of containers
  - Composable micro-services

Kubernetes – pod.json

```
{  "apiVersion": "v1beta1",
  "kind": "Pod",
  "id": "redis-master-pod",
  "desiredState": {
    "manifest": {
      "version": "v1beta1",
      "id": "redis-master-pod",
      "containers": [{
        "name": "redis-master",
        "image": "gurpartap/redis",
        "ports": [{ "name": "redis-server", "containerPort": 6379 }] ...  + labels
```

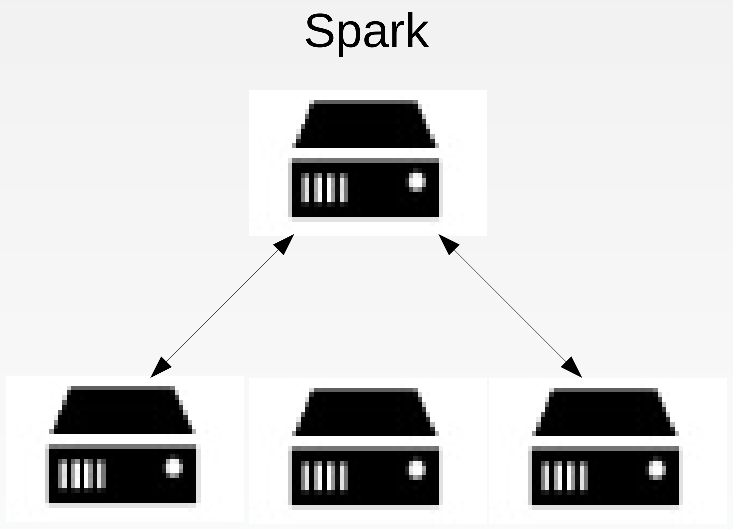
Kubernetes framework for Mesos (WIP)

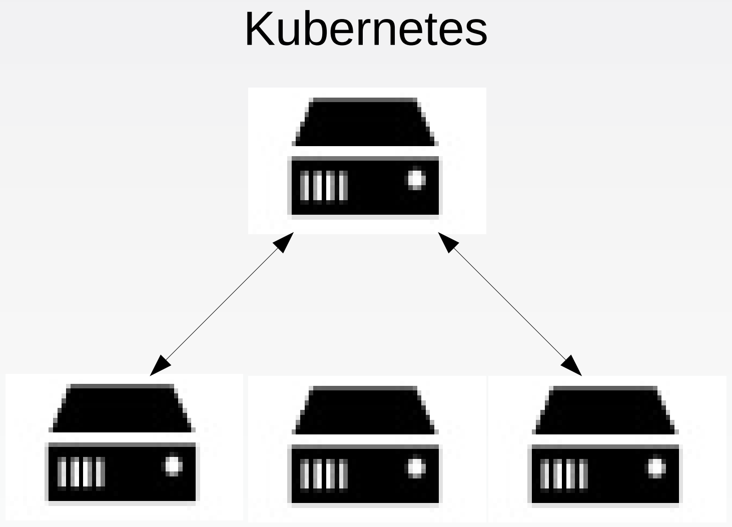
- Kubernetes enables the Pod (group of co-located containers) abstraction, along with Pod labels for service discovery, load-balancing, and replication control. Mesos provides the fine-grained resource allocations for pods across nodes in a cluster, and can make Kubernetes play nicely with other frameworks running on the same cluster resources.
  - Provide advanced scheduling capabilities (grouping, spreading ...)
  - Availability zone fail-over
  - Elasticity – Scale up and down kublets
  - **MORE TO COME...**

Kubernetes vs. Mesos?

- **Not at all!**
  - Kubernetes is an opinionated declarative model on how to address micro-services
  - Mesos provides an imperative framework by which application developers can define scheduling policy in a programmatic fashion.
  - When leveraged together it provides a data-center with the ability to both.

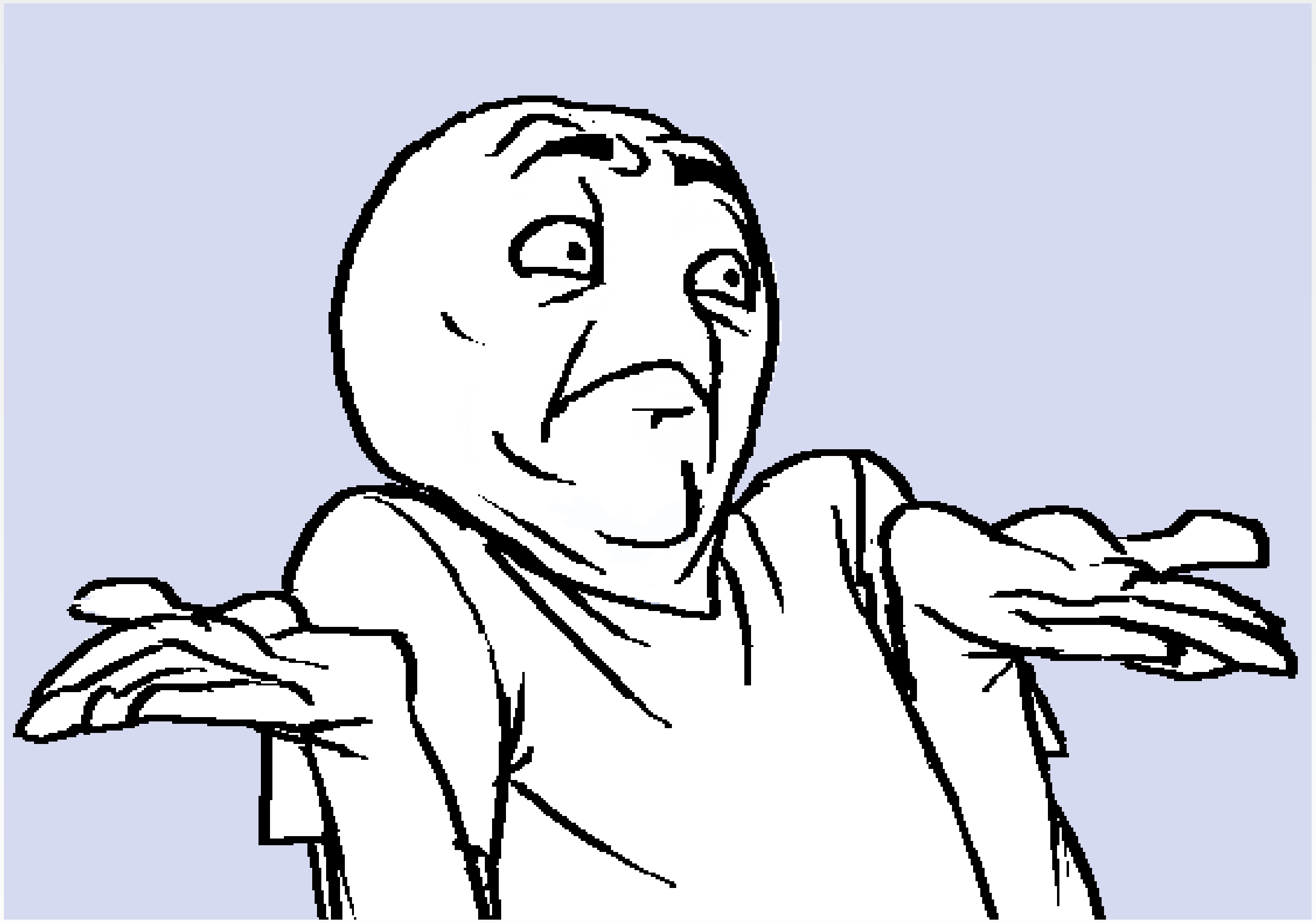
Kubernetes - Mesos = Static Provisioning





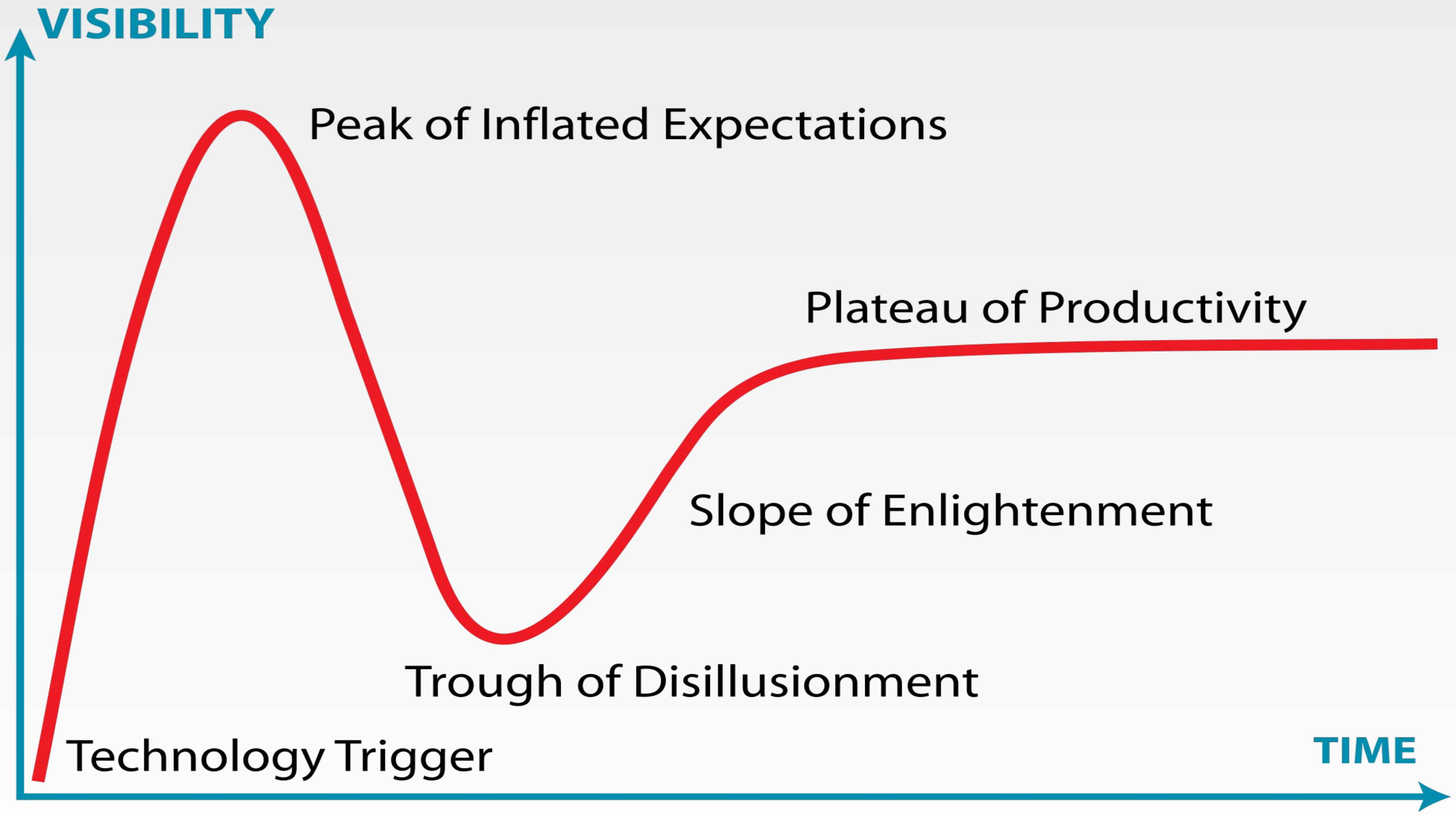
Hows your social media updates?

#Mesos + #ApacheCon FTW ¯_(ツ)_/¯



BEYOND

Mesos, Docker, and Kubernetes are Still Very Young. Expect change...



1st Classing PODS into Docker Core

- https://github.com/docker/docker/pull/8859
  - Make a POD become the fundamental unit of abstraction
  - Make the Container (Letter), while the POD (Envelope), Clustering becomes the mail service.
  - Problem – Most user have no idea what a POD is
  - Answer - They don't really need to, it could likely be hidden or renamed.

Docker Clustering

- https://github.com/docker/docker/pull/8859
  - Clustering will be done via a plugin whose API is TBD but due out later...
  - The Docker CLI is meant to be the primary interface to support Docker clustering, and the plugin will extend the CLI via options (TBD). e.g. --constraints=TBD
  - Kubernetes, Mesos, or possibly Marathon, could be added as a back end via the plugin API.
  - Any concepts, such as 1st classing PODs is a docker-core constraint.

Native Docker Multi-Host Networking

- https://github.com/docker/docker/issues/8951
  - The power of Docker is its simplicity, yet it scales to the demands of hyper-scale deployments. The same cannot be said today for the native networking solution in Docker. This proposal aims to bridge that gap. The intent is to implement a production-ready reliable multi-host networking solutions that is native to Docker while remaining laser focused on the user friendly needs of the developers environment that is at the heart of the Docker transformation.
  - Programmable vSwitch

Hand Wavy - Future of a Kubernetes Framework atop Mesos

- Drive towards parity with google cluster management tools #GCELive.
  - Elastic model around micro-services
    - If (ServiceX.cpuload>80) then; Mesos.addMoreCapacity()
- Workload migration and maintenance.
  - Offload nodes for maintenance.
  - Resize and defrag.

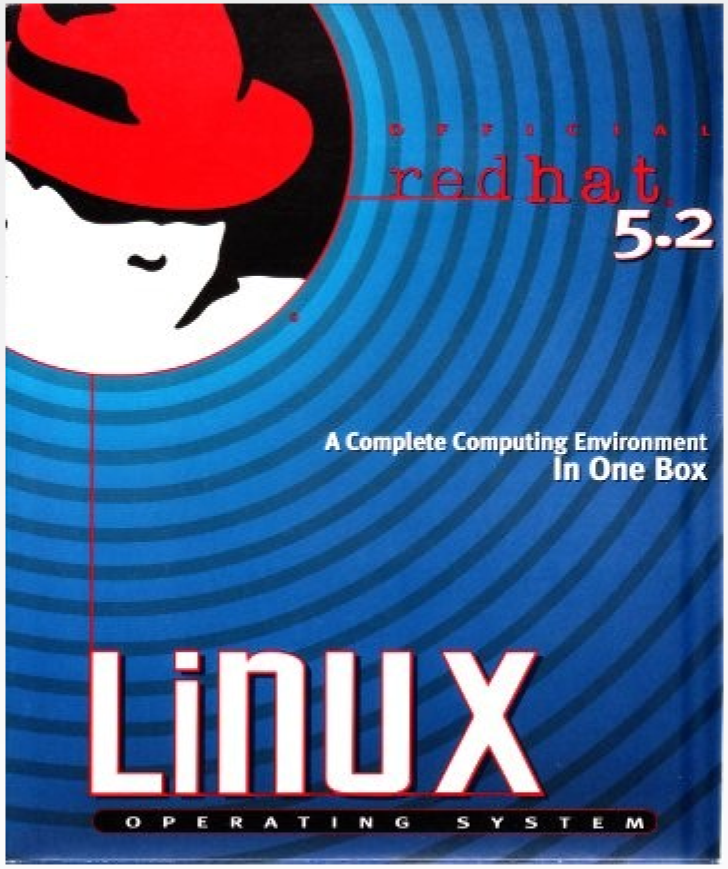
What Does This Mean for Mesos

- Integration changes on the horizon
  - Internal container handling.
  - Framework API – DockerInfo {}, will likely change
- Docker CLI could be used to drive containers on a Mesos cluster.
- Mesos mantra continues unabated: “Multiple models all supported by one common infrastructure”
  - Multiple service models (Marathon, Kubernetes, Aurora)
  - Multiple batch models (Spark, Chronos)

BUT WAIT! THERES MORE
On the Horizon – Core Mesos Features

- Primitives to support stateful services (HDFS, Cassandra, etc.)
  - Persistent disk data
  - Lazy resource reservation
  - Expanded Disk Isolation
    - I/O
    - FS/Mount namespace
    - Support multiple spindles
  - Dynamic slave attributes

Mesos and the Surrounding Ecosystem as an Operating System.



- Just getting started, first #MesosCon (Aug)
- Even though Mesos (kernel) has matured @ Twitter, it is still very early on.
- We are still in the very early stages for both Docker and Kubernetes.

THANK YOU!
@timothysc