Fault Tolerant Stateful Services on Kubernetes

Timothy St. Clair
@timothysc

heptio

Thank You For Attending!

Many thanks to our:

Sponsors!

Fast Forward Fest Organizers!

Big Data Wisconsin Organizers!

Who ... Kubernetes?
(has heard of, has tinkered with, is using)

More like a “Road Trip” through space-time of cluster management to arrive at running Stateful Services on Kubernetes

heptio

Ground Rules

- Questions GOOD, too many questions BAD.
- Spread the Word - #Kubernetes, @timothysc, @BigDataWI, #FwdFest, @Heptio
- Going to gloss over container technology

Who is this guy?

I'm no Kelsey Hightower, ***but...***

I've been working exclusively on grid, cluster management, and big data engines for about a decade.

- I love this space b/c it's the leatherman of computer science. (This $HI& is fractal) ~jbeda
- Condor alum
- Mesos committer and PMC member
- Active SIG lead and core contributor to the Kubernetes project since inception...
- Staff Engineer @ a Seattle Based Startup Heptio
    - We are spreading the gospel of kube

What I’m going to talk about today...

- Brief history and motivations of Kube
- Core concepts and components (Kube 101)
- BD Workload and application profiles
- Stateful integrations into the platform
- Hand-wavy “FuTuRe”

LET'S TALK ABOUT K8S!!!

heptio

What is Kubernetes … 2014 Definition

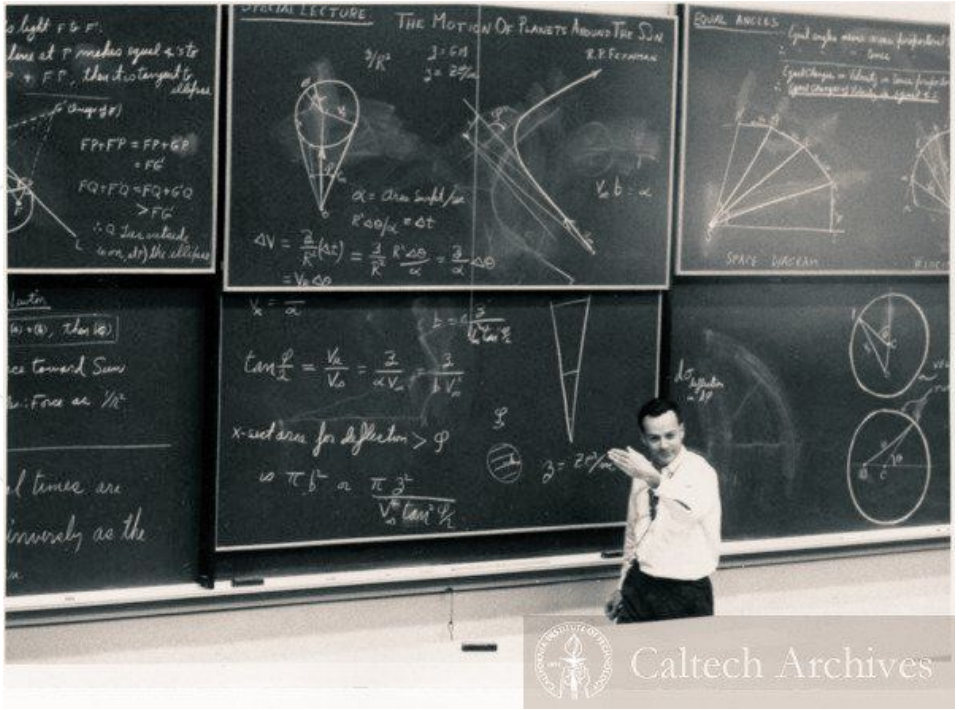
Kubernetes is an open source system for managing containerized applications across multiple hosts, providing basic mechanisms for deployment, maintenance, and scaling of applications.

What “It” is really depends on your perspective, as each actor will have their own take on kubernetes.

- Operator
- Developer
- IT/Manager

Richard Feynman

~You're unlikely to discover something new without a lot of practice on old stuff.



heptio

Background / History

Where did kubernetes come from?

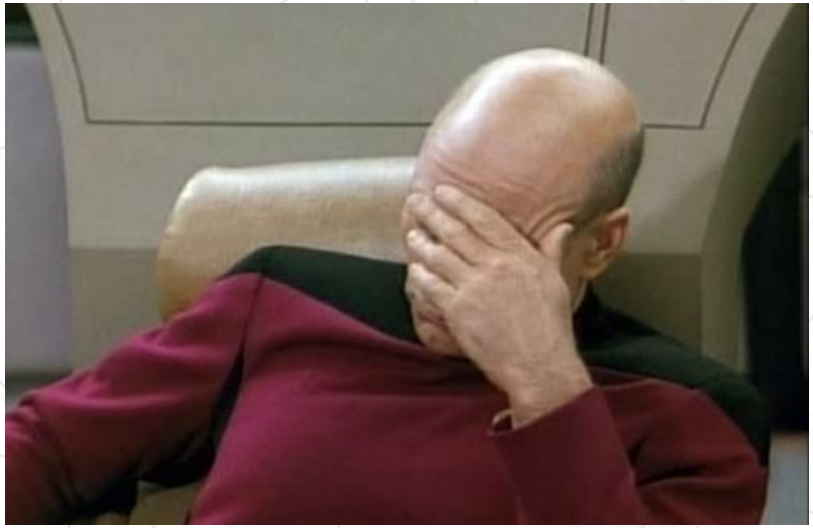
Kubernetes builds upon a decade and a half of experience at Google running production workloads at scale using a system called Borg, combined with best-of-breed ideas and practices from the community.

Why create Borg in the first place?

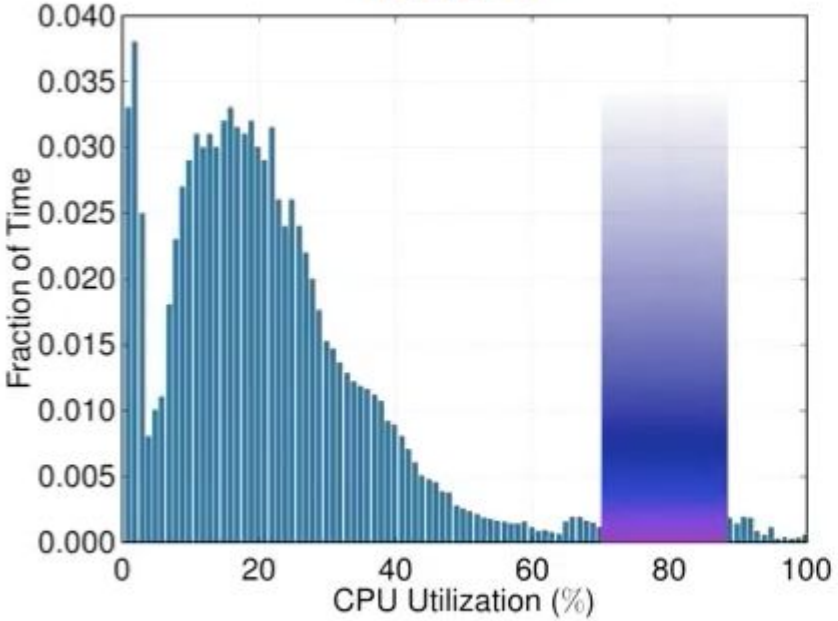
- Traditional models don't scale.
  - Poor Utilization
    - $$ - People and HW.
  - Application lifecycle management
    - Mean time to update SW

Motivations - Utilization

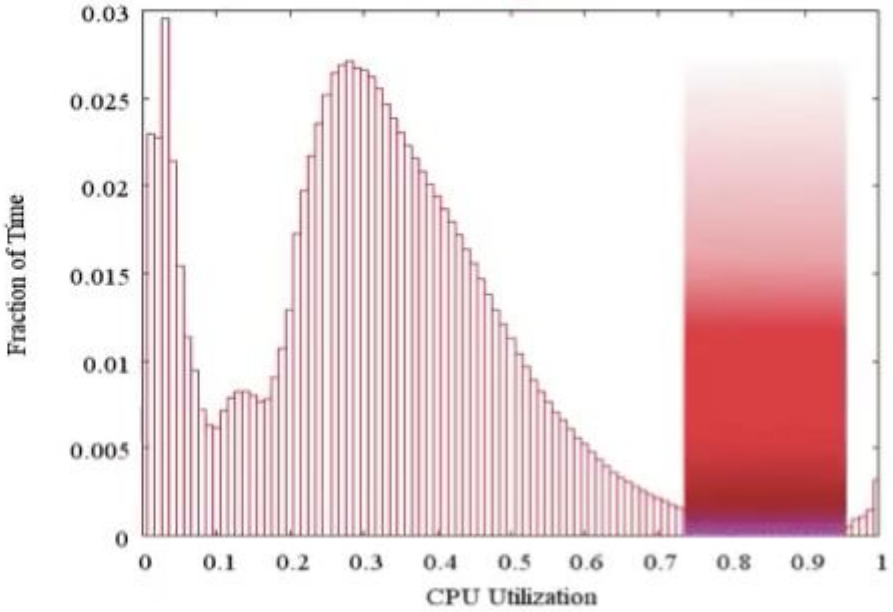
Average cluster utilization is ~5-10% if you're good. Sans Cluster Managers.



Twitter[1]



Google[2]



Improving Resource Efficiency with Apache Mesos ~ C. Delimitrou

“The reason you can do BIG DATA ‘in general’ is because there is a huge amount of slack resources in these systems”

– Eric Brewer (Dockercon 2014)

Other Dimensions to Utilization - “Good Fences”

Application density is and oversubscription mechanisms are another dimension that matters. In order to allow that there needs to be better isolation and accounting.

2005: cpusets

2006: cgroups

2009: bandwidth fair use, QoS level

2010: memcg for accounting.

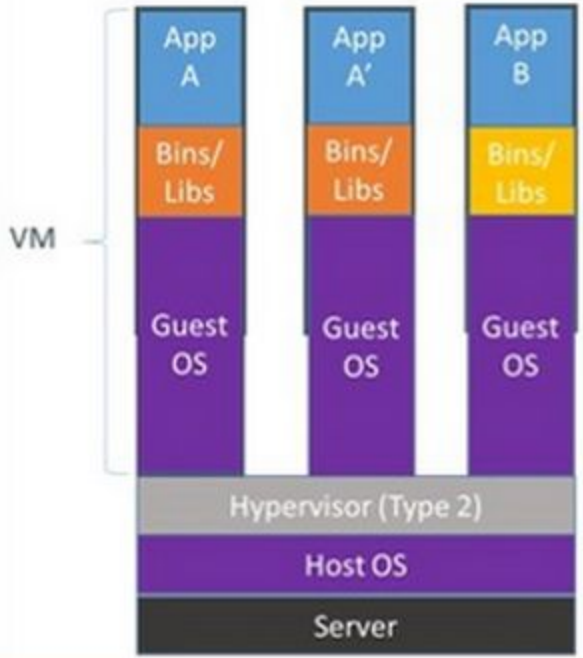
Namespacing.

Enter Containers

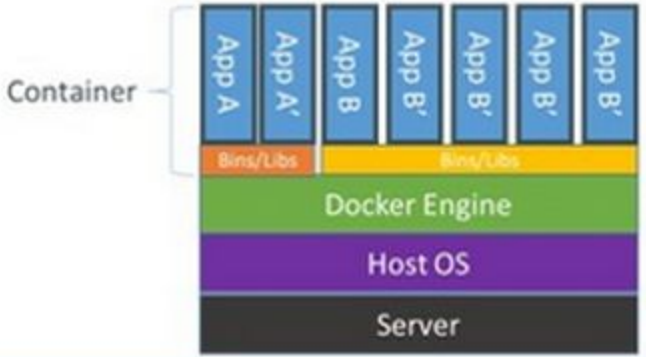
Around 2014 containers became more usable and provided a means to bundle applications and their dependencies and allow for applications to be sent transmitted to any machine.

This is a key differentiation from other cluster management tools such as YARN.

Containers vs. VMs



Containers are isolated, but share OS and, where appropriate, bins/libraries



heptio

Motivations-
Application Lifecycle Management

Application Lifecycle Management is a PITA

Even today there is a “cornucopia of tooling”:

- Chef
- Puppet
- Ansible
- Salt
- Terraform
- Custom Scriptery

Managing (N) service stacks in across a fleet is a mess without a set of unifying principles, and definitions of what those applications do.

heptio

Modeling of Applications

- Application components and aspects of lifecycle are broken down into ***declarative abstractions*** that can be used to deploy and manage applications.
- Applications reach ***eventual consistency*** through the use of independent controllers that are typically associated 1:1 with a given primitive.
- ***Connectivity of applications is abstracted*** through the use of services, or cluster resolvable endpoints.

Re-phrased

“Kubernetes is an application deployment, and lifecycle management system that enables higher utilization of your clusters resources.”

Or …

“It’s a scalable way of managing applications across a fleet of machines”

Or …

Core Concepts - Kube 101

heptio

Pods

- Pods are the atom of scheduling and are ***a group of containers*** that are scheduled onto the same host. “co-scheduling”
- Pods facilitate data sharing and communication between containers within a pod.
  - Shared mount point, ip, namespaces …
- Benefits
  - This is very unique… mesos, yarn, condor, lsf, sge don’t have this
  - Helps a lot with managed complexity
  - Great for producer / consumer models to reduce complexity
    - Security

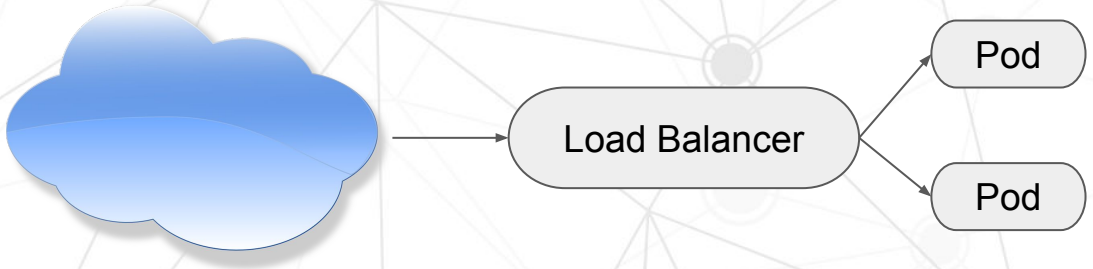
Declaratives & Controllers

- Eventual consistency is maintained by separate controllers typically associated per primitive (ReplicaSet, DaemonSet, Deployments, Pods)
- The api is constructed via a composable layered model
- Each controller's purpose is to rectify any discrepancy between the declared state of a primitive with the current state of the system

```
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: frontend
spec:
  replicas: 3
```

Services

- Services provide a single, stable name and address for a set of pods. The typically act as a load-balanced proxy endpoint. (non-colliding-nat)
- Cloud based implementations have native support for connecting to external load-balancers, otherwise users will need to manage ingress points
- Provides a construct which is used to lookup, name, and link pods (env injection).



Labels

- Are key/value pairs associated with pods and nodes
- Labels enable operators to map their own structures and deployment configurations onto objects in a loosely coupled fashion
- Newer resources, such as Job, Deployment, Replica Set, and Daemon Set, support *set-based* requirements as well.

```
metadata:
    labels:
      app: guestbook
      tier: frontend
```

```
$ kubectl get pods -l tier=frontend
```

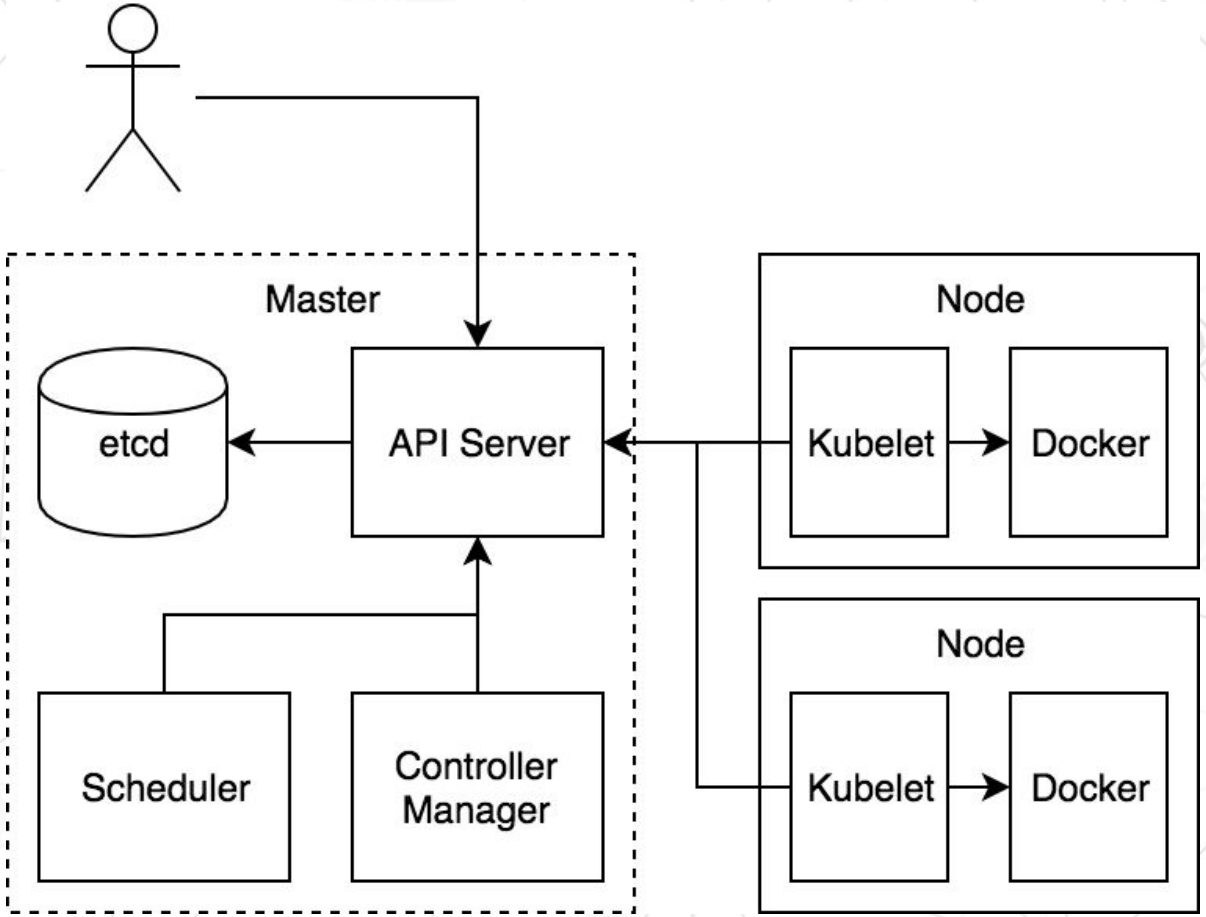
Storage - PVs and PVC

- Provides a means to attach a mounted network attached storage facility to a Pod
- Provides an API for users and administrators that abstracts details of how storage is provided from how it is consumed
- Great for stateless web-applications… Slow for stateful services

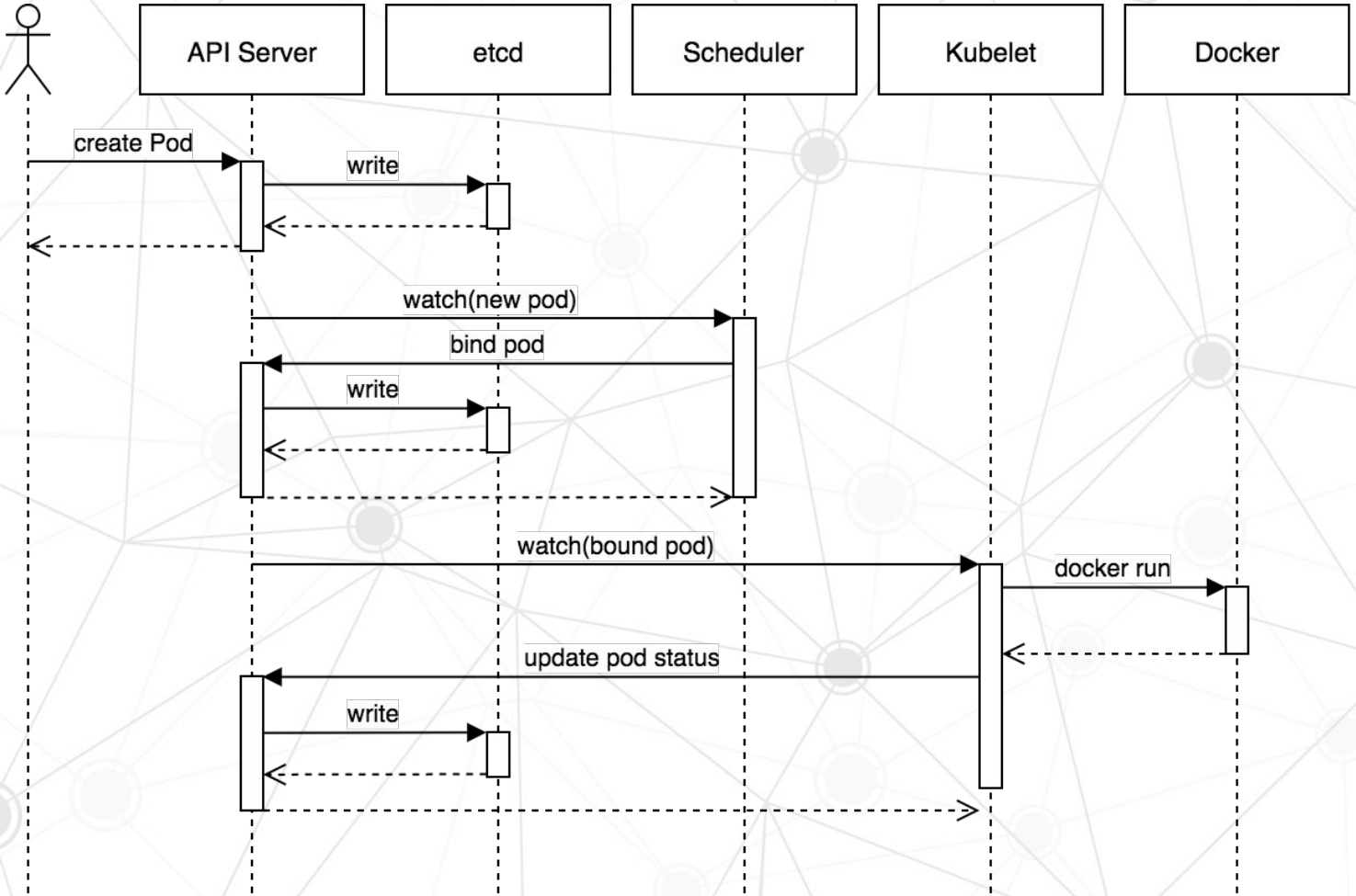
That was a lot if you are new to this, how are you doing?

Core Components and Behavior - Kube 101

heptio


Master
etcd
API Server
Scheduler
Controller Manager
Node
Kubelet
Docker
Node
Kubelet
Docker

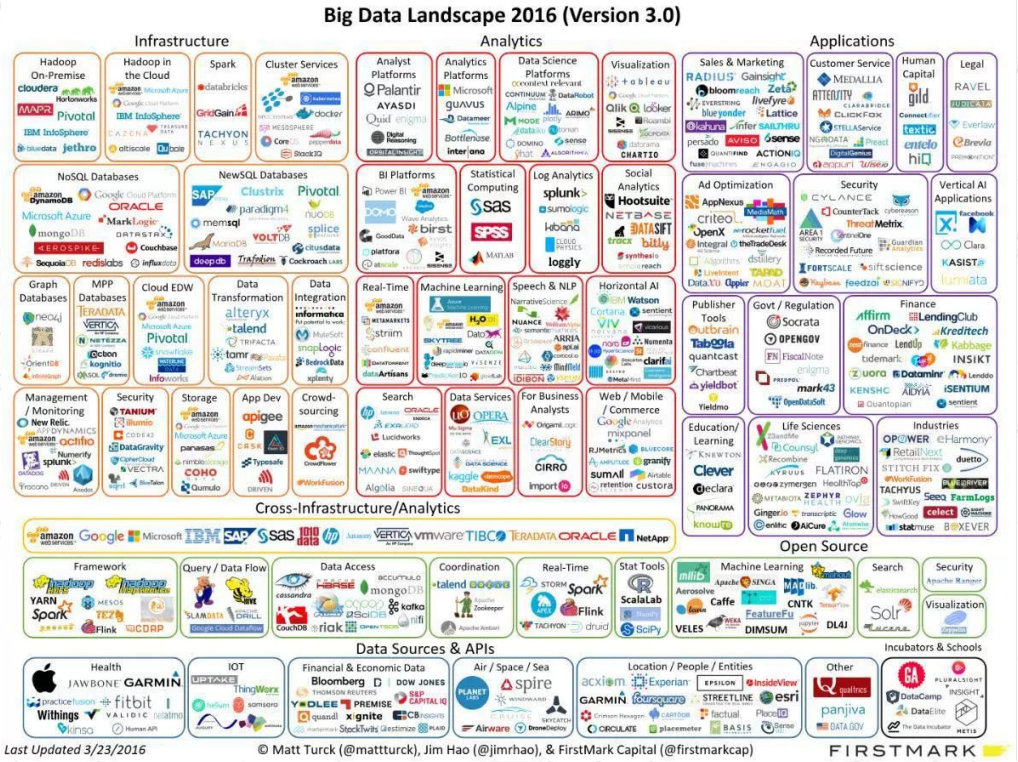
heptio


API Server
etcd
Scheduler
Kubelet
Docker
create Pod
write
watch(new pod)
bind pod
write
watch(bound pod)
docker run
update pod status
write

heptio

Workload & Application Profiles

Understanding your workloads



heptio

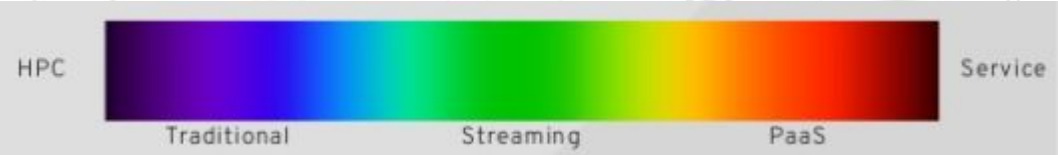
“The reason you can do BIG DATA ‘in general’ is because there is a huge amount of slack resources in these systems”

– Eric Brewer (Dockercon 2014)

“Sounds all rainbows and kittens, but there is a lot of tooling and policy required to make it not suck.”

– Me (Now)

Big Data Applications are … “Special”



- The more service oriented BD workloads “tend” to be…
  - More “stateful”
  - Have fault tolerance built in, and leverage local storage (ES, Cassandra)
  - Mindfully elastic, controlled scaling (Operator)
  - Require stricter naming and ordinality (ZK)
- The more batch oriented “tend” toward …
  - “Best-Effort” QoS tiers
  - Would love to be greedy use as many resources as it can
    - OM NOM NOM your resources are delicious

heptio

Gut Check, “How are you holding in there?”

Stateful Services
Guide Rails not Tracks

heptio

Stateful Services

- Requires Several Other Primitives:
  - StatefulSets are a declarative primitive for Pods / Applications Care about
    - Ordinality (0,1,2 …) [ZK]
      - Order stand-up and tear-down.
      - web-{0..N-1}.nginx.default.svc.cluster.local
    - Stable Network Matters
  - Taints and Tolerations
    - Storage needs to be treated with care.
      - Leverage Taints and tolerations
        - TolerationSeconds is useful
  - PodDisruptionBudgets
    - Prevent large disruptions
  - Anti-affinity (avoid same hosts)

Stateful Services

Let’s walk through an example vs. 20 more slides:

https://kubernetes.io/docs/tutorials/stateful-application/zookeeper/

Understanding the Boundaries of Declaratives

- Controlled Scaling Events
  - Rebalancing
  - Remove a Member / Assign a new Member
  - Controlled / planned replication
- When in doubt look for an “Operator” or purpose built controller
  - “An operator is a piece of software that essentially knows a lot of the operational best practices, for whatever piece of software the operator is built for and can deploy that piece of software in a good configuration on top of Kubernetes, and then keep it healthy over time.”

Crystal Ball Prognostications

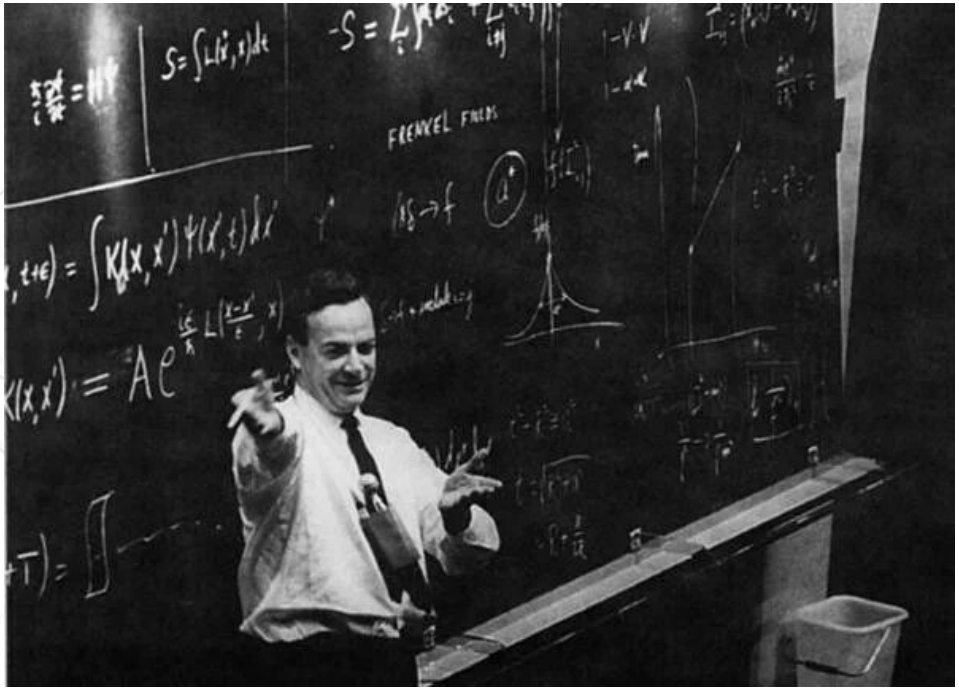
heptio

HaNd-WaVy “fUtUrE”

- Operators for more BD applications and management of legacy stateful services
  - ES, etcd, kafka, ZK.
  - Native Spark Integration
  - Kubernetes Arbitrator for more flexible policy and NS borrowing
- New native BD applications
  - Kubernetes has all the primitives available to create your own new framework which is native to the system
  - Don’t reinvent the square wheel

Richard Feynman

I have approximate answers and possible beliefs and different degrees of certainty about different things, but I'm not sure of anything!



heptio

Thank you!

inquiries@heptio.com