Progress for big data in Kubernetes

TED DUNNING

Chief Application Architect

Contact Information

Ted Dunning, PhD

Chief Application Architect, MapR Technologies

Board member, Apache Software Foundation

O'Reilly author

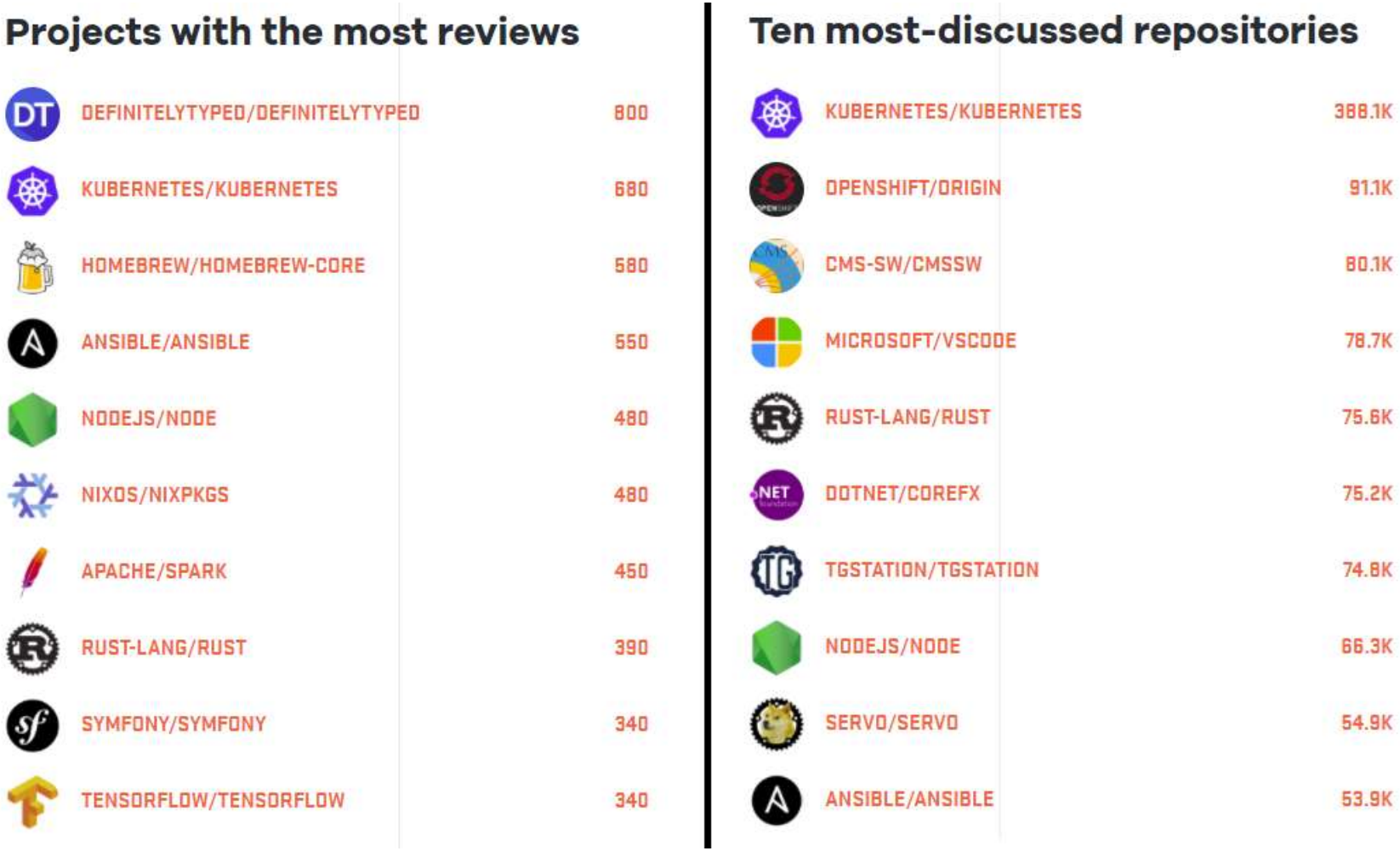
Email tdunning@mapr.com tdunning@apache.org

Twitter @ted_dunning

kubernetes is coming!

why?

kubernetes = major community support



Source: Shippable.com http://blog.shippable.com/why-the-adoption-of-kubernetes-will-explode-in-2018

every cloud supports kubernetes

https://www.sinax.be/en/aws/
https://www.westconcomstor.com/za/en/vendors/wc-vendors/microsoft-azure-EN-UK.html
https://www.g2crowd.com/products/google-kubernetes-engine-gke/details

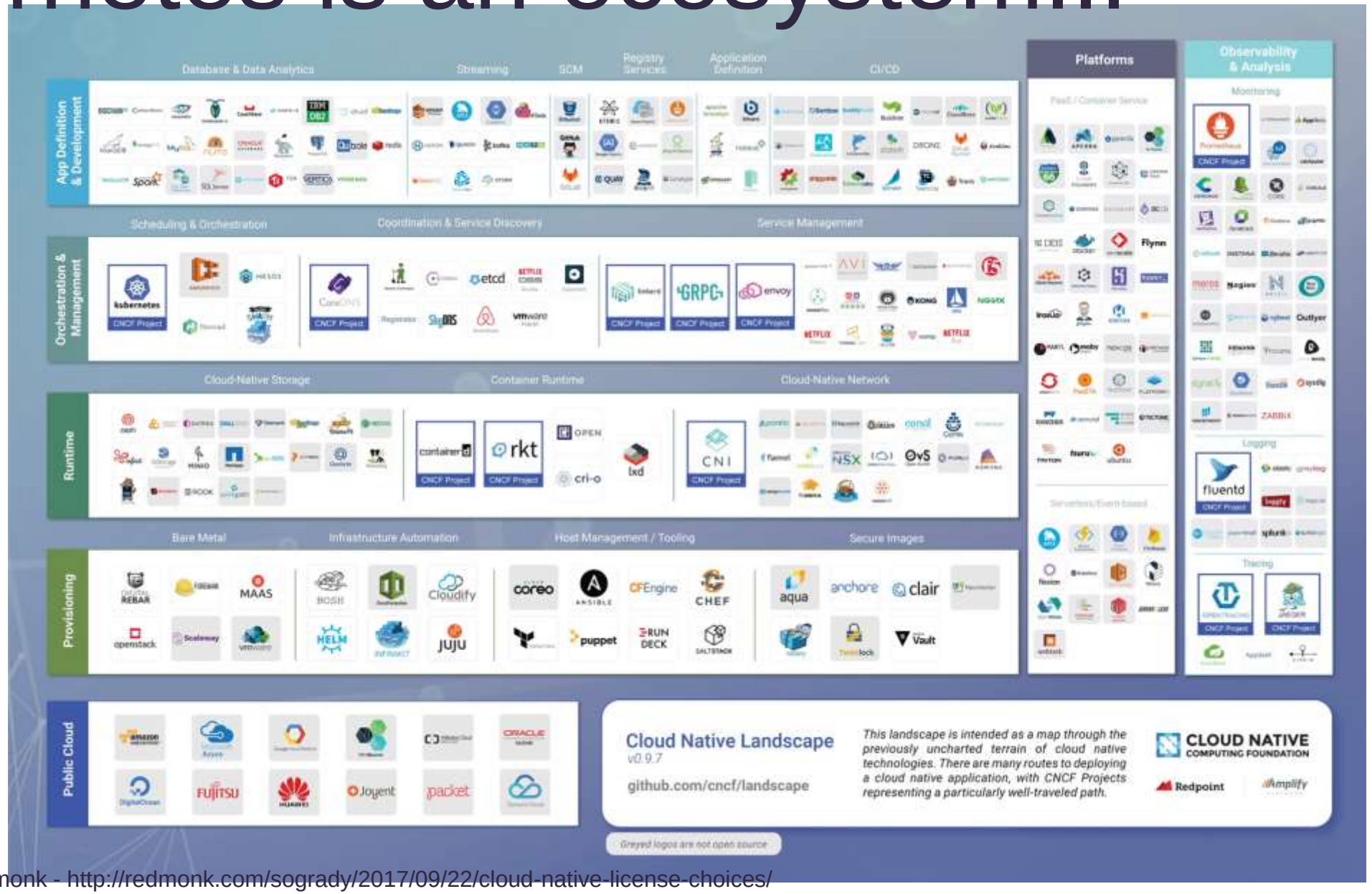
massive customer adoption rate

what is kubernetes?

kubernetes (n.) - *greek word for pilot or helm*

https://cloud.google.com/security/encryption-in-transit/

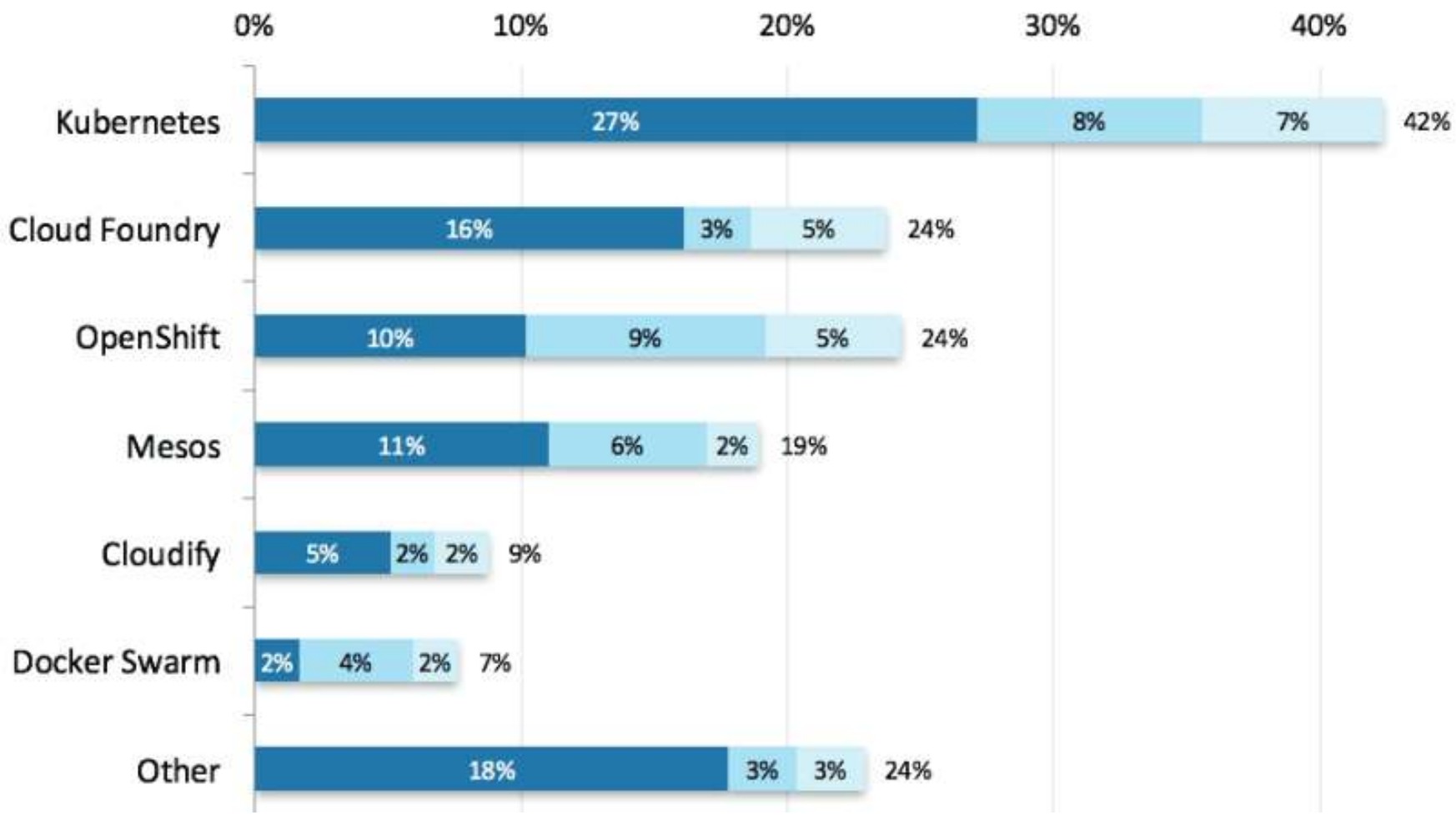
kubernetes is an ecosystem...



Source: Redmonk - http://redmonk.com/sograd y/2017/09/22/cloud-native-license-choices/

container and resource orchestration engine...

kubernetes won the container orchestration war...



Source: Shippable.com http://blog.shippable.com/why-the-adoption-of-kubernetes-will-explode-in-2018

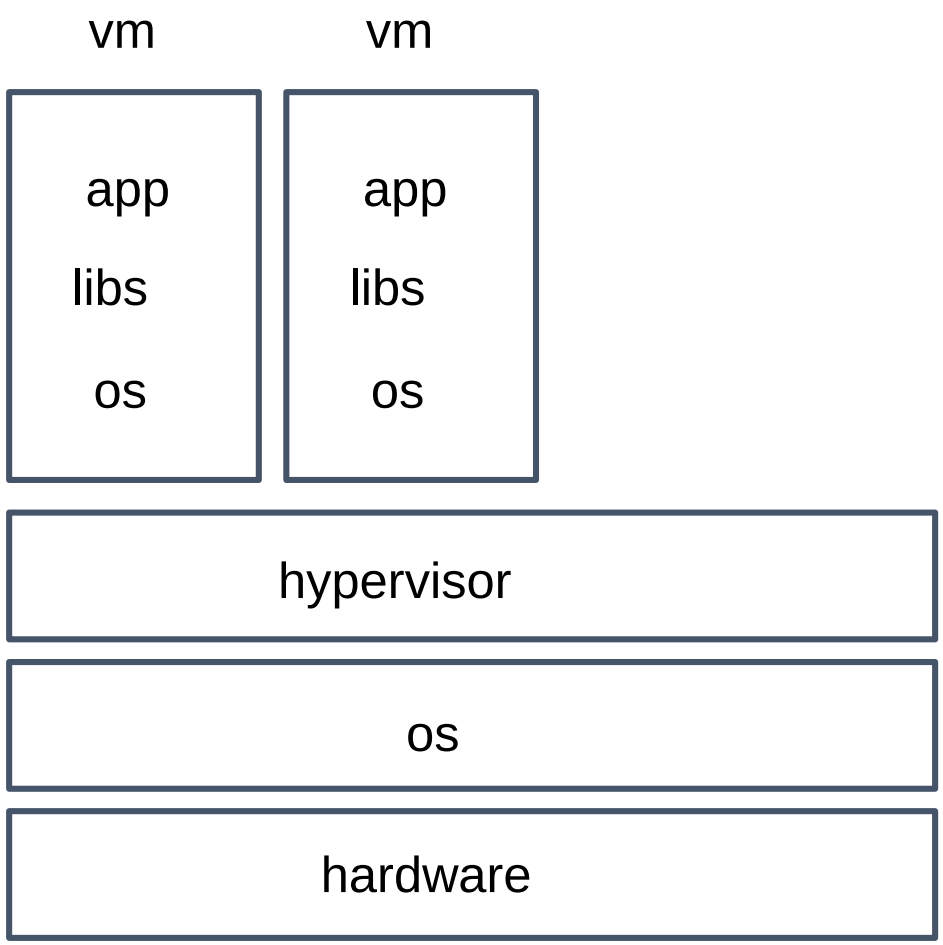
what is kubernetes?

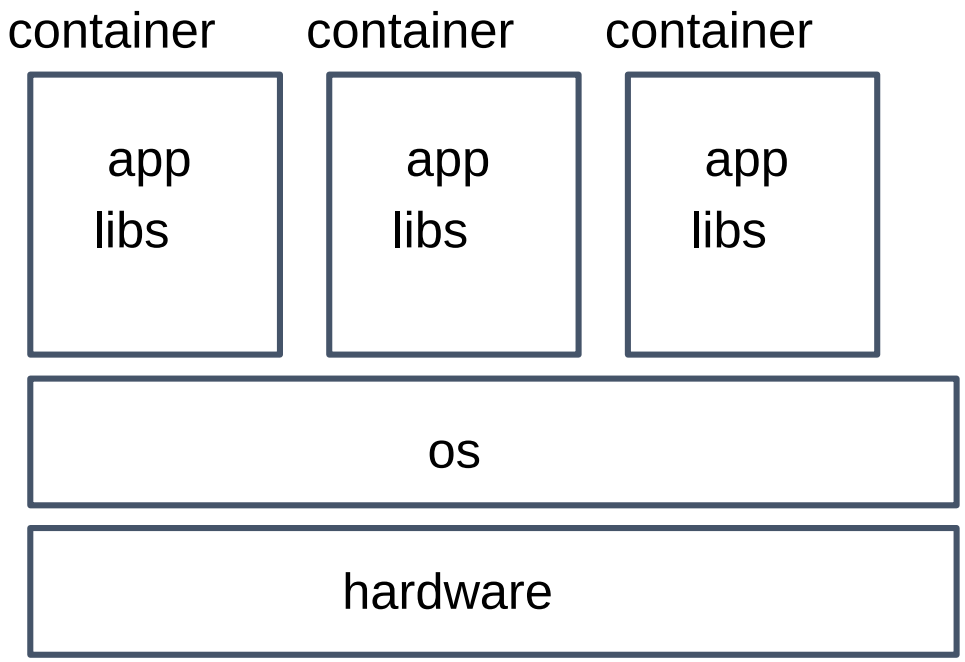
it runs containers

what is a container?

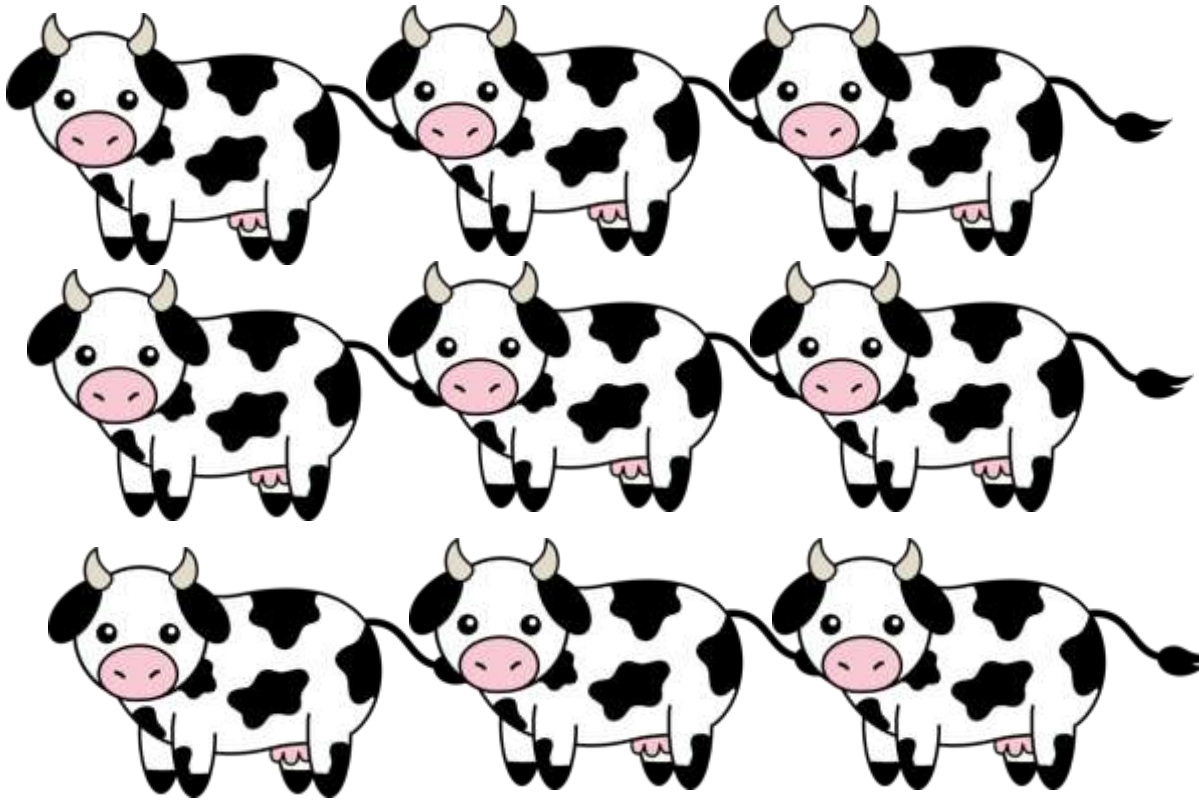
not a vm

vm vs container



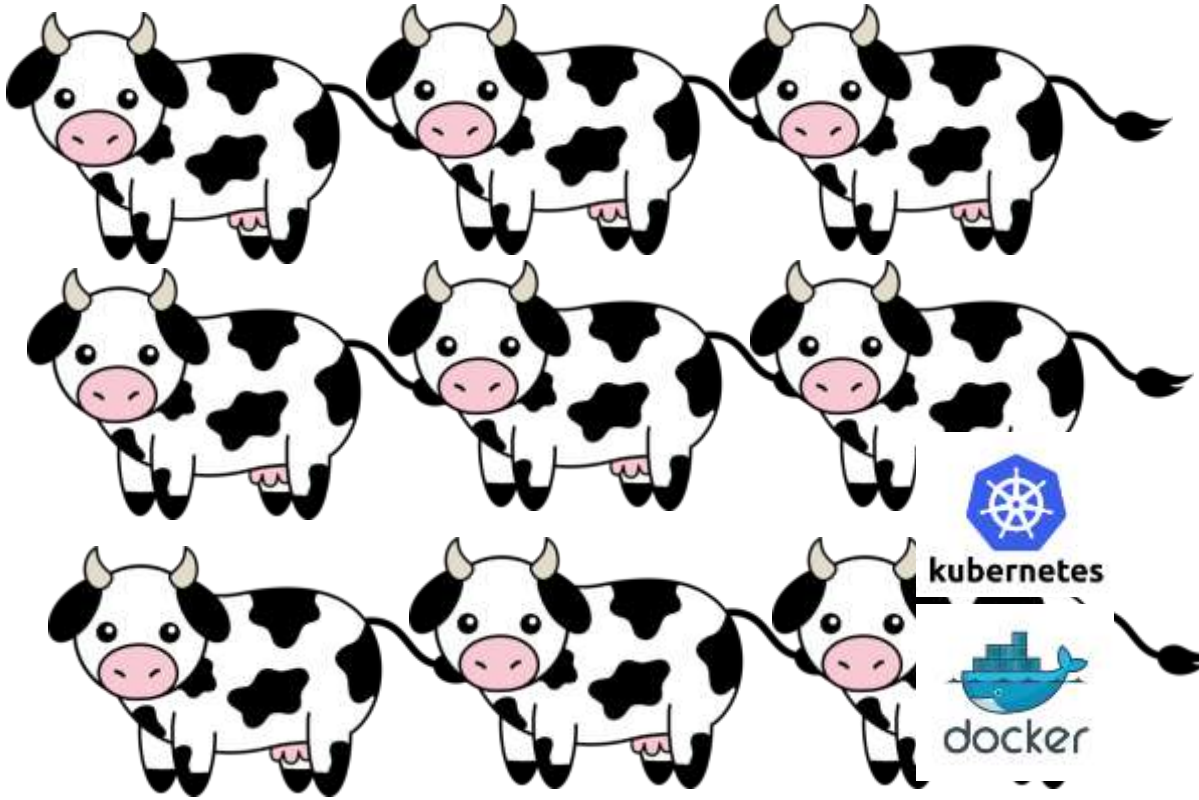


pets vs cattle



https://fwallpapers.com/view/cat-jeans
http://www.clipartpanda.com/clipart_images/free-clip-art-1083418

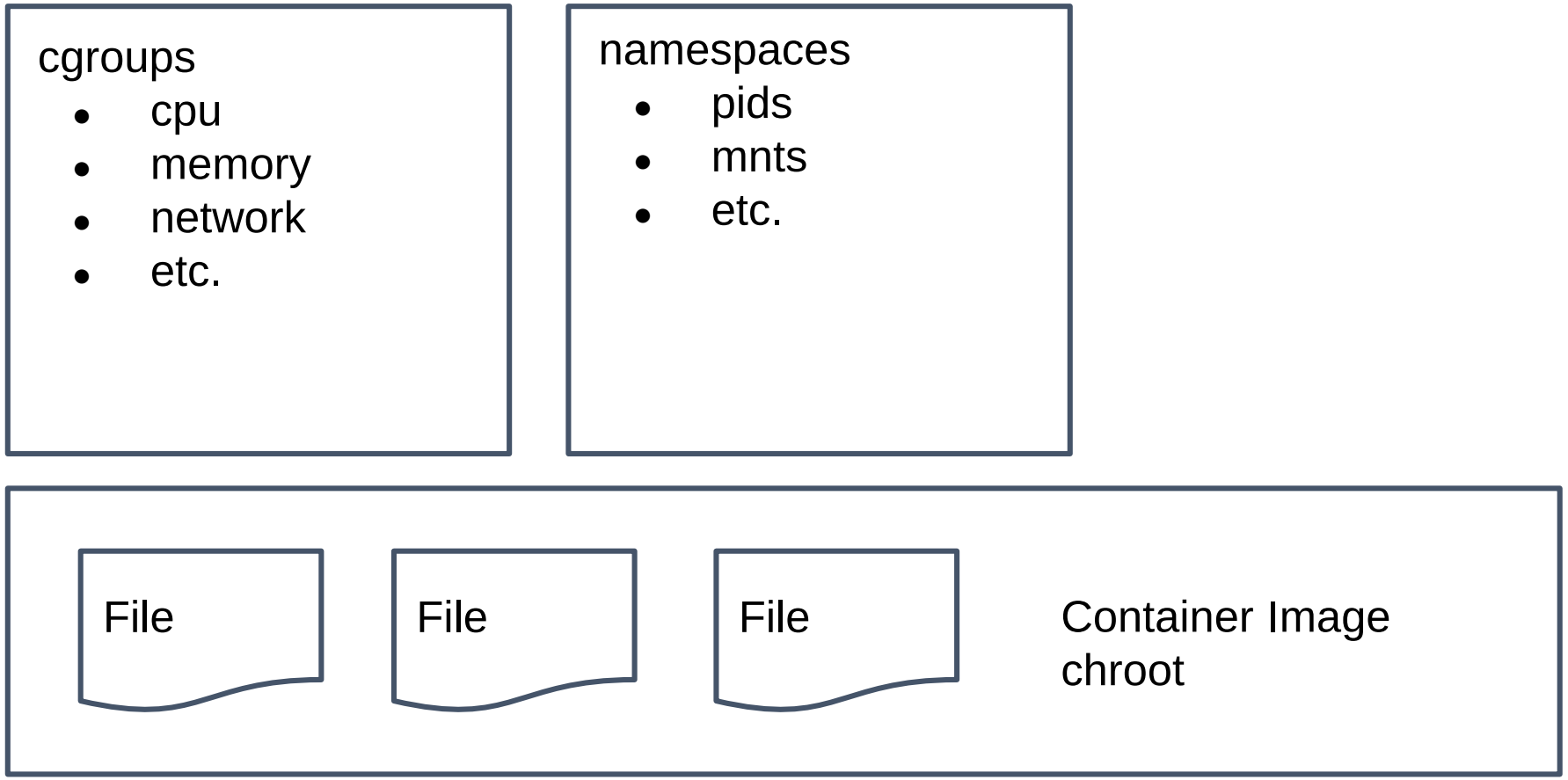
pets vs cattle



- long lived
- name them
- care for them

- ephemeral
- brand them with #’s
- well..vets are expensive

container = image + isolation



containers are good

containers are ~~good~~ excellent

containers have a problem

you can never get away from pets unless:
- you handle the problem of container state
- you need an environment to support cattle

MapR and kubernetes are the solution

Things docker can’t (or won’t) do...

solve port mapping hell

monitor running containers

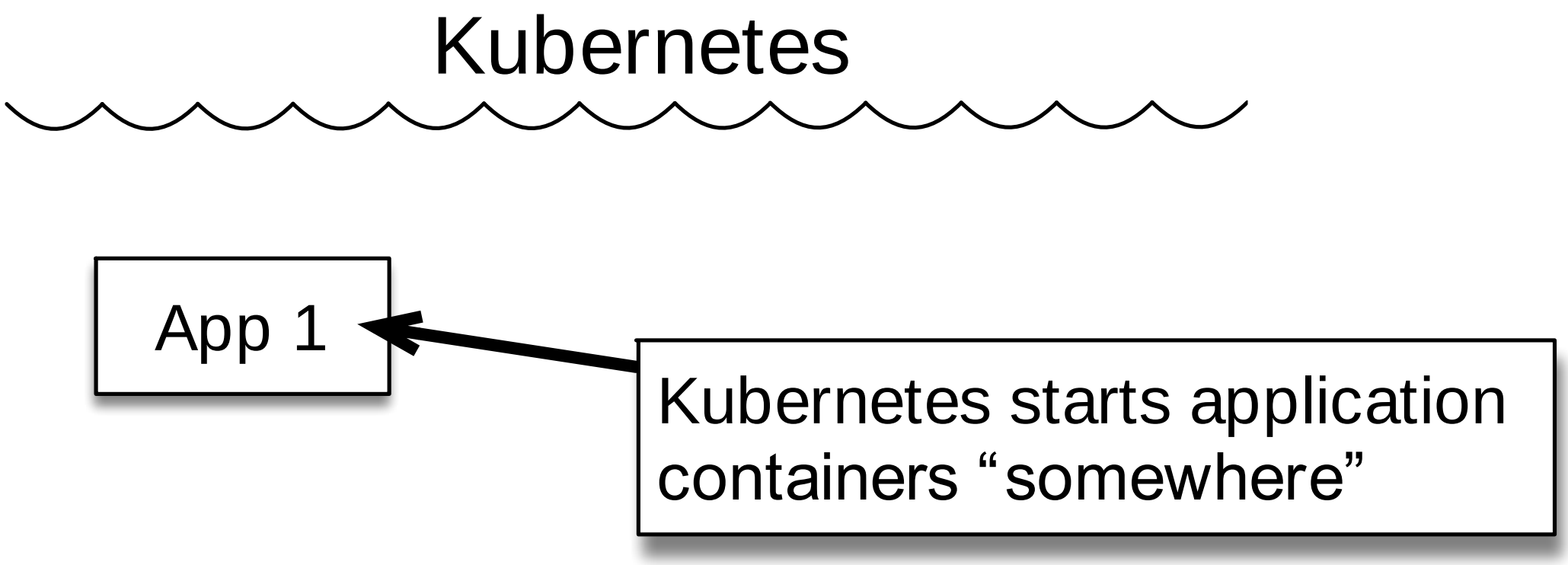
handle dead containers

move containers so utilization improves

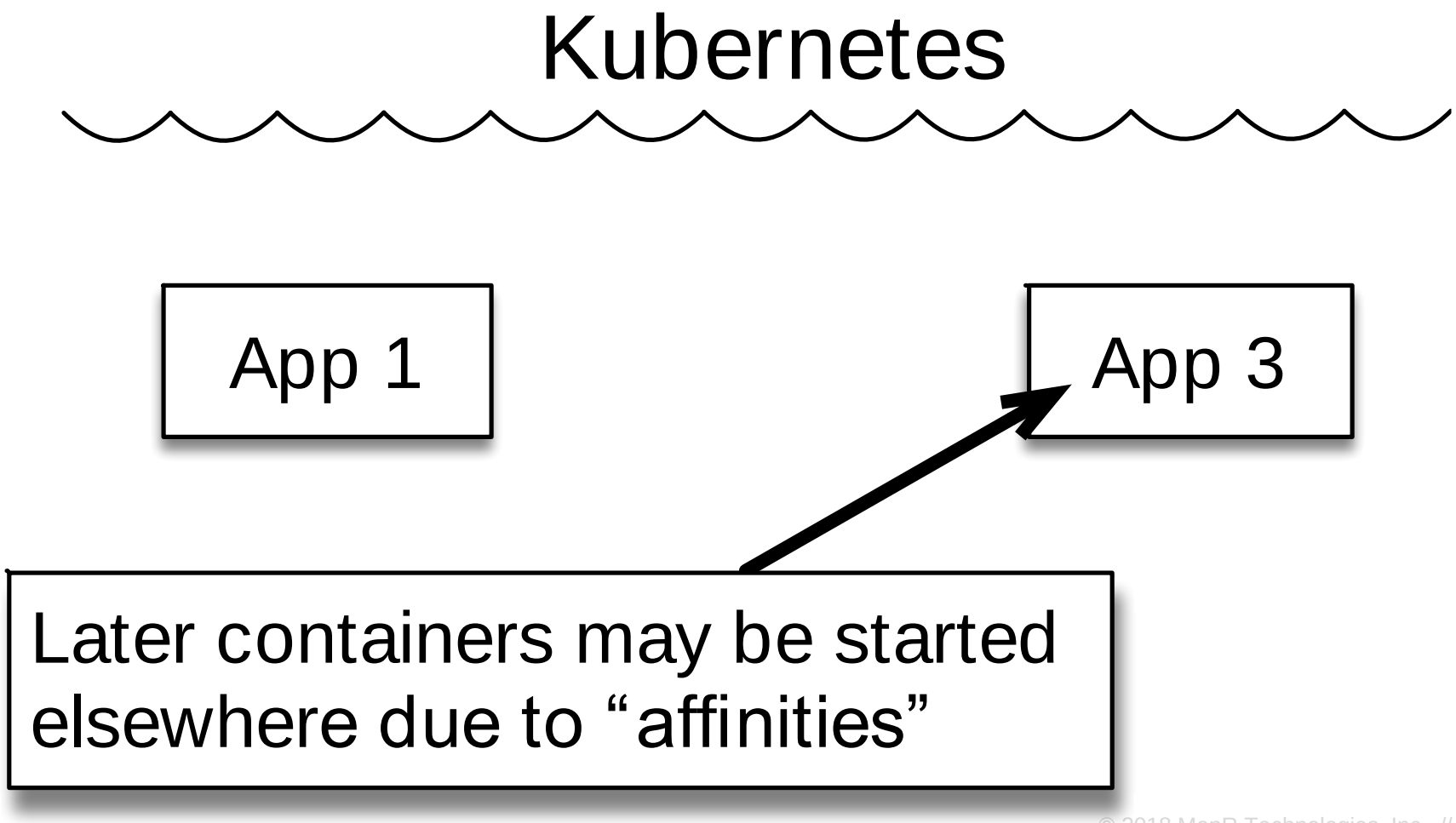
autoscale container instances to handle load

Magical View of Kubernetes

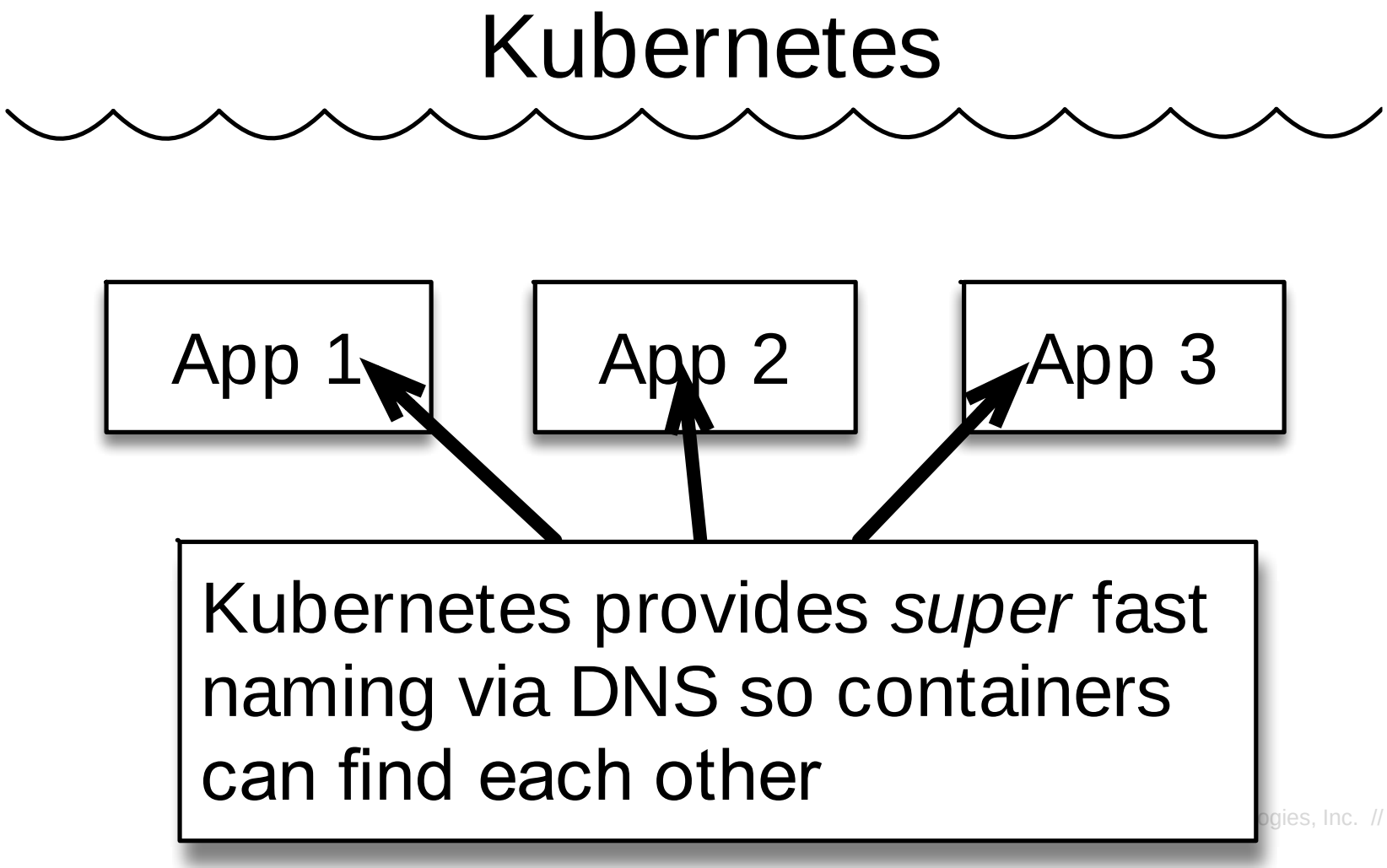
Magical View of Kubernetes



Magical View of Kubernetes



Magical View of Kubernetes



Note that you don’t think about which machine at all

You don’t think about which machine at all

No more names from The Hobbit

Just cattle

The Impact of Kubernetes

Software engineering can be viewed as freezing bits

Initially, everything is possible, nothing is actual

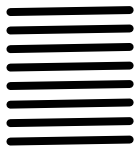
We freeze the source

Then the binary

Then the package

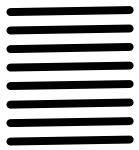
Then the environment

Ultimately the system



git

Build

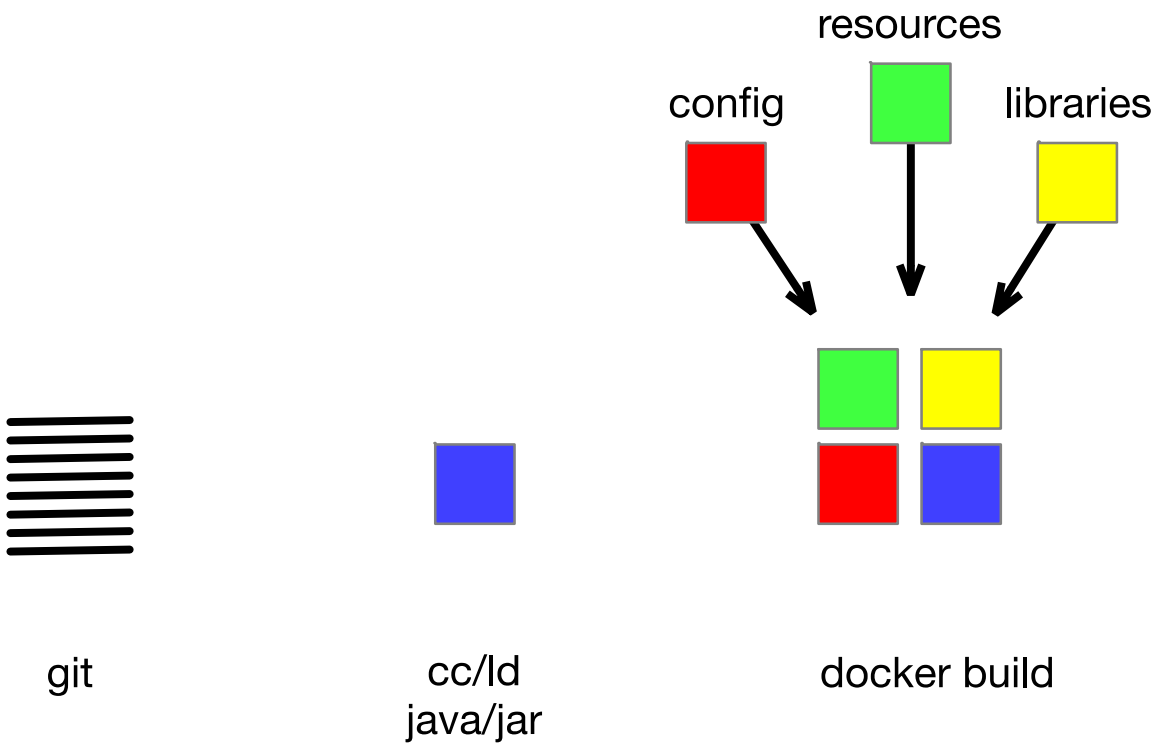


git

cc/ld
java/jar

Build

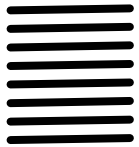
Package



Build

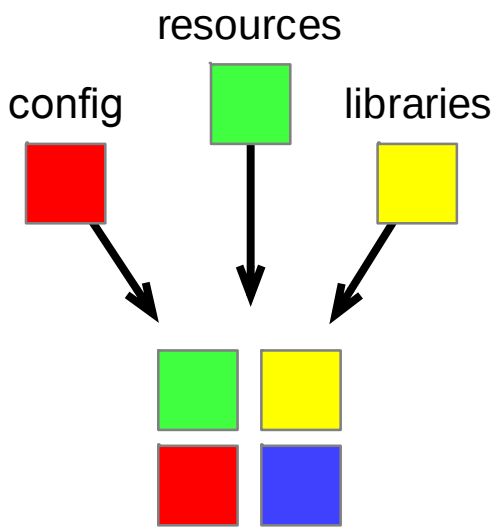
Package

Construct

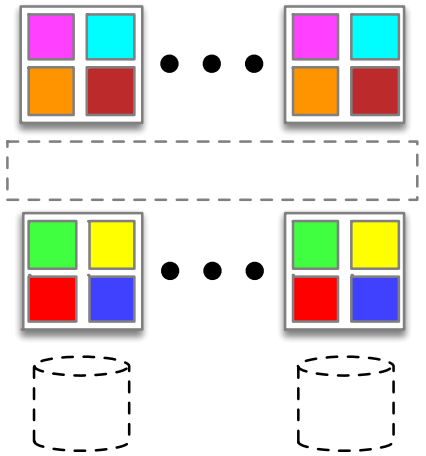


git

cc/ld
java/jar



docker build



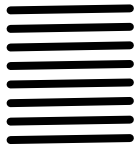
helm package

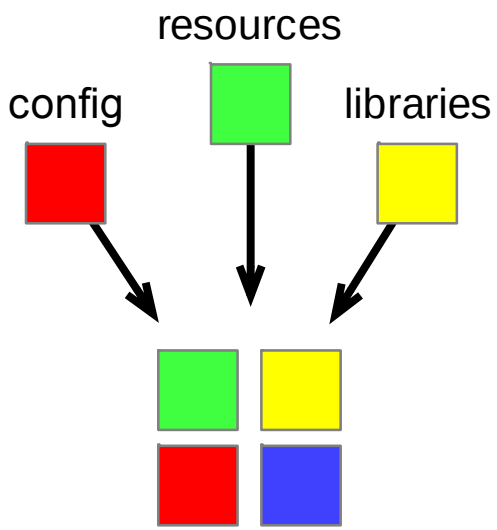
Build

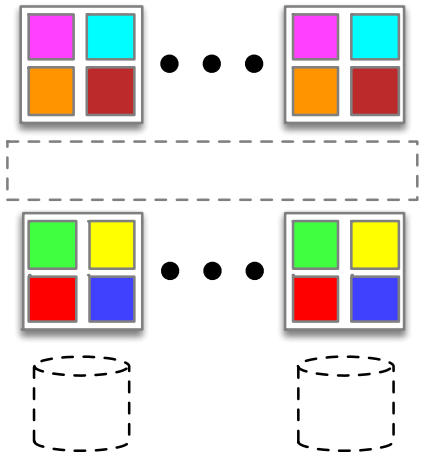
Package

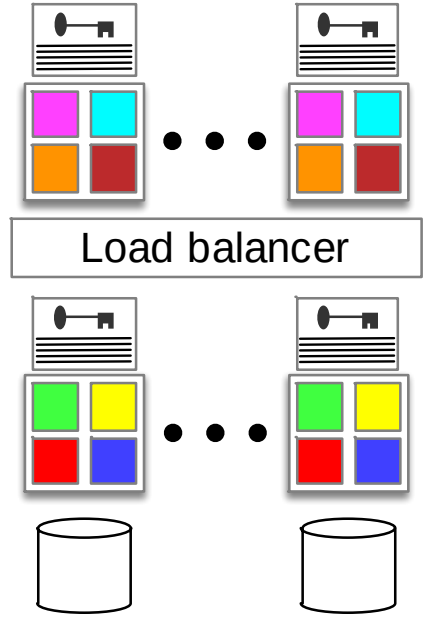
Construct

Deploy









git

cc/ld
java/jar

docker build

helm package

helm install/scale

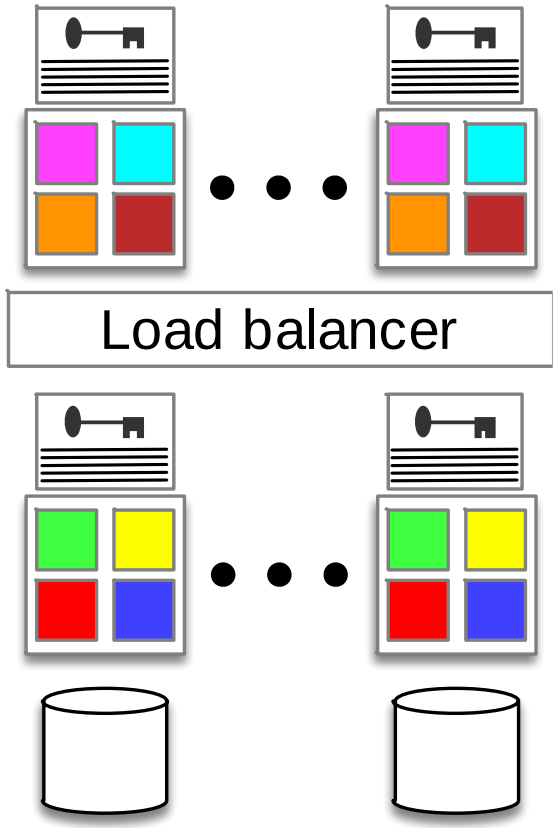
Hat tip to Eric Brewer

This is glorious

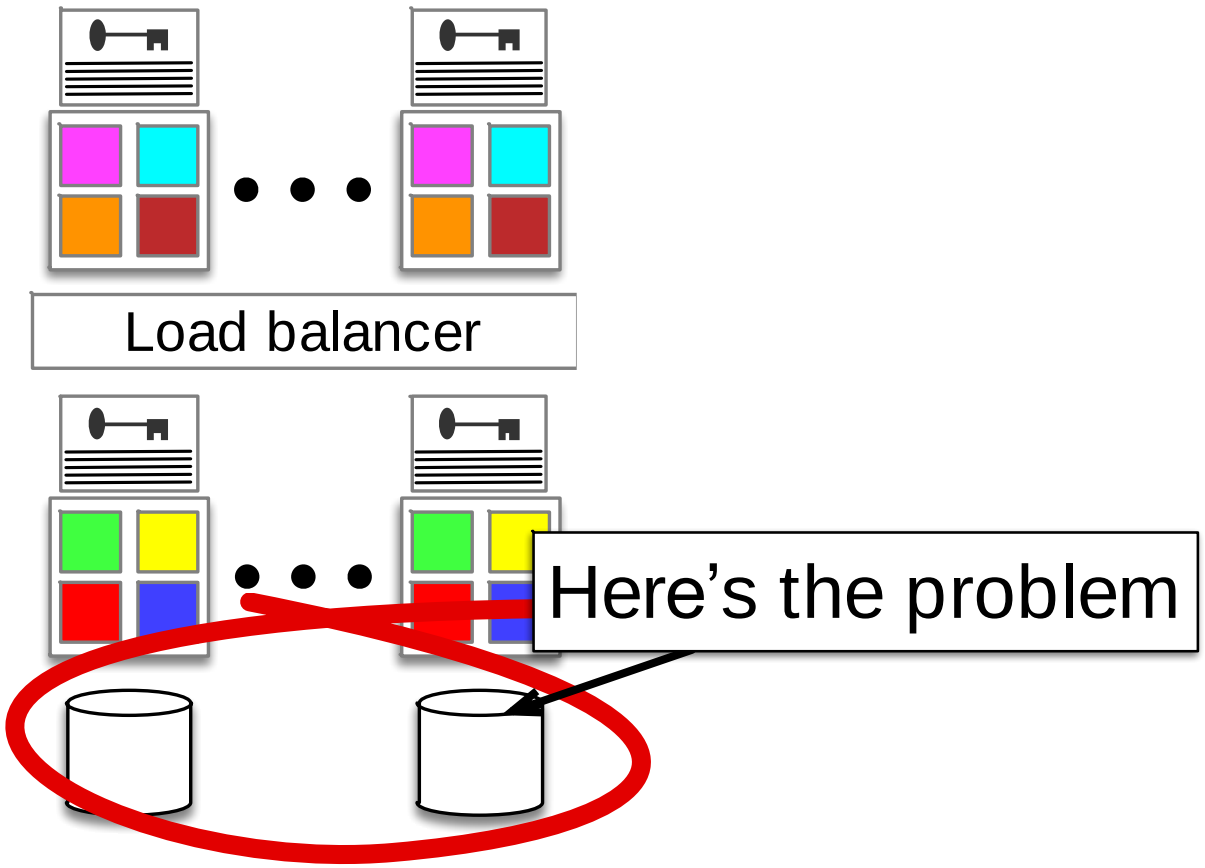
but we still have a problem

state

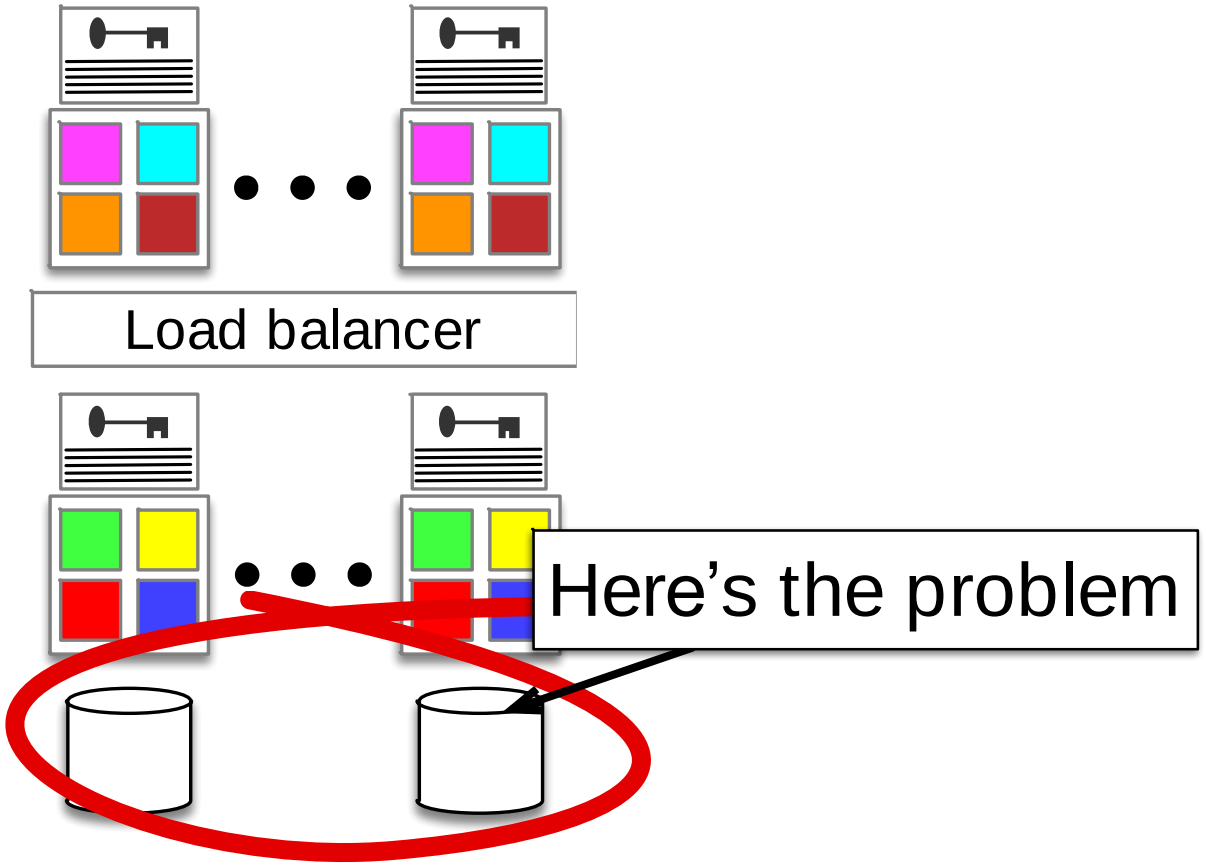
Not Done Yet



Not Done Yet



Not Really Ready at All



State in containers messes things up

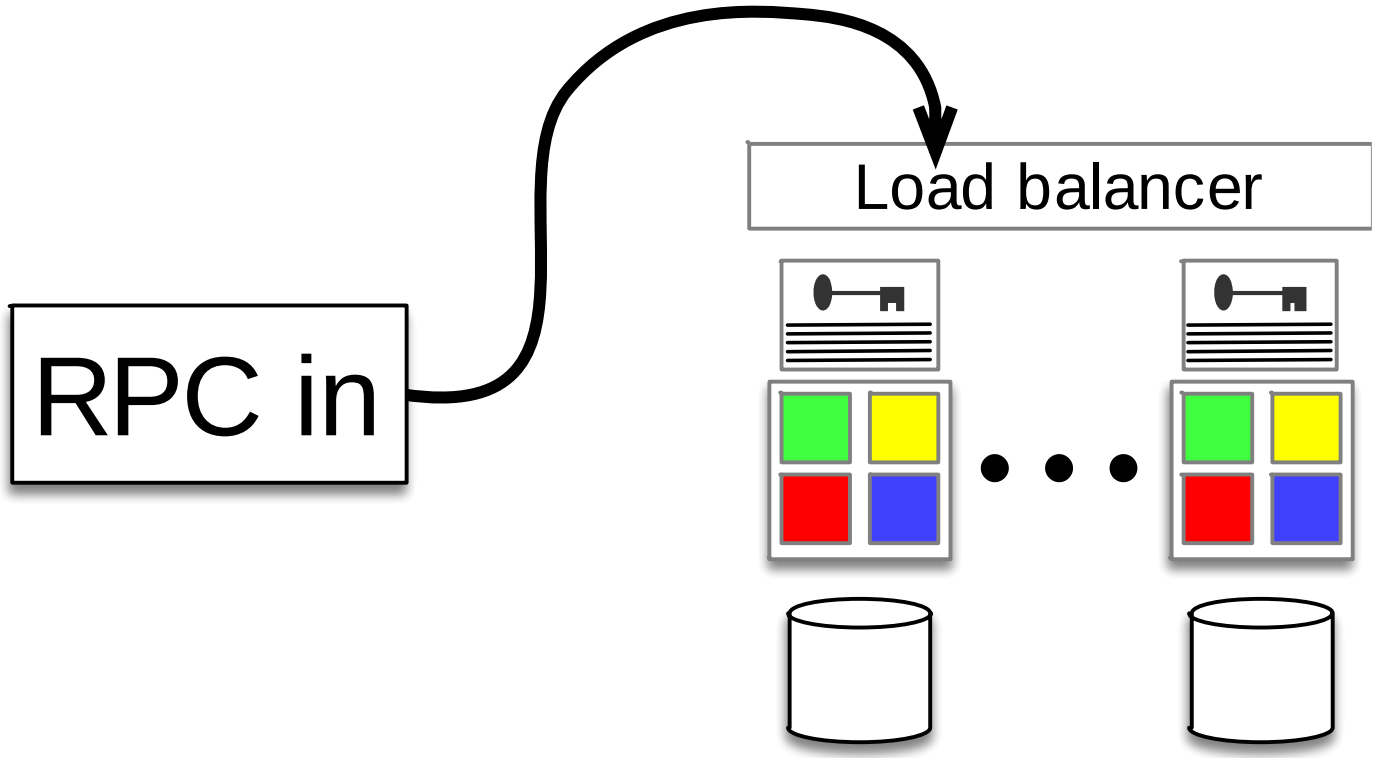
Restarts lose the state

Replicating state makes services complex

Application developers just aren't systems developers

State life-cycle doesn't match app life-cycle

What is a Service Anyway?



But … Not Entirely

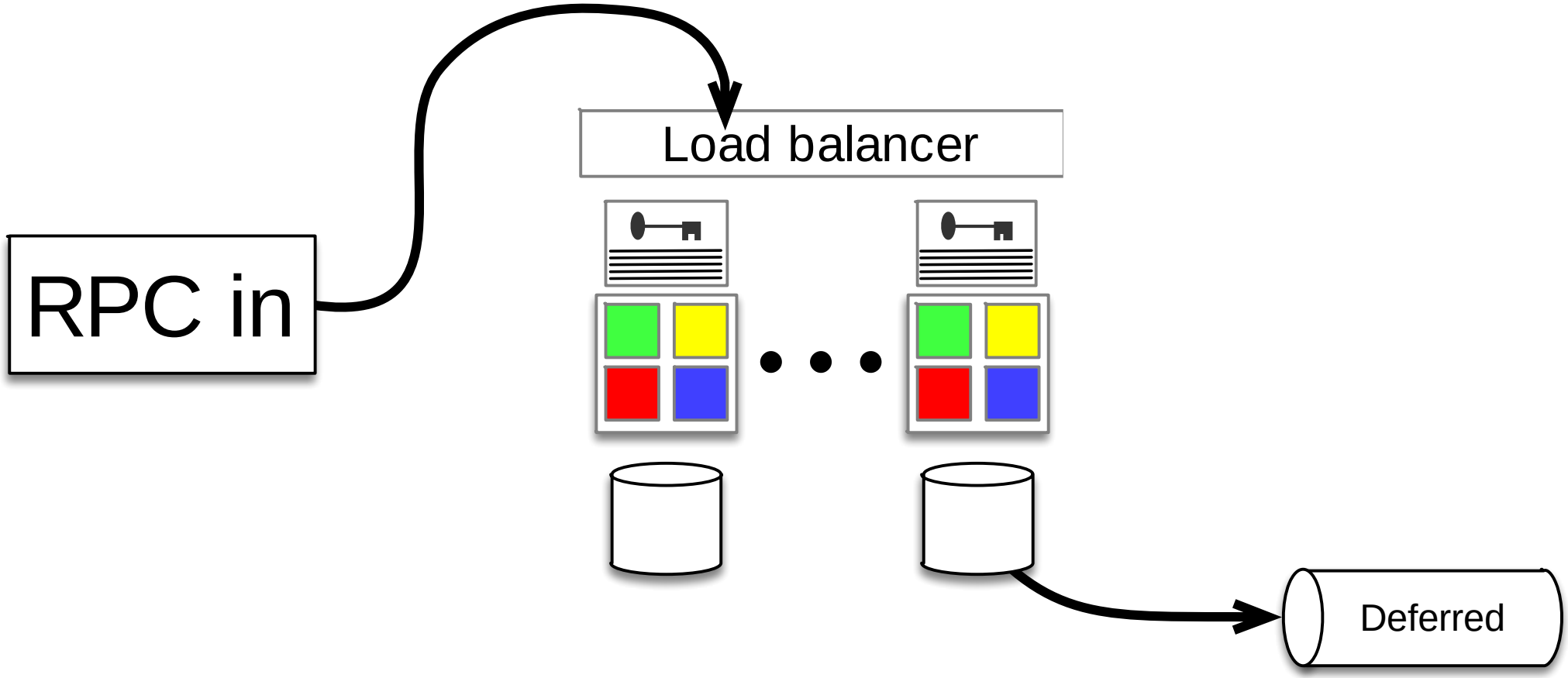
Synchronous RPC-based services only serve one need

In a synchronous service it's common to do some, defer some

But deferring work is hard in a synchronous world … we have to give up the return call in some sense

This is the germ of streaming architecture

What is a Service Anyway?



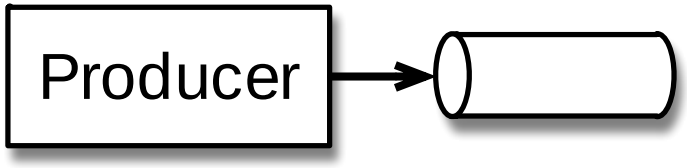
Isolation is The Defining Characteristic

If I can hide details of who and where, I have a service

If I can hide details of deployment, I have a micro-service

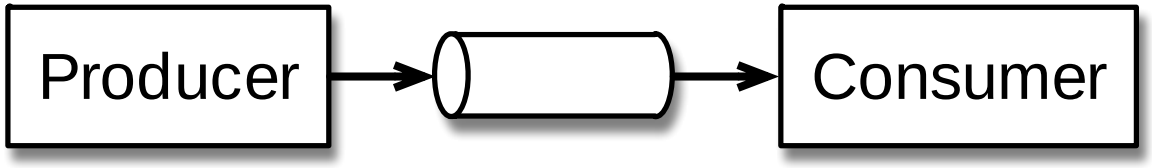
If I can hide details of ***when***, I have a streaming micro-service

Temporal and Geo Isolation

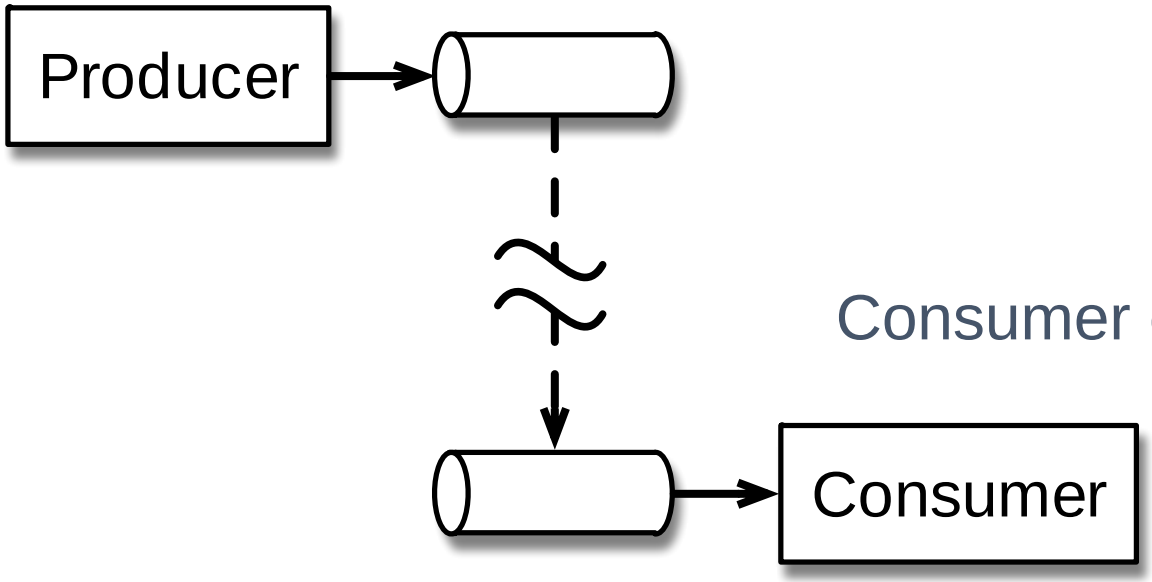


Consumer isn’t even running

Temporal and Geo Isolation



Temporal and Geo Isolation



Consumer could be an ocean away

We Need Multiple Forms of Persistence

Files are important

- Config files, image files, archival data data
- Legacy applications like machine learning, web

Tables are important

- Critical to have random update for some applications
- Should scale transparently without dedicated cluster

Streams are important

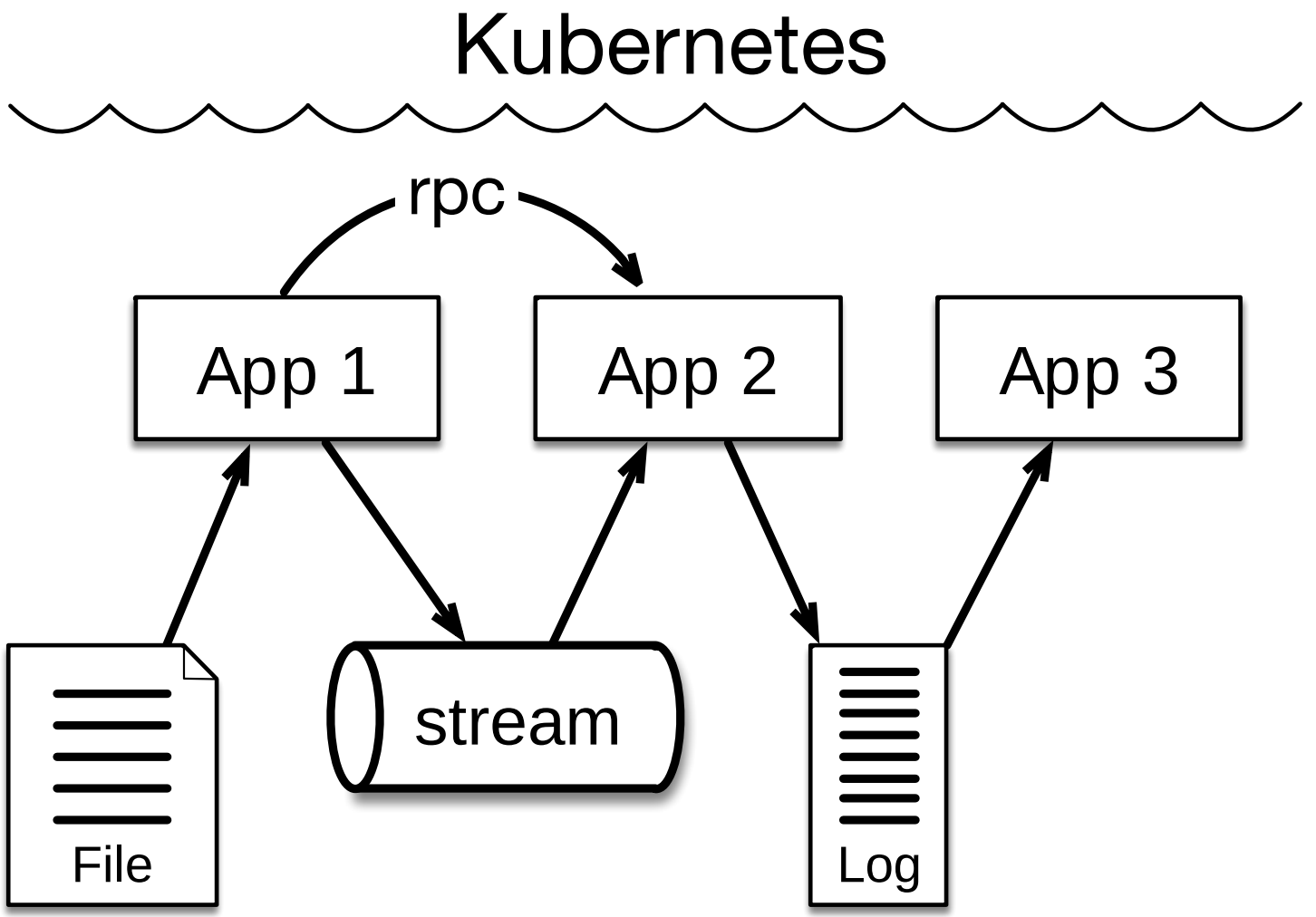
- Should be co-equal form of persistence

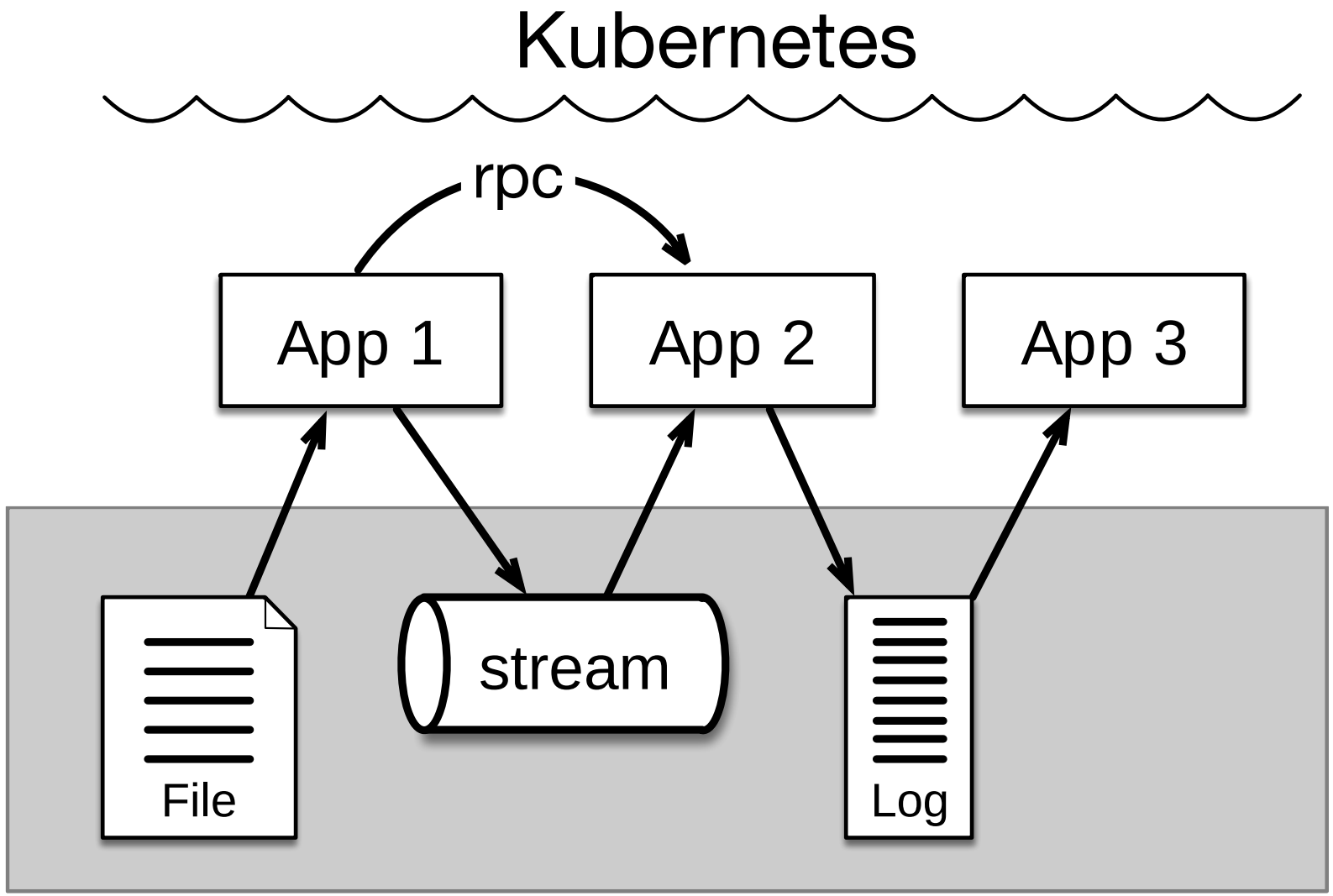
Kubernetes

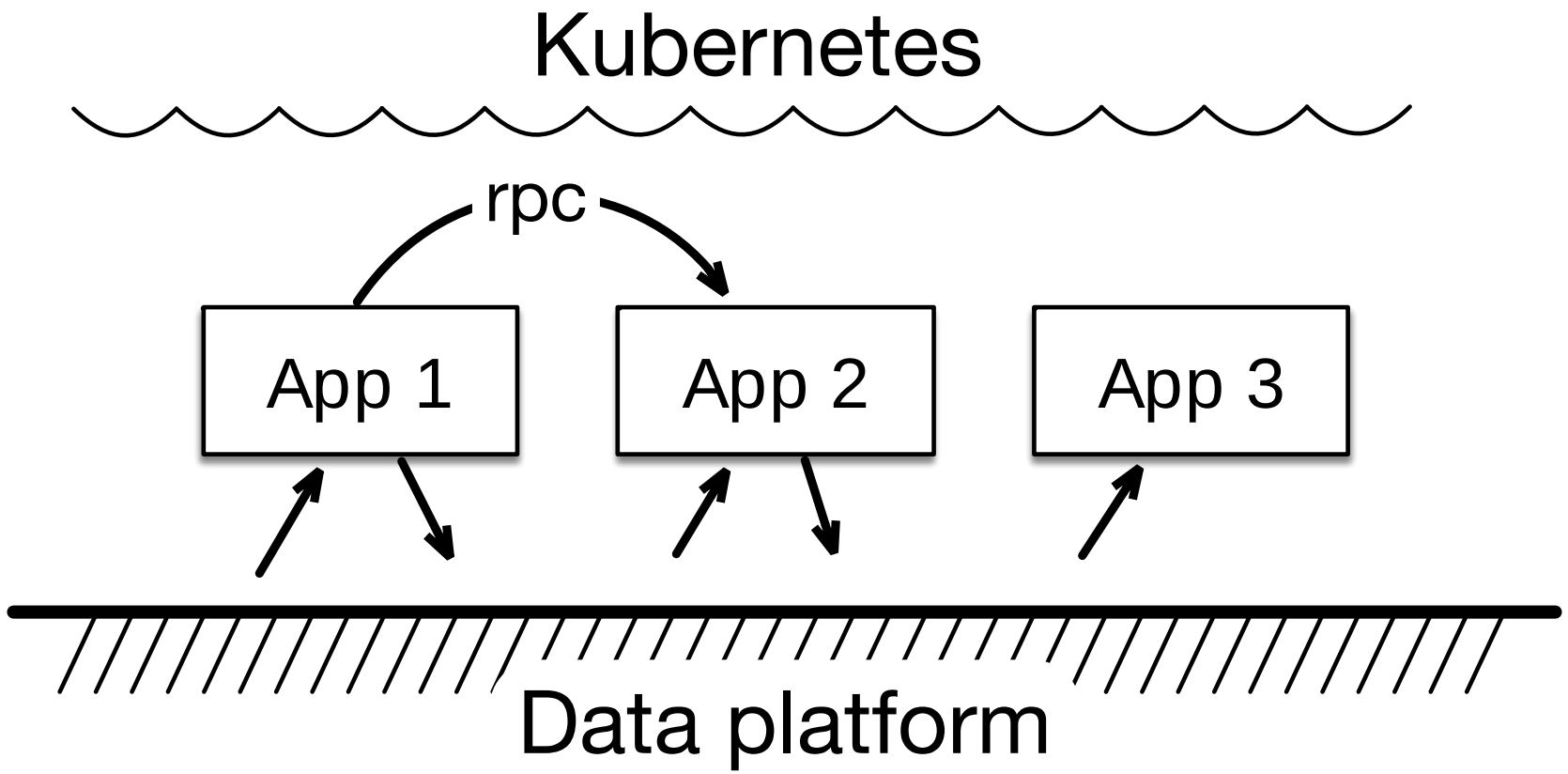
App 1

App 2

App 3







What Does This Data Platform Need to Have?

Global namespace across entire Kubernetes cluster

- Between clusters as well if possible

All three forms of primitive persistence

- Files, streams, tables

Inherently scalable

- Performance, cardinality, locality

Uniform access and control

- Path names for all objects, identical permission scheme

What Does This Data Platform Need to Have?

Global namespace across entire Kubernetes cluster

- Between clusters as well if possible

All three forms of primitive persistence

- Files, streams, tables

Inherently scalable

- Performance, cardinality, locality

Uniform access and control

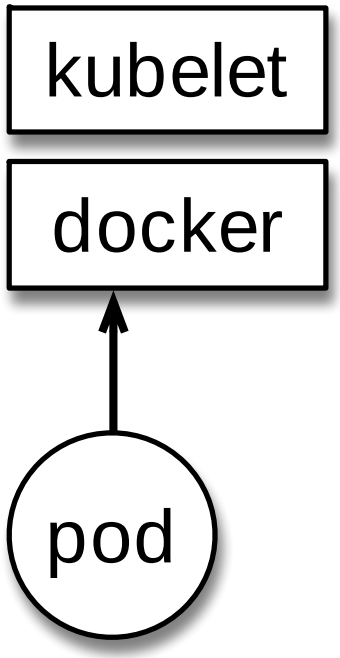
- Path names for all objects, identical permission scheme

Oh…. got that already. Just need to wire it up to Kubernetes

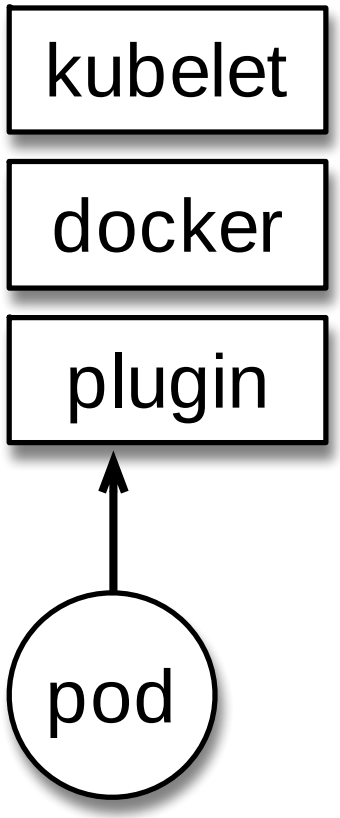
```yaml
# Copyright (c) 2009 & onwards. MapR Tech, Inc., All rights reserved
apiVersion: v1
kind: Pod
metadata:
  name: test-secure
  namespace: mapr-examples
spec:
  securityContext:
    runAsUser: 1000
    fsGroup: 2000
  containers:
  - name: busybox
    image: busybox
    args:
    - sleep
    - "1000000"
    imagePullPolicy: Always
    resources:
      requests:
        memory: "2Gi"
        cpu: "500m"
    volumeMounts:
    - mountPath: /mapr
      name: maprflex
  volumes:
    - name: maprflex
      flexVolume:
        driver: "mapr.com/maprfs"
        options:
          volumePath: "/"
          cluster: "mysecurecluster"
          cldbHosts: "cldb1 cldb2 cldb3"
          securityType: "secure"
          ticketSecretName: "mapr-ticket-secret"
          ticketSecretNamespace: "mapr-examples"

```

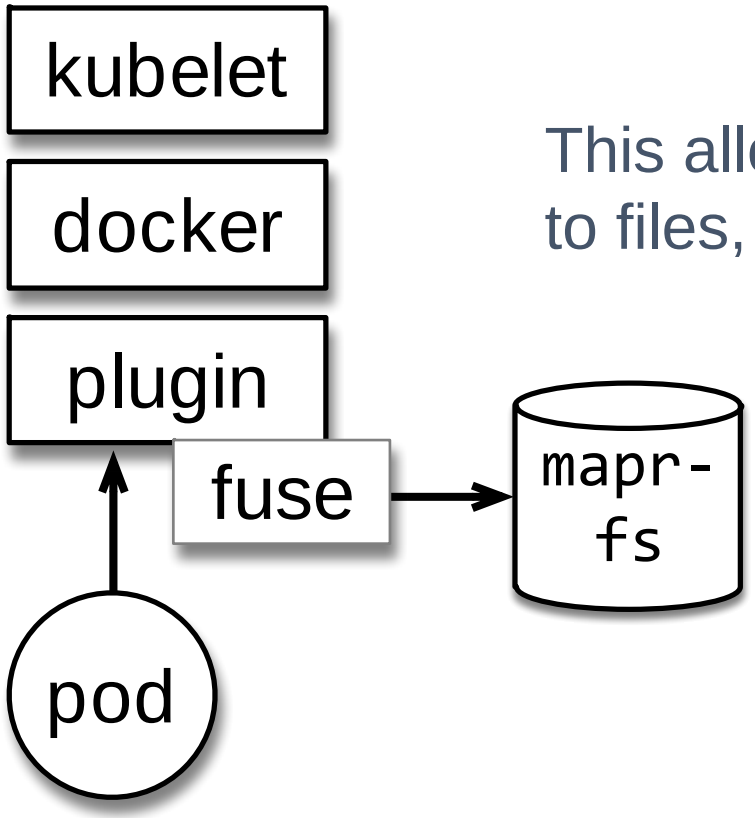
```
        cpu: "500m"
    volumeMounts:
    - mountPath: /mapr
      name: maprflex
volumes:
  - name: maprflex
    flexVolume:
      driver: "mapr.com/maprfs"
      options:
        volumePath: "/"
        cluster: "mysecurecluster"
        cldbHosts: "cldb1 cldb2 cldb3"
        securityType: "secure"
        ticketSecretName: "mapr-ticket-secret"
        ticketSecretNamespace: "mapr-examples"
```



Normally pods interact directly with node resources



We can install a volume plugin (recently introduced)



This allows uniform access
to files, tables and streams

Where does that take us?

Consequences

Installation of plugin is K8S level operation

- No per-node attention required

Use of plugin is overlay operation

- No change needed for an container
- Any Helm chart can use the plugin for conventional file access

Can share storage/compute or isolate or scale independently

More Consequences

State is no longer a dirty word for Kubernetes

HPC can run on K8S

Boring things can run on K8S without storage appliances

Previously crazy ideas can now be valuable

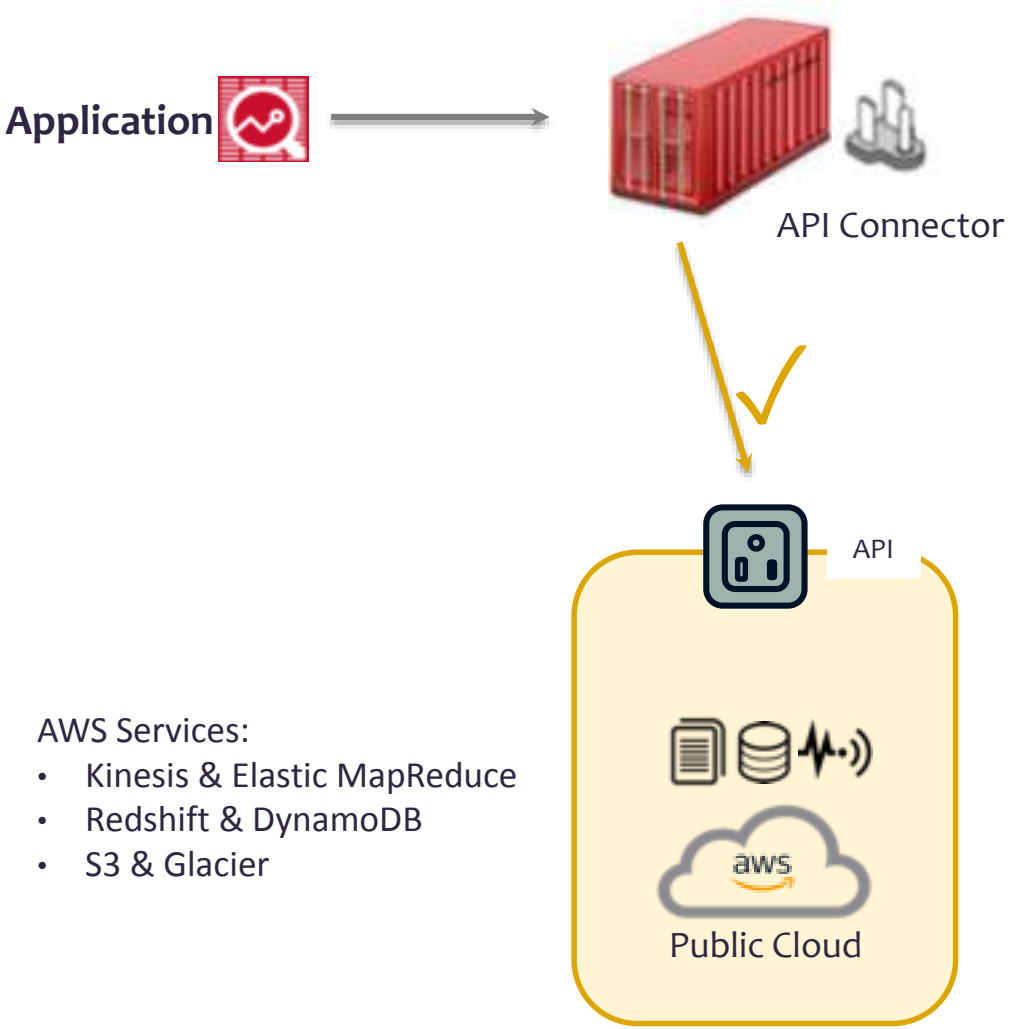
Complexity is largely not visible

Container orchestration is awesome

Container orchestration is awesome

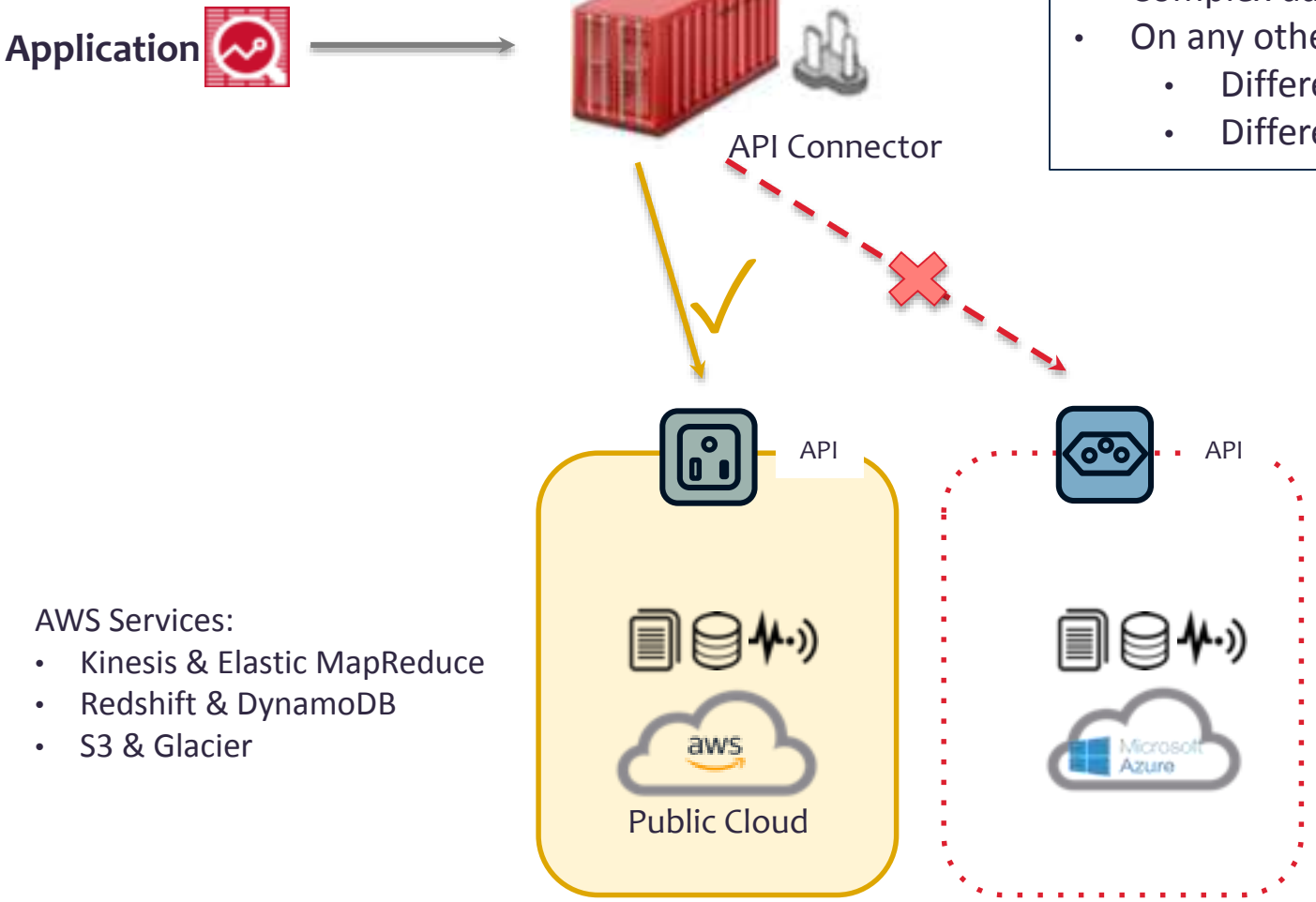
Data orchestration is, too

Cloud as-is: No unified data access or security concepts



Single cloud vendor strategy:
- Vendor lock in
- No failover in case of global outage
- Limited Edge capabilities

Cloud as-is: No unified data access or security concepts



Multi cloud strategy:

- Complex data movement between clouds
- On any other cloud:
  - Different API's: application breaks
  - Different Security concept

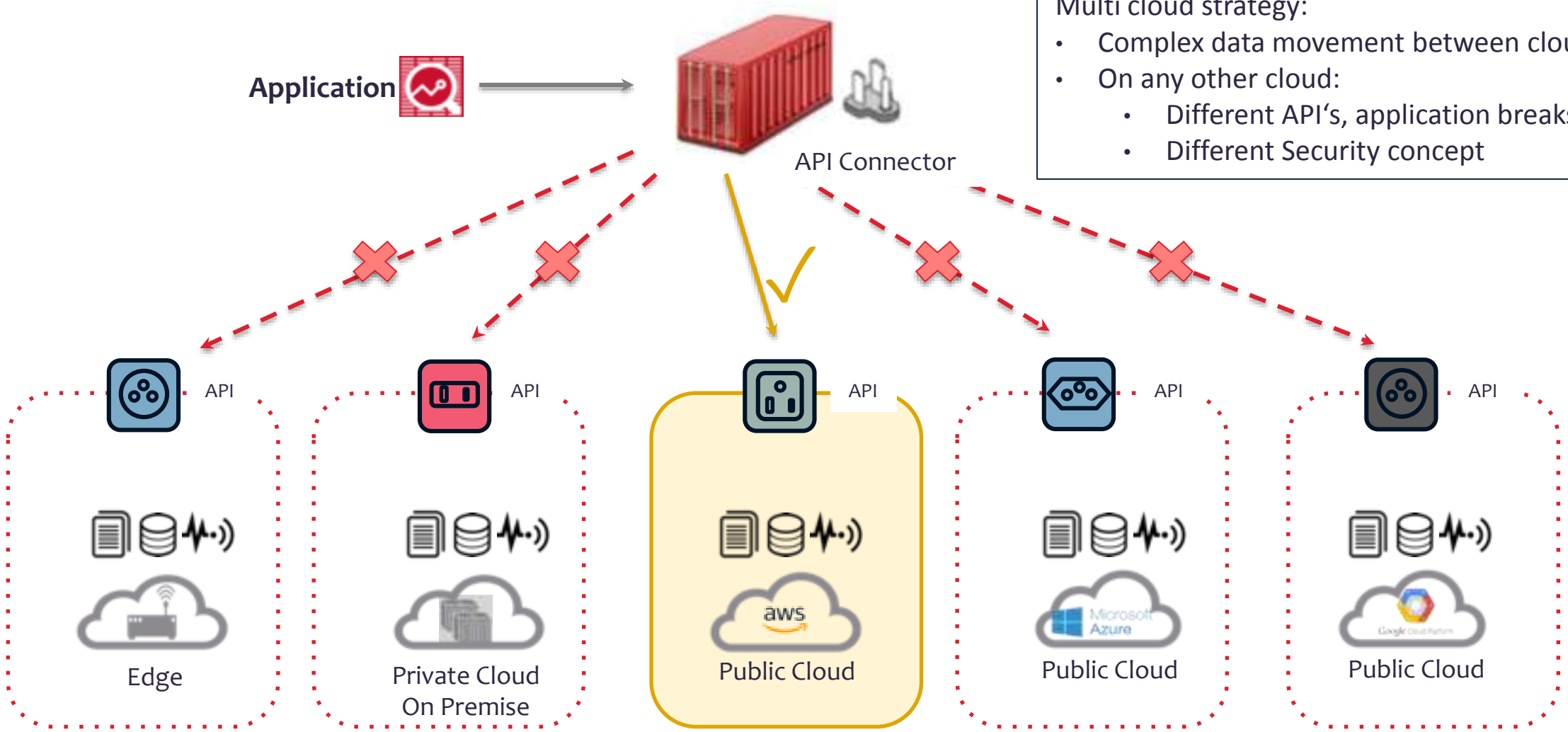
AWS Services:

- Kinesis & Elastic MapReduce
- Redshift & DynamoDB
- S3 & Glacier

Azure Services:

- HD Insight
- SQL Server & CosmosDB
- Blob & DataLakeStore

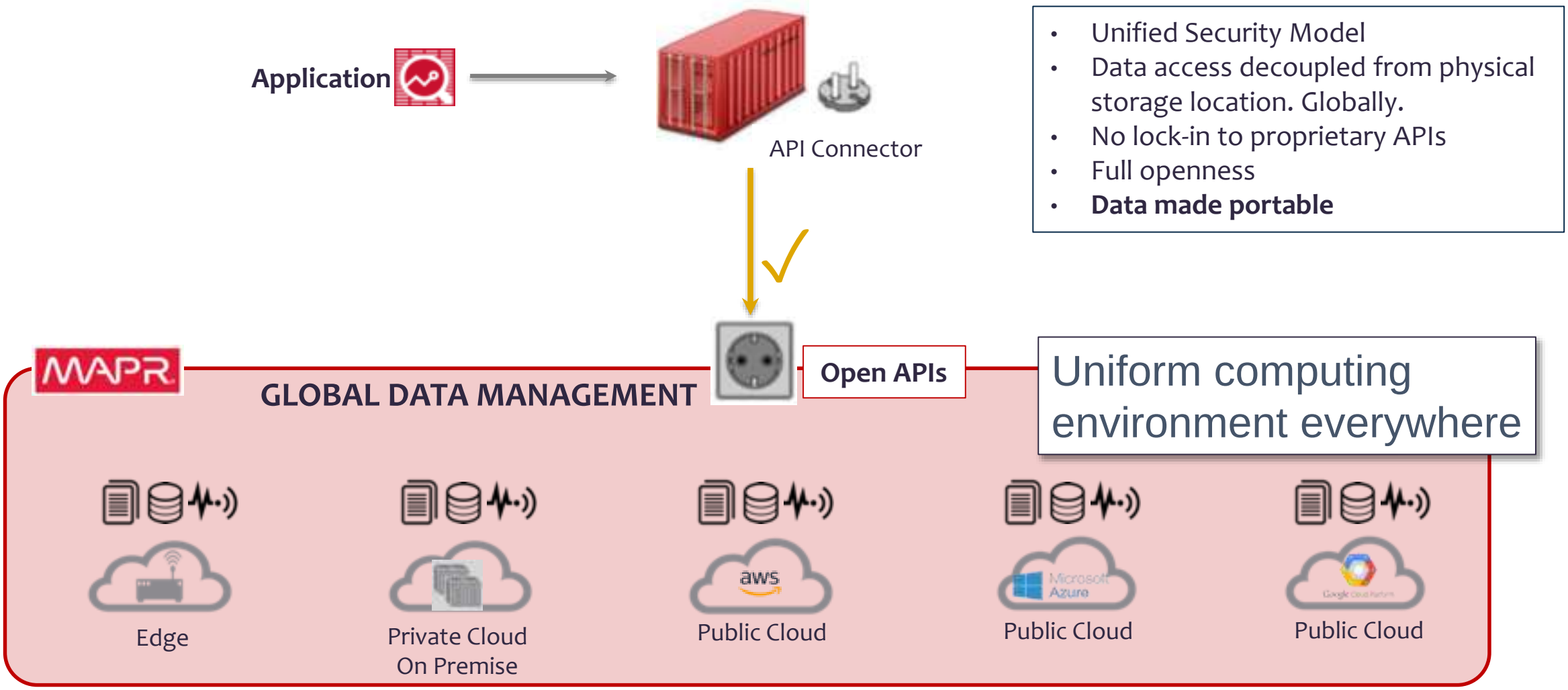
Cloud as-is: No unified data access or security concepts



Multi cloud strategy:

- Complex data movement between clouds
- On any other cloud:
  - Different API's, application breaks
  - Different Security concept

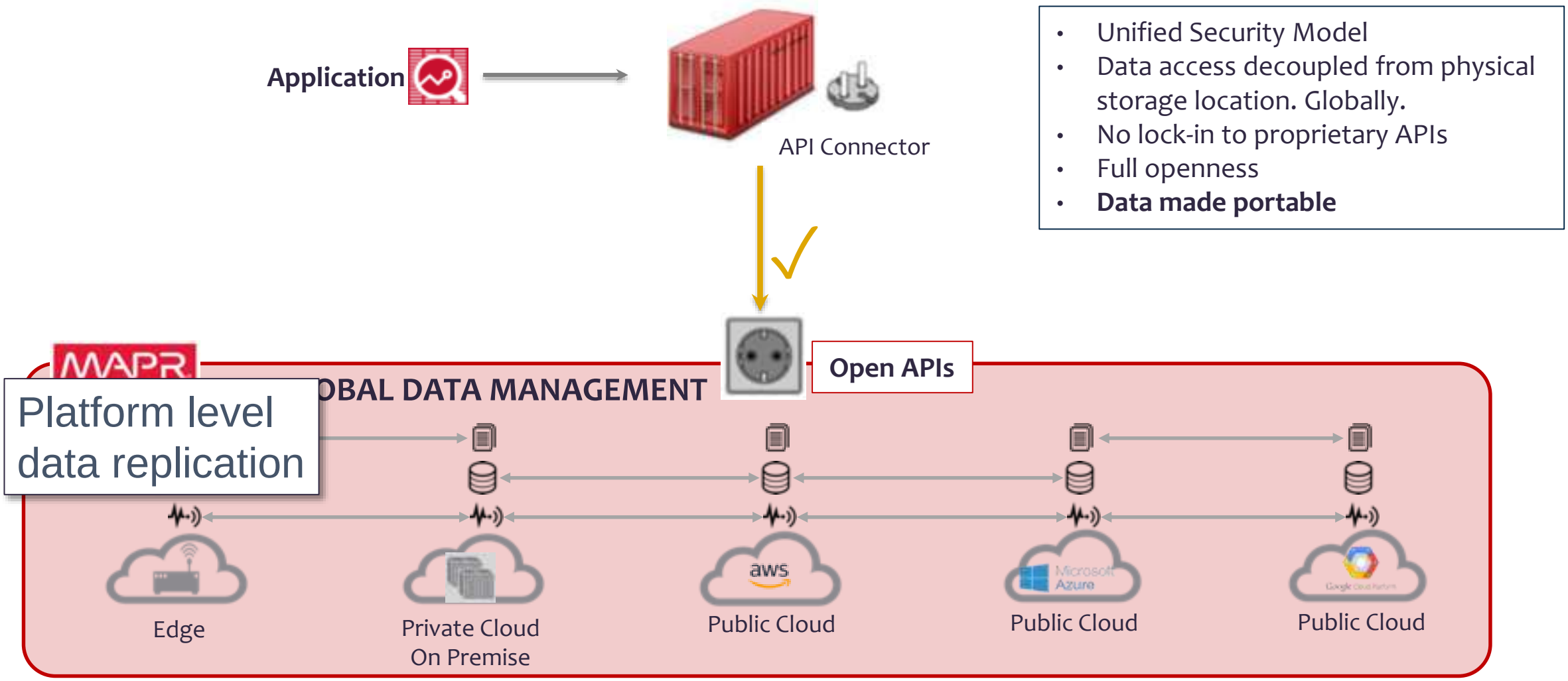
How a “Media Company” is Unifying Compute Environments



- Unified Security Model
- Data access decoupled from physical storage location. Globally.
- No lock-in to proprietary APIs
- Full openness
- **Data made portable**

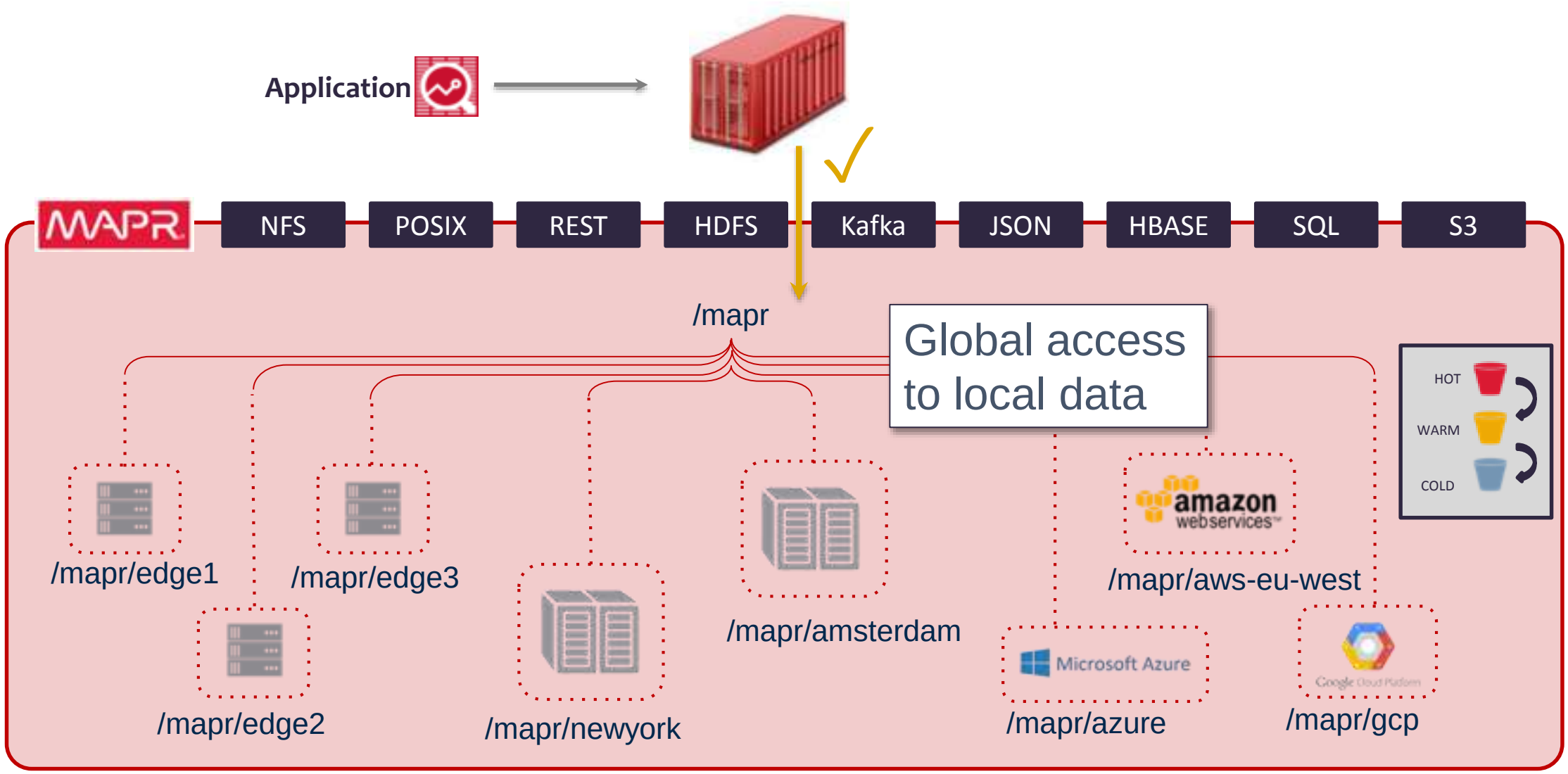
Uniform computing environment everywhere

How “Manufacturing Company” is Orchestrating Data

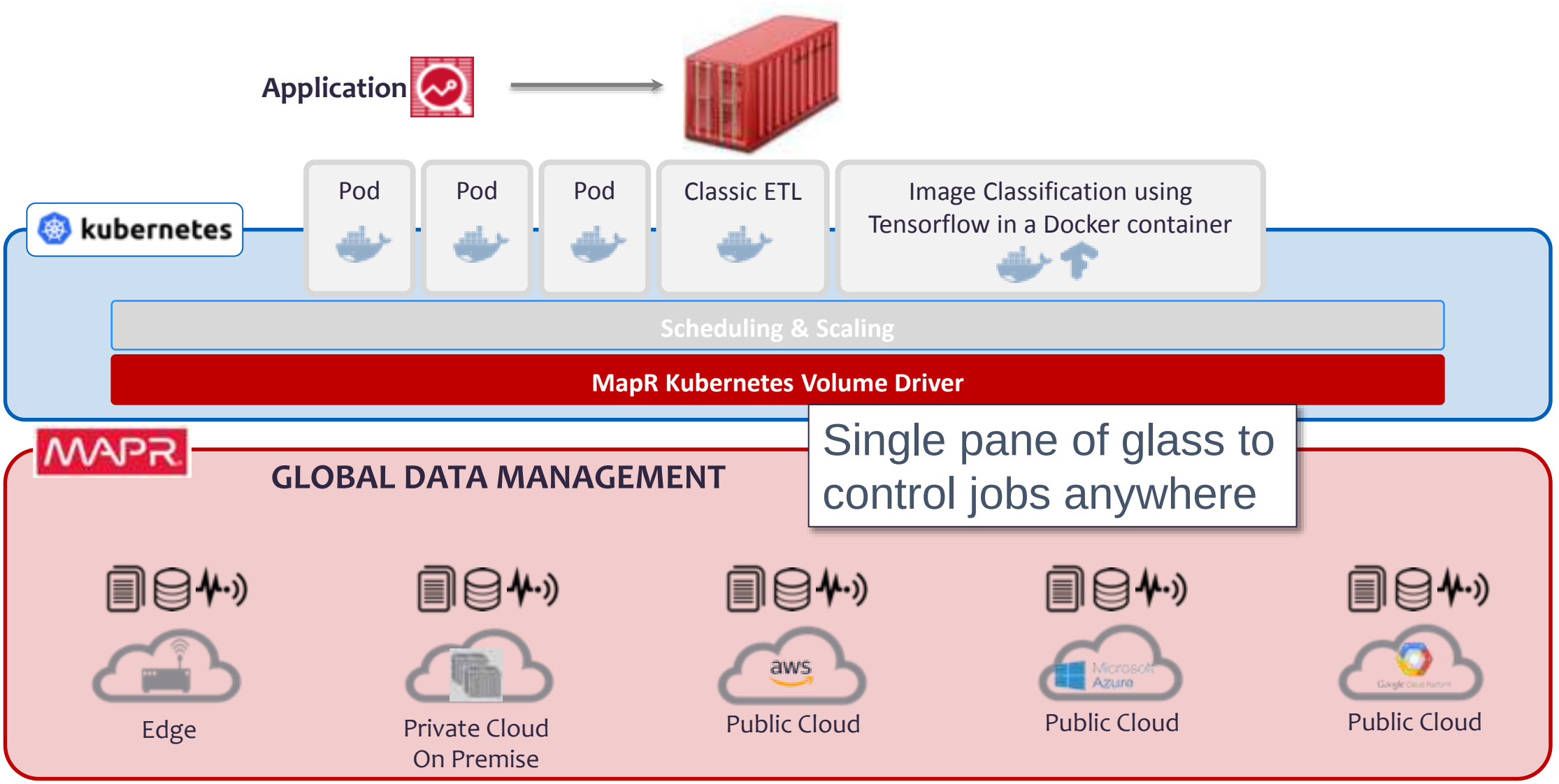


- Unified Security Model
- Data access decoupled from physical storage location. Globally.
- No lock-in to proprietary APIs
- Full openness
- **Data made portable**

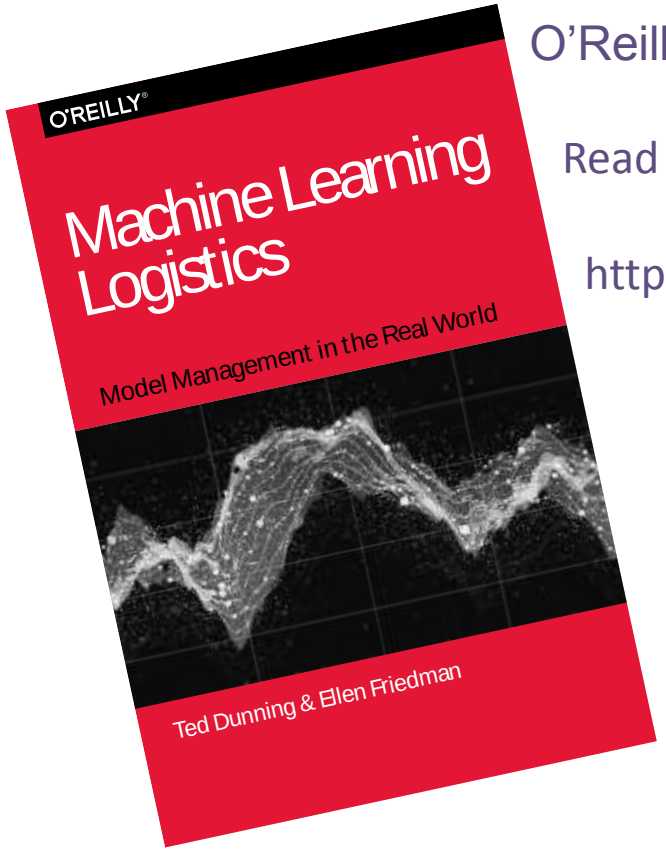
Tier 1 Bank #1 Creating a Global Filesystem



Tier 1 Bank #2: Creating a Universal Application Platform



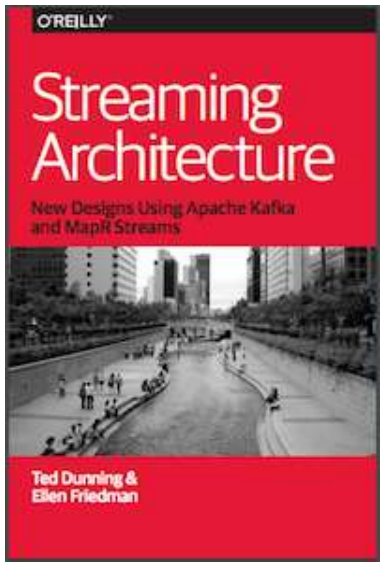
Additional Resources



O'Reilly report by Ted Dunning & Ellen Friedman © September 2017

Read free courtesy of MapR:

https://mapr.com/ebook/machine-learning-logistics/

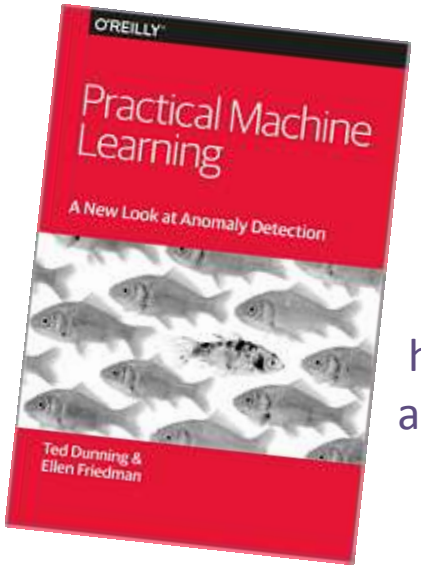


O'Reilly book by Ted Dunning & Ellen Friedman © March 2016

Read free courtesy of MapR:

https://mapr.com/streaming-architecture-using-apache-kafka-mapr-streams/

Additional Resources



O'Reilly book by Ted Dunning & Ellen Friedman
© June 2014

Read free courtesy of MapR:

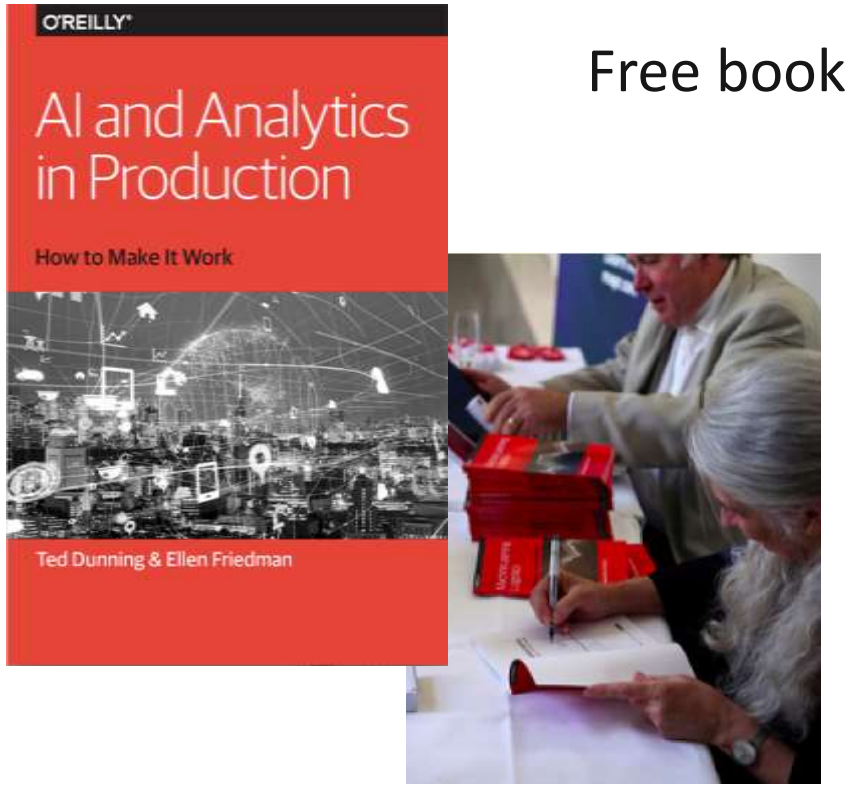
https://mapr.com/practical-machine-learning-new-look-anomaly-detection/

O'Reilly book by Ellen Friedman & Ted Dunning
© February 2014

Read free courtesy of MapR:

https://mapr.com/practical-machine-learning/

AI & Analytics in Production: How to Make It Work



Free book signing with authors Ted Dunning & Ellen Friedman

MapR stand #145:

Tues 1:00 pm – 1:45 pm
Wed 1:00 pm – 1:45 pm

Download free pdf via MapR:
https://mapr.com/ebook/ai-and-analytics-in-production/