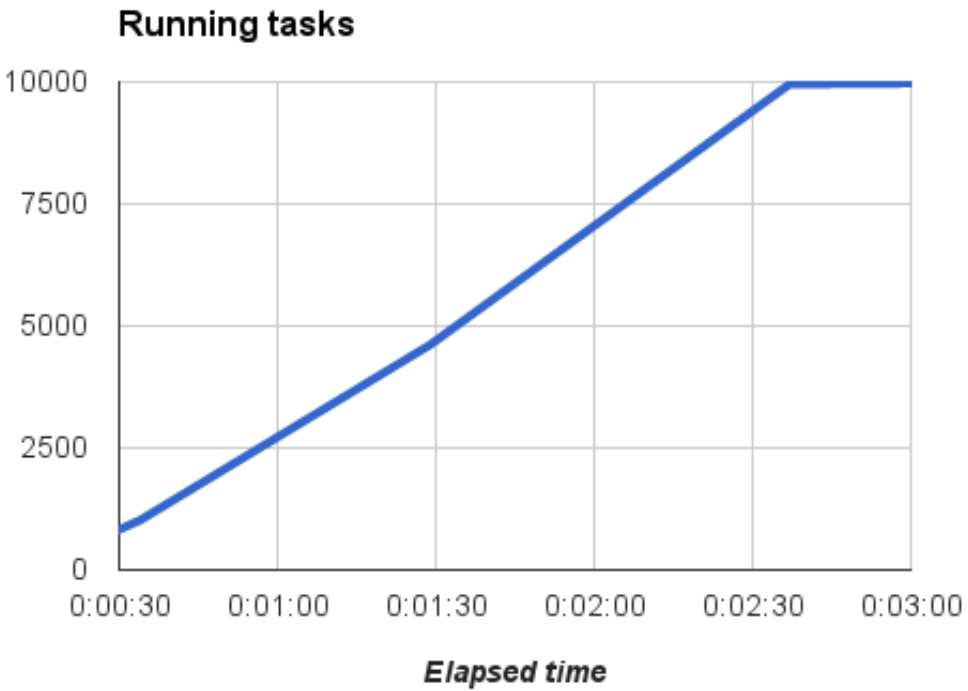
So, I have all these containers! Now what?


Image by Connie
Zhou

Developer View

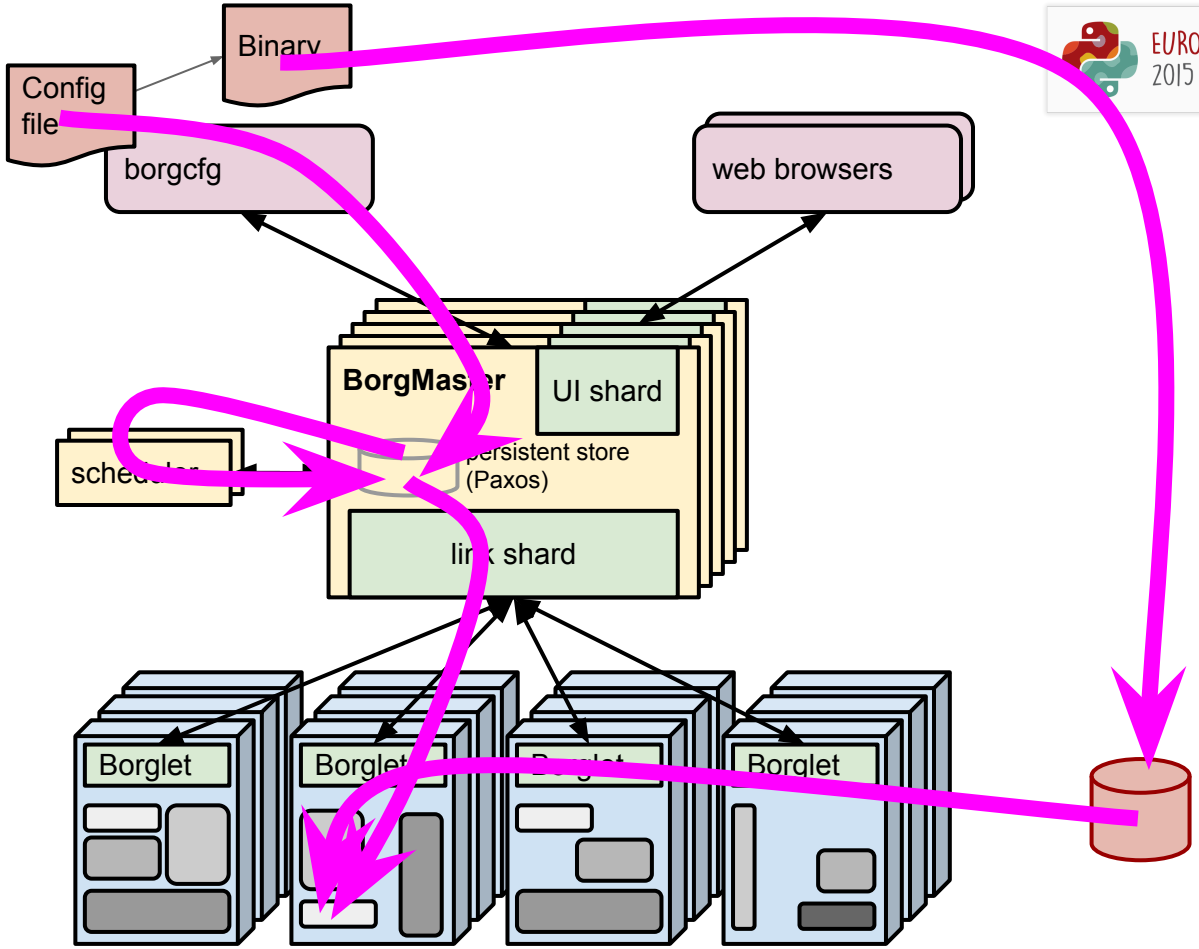
```
job hello_world = {
  runtime = { cell = 'ic' }              // Cell (cluster) to run in
  binary = '.../hello_world_webserver'   // Program to run
  args = { port = '%port%' }             // Command line parameters
  requirements = {     // Resource requirements
    ram = 100M
    disk = 100M
    cpu = 0.1
  }
  replicas = 10000     // Number of tasks
}
```

Developer View



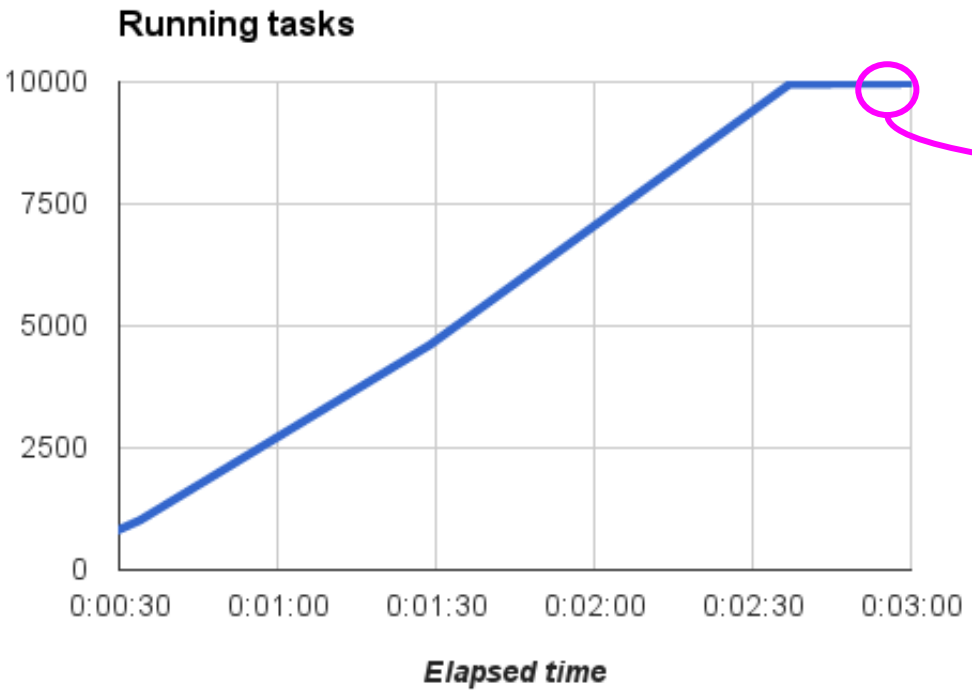
Developer View

What just happened?




Hello world!
Image by Connie Zhou

Developer View

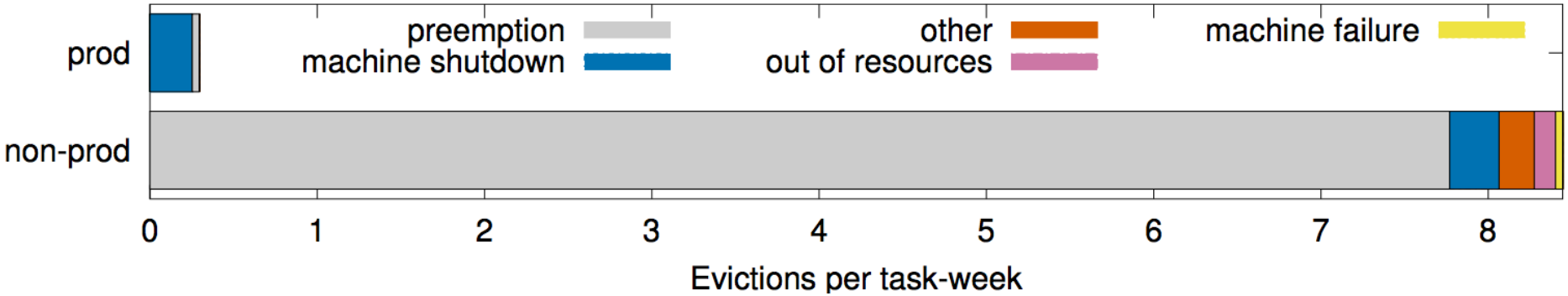


Tasks?

d	w	pending (1)	a	setup (5)	run (9993)

Failures

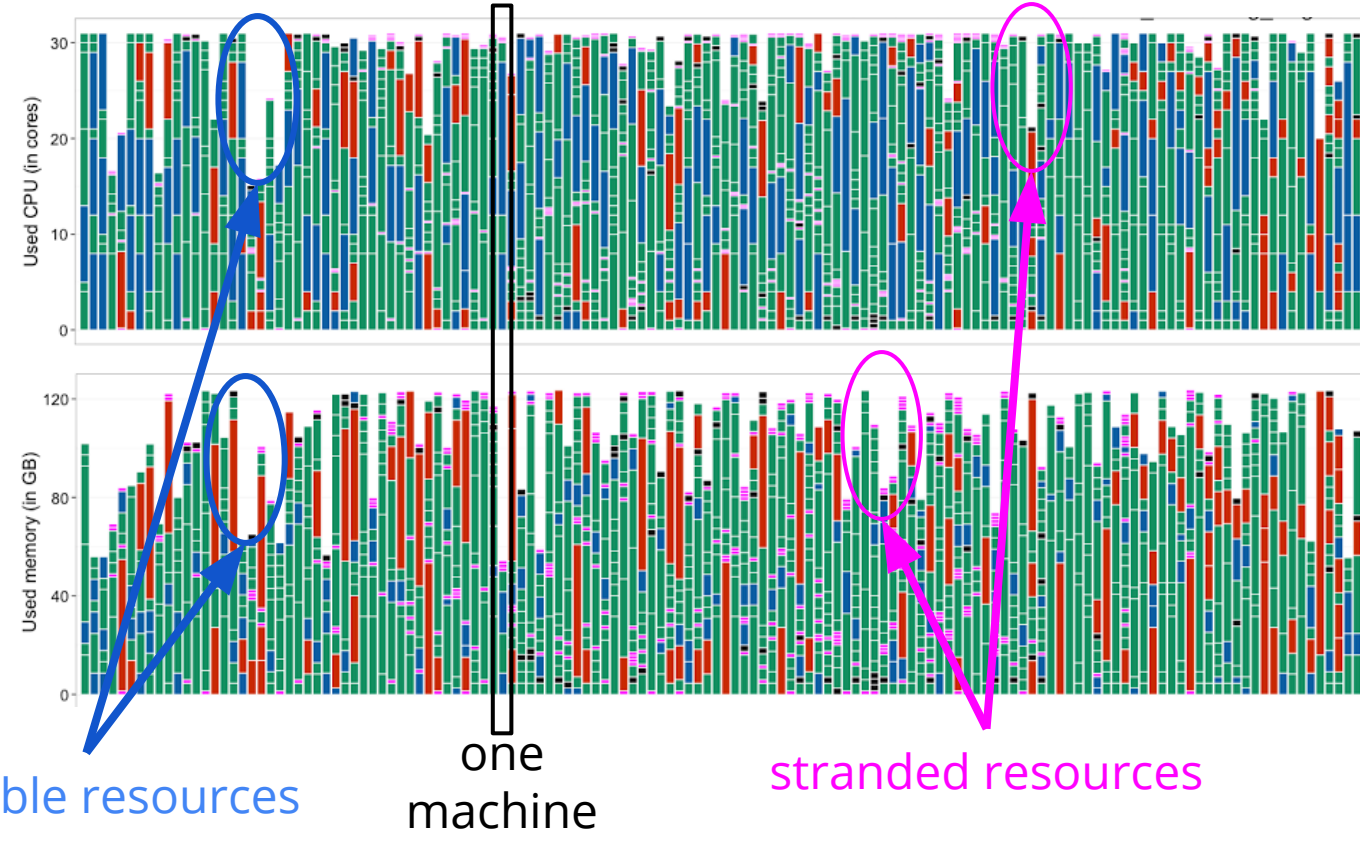
task-eviction rates and causes



Efficiency

Advanced bin-packing algorithms

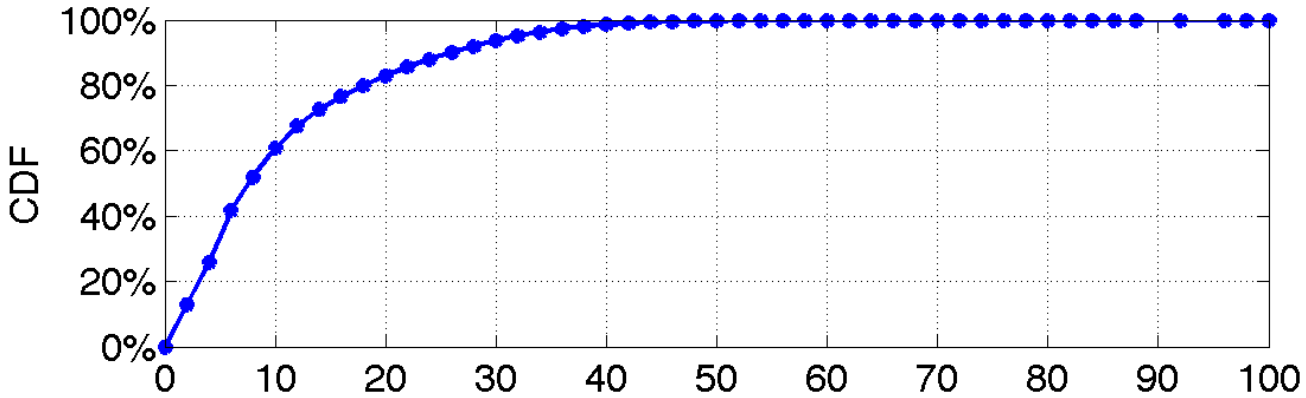
Experimental placement of production VM workload, July 2014



Efficiency

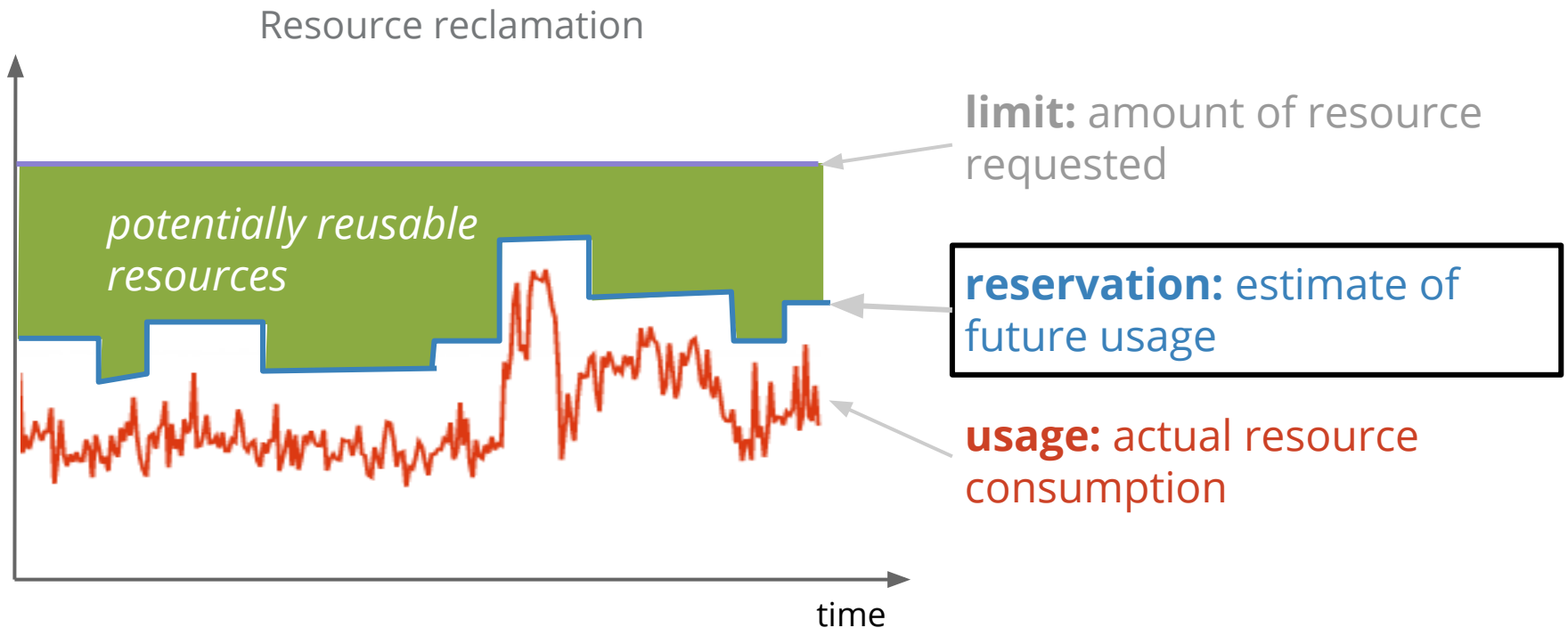
Multiple applications per machine

CPI^2 paper,
EuroSys 2013



tasks per machine

Efficiency


Resource reclamation
potentially reusable resources
time
limit: amount of resource requested
reservation: estimate of future usage
usage: actual resource consumption

Observations:

1. If your **developers** are spending time thinking about **individual machines**, you're operating at **too low-level of an abstraction**. You want to operate at the level of **applications**

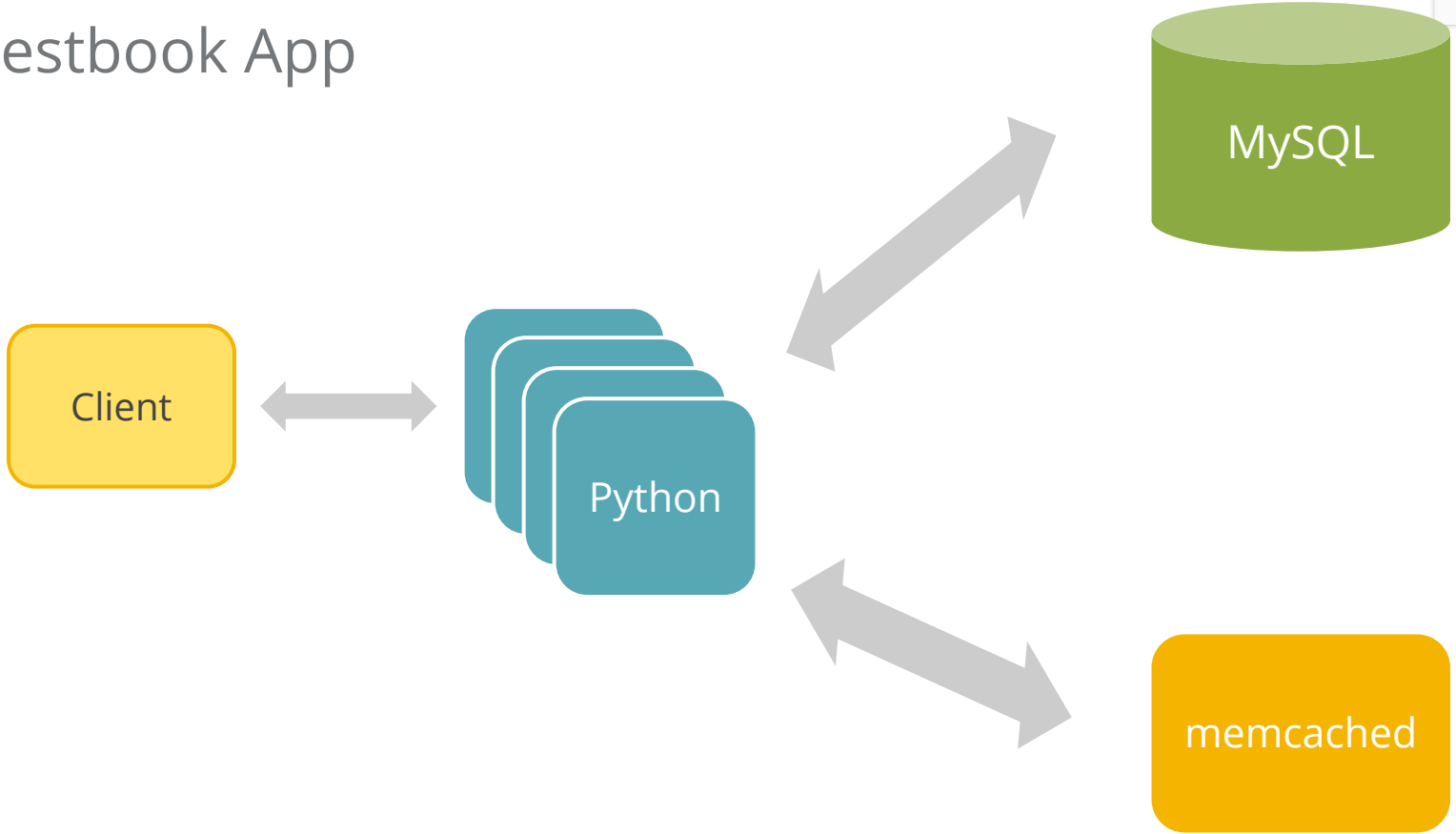
2. We get **efficiency** by:
   a. sharing resources
   b. reclaiming unused allocations

3. **Containers** make users more productive

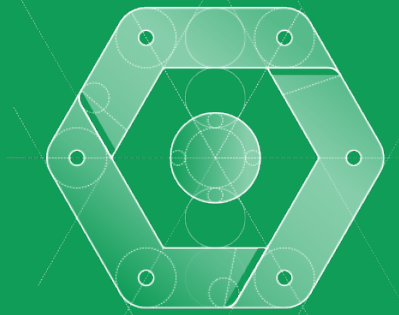
http://kubernetes.io
http://goo.gl/1C4nuo (Borg paper)

Images by Connie Zhou

Guestbook App



Containers



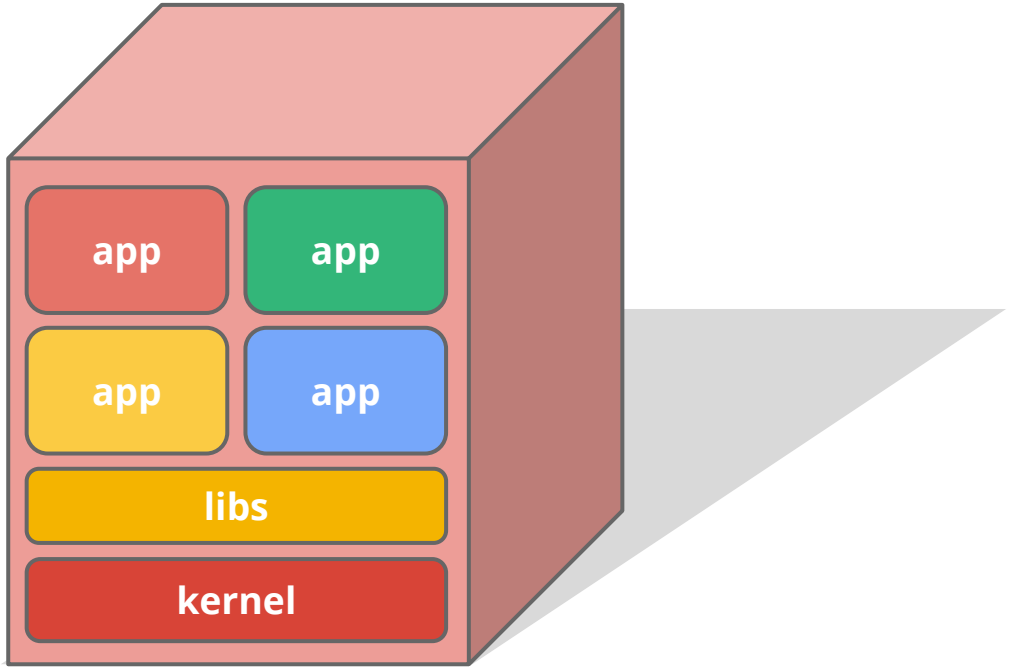
Old Way: Shared Machines

No isolation

No namespacing

Common libs

Highly coupled apps and OS



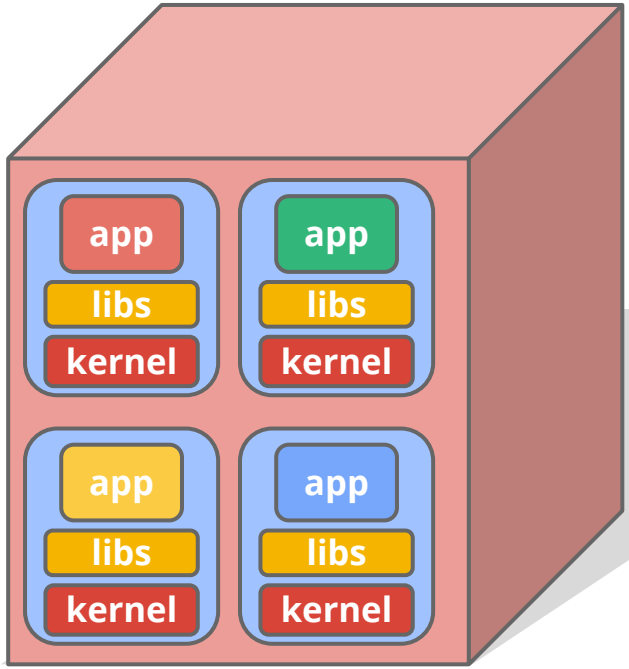
Old Way: Virtual Machines

Some isolation

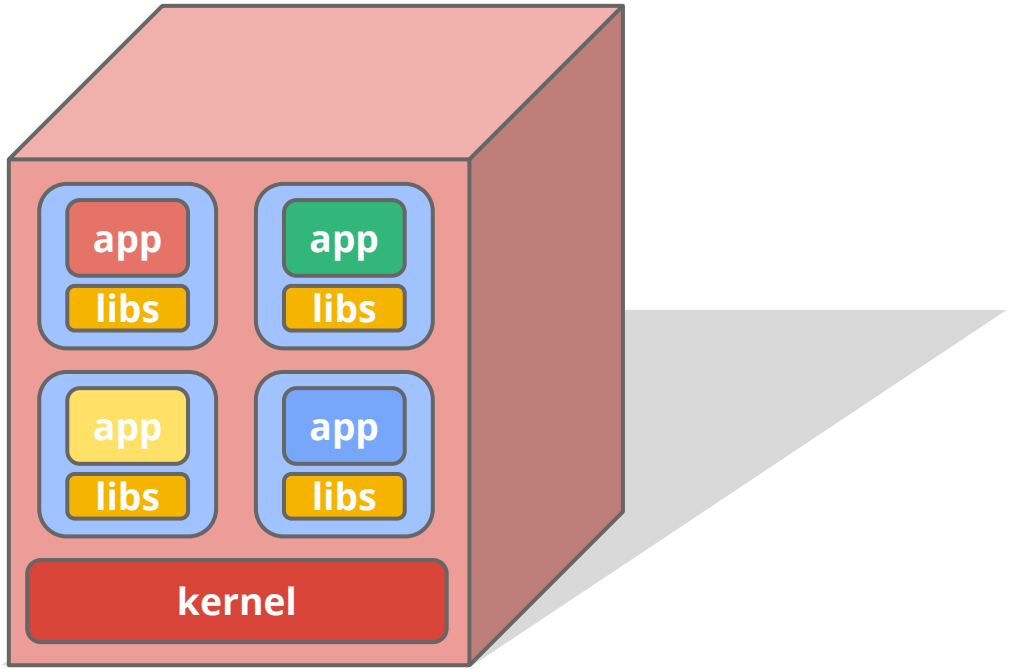
Inefficient

Still highly coupled to the guest OS

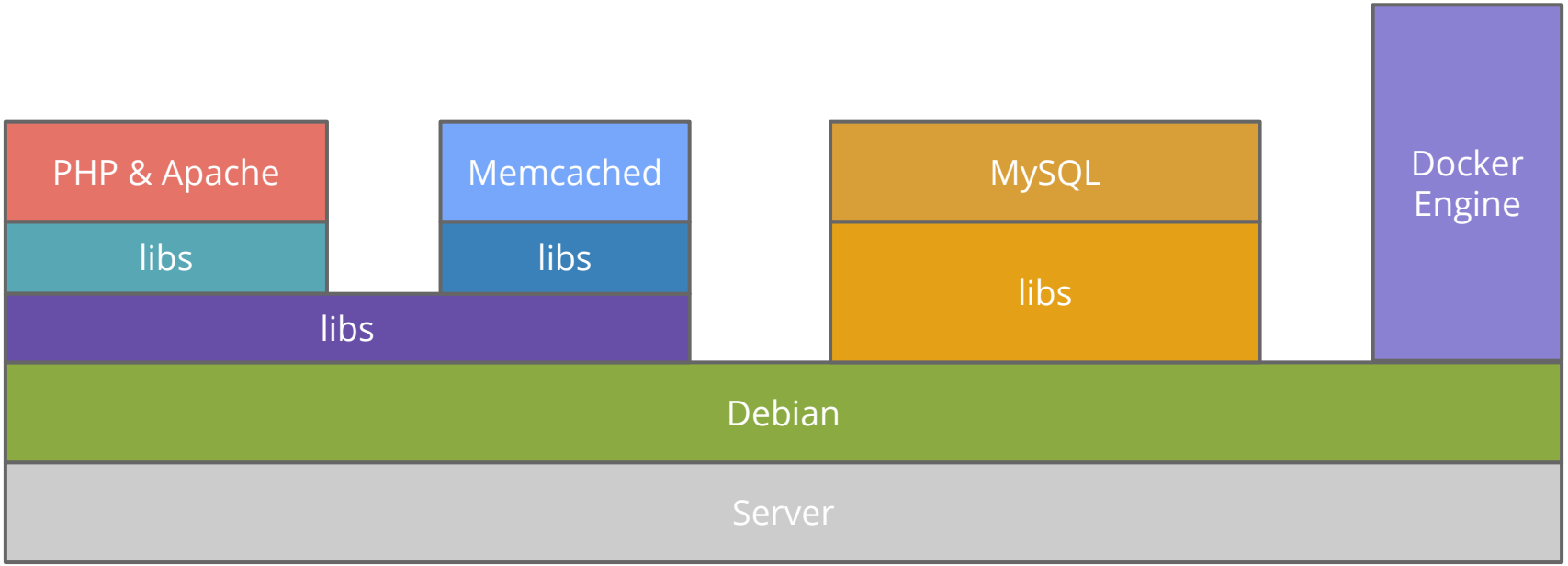
Hard to manage



New Way: Containers



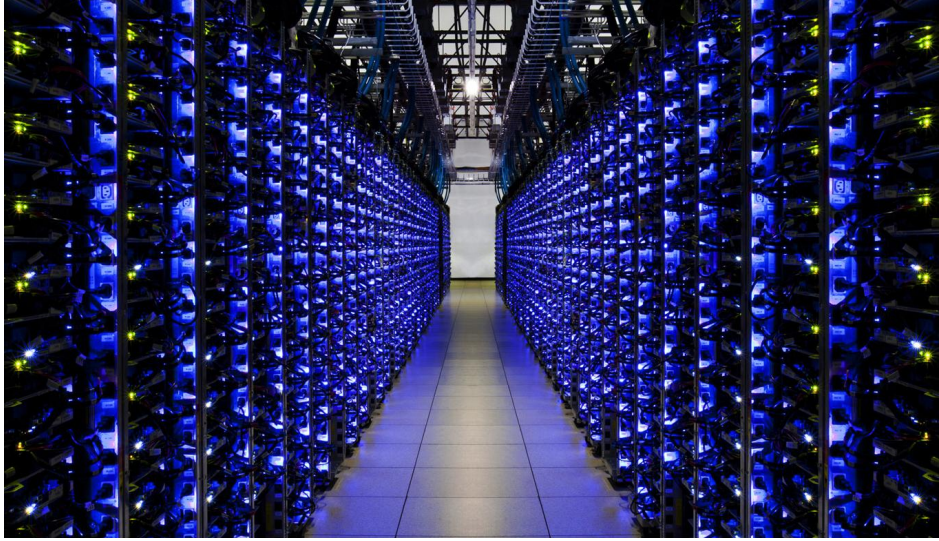
Docker Example

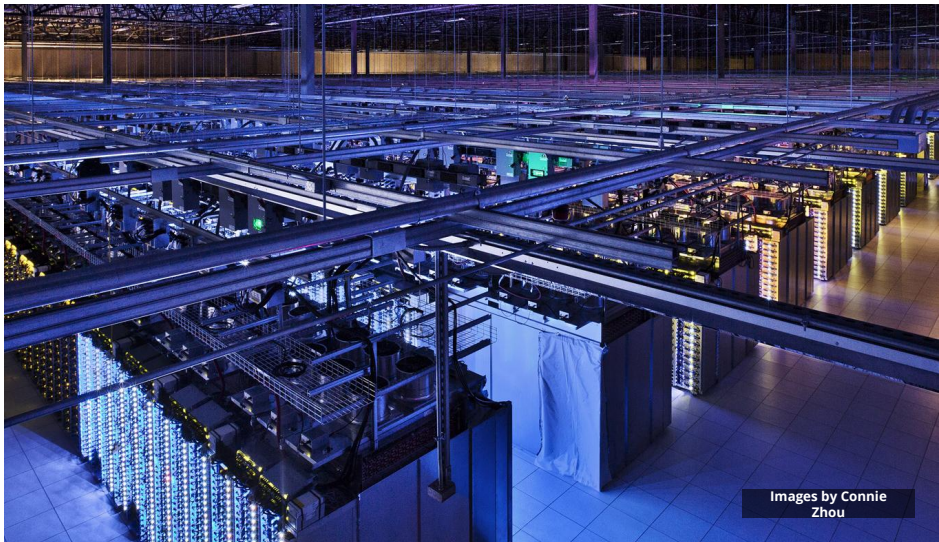


Why containers?

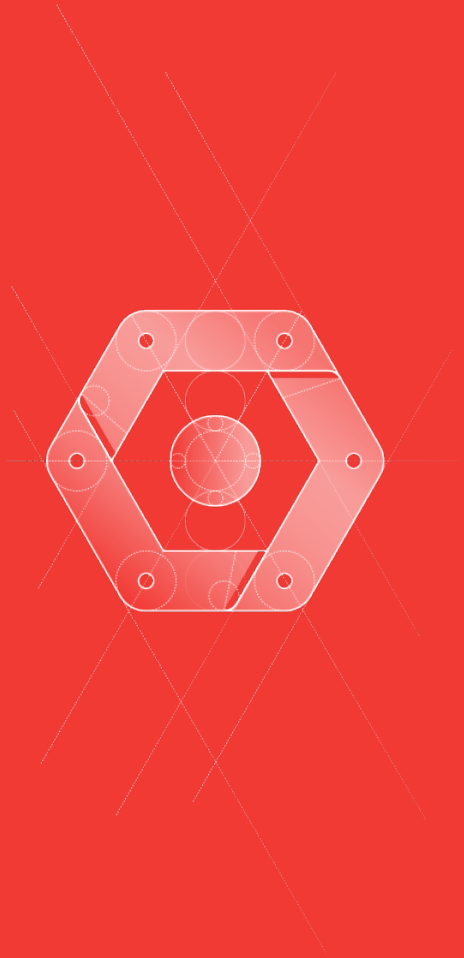
- Performance
- Repeatability
- Quality of service
- Accounting
- Portability

A **fundamentally different** way of managing **applications**





Demo



Kubernetes

Kubernetes

Greek for *“Helmsman”*; also the root of the word *“Governor”*

- Orchestrator for Docker containers
- Supports multi-cloud environments
- Inspired and informed by Google’s experiences and internal systems
- **Open source**, written in **Go**

Manage applications, not machines

Concepts Intro

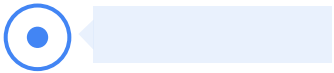
Container

Pod

Service

Volume

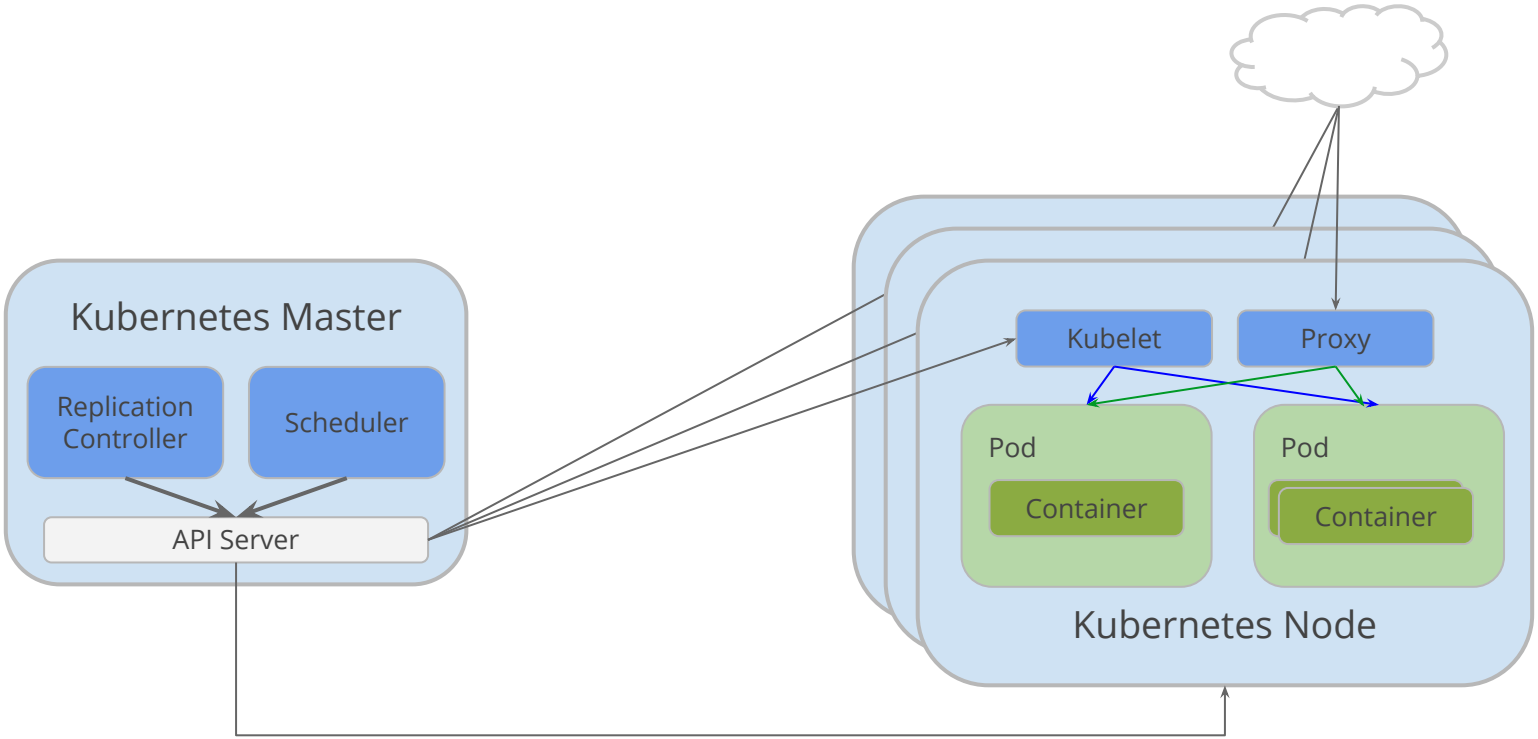
Label



Replication Controller

Node

Kubernetes Cluster



Cluster Options

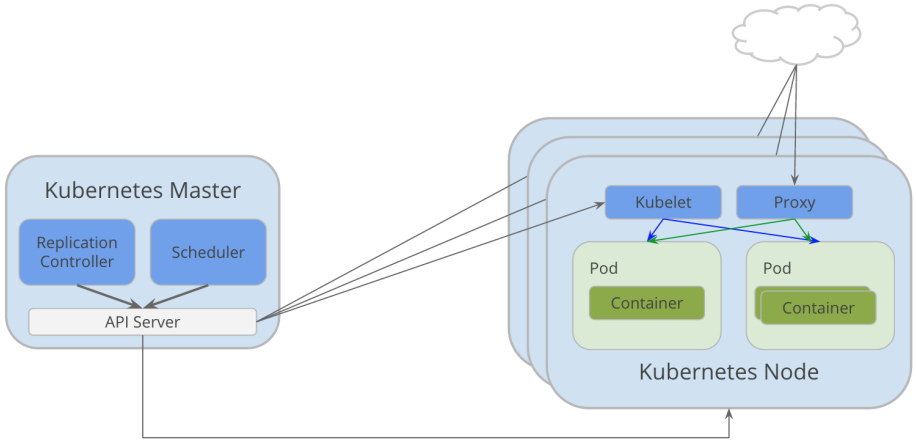
From **Laptop** to high-availability **multi-node cluster**

Hosted or **self managed**

On-Premise or **Cloud**

Bare Metal or **Virtual Machines**

Many options, See Matrix for details



Kubernetes Cluster Matrix: http://bit.ly/1MmhpMW

Pods

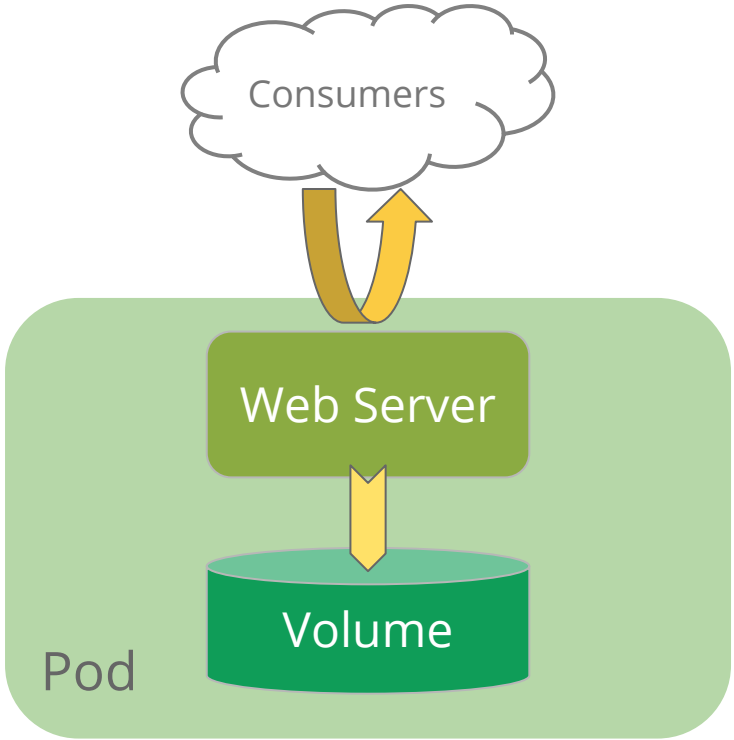
The atom of scheduling for containers

Application specific “logical host”

Ephemeral

- can die and be replaced

Single container pods can be created directly from a container image



Pods

Can be used to group containers & shared volumes

Containers are **tightly** coupled

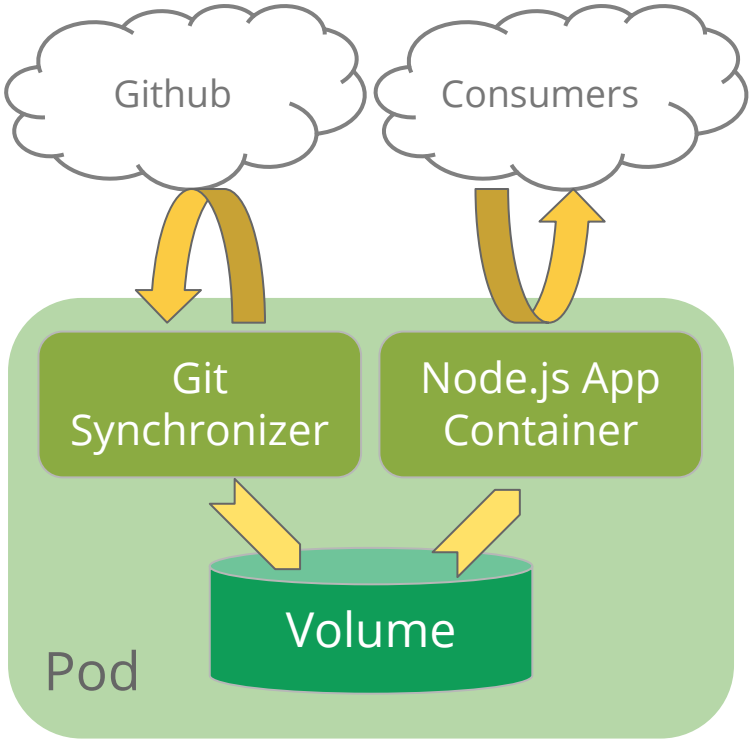
Shared namespace

- **Shared network IP and port namespace**

Ephemeral

- Containers in pods live and die together

Think in terms of services that you usually run on the same machine



Volume

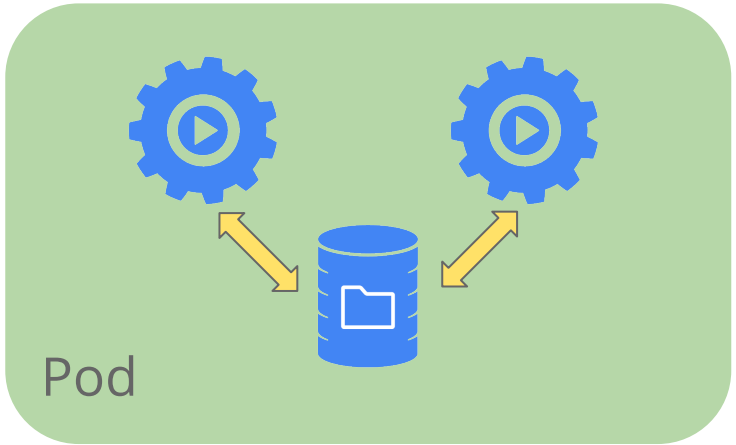
Bound to the Pod that encloses it

Look like Directories to Containers

What and where they are determined by Volume Type

Many Volume Type options

- **EmptyDir**
  - Lives with the pod
  - Can be backed by Memory (tmpfs)



Volume

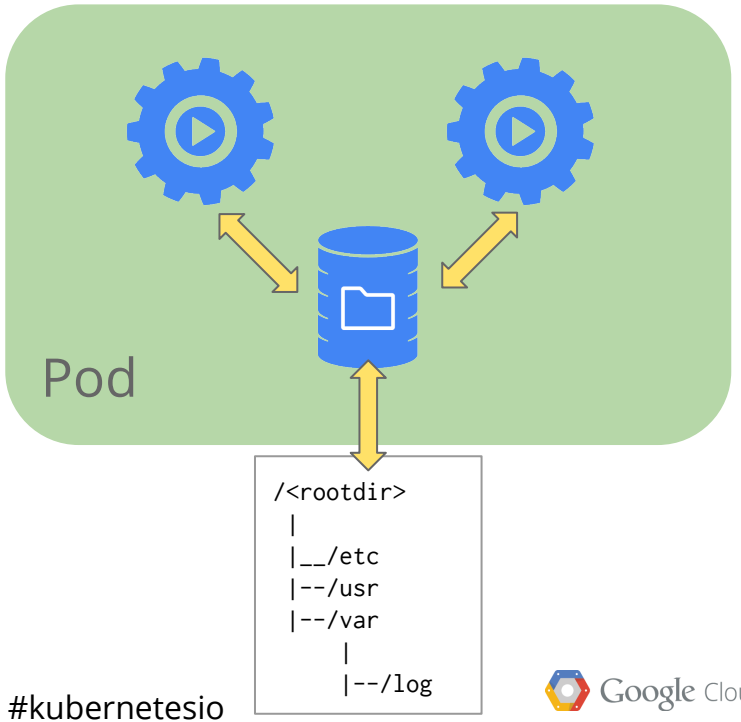
Bound to the Pod that encloses it

Look like Directories to Containers

What and where they are determined by Volume Type

Many Volume Type options

- EmptyDir
- **HostPath**
  - Maps to directory on host
  - Use with caution



Volume

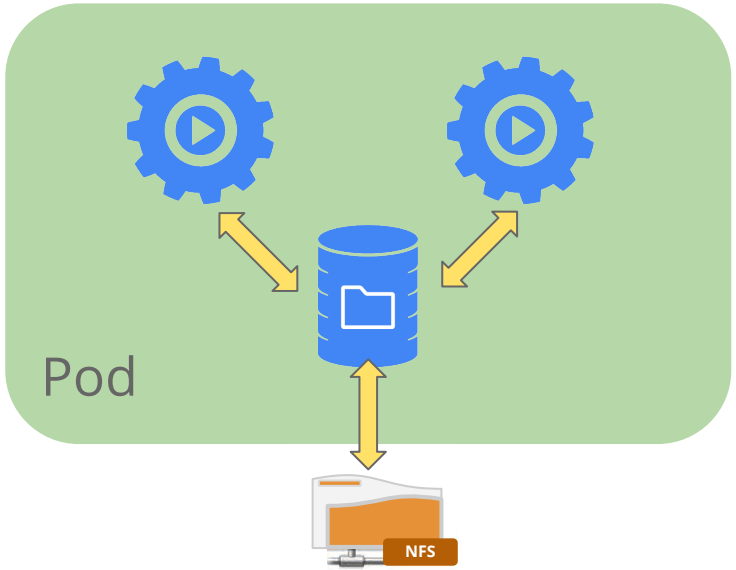
Bound to the Pod that encloses it

Look like Directories to Containers

What and where they are determined by Volume Type

Many Volume Type options

- EmptyDir
- HostPath
- **nfs (and similar services)**



Volume

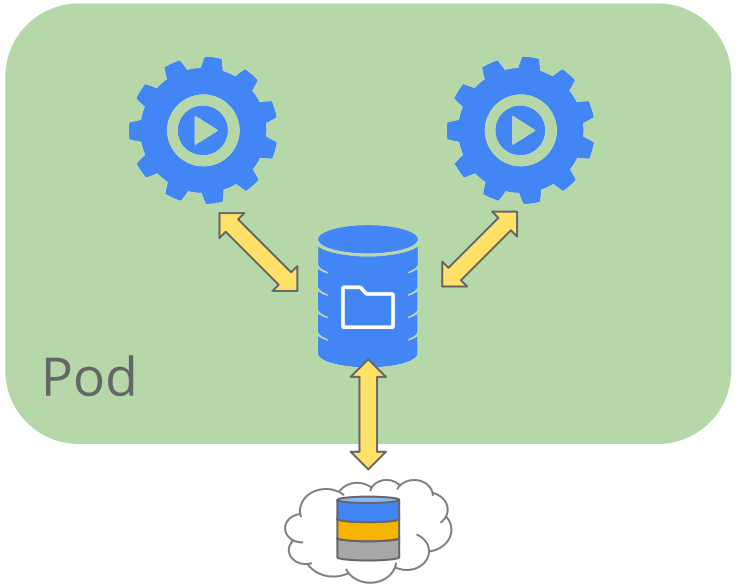
Bound to the Pod that encloses it

Look like Directories to Containers

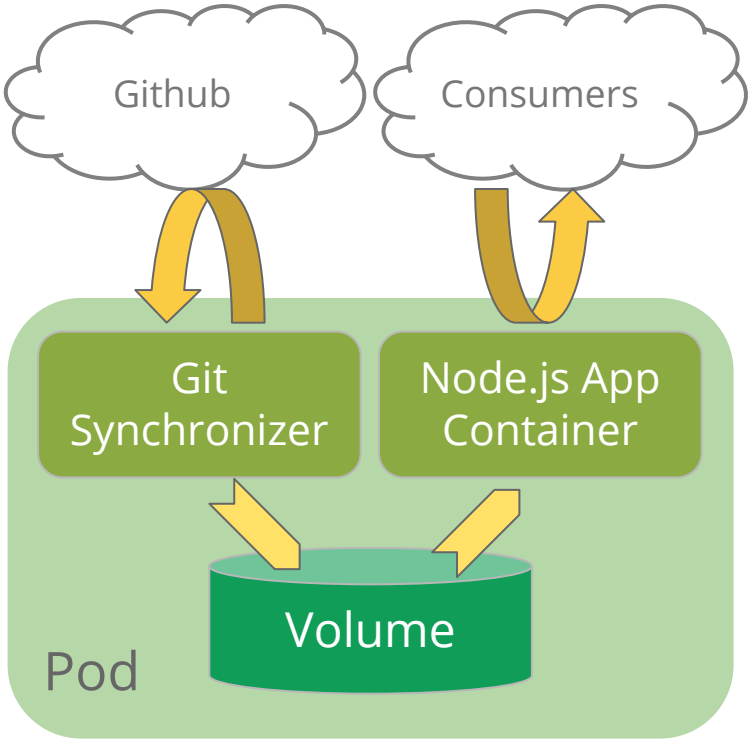
What and where they are determined by Volume Type

Many Volume Type options

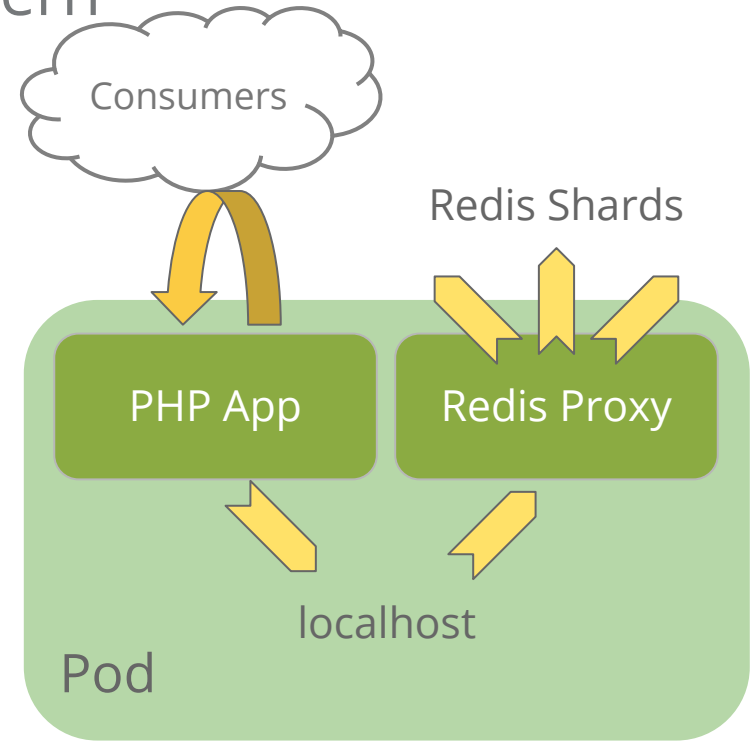
- EmptyDir
- HostPath
- nfs (and similar services)
- **Cloud Provider Persistent Block Storage**



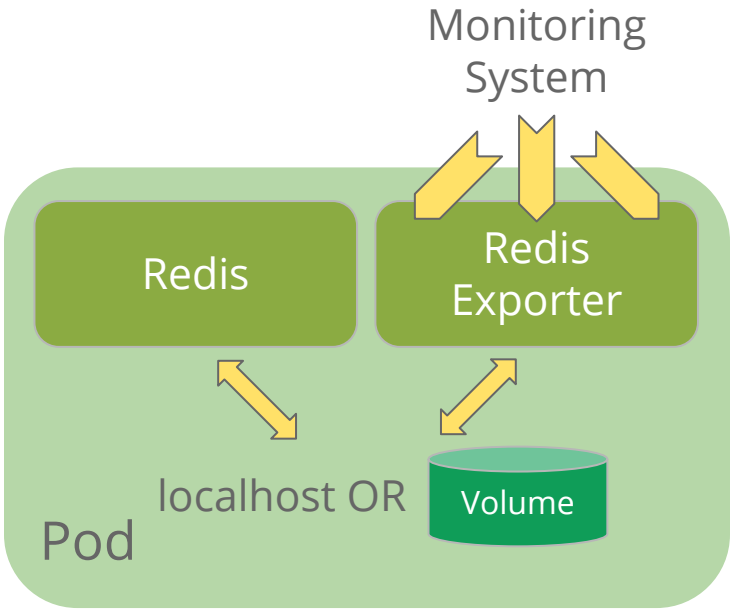
Sidecar Pattern



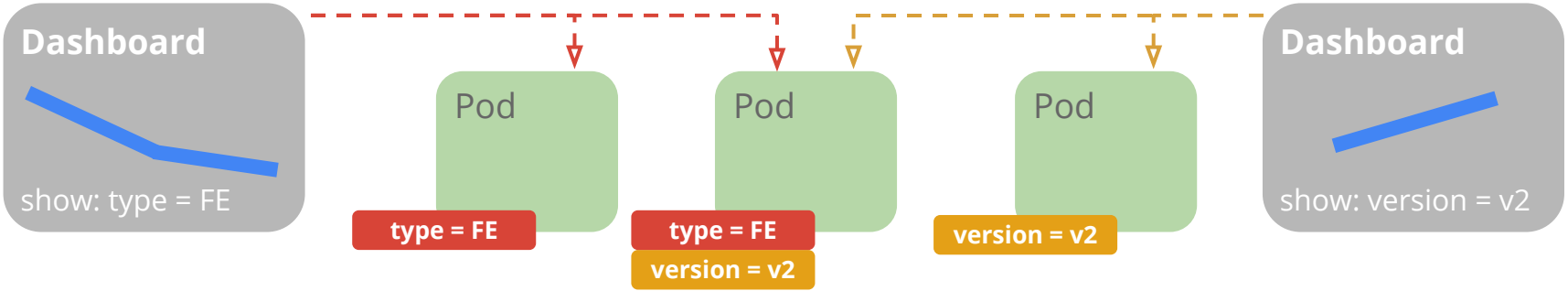
Ambassador Pattern



Adapter Pattern



Labels



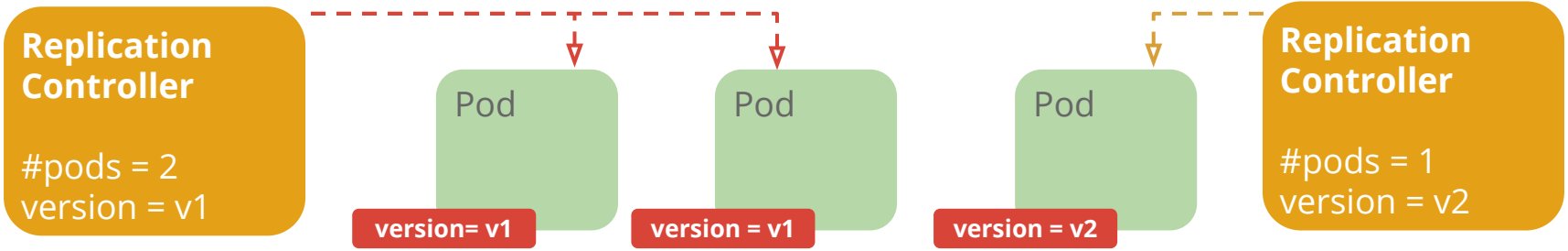
Behavior

- Metadata with semantic meaning
- Membership identifier
- The only Grouping Mechanism

Benefits

➔ Allow for intent of many users (e.g. dashboards)
➔ Build higher level systems ...
➔ Queryable by Selectors

Replication Controllers



Behavior

- Keeps Pods running
- Gives direct control of Pod #s
- Grouped by Label Selector

Benefits

- ➔ Recreates Pods, maintains desired state
- ➔ Fine-grained control for scaling
- ➔ Standard grouping semantics

Replication Controllers

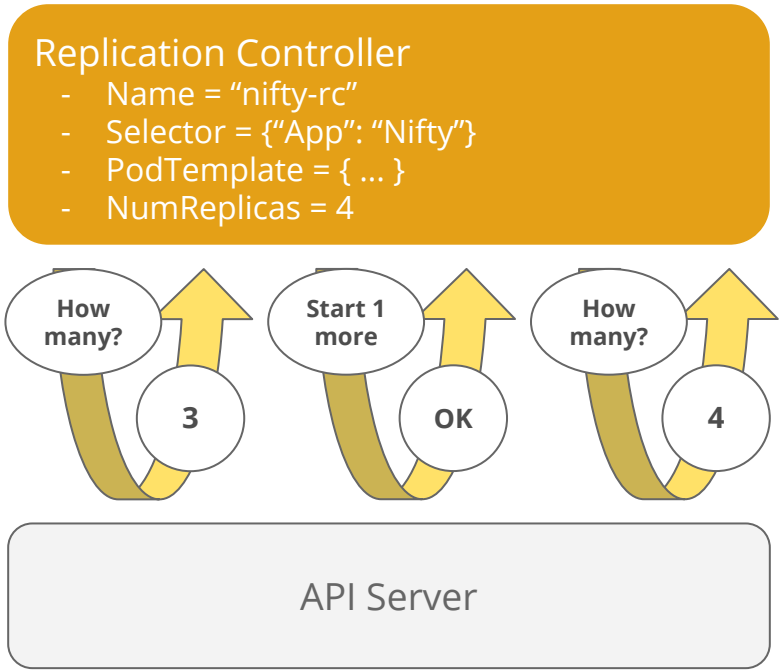
Canonical example of control loops

Have one job: ensure N copies of a pod

- if too few, start new ones
- if too many, kill some
- group == selector

Replicated pods are fungible

- No implied order or identity



Container Liveness

Process Level: Kubelet checks with Docker that Container is running

App Level: User defined health checks:

- HTTP Health checks (Kubelet calls a Web Hook)
- Container Exec (Kubelet runs command in container)
- TCP Socket (Kubelet attempts to open a socket to the container)

Image: IStockPhoto

Services

A group of pods that **act as one** == Service

- group == selector

Defines access policy

- only “load balanced” for now

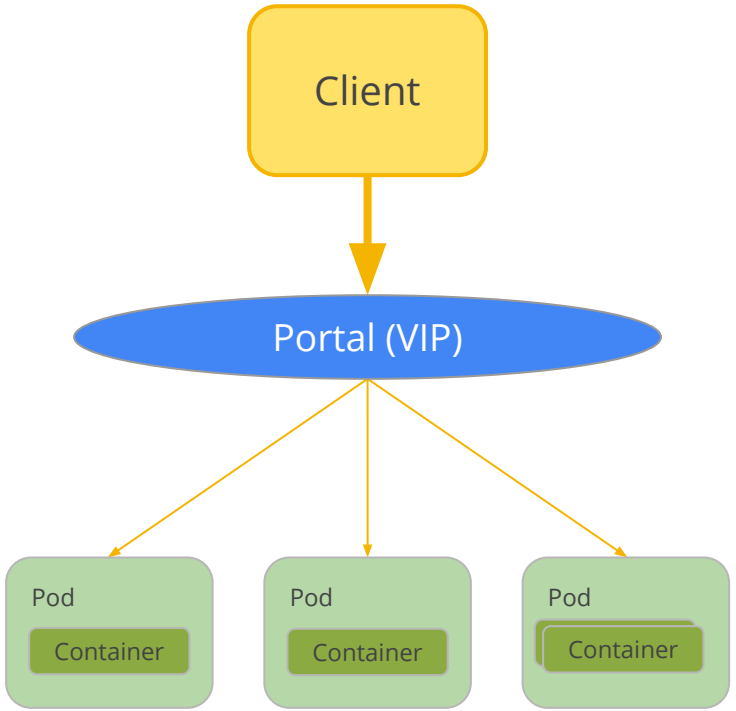
Gets a **stable** virtual IP and port

- called the service *portal*
- also a DNS name

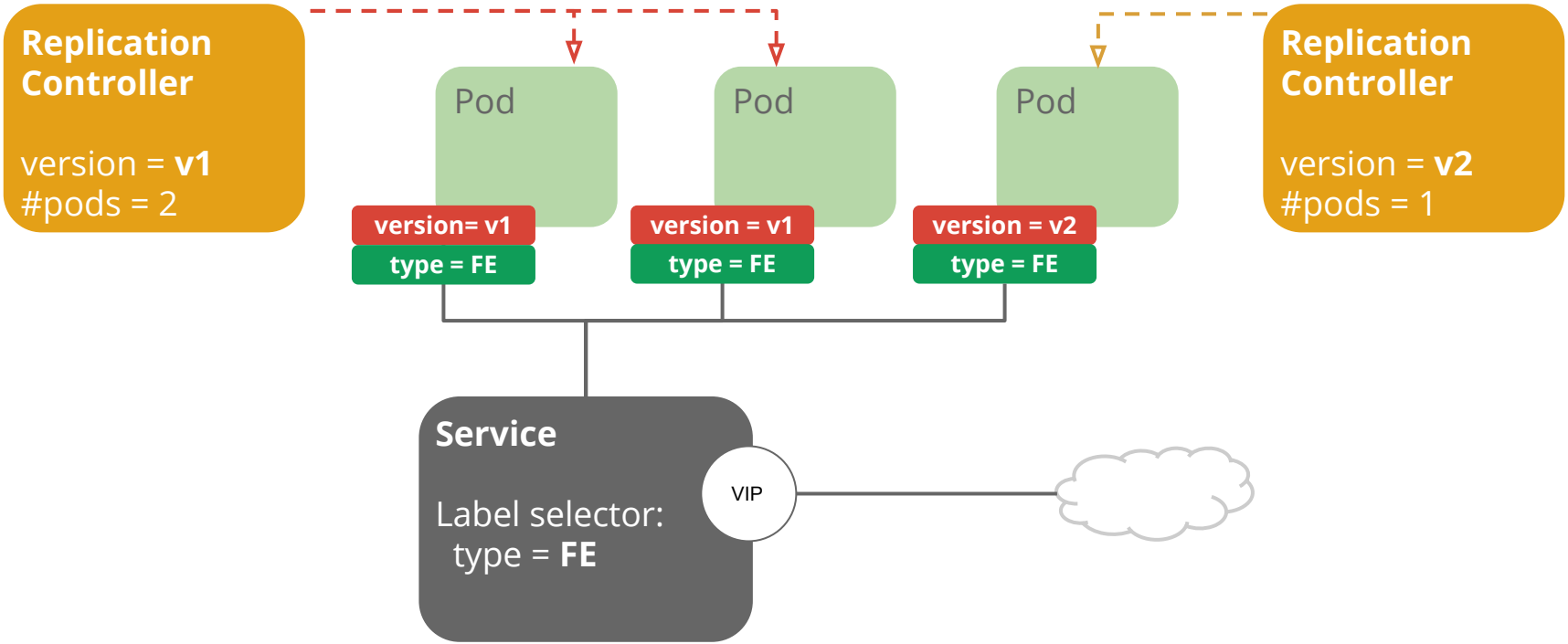
VIP is captured by *kube-proxy*

- watches the service **constituency**
- updates when backends change

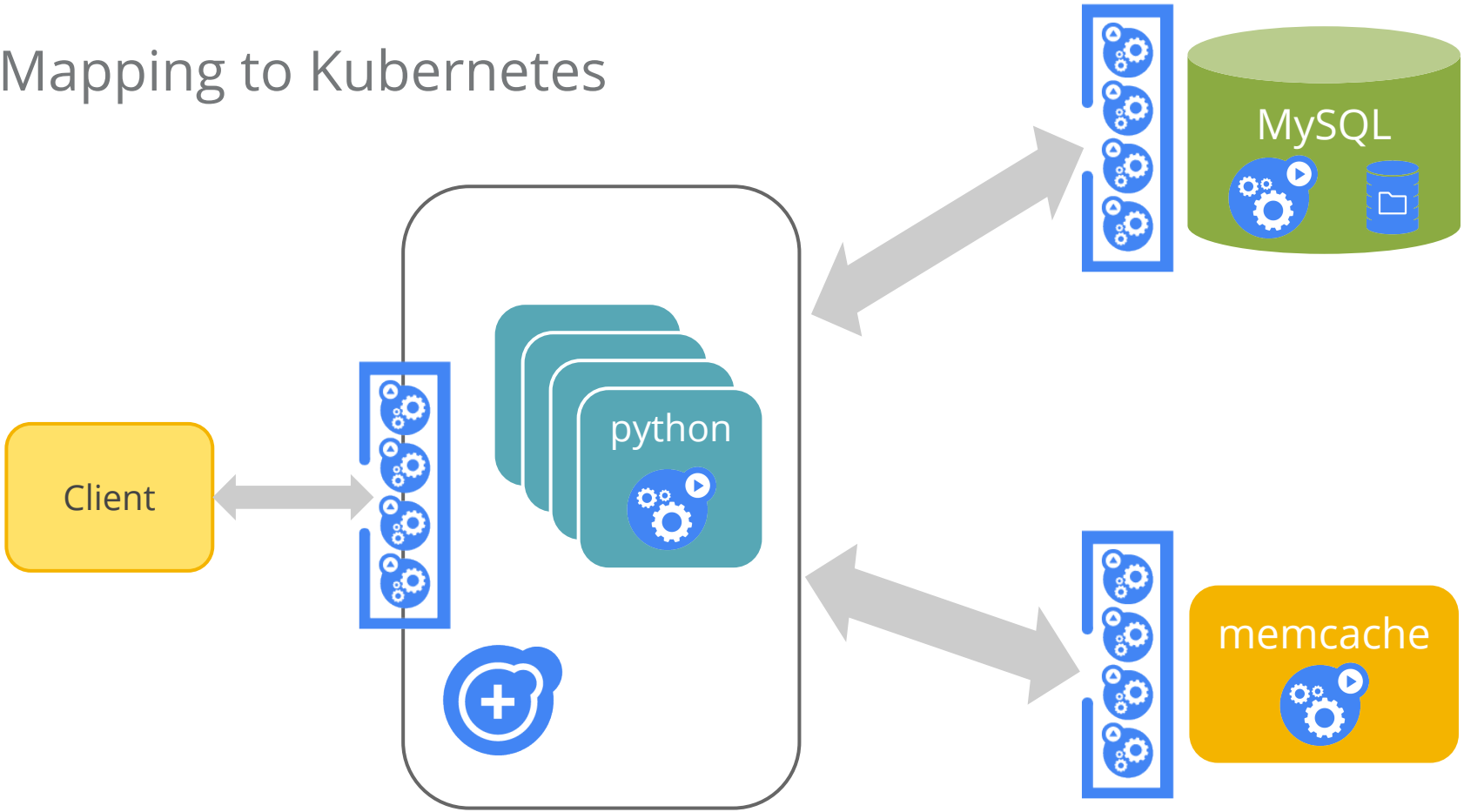
Hide complexity - ideal for non-native apps



Canary Example



Mapping to Kubernetes



Developer View (Replication Controller)

```
spec:
  containers:
  - name: php-guestbook
    image: php-guestbook:europython
    resources:
      limits:
        memory: "128Mi"
        cpu: "500m"
    ports:
    - containerPort: 80
      protocol: TCP
replicas: 10000
```

Scheduling Capabilities

Predicate based Currently

- Pod Selection
- Node Capacity (based on requested resource limits)

Prioritisation

- Nodes that match all predicates are ranked
- Priority for Node whose already-running pods consume the least resources

To Come

- Resource aware scheduling

Kubernetes Status

Kubernetes 1.0 as of mid July

- Formerly announced at OSCON this week

Open sourced in June, 2014

- won the BlackDuck “rookie of the year” award

Google launched **Google Container Engine** (GKE)

- hosted Kubernetes
- https://cloud.google.com/container-engine/

Roadmap:

- https://github.com/GoogleCloudPlatform/kubernetes/milestones

Google Container Engine (Beta)

Managed Kubernetes (Kubernetes v1)

Manages Kubernetes master uptime

Manages Updates

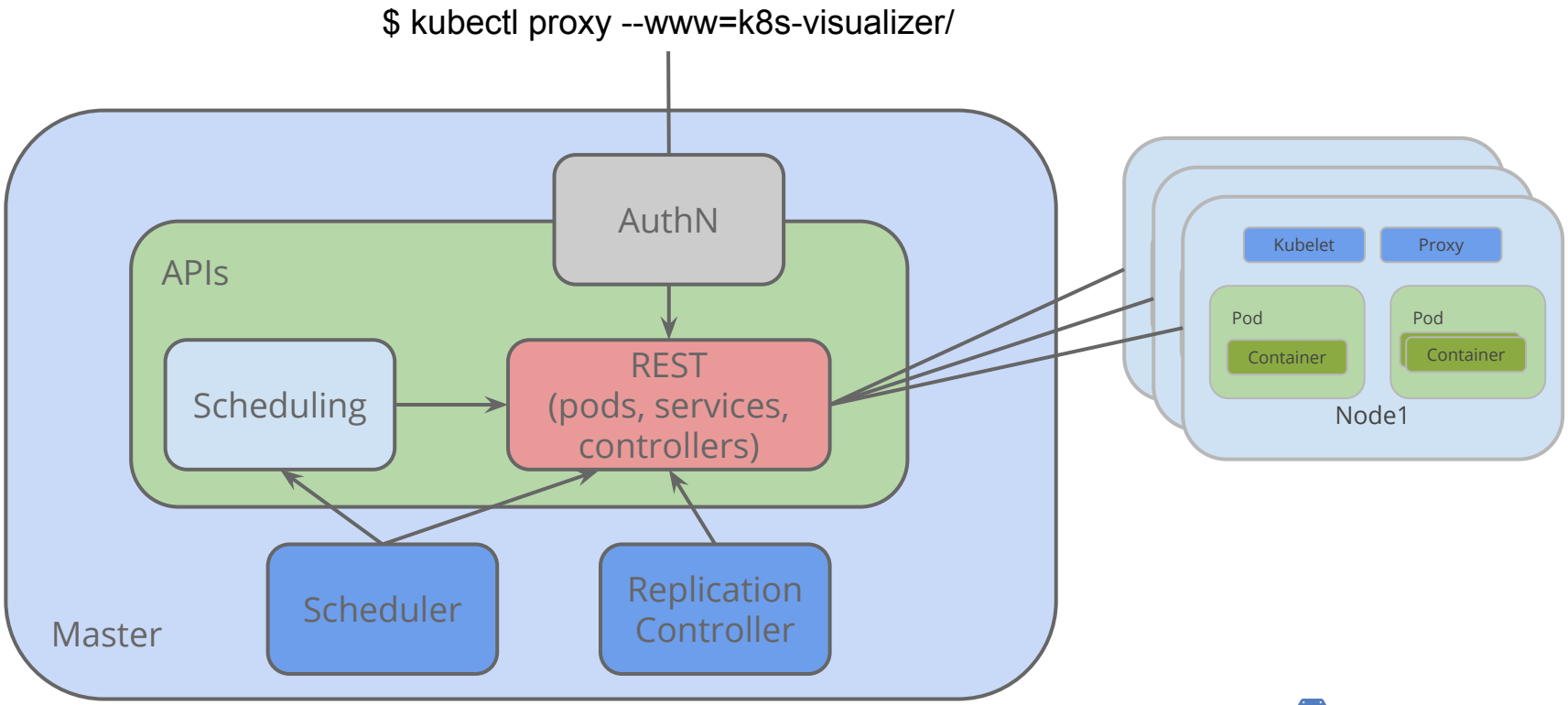
Cluster Resize via Managed Instance Groups

Centralised Logging

Google Cloud VPN support

Demo - Clusters and resizing

Visualizing Kubernetes



Container Engine Cluster Scaling

Managed Instance Group (MIG)

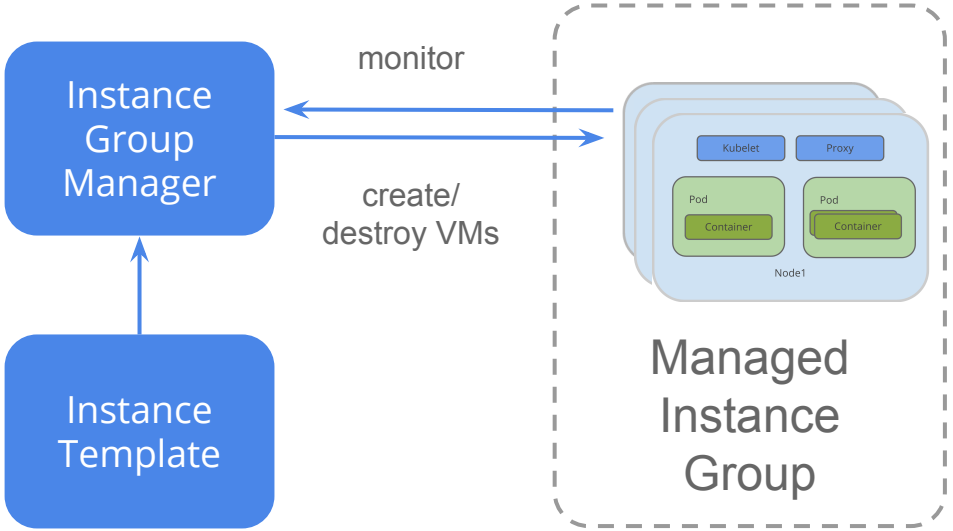
- Grouping of identical instances
- Instances are cluster nodes

Nodes Created from Template

- Includes Credentials and Tokens required to auth to the Master

MIG can be manually scaled

- New nodes request to be added to cluster
- Will eventually facilitate node level autoscale



Container Engine Cluster Creation

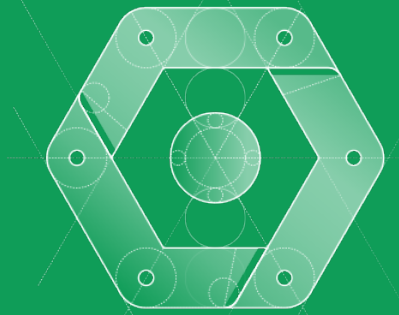
Google Developers Console

Google Deployment Manager

HashiCorp Terraform

```
resource "google_container_cluster"
"primary" {
    name = “europython-gke”
    zone = “europe-west1-c”
    initial_node_count = 3
    node_config {
        machine-type = “n1-standard-4”
    }
    master_auth {
        username = “bilbao”
        password = “notverysecret”
    }
}
```

Demo - Visualization



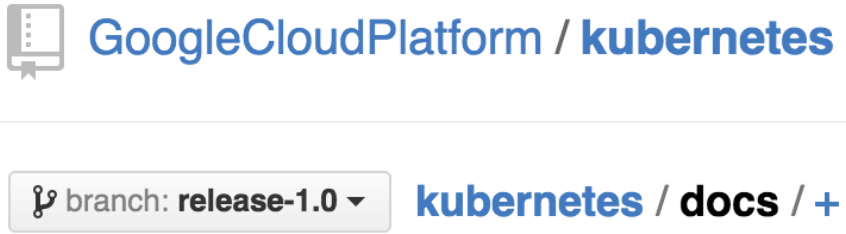
Frequently Asked Questions

Q. How does Kubernetes handle secrets?
docs/secrets.md

Q. How will Kubernetes scale pods?
docs/proposals/autoscaling.md
(Based on Traffic, predictive analysis or arbitrary data)

Q. How will Kubernetes scale nodes?
http://www.slideshare.net/craigbox/autoscaling-kubernetes
(Likely based on resources, signals from scheduler and pending queue)

Q. How can we make Kubernetes highly available
docs/availability.md



Kubernetes is **Open Source**

We want your help!

http://kubernetes.io

https://github.com/GoogleCloudPlatform/kubernetes

irc.freenode.net *#google-containers*

@kubernetesio

Also tweet questions to:
@tekgrrl
Or find me at the Google Booth today

Questions