Docker containers ...

Sebastien Goasguen,

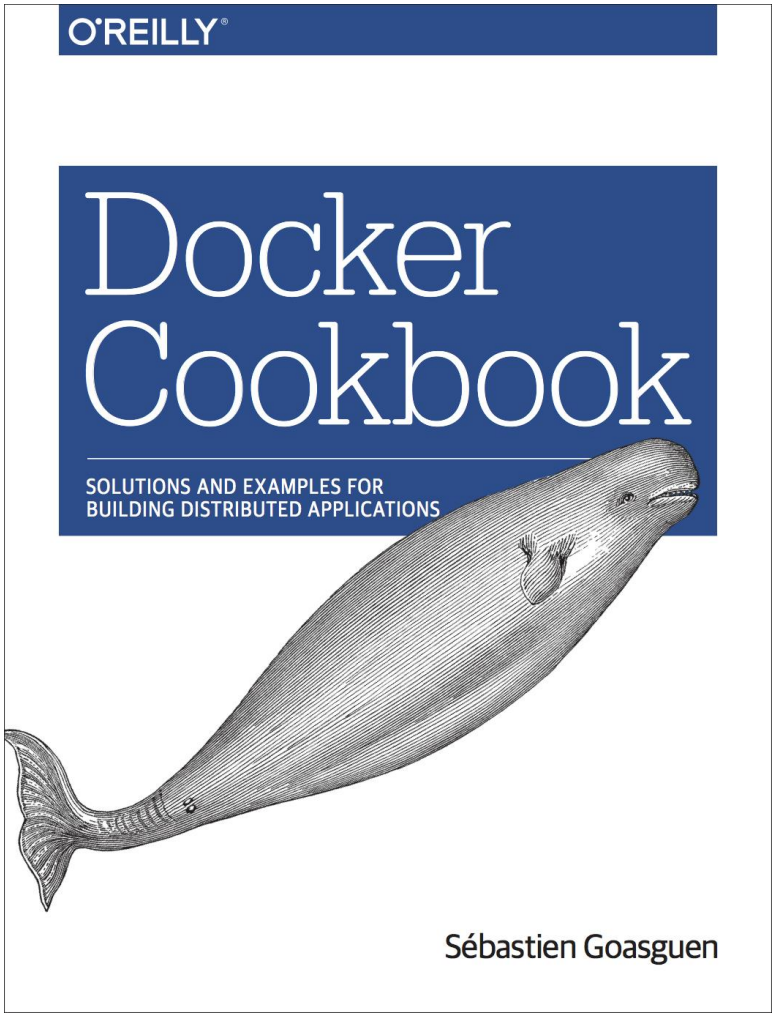
@sebgoa

Background

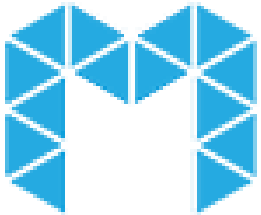
- Joined Citrix OSS team in July 2012
- Associate professor at Clemson University prior
- High Performance Computing, Grid computing (OSG, TG)
- At CERN summer 2009/2010, help build LXCLOUD based on opennebula
- http://sebgoa.blogspot.com

What do I do ?

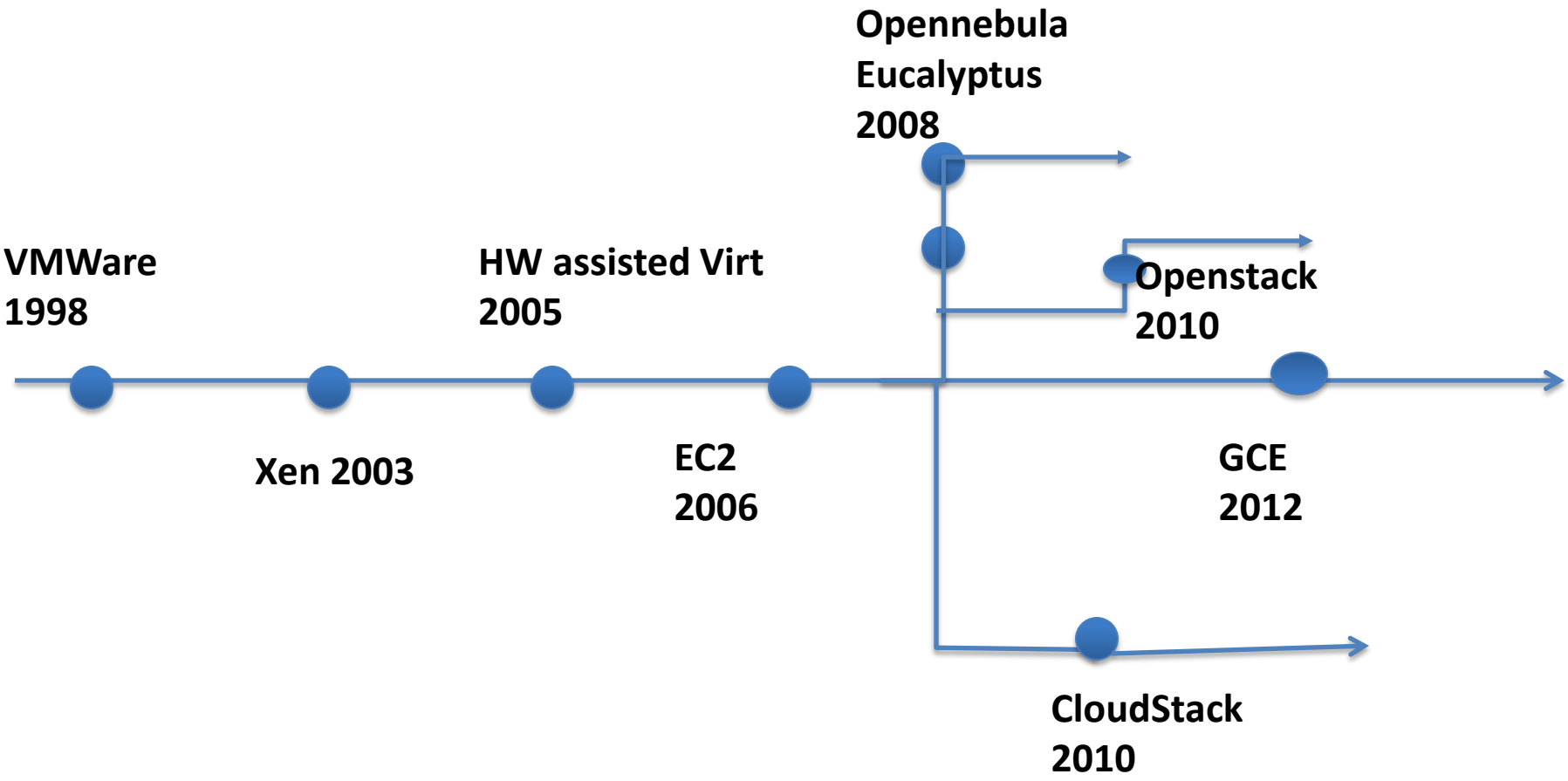
- Apache CloudStack and licloud committer + PMC member
- Looking at techs and how they work together
- Half dev, half community manager, + half event planner



Today's talk



IaaS History


Opennebula
Eucalyptus
2008
VMWare
1998
HW assisted Virt
2005
Openstack
2010
Xen 2003
EC2
2006
GCE
2012
CloudStack
2010

Goals

- Utility computing
- Elasticity of the infrastructure
- On-demand
- Pay as you go
- Multi-tenant
- Programmable access

So what…

Let’s assume this is solved.

What is not solved:

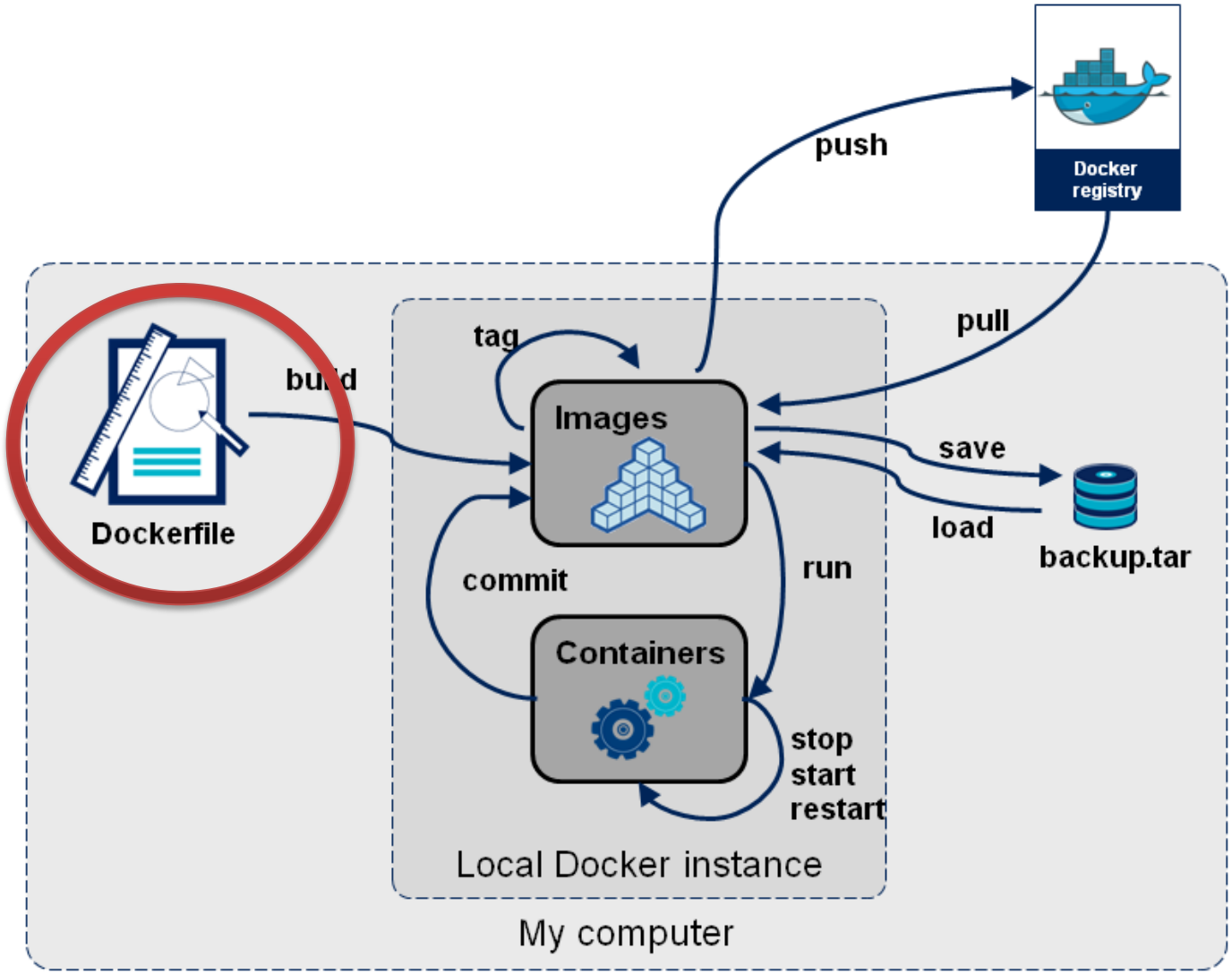
- Application deployment
- Application scalability
- Application portability
- Application composability

Docker

Docker

- Linux container (LXC +)
- Application deployment
- PaaS
- Portability
- Image sharing via DockerHub
- Ease of packaging applications

Building docker images



Fair use from http://blog.octo.com/en/docker-registry-first-steps/

Eureka moment #1

Installation

```
$ sudo curl -sSL
https://get.docker.com/ubuntu/ |
sudo sh

$ sudo yum install docker
```

Use

```
$ docker run busybox echo foobar
Foobar

$ docker run -ti ubuntu:14.04
/bin/bash
root@0156ad334ca4:/#
```

The App store

```
$ docker push runseb/application

$ docker pull runseb/application

$ docker run -d runseb/application
```

Docker gotchas

Networking

Bridge in the host

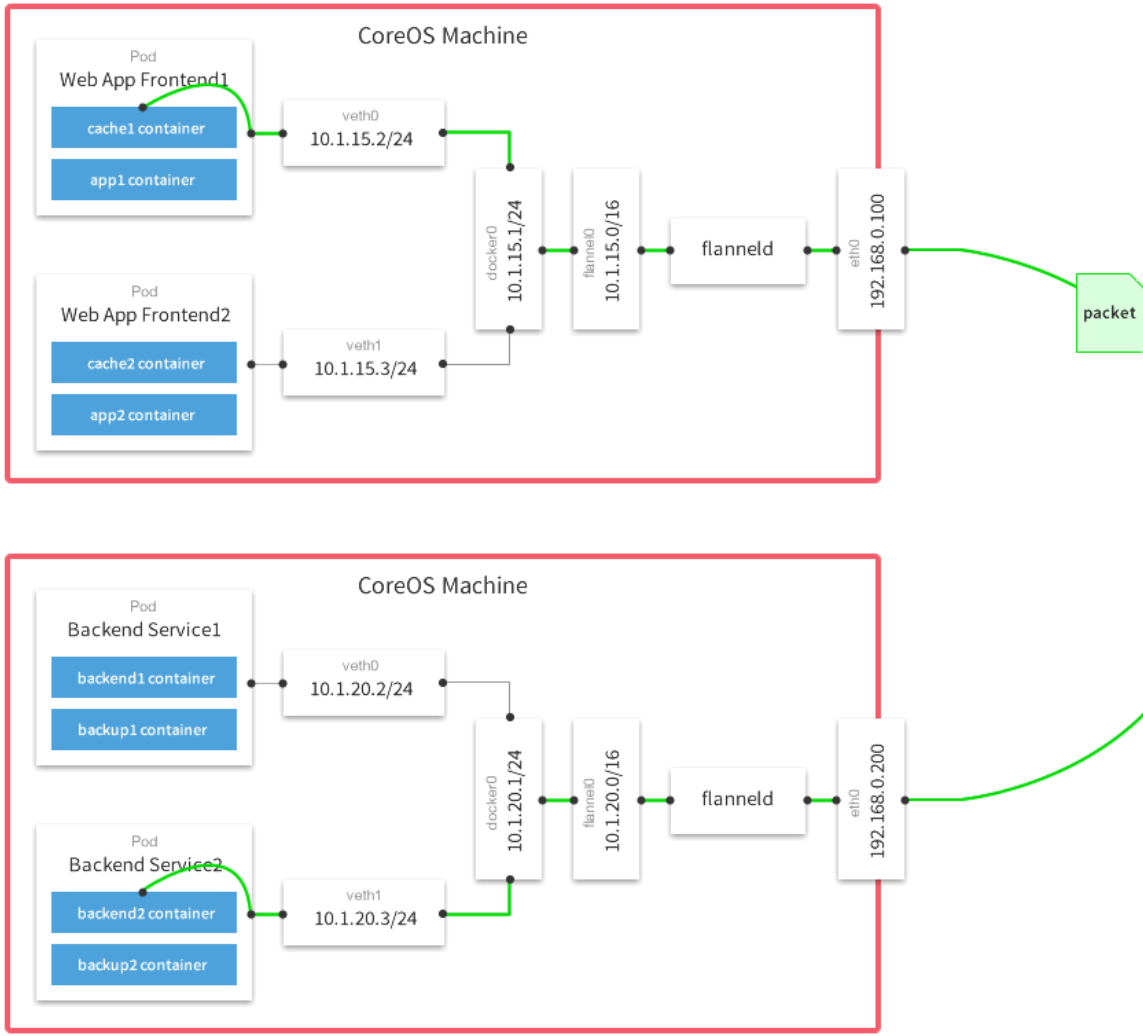
Port mapping to expose services on the host

```
Chain DOCKER (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             172.17.0.4
tcp dpt:www
```

Multi-Host networking

Weave.works

Flannel

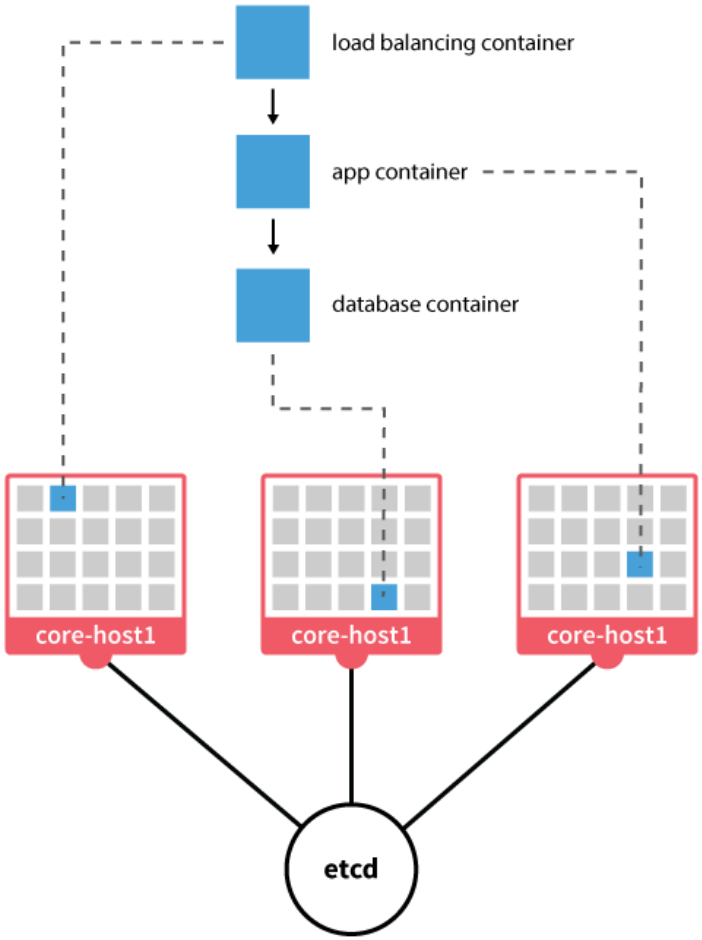


Other gotchas

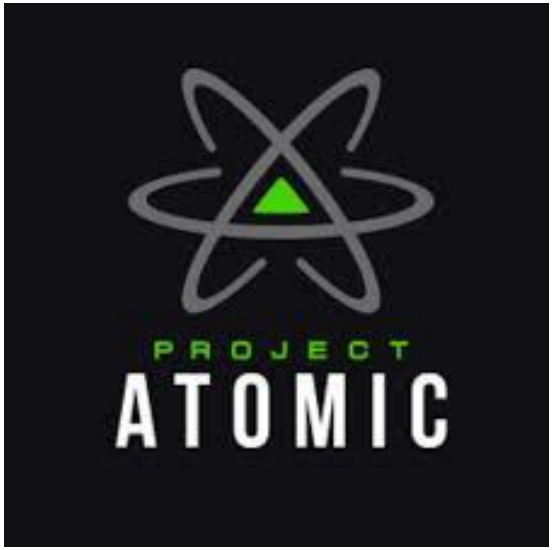
- No init system in the container
- Foreground processes
- Root
- Data volumes
- Data persistence
- How small does an image get for real applications ?

Eureka moment #2

CoreOS


load balancing container
app container
database container
core-host1
core-host1
core-host1
etcd

Similar projects



coreOS

CoreOS

- Linux distribution
- Rolling upgrades
- Minimal OS
- Docker support
- `etcd` and `fleet` tools to manage distributed applications based on containers.
- Cloud-init support
- **Systemd units**

coreOS “OEM”

oem-azure	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-cloudstack	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-digitalocean	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-ec2-compat	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-exoscale	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-gce	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-hyperv	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-rackspace-onmetal	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-rackspace	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-vagrant-key	add(coreos-base/oem-vagrant-key): Add oem for vmware_insecure images.	6 months ago
oem-vagrant	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago

http://github.com/coreos/coreos-overlay

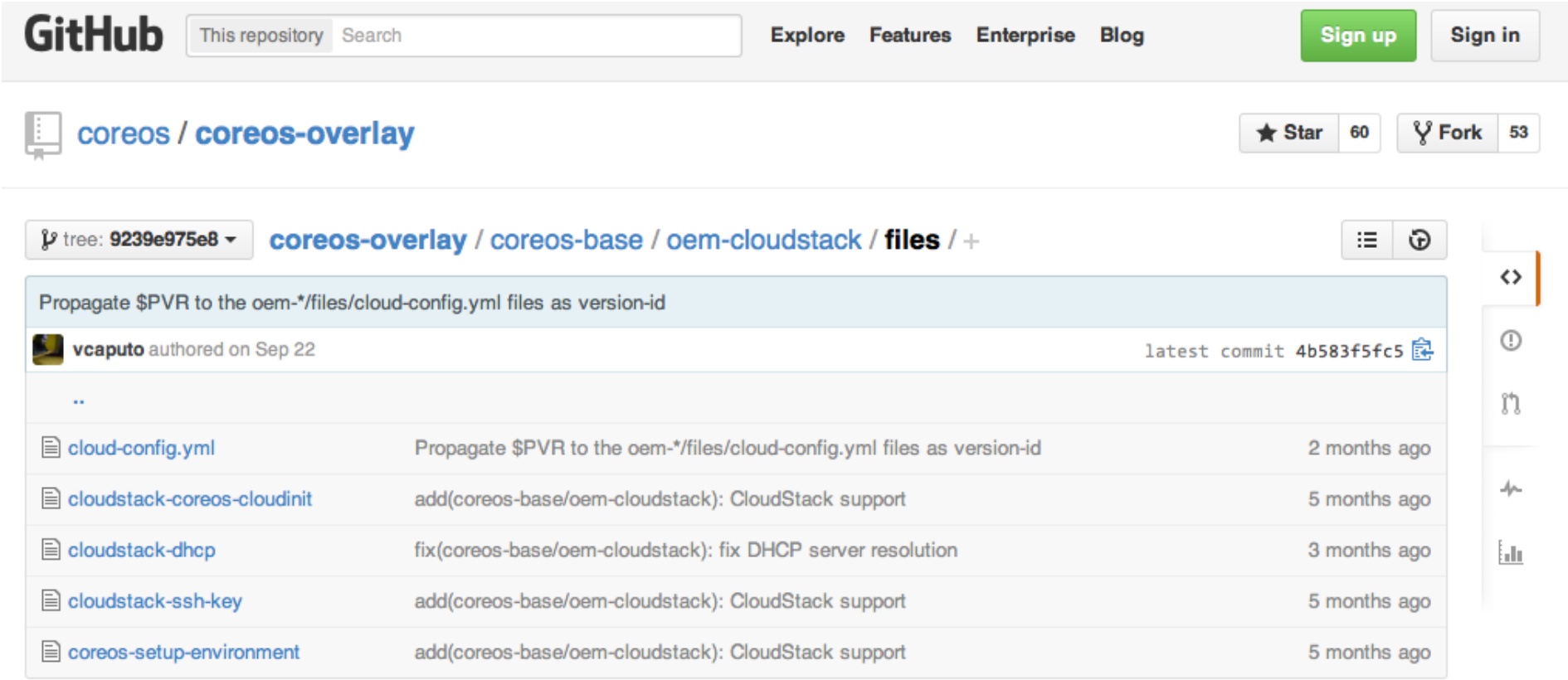
coreOS

“OEM”

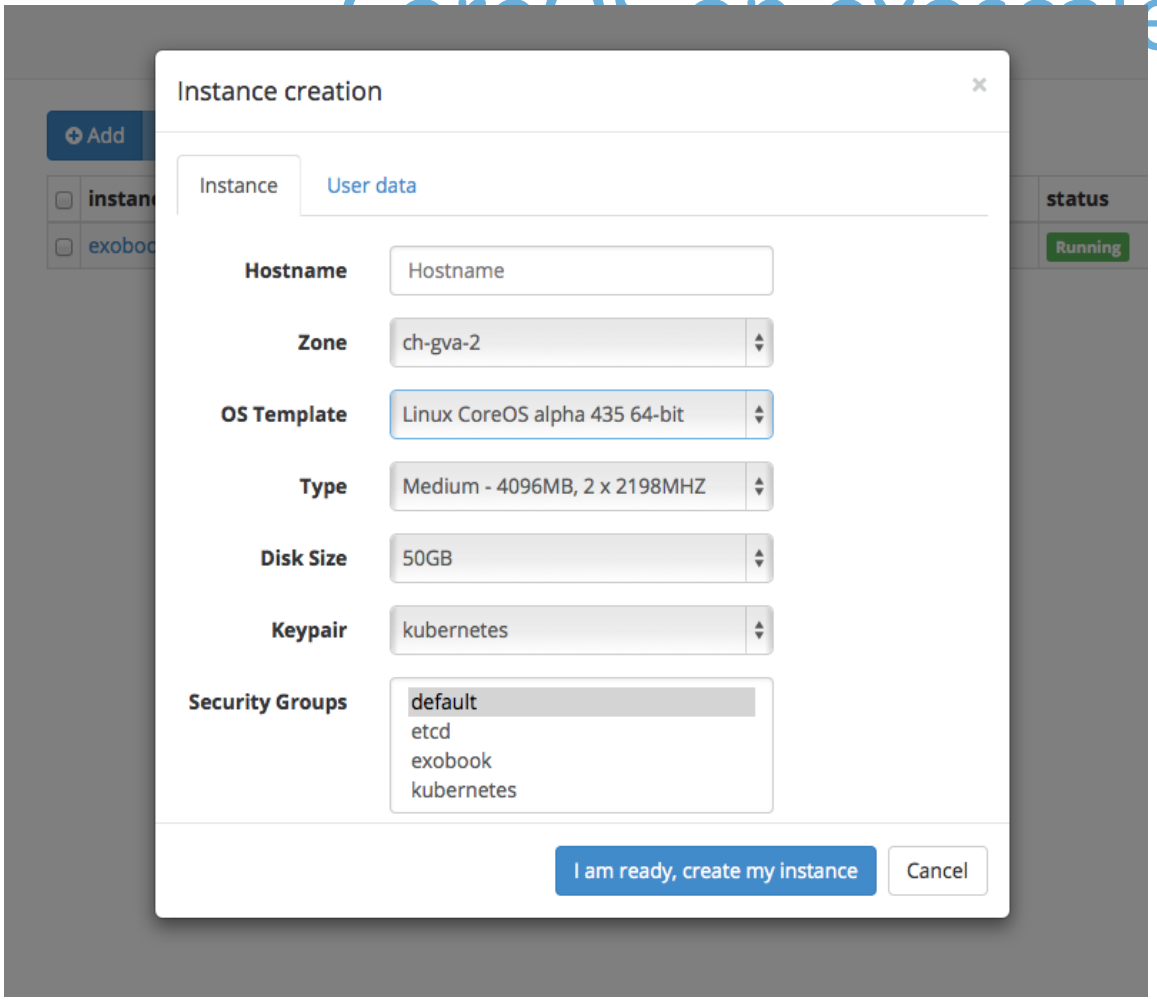
oem-azure	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-cloudstack	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-digitalocean	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-ec2-compat	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-exoscale	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-gce	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-hyperv	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-rackspace-onmetal	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-rackspace	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago
oem-vagrant-key	add(coreos-base/oem-vagrant-key): Add oem for vmware_insecure images.	6 months ago
oem-vagrant	Propagate $PVR to the oem-*/files/cloud-config.yml files as version-id	2 months ago

http://github.com/coreos/coreos-overlay

The cloudinit magic



CoreOS on exoscale



Starting containers

```
#cloud-config

coreos:
  units:
    - name: docker.service
      command: start
    - name: es.service
      command: start
      content: |
        [Unit]
        After=docker.service
        Requires=docker.service
        Description=starts ElasticSearch container

        [Service]
        TimeoutStartSec=0
        ExecStartPre=/usr/bin/docker pull dockerfile/elasticsearch
        ExecStart=/usr/bin/docker run -d -p 9200:9200 -p 9300:9300
dockerfile/elasticsearch
```

Opportunity

CERN cloud to offer templates for:

- Coreos
- Snappy
- Atomic

Create a coreOS OEM upstream with cern specific contextualization

DEMO ?

CoreOS clustering

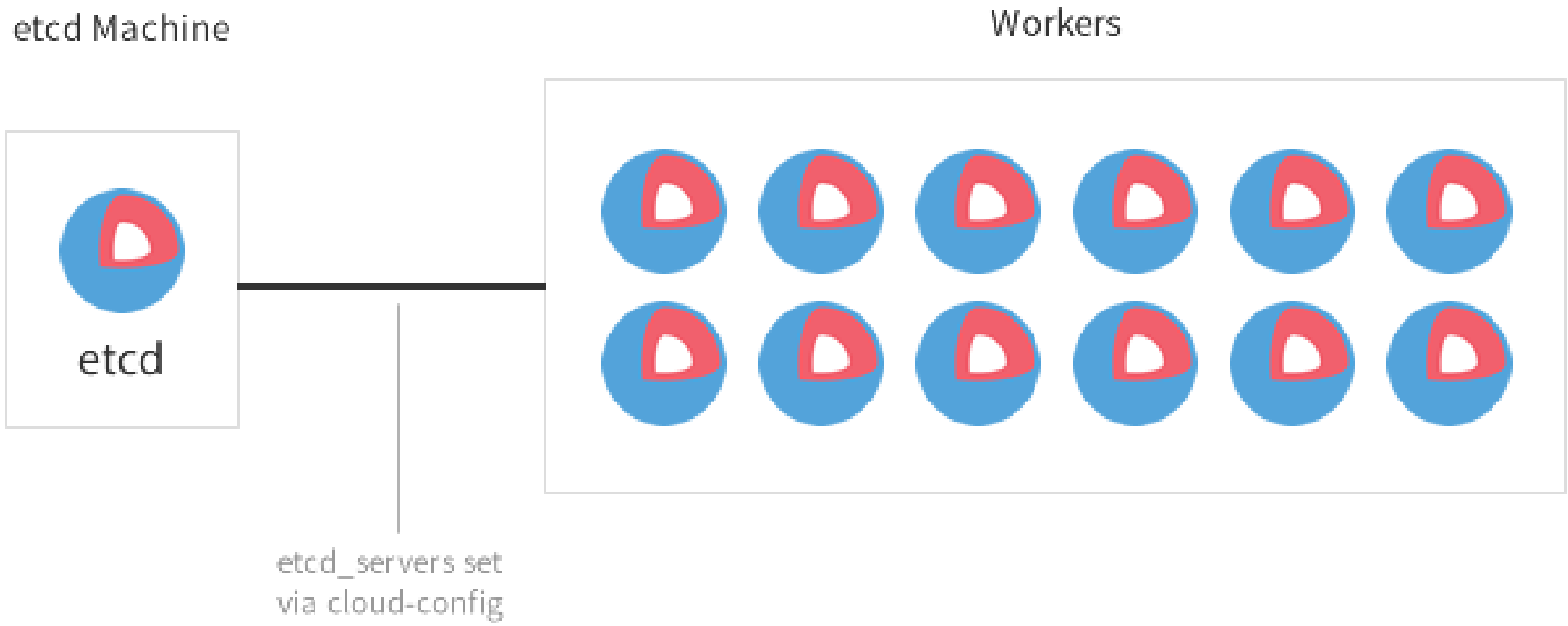
`etcd` HA key value store

- Raft election algorithm
- Writes when majority in cluster has committed update
- e.g 5 nodes, tolerates 2 nodes failure

`fleet` distributed init system (schedules systemd units in a cluster)

- Submits systemd units cluster wide
- Affinity, anti-affinity, global “scheduling”

CoreOS Cluster



“Where are you going to run coreOS ?”

“Where are you going to run Docker ?“

- *Bare metal cluster*
- *Public Clouds*
- *Private Clouds*

“How are you going to manage containers running on multiple Docker Hosts ?”

Docker schedulers

- Docker Swarm
- Citadel
- CoreOS Fleet
- Lattice from CF incubator
- Clocker (via blueprints)
- …
- **Kubernetes**

Opportunity

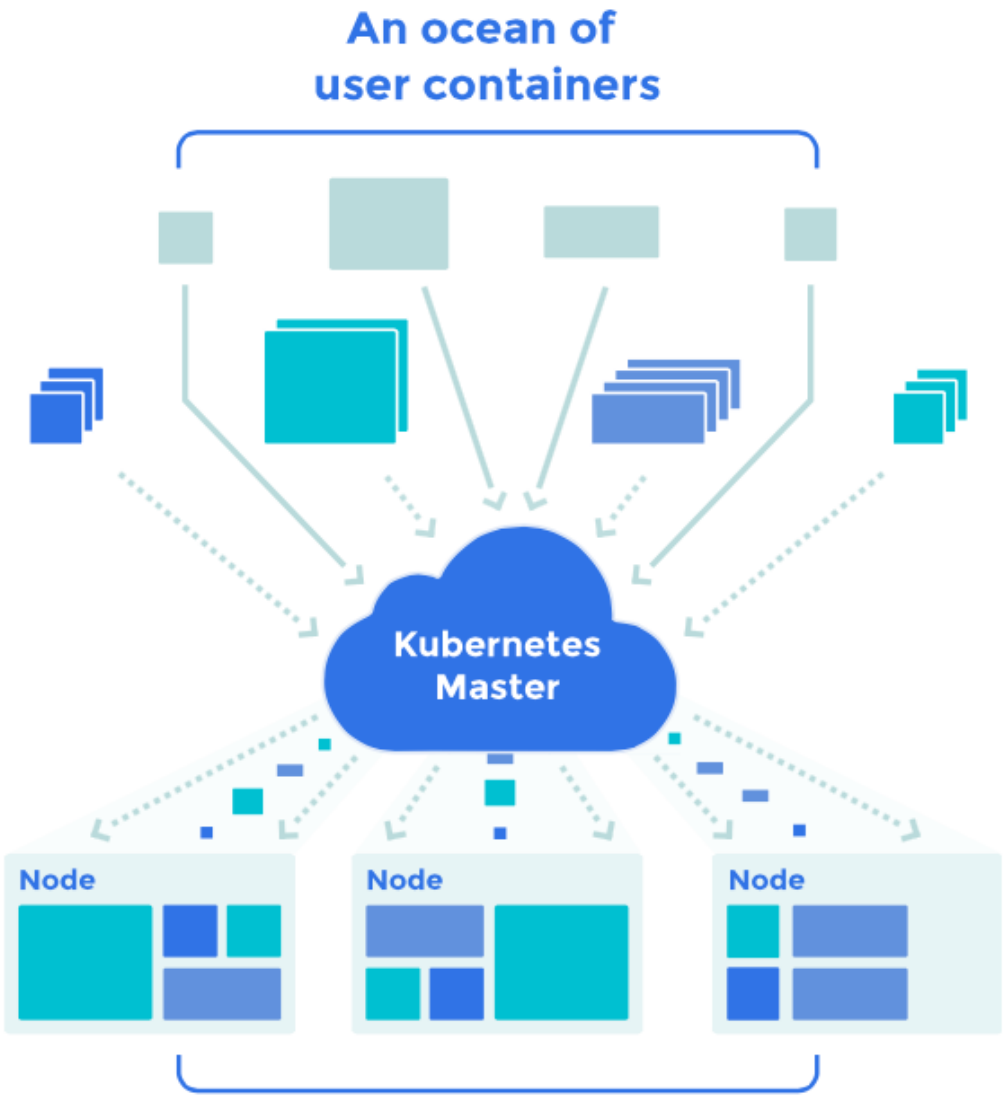
Experiment with a dedicated cluster for container based applications.

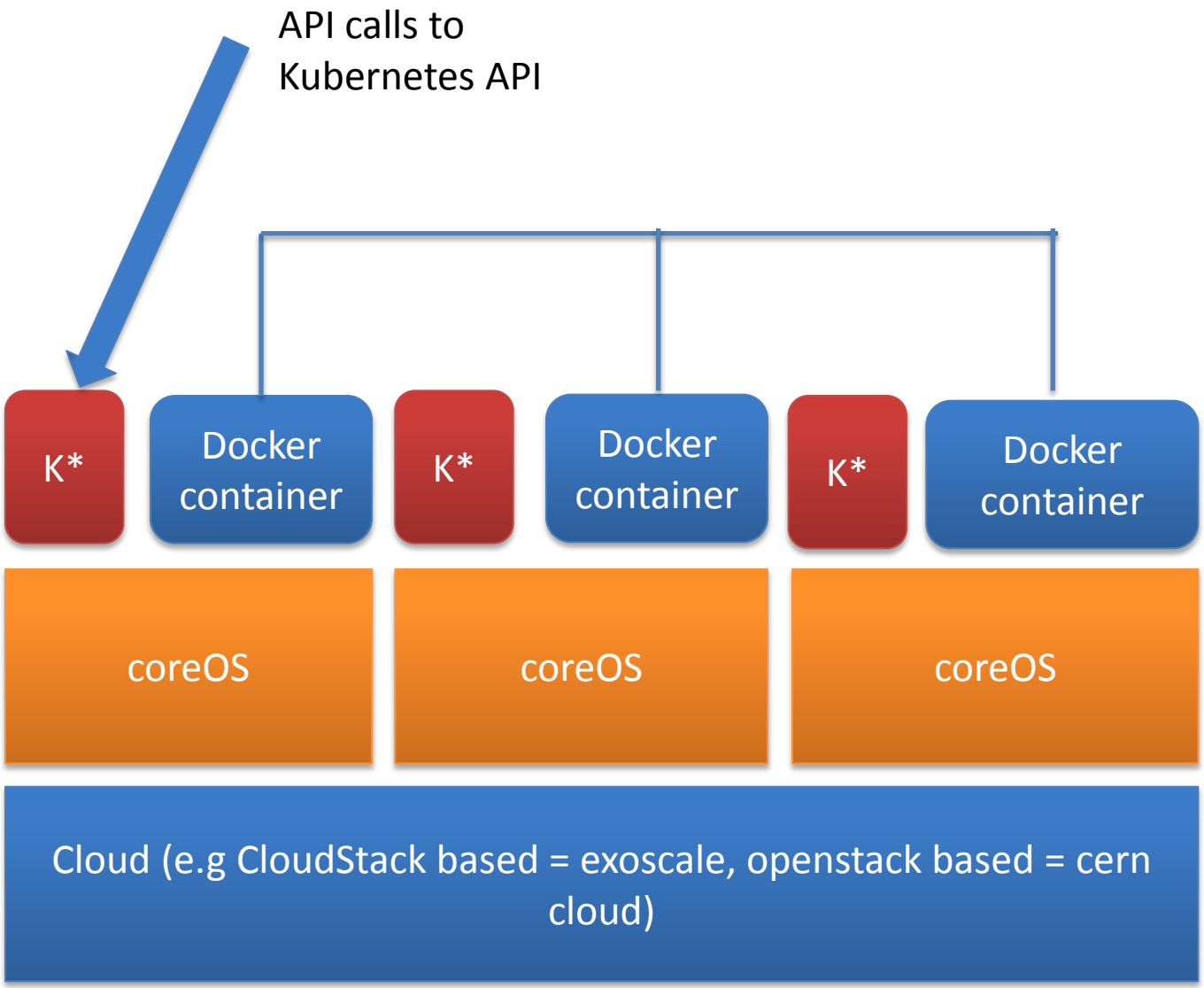
Or use a public cloud one:

Kubernetes

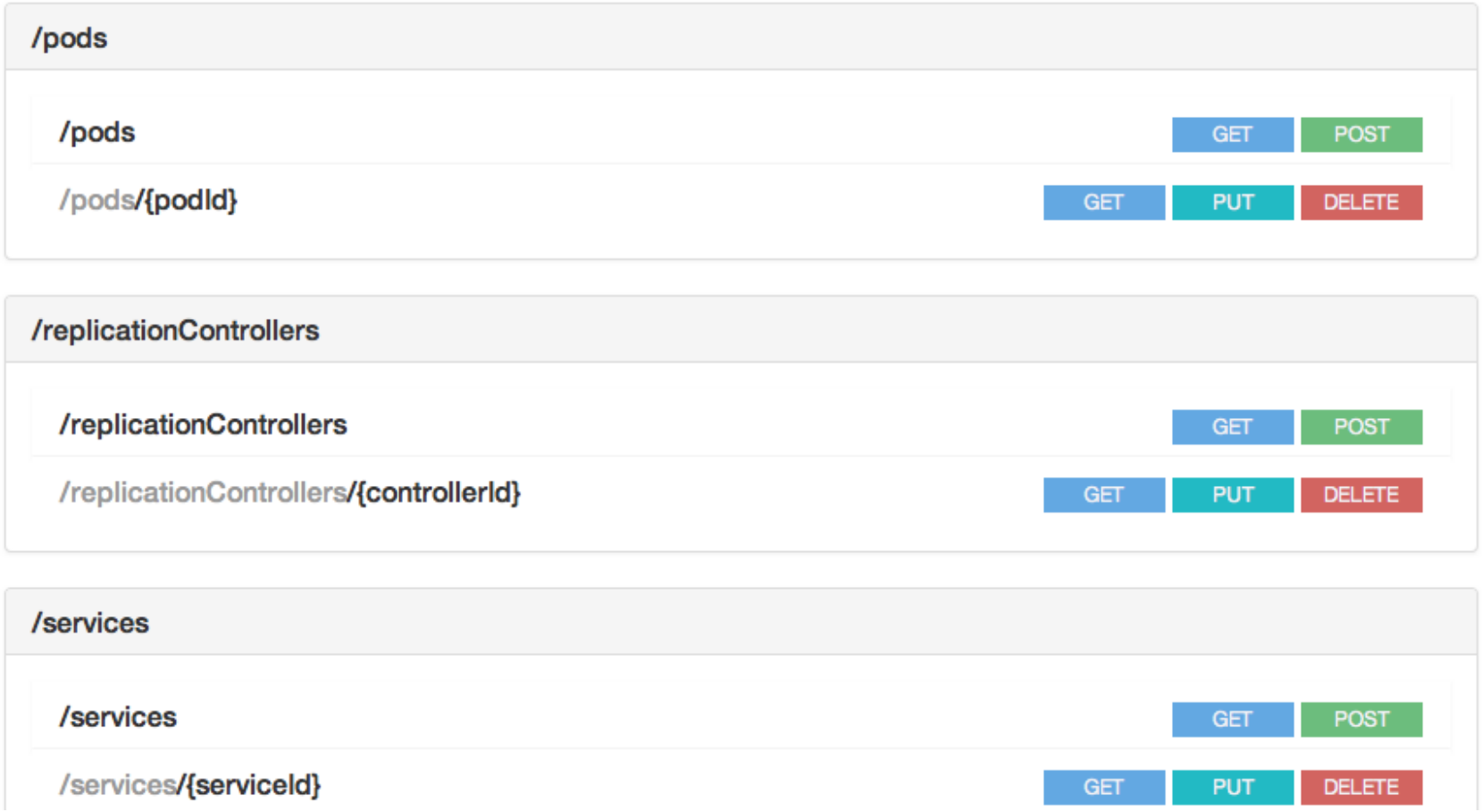
Kubernetes

- Docker application orchestration
- Google GCE, rackspace, Azure providers
- Deployable on CoreOS
- Container replication
- HA services


An ocean of
user containers
Kubernetes
Master
Node
Node
Node
Scheduled and packed
dynamically onto nodes


API calls to
Kubernetes API
K*
Docker container
K*
Docker container
K*
Docker container
coreOS
coreOS
coreOS
Cloud (e.g CloudStack based = exoscale, openstack based = cern cloud)

Kubernetes API



Kubernetes Pod

```
{
  "id": "redis-master-2",
  "kind": "Pod",
  "apiVersion": "v1beta1",
  "desiredState": {
    "manifest": {
      "version": "v1beta1",
      "id": "redis-master-2",
      "containers": [{
        "name": "master",
        "image": "dockerfile/redis",
        "ports": [{
          "containerPort": 6379,
          "hostPort": 6379
…
  "labels": {
    "name": "redis-master"
  }
}
```

Standardizing on pod

Look at the differences between:

- k8s pod
- AWS ECS task
- Ansible Docker playbook
- Fig file

?

```
- hosts: wordpress
  tasks:

  - name: Run mysql container
    docker:
      name=mysql
      image=mysql
      detach=true
      env="MYSQL_ROOT_PASSWORD=wordpressdocker,MYSQL_DATABASE=wordpress, \
          MYSQL_USER=wordpress,MYSQL_PASSWORD=wordpresspwd"

  - name: Run wordpress container
    docker:
      image=wordpress
      env="WORDPRESS_DB_NAME=wordpress,WORDPRESS_DB_USER=wordpress, \
           WORDPRESS_DB_PASSWORD=wordpresspwd"
      ports="80:80"
      detach=true
      links="mysql:mysql"
```

?

```
wordpress:
  image: wordpress
  links:
   - mysql
  ports:
   - "80:80"
  environment:
   - WORDPRESS_DB_NAME=wordpress
   - WORDPRESS_DB_USER=wordpress
   - WORDPRESS_DB_PASSWORD=wordpresspwd
mysql:
  image: mysql
  volumes:
   - /home/docker/mysql:/var/lib/mysql
  environment:
   - MYSQL_ROOT_PASSWORD=wordpressdocker
   - MYSQL_DATABASE=wordpress
   - MYSQL_USER=wordpress
   - MYSQL_PASSWORD=wordpresspwd
```

?

```
apiVersion: v1beta1
           id: wordpress
           desiredState:
             manifest:
               version: v1beta1
               id: wordpress
               containers:
                 - name: wordpress
                   image: wordpress
                   ports:
                     - containerPort: 80
                   volumeMounts:
                       # name must match the volume name below
                     - name: wordpress-persistent-storage
                       # mount path within the container
                       mountPath: /var/www/html
                   env:
                     - name: WORDPRESS_DB_PASSWORD
                       # change this - must match mysql.yaml password
                       value: yourpassword
               volumes:
                 - name: wordpress-persistent-storage
                   source:
                     # emptyDir: {}
                     persistentDisk:
                       # This GCE PD must already exist.
                       pdName: wordpress-disk
                       fsType: ext4
           labels:
             name: wpfrontend
           kind: Pod
```

?

```
[
  {
    "image": "wordpress",
    "name": "wordpress",
    "cpu": 10,
    "memory": 200,
    "essential": true,
    "links": [
      "mysql"
    ],
    "portMappings": [
      {
        "containerPort": 80,
        "hostPort": 80
      }
    ],
    "environment": [
      {
        "name": "WORDPRESS_DB_NAME",
        "value": "wordpress"
      },
     …
```

Opportunity

What type of LHC applications could take advantage of such a model ?

- Highly distributed (in the sense of many isolated functions, not X jobs)
- Long running services
- Scalable layers

Big Data

Clouds and BigData

- Object store + compute IaaS to build **EC2+S3 clone**
- BigData solutions as **storage backends** for image catalogue and large scale instance storage.
- BigData solutions as **workloads** to clouds.

EC2, S3 clone

- An open source IaaS with an EC2 wrapper e.g Opennebula, CloudStack
- Deploy a S3 compatible object store – separately- e.g riakCS
- Two independent distributed systems deployed

Cloud = EC2 + S3

Big Data

as IaaS backend

“Big Data” solutions can be used as image catalogue

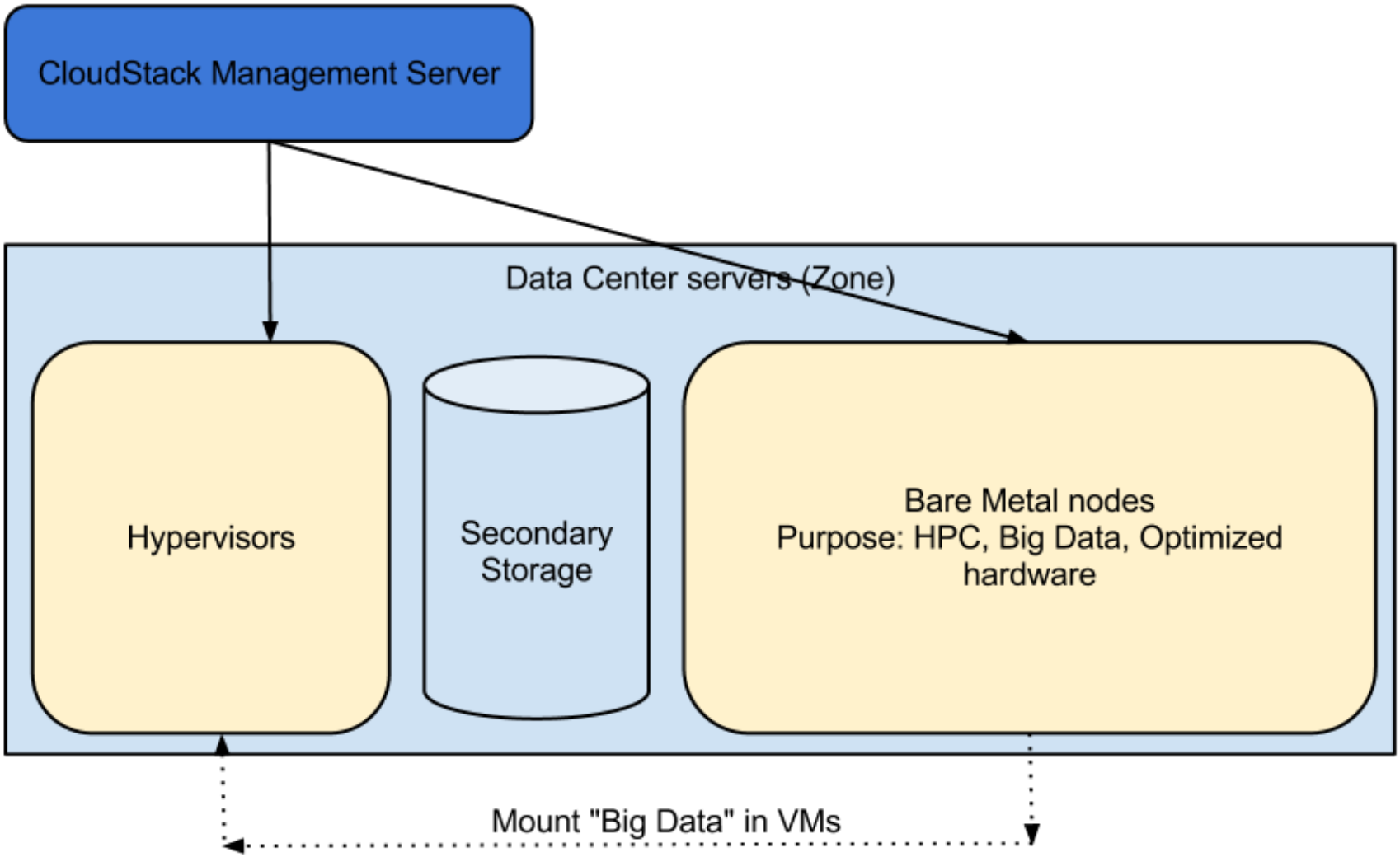
.

riakCS

openstack™

ceph

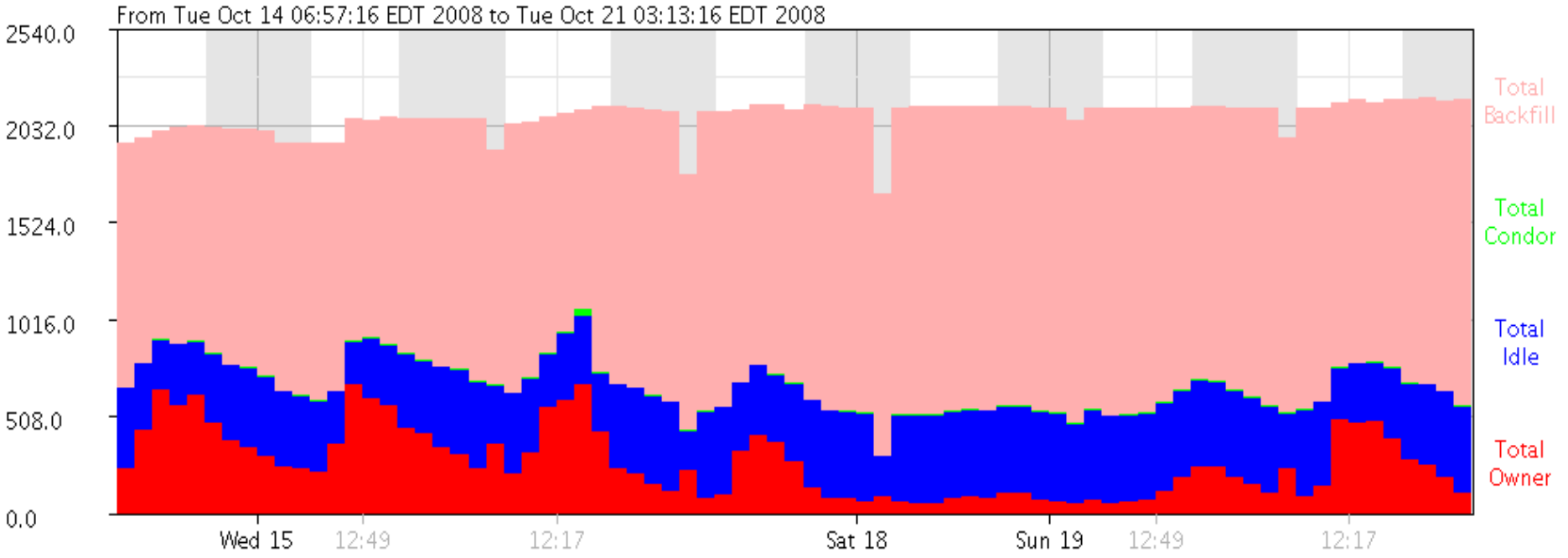
Even use Bare Metal



A note on Scheduling

- Core problem of computer science
- knapsack is NP complete
- Central scheduling has been used for a long time in HPC
- Optimizing the cluster utilization requires multi-level scheduling (e.g backfill, preemption etc..)
- Google Omega paper 2013
- Mesos 2009/2011, ASF Dec 2011

Past: BOINC/Condor Backfill



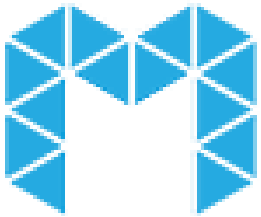
Food for thought

If Mesos is the answer…

Mesos Framework for managing VM ?

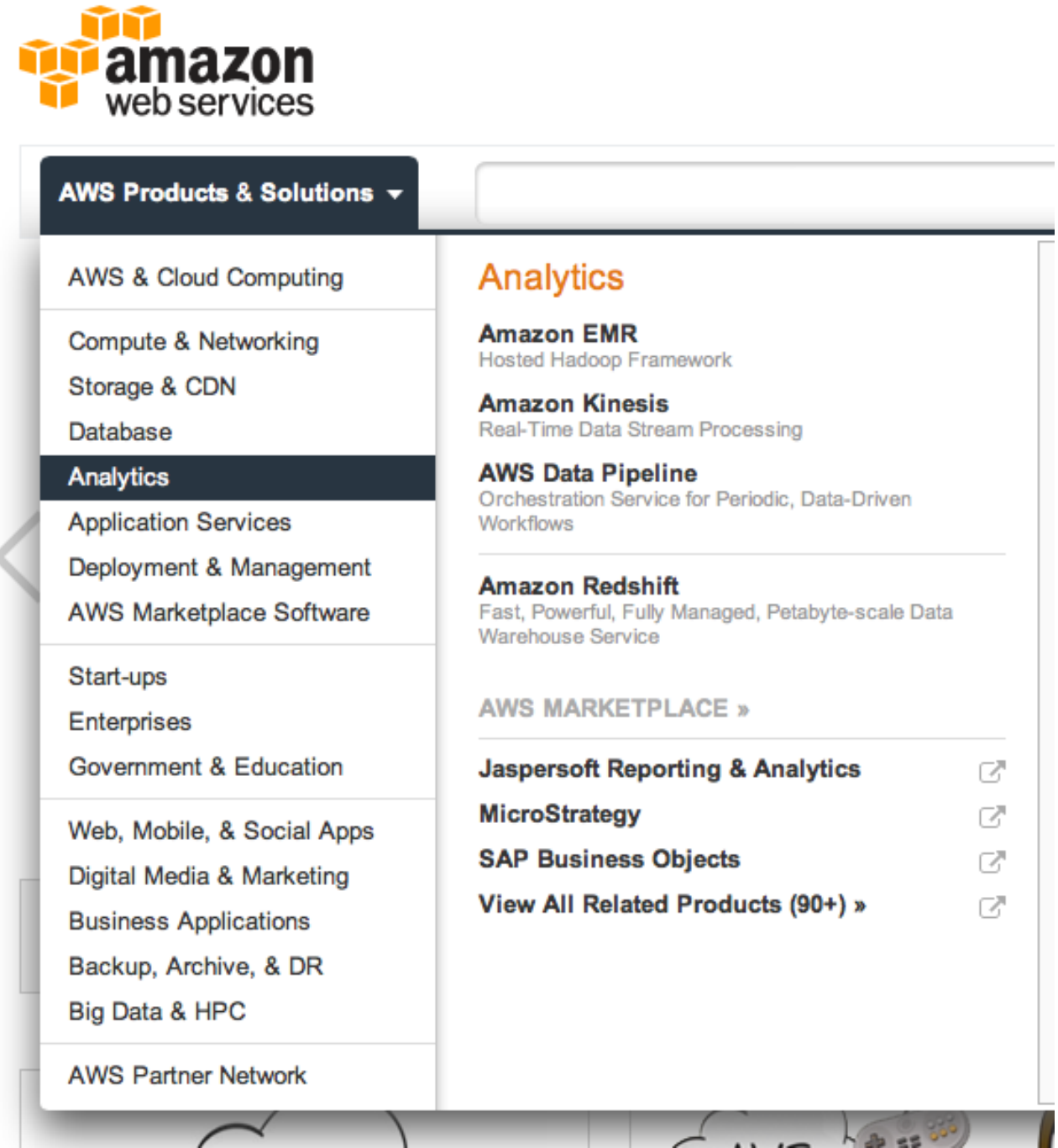
Workload sharing in your data-center:

- Big Data
- VM
- Services
- **Containers**



Conclusions

- Docker is a technology to watch to create distributed applications
- Not a replacement for VMs
- Packaging experiments applications could be challenging
- Supporting the docker networking model in the CERN environment will be difficult.
- Could Mesos be used to fill up the clusters and collocate batch and interactive services ?



Still behind !

Thanks

Web: http://sebgoa.blogspot.com

Twitter: @sebgoa