Bright Computing

Kubernetes + Ceph

Give your containers some storage, man!

Michele Bertasi
Team Lead TPSI team

Container Days - Hamburg 2017

Introduction

- Bright Computing
  - ~75 people
  - Bright cluster manager
  - Traditionally involved in HPC
  - Recently able to deploy Big data, OpenStack, Container orchestrators
- Myself
  - Team lead for TPSI team
  - Containers, Ceph, WLM
  - I come from C++ development in industrial automation

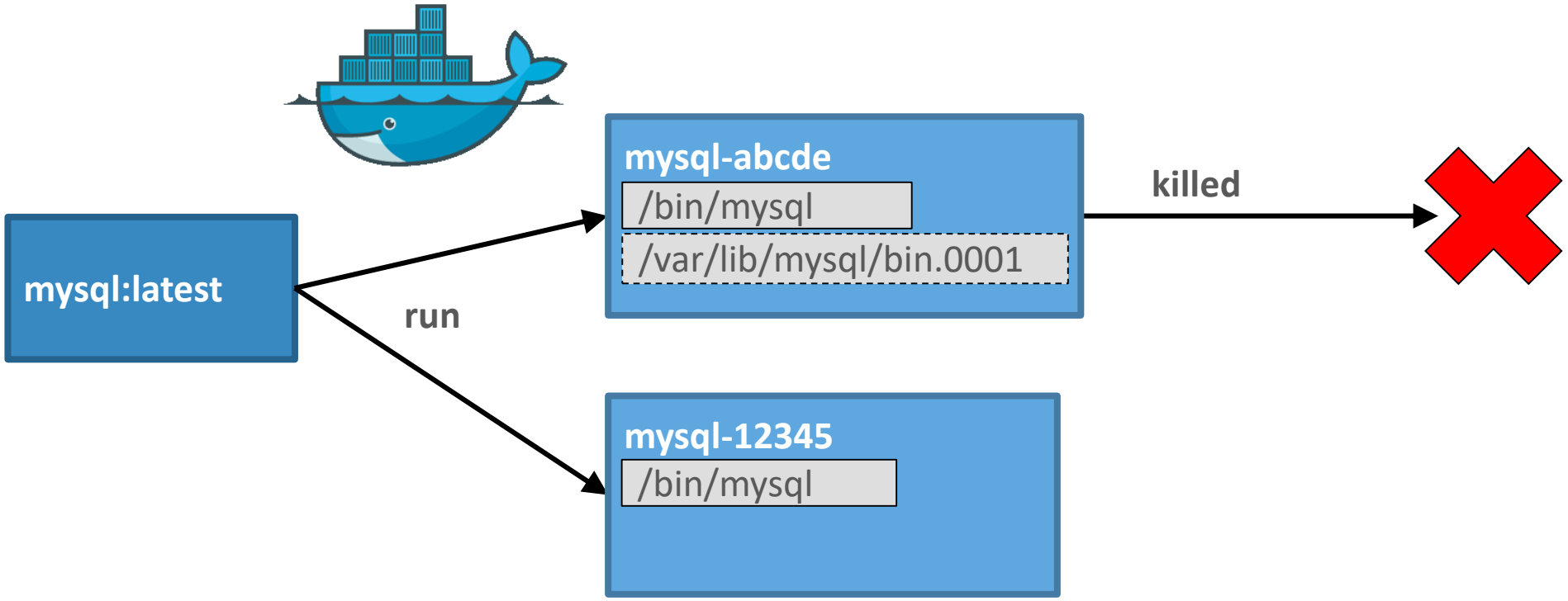
Kubernetes

Kubernetes

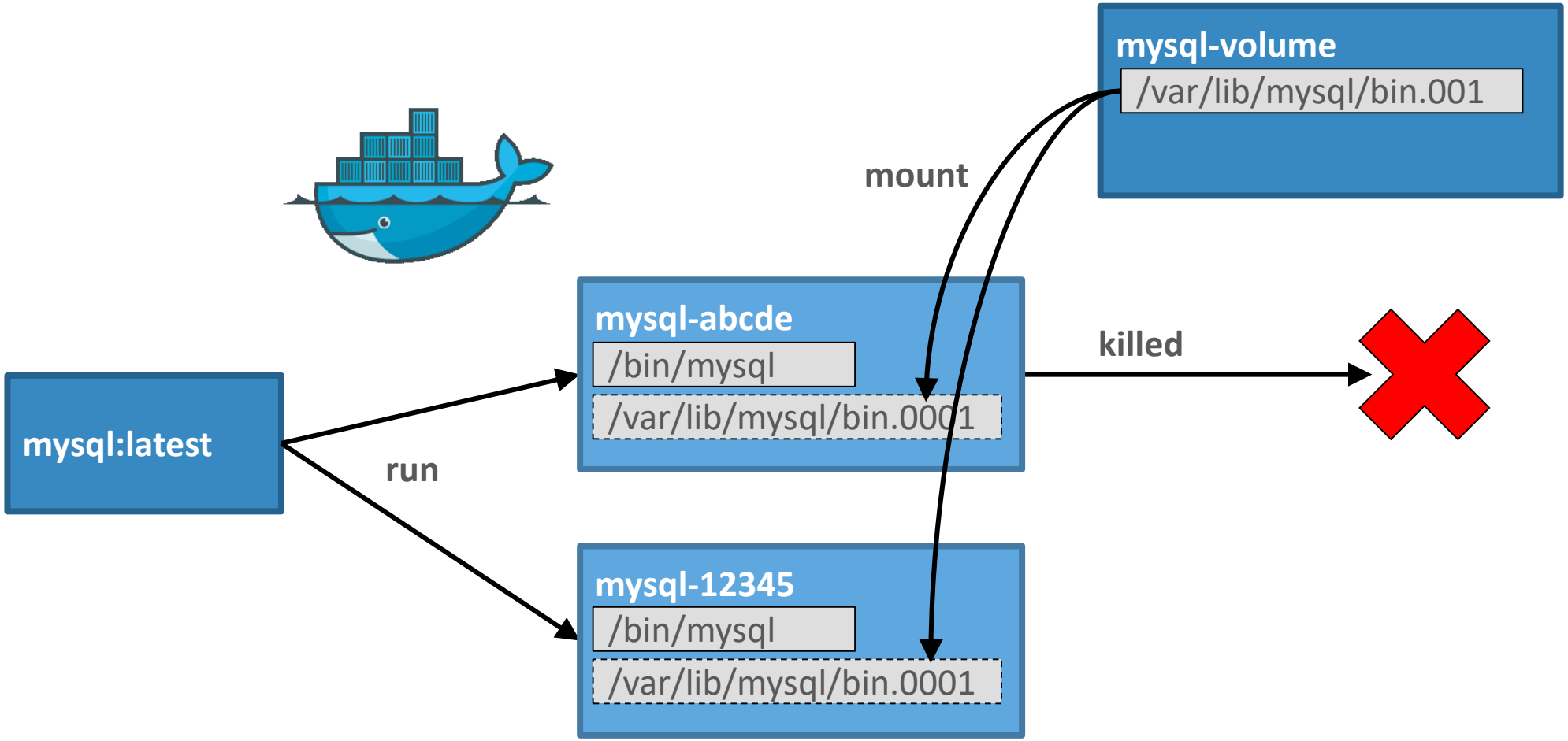
- Kubernetes is becoming de-facto standard in cloud native environments
  - In public cloud
  - In private cloud
  - Bare metal
- How does Kubernetes manage storage?
- Where do you put your storage?

The storage challenge

- A container is ephemeral
- The files on its disk are cleared when the container is restarted



Introducing volumes



Volumes in Kubernetes

- The volume is specified directly in the pod definition
  - The lifetime is the same as the pod one

↓

so it survives container restarts

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: gcr.io/google_containers/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

Decoupling storage and compute (I)

- Most of the times, cluster administrators and cluster users are different people (or teams)
  - Their responsibilities are different
  - Specifying volumes in pod definitions couple these two roles strongly
  - Enforcing policies by cluster admins is not possible

Decoupling storage and compute (II)

- The solution is to separate storage requests and storage offers
- Abstract details of how storage is provided from how it is consumed
- Kubernetes offers two API resources for that:
  - **PersistentVolumeClaim** → request
  - **PersistentVolume** → offer

Cluster administrator

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gce-disk-1
spec:
  capacity:
    storage: 10Gi
  accessModes:
  - ReadWriteOnce
  gcePersistentDisk:
    fsType: ext4
    pdName: gce-disk-1
```

- A PersistentVolume maps a piece of storage in the infrastructure
- The admin can create volumes of different types
- They can tweak the parameters as they like
- They are always in control of the allocated storage

Cluster user

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pv-claim
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: gcr.io/google_containers/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    persistentVolumeClaim:
      claimName: task-pv-claim
```

PersistentVolumes manual management

- Creating volumes manually is tedious
  - The admin has to:
    - create the storage manually in their infrastructure
    - and the corresponding object in Kubernetes
  - It requires communication between users and admins every time a new volume is required
  - The users might need to resort to do this job themselves (no more separation of concerns)

StorageClass

- Allows to create PersistentVolumes on demand
- No more manual creation for cluster admins
- Allows to expose multiple flavors, by specifying parameters to the provisioners
- The users are able to choose among different options without having to know the details
- A volume is then reclaimed when the PVC is gone
- Various reclaim policies (retain, recycle, delete)
- Default class

StorageClass

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: slow
  labels:
    release: stable
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
```

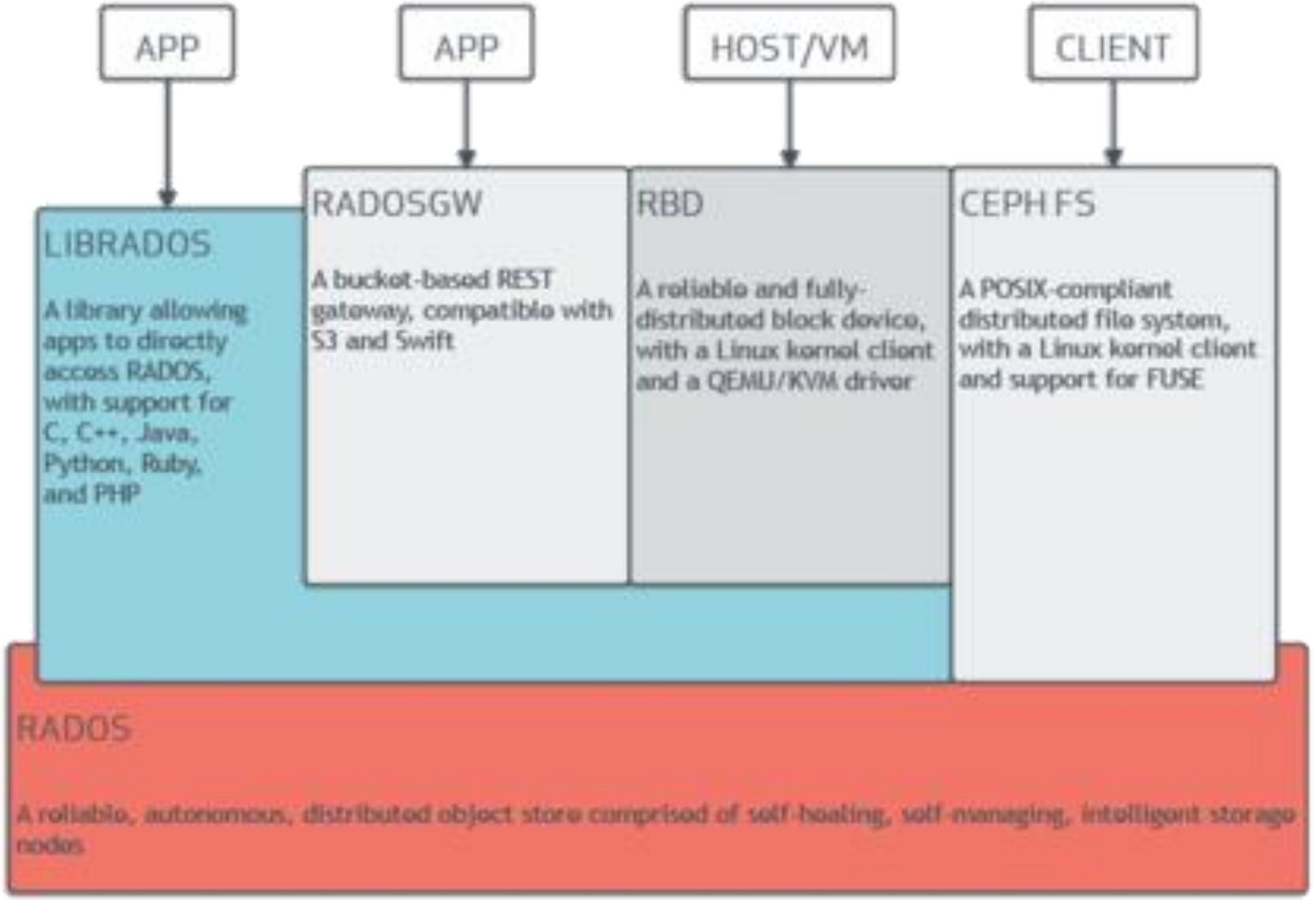
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
  storageClassName: slow
  selector:
    matchLabels:
      release: stable
```

Going on-premise

- The previous example involved public cloud
- If on-premise
  - we need to setup our own storage
  - we need to expose it to Kubernetes
- Multiple possibilities, but we need:
  - Distributed storage
  - Reliable
  - (at minimum)

Ceph

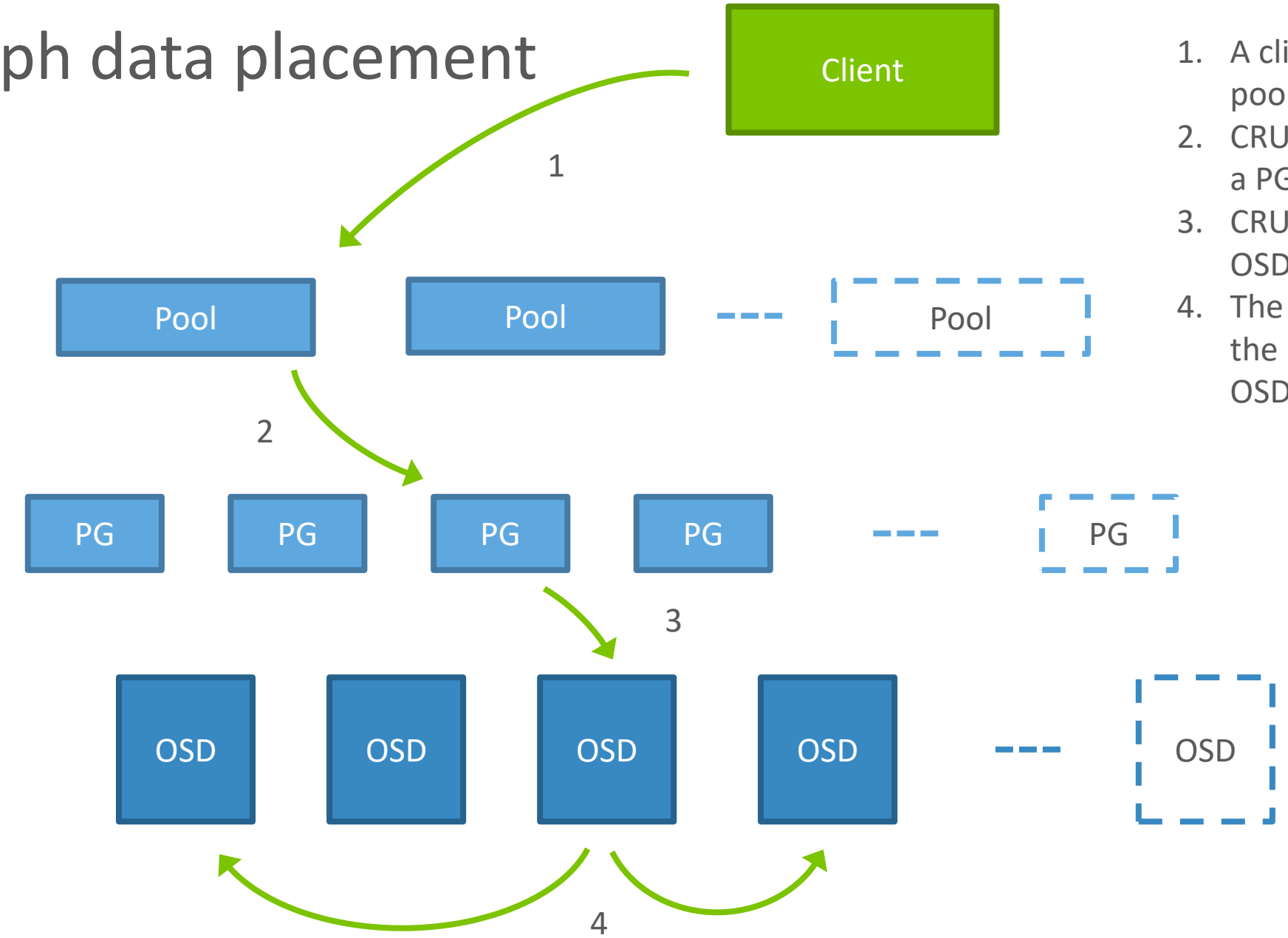
What is Ceph?



Ceph nodes

- OSD nodes
  - store the data
  - multiple OSD daemons per node
- Monitor nodes
  - keep track of the state of the cluster
  - allow clients to retrieve the cluster map
- Optional metadata servers
  - file storage for CephFS

Ceph data placement



1. A client writes an object to a pool
2. CRUSH assigns the object to a PG
3. CRUSH assigns the PG to an OSD
4. The primary OSD replicates the PG to the secondary OSDs

Setup Ceph for Kubernetes

- We have to create a pool
  - A logical group for storing objects
  - A pool has its own set of parameters
- Authentications
  - An admin auth, able to create images in the pool
  - A client auth, able to write the images

Ceph RBD storage class

```
kubectl create secret generic \
  ceph-secret --type=kubernetes.io/rbd \
  --from-literal-key='<key>' \
  --namespace=kube-system
```

- We need
  - One admin secret
  - User secrets (one per namespace)

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: fast
provisioner: kubernetes.io/rbd
parameters:
  monitors: 10.16.153.105:6789
  adminId: kube
  adminSecretName: ceph-secret
  adminSecretNamespace: kube-system
  pool: kube
  userId: kube
  userSecretName: ceph-secret-user
```

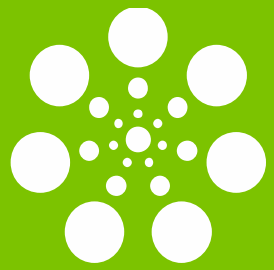
Demo

Conclusion

- Kubernetes is able to manage containers and storage resources of a cluster automatically
- Separation of concerns between cluster administrators and cluster users with **StorageClass**, **PersistentVolumeClaim.**
  - Cluster admins can tweak the storage parameters
  - Cluster users can ask for storage without to know the details
- Ceph provides a distributed, reliable storage for Kubernetes
  - Production ready
  - You can reuse it for other purposes, like OpenStack

References

- Demo files in my GitHub: https://github.com/mbrt/kube-ceph-demo
- Ceph
  - http://docs.ceph.com/docs/master/rados/operations/pools/
  - http://docs.ceph.com/docs/master/start/hardware-recommendations/
- Kubernetes
  - https://kubernetes.io/docs/concepts/storage/persistent-volumes/



Bright Computing