KubeCon | CloudNativeCon

Europe 2018

Kubernetes Storage Lingo 101

Saad Ali
Senior Software Engineer, Google

May 4, 2018

Agenda

- What do these words mean and how do they fit together?

Persistent Volume Claims **Driver** **Persistent Volumes**

Remote **File** **Flex** **Block** **CSI** **Stateless**

Storage Classes **Ephemeral** **Local** **Out-of-tree**

Dynamic Provisioning **In-tree** **Volume** **Object**

Stateful **Plugin**

Kubernetes Principle
Workload portability

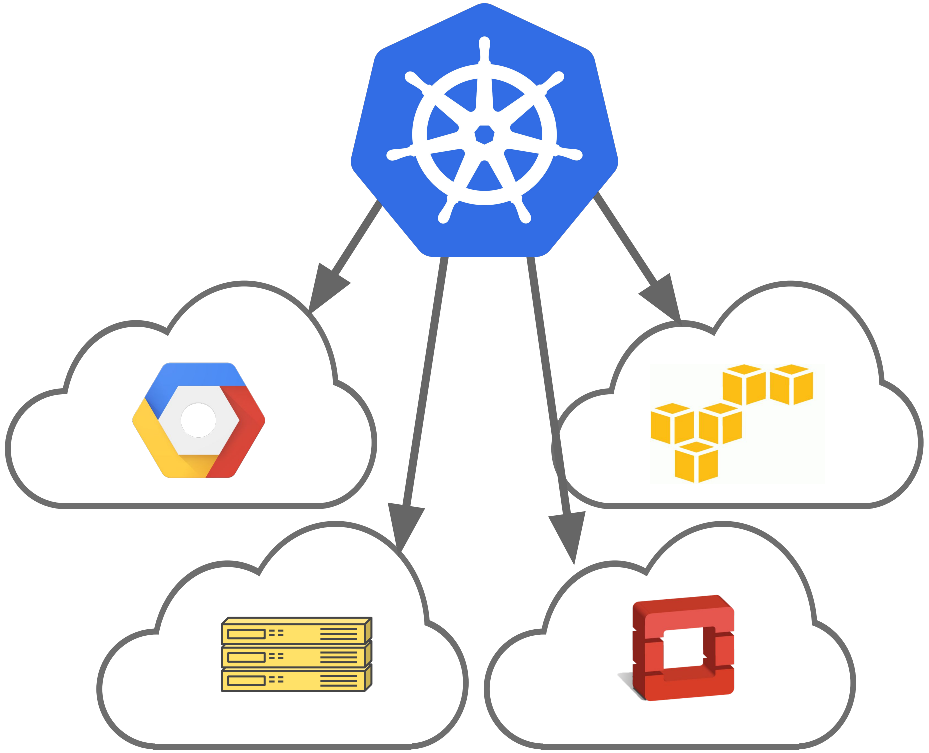
Kubernetes: Workload Portability

Kubernetes Goal

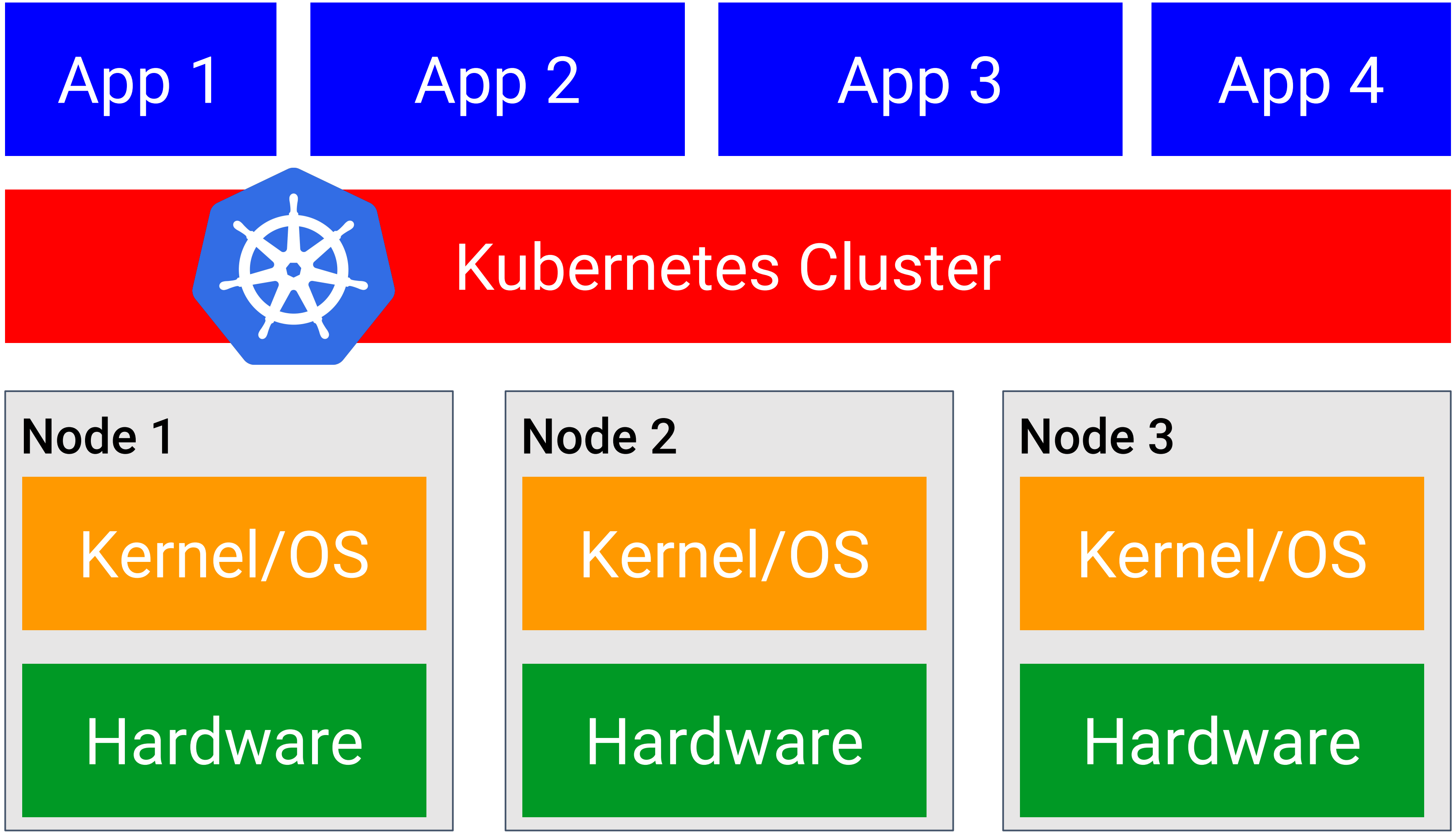
- Abstract away cluster details
- Decouple apps from infrastructure

To enable users to

- Write once, run anywhere (workload portability!)
- Avoid vendor lock-in



Kubernetes



App 1

App 2

App 3

App 4

Kubernetes Cluster

Node 1

Kernel/OS

Hardware

Node 2

Kernel/OS

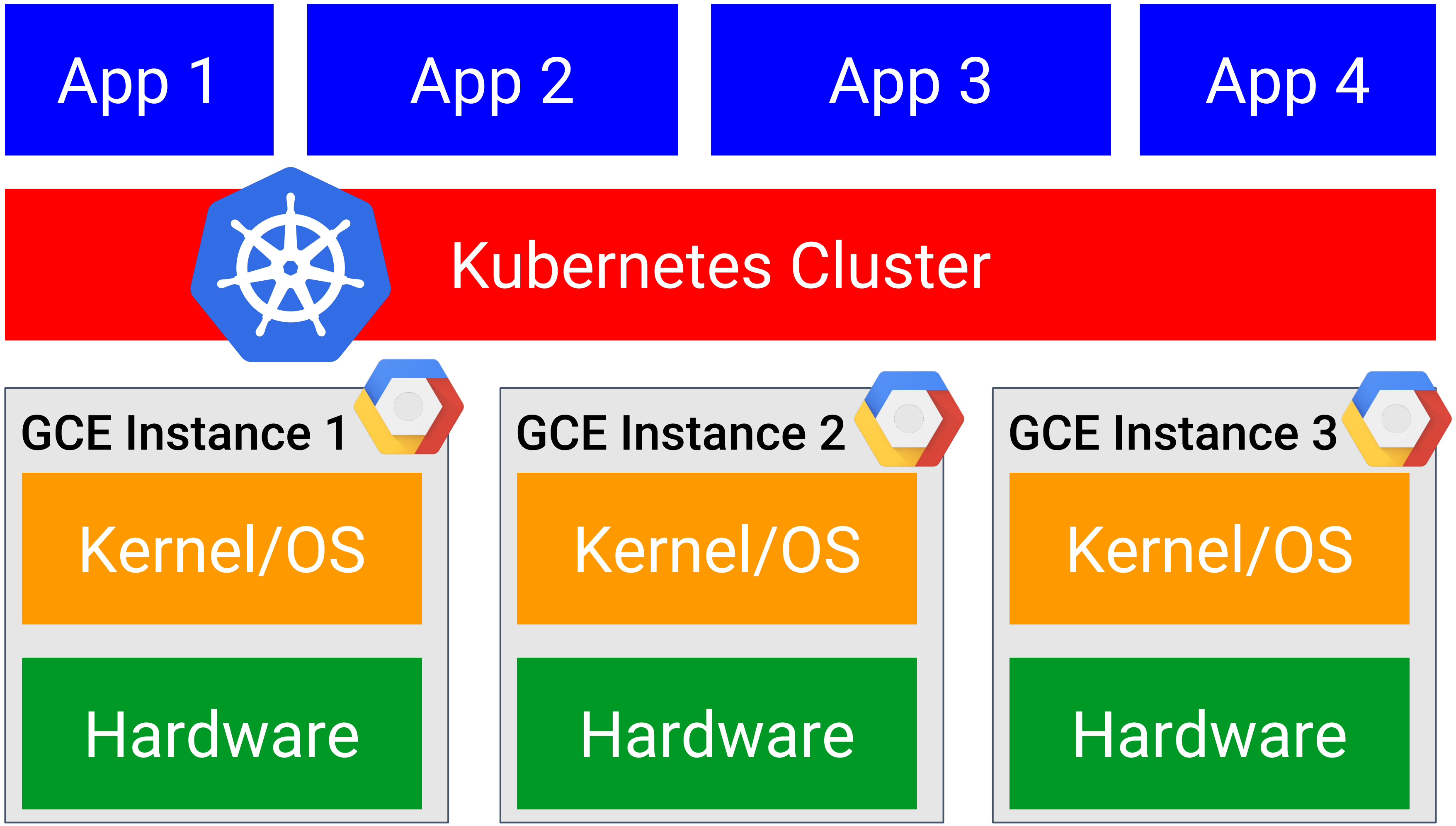
Hardware

Node 3

Kernel/OS

Hardware

Kubernetes



App 1

App 2

App 3

App 4

Kubernetes Cluster

GCE Instance 1

Kernel/OS

Hardware

GCE Instance 2

Kernel/OS

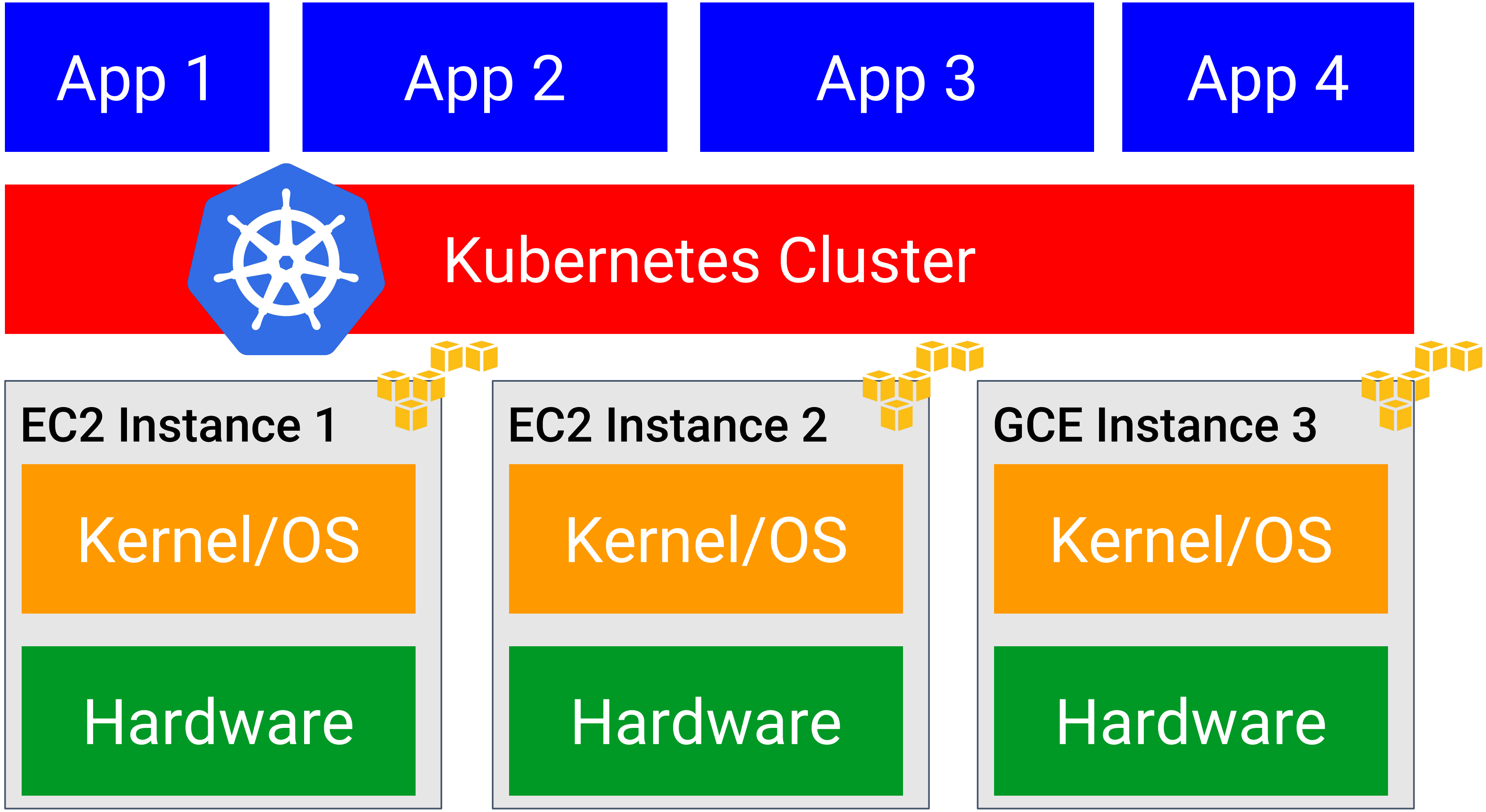
Hardware

GCE Instance 3

Kernel/OS

Hardware

Kubernetes



App 1

App 2

App 3

App 4

Kubernetes Cluster

EC2 Instance 1

Kernel/OS

Hardware

EC2 Instance 2

Kernel/OS

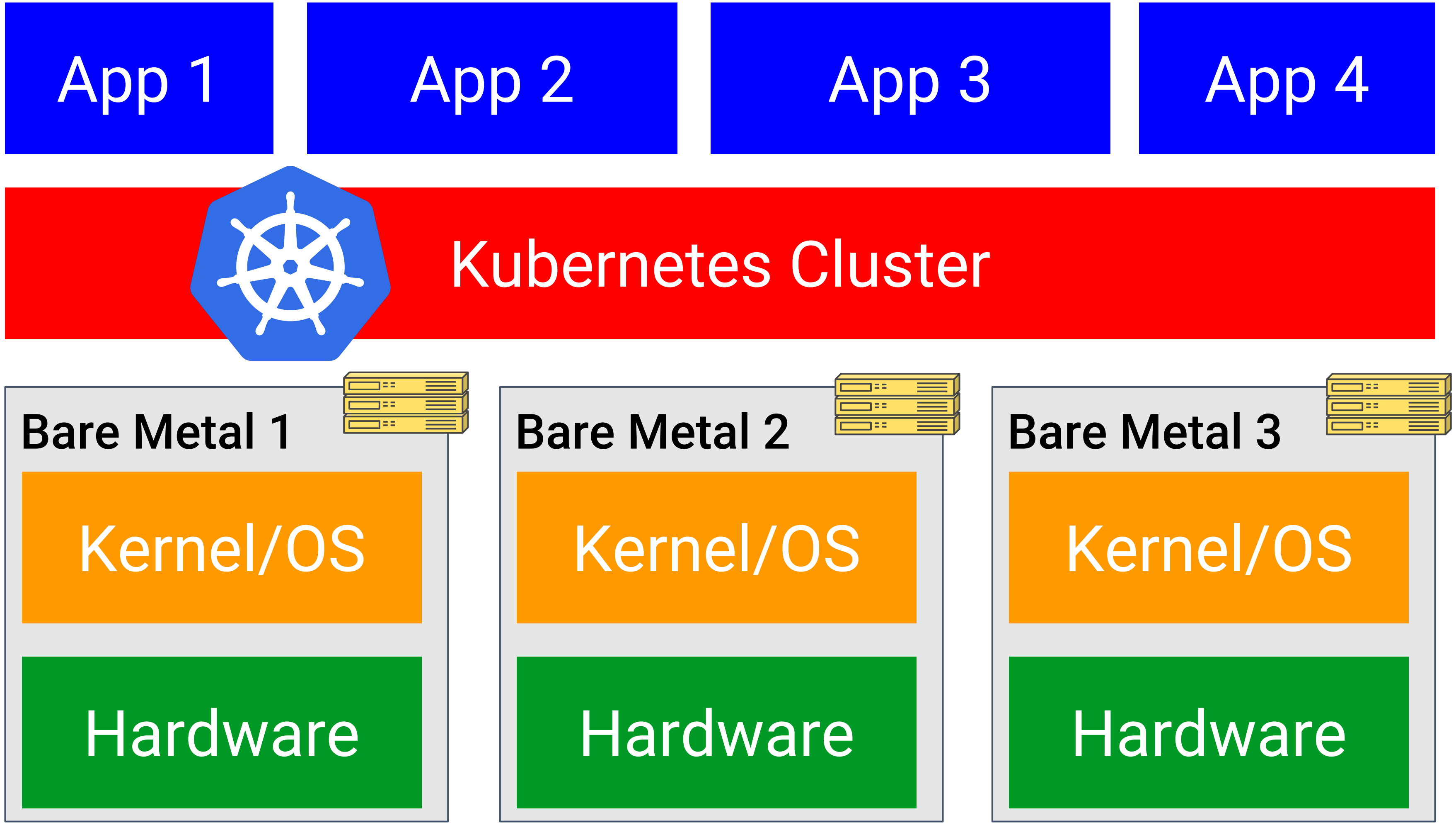
Hardware

GCE Instance 3

Kernel/OS

Hardware

Kubernetes



App 1

App 2

App 3

App 4

Kubernetes Cluster

Bare Metal 1

Kernel/OS

Hardware

Bare Metal 2

Kernel/OS

Hardware

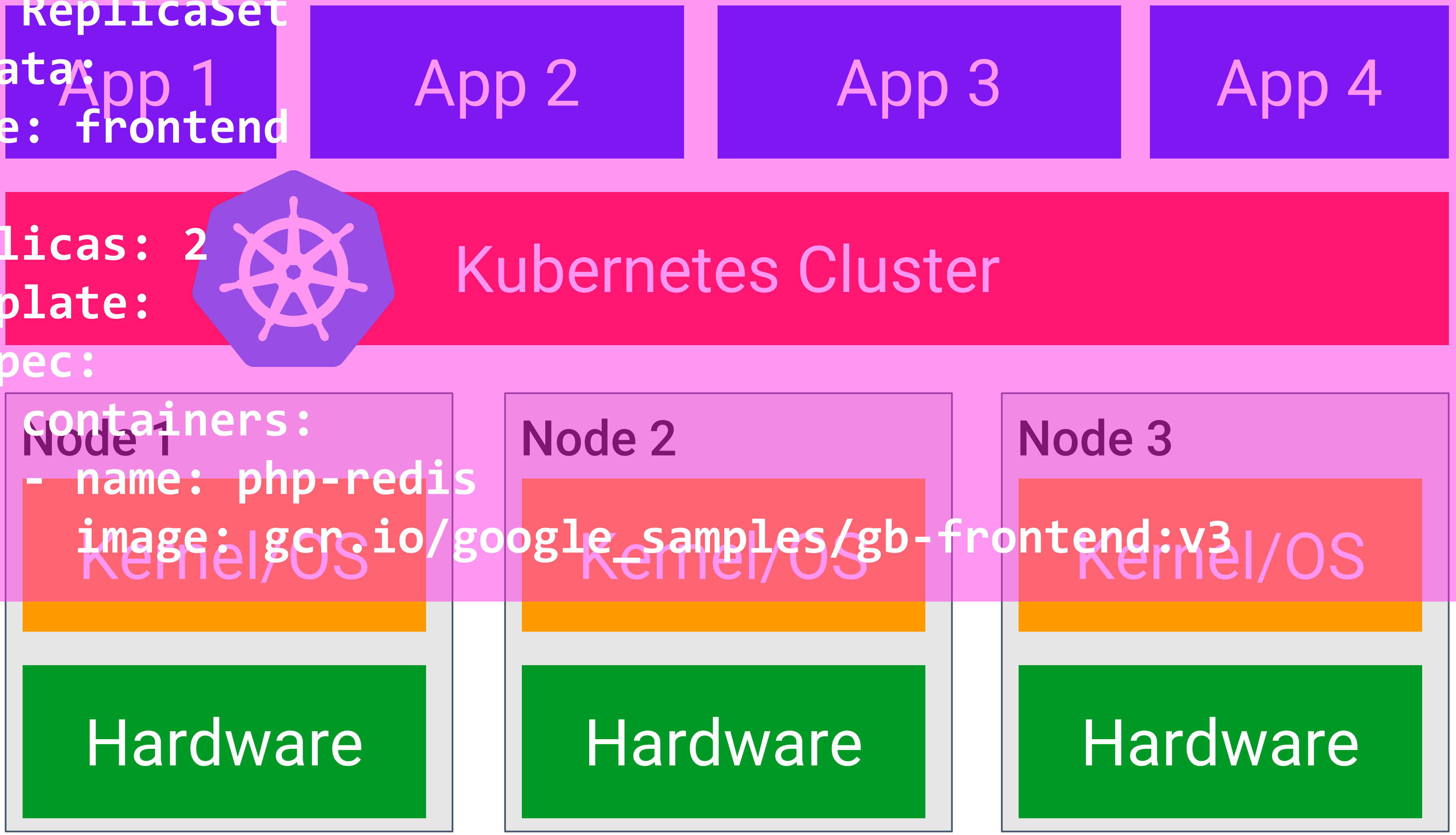
Bare Metal 3

Kernel/OS

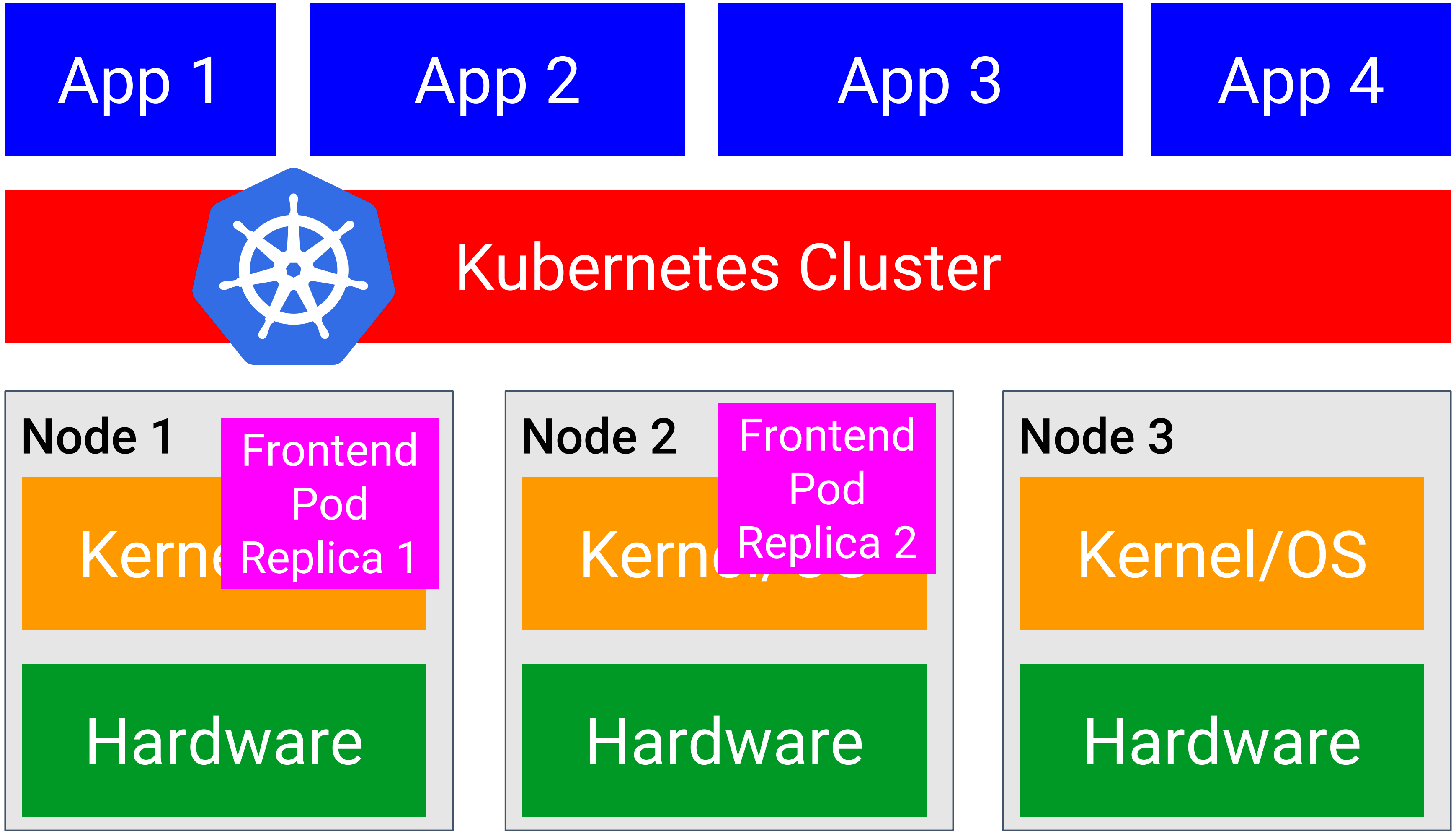
Hardware

Kubernetes

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
spec:
  replicas: 2
  template:
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
```



Kubernetes

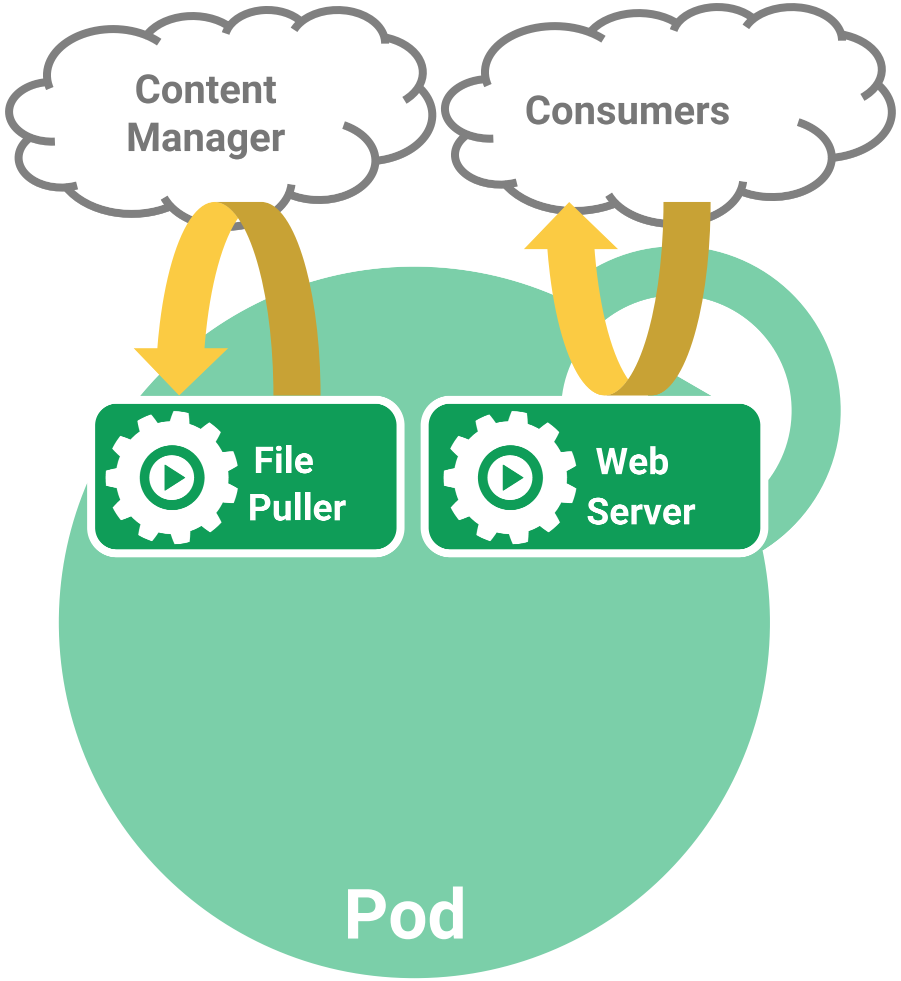

App 1
App 2
App 3
App 4
Kubernetes Cluster
Node 1
Frontend Pod Replica 1
Kernel/OS
Hardware
Node 2
Frontend Pod Replica 2
Kernel/OS
Hardware
Node 3
Kernel/OS
Hardware

Problem with Containers and State

What about stateful apps?

Pod and ReplicaSet abstract compute and memory.

1. Containers are ephemeral: no way to persist state
   - Container termination/crashes result in loss of data
   - Can’t run stateful applications
2. Containers can’t share data between each other.



Challenges with Abstracting Storage

- So many different types of storage
  - Object Stores
    - AWS S3, GCE GCS, etc.
  - SQL Databases
    - MySQL, SQL Server, Postgres, etc.
  - NoSQL Databases
    - MongoDB, ElasticSearch, etc.
  - Pub Sub Systems
    - Apache Kafka, Google Cloud Pub/Sub, AWS SNS, etc.
  - Time series databases
    - InfluxDB, Graphite, etc.
  - File Storage
    - NFS, SMB, etc.
  - Block Storage
    - GCE PD, AWS EBS, iSCSI, Fibre Channel, etc.
  - File on Block Storage
  - And more!
- What do we focus on?

Kubernetes Principle

Workload portability

What do we focus on?

In scope:

- File Storage
  - NFS, SMB, etc.
- Block Storage
  - GCE PD, AWS EBS, iSCSI, Fibre Channel, etc.
- File on Block Storage

Out of scope:

- Object Stores
  - AWS S3, GCE GCS, etc.
- SQL Databases
  - MySQL, SQL Server, Postgres, etc.
- NoSQL Databases
  - MongoDB, ElasticSearch, etc.
- Pub Sub Systems
  - Apache Kafka, Google Cloud Pub/Sub, AWS SNS, etc.
- Time series databases
  - InfluxDB, Graphite, etc.
- etc.

What do we focus on?

In scope:

- File Storage
  - NFS, SMB, etc.
- Block Storage
  - GCE PD, AWS EBS, iSCSI, Fibre Channel, etc.
- File on Block Storage

Data Path Standardized (Posix, SCSI)

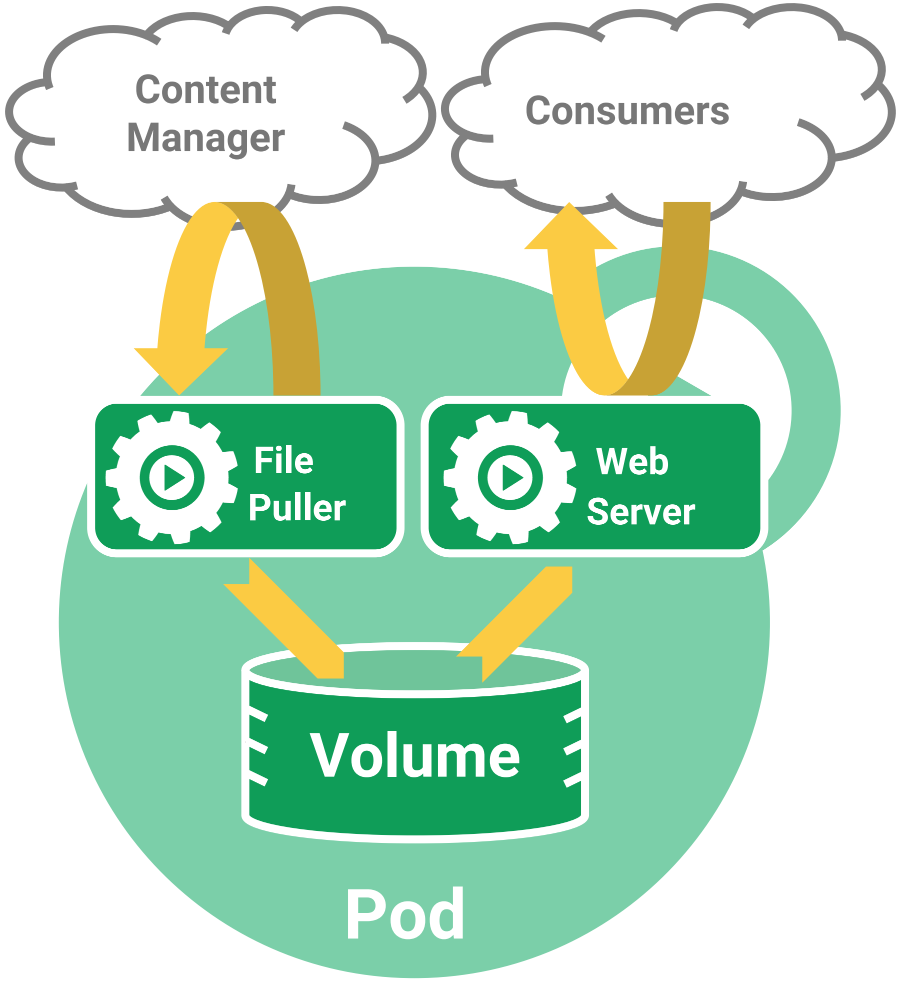
Out of scope:

- Object Stores
  - AWS S3, GCE GCS, etc.
- SQL Databases
  - MySQL, SQL Server, Postgre, etc.
- NoSQL Databases
  - MongoDB, ElasticSearch, etc.
- Pub Sub Systems
  - Apache Kafka, Google Cloud Pub/Sub, AWS SNS, etc.
- Time series databases
  - InfluxDB, Graphite, etc.
- etc.

Data Path Not Standardized, yet

Kubernetes Volume Plugins

- A way to reference **block device** or **mounted filesystem** (possibly with some data in it)
- Accessible by all containers in pod
- Volume plugins specify
  - How volume is setup in pod
  - Medium that backs it
- Lifetime of volume is same as the pod or longer



Kubernetes Volume Plugins

Kubernetes has many volume plugins

Remote Storage

- GCE Persistent Disk
- AWS Elastic Block Store
- Azure File Storage
- Azure Data Disk
- Dell EMC ScaleIO
- iSCSI
- Flocker
- NFS
- vSphere
- GlusterFS
- Ceph File and RBD
- Cinder
- Quobyte Volume
- FibreChannel
- VMware Photon PD

Ephemeral Storage

- EmptyDir
- Expose Kubernetes API
  - Secret
  - ConfigMap
  - DownwardAPI

Local Persistent Volume (Beta)

Out-of-Tree

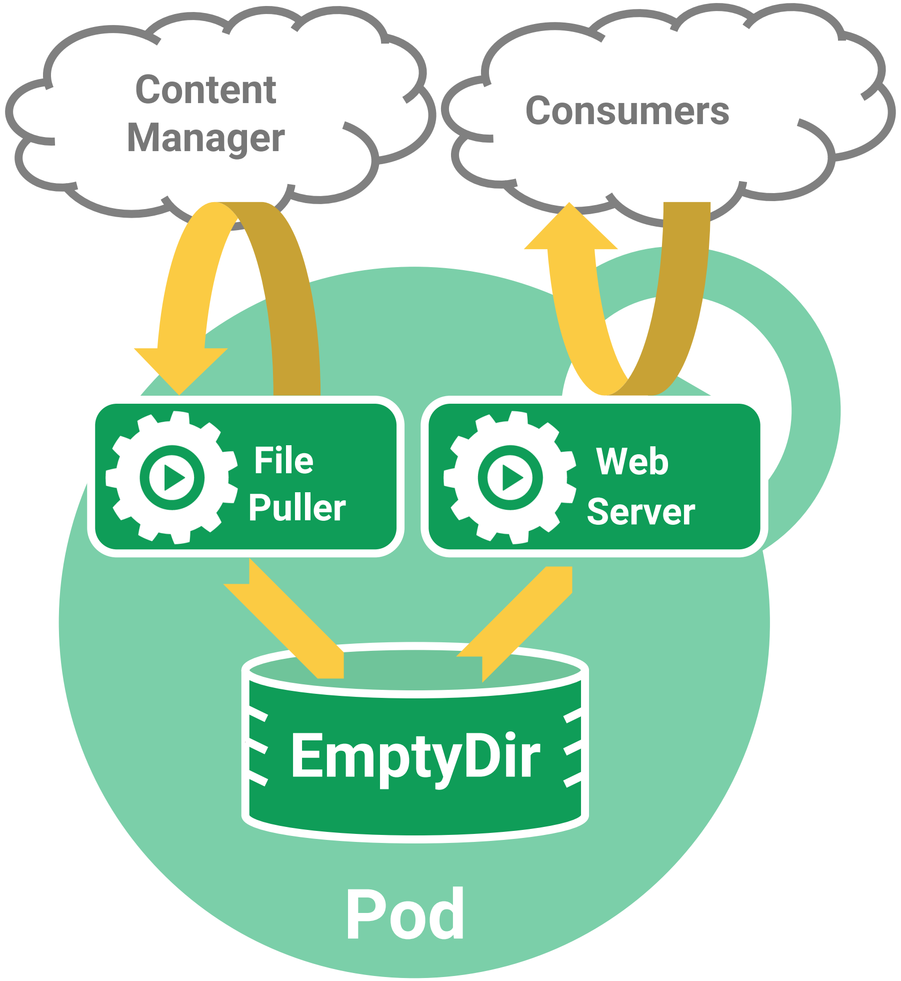
- Flex (exec a binary)
- CSI (Beta)

Other

- Host path

Ephemeral Storage

- Temp scratch file space from host machine
- Data exists only for lifecycle of pod.
- Can only be referenced “in-line” in pod definition not via PV/PVC.
- Volume Plugin: EmptyDir



Ephemeral Storage

- Temp scratch file space from host machine
- Data exists only for lifecycle of pod.
- Can only be referenced “in-line” in pod definition not via PV/PVC.
- Volume Plugin: EmptyDir

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
  - image: k8s.gcr.io/container1
    name: container1
    volumeMounts:
    - mountPath: /shared
      name: shared-scratch-space
  - image: k8s.gcr.io/container2
    name: container2
    volumeMounts:
    - mountPath: /shared
      name: shared-scratch-space
  volumes:
  - name: shared-scratch-space
    emptyDir: {}
```

Kubernetes Principle

Workload portability

Ephemeral Storage

- Built on top of EmptyDir:
  - Secret Volume
  - ConfigMap Volume
  - DownwardAPI Volume
- Populate Kubernetes API as files in to an EmptyDir

Kubernetes Principle

Meet the user where they are

Ephemeral Storage

- Built on top of EmptyDir:
  - Secret Volume
  - ConfigMap Volume
  - DownwardAPI Volume
- Populate Kubernetes API as files in to an EmptyDir

Remote Storage

- Data persists beyond lifecycle of any pod
- Examples:
  - GCE Persistent Disk
  - AWS Elastic Block Store
  - Azure Data Disk
  - iSCSI
  - NFS
  - GlusterFS
  - Cinder
  - Ceph File and RBD
  - And more!
- Referenced in pod either in-line or via PV/PVC

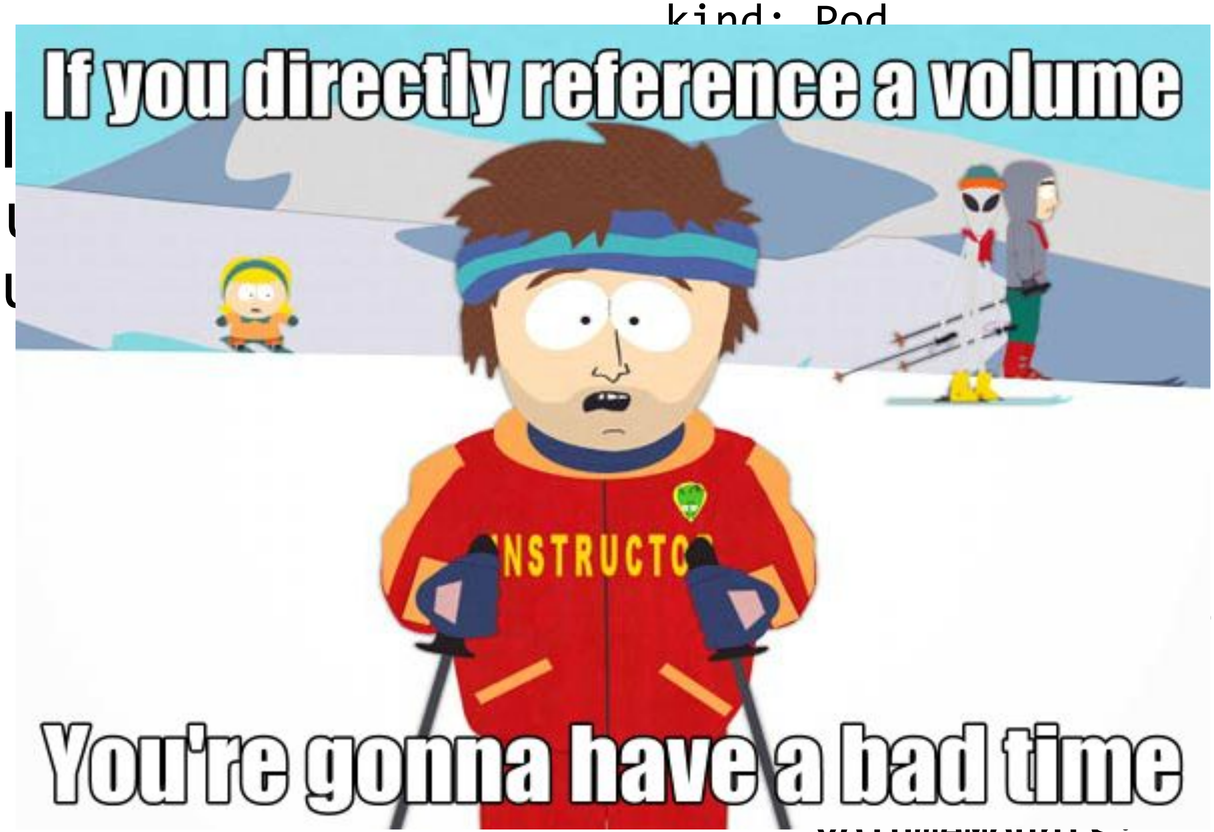
Remote Storage

- Kubernetes will automatically:
  - Attach volume to node
  - Mount volume to pod

```
apiVersion: v1
kind: Pod
metadata:
  name: sleepypod
spec:
  volumes:
    - name: data
      gcePersistentDisk:
        pdName: panda-disk
        fsType: ext4
  containers:
    - name: sleepycontainer
      image: gcr.io/google_containers/busybox
      command:
        - sleep
        - "6000"
      volumeMounts:
        - name: data
          mountPath: /data
          readOnly: false
```

Remote Storage

- Kubernetes
  automatical
  - Attach volu
  - Mount volu


If you directly reference a volume
NSTRUCTO
You're gonna have a bad time

```
apiVersion: v1
kind: Pod
                      k:
                      disk
                      ainer
                      ogle_containers/busybox
      volumeMounts:
        - name: data
          mountPath: /data
          readOnly: false
```

Kubernetes Principle
Workload portability

Remote Storage

- Pod yaml is no longer portable across clusters!!

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sleepypod
spec:
  volumes:
    - name: data
      gcePersistentDisk:
        pdName: panda-disk
        fsType: ext4
  containers:
    - name: sleepycontainer
      image: gcr.io/google_containers/busybox
      command:
        - sleep
        - "6000"
      volumeMounts:
        - name: data
          mountPath: /data
          readOnly: false
```

Persistent Volumes & Persistent Volume Claims

PV/PVC

- PersistentVolume and PersistentVolumeClaim Abstraction
    - Decouple storage implementation from storage consumption

PersistentVolume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name : myPV1
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  persistentVolumeReclaimPolicy: Retain
  gcePersistentDisk:
    fsType: ext4
    pdName: panda-disk
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name : myPV2
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 100Gi
  persistentVolumeReclaimPolicy: Retain
  gcePersistentDisk:
    fsType: ext4
    pdName: panda-disk2
```

PersistentVolumeClaim

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc
  namespace: testns
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
```

PV to PVC Binding

```
$ kubectl create -f pv.yaml
persistentvolume "pv1" created
persistentvolume "pv2" created

$ kubectl get pv
NAME          CAPACITY   ACCESSMODES   STATUS      CLAIM                        REASON    AGE
pv1           10Gi       RWO           Available                                          1m
pv2           100Gi      RWO           Available                                          1m

$ kubectl create -f pvc.yaml
persistentvolumeclaim "mypvc" created

$ kubectl get pv
NAME          CAPACITY   ACCESSMODES   STATUS      CLAIM                        REASON    AGE
pv1           10Gi       RWO           Available                                          3m
pv2           100Gi      RWO           Bound       testns/mypvc                           3m
```

Remote Storage

- Volume referenced via PVC
- Pod YAML is portable across clusters again!!

```
apiVersion: v1
kind: Pod
metadata:
  name: sleepypod
~~spec:~~
~~  volumes:~~
~~    - name: data~~
~~      gcePersistentDisk:~~
~~        pdName: panda-disk~~
~~        fsType: ext4~~
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: mypvc
  containers:
    - name: sleepycontainer
      image: gcr.io/google_containers/busybox
      command:
        - sleep
        - "6000"
      volumeMounts:
        - name: data
          mountPath: /data
          readOnly: false
```

Dynamic Provisioning & Storage Classes

Dynamic Provisioning

- Cluster admin pre-provisioning PVs is painful and wasteful.
- Dynamic provisioning creates new volumes on-demand (when requested by user).
- Eliminates need for cluster administrators to pre-provision storage.

Dynamic Provisioning

- Dynamic provisioning “enabled” by creating StorageClass.
- StorageClass defines the parameters used during creation.
- StorageClass parameters opaque to Kubernetes so storage providers can expose any number of custom parameters for the cluster admin to use.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: slow
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard
--
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: fast
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-ssd
```

Dynamic Provisioning

- Users consume storage the same way: PVC
- “Selecting” a storage class in PVC triggers dynamic provisioning

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc
  namespace: testns
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: fast
```

Dynamic Provisioning

```
$ kubectl create -f storage_class.yaml
storageclass "fast" created

$ kubectl create -f pvc.yaml
persistentvolumeclaim "mypvc" created

$ kubectl get pvc --all-namespaces
NAMESPACE   NAME                       STATUS    VOLUME                                     CAPACITY   ACCESSMODES   AGE
testns      mypvc                      Bound     pvc-331d7407-fe18-11e6-b7cd-42010a8000cd   100Gi      RWO           6s

$ kubectl get pv pvc-331d7407-fe18-11e6-b7cd-42010a8000cd
NAME                                       CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS    CLAIM          REASON    AGE
pvc-331d7407-fe18-11e6-b7cd-42010a8000cd   100Gi      RWO           Delete          Bound     testns/mypvc             13m
```

Dynamic Provisioning

Volume referenced via PVC

```
apiVersion: v1
kind: Pod
metadata:
  name: sleepypod
spec:
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: mypvc
  containers:
    - name: sleepycontainer
      image: gcr.io/google_containers/busybox
      command:
        - sleep
        - "6000"
      volumeMounts:
        - name: data
          mountPath: /data
          readOnly: false
```

Dynamic Provisioning

- Default Storage Classes
  - Enable dynamic provisioning even when StorageClass not specified.
- Pre-installed Default Storage Classes
  - Amazon AWS - EBS volume
  - Google Cloud (GCE/GKE) - GCE PD
  - Openstack - Cinder Volume

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: slow
  annotations:
    storageclass.beta.kubernetes.io/is-default-class:
"true"
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard

--

kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: fast
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-ssd
```

Hostpath Volumes

- Expose a directory on the host machine to pod
- What happens if your pod is moved to a different node?
- Don't use hostpath (unless you know what you are doing)!!

Local Persistent Volumes

- Expose a local block or file as a PersistentVolume
- Reduced durability
- Useful for building distributed storage systems
- Useful for high performance caching
- Kubernetes takes care of data gravity
- Referenced via PV/”PVC so workload portability is maintained
- Kubecon EU Talk: Using Kubernetes Local Storage for Scale-Out Storage Services in Production”by Michelle Au

In-Tree Volume Plugins

- Kubernetes “In-tree” Volume Plugins are awesome =)
  - Powerful abstraction for file and block storage
  - Automate provisioning, attaching, mounting, and more!
  - Storage portability via PV/PVC/StorageClass objects

In-Tree Volume Plugins

- Kubernetes “In-tree” Volume Plugins are painful =(
  - Painful for Kubernetes Developers
    - Testing and maintaining external code
    - Bugs in volume plugins affect critical Kubernetes components
    - Volume plugins get full privileges of kubernetes components (kubelet and kube-controller-manager)
  - Painful for Storage Vendors
    - Dependent on Kubernetes releases
    - Source code forced to be open source

Out-of-Tree Volume Plugins

- Container Storage Interface (CSI) - Beta in v1.10
  - Follows in the steps of CRI and CNI
  - Collaboration with other cluster orchestration systems
  - CSI makes Kubernetes volume layer truly extensible
  - Plugins may be containerized
  - Kubecon EU Talk “Container Storage Interface: Present and Future” by Jie Yu
- Flex Volumes
  - Legacy attempt at out-of-tree
  - Exec based
  - Deployment difficult
  - Doesn't support clusters with no master access

Questions?

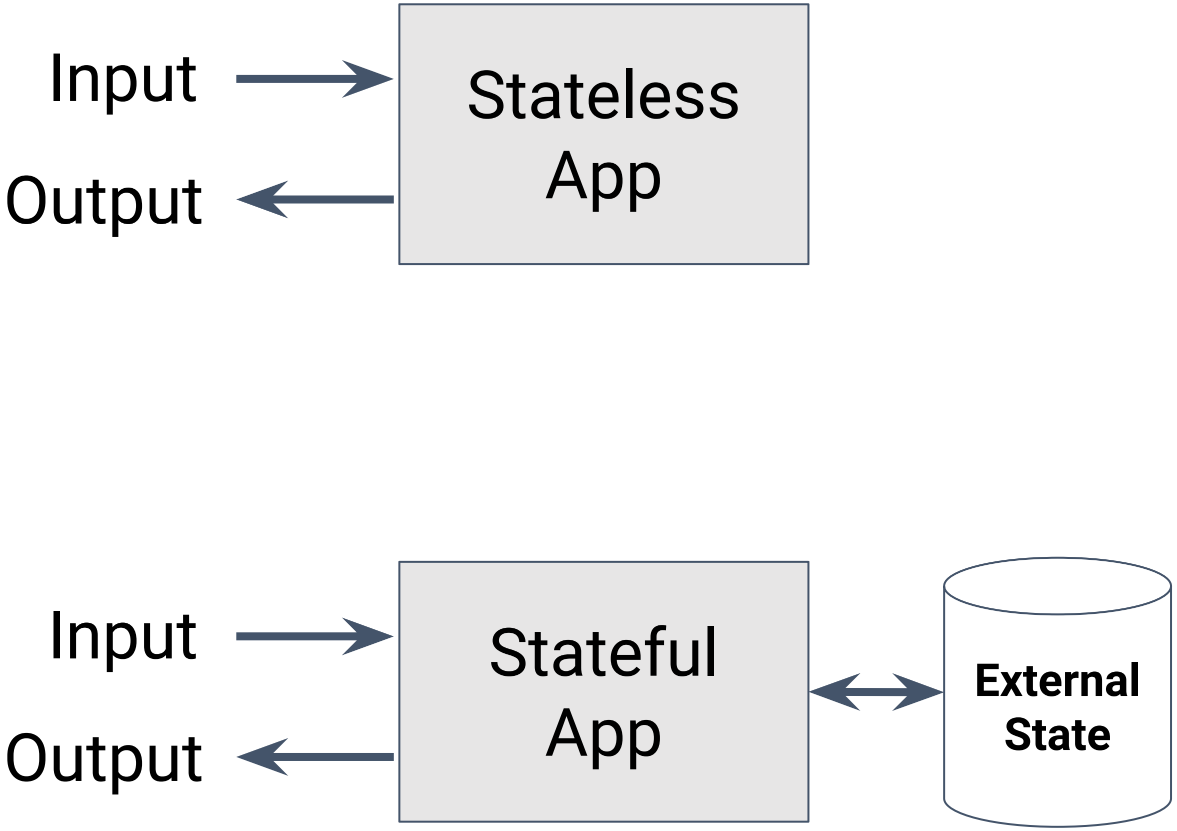
- Get Involved!
  - Kubernetes Storage Special-Interest-Group (SIG)
    - github.com/kubernetes/community/tree/master/sig-storage
    - Meeting every 2 weeks, Thursdays at 9 AM (PST)
    - Mailing list:
      - kubernetes-sig-storage@googlegroups.com
- Contact me:
  - Saad Ali, Google
  - github.com/saad-ali
  - twitter.com/the_saad_ali

What are stateful apps?

KubeCon | CloudNativeCon
Europe 2018

Good for ***stateless*** apps (apps dependent only on input parameters and app code).

What about ***stateful*** apps (apps that depend on reading or writing some external state in addition to input parameters and app code)?


Input
Output
Stateless App
Input
Output
Stateful App
External State