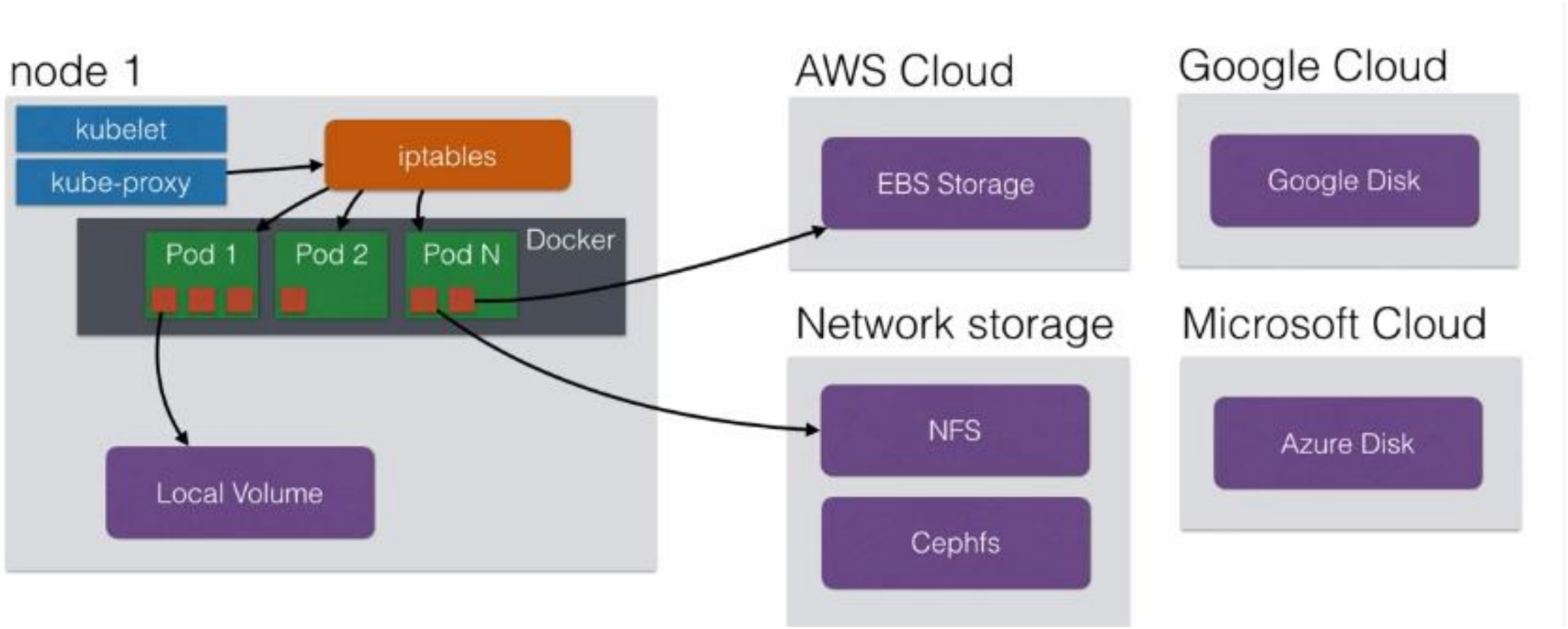
Volumes

- Volumes in kubernetes allow you to **store data outside the container**
- When a containers **stops,** all data on the container itself is **lost**
  - That’s why up until now I’ve been using **stateless** apps: apps that don’t keep a **local** state, but store theirs state in an **external service**
    - External Service like a database, cashing server (e.g. MySQL, AWS S3)
- Persistent Volumes in Kubernetes allow you **attach a volume** to a container that will **exists** even when the **container** stops

Kubernetes Volumes

- Volumes can be attached using different volumes plugins:

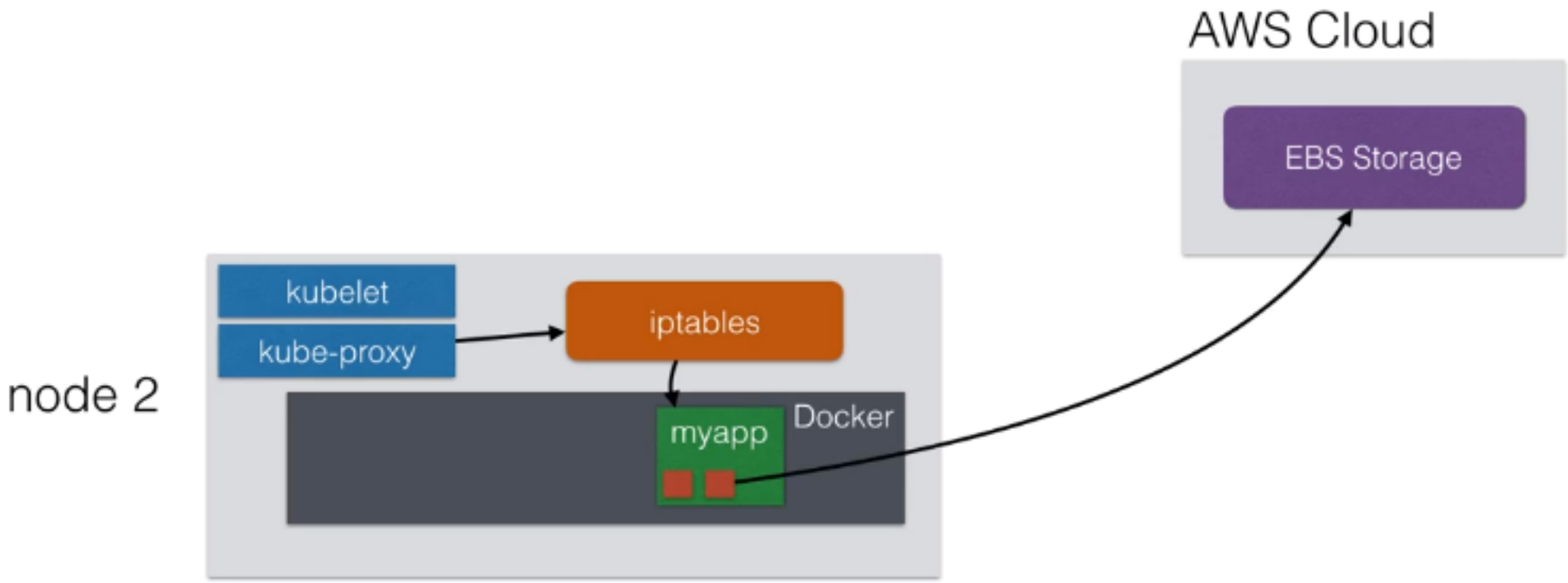


Volumes

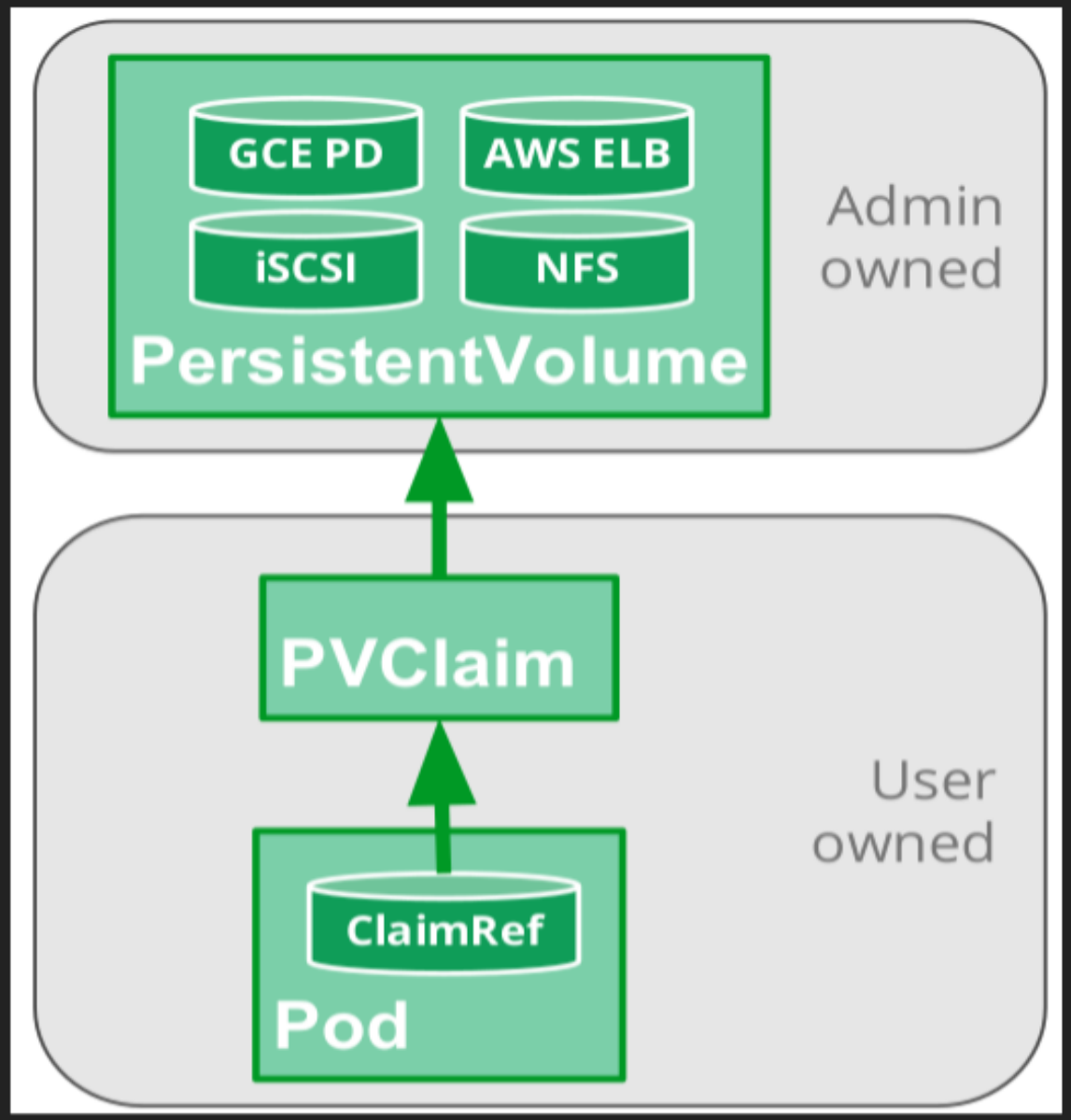
- Using volumes, you could deploy **applications with state** on your cluster
  - Those applications need to read/write to files on the **local filesystem** that need to be persistent in time
- You could run a MySQL database using persistent volumes
  - Although this might not be ready for production (yet)
  - Volumes are new since the June 2016 release in Kubernetes

Volumes

- If your **node stops** working, the pod can be rescheduled on another node, and the volumes can be attached to new node



Persistent Volumes


GCE PD
AWS ELB
iSCSI
NFS
PersistentVolume
Admin owned
PVClaim
ClaimRef
Pod
User owned

GCEPersistentDisk	CephFS
AWSElasticBlockStore	Cinder (OpenStack block storage)
AzureFile	Glusterfs
AzureDisk	VsphereVolume
FC (Fibre Channel)	Quobyte Volumes
FlexVolume	HostPath
Flocker	VMware Photon
NFS	vPortworx Volumes
iSCSI	ScaleIO Volumes
RBD (Ceph Block Device)	StorageOS

Persistent Volume Usage Design

Persistent Volume Provisioning Types

- **Static**
  - Manually map physical storage to PVs
- **Dynamic**
  - When static PVs are unavailable, K8S will check the availability of dynamic PVs
  - It dynamically creates volumes using storage provisioners
  - The storage provisioner may use a single physical storage for providing multiple dynamic PVs

https://kubernetes.io/docs/concepts/storage/persistent-volumes/
https://github.com/kubernetes-incubator/external-storage/

Persistent Volume Claims

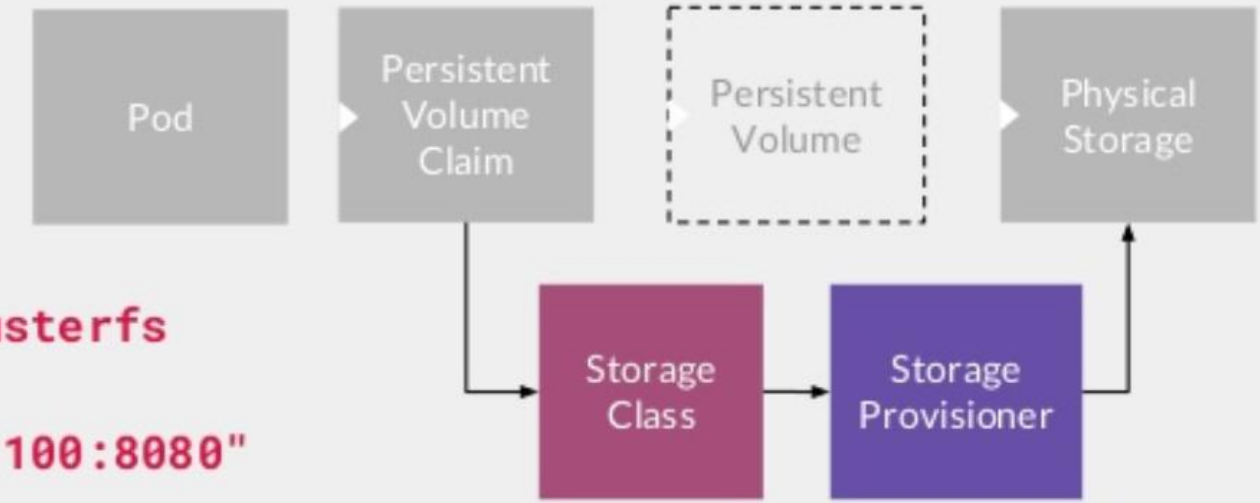
```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 8Gi
  storageClassName: slow
  selector:
    matchLabels:
      release: "stable"
    matchExpressions:
      - {key: environment, operator: In, values: [dev]}
```

Static Persistent Volumes

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0003
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /tmp
    server: 172.17.0.2
```

Dynamic Persistent Volumes

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gluster-vol-default
provisioner: kubernetes.io/glusterfs
parameters:
  resturl: "http://192.168.10.100:8080"
  restuser: ""
  secretNamespace: ""
  secretName: ""
allowVolumeExpansion: true
```



Persistent Volume Volume Modes

- Filesystem
- Block

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: block-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  volumeMode: Block
  persistentVolumeReclaimPolicy: Retain
  fc:
    targetWWNs: ["50060e801049cfd1"]
    lun: 0
    readOnly: false
```

https://kubernetes.io/docs/concepts/storage/persistent-volumes/#volume-modes

Persistent Volume Access Modes

- **ReadWriteOnce**
  - read-write by a single node
- **ReadOnlyMany**
  - read-only by many nodes
- **ReadWriteMany**
  - read-write by many nodes

https://kubernetes.io/docs/concepts/storage/persistent-volumes/#access-modes

Persistent Volume Binding

- Binding a PVC to PV will be done based on requested:
  - Amount of storage
  - Access mode
  - Storage Class (Optional)

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

Persistent Volume Re-claim Policies

- **Retain**
  - Manual reclamation
- **Recycle**
  - Basic scrub (rm -rf /thevolume/*) on the volume and makes it available again for a new claim
- **Delete**
  - Deletion removes both the PV object from Kubernetes, as well as the associated storage asset in the external infrastructure

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

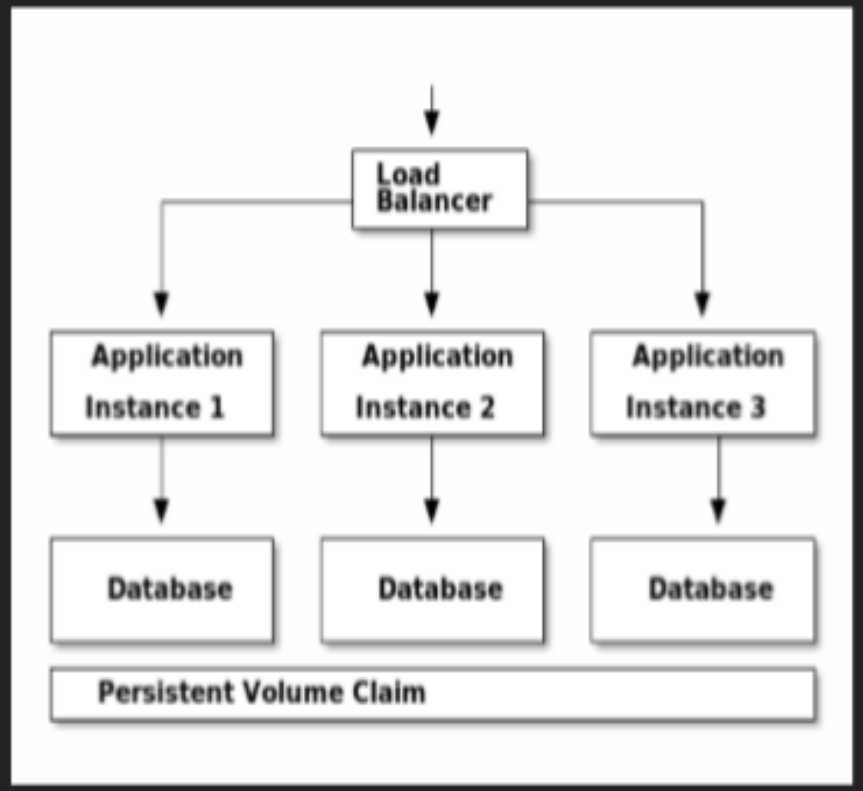
STORAGECLASS

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
```

USING A STORAGECLASS

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
  storageClassName: standard
```

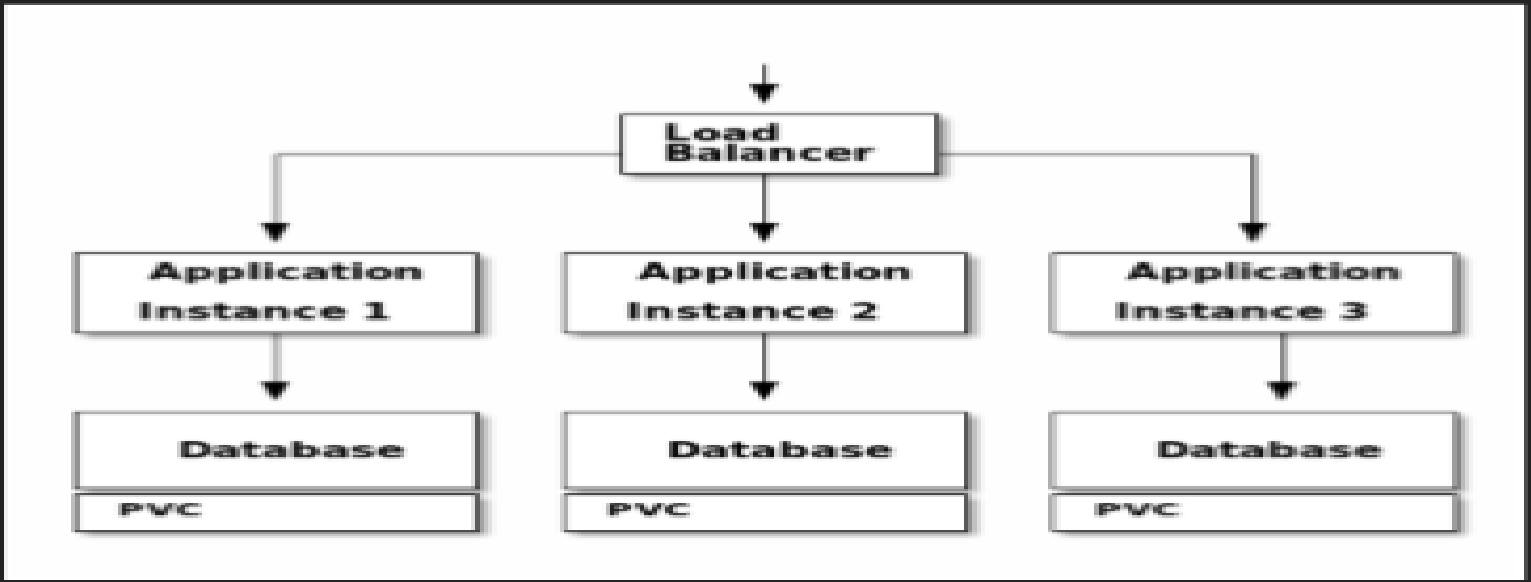
ARCHITECTURE REVISITED



SHARING PVCS

- Need different paths per Pod.
- Share needs to be available on every host.
- Corruption of file system affects all database nodes.
- Sharing a Persistent Volume Claim is not a good idea.

ARCHITECTURE STATEFULSETS



STATEFULSETS

- Pod template mechanism
- Hostnames are atomically increased:
  - pod-0
  - pod-1
  - ...
- Volume Claims can be provisioned on-the-fly

STATEFULSET EXAMPLE

```
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: gcr.io/google_containers/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
```