Kubernetes Storage Architecture and Evolution

任玉泉

Content

- kubernetes storage overview
- kubernetes storage implementation
- kubernetes storage usage evolution
- kubernetes storage future features
- Q&A

架构迎接未来变化
IAS2017 • NANJING

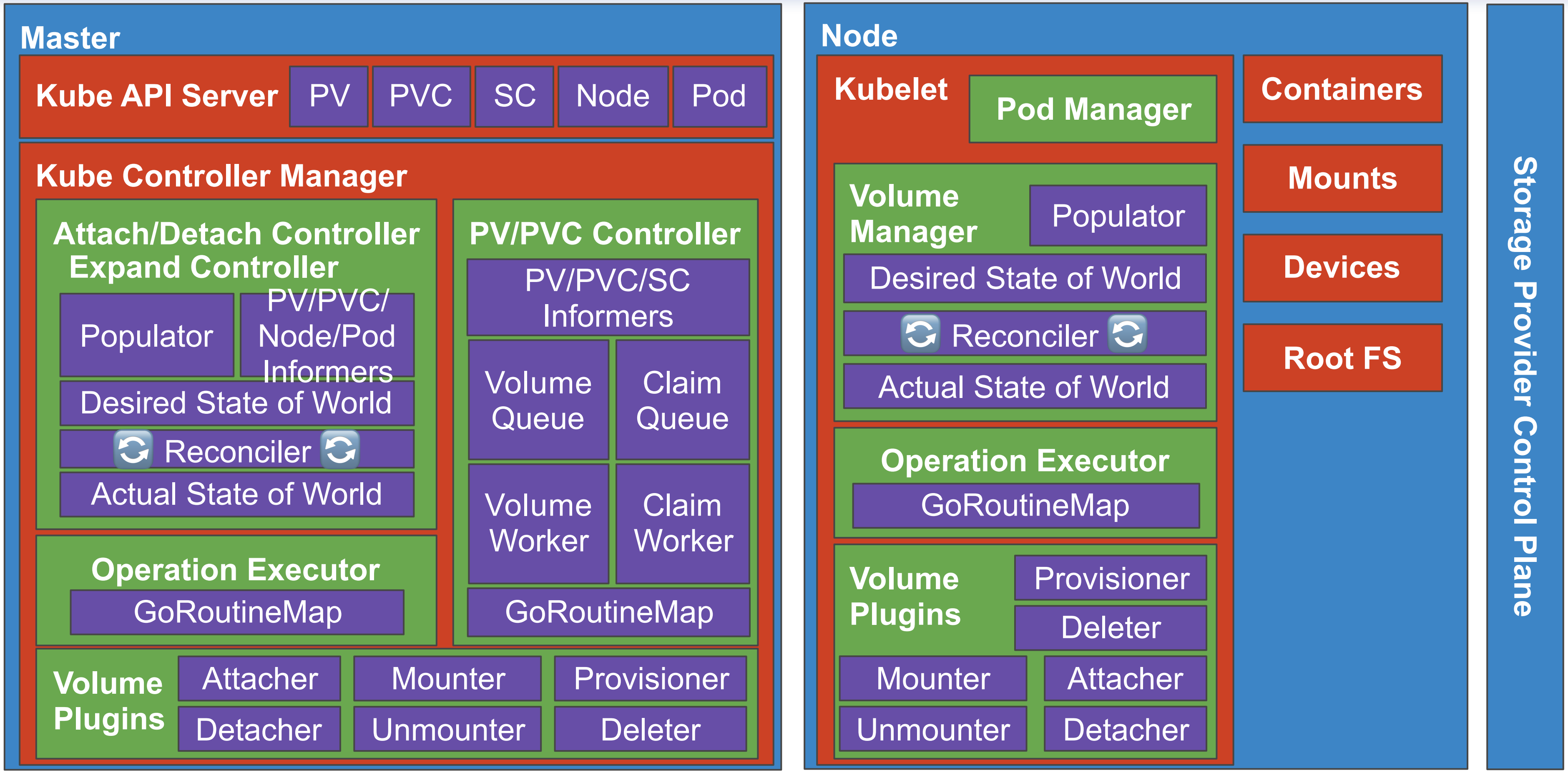
Kubernetes Design Principles

Content

- **Declarative > imperative:** State your desired results, let the system actuate
- **Control loops:** Observe, rectify, repeat
- **Simple > Complex:** Try to do as little as possible
- **Modularity:** Components, interfaces, & plugins
- **Legacy compatible:** Meet users where they are, requiring apps to change is a non-starter
- **Open > Closed:** Open Source, standards, REST, JSON, etc.

架构迎接未来变化
IAS2017 • NANJING

Storage Architecture Overview



Kubernetes Supported Storage

Persistent

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

Ephemeral

- Empty dir (and tmpfs)
- Expose Kubernetes API
  - Secret
  - ConfigMap
  - DownwardAPI

New

- Local Storage

Kubernetes Storage Implementation

- Volume Plugin Interface
- Kubelet Volume Manager
- Attach/Detach Controller
- PV/PVC Controller
- ExpandVolume Controller

架构迎接未来变化
IAS2017 • NANJING

Kubernetes Storage Implementation

Volume Plugin Interface

- Golang packages in core Kubernetes repository
  - `kubernetes/pkg/volume/`
- Implement golang interfaces
  - `Mounter`
  - `Unmounter`
  - Optionally
    - `Attacher`
    - `Detacher`
    - `Provisioner`
    - `Deleter`
    - `Recycler`

Kubernetes Storage Implementation

Volume Plugin Interface

- Mounter/Unmounter Interface
  - Make data source (volume, block device, network share, or something else) available as a directory on host's root FS.
  - Directory then mounted into pods by kubelet
  - Methods always called from node (Kubelet binary)

- Methods
  - `SetUpAt(dir, ...)`
  - `TearDownAt(dir)`
  - `...`

架构迎接未来变化
IAS2017 • NANJING

Kubernetes Storage Implementation

Volume Plugin Interface

- Attacher/Detacher Interface
  - Make block device available on specified host.
  - Attach & VolumesAreAttached methods called from master (kube controller binary).

- Methods
  - `Attach(spec, nodeName)`
  - `VolumesAreAttached (specs, nodeName)`
  - `WaitForAttach (spec, devicePath, timeout)`
  - `MountDevice (spec, devicePath, deviceMountPath)`
  - `UnmountDevice (deviceMountPath)`
  - `...`

Kubernetes Storage Implementation

Volume Plugin Interface

- Provisioner/Deleter Interface
  - Create and delete new pieces of physical storage and the k8s PV object to represent it.
  - Methods called from master (kube controller binary).
- Methods
  - `Provision()`
  - `Delete()`

Kubernetes Storage Implementation

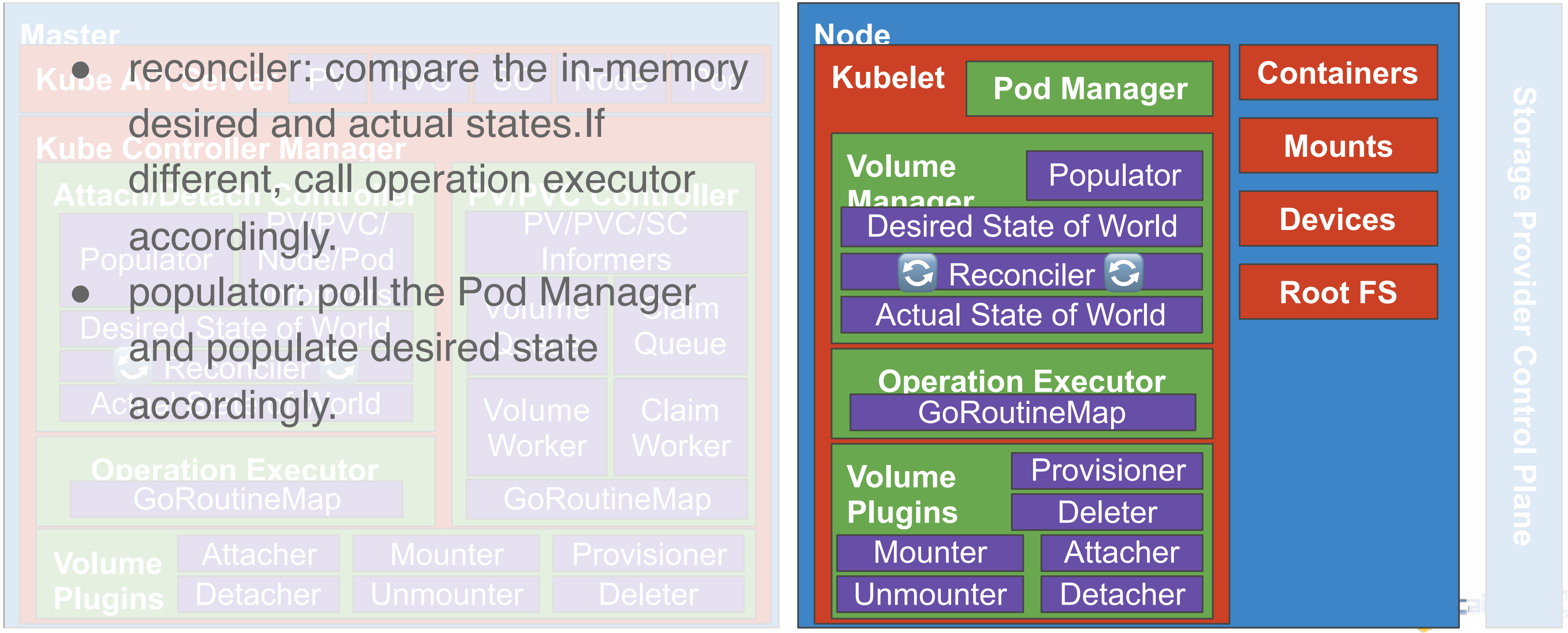
Volume Plugin Interface

- Take cinder as an example
  - create cinder volume (provision)
  - attach to instance
  - mount device (/var/lib/kubelet/plugins/kubernetes.io/cinder/mounts/cinder-volume-id)
  - mounted to pod volume dir (/var/lib/kubelet/pods/{podUID}/volumes/kubernetes.io~cinder/{outerVolumeSpecName}/)

Kubernetes Storage Implementation

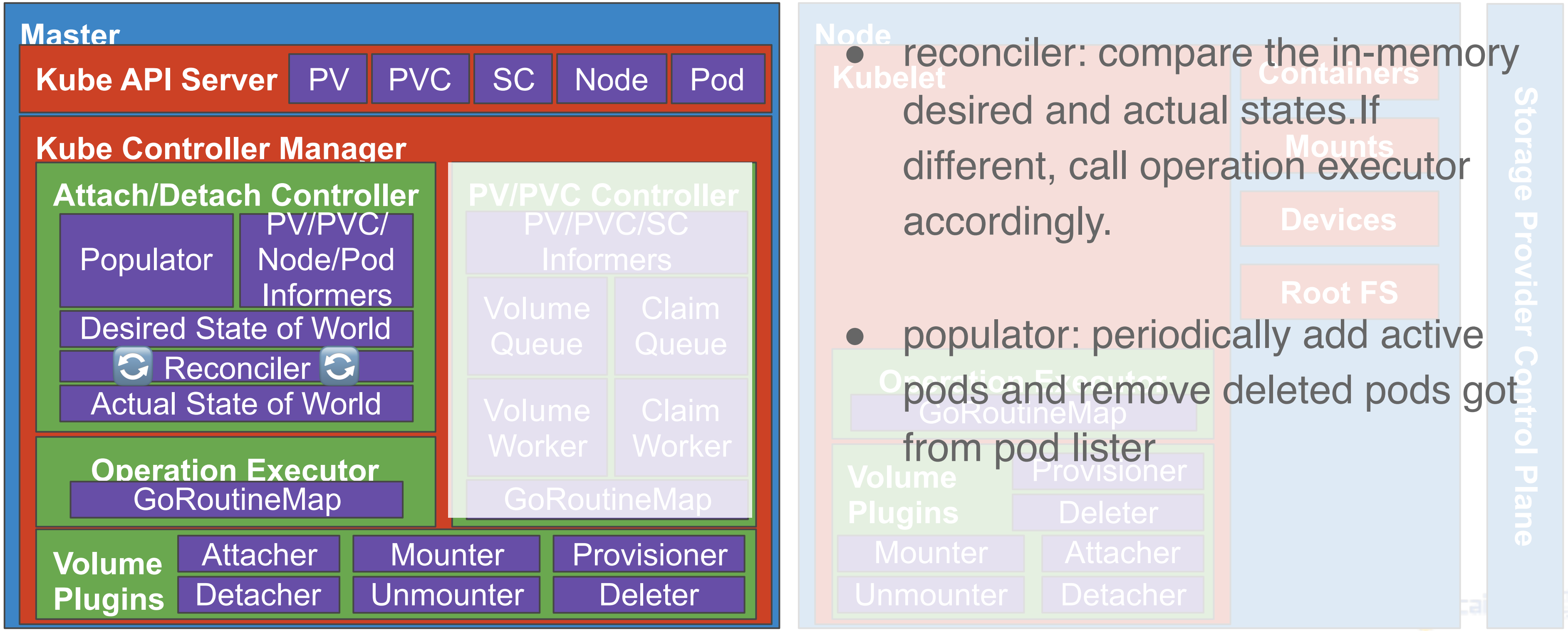
Kubelet Volume Manager

- reconciler: compare the in-memory desired and actual states.If different, call operation executor accordingly.
- populator: poll the Pod Manager and populate desired state accordingly.



Kubernetes Storage Implementation

Attach/Detach Controller

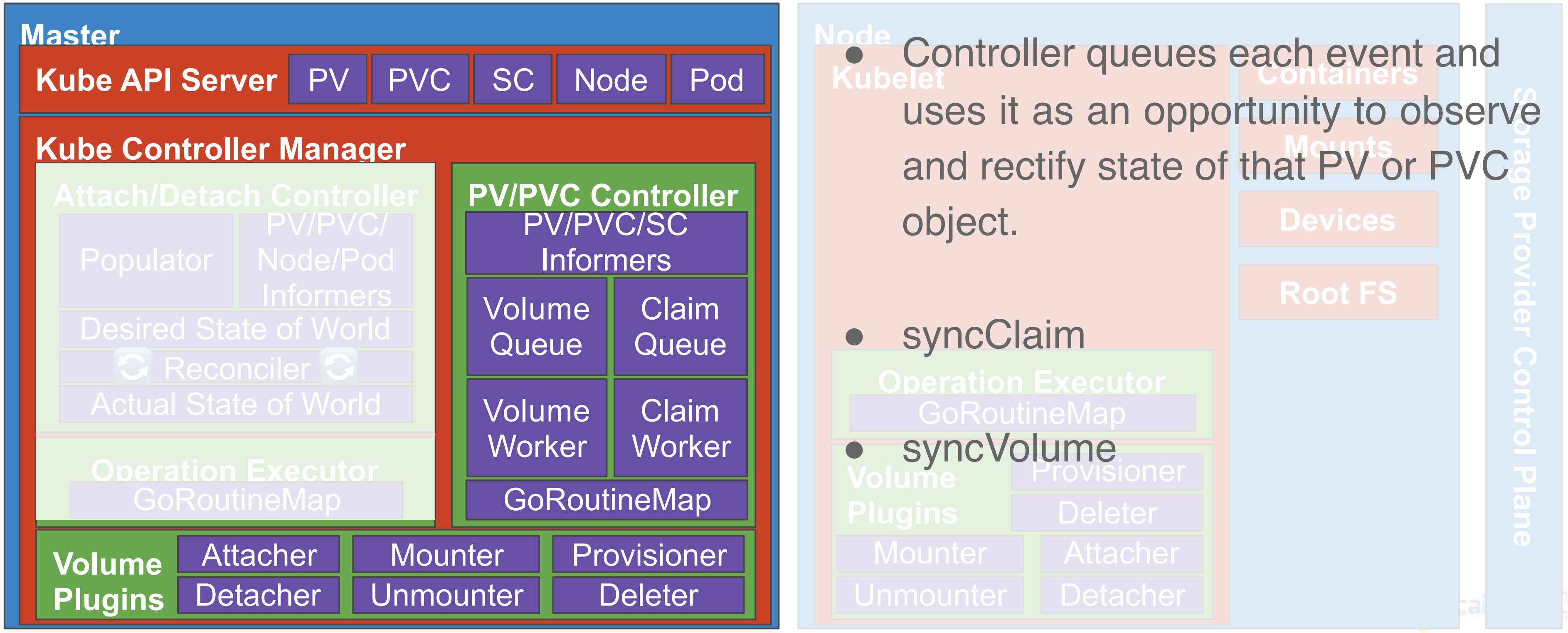


- reconciler: compare the in-memory desired and actual states.If different, call operation executor accordingly.

- populator: periodically add active pods and remove deleted pods got from pod lister

Kubernetes Storage Implementation

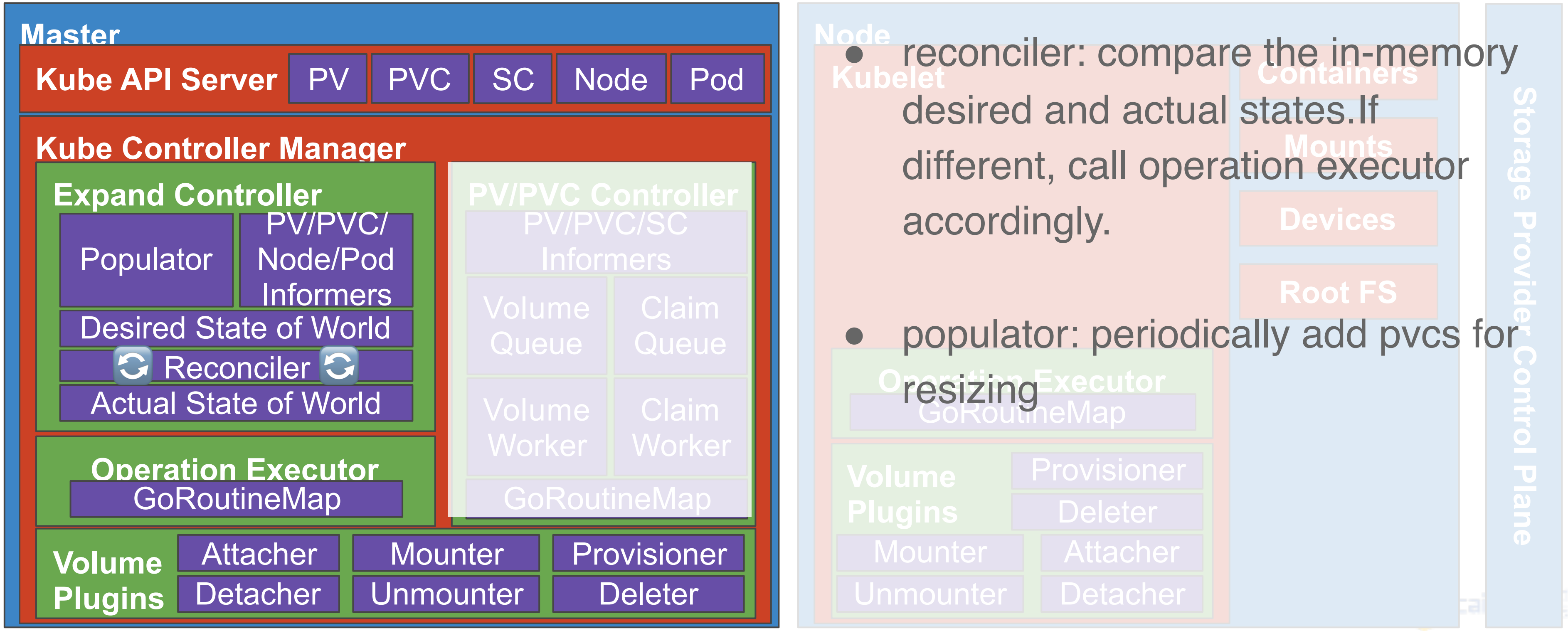
PV/PVC Controller



- Controller queues each event and uses it as an opportunity to observe and rectify state of that PV or PVC object.
- syncClaim
- syncVolume

Kubernetes Storage Implementation

ExpandVolume Controller

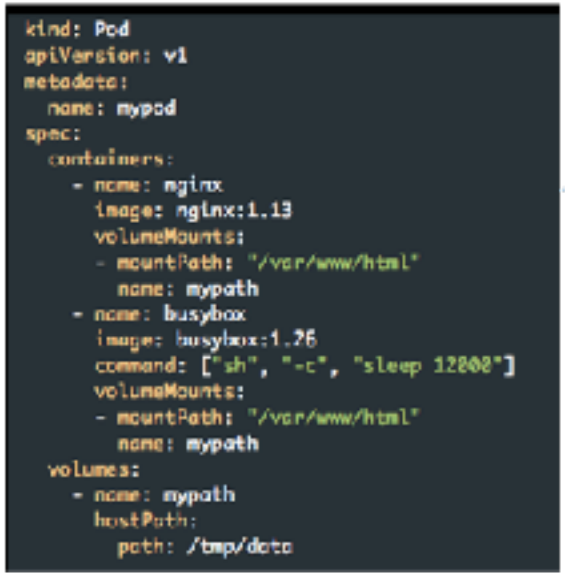


- reconciler: compare the in-memory desired and actual states.If different, call operation executor accordingly.
- populator: periodically add pvcs for resizing

Kubernetes Storage Usage evolution

Direct Access:

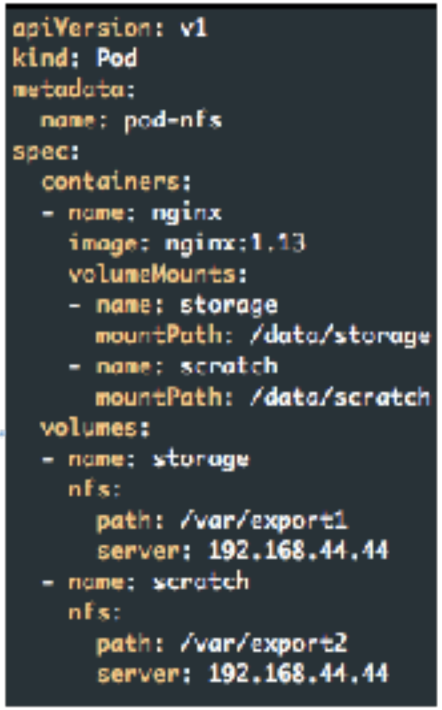
- Directly write volume details in Pod configuration
- Same approach for all kinds of volumes, i.e. persistent, local, ephemeral, etc



```yaml
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: nginx
      image: nginx:1.13
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypath
    - name: busybox
      image: busybox:1.26
      command: ["sh", "-c", "sleep 12000"]
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypath
  volumes:
    - name: mypath
      hostPath:
        path: /tmp/data
```

Host Path

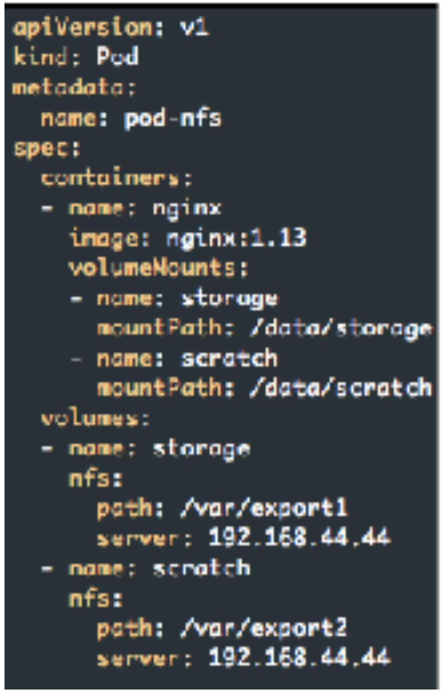
NFS



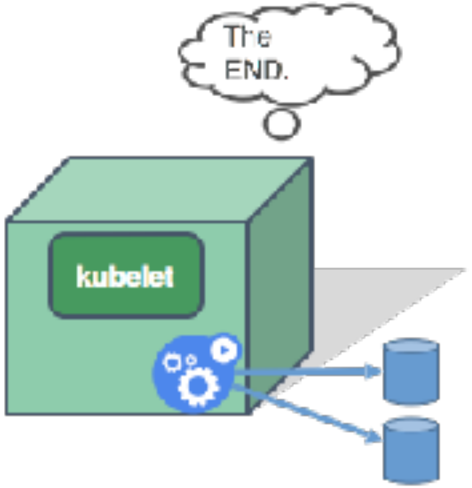
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-nfs
spec:
  containers:
  - name: nginx
    image: nginx:1.13
    volumeMounts:
    - name: storage
      mountPath: /data/storage
    - name: scratch
      mountPath: /data/scratch
  volumes:
  - name: storage
    nfs:
      path: /var/export1
      server: 192.168.44.44
  - name: scratch
    nfs:
      path: /var/export2
      server: 192.168.44.44
```

Kubernetes Storage Usage evolution

Direct Access:



```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nfs
spec:
  containers:
  - name: nginx
    image: nginx:1.13
    volumeMounts:
    - name: storage
      mountPath: /data/storage
    - name: scratch
      mountPath: /data/scratch
  volumes:
  - name: storage
    nfs:
      path: /var/export1
      server: 192.168.44.44
  - name: scratch
    nfs:
      path: /var/export2
      server: 192.168.44.44
```



Observation:

- Pod is created and scheduled on a Node
  - scheduling is **independent** of volume
- Kubelet has built-in plugin libraries
  - one for each supported volume type
- Two **existing** NFS volumes are attached to Pod
  - no provisioning
  - no configuration knob
- More

Kubernetes Storage Usage evolution

- Root problem with direct access
  - Tight coupling between setting up storage and request/use storage
- Solution
  - Add another layer which separate the complexity: admin sets up storage, user requests storage

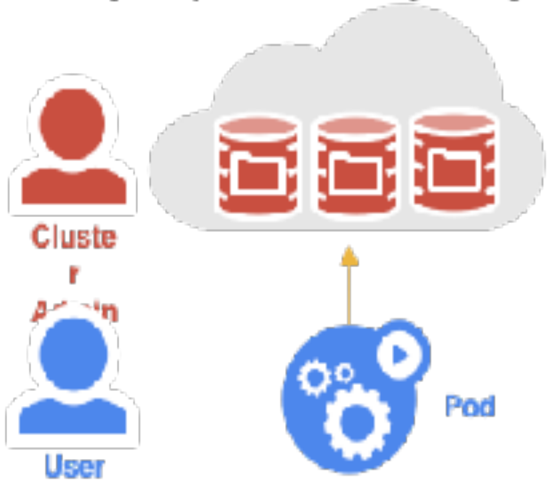


Image Source: Google

Kubernetes Storage Usage evolution

- Admin <- PersistentVolume (PV)
  - Persistent volume represents a schedulable, requestable storage identity
  - Can be networked storage, local storage, etc
- User <- PersistentVolumeClaim (PVC)
  - Claim volumes of specific size and modes

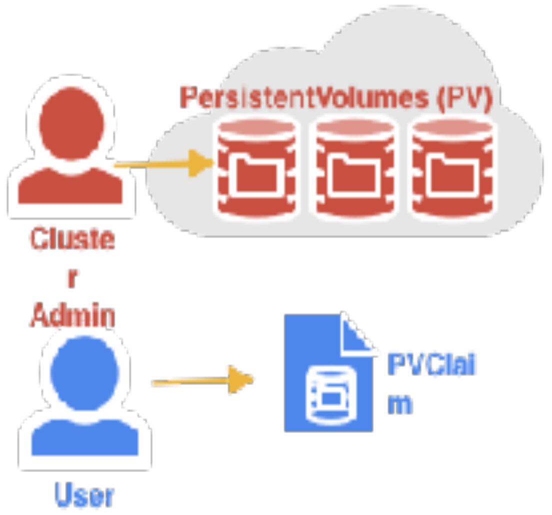
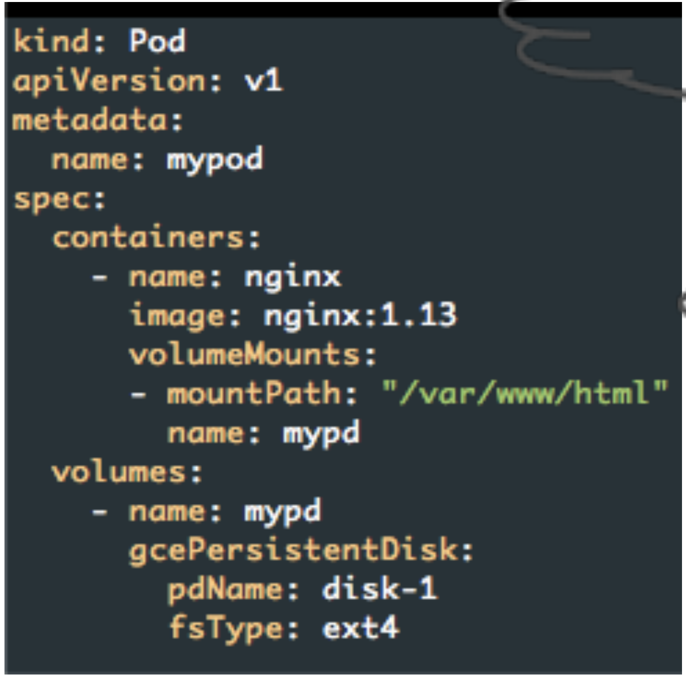
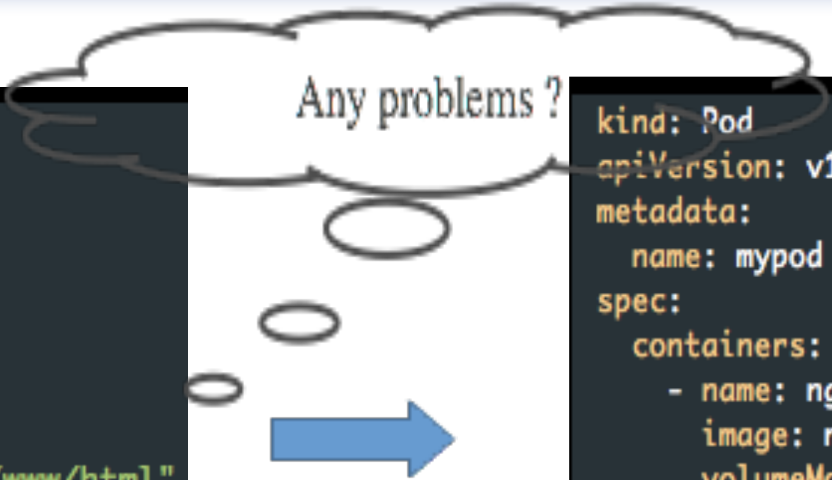


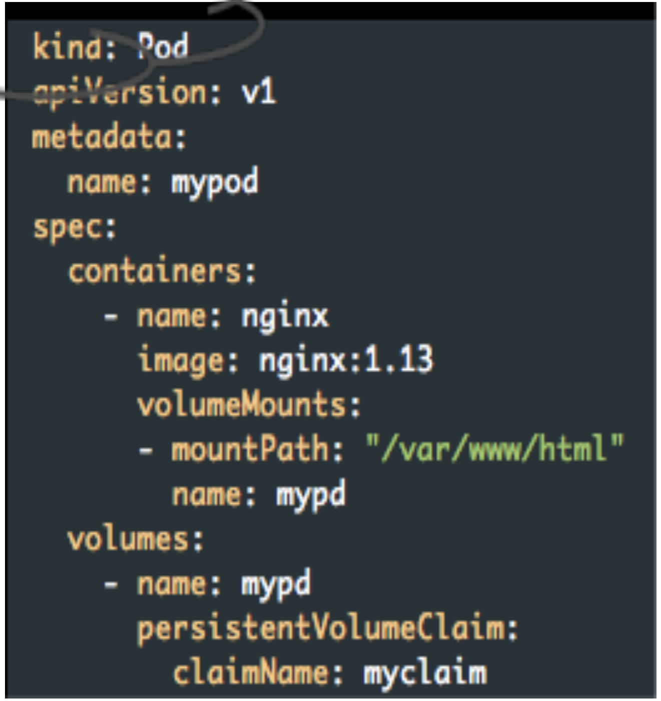
Image Source: Google

Kubernetes Storage Usage evolution



```
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: nginx
      image: nginx:1.13
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypd
  volumes:
    - name: mypd
      gcePersistentDisk:
        pdName: disk-1
        fsType: ext4
```

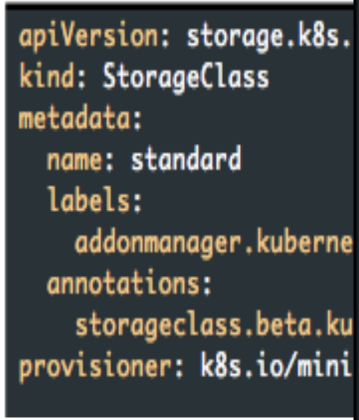




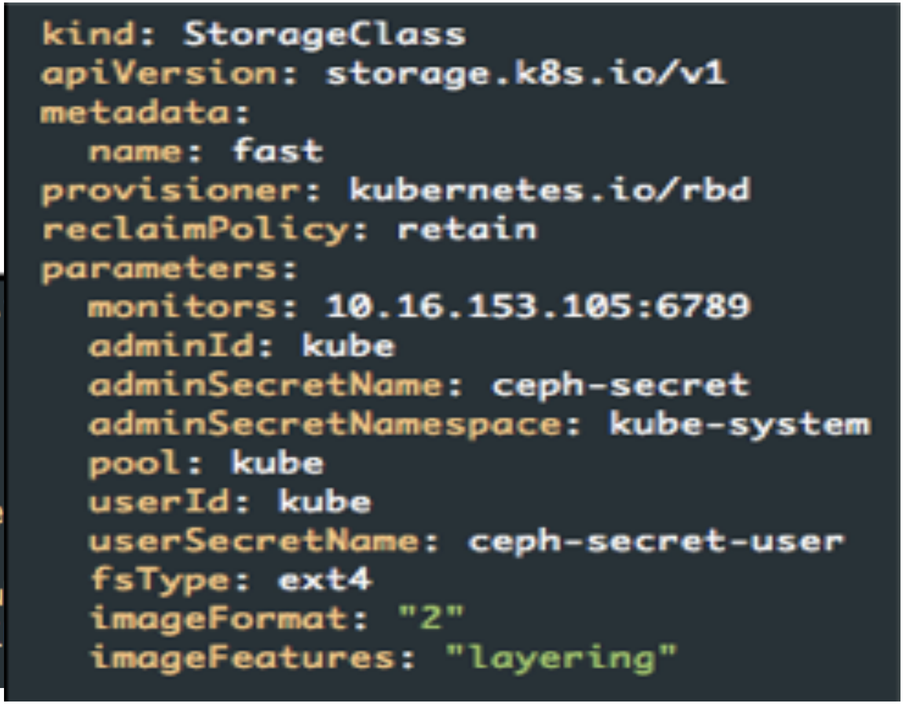
```
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: nginx
      image: nginx:1.13
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim
```

Kubernetes Storage Usage evolution

- StorageClass is an API object created by admin to enable dynamic provisioning
  - Create PersistentVolume on request
  - Allow more configuration parameters

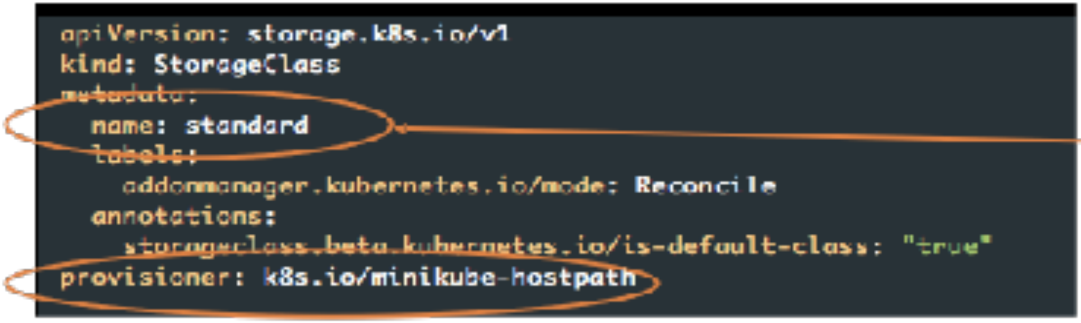


```
apiVersion: storage.k8s.
kind: StorageClass
metadata:
  name: standard
  labels:
    addonmanager.kuberne
  annotations:
    storageclass.beta.ku
provisioner: k8s.io/mini
```

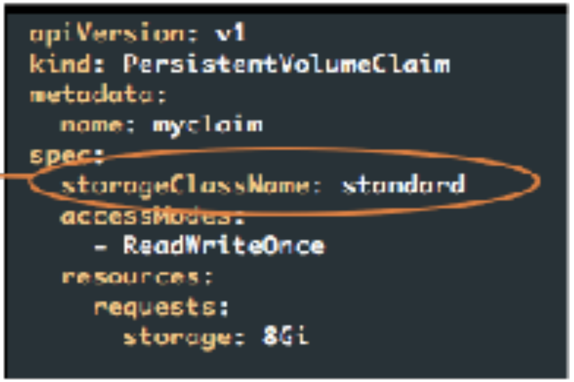


```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: fast
provisioner: kubernetes.io/rbd
reclaimPolicy: retain
parameters:
  monitors: 10.16.153.105:6789
  adminId: kube
  adminSecretName: ceph-secret
  adminSecretNamespace: kube-system
  pool: kube
  userId: kube
  userSecretName: ceph-secret-user
  fsType: ext4
  imageFormat: "2"
  imageFeatures: "layering"
```

Kubernetes Storage Usage evolution



```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
  labels:
    addonmanager.kubernetes.io/mode: Reconcile
  annotations:
    storageclass.beta.kubernetes.io/is-default-class: "true"
provisioner: k8s.io/minikube-hostpath
```

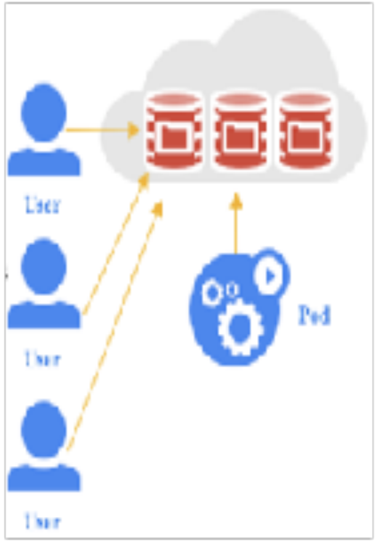


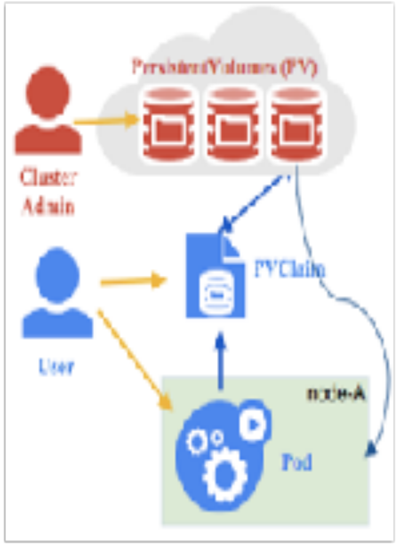
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  storageClassName: standard
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
```

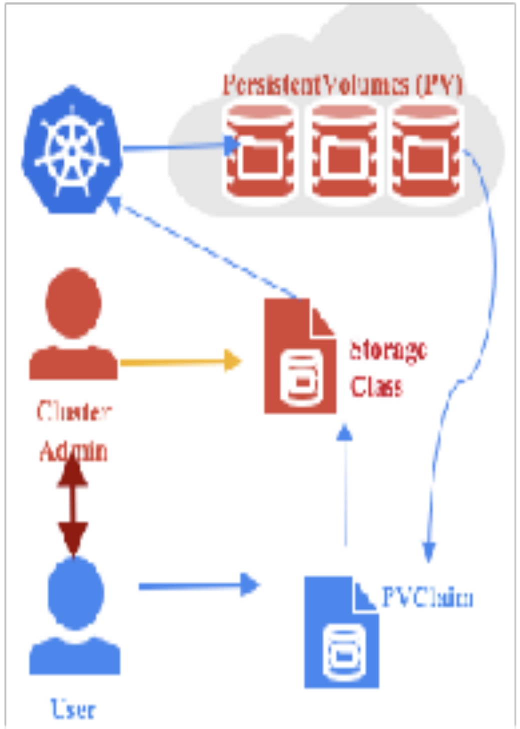
Watch All new Claims, for each one, find its StorageClass based on spec.storageClassName, then provision new PV if class.provisioner match my name.

Kubernetes Storage Usage evolution

- Evolution Path







Kubernetes Storage Future Features

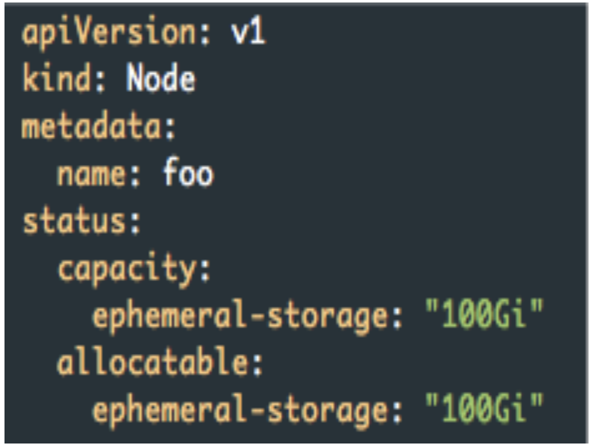
Content

- Local ephemeral storage
- PVC resize
- Local persistent storage

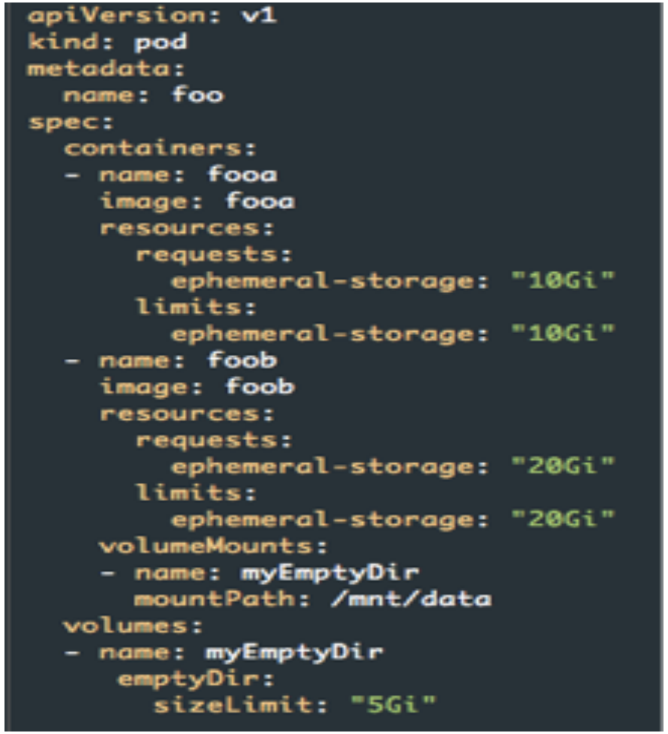
架构迎接未来变化
IAS2017 • NANJING

Kubernetes Storage Future Features

- Local ephemeral storage



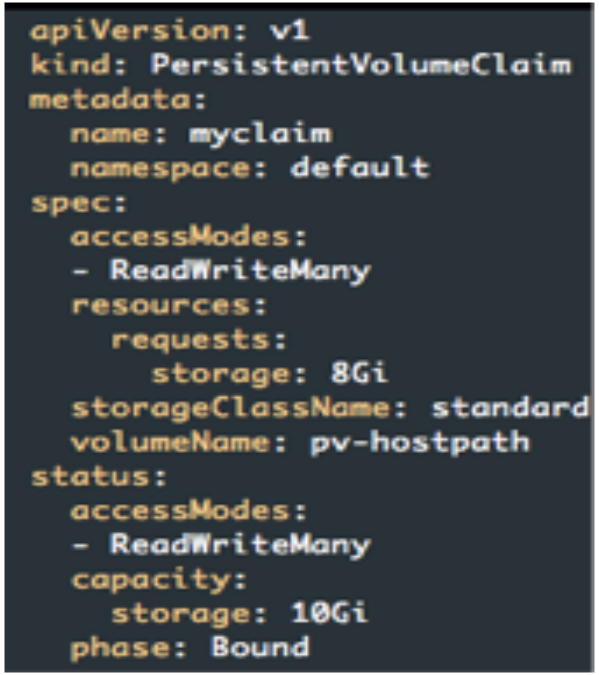
```yaml
apiVersion: v1
kind: Node
metadata:
  name: foo
status:
  capacity:
    ephemeral-storage: "100Gi"
  allocatable:
    ephemeral-storage: "100Gi"
```



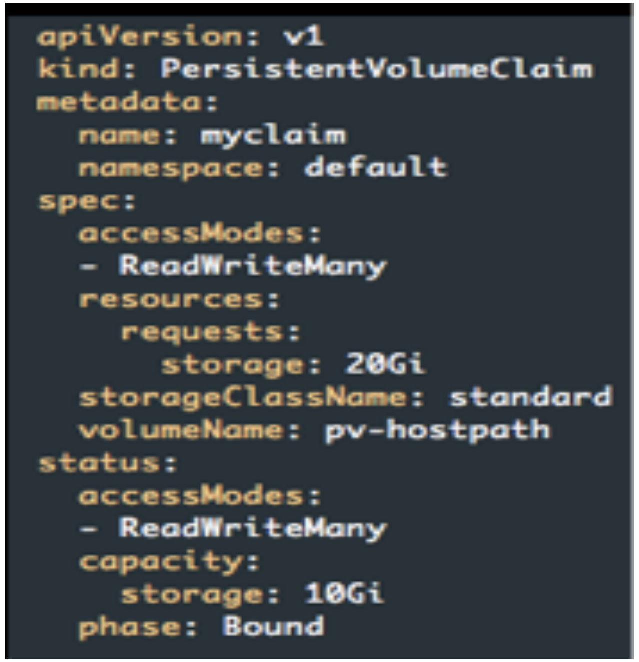
```yaml
apiVersion: v1
kind: pod
metadata:
  name: foo
spec:
  containers:
  - name: fooa
    image: fooa
    resources:
      requests:
        ephemeral-storage: "10Gi"
      limits:
        ephemeral-storage: "10Gi"
  - name: foob
    image: foob
    resources:
      requests:
        ephemeral-storage: "20Gi"
      limits:
        ephemeral-storage: "20Gi"
    volumeMounts:
    - name: myEmptyDir
      mountPath: /mnt/data
  volumes:
  - name: myEmptyDir
    emptyDir:
      sizeLimit: "5Gi"
```

Kubernetes Storage Future Features

- PVC resize



```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 8Gi
  storageClassName: standard
  volumeName: pv-hostpath
status:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  phase: Bound
```

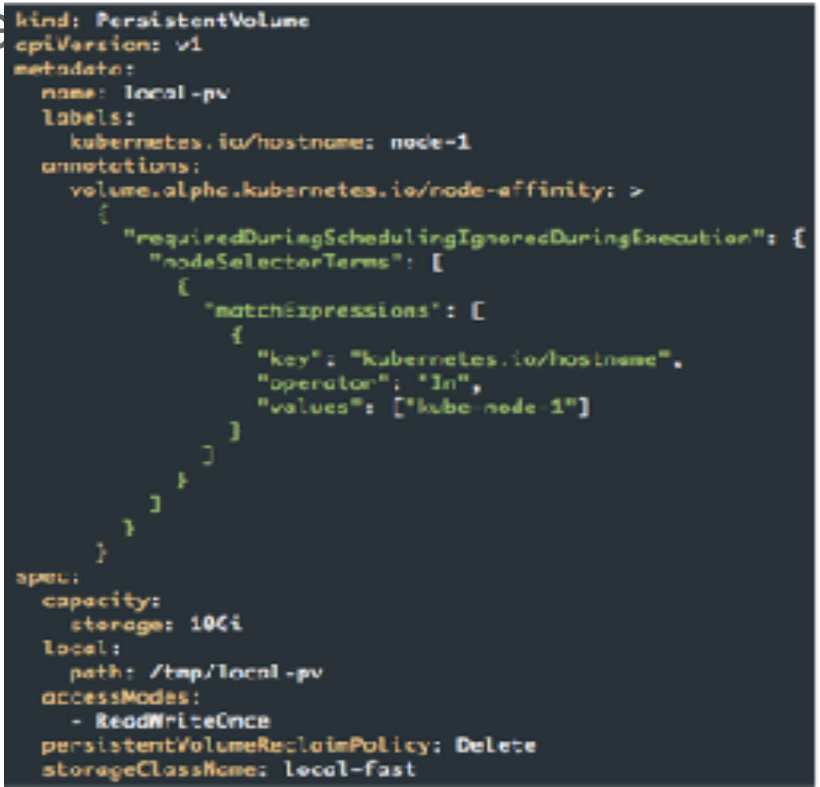


```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
  namespace: default
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 20Gi
  storageClassName: standard
  volumeName: pv-hostpath
status:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  phase: Bound
```

Kubernetes Storage Future Features

Content

- Local persistent storage



```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: local-pv
  labels:
    kubernetes.io/hostname: node-1
  annotations:
    volume.alpha.kubernetes.io/node-affinity: >
      {
        "requiredDuringSchedulingIgnoredDuringExecution": {
          "nodeSelectorTerms": [
            {
              "matchExpressions": [
                {
                  "key": "kubernetes.io/hostname",
                  "operator": "In",
                  "values": ["kube-node-1"]
                }
              ]
            }
          ]
        }
      }
spec:
  capacity:
    storage: 10Gi
  local:
    path: /tmp/local-pv
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-fast
```

End

Content

Thank you !

架构迎接未来变化

IAS2017 • NANJING