Understanding Kubernetes Storage

Getting in Deep by Writing a CSI Driver

Gerry Seidman

AURISTOR
YOUR DATA · YOUR FILE SYSTEM

Agenda

- The Kubernetes Storage Journey
- Understanding the Kubernetes API Model
- Understanding Kubernetes Storage
- Container Storage Initiative (CSI) plugin model
- My Experience Writing the AFS/AuriStor CSI plugin

The Kubernetes Storage Journey

- Take 1: Containers are Cattle, who needs persistent storage?
- Take 2: Storage was statically created ‘Persistent Volumes’
  - Hardcoded and in-tree
  - Terrible idea
  - Lots of junk leftover both code and ‘API’ stuff that is being culled
- Take 3: Flex Volumes
  - Kludgy idea while the standards were being fleshed out
- Take 4: Container Storage Initiative (CSI)

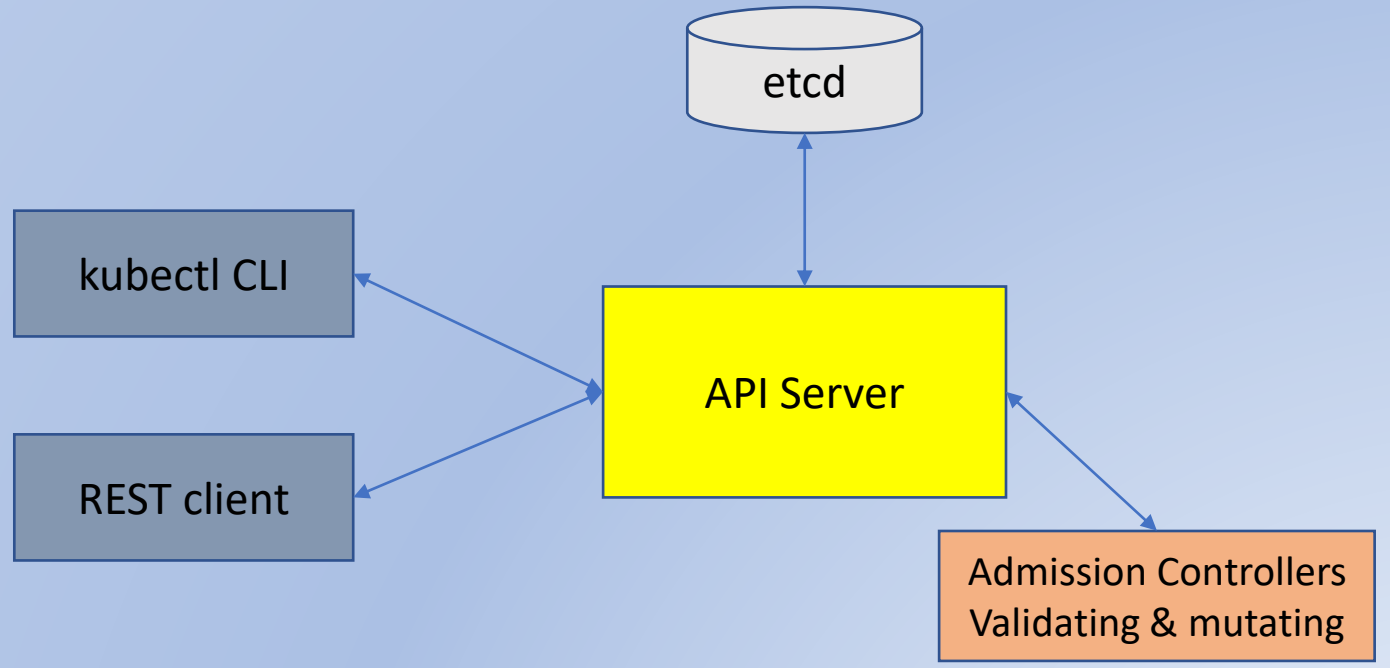
Kubernetes Does not have an Imperative API

- Imperative APIs aren’t suited for resource orchestration
  - Things may take a long time to schedule, set up and make available
- It makes more sense to have a Declarative API
  - ‘Objects’ specify the state they would like to be in
  - Expressed in JSON/YAML
- Kubernetes API Server
  - API Objects are posted, updated or deleted
  - Validates API ‘Objects’ and access control for user action.
- Software Controllers and Operators
  - Help bring objects to their desired specification
  - May involve creating, modifying or deleting API Objects

Kubernetes API Objects (53 different kinds in k8s 1.17)

- Container Creators
  - **Pods**, Deployments, **StatefulSets**, DaemonSets, ReplicaSets, CronJobs, Jobs
- Storage
  - **PersistentVolumeClaims, PersistentVolumes, VolumeAttachments, StorageClasses**
  - **CsiDrivers, CsiNodes**
  - **VolumeSnapshots, VolumeSnapshotClasses, VolumeSnapshotContents**
- General
  - ConfigMaps, NameSpaces, Nodes, CustomResourceDefinitions, **Events**
- Networking
  - Endpoints, Service, EndpointSlice, Ingress
- Security
  - Secret, ServiceAccount, ClusterRole, ClusterRoleBinding, Role, RoleBinding

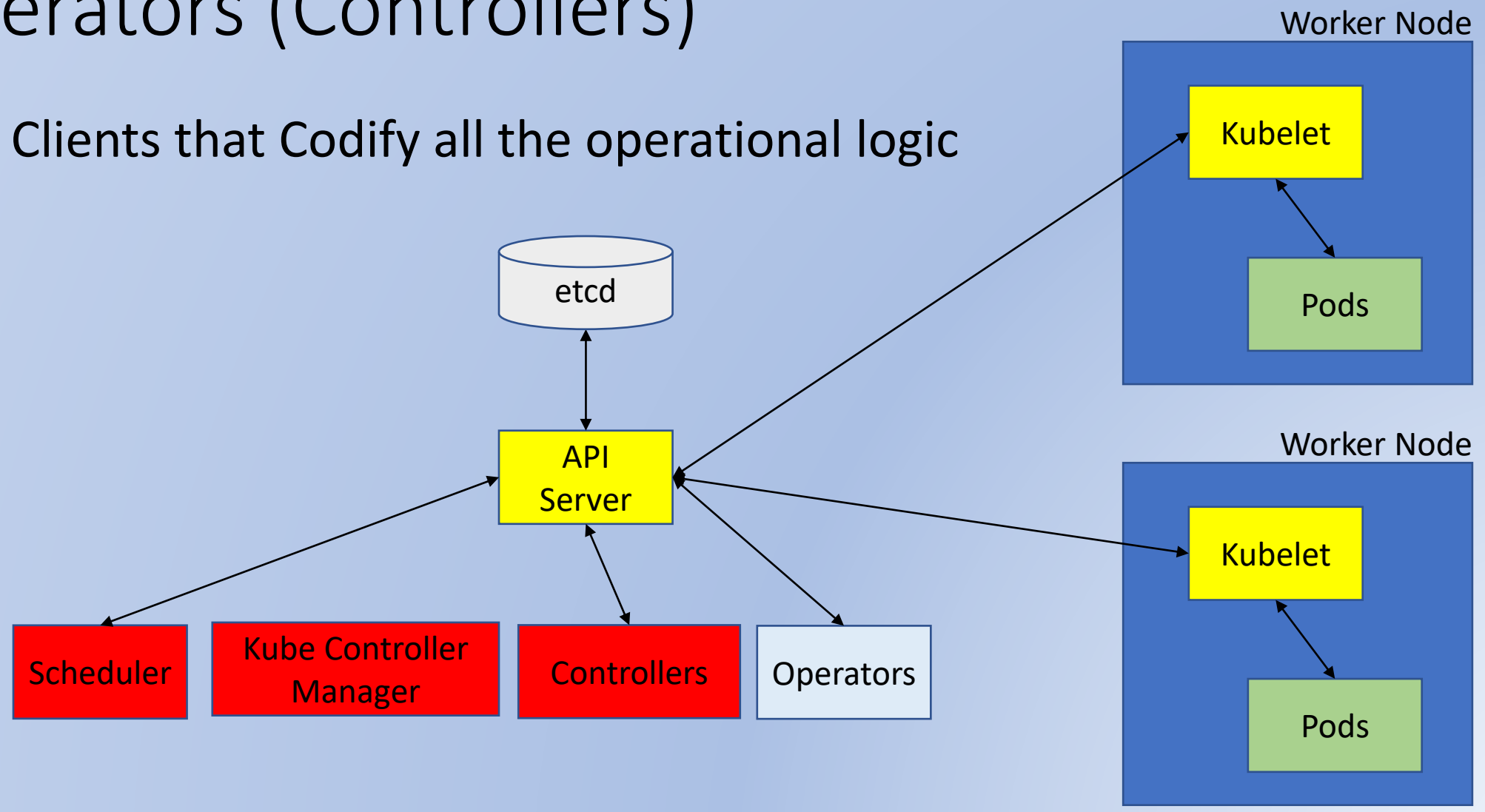
Kubernetes API Server



- Restful API
  - Post/Watch/Update/Delete Kubernetes Objects
  - Backed by etcd, resilient Key/Value Store
- CLI tool
  - kubectl
- Admission Controllers
  - Validate API Object
  - Mutate API Object (ie defaults)

Operators (Controllers)

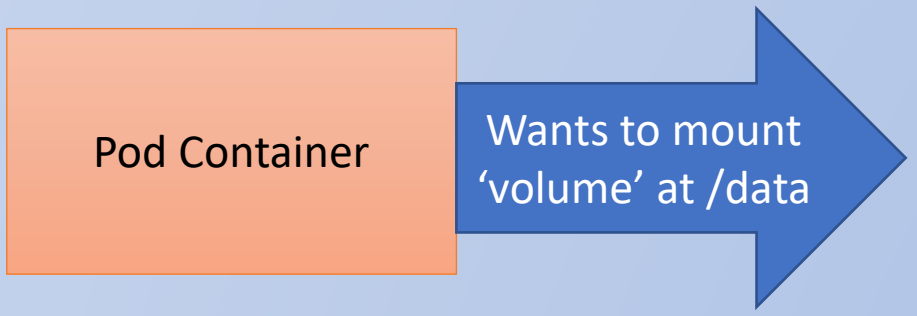
- API Clients that Codify all the operational logic



No Container Objects -- Pods

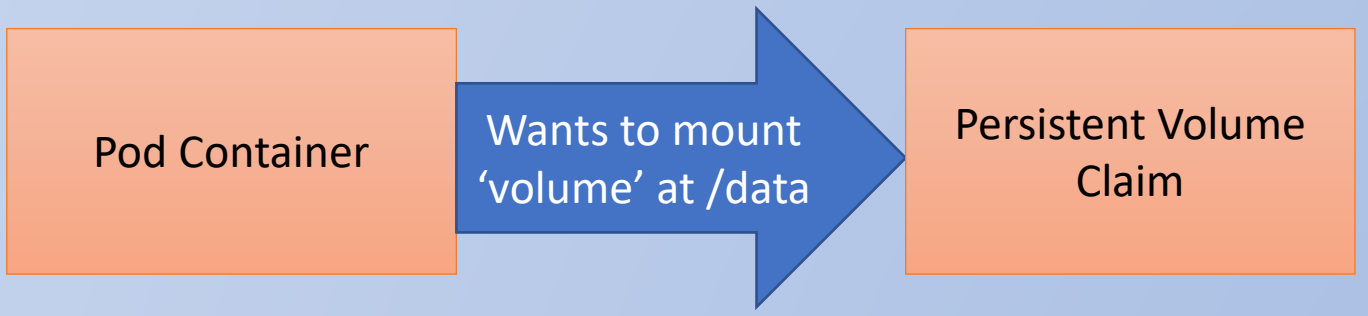
- Container Creators
  - **Pods**, Deployments, **StatefulSets**, DaemonSets, ReplicaSets, CronJobs, Jobs
- Pods contain one or more Containers
  - Share Localhost
  - The same volume can be mounted to 1+ (in same or different place)

Not Quite Simplest Example




```
kind: Pod
apiVersion: v1
metadata:
  name: my-csi-app
spec:
    containers:
    - name: my-frontend
      image: busybox
      volumeMounts:
      - mountPath: "/data"
        name: my-csi-volume
      command: [ "sleep", "1000000" ]
  volumes:
    - name: my-csi-volume
      persistentVolumeClaim:
        claimName: csi-pvc
```

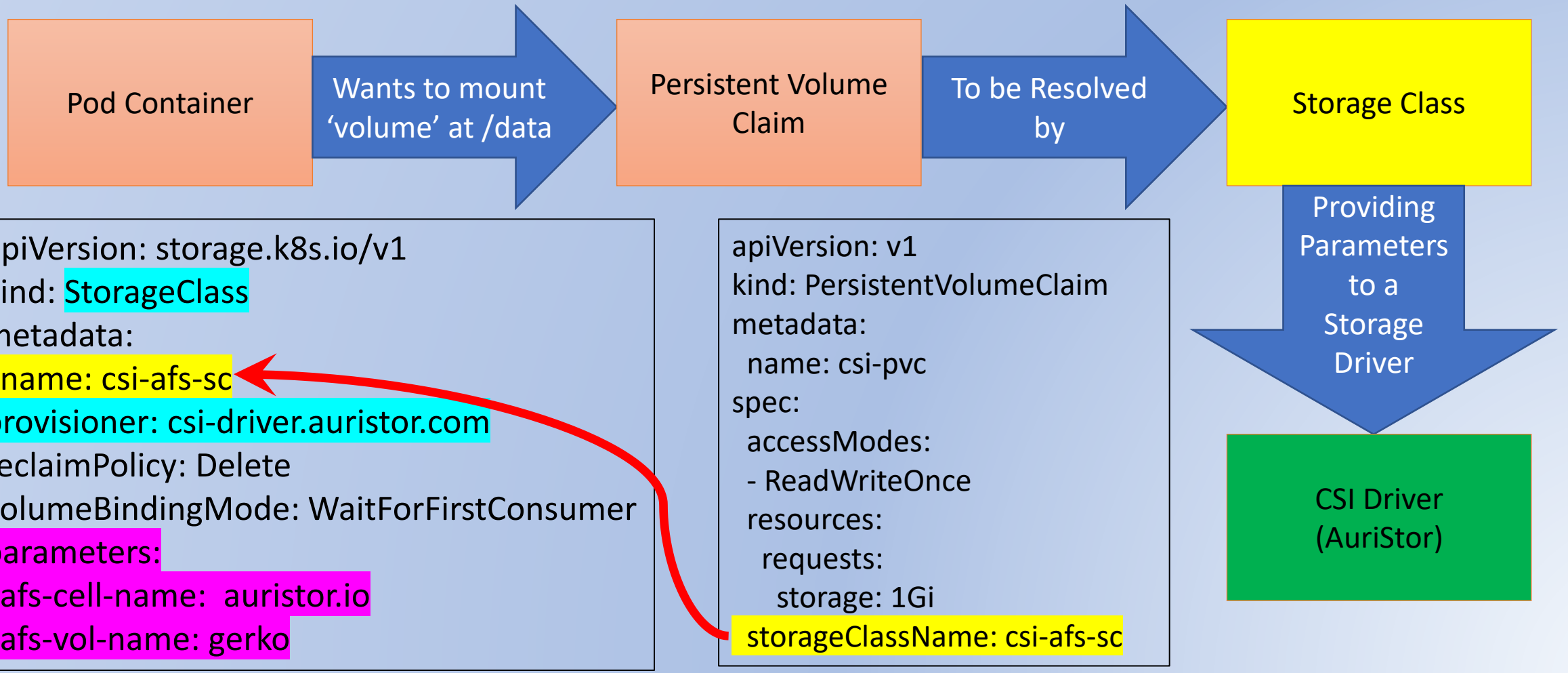
What Kind of Volume does the end-user want?



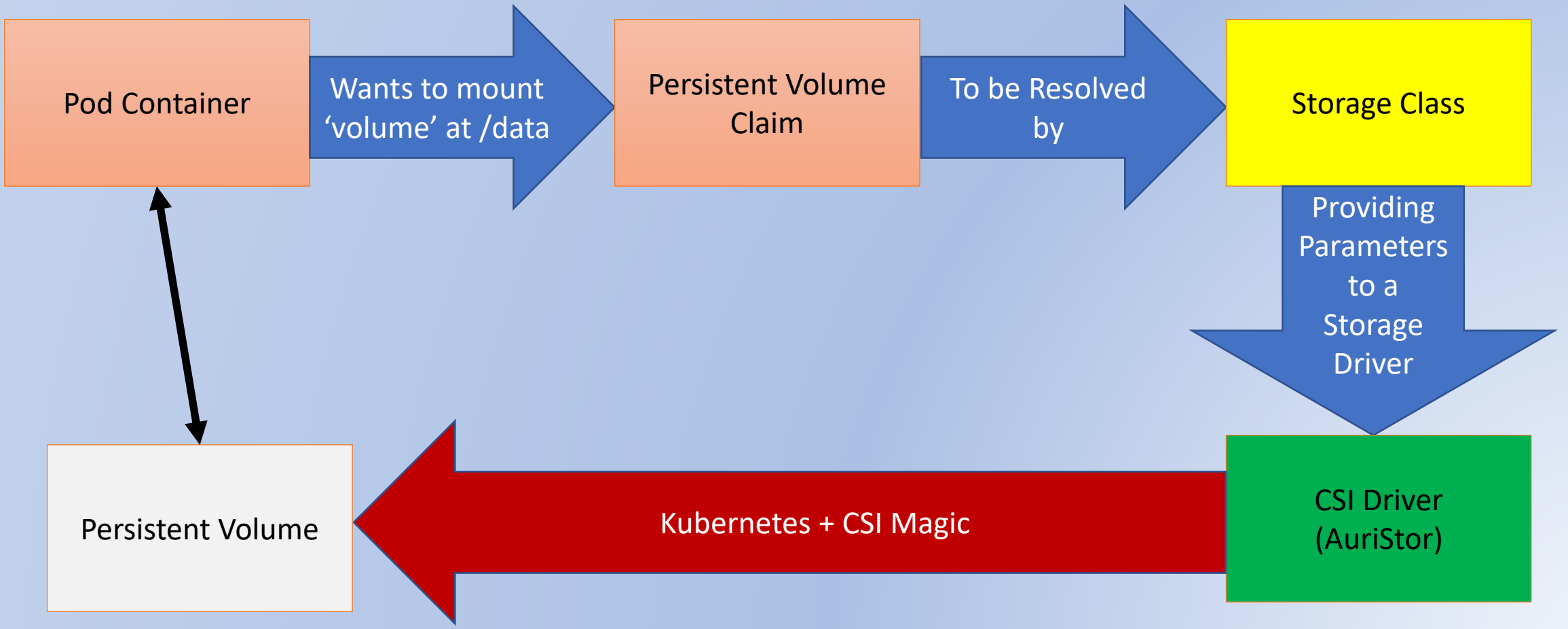

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-afs-sc
```

```
kind: Pod
apiVersion: v1
metadata:
  name: my-csi-app
spec:
    containers:
    - name: my-frontend
      image: busybox
      volumeMounts:
      - mountPath: "/data"
        name: my-csi-volume
      command: [ "sleep", "1000000" ]
  volumes:
    - name: my-csi-volume
      persistentVolumeClaim:
       claimName: csi-pvc
```

What the Cluster (Storage) admin Specifies



PersistentVolume Object: CSI + Driver Magic



CSI Driver Responsibility

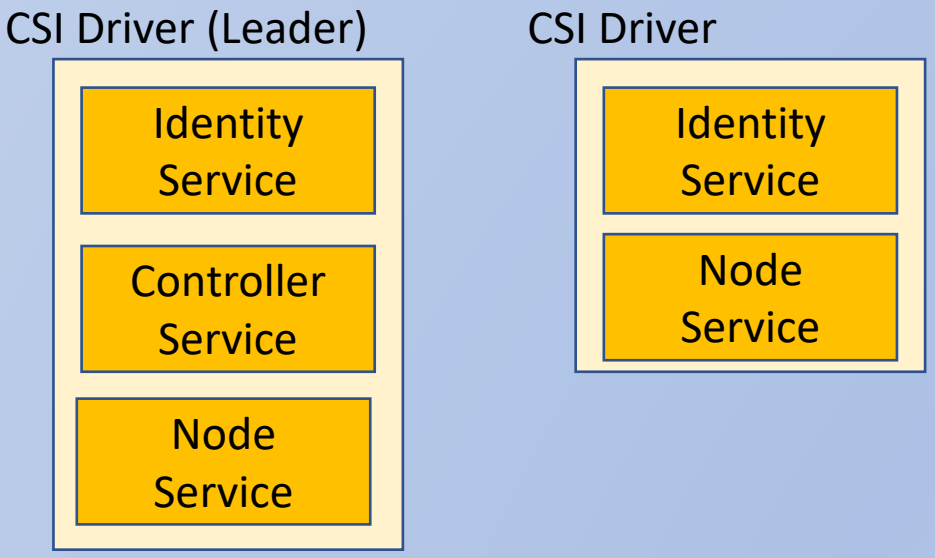
- CSI Driver on each node that would provide this kind of storage
  - Typically as a privileged Kubernetes Pod
  - Provides 3 services: accessible via Unix Domain Sockets as gRPC Service

- Identity Service – must be on each Node that can will use this driver
  - Informs Node about the instance and Driver capabilities
    - Ie Is Storage Topology Aware Pod Scheduling supported?

- Controller Service
  - Makes higher level decisions. Doesn’t have to run on a worker node

- Node Service
  - Must run on every node that will use this driver

Controller Service and Node Service



ControllerService

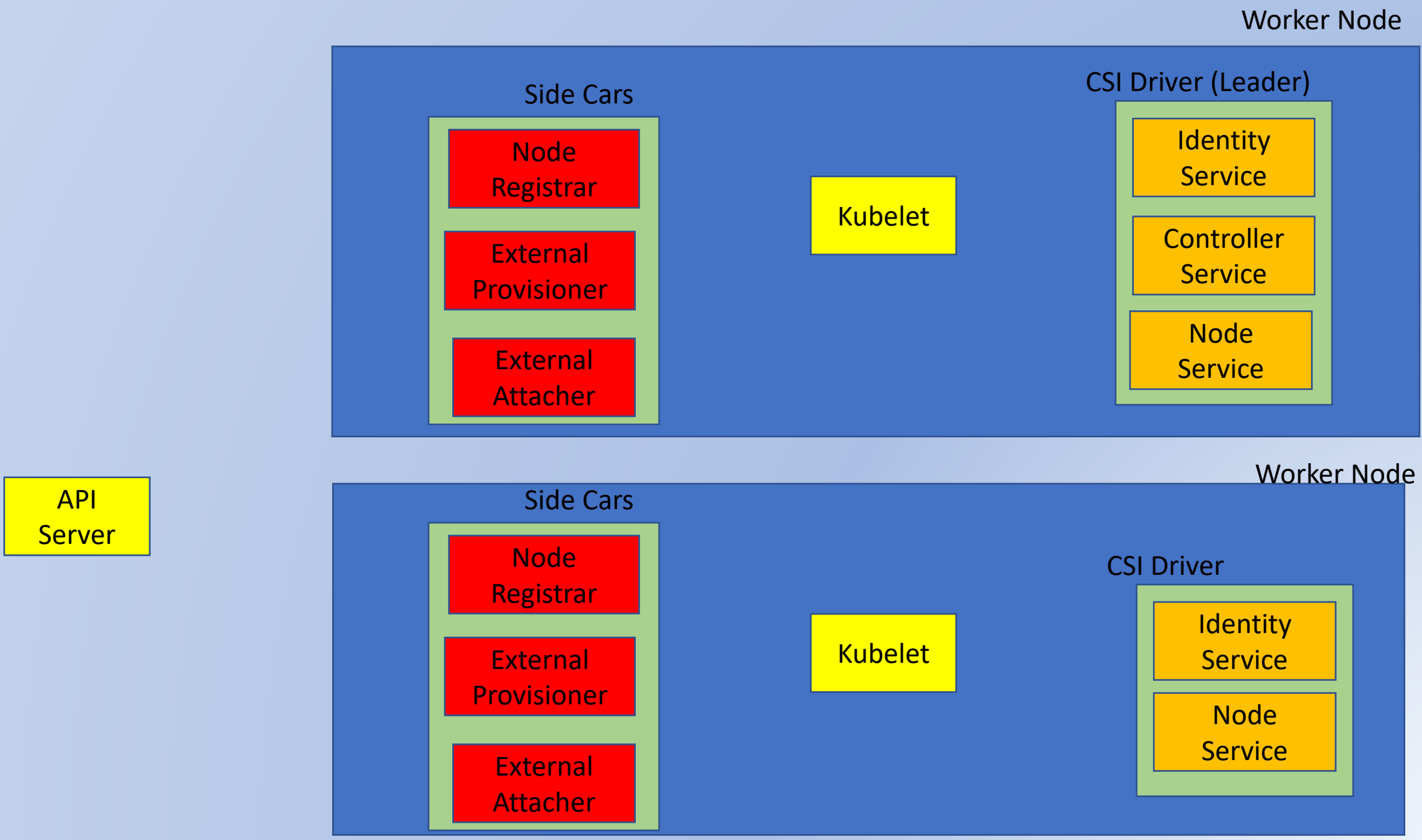
- CreateVolume()
- PublishVolume()
- UnpublishVolume()
- DeleteVolume()

NodeService

- NodeStageVolume()
- NodePublishVolume()
- NodeUnstageVolume()
- NodeUnpublishVolume()

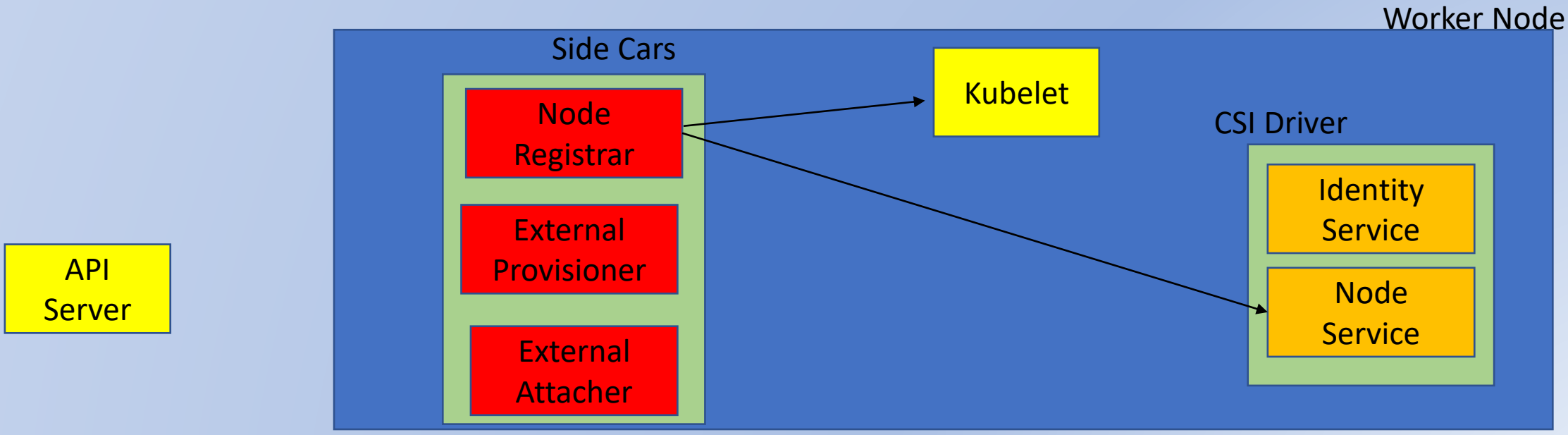
Contexts are passed along to future calls

- The CSI Driver is may be ephemeral (ie may crash and restart
  - The Sidecars cooperatively will retain ‘context’ data
- Volume Context
  - Created at Controller.createVolume time
    - Typically include copies of some of the ‘parameters’ (ie parameters in the storage class)
  - Augmented by CO for future calls
    - Future: publishVolume, unpublishVolme, deleteVolume
- Publish Context
  - Created at Controller.publishVolume
  - Future calls on both the Controller and Node Services will get the volumeContext
    - Controller NodeStageVolume, NodePublishVoume


Worker Node
Side Cars
Node Registrar
External Provisioner
External Attacher
Kubelet
CSI Driver (Leader)
Identity Service
Controller Service
Node Service
API Server
Worker Node
Side Cars
Node Registrar
External Provisioner
External Attacher
Kubelet
CSI Driver
Identity Service
Node Service

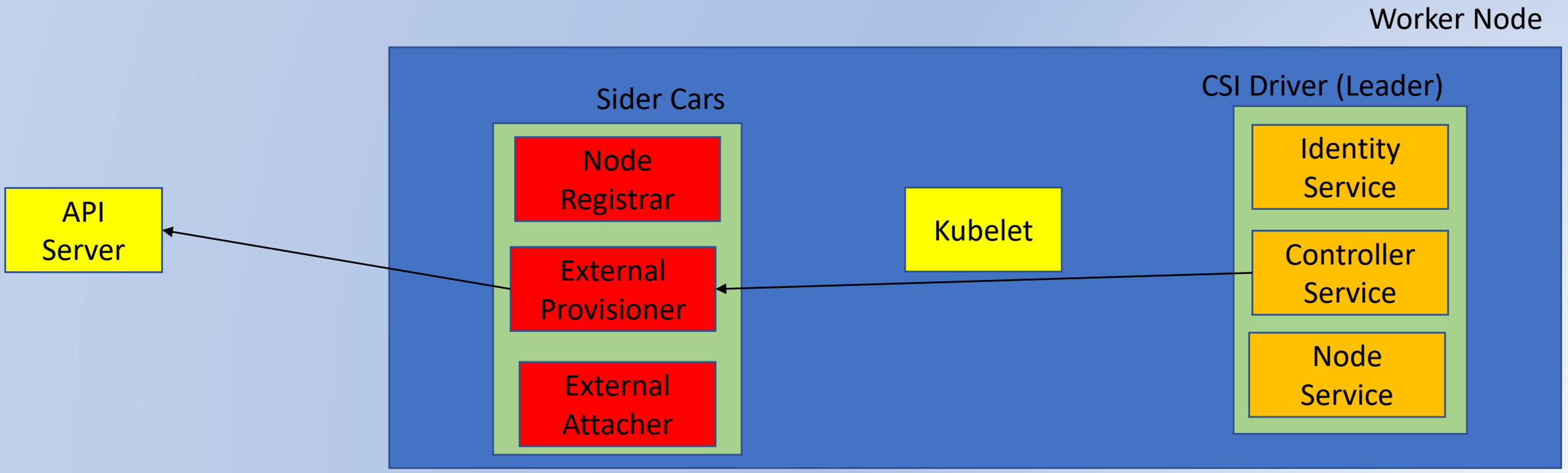
Node Registration (All Nodes)

- Gets Node Info from NodeService
- Register via kublet



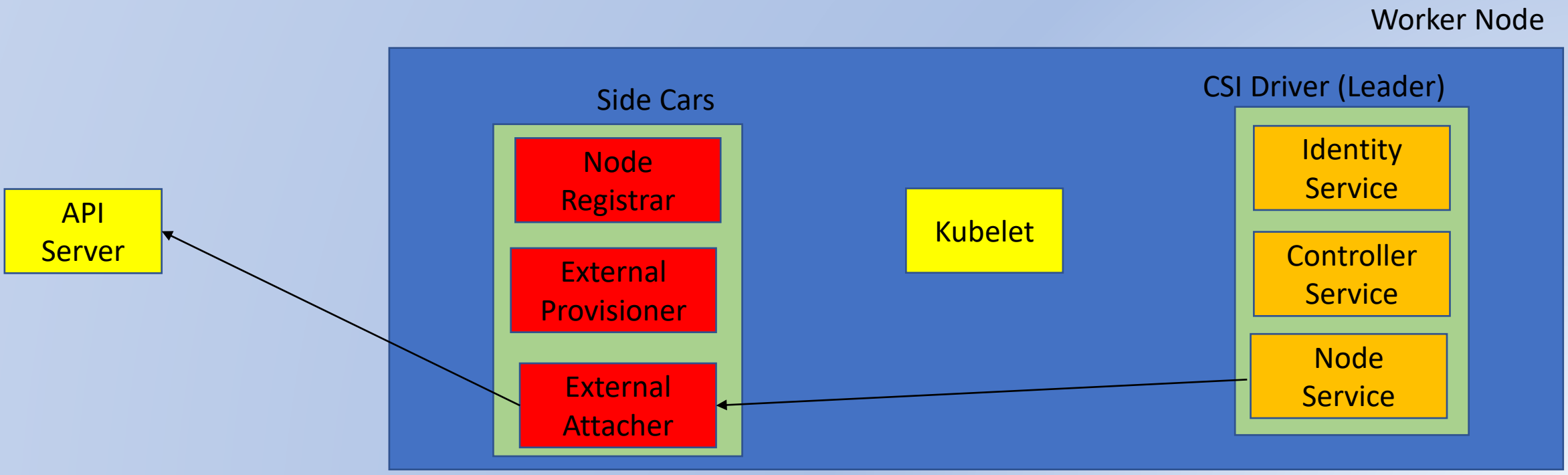
Desire to Create Storage

1. External Provisioner ‘notices’ a PVC (via API Server) referencing a storageClass Object
2. External Provisioner ‘notices matching StorageClass object is for this ‘provisioner’ (CSI Driver)
3. Asks Driver ControllerService to ‘create Volume’
4. Creates Kubernetes Persistent Volume Object



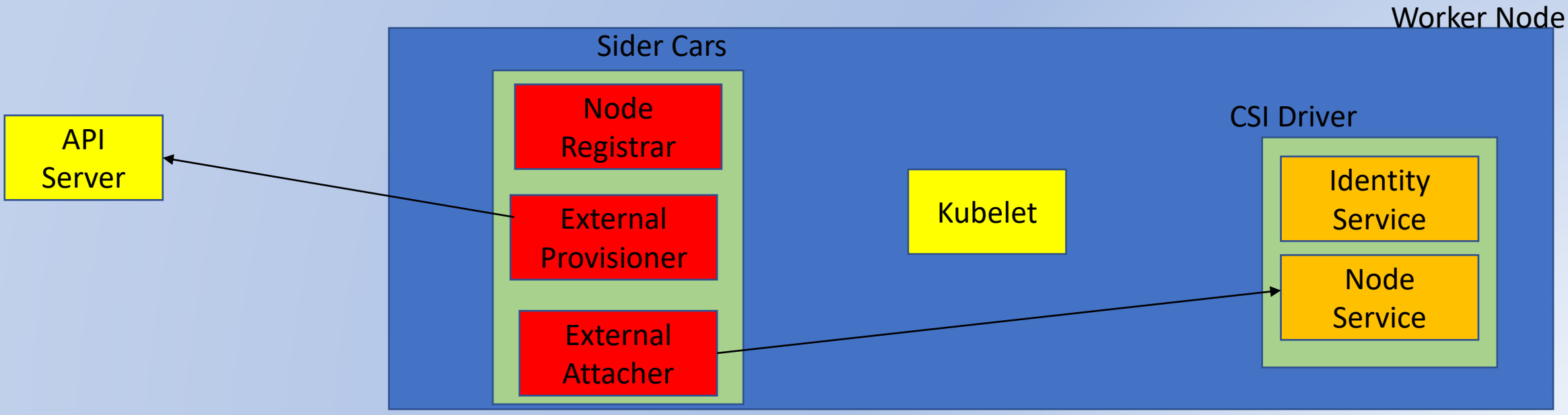
Scheduling

1. External Provisioner ‘notices’ a Pod is Scheduled
2. Asks Driver ControllerService to ‘publish Volume’

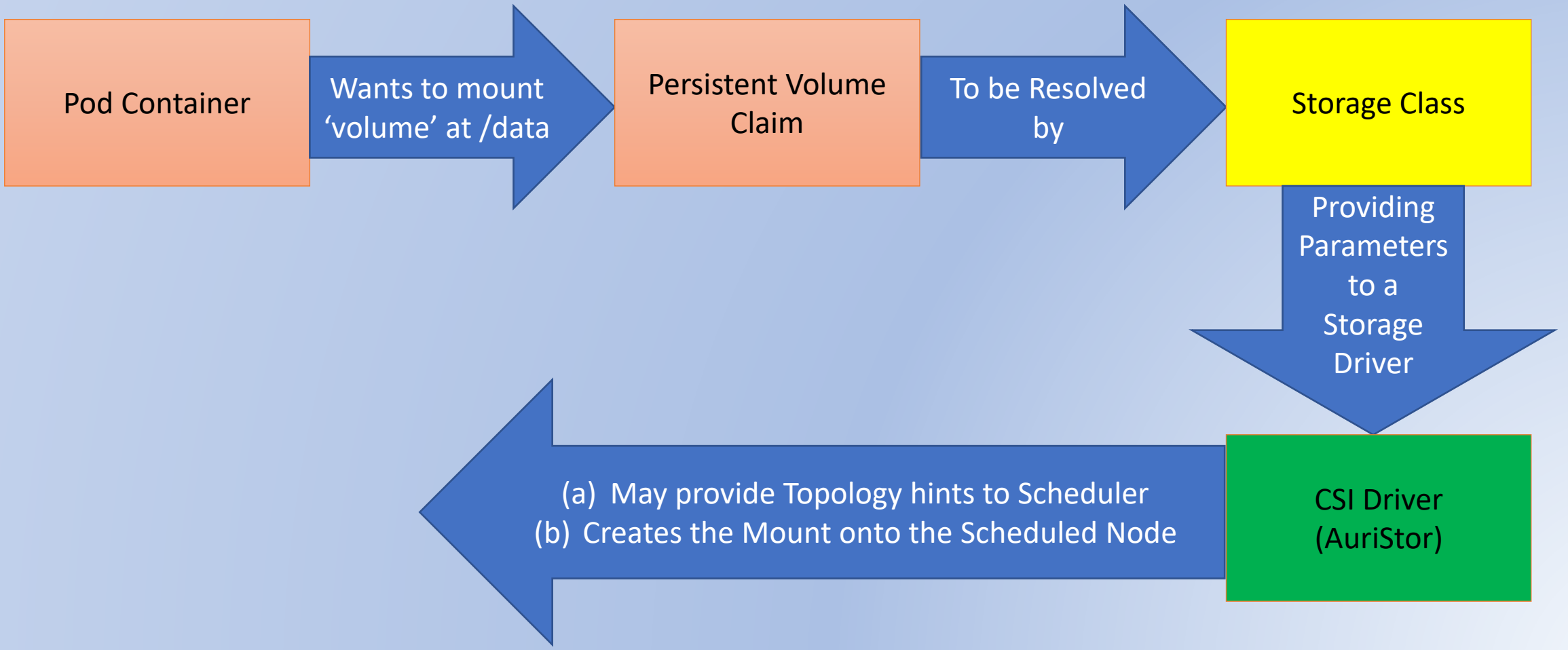


Attaching

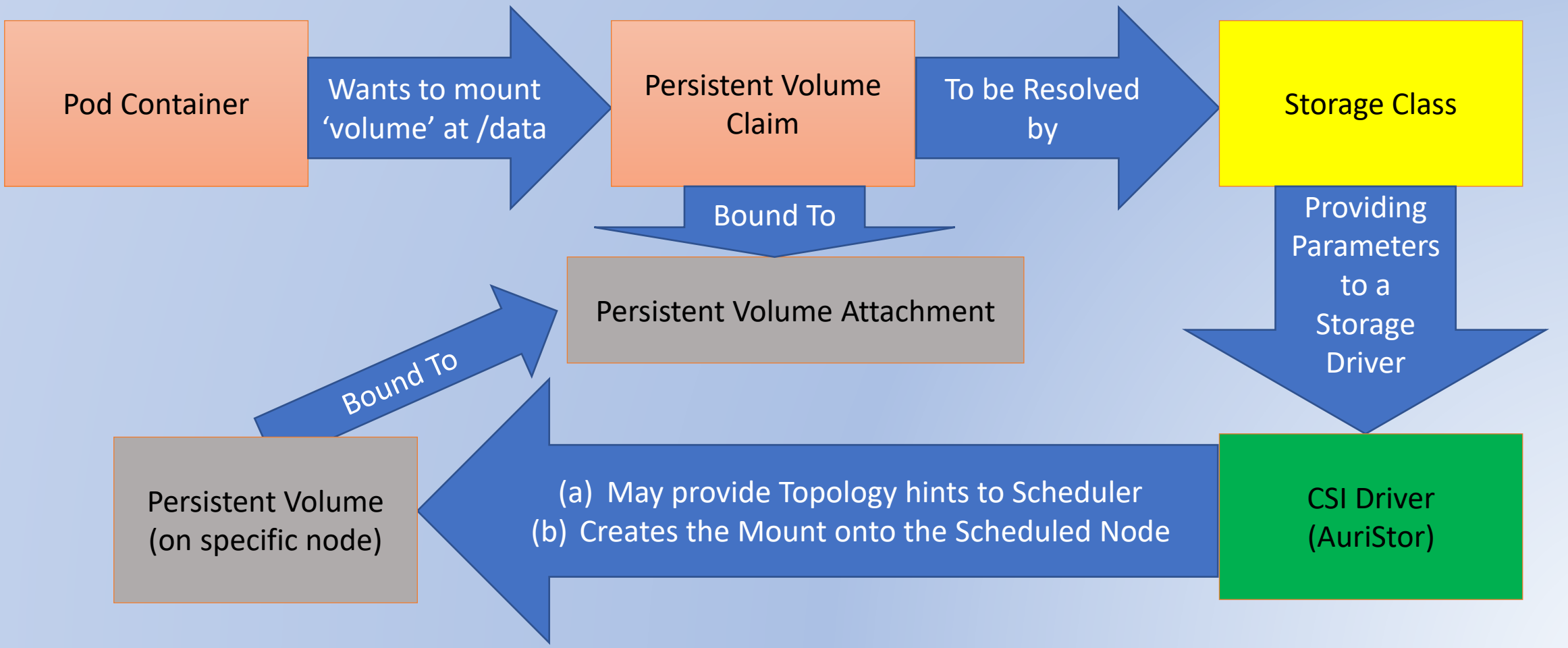
1. Looks for Volume Attachments (Pairing of PV and PVC)
2. Node Stage
3. Node Publish



What the Kubernetes CSI Driver Does



Upon Scheduling the External Attacher does…



AuriStor / AFS

- Andrew File System (AFS)
  - What is it
    - Secure Distributed File System
      - AuriStor adds Combined Identity ACL
    - WAN/LAN optimized with local cache consistency
    - Zero client configuration Global Namespace
      - cat /afs/umich.edu/README
    - Platform Independent
  - Unit of Management and Policy is an AFS Volume
    - Rooted Directory tree that can be mounted locally
    - Can be mounted to the AFS global namespace
    - Example Policy is replicas

AuriStor CSI Storage Class

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
        name: csi-afs-sc
provisioner: afs.csi.auristor.com
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
parameters:
    afs.csi.auristor.com/volumeType: "normal"    # scratch, normal, createIfDoesntExist
    afs.csi.auristor.com/cellName:  "auristor.io"
    afs.csi.auristor.com/volumeName: "allPlay"
    afs.csi.auristor.com/VolumeProximityRequired: "true"
    afs.csi.auristor.com/schedulingDeferralType: "immediate"     # options: immediate, eventual
    afs.csi.auristor.com/scratchVolumePrefix: "myExperiment"
```

Understanding Kubernetes Storage

Getting in Deep by Writing a CSI Driver

Thank you!

Gerry Seidman
gerry@auristor.com

CSI Specification: https://github.com/container-storage-interface/spec/blob/master/spec.md
Kubernetes CSI Documentation: https://kubernetes-csi.github.io/docs/
AuriStorFS: https://www.auristor.com/filesystem

AURISTOR
YOUR DATA · YOUR FILE SYSTEM