CKS Enhancements in CloudStack

CloudStack's Container Kubernetes Service (CKS) is evolving rapidly. These enhancements aim to streamline container orchestration, improve scalability, and boost overall performance. Let's explore the key developments.

P by Pearl Dsilva

About Me: **Pearl Dsilva**

SOFTWARE ENGINEER @ SHAPEBLUE

APACHE CLOUDSTACK COMMITTER

BEEN PART OF THE COMMUNITY SINCE 2019

Email: pearl11594@apache.org

Github: Pearl1594

Glossary

CAPI - Cluster API

A declarative API for managing Kubernetes clusters across various environments, such as on-premises, cloud platforms, and edge deployments.

CAPC - Cluster API for CloudStack

A CAPI provider that enables you to deploy and manage Kubernetes clusters on CloudStack, a widely-used open-source cloud platform.

CKS - CloudStack Kubernetes Service

A managed Kubernetes service offered by CloudStack for simplified deployment and management, providing an easy way to run containerized applications on CloudStack.

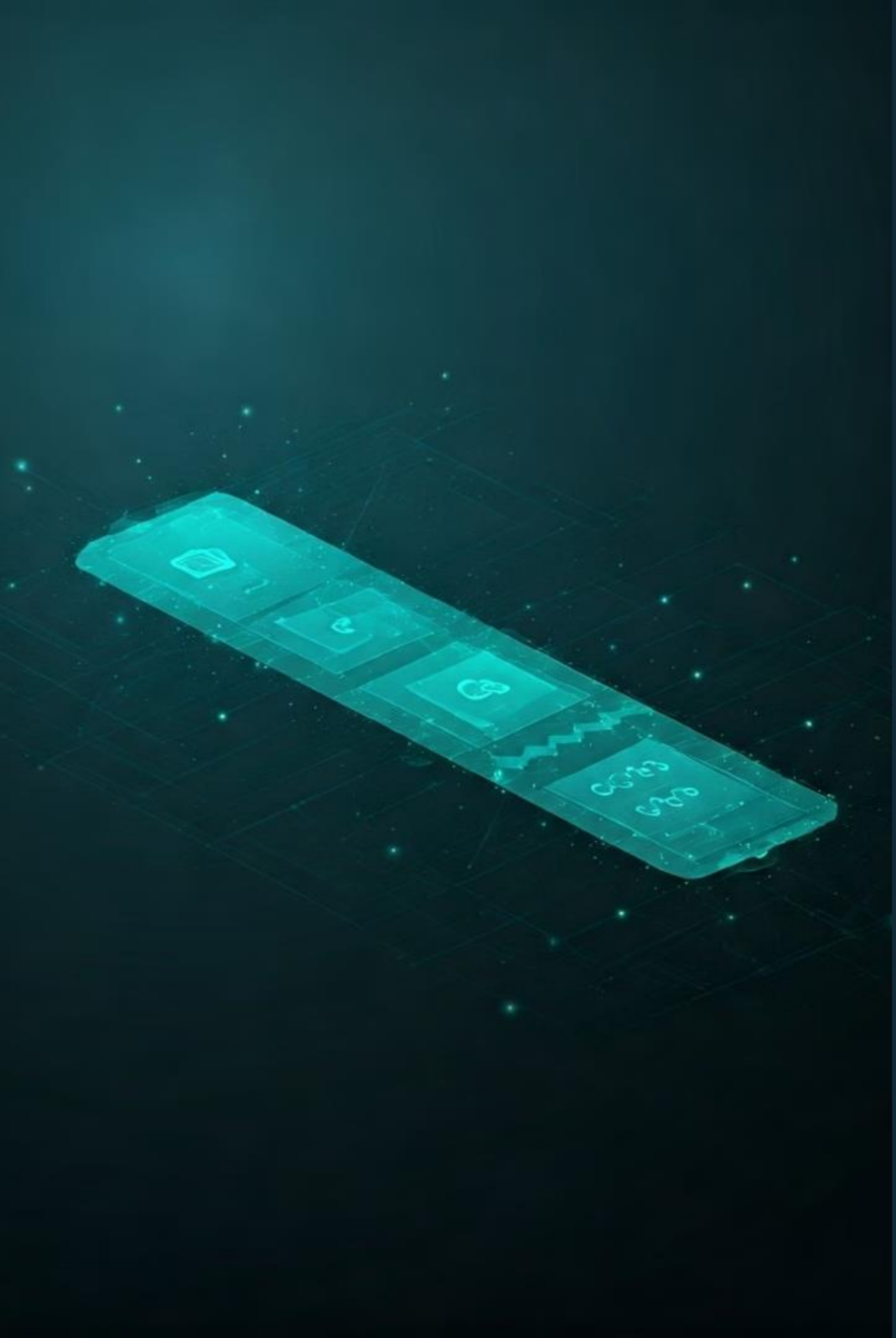
CNI - Container Network Interface

A specification that defines how containers connect to networks and communicate with each other, enabling them to share resources and collaborate.

CSI - Container Storage Interface

A specification for how containerized applications interact with storage systems, allowing them to access data from a variety of sources.

#CSCollab2024



Evolution of K8S in ACS

1. Initial Release - CCS (2016)
2. Introducing CKS (ACS 4.14 - Nov 2021)
3. Debian-based CKS, Auto-scaling, CloudStack Kubernetes Provider (ACS 4.16 - Jan 2021)
4. Introducing CAPC - (2022)
5. **CAPC as unmanaged CKS clusters**
   Enhancing CKS flexibility
   Support for CloudStack CSI - (2025)
6. What Next - You tell us!!

CAPC as Unmanaged CKS Clusters

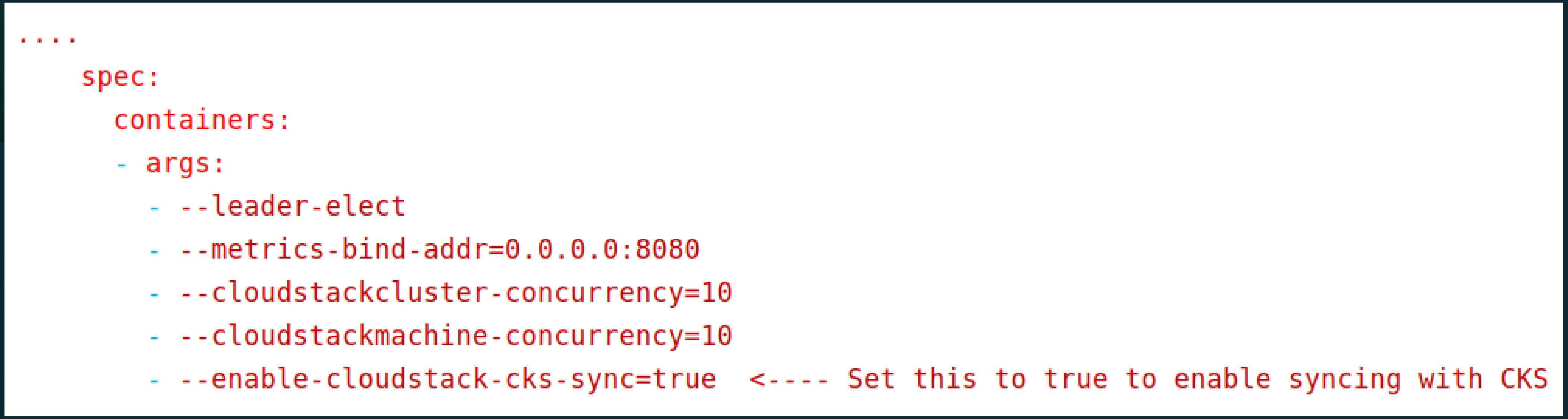
Centralized View

Comprehensive view of CAPC resources in CloudStack

Resource Allocation

Easily monitors resources across multiple clusters.

export CAPC_CLOUDSTACKMACHINE_CKS_SYNC=true



```
....
    spec:
      containers:
      - args:
        - --leader-elect
        - --metrics-bind-addr=0.0.0.0:8080
        - --cloudstackcluster-concurrency=10
        - --cloudstackmachine-concurrency=10
        - --enable-cloudstack-cks-sync=true  <---- Set this to true to enable syncing with CKS
```

K8S Node Types

	Control Node	Worker Node	Etcd Node
Description	Manages overall state and configuration of the k8s cluster	Compute nodes that run applications deployed to the cluster	Distributed key-value store that stores all Kubernetes cluster state and configuration data
Recommended Specifications	**CPU**: 2-4 cores **Memory**: 8 – 16GB RAM **Storage**: minimum of 50 GB (more if etcd is running on the control plane)	**CPU**: 4-16 cores, based on workload requirements **Memory**: 16-64 GB RAM or more **Storage**: at least 100 GB or more depending on workload storage needs	**CPU**: 2-4 cores **Memory**: 8-16 GB RAM **Storage**: Fast SSD storage, at least 50 GB (low-latency disk I/O is crucial for etcd performance)

Node Type Service Offerings

Standard Nodes

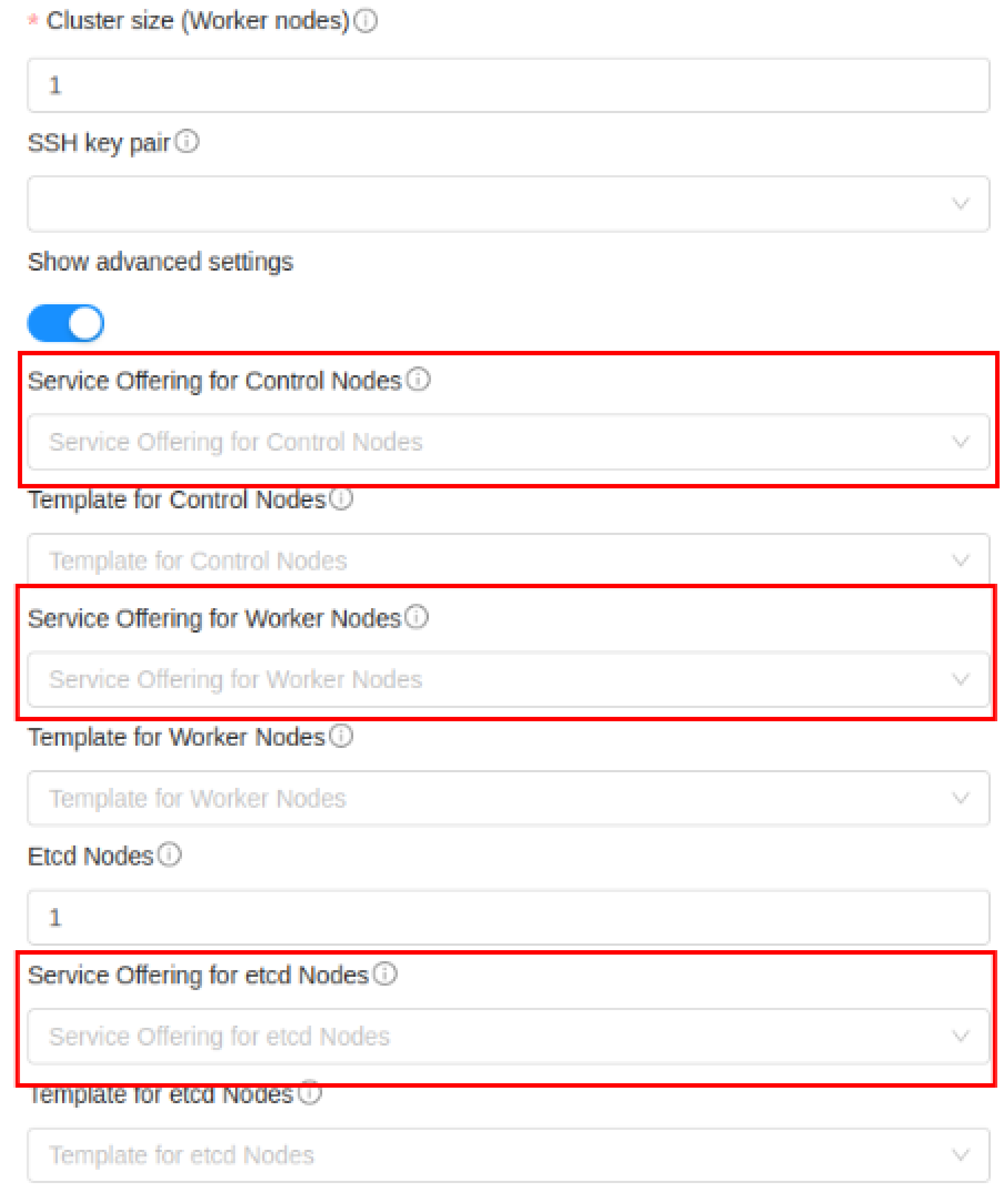
General-purpose nodes for typical workloads. Balanced CPU and memory resources.

High-Performance Nodes

Optimized for compute-intensive tasks. Enhanced CPU and GPU capabilities.

Memory-Optimized Nodes

Ideal for data-intensive applications. Large memory allocations for improved caching.



* Cluster size (Worker nodes) ⓘ

1

SSH key pair ⓘ

Show advanced settings

Service Offering for Control Nodes ⓘ

Service Offering for Control Nodes

Template for Control Nodes ⓘ

Template for Control Nodes

Service Offering for Worker Nodes ⓘ

Service Offering for Worker Nodes

Template for Worker Nodes ⓘ

Template for Worker Nodes

Etcd Nodes ⓘ

1

Service Offering for etcd Nodes ⓘ

Service Offering for etcd Nodes

Template for etcd Nodes ⓘ

Template for etcd Nodes

#CSCollab2024

Node Type Templates

1 **Customizable Configurations**

Create templates with pre-defined resource allocations and software stacks.

2 **Rapid Deployment**

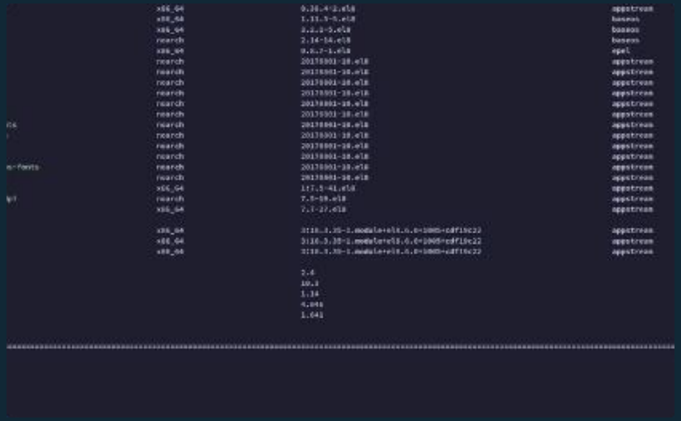
Quickly spin up new nodes using pre-configured templates.

3 **Version Control**

Maintain multiple versions of templates for different use cases.

#CSCollab2024

CKS Template Creation



Required Dependencies

The template installs necessary dependencies for CKS to function properly.

Cloud User Creation

The template creates the `cloud` user and adds it to the sudoers list for administrative privileges.

CKS Script Directory

The template creates the necessary directory - `/opt/bin` - to store CKS scripts.

SSH Key Setup

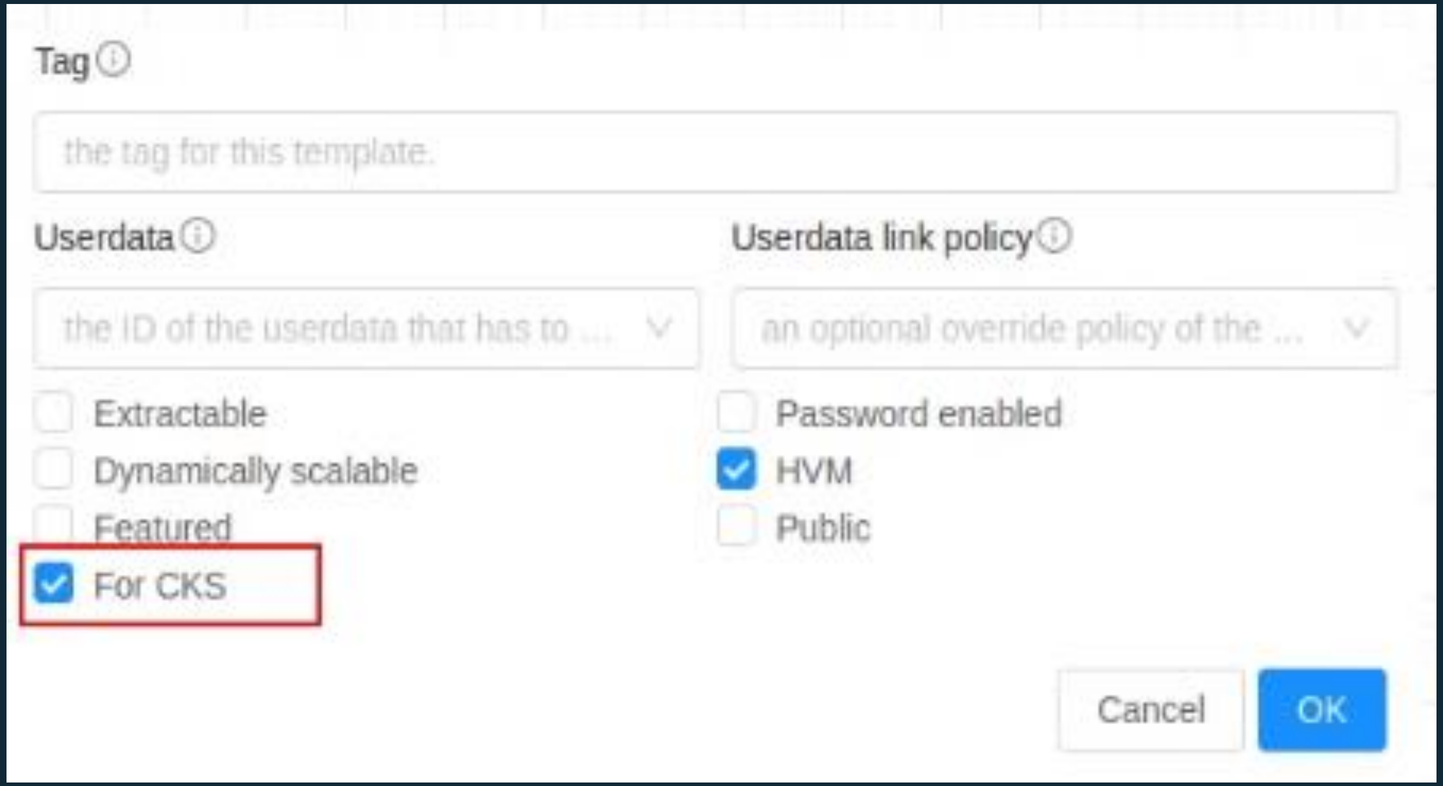
Any VM deployed with this template must have the required SSH key in the `cloud` user's `~/.ssh/authorized_keys` file for secure access.

Template Registration for CKS

Needs `forCks` flag to be true

CKS Template Registration

Register Template



Tag ⓘ

the tag for this template.

Userdata ⓘ

the ID of the userdata that has to ...

Userdata link policy ⓘ

an optional override policy of the ...

□ Extractable

□ Dynamically scalable

□ Featured

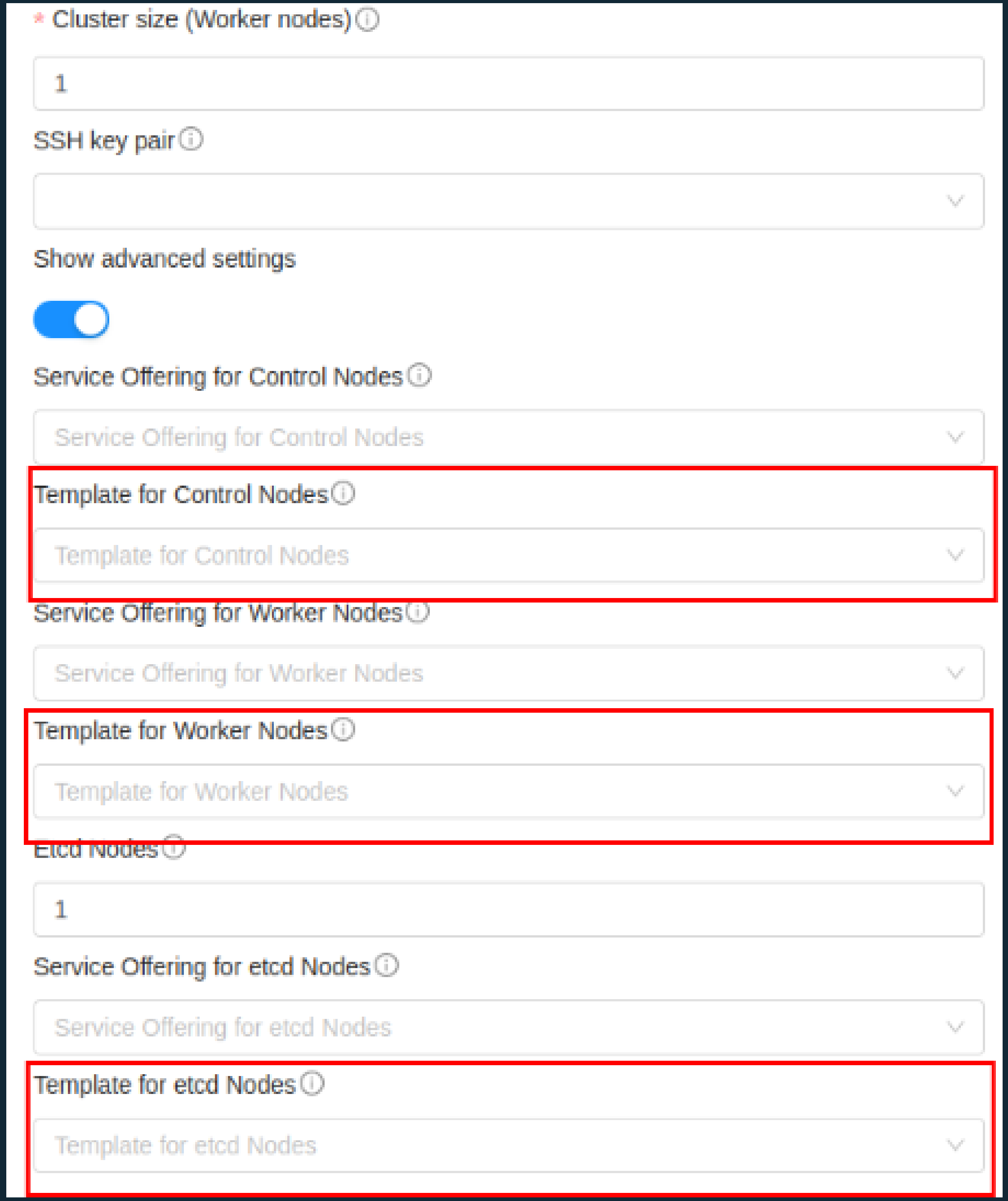
☑ For CKS

□ Password enabled

☑ HVM

□ Public

Create CKS cluster



* Cluster size (Worker nodes) ⓘ

1

SSH key pair ⓘ

Show advanced settings

Service Offering for Control Nodes ⓘ

Service Offering for Control Nodes

Template for Control Nodes ⓘ

Template for Control Nodes

Service Offering for Worker Nodes ⓘ

Service Offering for Worker Nodes

Template for Worker Nodes ⓘ

Template for Worker Nodes

Etcd Nodes ⓘ

1

Service Offering for etcd Nodes ⓘ

Service Offering for etcd Nodes

Template for etcd Nodes ⓘ

Template for etcd Nodes

#CSCollab2024

Importing External Node in CKS Clusters

Node Validation

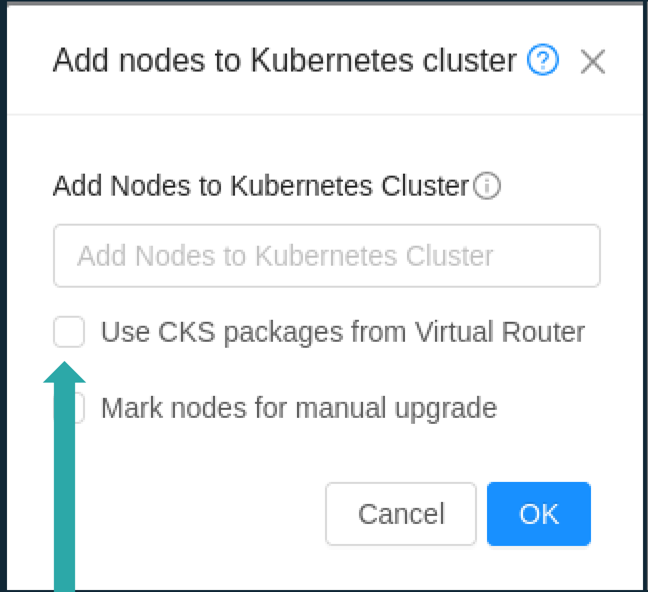
Firewall and DNAT

Reboot Node with Userdata

ISO Attachment

Requirements to import nodes

- Template needs to be registered as a CKS template
- Default NIC needs to be on the CKS cluster network
- Should not already be part of the cluster



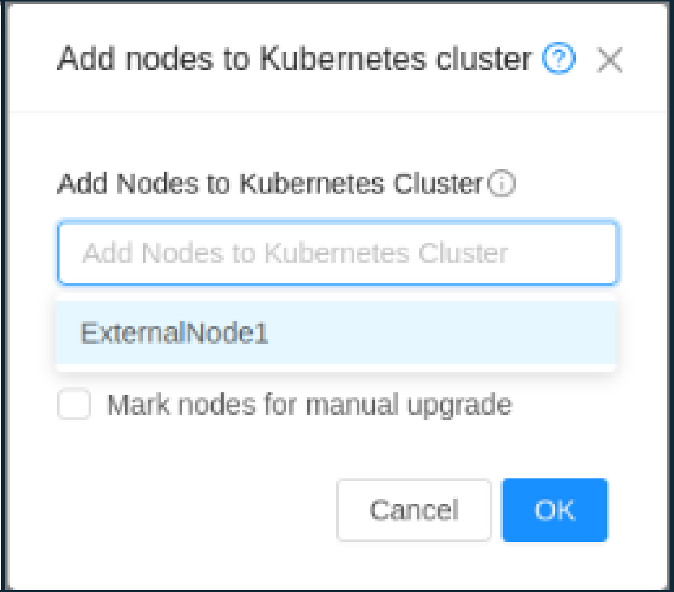
Add nodes to Kubernetes cluster ⓘ ✕

Add Nodes to Kubernetes Cluster ⓘ

Add Nodes to Kubernetes Cluster

□ Use CKS packages from Virtual Router

□ Mark nodes for manual upgrade



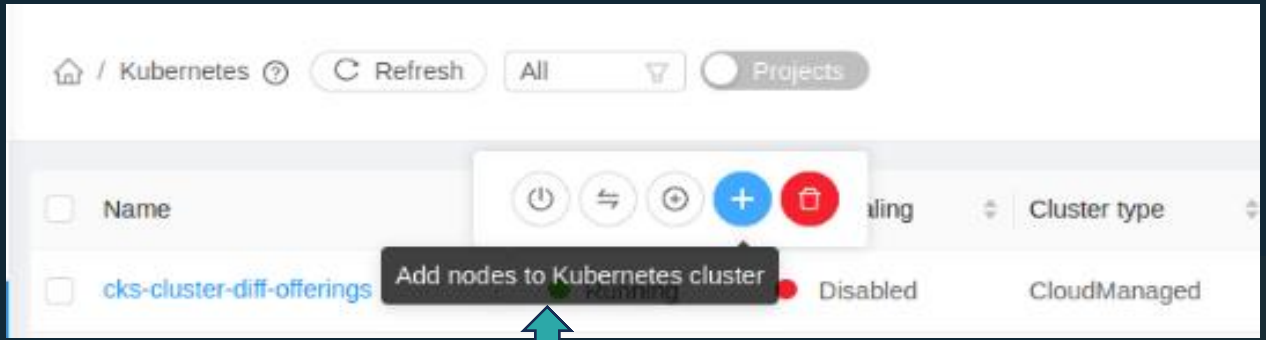
Add nodes to Kubernetes cluster ⓘ ✕

Add Nodes to Kubernetes Cluster ⓘ

Add Nodes to Kubernetes Cluster

ExternalNode1

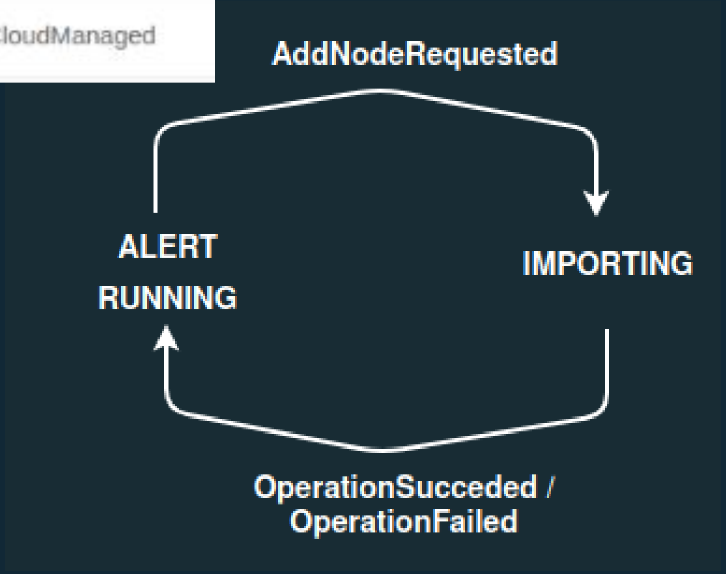
□ Mark nodes for manual upgrade



Name	aling	Cluster type
cks-cluster-diff-offerings	Disabled	CloudManaged

Add nodes to Kubernetes cluster

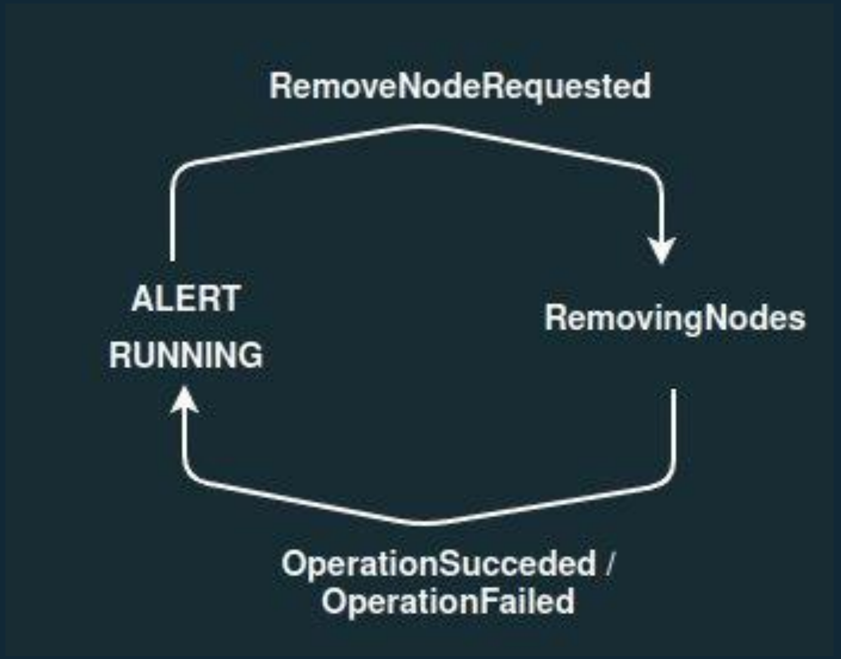
Add external Node to Cluster



- ***Legacy way –*** *attach CKS iso to the cluster nodes*
- ***mountcksisoonvr –***
  - *Specifically added to support baremetal nodes*
  - *ISO is mounted to the network's VR and served via http to the nodes*

#CSCollab2024

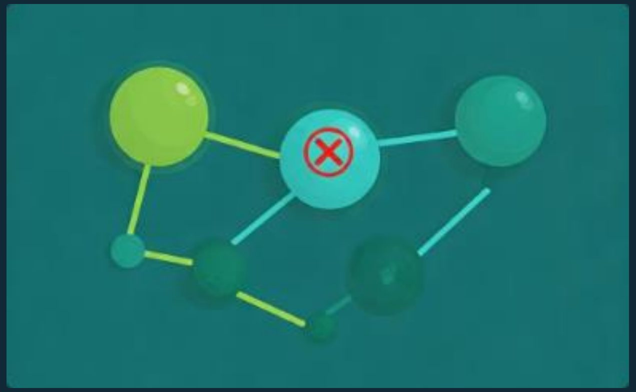
Removing external nodes from CKS



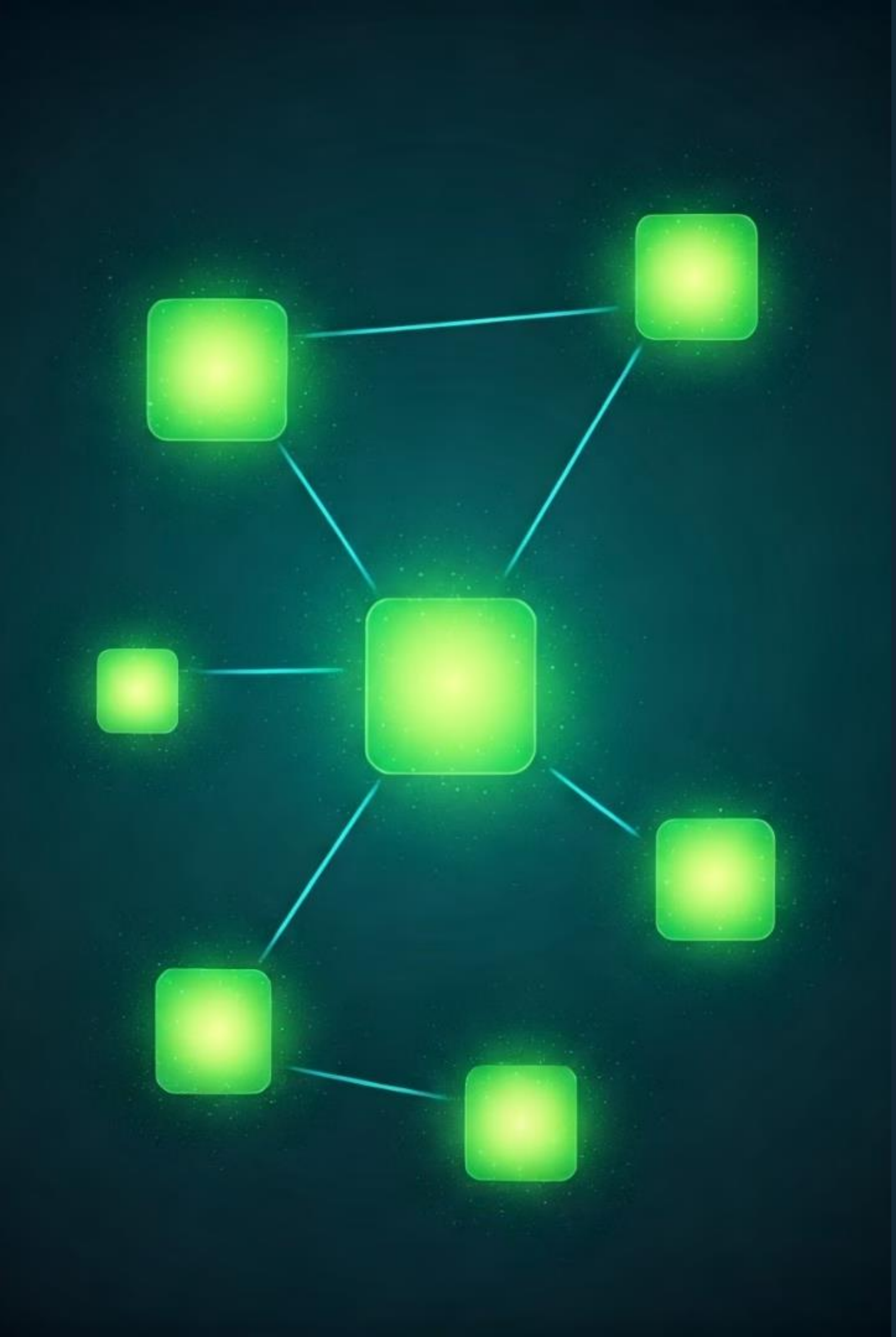
Drain the node

Reset the node

Delete the node


Remove DNAT and Firewall Rules

#CSCollab2024



External etcd Cluster

Improved Reliability

Separating etcd from control plane enhances system stability.

Enhanced Scalability

Independent scaling of etcd and control plane components.

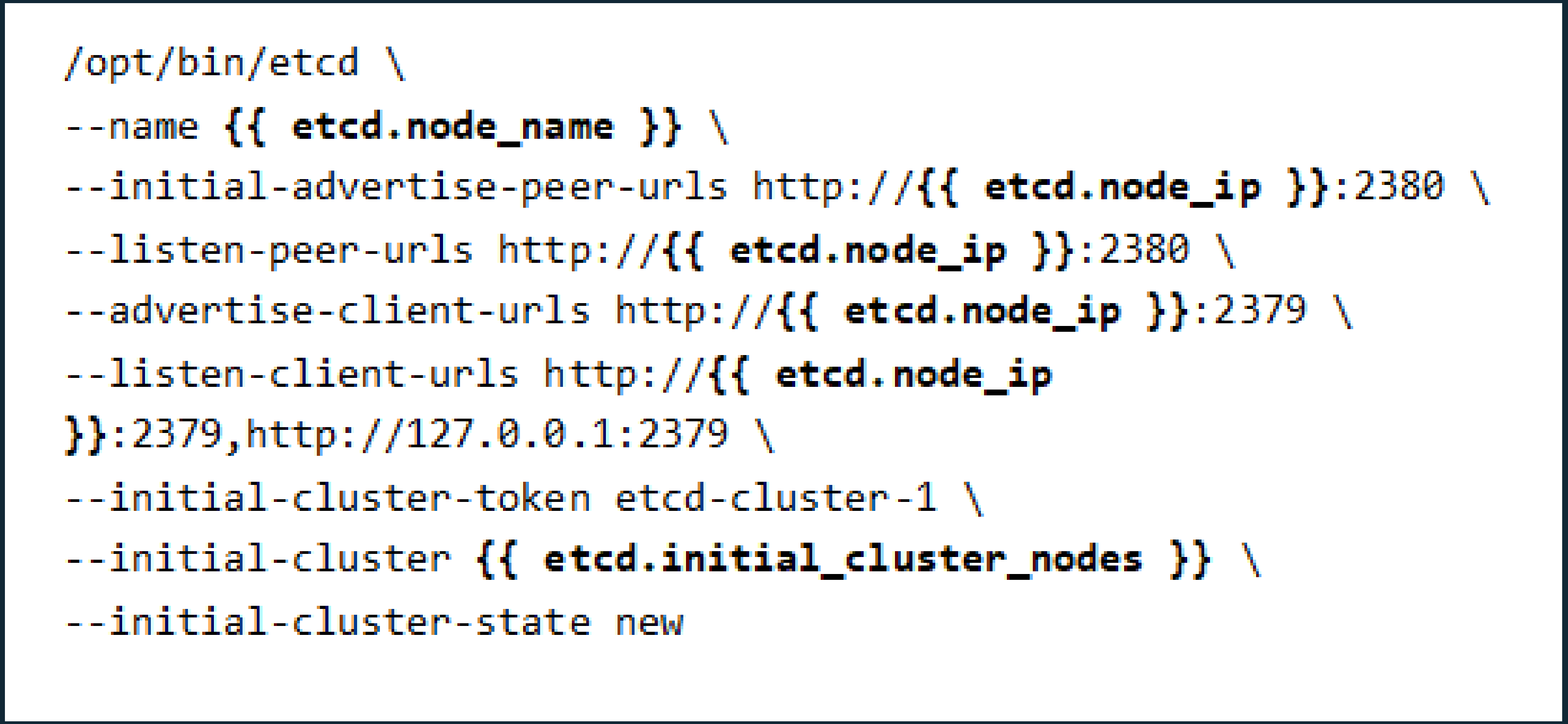
Increased Security

Isolation of etcd reduces attack surface and improves data protectio

#CSCollab2024

External etcd Cluster – contd…

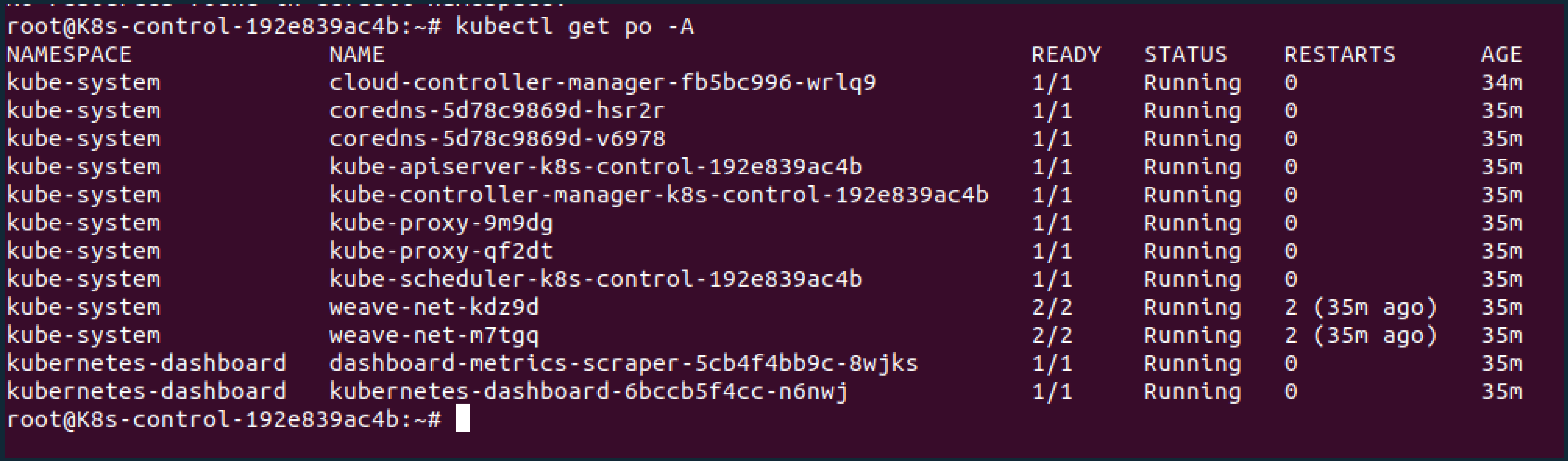
- CKS ISO is attached to the node and the etcd binaries are installed from it
- Etcd service is started:



```
/opt/bin/etcd \
--name {{ etcd.node_name }} \
--initial-advertise-peer-urls http://{{ etcd.node_ip }}:2380 \
--listen-peer-urls http://{{ etcd.node_ip }}:2380 \
--advertise-client-urls http://{{ etcd.node_ip }}:2379 \
--listen-client-urls http://{{ etcd.node_ip
}}:2379,http://127.0.0.1:2379 \
--initial-cluster-token etcd-cluster-1 \
--initial-cluster {{ etcd.initial_cluster_nodes }} \
--initial-cluster-state new
```

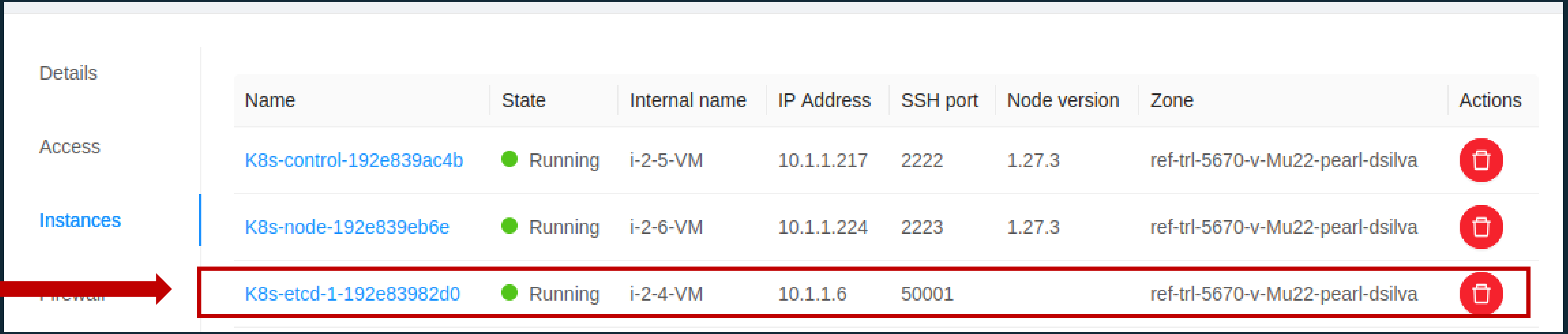
- SSH port forwarding rule is created on the cluster's public IP to the public ports starting on 'cloud.kubernetes.etcd.node.start.port', incrementing on each node.

- '*cloud.kubernetes.etcd.node.start.port*'- Indicates the start port for etcd nodes SSH access port forwarding rules on the cluster public IP address. Default value = 50000

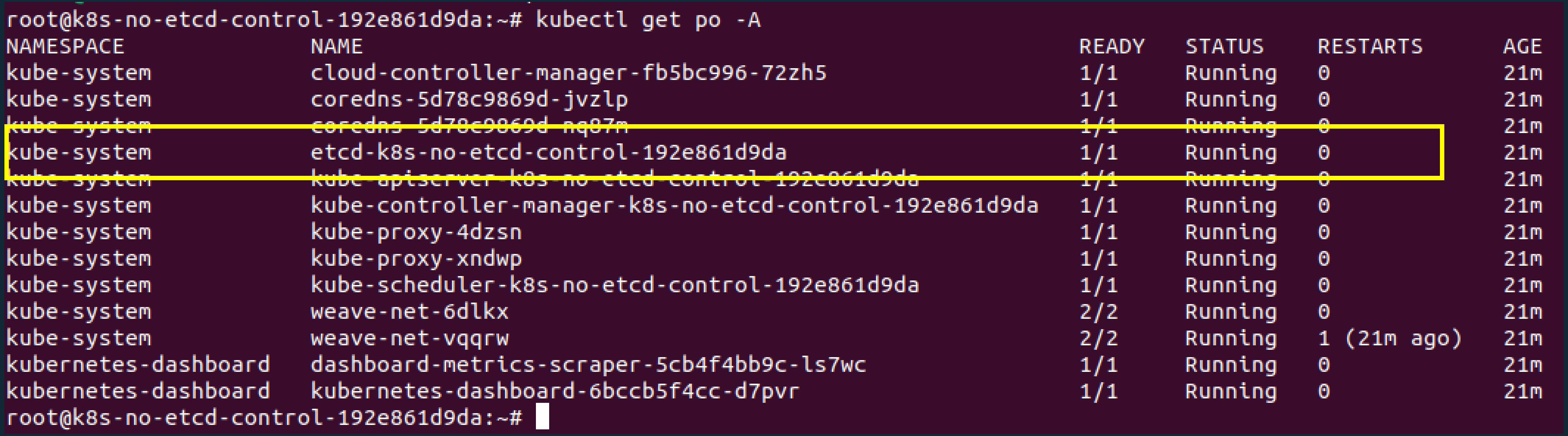


```
No resources found in default namespace.
root@K8s-control-192e839ac4b:~# kubectl get po -A
NAMESPACE              NAME                                          READY   STATUS    RESTARTS      AGE
kube-system            cloud-controller-manager-fb5bc996-wrlq9       1/1     Running   0             34m
kube-system            coredns-5d78c9869d-hsr2r                      1/1     Running   0             35m
kube-system            coredns-5d78c9869d-v6978                      1/1     Running   0             35m
kube-system            kube-apiserver-k8s-control-192e839ac4b        1/1     Running   0             35m
kube-system            kube-controller-manager-k8s-control-192e839ac4b   1/1     Running   0             35m
kube-system            kube-proxy-9m9dg                              1/1     Running   0             35m
kube-system            kube-proxy-qf2dt                              1/1     Running   0             35m
kube-system            kube-scheduler-k8s-control-192e839ac4b        1/1     Running   0             35m
kube-system            weave-net-kdz9d                               2/2     Running   2 (35m ago)   35m
kube-system            weave-net-m7tgq                               2/2     Running   2 (35m ago)   35m
kubernetes-dashboard   dashboard-metrics-scraper-5cb4f4bb9c-8wjks    1/1     Running   0             35m
kubernetes-dashboard   kubernetes-dashboard-6bccb5f4cc-n6nwj         1/1     Running   0             35m
root@K8s-control-192e839ac4b:~#
```

External Etcd node



Name	State	Internal name	IP Address	SSH port	Node version	Zone	Actions
K8s-control-192e839ac4b	● Running	i-2-5-VM	10.1.1.217	2222	1.27.3	ref-trl-5670-v-Mu22-pearl-dsilva	🗑
K8s-node-192e839eb6e	● Running	i-2-6-VM	10.1.1.224	2223	1.27.3	ref-trl-5670-v-Mu22-pearl-dsilva	🗑
K8s-etcd-1-192e83982d0	● Running	i-2-4-VM	10.1.1.6	50001		ref-trl-5670-v-Mu22-pearl-dsilva	🗑



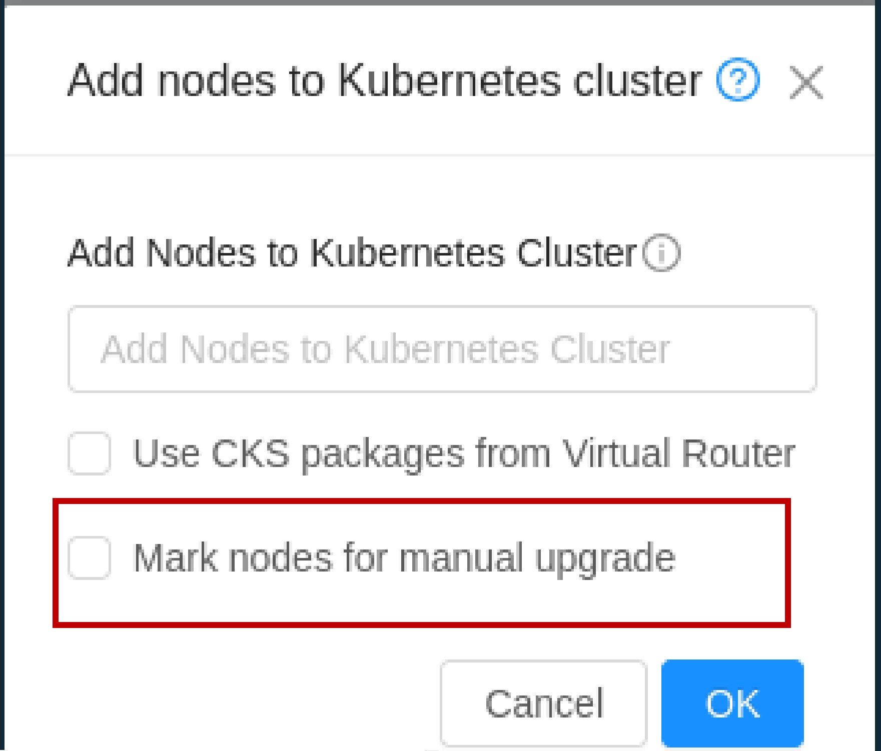
```
root@k8s-no-etcd-control-192e861d9da:~# kubectl get po -A
NAMESPACE              NAME                                                      READY   STATUS    RESTARTS      AGE
kube-system            cloud-controller-manager-fb5bc996-72zh5                   1/1     Running   0             21m
kube-system            coredns-5d78c9869d-jvzlp                                  1/1     Running   0             21m
kube-system            coredns-5d78c9869d-nq87m                                  1/1     Running   0             21m
kube-system            etcd-k8s-no-etcd-control-192e861d9da                      1/1     Running   0             21m
kube-system            kube-apiserver-k8s-no-etcd-control-192e861d9da            1/1     Running   0             21m
kube-system            kube-controller-manager-k8s-no-etcd-control-192e861d9da   1/1     Running   0             21m
kube-system            kube-proxy-4dzsn                                          1/1     Running   0             21m
kube-system            kube-proxy-xndwp                                          1/1     Running   0             21m
kube-system            kube-scheduler-k8s-no-etcd-control-192e861d9da            1/1     Running   0             21m
kube-system            weave-net-6dlkx                                           2/2     Running   0             21m
kube-system            weave-net-vqqrw                                           2/2     Running   1 (21m ago)   21m
kubernetes-dashboard   dashboard-metrics-scraper-5cb4f4bb9c-ls7wc                1/1     Running   0             21m
kubernetes-dashboard   kubernetes-dashboard-6bccb5f4cc-d7pvr                     1/1     Running   0             21m
root@k8s-no-etcd-control-192e861d9da:~#
```

Legacy CKS deployment

#CSCollab2024

Manual Upgrades of CKS nodes

- Designed for externally added nodes – designed specifically for baremetal nodes
- Marked for manual upgrade at the time of its addition to the cluster
- Kubernetes Cluster response now returns the version of Kubernetes on each node



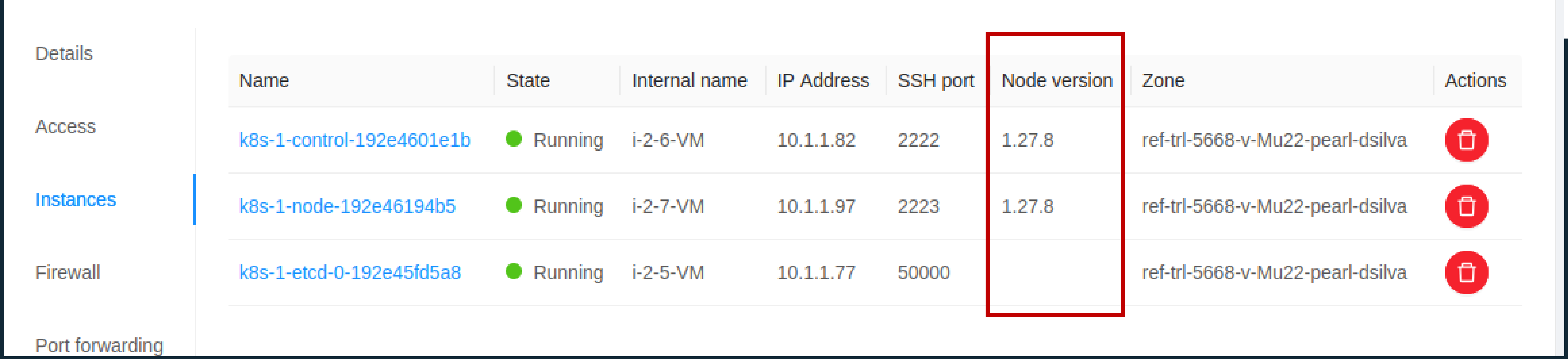
Add nodes to Kubernetes cluster

Add Nodes to Kubernetes Cluster

Add Nodes to Kubernetes Cluster

□ Use CKS packages from Virtual Router

□ Mark nodes for manual upgrade

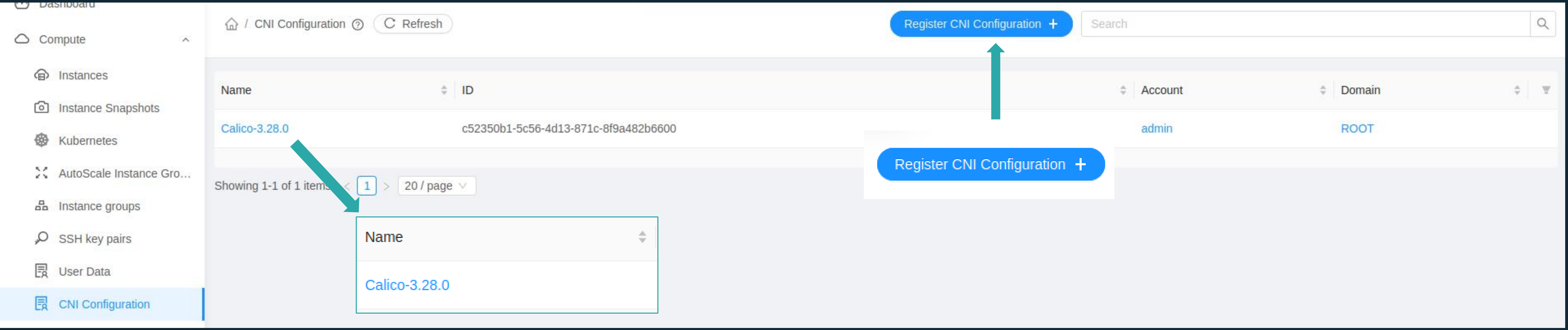


Name	State	Internal name	IP Address	SSH port	Node version	Zone	Actions
k8s-1-control-192e4601e1b	● Running	i-2-6-VM	10.1.1.82	2222	1.27.8	ref-trl-5668-v-Mu22-pearl-dsilva	
k8s-1-node-192e46194b5	● Running	i-2-7-VM	10.1.1.97	2223	1.27.8	ref-trl-5668-v-Mu22-pearl-dsilva	
k8s-1-etcd-0-192e45fd5a8	● Running	i-2-5-VM	10.1.1.77	50000		ref-trl-5668-v-Mu22-pearl-dsilva	

#CSCollab2024

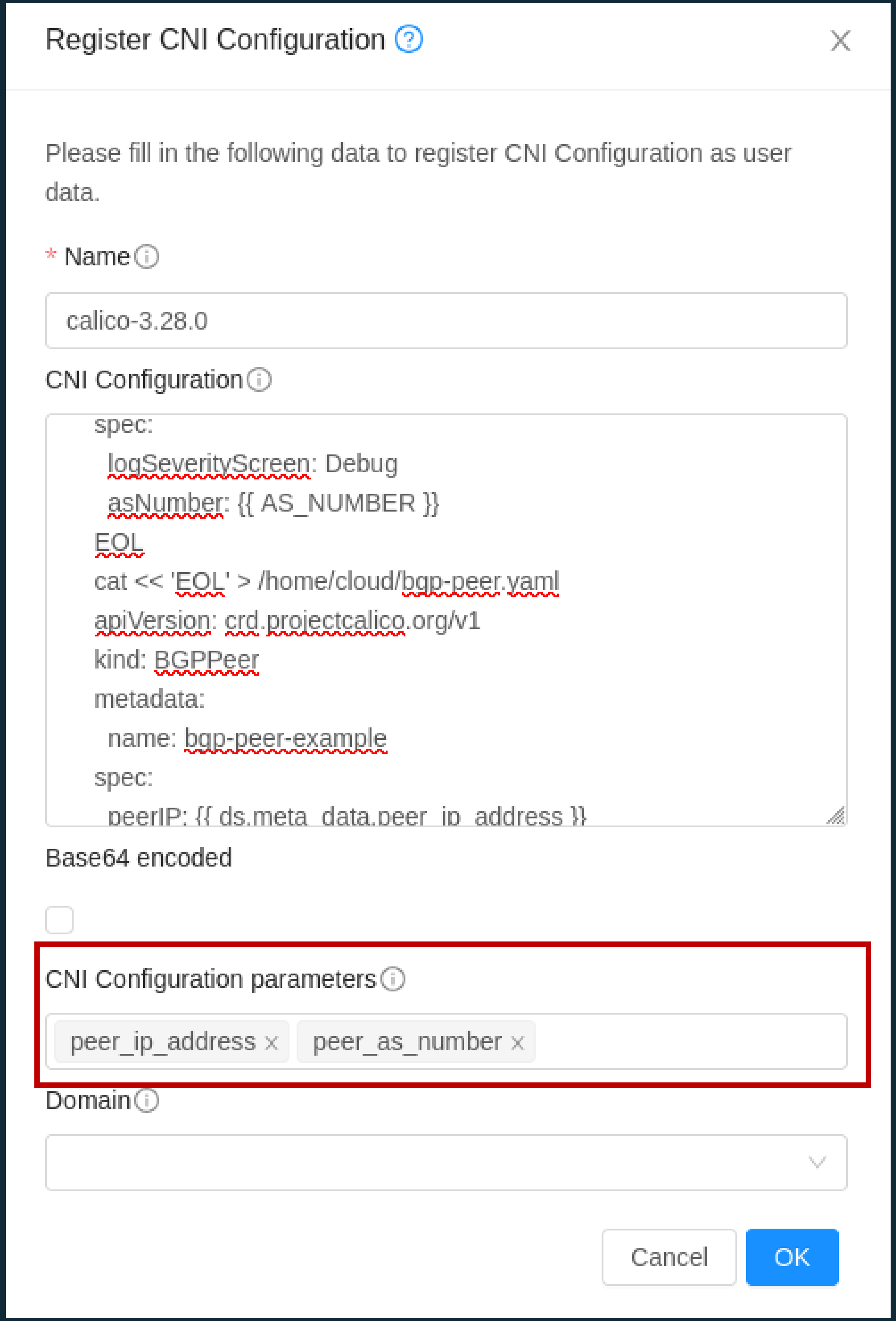
CNI as a first-class citizen

- Uses UserData to provide CNI configuration as userdata
- In ACS 4.20 - support has been added for Calico and Cilium – this would require building the CKS ISOs with the relevant images for the CNI of your choice
- This feature provides an alternative to the above approach



Name	ID	Account	Domain
Calico-3.28.0	c52350b1-5c56-4d13-871c-8f9a482b6600	admin	ROOT

Showing 1-1 of 1 items



Register CNI Configuration ?

Please fill in the following data to register CNI Configuration as user data.

* Name ⓘ

calico-3.28.0

CNI Configuration ⓘ

```
  spec:
    logSeverityScreen: Debug
    asNumber: {{ AS_NUMBER }}
  EOL
  cat << 'EOL' > /home/cloud/bgp-peer.yaml
  apiVersion: crd.projectcalico.org/v1
  kind: BGPPeer
  metadata:
    name: bgp-peer-example
  spec:
    peerIP: {{ ds.meta_data.peer_ip_address }}
```

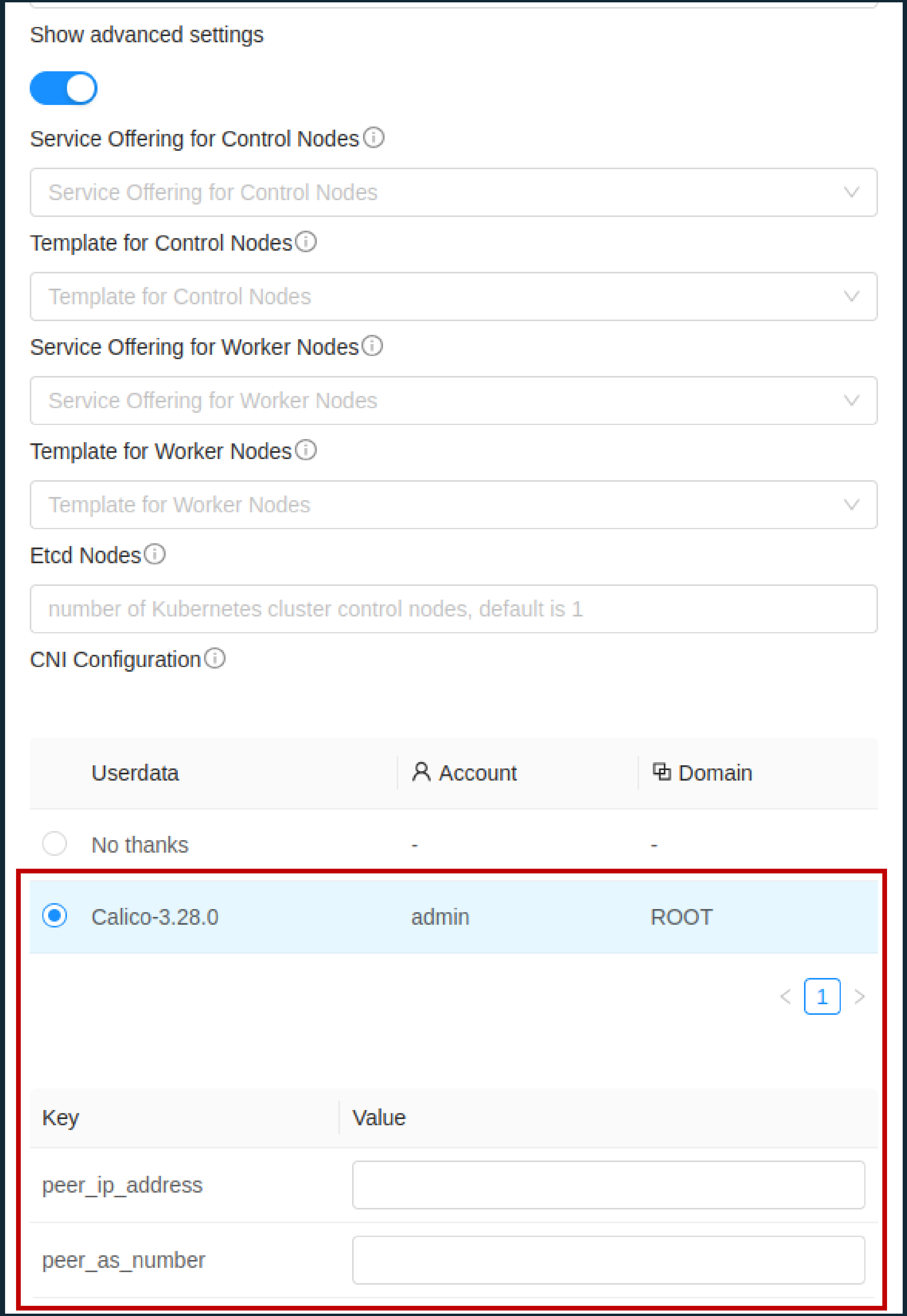
Base64 encoded

□

CNI Configuration parameters ⓘ

peer_ip_address × peer_as_number ×

Domain ⓘ



Show advanced settings

Service Offering for Control Nodes ⓘ

Service Offering for Control Nodes

Template for Control Nodes ⓘ

Template for Control Nodes

Service Offering for Worker Nodes ⓘ

Service Offering for Worker Nodes

Template for Worker Nodes ⓘ

Template for Worker Nodes

Etcd Nodes ⓘ

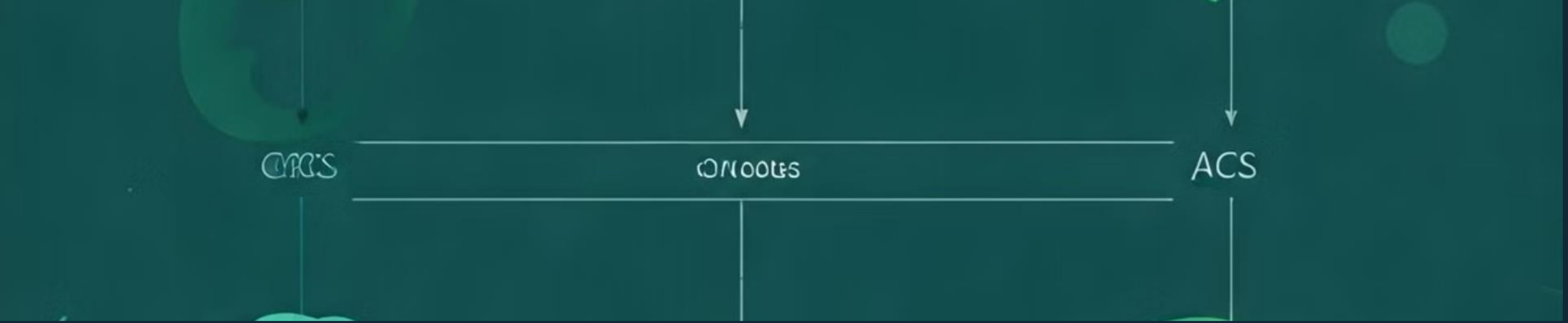
number of Kubernetes cluster control nodes, default is 1

CNI Configuration ⓘ

	Userdata	Account	Domain
○	No thanks	-	-
●	Calico-3.28.0	admin	ROOT

Key	Value
peer_ip_address	
peer_as_number	

#CSCollab2024

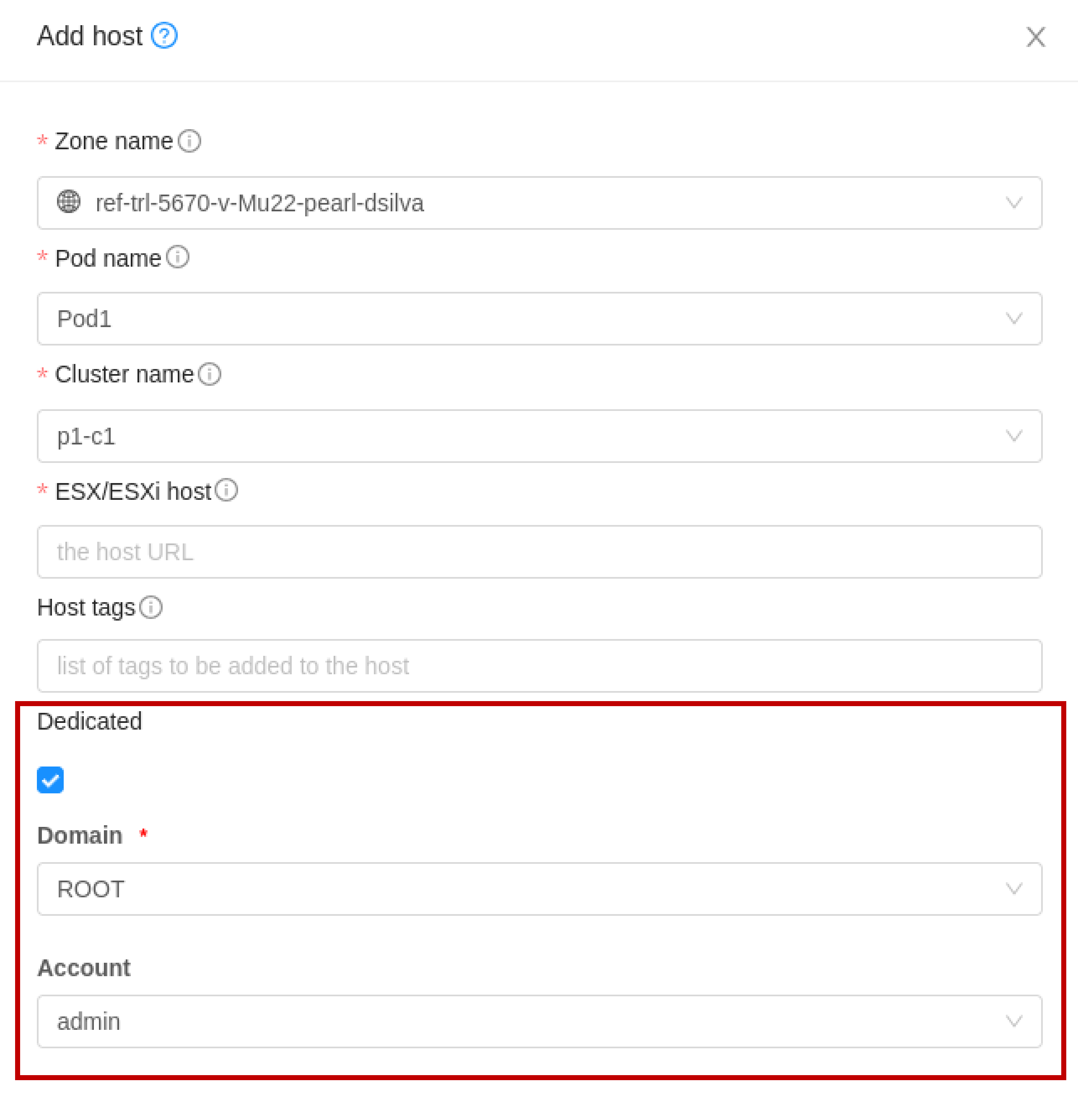


Dedicating CKS nodes to ACS hosts

Feature	Benefit
Resource Isolation	Improved performance and security
Predictable Performance	Consistent workload execution
Simplified Management	Easier tracking of resource allocation

Dedication of CKS nodes to ACS hosts

- Nodes will be deployed on any host when no dedicated hosts for a domain/account
- Nodes will be deployed on the dedicated hosts when present for a domain/account



Add host

* Zone name
ref-trl-5670-v-Mu22-pearl-dsilva

* Pod name
Pod1

* Cluster name
p1-c1

* ESX/ESXi host
the host URL

Host tags
list of tags to be added to the host

Dedicated
☑

Domain *
ROOT

Account
admin

CloudStack CSI Driver

CloudStack CSI Driver is a powerful tool that allows Kubernetes to interact with CloudStack's storage capabilities. It's a key component for utilizing CloudStack's infrastructure in a containerized environment.

Need for CSI Driver in a CloudProvider

Improved Interoperability

Ensures seamless integration with Kubernetes. It bridges the gap between CloudStack and the container orchestration platform.

Simplified Management

Streamlines the process of managing and orchestrating storage volumes within the Kubernetes ecosystem. It centralizes storage management within Kubernetes.

Enhanced Flexibility

Allows Kubernetes to leverage different storage solutions offered by CloudStack, such as block storage, object storage, and file storage.

Volumes on Kubernetes

1 File Sharing

CSI drivers enable file sharing between containers in a Pod, solving inter-container communication issues.

2 Data Persistence

They allow stateful applications by providing persistent storage, preventing data loss during container crashes.

3 Storage Accessibility

Volumes in Kubernetes are directories accessible to all containers in a pod, with configurable lifetimes.

Types of Volume Plugins

Persistent Volumes

These outlive the pod lifecycle and are independent. They provide durable storage for long-term data retention.

Ephemeral Volumes

Tied to pod lifecycle, including EmptyDir, Secret, ConfigMap, and HostPath. Suitable for temporary storage needs.

Provisioning Volumes on k8s

Static Provisioning

- A user has to provision a volume before an app is deployed to a node
- Not scalable

Dynamic Provisioning

During application deployment the amount of disk space is specified, and the necessary volume is then automatically deployed

Persistent Volume Claims – resource k8s provides to provision volumes on SPs for its workloads

Direct Volume Reference Example

1

Volume Definition

Specify the volume type and details in the pod specification.

2

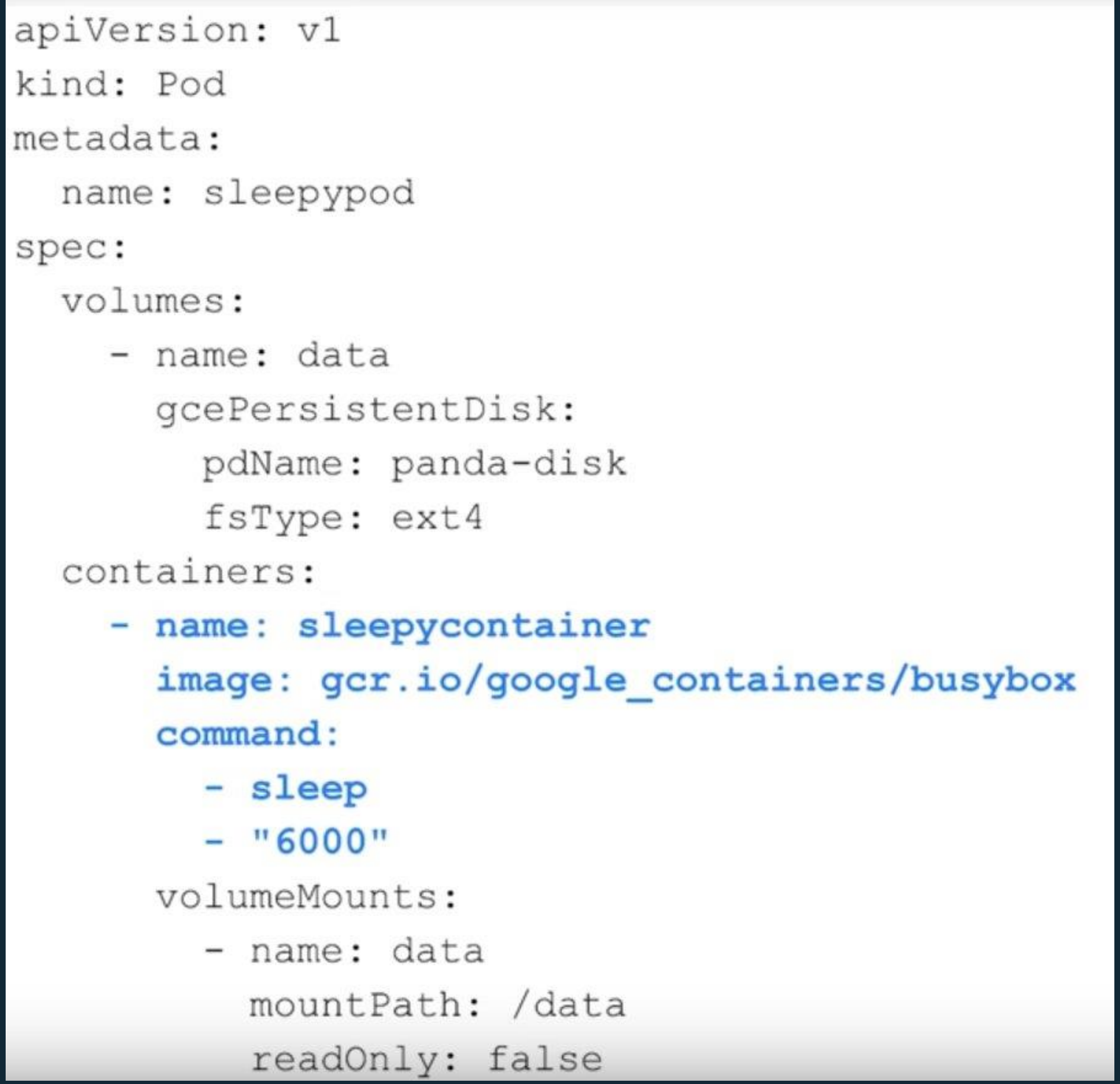
Volume Mount

Define the mount path for the volume within the container.

3

Container Usage

The container can now access the mounted volume at the specified path.



```
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
```

#CSCollab2024

Persistent Volumes and Claims

1 **PersistentVolume Creation**

Administrator creates a PersistentVolume, representing available storage in the cluster.

2 **PersistentVolumeClaim Definition**

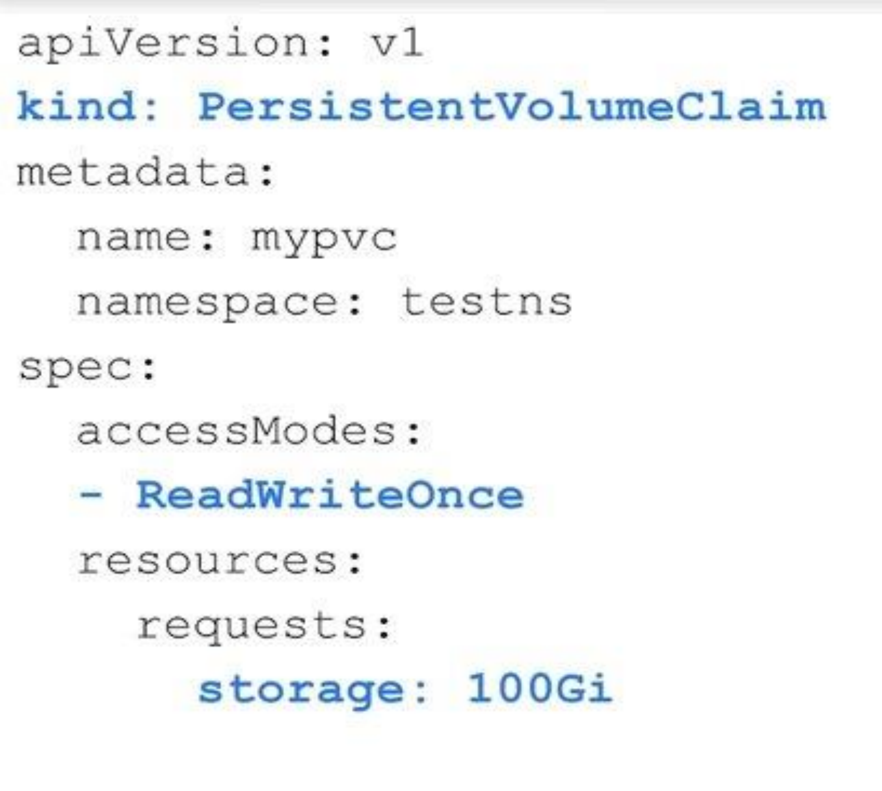
User creates a PersistentVolumeClaim, specifying storage requirements and access mode.

3 **Claim Binding**

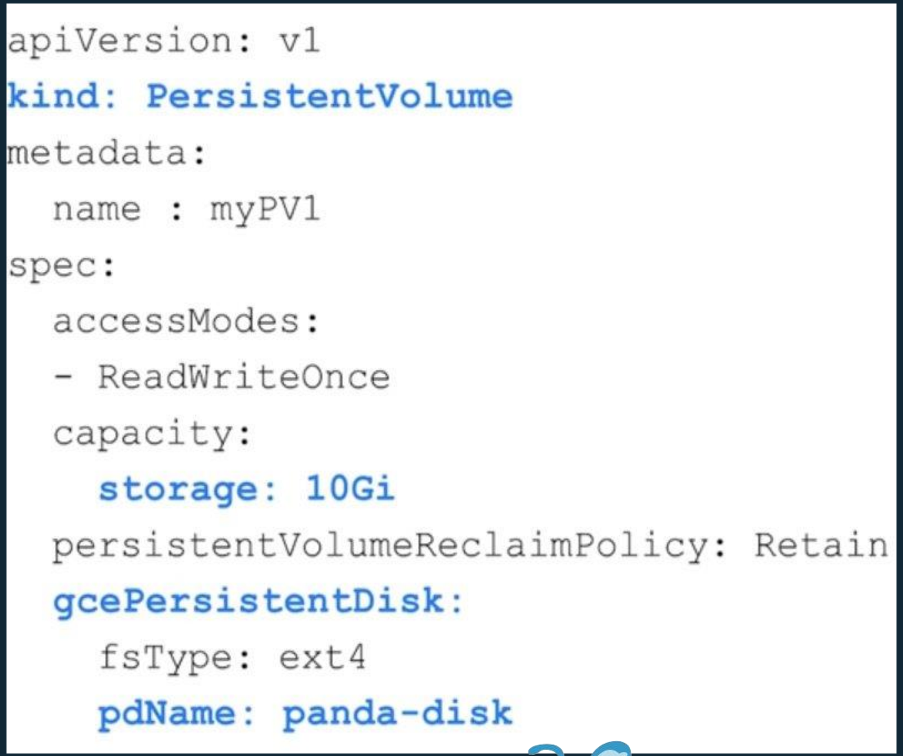
Kubernetes binds the PVC to an appropriate PV based on the claim's requirements.

4 **Pod Usage**

Pod references the PVC, gaining access to the underlying storage.



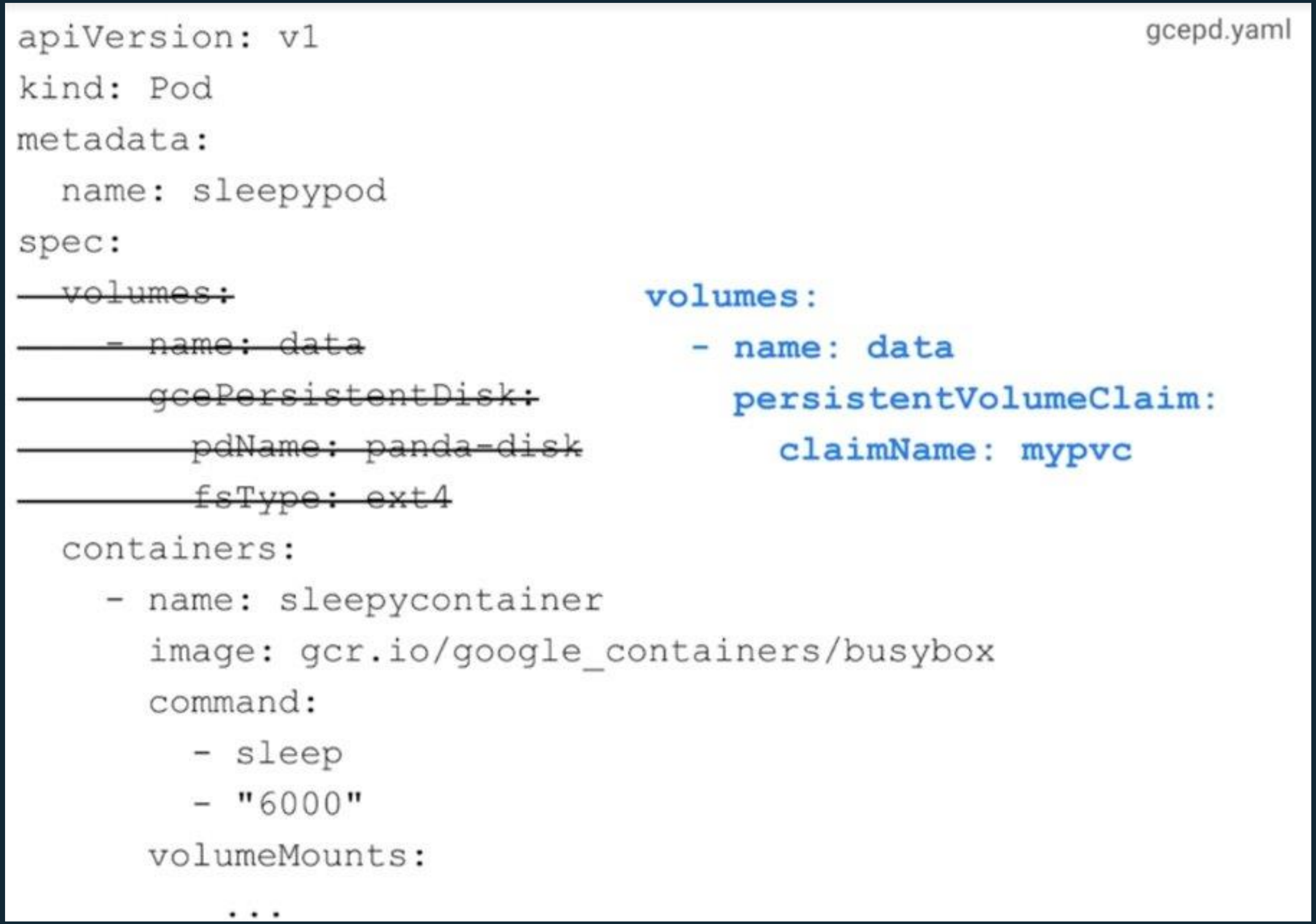
```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc
  namespace: testns
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
```



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

#CSCollab2024

PVC-Referenced Volume Example



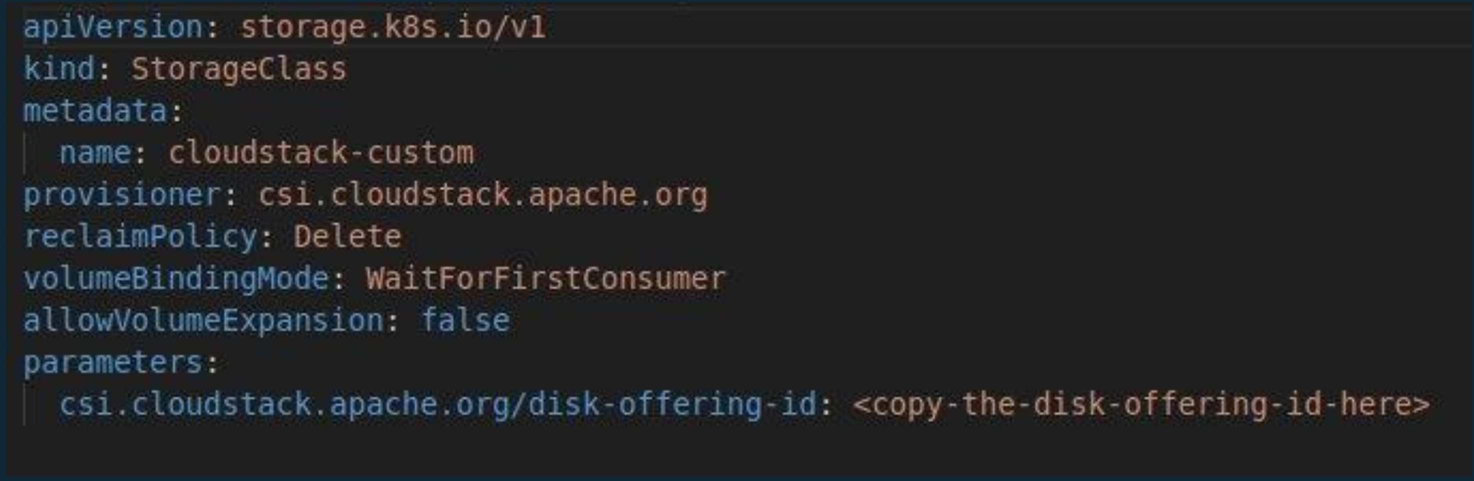
gcepd.yaml

```
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:
          ...
```

Dynamic Provisioning



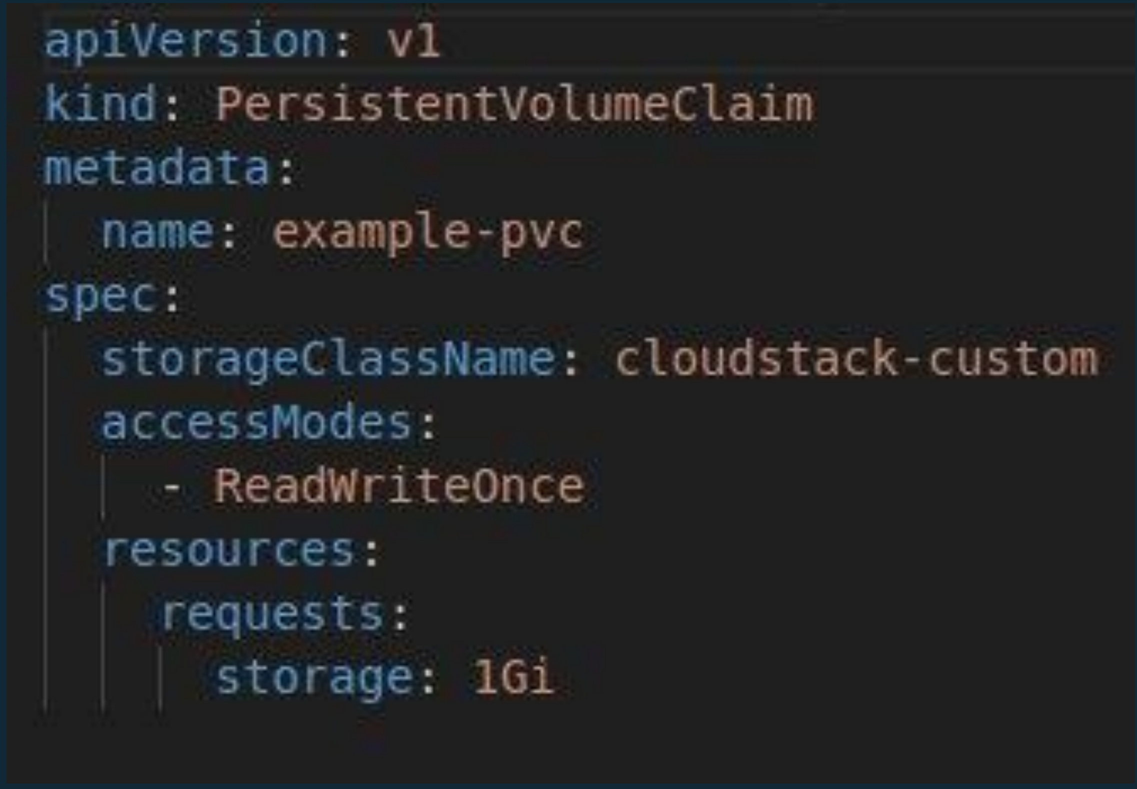
```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cloudstack-custom
provisioner: csi.cloudstack.apache.org
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: false
parameters:
  csi.cloudstack.apache.org/disk-offering-id: <copy-the-disk-offering-id-here>
```

StorageClass

Defines the volume plugin and parameters for dynamic provisioning. Administrators create these classes.

PVC Creation

Users specify a StorageClass in their PVC, triggering automatic PV creation.



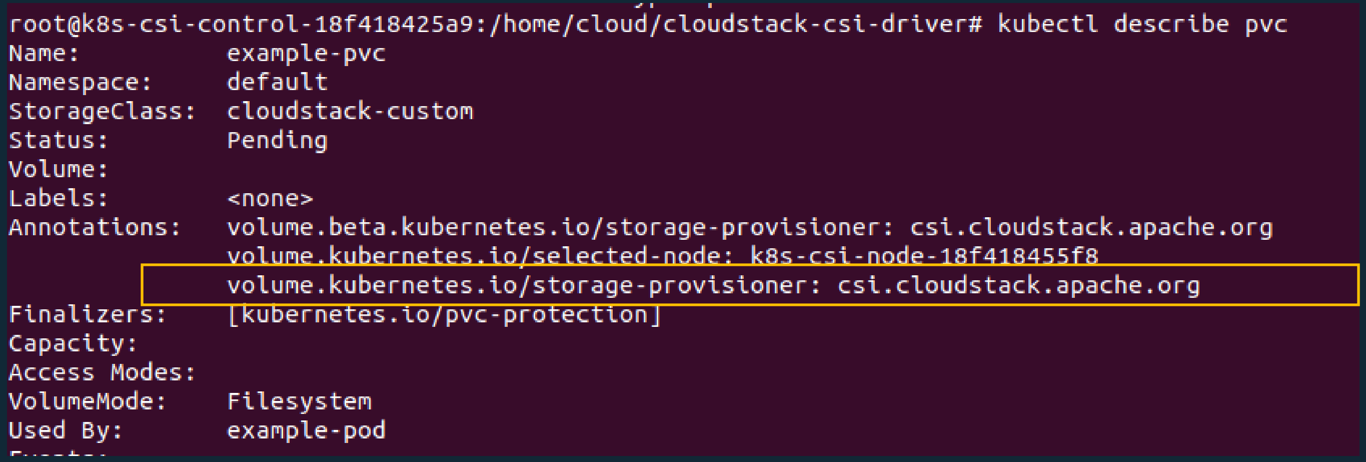
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: example-pvc
spec:
  storageClassName: cloudstack-custom
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Provisioner

The CSI driver acts as a provisioner, creating volumes based on StorageClass parameters.

Automation

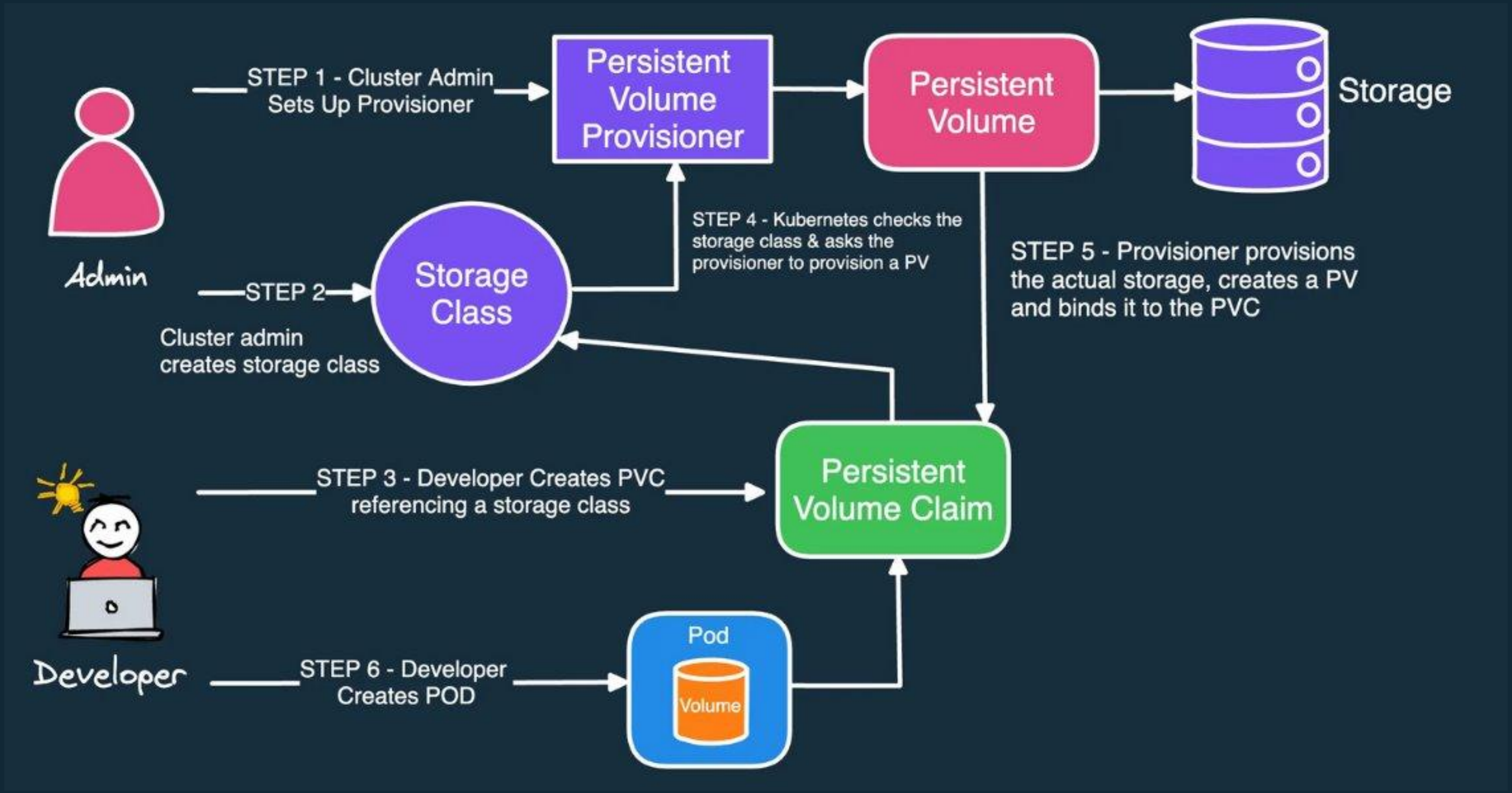
Reduces manual intervention, allowing on-demand storage allocation for applications.



```
root@k8s-csi-control-18f418425a9:/home/cloud/cloudstack-csi-driver# kubectl describe pvc
Name:          example-pvc
Namespace:     default
StorageClass:  cloudstack-custom
Status:        Pending
Volume:
Labels:        <none>
Annotations:   volume.beta.kubernetes.io/storage-provisioner: csi.cloudstack.apache.org
               volume.kubernetes.io/selected-node: k8s-csi-node-18f418455f8
               volume.kubernetes.io/storage-provisioner: csi.cloudstack.apache.org
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Used By:       example-pod
Events:
```

#CSCollab2024

Working of StorageClass in K8S


Admin
STEP 1 - Cluster Admin Sets Up Provisioner
Persistent Volume Provisioner
Persistent Volume
Storage
STEP 2
Cluster admin creates storage class
Storage Class
STEP 4 - Kubernetes checks the storage class & asks the provisioner to provision a PV
STEP 5 - Provisioner provisions the actual storage, creates a PV and binds it to the PVC
Developer
STEP 3 - Developer Creates PVC referencing a storage class
Persistent Volume Claim
STEP 6 - Developer Creates POD
Pod
Volume

#CSCollab2024

CSI Specifications

Identity Service

Provides plugin identification and capability information to Kubernetes.

Needs to be implemented by both Node and Controller Plugins

Controller Service

Manages volume lifecycle operations like creation and deletion on the storage provider

Deployed as a Kubernetes Deployment

Controller Plugin needs to Implement the RPCs defined in the Controller Service

ControllerGetCapabilities

Node Service

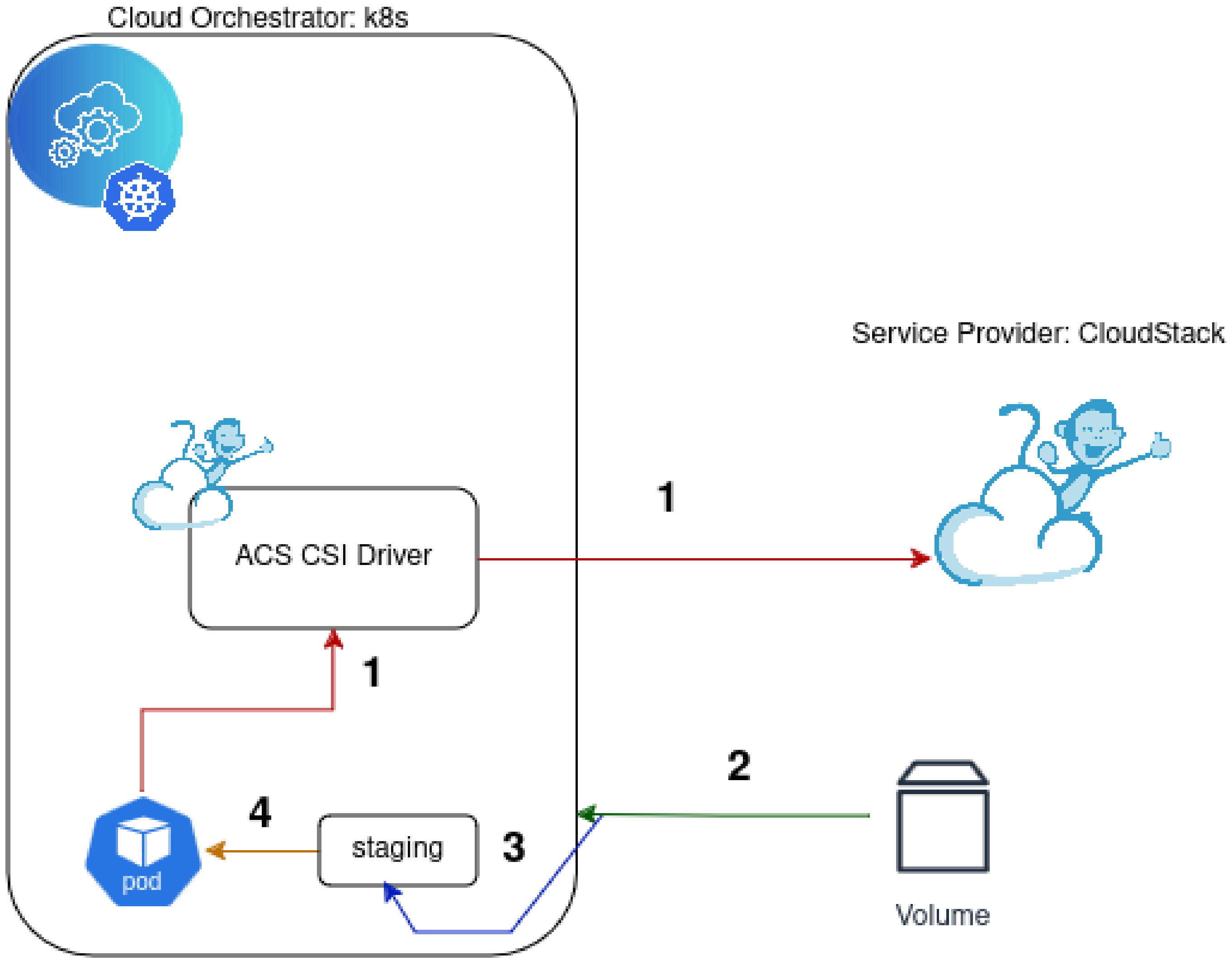
Handles volume operations on the node, such as mounting and unmounting

Node Plugin needs to implement the RPCs defined in the Node Service

Generally deployed as DaemonSet

NodeGetCapabilities

Example workflow of a CSI Driver for Creating a Volume



1. CreateVolume – creates volume in SP
2. ControllerPublishVolume Attaches volume to the node
3. NodeStageVolume makes volume available in a staging path
4. NodePublishVolume mounts volume in the workload i.e., pod

Demo

#CSCollab24
@CloudStack

CloudStack CSI Driver - Demo

Microsoft Teams

Meeting with Pearl d'Silva

2024-11-13 04:27 UTC

Recorded by
Pearl d'Silva

Organized by
Pearl d'Silva

#CSCollab2024

CloudStack Enhancements - Demo

Microsoft Teams

Meeting with Pearl d'Silva

2024-11-14 19:56 UTC

Recorded by
Pearl d'Silva

Organized by
Pearl d'Silva

#CSCollab2024

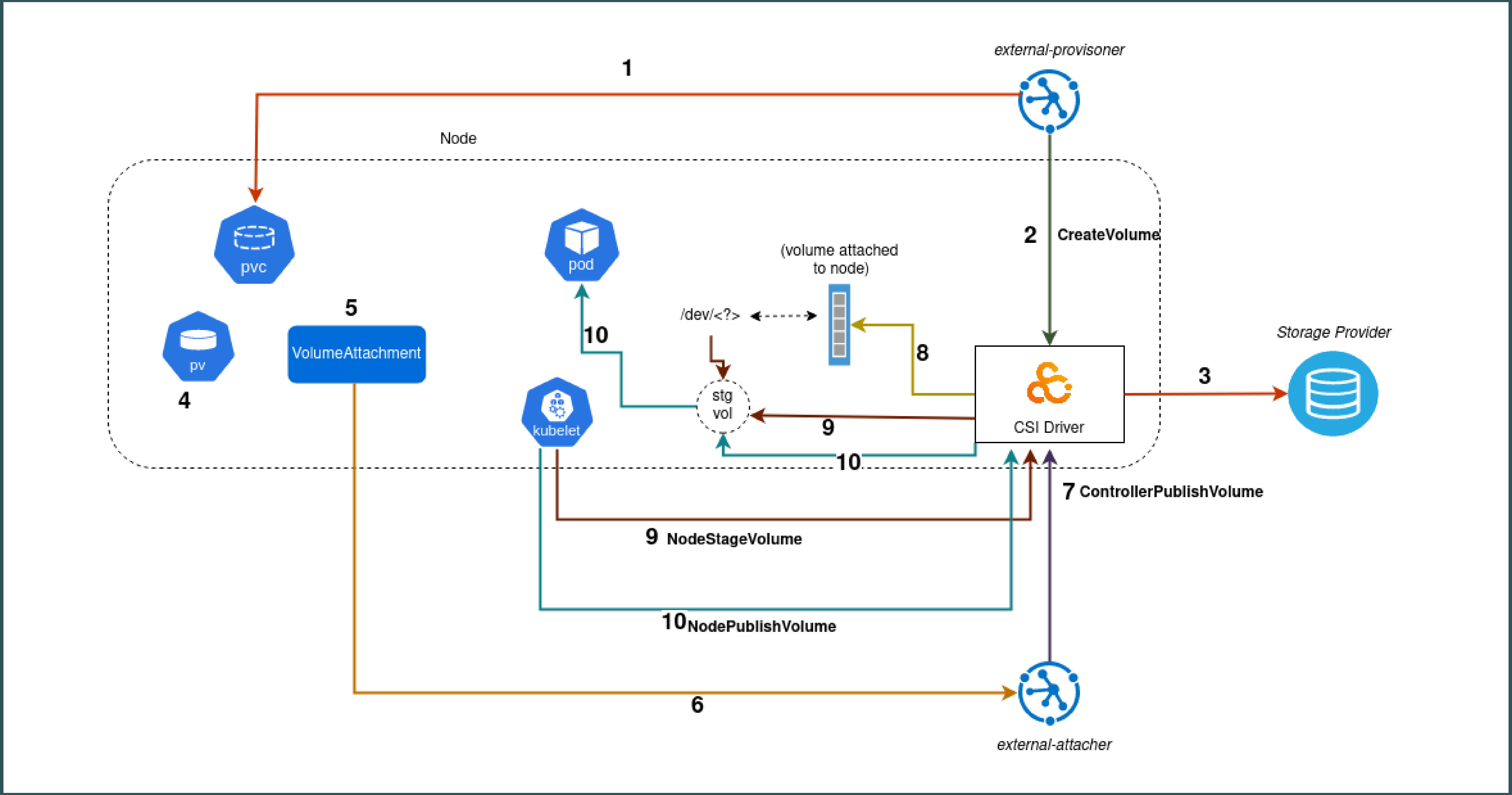
Questions

#CSCollab24
@CloudStack

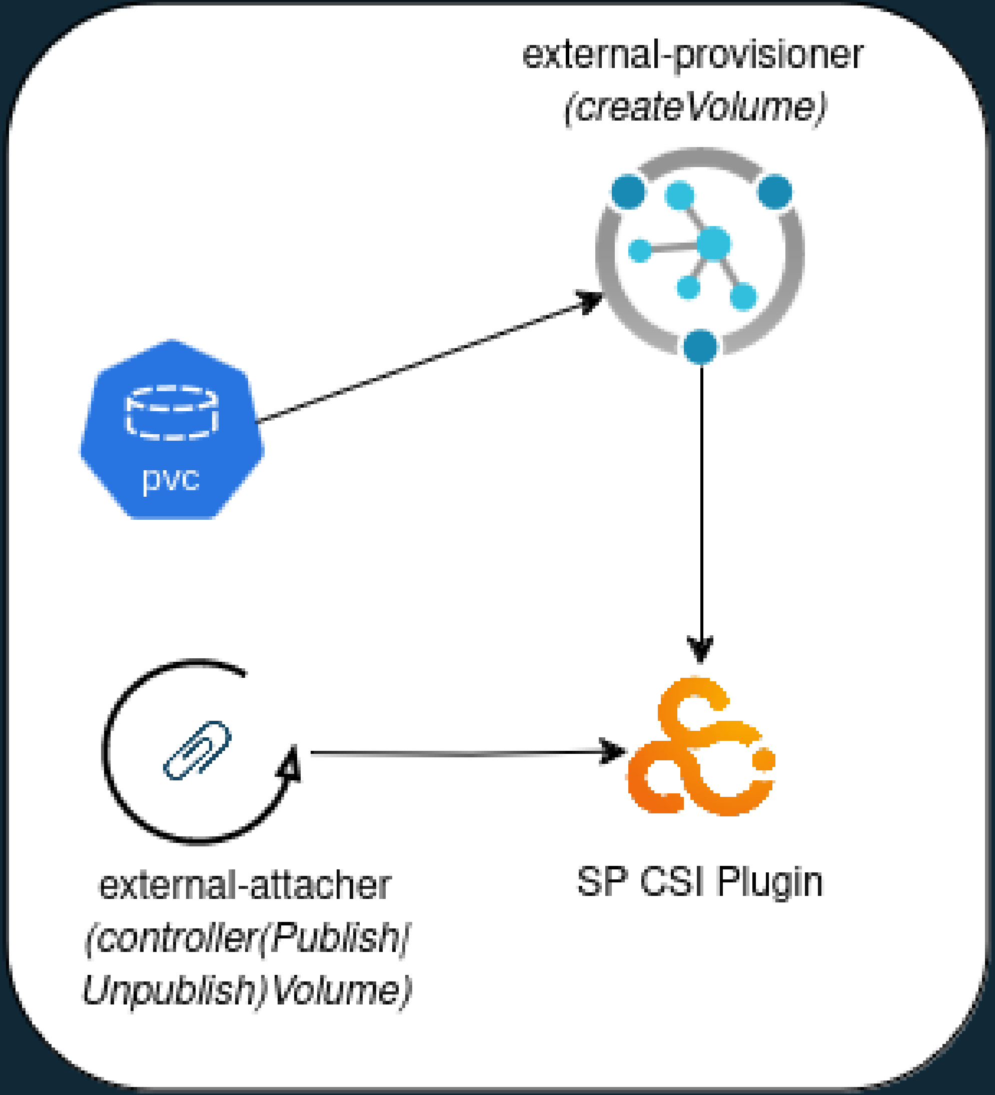
Thank you!

#CSCollab24
@CloudStack

E2E workflow to provision volume



Communication between k8s & CSI Plugin



- Interaction b/w k8s & CSI Plugins is managed by several sidecar containers

- Watch k8s API server requests and translates them to CSI RPC requests and update the Kubernetes API

- The use of these containers are optional

Examples:

- To make a volume attached to a node available to a pod, kubelet talks to the CSI driver – ***node-driver-registrar*** helps it know where the CSI driver is running

Communication between k8s & CSI Plugin

external-attacher:

Watches: VolumeAttachment

Triggers: Controller(Publish|Unpublish)Volume

Required by/ Sidecar to : Controller Service / Plugin Pod

external-provisioner:

Watches: PersistentVolumeClaim

Triggers: Volume(Create|Delete) of controller service

Required by/ Sidecar to : Controller Service / Plugin

external-resizer:

Watches: PesistenVolumeClaim edits

Triggers: ControllerExpandVolume

external-snapshotter:

Watches: VolumeSnapshotContent

Triggers: (Create|Delete|List)Snapshots

livenessprobe: monitors health of CSI driver

node-driver-registrar: informs kubelet where the CSI driver is deployed to interact with it to make available the volume attached to the node to a workload (Pod). Required by/ Sidecar to: Node Plugin Pod

#CSCollab2024