STORAGE DEVELOPER CONFERENCE
SDC EMEA 21
BY Developers FOR Developers
Virtual Conference
June 8, 2021

# Object Storage for Cloud Native Applications

## Beyond CSI

Nicolas Trangez - Principle Architect, Scality
Vianney Rancurel - Head of Research, Scality

# Object Storage

## A Primer

# Object Storage Paradigm

- Unstructured, non-hierarchical, self-contained, uniquely-identified blobs with metadata
  - Has its place next to ‘traditional’ file and block storage
- Various consistency models
- Objects collected in buckets, flat namespace
- Access at object level, often through HTTP/RESTful APIs
- Fine-grained authn/authz, policy driven
- Originally CRUD, nowadays versioning, CRR, metadata search, server-side payload queries,...
- Historically ‘slow’ (high throughput, high latency), nowadays ‘fast’ (high throughput, low and predictable latency)
  - Allows for new types of workloads

# Cloud Native Applications

## Architectural Principles

# Cloud Native?

"Cloud native is structuring teams, culture, and technology to utilize automation and architectures to manage complexity and unlock velocity."

Joe Beda - Co-Founder, Kubernetes and Principal Engineer, VMware

# Cloud Native Architecture

- Loosely coupled, service/contract-oriented
- Pervasive automation
- Principles help to design scalable software
  - In “performance” and business growth
  - In engineering teams
  - In feature development agility/velocity
- Clear distinction between stateful and stateless services
  - Stateless contributes to (“performance”) scalability
- Consume “as a service” technologies
  - Database, storage, BigData & ML pipelines,...
  - All API/automation driven
- No (or not many) ‘pets’
- Portability (e.g., using containers)

# Kubernetes

## Running Containerized Cloud Native Applications at Scale

# Kubernetes from 30,000 Feet

- A consistent highly-available database (*etcd*, based on RAFT)
- A data model for objects stored in the database
- A RESTful API to CRUD and watch objects
- Many controllers to watch for creations/updates/deletes, and CRUD other objects & update watched object status
  - Often refining high-level objects into lower-level ones
  - Sometimes calling into external systems (e.g., call into a CSI driver to create a volume in an external storage system)
- Agents running on cluster nodes to realize node-local *'things'* in the real world
  - *kubelet* to set up container networking (through CNI, e.g., Calico or Cilium), set up storage (through CSI) and start/stop/monitor containers (through CRI, e.g., *containerd* or *cri-o*)
  - *kube-proxy* to configure node-local *iptables*/*IPVS* for in-cluster virtual IPs (and more)
  - CNI daemon to set up network routes etc., if applicable

# Building a Storage Service on Kubernetes

## Embracing Cloud Native Technologies in a Storage Product

# Cloud Native Principles for Storage Systems

- Scale
- Loose coupling
- Portability

# A Storage System Running on Kubernetes

- Built-in cluster management
- Deployment automation: *operators*
- Whole stack, unified operational tooling
- Minimal performance impact (disk access, networking,...)
- Limited support for “non-cloud” environments

# Scality's Journey to Kubernetes

SCALITY

- Zenko open-source multi-cloud data controller (see *"Data Management in a Cloud-Agnostic World"*, SNIA SDC 2019)

zenko.io

- MetalK8s open-source K8s distribution for on-premises deployment (see *"MetalK8s, an opinionated Kubernetes distribution optimized for data management"*, SNIA SDC 2019)

METALK8S

- ARTESCA object storage for the new cloud-native era

ARTESCA

STORAGE DEVELOPER CONFERENCE SDC 21 EMEA

# Consuming Storage in Kubernetes

## Patterns for Stateful Cloud Native Applications

# Blocks & Files

- API-driven provisioning: *StorageClass*, *PersistentVolumeClaim*
- Results in calls into storage-system-specific CSI driver (GRPC server)
- Once Pod scheduled:
  - CSI controller service called to expose/attach storage to a node, if applicable,
  - Node-local CSI driver called to *mount* storage on a node, then exposed in Pod containers by *kubelet*
- Set up access to block and file storage requires OS/kernel-level operations
  - Requires access to privileged system calls
  - Not allowed from inside containers, so CSI driver *required* to consume storage from K8s workloads

# Object Storage

- 100% userspace access (HTTPS)
  - No need for OS/kernel-level privileged operations
  - Hence, container orchestrator support not strictly required
- Requires provisioning of buckets and access credentials
  - Without automation:
    - Create bucket out-of-band
    - Mint credentials out-of-band
    - Provide credentials, endpoint,... to containerized application
  - With automation: new Kubernetes API, COSI
    - Include object storage provisioning with workload provisioning
    - Fits with cloud-native principles: CI/CD automation, *infrastructure-as-code*,...
    - Control-plane (provisioning) only

# COSI

## The Container Object Storage Interface

# Background

- Cross-vendor API specification effort in Kubernetes SIG-Storage
- Cross-vendor GRPC API specification for vendor-specific integration
- Vendor-neutral user-facing provisioning API as K8s API extension
  - Ensures portability of workloads
  - User-facing objects to request buckets and credentials to be created
  - Application-facing API to expose credentials, bucket name, storage system address,... to Pod/application
- Vendor-neutral controller and sidecars to bridge vendor-specific integration
  - Similar model to CSI
- Not a data-plane API. Storage access through S3, Azure Blob,... whatever the storage system supports

# Use-Cases

- Greenfield
  - Deploy application (YAML manifests, CICD,...)
  - Include *BucketRequest* and *BucketAccessRequest* in set of manifests
  - Consume *BAR* from a Pod
- Brownfield
  - Administrator creates *Bucket* object, referencing an existing bucket, in K8s cluster
  - Application deployments include *BAR*s to request access to bucket
  - Consume BAR from a Pod

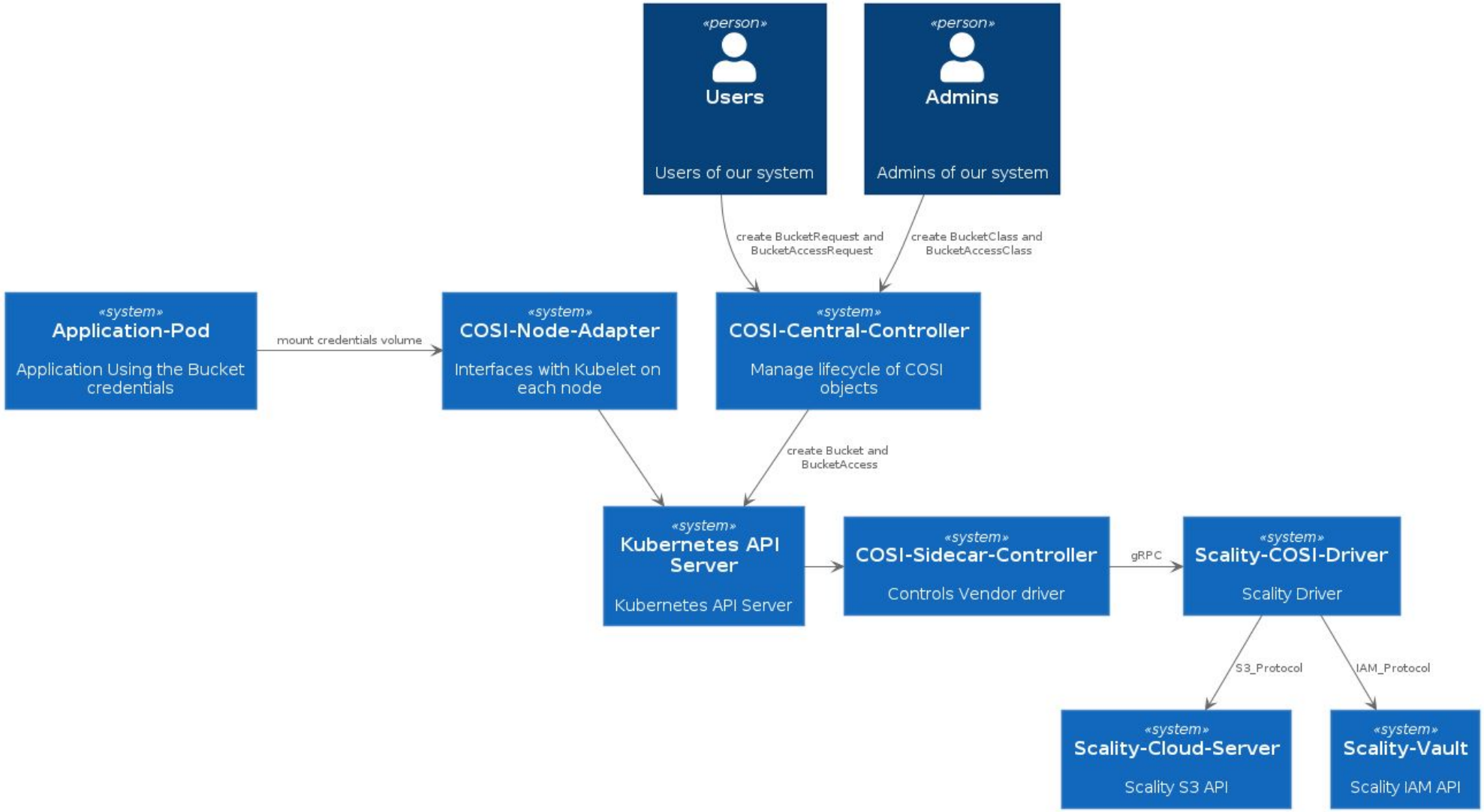
 

# Architecture

- Architecture Overview
  - Components
  - Custom Resources Definitions (CRDs)
- Scenarios
  - Bucket Creation Overview
  - Pod Creation Overview
  - Bucket Access Overview
- An Example Implementation
  - Driver Methods
  - Driver Create/Delete Bucket Sequences
  - Driver Grant/Revoke Access Sequences

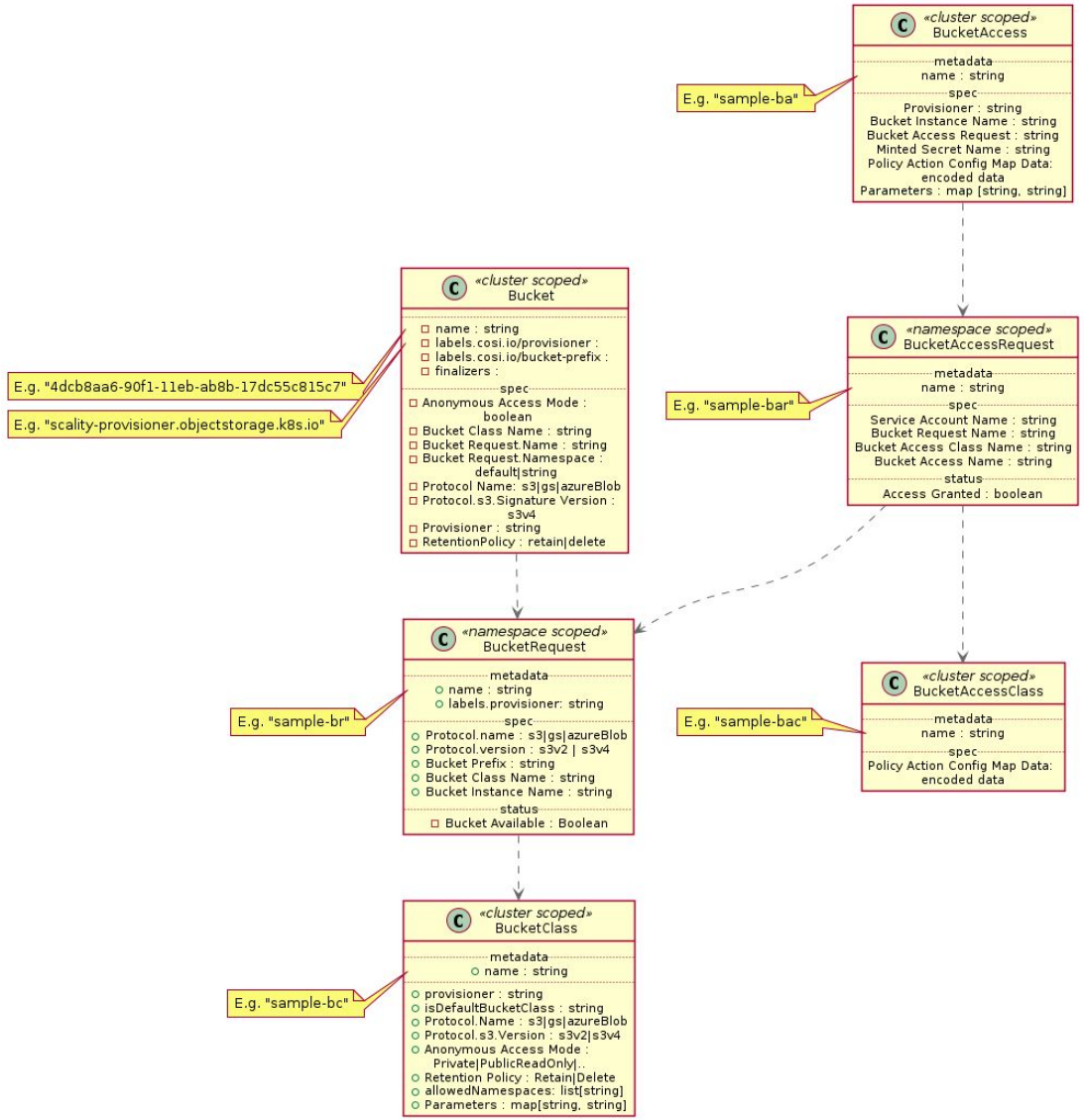
 

# Architecture - Overview



# Architecture - Custom Resource Definitions (CRDs)



# BucketClass, BucketRequest and Bucket

- ***BucketClass***: admin-managed class of buckets, managed by a COSI driver, including parameters for the driver to use when provisioning a bucket of the class (~ *StorageClass* for block/file volume provisioning)

- ***BucketRequest***: user-managed object, requesting a bucket to be provisioned (~ *PersistentVolumeClaim* for dynamic volume provisioning)

- ***Bucket***: object representing a provisioned (or brownfield pre-provisioned) bucket available for access (~ *PersistentVolume*)

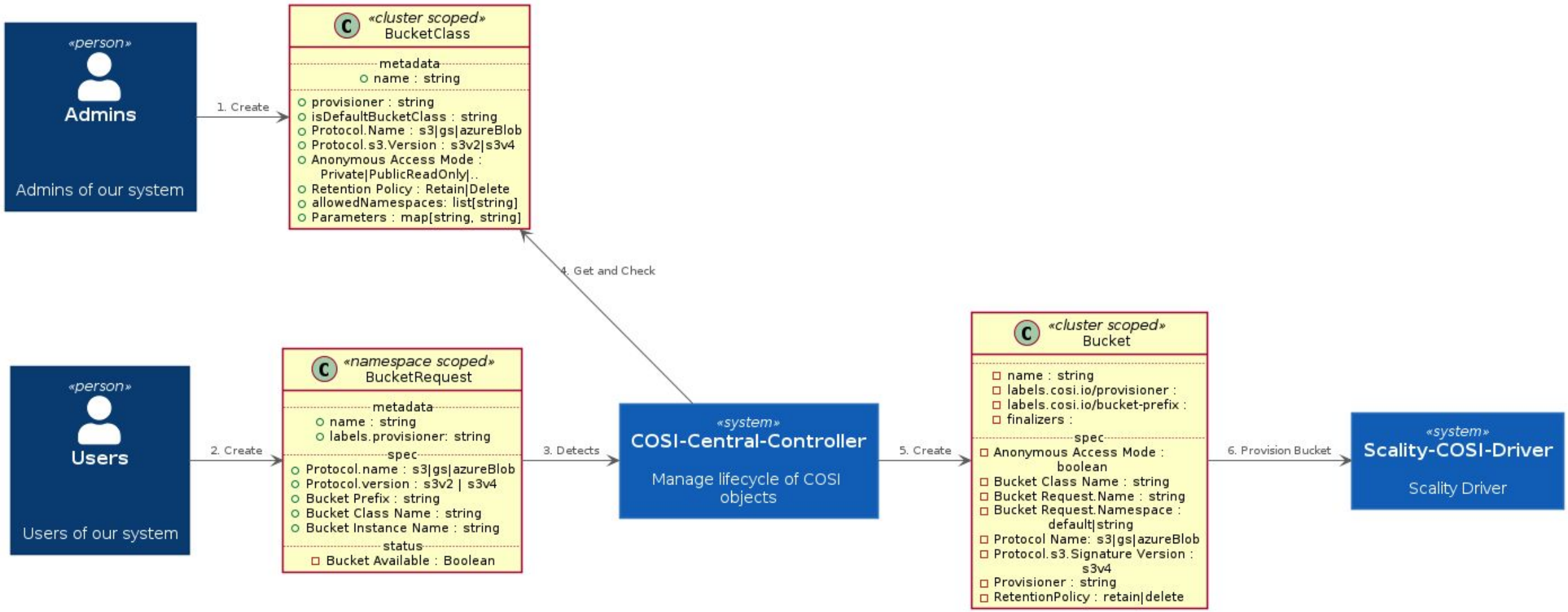
 

# BucketAccessClass, BucketAccessRequest and BucketAccess

- ***BucketAccessClass***: admin-managed class of access that can be requested
- ***BucketAccessRequest***: user-managed object requesting access to a bucket, according to a *BAC* (~ *PersistentVolumeClaim* but not for provisioning)
- ***BucketAccess***: object consumed by a Pod to access (~ get information on how to access, authenticate against,...) a bucket (~ a bound *PersistentVolumeClaim*, can consume bound storage)

# Architecture - Bucket Creation Overview



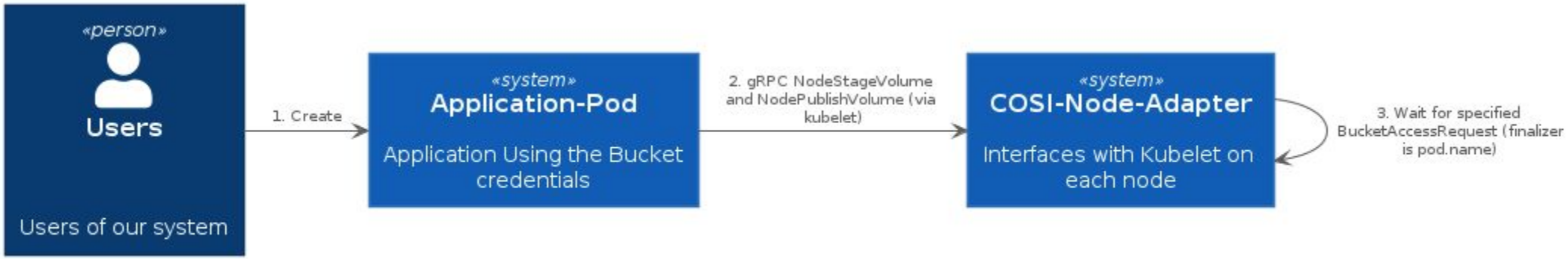
# Example BucketClass & BucketRequest CRs

```
kind: BucketClass
apiVersion: objectstorage.k8s.io/v1alpha1
metadata:
  name: sample-bc
protocol:
  name: s3
  s3:
    signatureVersion: s3v4
provisioner: sample-provisioner.objectstorage.k8s.io
```

```
kind: BucketRequest
apiVersion: objectstorage.k8s.io/v1alpha1
metadata:
  name: sample-br
spec:
  bucketClassName: sample-bc
```

# Architecture - Pod Creation Overview



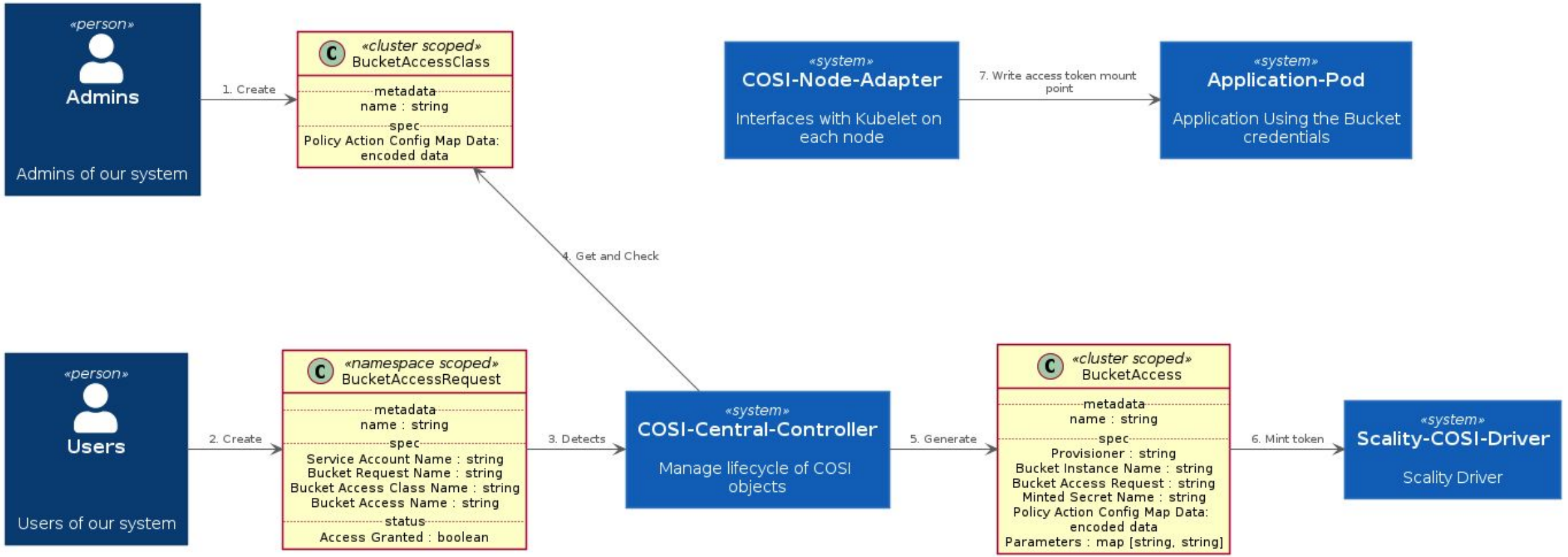

 

# Example Application Pod CR

```
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
spec:
  containers:
  - name: app
    image: app
...
    volumeMounts:
    - name: cosi-secrets
      mountPath: /data/cosi
    env:
      - name: MOUNT_PATH
        value: "/data/cosi"
  volumes:
  - name: cosi-secrets
    csi:
      driver: objectstorage.k8s.io
      volumeAttributes:
        bar-name: sample-bar
        bar-namespace: default
```

# Architecture - Bucket Access Overview



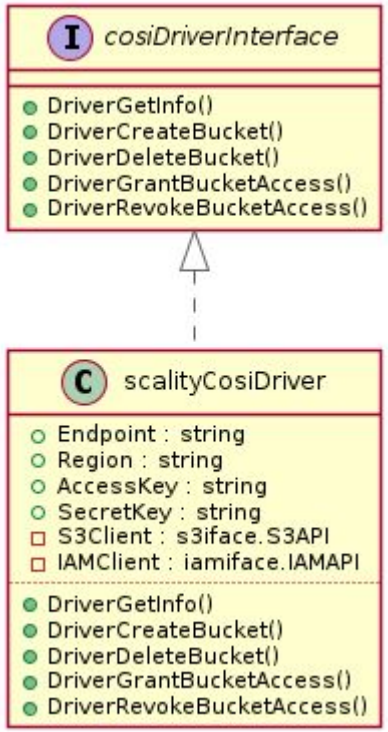
# Example BucketAccessClass, BucketAccessRequest CRs

```
kind: BucketAccessClass
apiVersion: objectstorage.k8s.io/v1alpha1
metadata:
    name: sample-bac
```

```
kind: BucketAccessRequest
apiVersion: objectstorage.k8s.io/v1alpha1
metadata:
  name: sample-bar
spec:
  bucketAccessClassName: sample-bac
  bucketAccessName: sample-ba
  bucketRequestName: sample-br
```

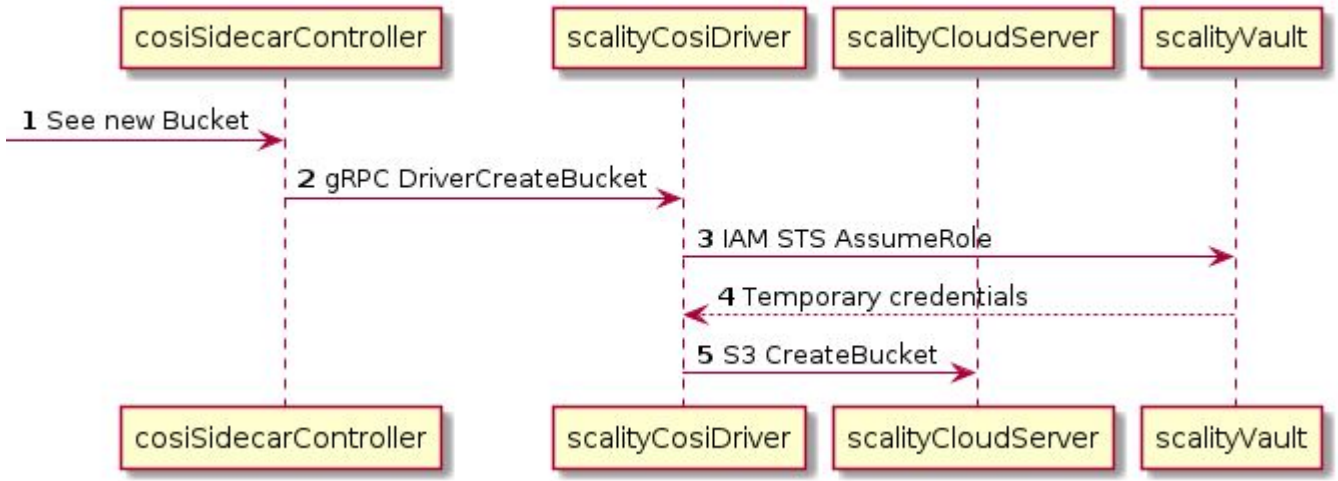
 

# Implementation - Driver Methods



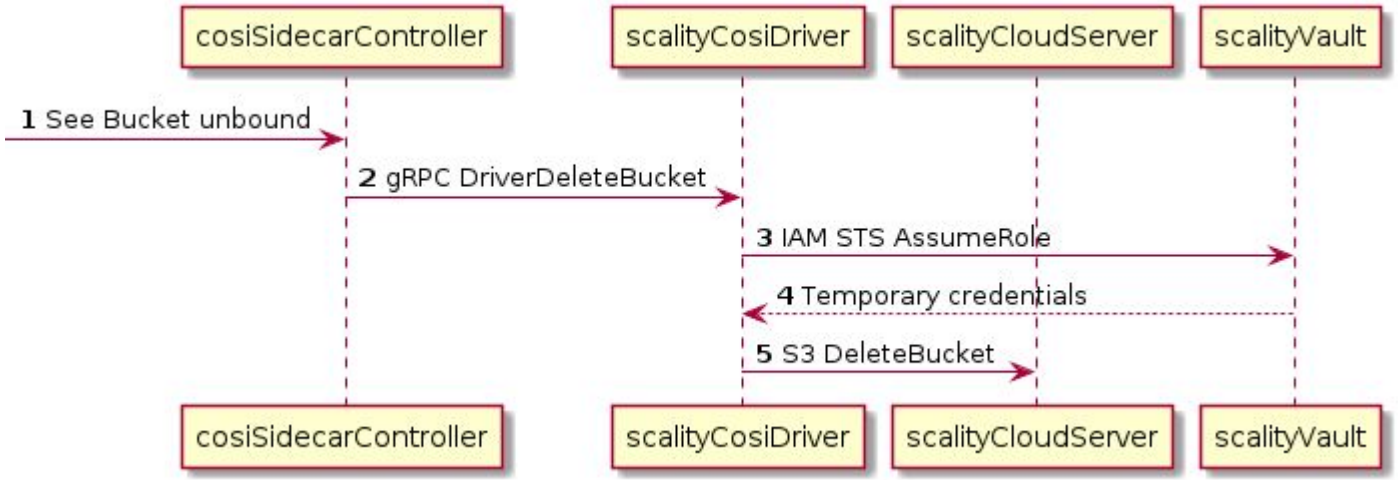

 

# Example Implementation - Driver Bucket Creation Sequence



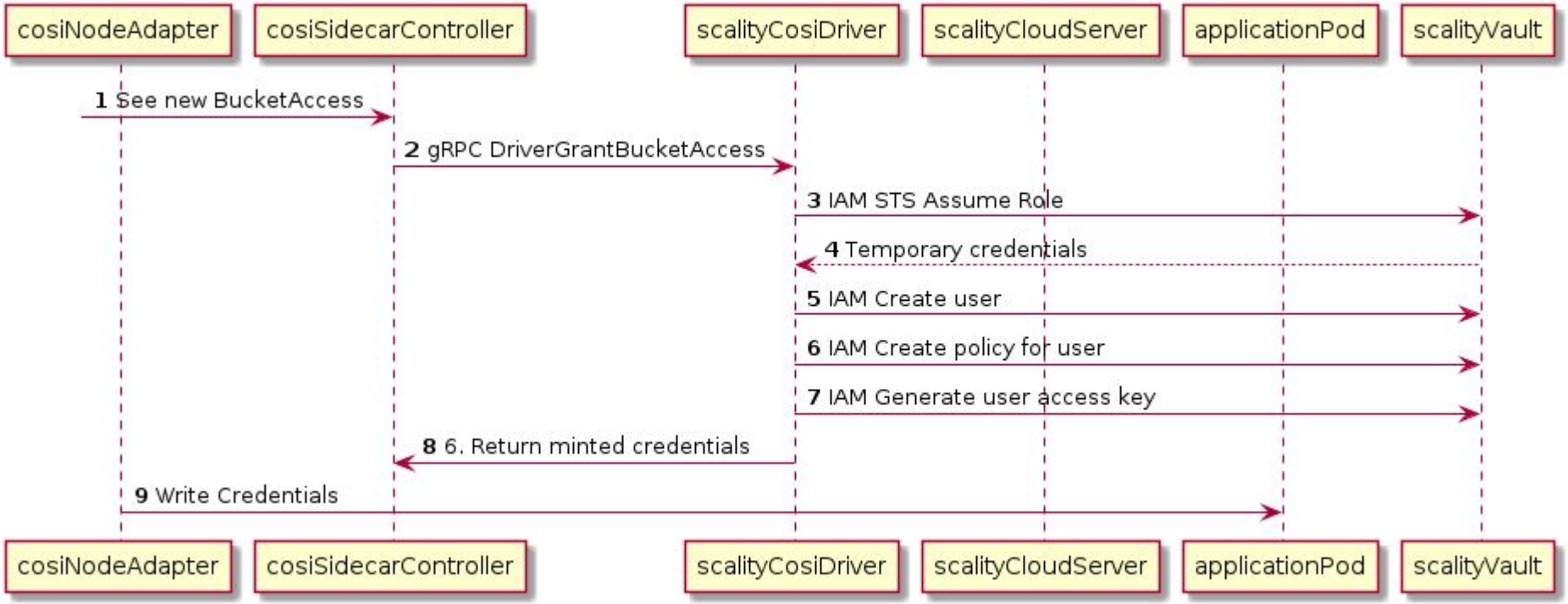

 

# Example Implementation - Driver Bucket Deletion Sequence



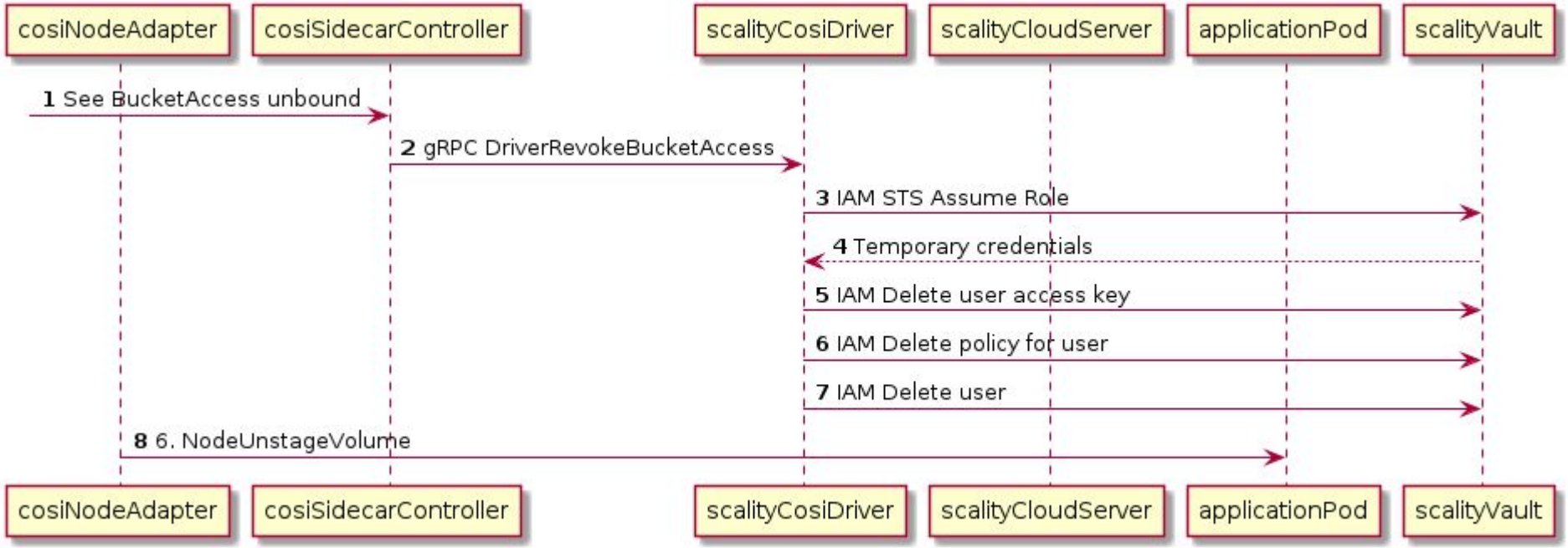

 

# Example Implementation - Grant Access Sequence



# Example Implementation - Revoke Access Sequence



# Additional Links

The COSI spec:
https://github.com/kubernetes/enhancements/blob/master/keps/sig-storage/1979-object-storage-support/README.md

# Please take a moment to rate this session.

Your feedback is important to us.