SNIA CSTI | CLOUD STORAGE TECHNOLOGIES

Kubernetes in the Cloud (Part 2)

Live Webcast
July 17, 2019
10:00 am PT

Today's Presenters

SNIA CSTI | CLOUD STORAGE TECHNOLOGIES

Michelle Tidwell
Program Director, Global Offering Manager, Software Defined Storage, IBM

Matt LeVan
IBM Storage Solutions Architect

Tom Clark
IBM Distinguished Engineer, Chief Architect Storage Software

SNIA Legal Notice

SNIA CSTI | CLOUD STORAGE TECHNOLOGIES

- The material contained in this presentation is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

SNIA-At-A-Glance

SNIA CSTI | CLOUD STORAGE TECHNOLOGIES

185
industry leading organizations

2,000
active contributing members

50,000
IT end users & storage pros worldwide

SNIA CSTI | CLOUD STORAGE TECHNOLOGIES

What We Do

Educate vendors and users on cloud storage, data services and orchestration

Support & promote business models and architectures: OpenStack, Software Defined Storage, Kubernetes, Object Storage

Understand Hyperscaler requirements Incorporate them into standards and programs

Collaborate with other industry associations

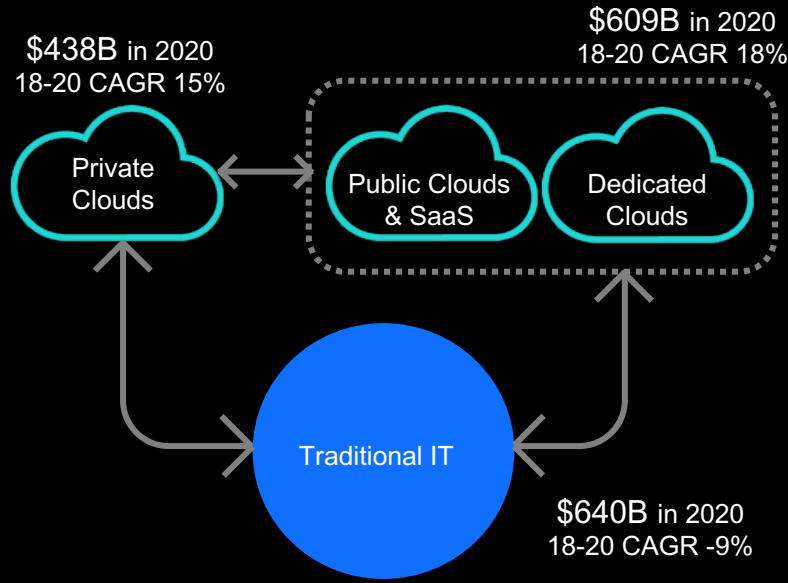
Agenda

SNIA CSTI | CLOUD STORAGE TECHNOLOGIES

- Transition to a Hybrid, Multicloud World
- Kubernetes, Containers and Container Ready Storage
- Container Storage Interface (CSI)
- FlexVolume
- Demo
- Q&A

IT Transition to a Hybrid, Multicloud World

SNIA CSTI | CLOUD STORAGE TECHNOLOGIES



A real world look at multicloud

94% Share of enterprise customers using multiple clouds

67% Share of enterprise customers using more than one public cloud provider

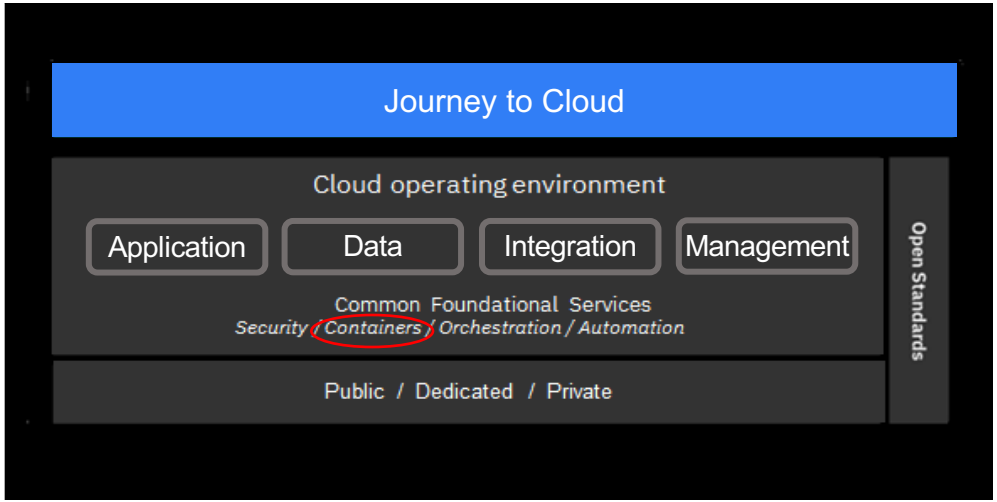
Movement between clouds	73%	priority concern
Connectivity between clouds	82%	priority concern
Consistency of management	67%	priority concern

Source: IBM MD&I; :McKinsey research

Journey to Cloud on Container Infrastructure

Todd
Operations / Admin
Responsible for infrastructure, security, and management of the environment

Jane
Enterprise Developer
Responsible for modernizing existing applications and creating new Cloud Native Workloads



Migrate
Lift & shift applications and workloads

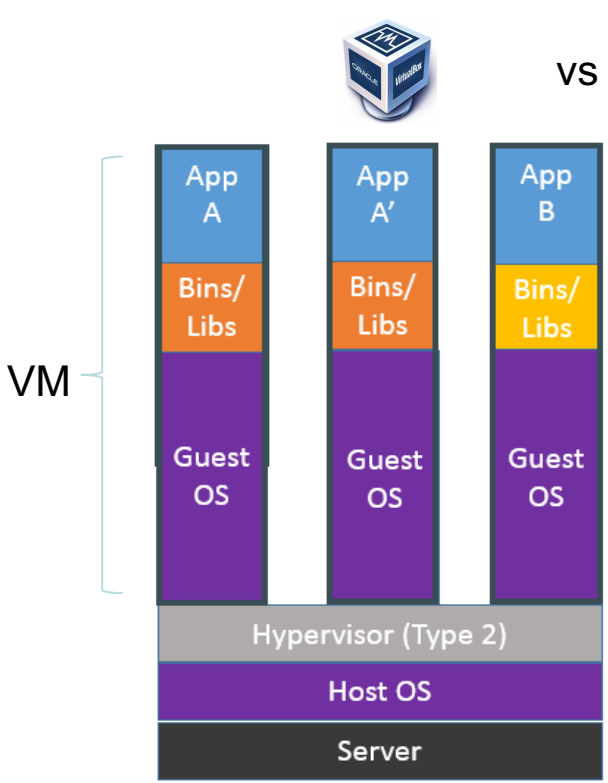
Modernize
Update using containers and microservices

Build
Create new cloud native applications

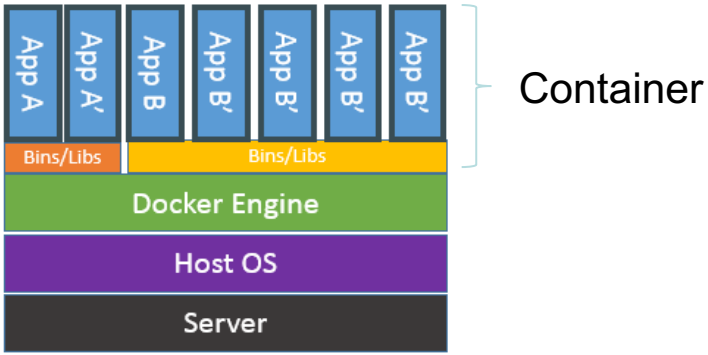
Manage
Integrate and manage, multicloud

Overview of Kubernetes, Containers and Container Ready Storage

Virtual Machines and Containers



Containers are isolated, but share OS and, wherever it's appropriate, bin/libs



The Storage Challenge with Running Containerized App

SNIA CSTI | CLOUD STORAGE TECHNOLOGIES

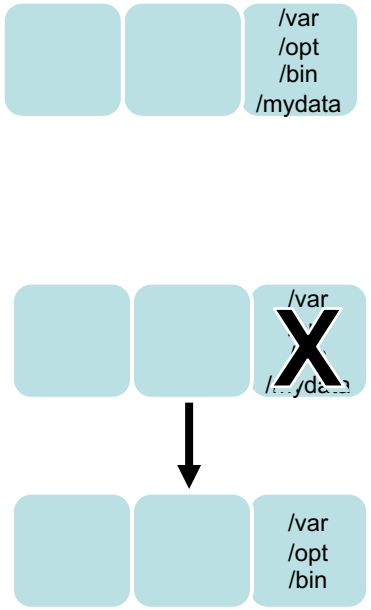
Challenge – Containers are Ephemeral

1. When I run a Stateless (non-persistent Storage) application in a container, I don't expect my data to be stored for future usage.

Note: One container holds the directory and structure of the application.

2. If container should reconstitute, there will be no reference to any data-set outside of the container.

Stateless Environment



Solution for Stateful applications – Storage Enabler for Containers

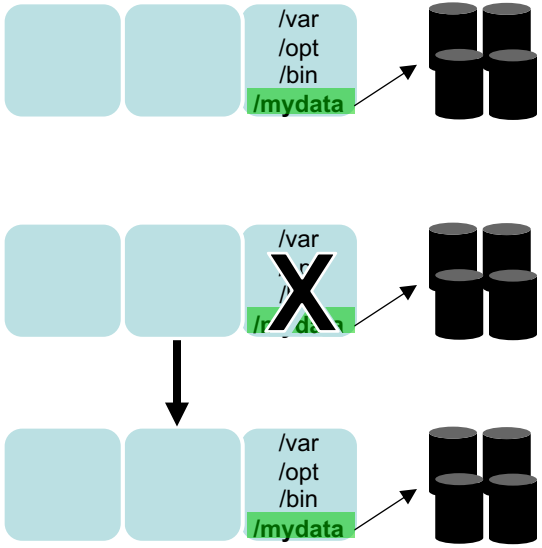
1. When I run a Stateful (Persistent Storage) application in a container, I expect my data to be stored for future usage.

Note: Persistent Volume is created on attached storage.

2. If container should reconstitute, data remains intact on the attached volume.

Note: Volume can be attached to a new container on different host

Stateful Environment



Kubernetes Types of Drivers

In Tree Drivers

- Drivers that were built, linked, compiled and shipped with the core Kubernetes binaries and extend the core Kubernetes API.
- Challenging to add support for new volume plugins to Kubernetes.
- Vendors were forced to align with the Kubernetes release process.
- Third-party storage code caused reliability and security issues in core Kubernetes binaries.
- Have not been accepted since Kubernetes 1.8.

FlexVolume

- Uses an exec-based model to interface with drivers.
- Vendor drivers would be installed in the volume plugin path on every node, and on master if the driver requires attach capability controlled by master nodes.

Container Storage Interface (CSI)

- The Container Storage Interface (CSI) is a standard for exposing arbitrary block and file storage storage systems to containerized workloads on Container Orchestration Systems (COs) like Kubernetes.
- Using CSI third-party storage providers can write and deploy plugins exposing new storage systems in Kubernetes without ever having to touch the core Kubernetes code.

Basic Storage Operations for FlexVolume and CSI

Create/Delete Volume

- Function responsible for creation and deletion of volumes.

Attach/Detach Volume

- Attach/Map volumes to a node or the whole cluster depending on driver.
- Detach/Unmap the volume from the node. Called from Controller Manager.
- With flex volume can be handled by master or non-master nodes depending on driver.

Mount/Unmount Volume

- Mounts the device to a global path which individual pods can then bind mount. Called only from Kubelet. Some drivers will also create a file system for Block storage devices.
- Unmounts the global mount for the device. This is called once all bind mounts have been unmounted. Called only from Kubelet.
- The function will be handled by a Kubernetes sidecar with CSI drivers.

Kubernetes Storage Terminology

PersistentVolume (PV)

- Storage that has been static provisioned by an administrator or dynamically provisioned using a StorageClass.
- Lifecycle independent of any individual pod that use the PV.
- API object that captures the the details of the implementation of the storage.

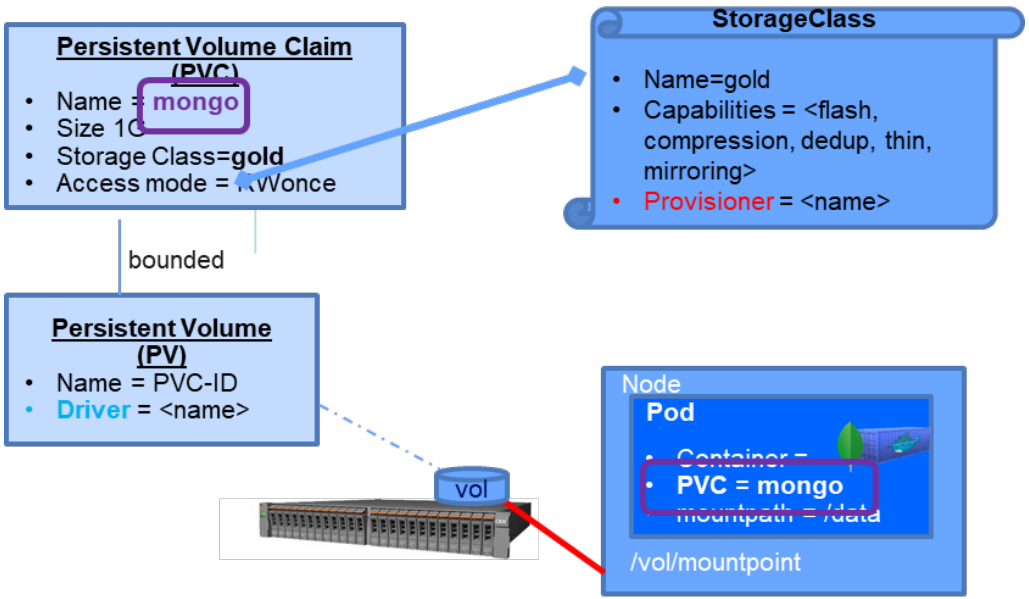
PersistentVolumeClaim (PVC)

- Request for storage by a user requesting size and access modes.
- Allows a user to consume abstract storage resources with varying properties.
- PersistentVolumeClaim will consume a PersistentVolume that meets its requirements with static provisioning or create a PersistentVolume with dynamic provisioning.

StorageClass (SC)

- Provides an administrator the ability to describe the classes of storage offered.
- Can be use to map to different quality-of-services, backup options, etc.
- Provisioner (volume plugin) is defined here to determine where to send the request to.

Kubernetes Storage Terminology



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

Storage Class

To achieve dynamic volume creation, the admin must define a k8s **StorageClass** (e.g : gold, silver).

Provision a volume

1. The user creates a claim for volume (**PVC).**
2. The "Provisioner" (vendor specific) listens to new PVC requests, and dynamically creates the volume on the storage system (if no PV already matched)
   - The **PV** is created with "**Driver**" setting. The Driver(vendor specific) handles the volume attach\detach to the node.

Create a stateful POD

1. The user creates a **POD** with the new PVC.
2. K8s triggers the "**Driver**" in order to attach the PV to the node.
3. The volume is now mapped and mounted to the node.
4. k8s starts the POD with the PV mounted to /data inside the container.

✓ The stateful container is UP

Kubernetes Volume Information

Reclaim Policy

- Retain – manual reclamation required.
- Delete – associated storage asset is deleted.
- Recycle – basic scrub (rm –rf /thevolume/*). Recycle has been deprecated, instead the recommended approach is to utilize Dynamic Provisioning.

Volumes that were dynamically provisioned will inherit the Reclaim Policy of their StorageClass, which defaults to Delete.

Access modes

- ReadWriteOnce (RWO) – the volume can be mounted as read-write by a single node
  - Appropriate for databases.
- ReadWriteMany (RWX) – the volume can be mounted as read-write by many nodes
  - Appropriate for shared logs, web files, AI ML DL data sets, etc.
- ReadOnlyMany (ROX) – the volume can be mounted as read-only by many nodes.

What Type of Storage to Use When?

Block

- Block is good for RWO once access mode and could be used for RWX if the application can maintain consistency.
- Use cases include databases.

File

- File is good for both RWO and RWX as the underlying file system is designed for multiple access.
- Different access methods are available, GPFS, NFS or SMB.
- Use cases include logs, AI ML DL data sets.

Object

- Object does not currently have a construct in Kubernetes.
- Access is provided through the S3 API accessed by the application or through the use of S3 fuse based file system backed by an object store.
- Use cases include images, documents, backups.

Overview of Container Storage Interface (CSI)

What's Coming for Containers – Container Storage Interface (CSI)

Standardized storage interface for Container Orchestration (CO) Systems

- Kubernetes, Mesos, Docker, and Cloud Foundry
- In addition, makes installing new volume plugins as easy as deploying a pod

Driven and specified by the CNCF (Cloud Native Computing Foundation)

CSI spec V1.0 GA level of support in Kubernetes 1.13 (Dec 2018)

Flex Volume plugins can coexist with CSI plugins. SIG Storage will continue to maintain the Flex API so that existing plugins will continue to work.

CONTAINER STORAGE INTERFACE

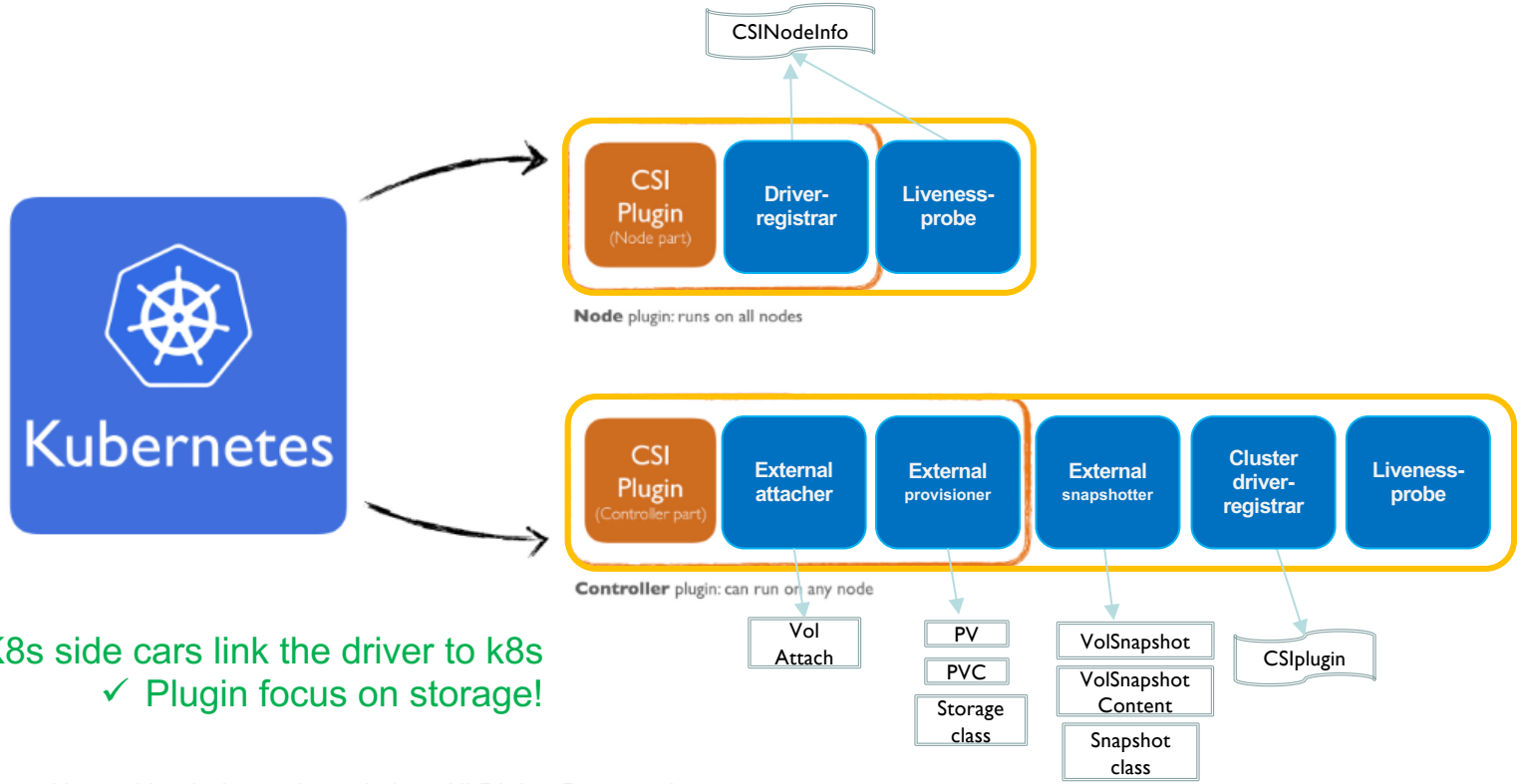
What's Coming for Containers – Container Storage Interface (CSI)

New volume features will be added only to CSI (not to current Flex Volume)

Future volume features to be added to CSI and adopted by future Kubernetes versions include:

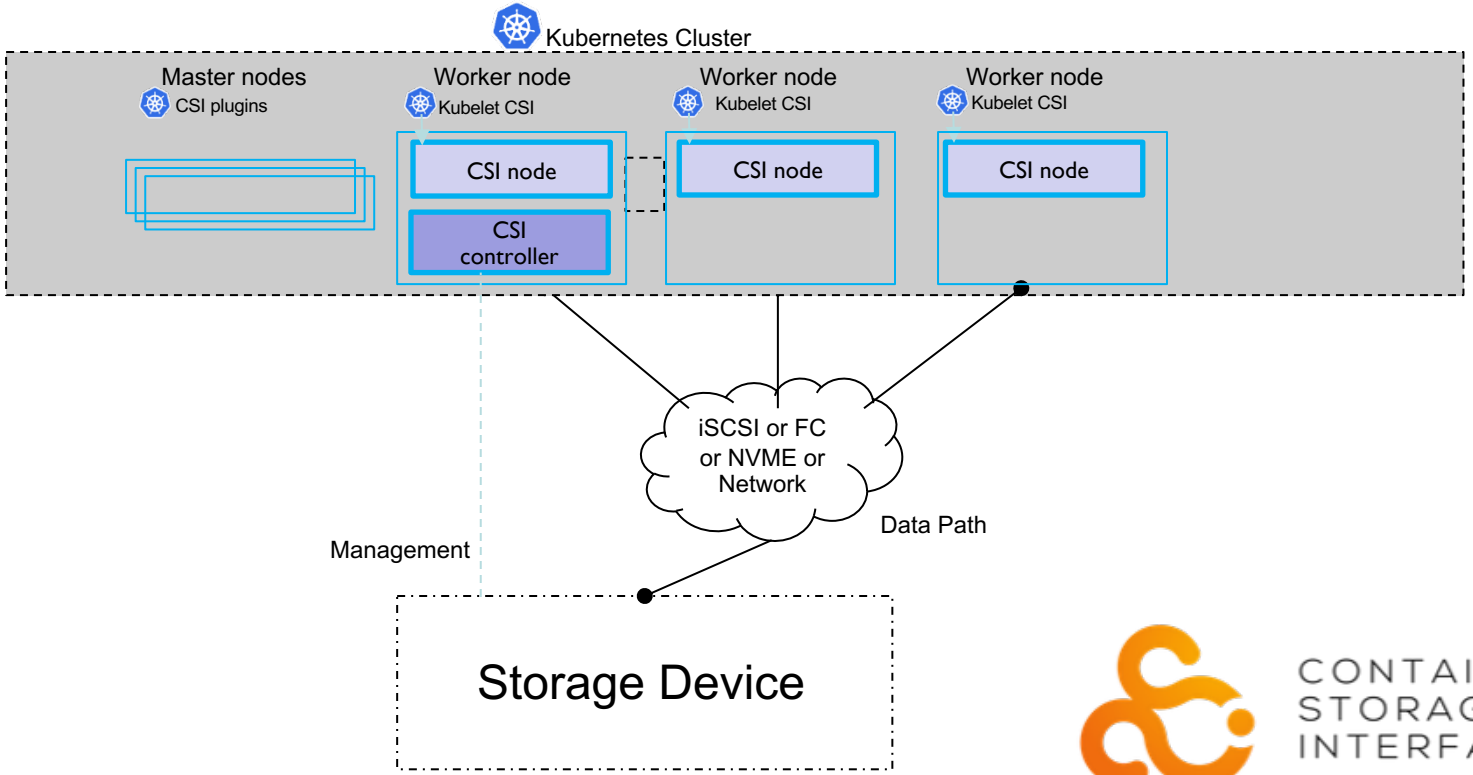
- Snapshots (Alpha in Kubernetes 1.13)
  - Planned to be beta in either 1.15 or 1.16 (2H2019)
- Cloning
- Topology aware volume provisioning
- Volume resize (Alpha in Kubernetes 1.14)
  - Target GA ~ end of 2019
- Raw block support

CSI in Kubernetes



✓ K8s side cars link the driver to k8s
✓ Plugin focus on storage!

Kubernetes CSI Block Plugin – High Level Architecture



Volume Snapshots

SNIA CSTI | CLOUD STORAGE TECHNOLOGIES

VolumeSnapshot

- Request by a user for a snapshot of a volume and support is only available with CSI drivers.
- Kubernetes sidecar, external-snapshotter, watches VolumeSnapshot objects and triggers CreateSnapshot and Deletesnapshot operations against CSI endpoints.
- Similar to PersistentVolumeClaim.

VolumeSnapshotContent

- Snapshot taken from a volume in the cluster.
- Contains a specification of the volume snapshots, such as source, creation time, snapshot handle.
- Similar to PersistentVolume.

VolumeSnapshotClass

- Used by VolumeSnapshot with the attribute, snapshotClassName.
- Contains snapshotter and parameters to be used with dynamic provisioned snapshot.

Persistent Volume in Use Protection

- Ensures that in-use PVC API objects are not removed from the system (as this may result in data loss).
- If a user deletes in active use, the PVC is not immediately removed and postponed until the PVC is no longer used by any snapshots.

Overview of FlexVolume

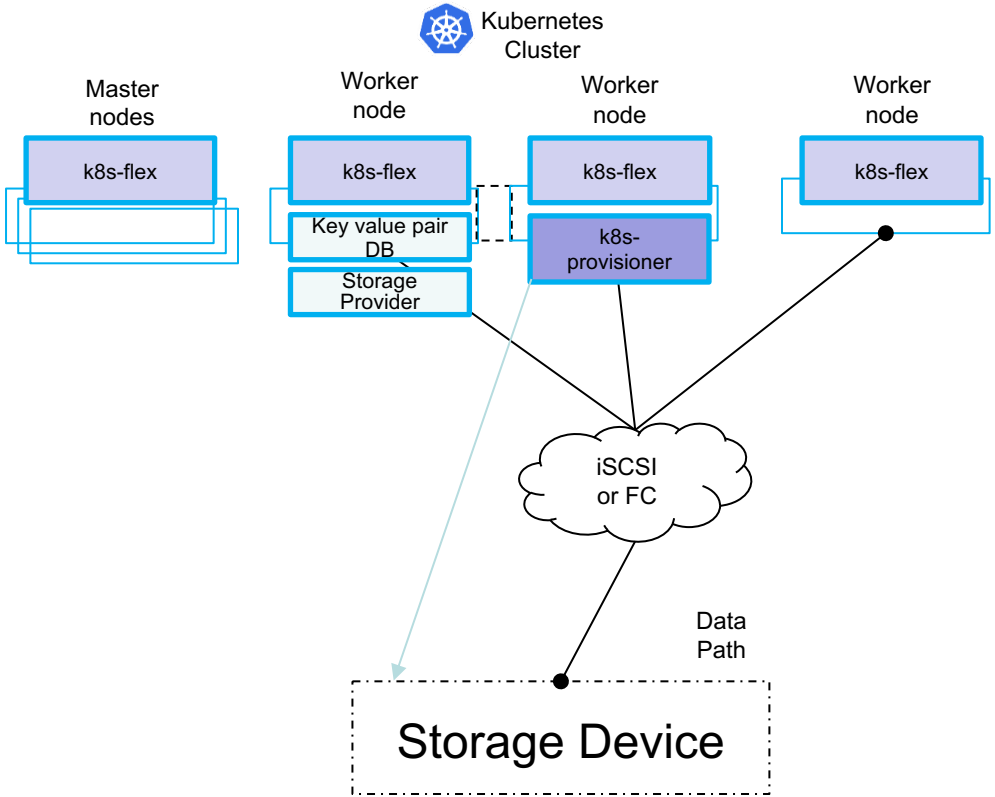
Solution Architecture for Block

Provision flow

The k8s user creates a new PVC and then:

1. The provisioner identifies the pending PVC and passes the request to provider.
2. Provider validates the authentication of the request, and then passes it to the storage device.
3. Storage device provisions the volume on the relevant array based on the profile.
4. A volume is created on the storage.
5. The provisioner creates PV object for the newly created volume.

✓ Provisioning done



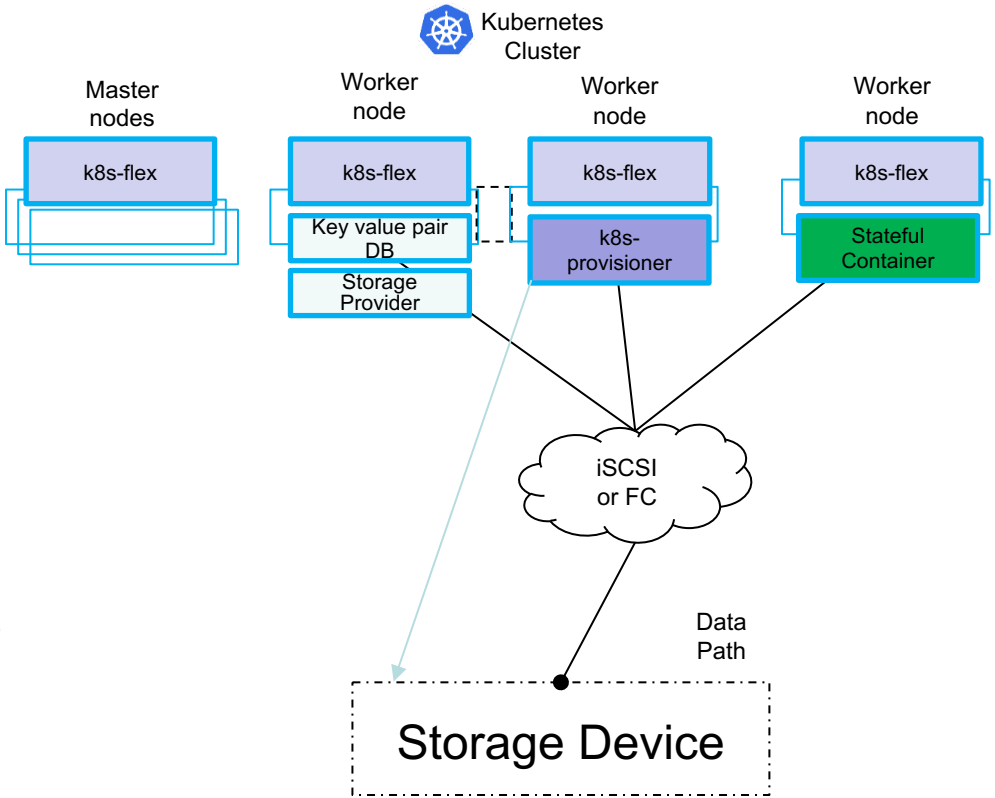
Solution Architecture for Block

Create stateful container

The k8s user creates a Pod (with the PVC) and then:

1. K8s controller-manager (master node) triggers the flex attach API (with PV and host). Flex passes the request to provider.
2. Provider validates the authentication of the request, and then passes it to storage device.
3. Storage device maps the volume on the storage to the worker node.
4. Flex on the worker node rescans the OS and identifies the new multipath device, creates a filesystem and mount to /path/<WWN>.
5. K8s starts the container and exposes the hosts mountpoint inside the container.

✓ Stateful container is running



Demo

Static provisioning with NFS

SNIA CSTI | CLOUD STORAGE TECHNOLOGIES

```
# snia-nfs-pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: snia-nfs-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    path: /exports/snia-nfs-pv
    server: 10.10.32.84
    readOnly: false
```

```
# snia-nfs-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: snia-nfs-pvc
spec:
  storageClassName: ""
  accessModes:
  - ReadWriteMany
  resources:
     requests:
       storage: 1Gi
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: snia-webinar-nfs
  namespace: default
  labels:
    app: snia-webinar-nfs
spec:
  replicas: 1
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  selector:
    matchLabels:
      app: snia-webinar-nfs
  template:
    metadata:
      labels:
        app: snia-webinar-nfs
    spec:
      containers:
      - name: container-nfs
        image: alpine:latest
        command: [ "/bin/sh", "-c", "--" ]
        args: [ "while true; do sleep 30; done;" ]
        volumeMounts:
          - name: snia-nfs-pvc
            mountPath: "/data"
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
      volumes:
        - name: snia-nfs-pvc
          persistentVolumeClaim:
            claimName: snia-nfs-pvc
```

Dynamic Provisioning Example

```
# gold-storageclass.yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: "gold"                # Storage Class
name
#  annotations:
#   storageclass.beta.kubernetes.io/is-
default-class: "true"
provisioner: "ubiquity/flex"
parameters:
  profile: "gold"
  fstype: "xfs"
  backend: "scbe"
```

```
# snia-webinar-pvc.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: "snia-webinar-pvc"
spec:
  storageClassName: gold
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

```
# snia-webinar-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: snia-webinar-deploy
  namespace: default
  labels:
    app: snia-webinar-deploy
spec:
  replicas: 1
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  selector:
    matchLabels:
      app: snia-webinar-deploy
  template:
    metadata:
      labels:
        app: snia-webinar-deploy
    spec:
      containers:
      - name: container1
        image: alpine:latest
        command: [ "/bin/sh", "-c", "--" ]
        args: [ "while true; do sleep 30; done;" ]
        volumeMounts:
          - name: snia-webinar-pvc
            mountPath: "/data"
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
      volumes:
        - name: snia-webinar-pvc
          persistentVolumeClaim:
            claimName: snia-webinar-pvc
```

Dynamic Storage Provisioning Demo via Helm

Deploy MongoDB with Dynamic Storage Provisioning

1. Select Menu -> Catalog -> Helm Charts
2. Select ibm-mongodb-dev from the Catalog
3. Select Configure
4. Enter release name (mongodb-demo), target namespace (default), Existing storage class name (gold)
5. Select Use dynamic provisioning for persistent volume
6. Accept license agreement and select Install
7. View Helm Release
8. Select mongodb-test helm release
9. Verify PersistentVolume has been created and Bound

Persistent Volume Claim

NAME	STATUS	VOLUME	CAPACITY	ACCESSMODES	STORAGECLASS	AGE
mongodb-demo-ibm-mongodb-dev-datavolume	Bound	pvc-a377b237-4198-11e8-b9f7-00505697a136	20Gi	RWO	ibmc-block-gold	38s

Persistent Volume Claim Provisioning Demo via Helm

Creating the PVC

1. Select Menu -> Platform -> Storage
2. Select PersistentVolumeClaim from the tabs
3. Create PersistentVolumeClaim
4. Enter Name (mariadb-pvc), Storage class name (gold), Storage requests (10 Gi)
5. Select Create and verify status is Bound and PersistentVolume created

Deploy MariaDB with Existing PVC

1. Select Menu -> Catalog -> Helm Charts
2. Select ibm-mariadb-dev from the Catalog
3. Select Configure
4. Enter release name (mariadb-demo), target namespace (default), Existing volume claim (mariadb-pvc)
5. Accept license agreement and select Install
6. View Helm Release
7. Select mongodb-test helm release
8. Select Deployment and then select Pods and Events. Verify the PVC mountvolume succeeded

This is a Webcast Series…

SNIA CSTI | CLOUD STORAGE TECHNOLOGIES

- Kubernetes in the Cloud (Part 1) - Available on demand at https://www.brighttalk.com/webcast/663/350613
- Kubernetes in the Cloud (Part 3) – Coming Soon! Follow us on Twitter @SNIACloud for date & time

After This Webcast

SNIA CSTI | CLOUD STORAGE TECHNOLOGIES

- Please rate this webcast and provide us with feedback
- This webcast and a PDF of the slides will be posted to the SNIA Cloud Storage Technologies Initiative website and available on-demand at https://www.snia.org/forum/csti/knowledge/webcasts
- A full Q&A from this webcast will be posted to the SNIA Cloud blog: www.sniacloud.com/
- Follow us on Twitter @SNIACloud

Thank You