SDC 20

BY Developers FOR Developers

Storage Developer Conference
September 22-23, 2020

SPDK-CSI :
Bring SPDK to Kubernetes Storage

Yibo Cai

arm

About Me

- Principal Software Engineer from Arm
- 20 years in software development
- Work on improving Open Source Software (OSS) ecosystem on Arm servers
- Focused area
  - Storage
  - Data Science

Agenda

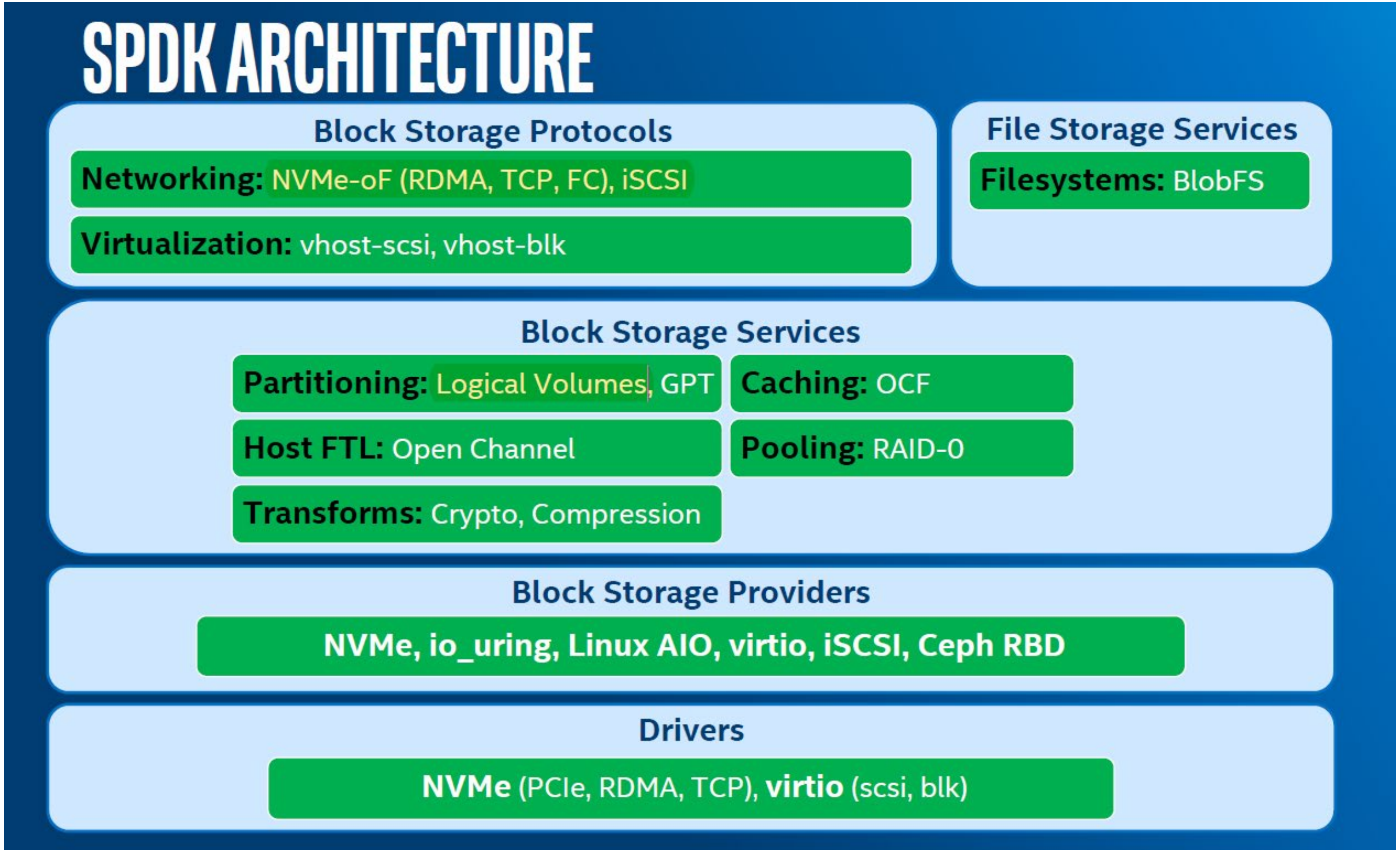
- SPDK and Kubernetes Storage overview
- Container Storage Interface (CSI)
  - CSI Internals
  - Kubernetes CSI support
- SPDK-CSI Implementation details
- Community
  - Contribution guidelines
  - Project status and plans

SPDK, Kubernetes Storage

What is SPDK

- Quoted from https://spdk.io/
  - The Storage Performance Development Kit (SPDK) provides a set of tools and libraries for writing *high performance, scalable, user-mode* storage applications.
- Key techniques
  - Interact with hardware directly in user space
  - Polling data readiness instead of interrupt
  - No locks in I/O path

What is SPDK



SPDK ARCHITECTURE

Block Storage Protocols

Networking: NVMe-oF (RDMA, TCP, FC), iSCSI

Virtualization: vhost-scsi, vhost-blk

File Storage Services

Filesystems: BlobFS

Block Storage Services

Partitioning: Logical Volumes, GPT

Caching: OCF

Host FTL: Open Channel

Pooling: RAID-0

Transforms: Crypto, Compression

Block Storage Providers

NVMe, io_uring, Linux AIO, virtio, iSCSI, Ceph RBD

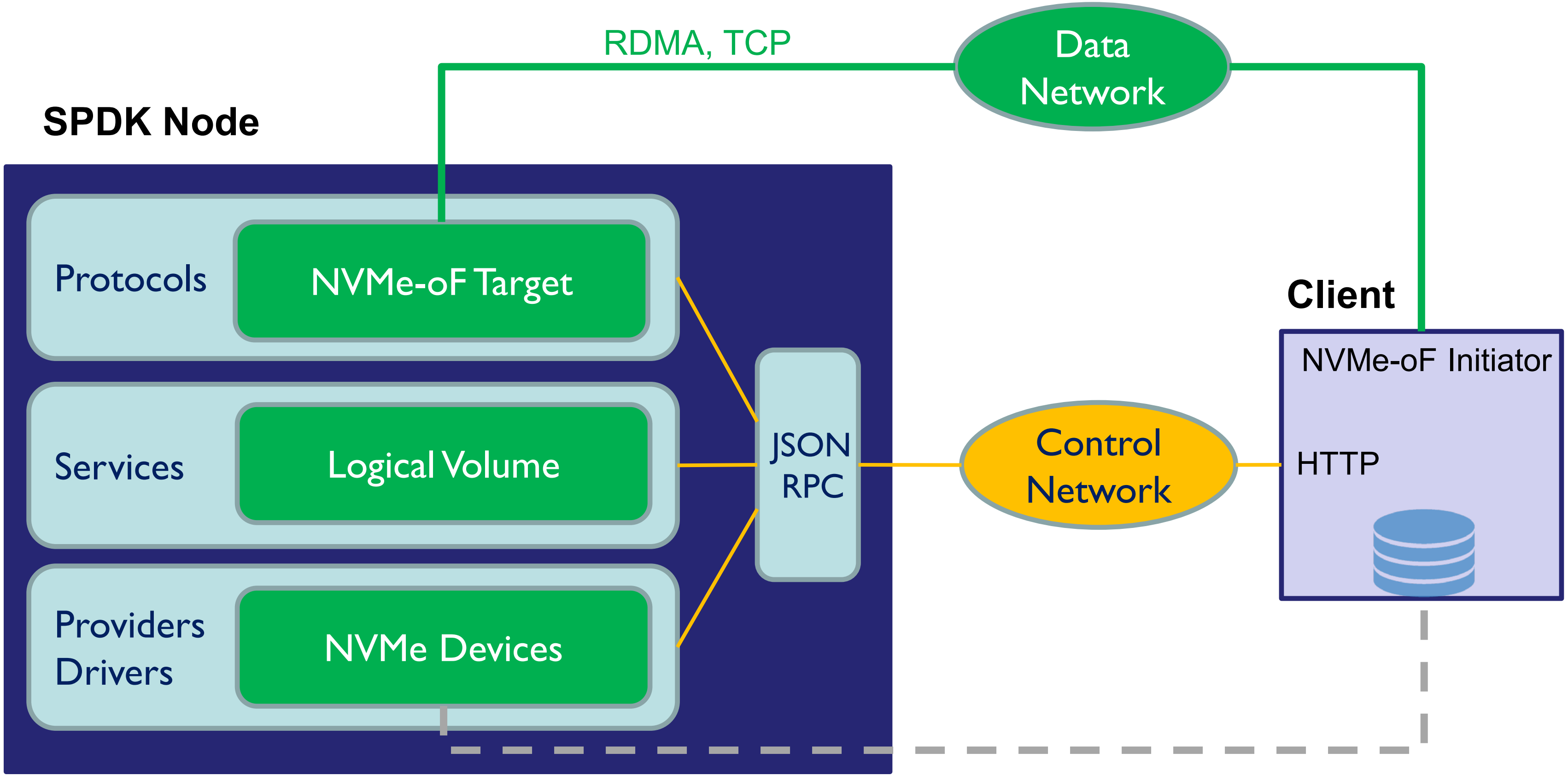
Drivers

NVMe (PCIe, RDMA, TCP), **virtio** (scsi, blk)

SPDK Network Storage

- Network storage protocols
  - SPDK implements NVMe-oF and iSCSI targets above its block device layer
- Volume management
  - SPDK Logical Volume provides flexible block device interface to local applications and network targets
- Remote configuration
  - SPDK supports JSON-RPC for remote configuration of Logical Volumes and NVMe-oF/iSCSI targets

SPDK Network Storage



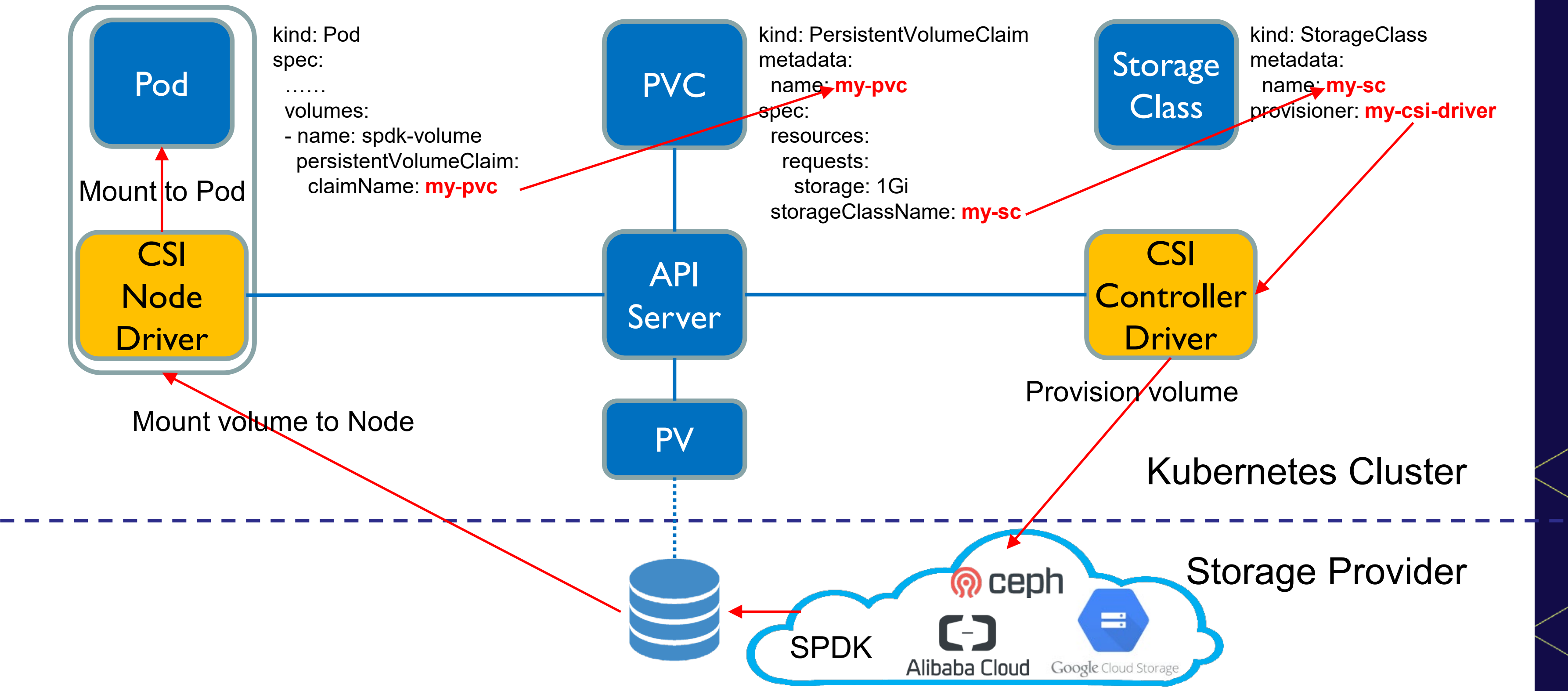
Kubernetes Storage

- Kubernetes volume driver: a brief history
  - In-Tree: storage driver coupled in Kubernetes code base. Deprecated, legacy code will be removed.
  - FlexVolume: exec based API for volume plugins. Hard to deploy and manage dependency.
  - Container Storage Interface (CSI): Addresses pains of In-Tree and FlexVolume. Standardizes storage system integration with Kubernetes.
    - Kubernetes CSI Drivers List

Kubernetes Storage

- Dynamic volume management
  - Storage Class
  - Persistent Volume (PV)
  - Persistent Volume Claim (PVC)
- How dynamic volume provisioning works
  - Pod claims block storage from a storage class (CSI driver)
  - CSI controller driver creates the block device through storage provider (maybe local or on the cloud)
  - CSI node driver mounts the block device to Pod

Dynamic Volume Provisioning



```
kind: Pod
spec:
  ……
  volumes:
  - name: spdk-volume
    persistentVolumeClaim:
      claimName: my-pvc
```

```
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  resources:
    requests:
      storage: 1Gi
  storageClassName: my-sc
```

```
kind: StorageClass
metadata:
  name: my-sc
provisioner: my-csi-driver
```

SPDK-CSI: Bring SPDK to Kubernetes

- Goals
  - Bring SPDK to Kubernetes storage through NVMe-oF, iSCSI
  - Supports dynamic volume provisioning
  - Enables Pods to use SPDK for transient or persistent storage
- Non-goals
  - Not a complete storage solution
  - SPDK services management is out of the scope

Container Storage Interface (CSI)

Container Storage Interface (CSI)

- From Kubernetes CSI Developer Documentation
  - The Container Storage Interface (CSI) is a standard for exposing arbitrary block and file storage systems to containerized workloads on Container Orchestration Systems (COs) like Kubernetes.
  - NOTE: CSI is a general protocol, not for Kubernetes only.
- My point of view
  - Defines a bunch of messages (RPC) between CO and third-party storage driver
  - Volume lifecycle management based on these messages

Container Storage Example

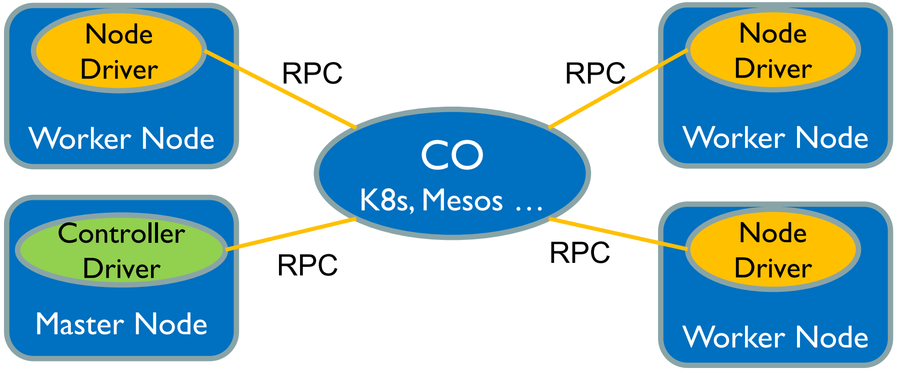
To use Ceph RBD (block device) in a containerized app		
No.	**Steps**	**Run command at**
App starts		
1	Create RBD volume in Ceph cluster through Ceph API	Any host can access Ceph
2	Mount/Format RBD volume to some directory on host	Host where the app runs
3	Mount host RBD directory to container directory	Host where the app runs
App stops		
4	Unmount container directory	Host where the app runs
5	Unmount host RBD directory	Host where the app runs
6	Delete RBD volume in Ceph cluster through Ceph API	Any host can access Ceph

Container Storage Example

To automate above steps in a container cloud	
What we need	**In CSI Term**
[Step 1, 6] A storage driver to handle Ceph API and create/delete RBD on demand. It can run on any host which has access to Ceph cluster control plane.	Controller Driver
[Step 2, 3, 4, 5] A storage driver to (un)mount Ceph RBD volumes. It must run on all hosts where containerized app may be scheduled.	Node Driver
A protocol to define messages between CO and the plugin, so they can cooperate to finish the job.	RPC and Volume Lifecycle

CSI Drivers

- Controller Driver
  - Talk to Service Provider (SP) to create/delete volumes
- Node Driver
  - Mount/unmount remote volumes to local host



~ Controller driver on CO master node

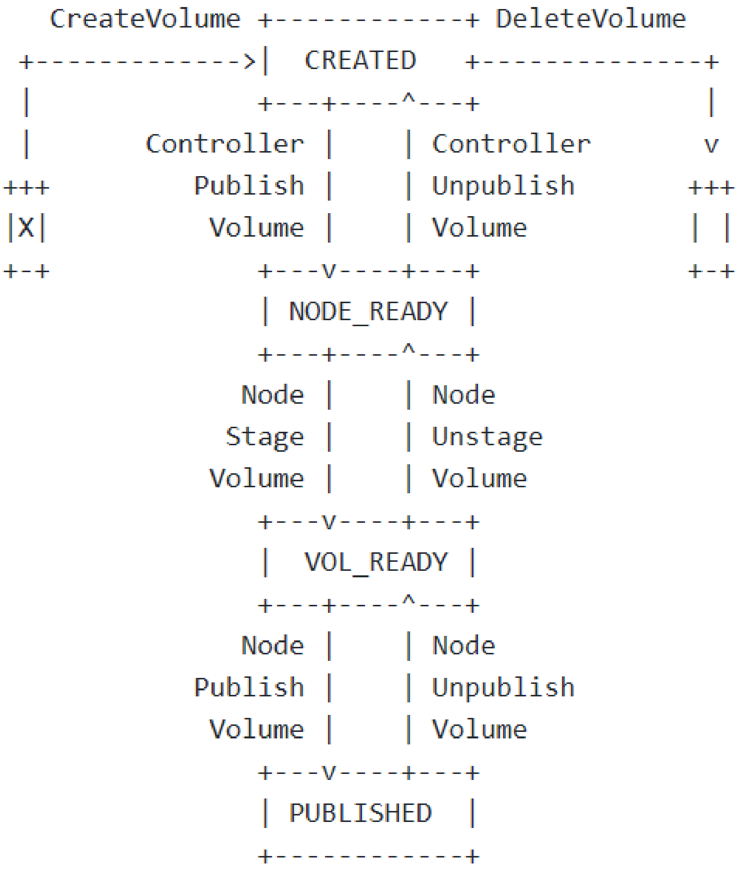
~ Node driver instances per CO worker

~ CO talks to CSI Drivers with CSI RPC messages

Key CSI RPCs

RPC	Explains
CO → Controller Driver	
CreateVolume	Create a volume with specific parameters in storage provider
DeleteVolume	Revert creating
ControllerPublishVolume	Expose the volume to be accessible from worker node
ControllerUnpublishVolume	Revert publishing
CO → Node Driver	
NodeStageVolume	Import remote volume and mount to worker node host
NodeUnstageVolume	Revert staging
NodePublishVolume	Bind mount host staging directory to container internal directory
NodeUnpublishVolume	Revert publishing

Volume Lifecycle



```
   CreateVolume +------------+ DeleteVolume
 +------------->|  CREATED   +--------------+
 |              +---+----^---+              |
 |       Controller |    | Controller       v
+++         Publish |    | Unpublish       +++
|X|          Volume |    | Volume          | |
+-+             +---v----+---+             +-+
                | NODE_READY |
                +---+----^---+
               Node |    | Node
              Stage |    | Unstage
             Volume |    | Volume
                +---v----+---+
                |  VOL_READY |
                +---+----^---+
               Node |    | Node
            Publish |    | Unpublish
             Volume |    | Volume
                +---v----+---+
                | PUBLISHED  |
                +------------+
```

From CSI Spec

Figure 6: The lifecycle of a dynamically provisioned volume, from creation to destruction, when the Node Plugin advertises the STAGE_UNSTAGE_VOLUME capability.

Kubernetes CSI Support

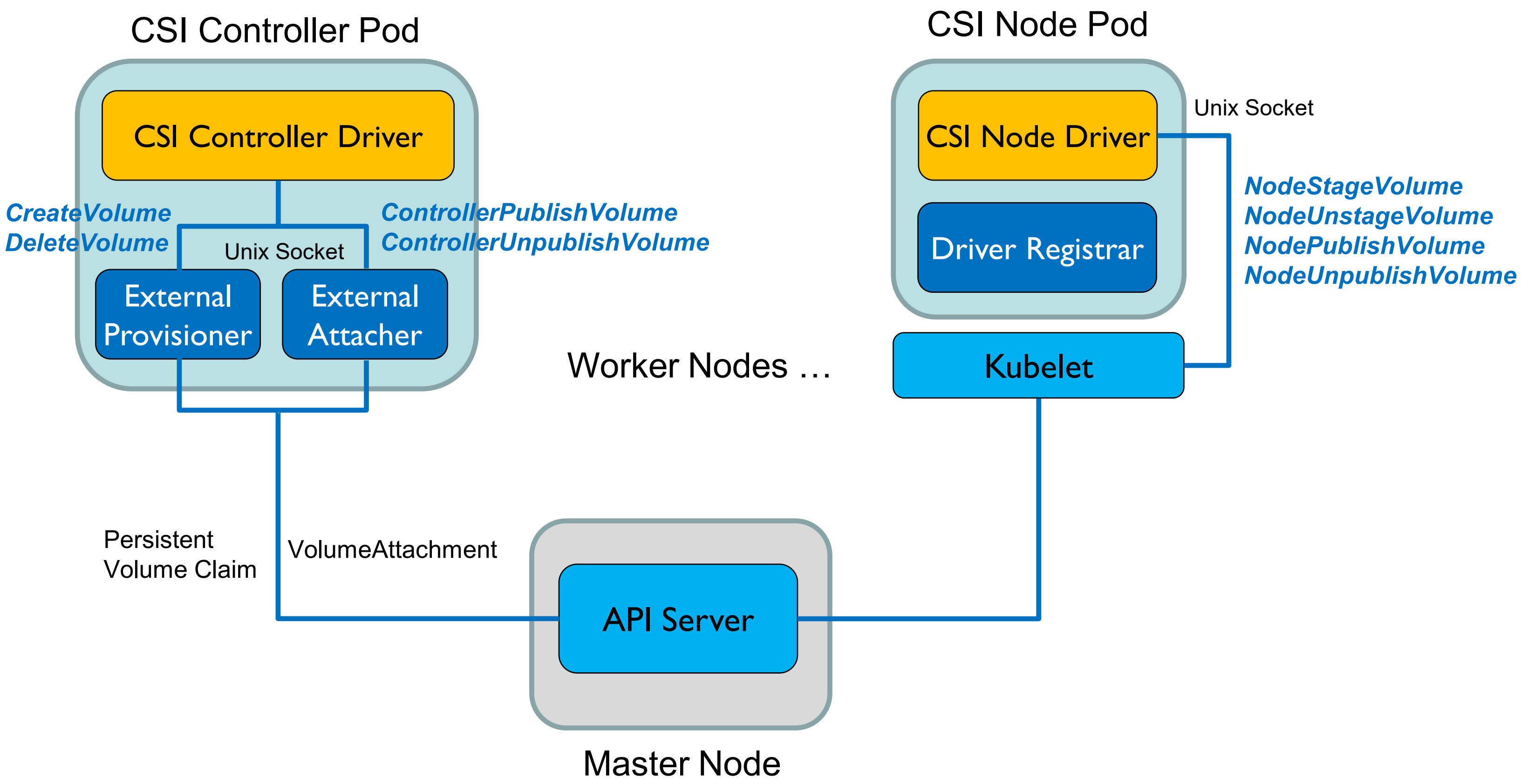
- Kubernetes supports CSI well
  - CSI spec 1.0 supported since Kubernetes 1.13
- Common practice
  - Wrap Controller and Node driver in a single binary
    - Select functionality per command line parameters
  - Deploy Controller driver as Deployment or StatefulSet
  - Deploy Node driver as DaemonSet
    - Exactly one instance on each worker node
  - Leverage CSI Sidecar containers to reduce boilerplate code

Kubernetes CSI Support

- Kubernetes Sidecar Containers
  - Watch Kubernetes objects and send RPC to CSI drivers
  - Free CSI drivers from talking directly to API server

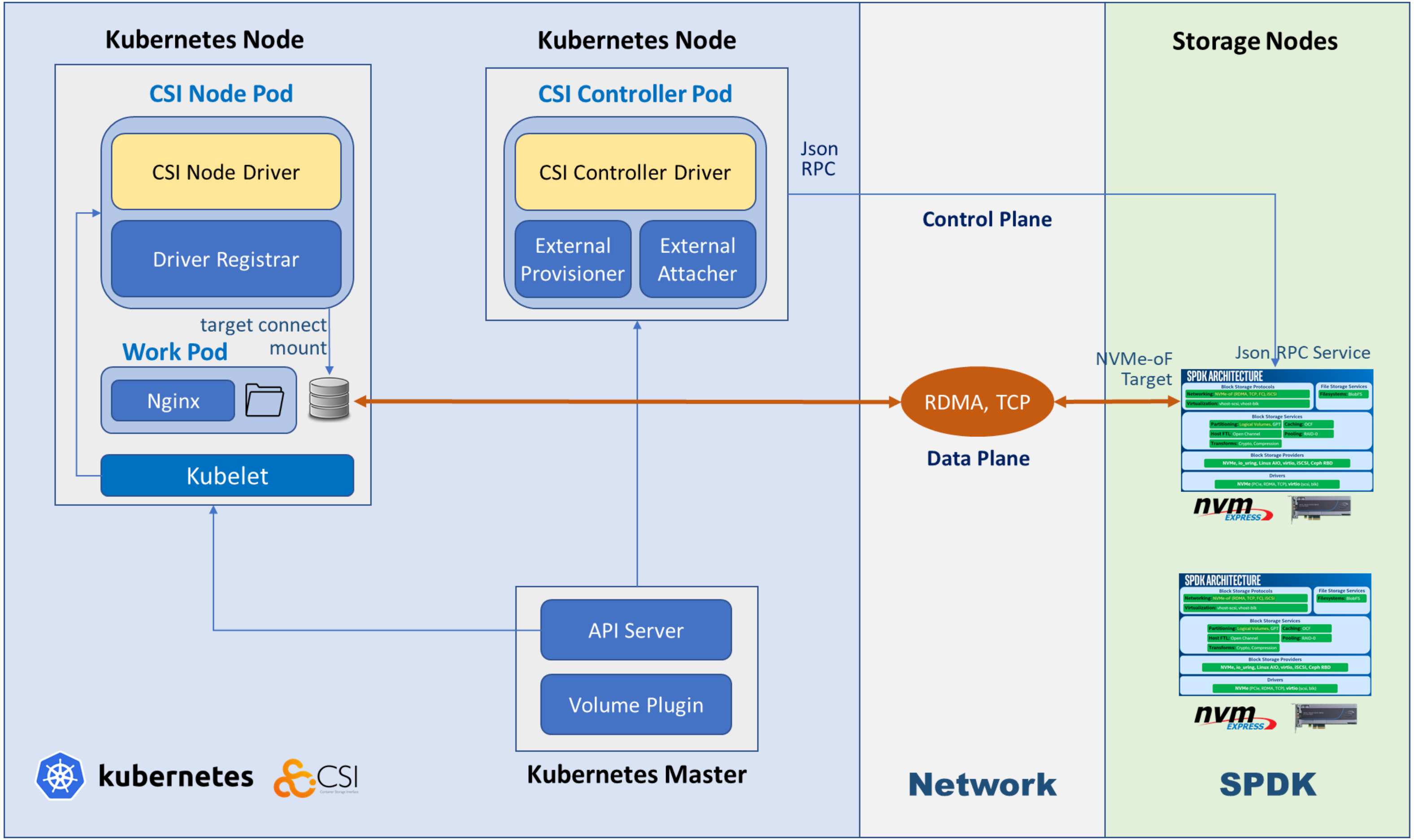
Sidecar	Purpose
External Provisioner	Watches for PersistentVolumeClaim objects and triggers [Create|Delete]Volume operations
External Attacher	Watches for VolumeAttachment objects and triggers Controller[Publish|Unpublish]Volume operations
Node Driver Registrar	Registers the CSI driver with Kubelet to receive Node[Stage|Unstage|Publish|Unpublish]Volume operations
……	……

Kubernetes CSI Support



SPDK-CSI

SPDK-CSI Overview



SPDK-CSI Controller Driver

Controller configures SPDK network target through JSON-RPC

CSI Message	JSON-RPC (NVMf)	JSON-RPC (iSCSI)
CreateVolume	bdev_lvol_create	bdev_lvol_create
DeleteVolume	bdev_lvol_delete	bdev_lvol_delete
ControllerPublishVolume	nvmf_subsystem_add_ns nvmf_subsystem_add_listener	iscsi_create_portal_group iscsi_create_initiator_group iscsi_create_target_node
ControllerUnpublishVolume	nvmf_subsystem_remove_ns	iscsi_delete_target_node

SPDK-CSI Node Driver

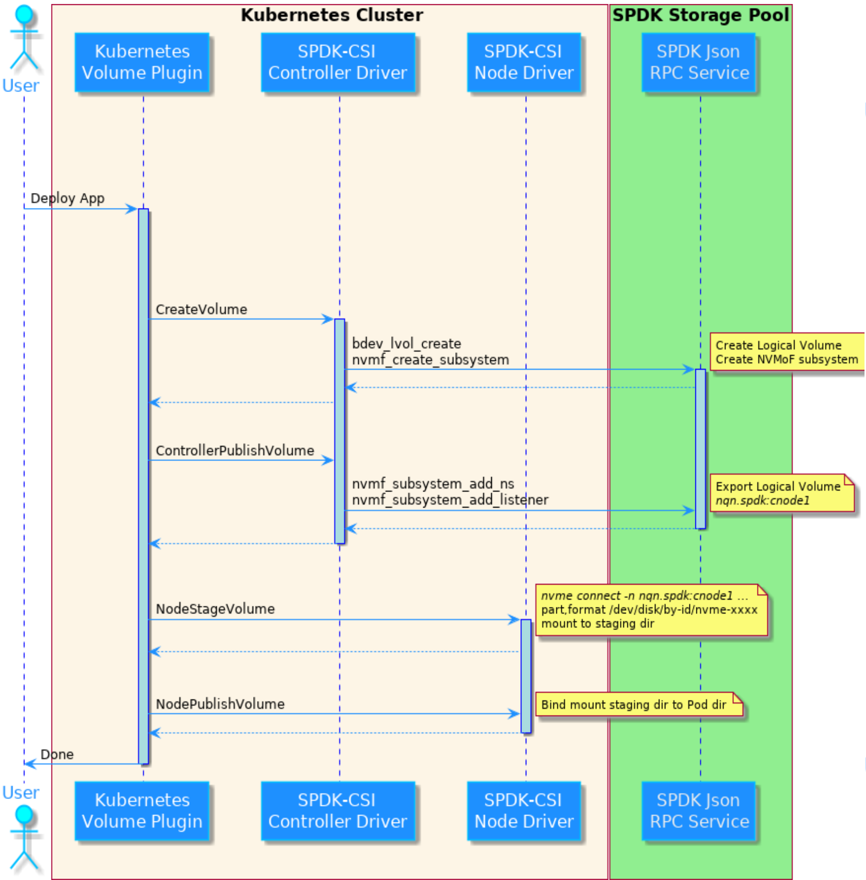
Node connects to SPDK target and mounts remote volume

CSI Message	Node (NVMf)	Node (iSCSI)
StageVolume	nvme connect -n *nqn* -a *ip* -s *port* … mount /dev/disk/by-id/*diskid* stagePath	iscsiadm -p *ip*:*port* -m discovery … iscsiadm -T *iqn* -p *ip*:*port* --login … mount /dev/disk/by-id/*diskid* stagePath
UnstageVolume	nvme disconnect -n *nqn* umount stagePath	iscsiadm -T *iqn* -p *ip*:*port* --logout … umount stagePath
PublishVolume	mount -o bind stagePath podPath	mount -o bind stagePath podPath
UnpublishVolume	umount podPath	umount podPath

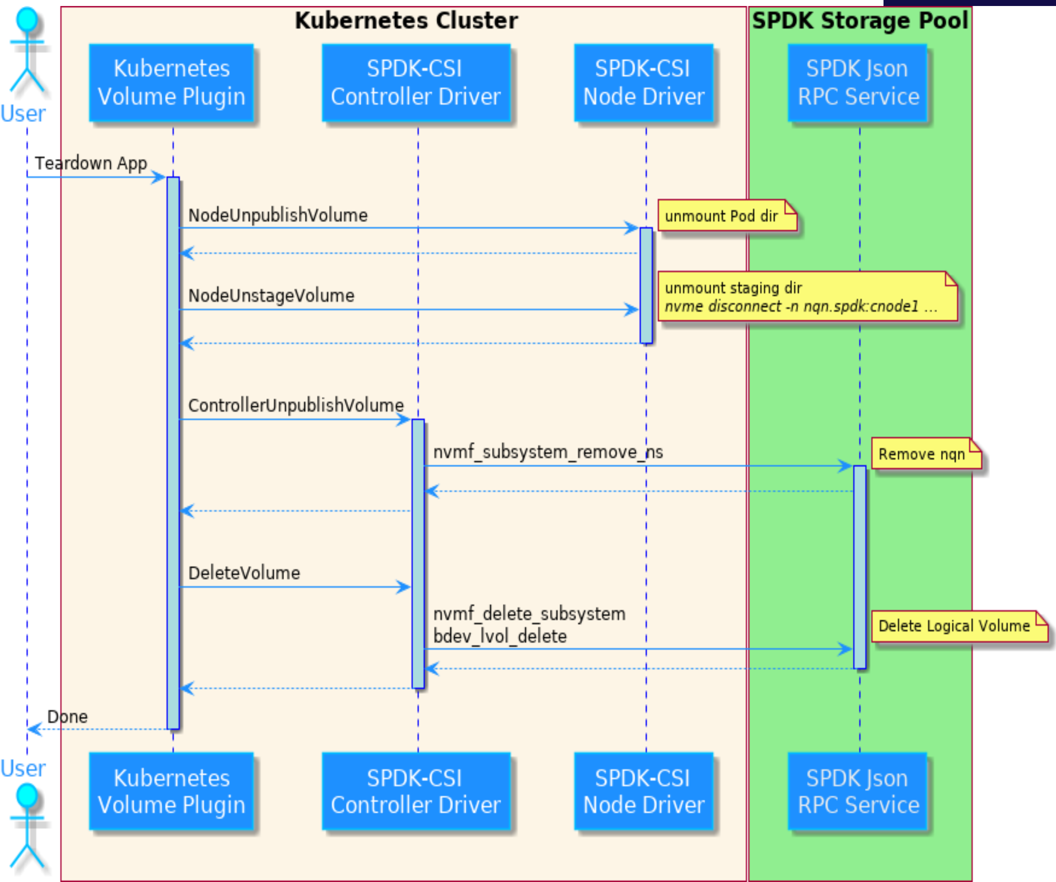
* "*nqn, ip, port, diskid, iqn*" are passed from Controller Driver

Sequence Diagram

Create and Attach Volume



Teardown Volume



SPDK-CSI Deployment

- Deploy Kubernetes cluster
  - Minikube is convenient for development purpose
- Deploy SPDK service
- Deploy SPDK-CSI components
  - Controller, Node, StorageClass, RBAC, etc.
- Validate SPDK-CSI driver with End to End testing
  - Standard way to test third party CSI drivers in Kubernetes
- Too many details, see source code ☺

Some Tips

- Idempotency and concurrency
  - Handle duplicated and out of order messages, e.g.,
    - CreateVolume may come again even if volume already created
    - UnpublishVolume may come after DeleteVolume
  - Take care of concurrency, e.g.,
    - Two DeleteVolume messages try to delete same volume at same time
- Keep it simple
  - Leverage Kubernetes declarative model in error handling
  - Suppress Controller(Un)PublishVolume messages
    - Do everything in CreateVolume/DeleteVolume handlers

Community

Welcome Contribution

- Code review at SPDK Gerrit
  - *git clone https://review.spdk.io/spdk/spdk-csi*
  - Github mirror: https://github.com/spdk/spdk-csi
- Development Guidelines
  - https://spdk.io/development/
- Trello Board
  - https://trello.com/b/nBujJzya/kubernetes-integration

Project Status and Plan

- Status: Alpha
  - Mandatory CSI functionalities are ready
- Plans:
  - Tests and improvements for production level quality
  - New features
    - Topology, volume expansion, snapshot, etc.
    - See Backlogs and Todos at Trello Board
  - Integration with Rook
    - Build a total solution of leveraging SPDK in Kubernetes

References

- Container Storage Interface (CSI) Spec
  - https://github.com/container-storage-interface/spec/
- Kubernetes CSI Documentation
  - https://kubernetes-csi.github.io/docs/
- SPDK JSON-RPC
  - https://spdk.io/doc/jsonrpc.html
- SPDK-CSI Design Document
  - https://tinyurl.com/spdkcsi-design-doc

Please take a moment to rate this session.

Your feedback matters to us.