Auto-deployment of Ceph cluster with Rook on top of k8s

Dennis Chen

Staff Software Engineer

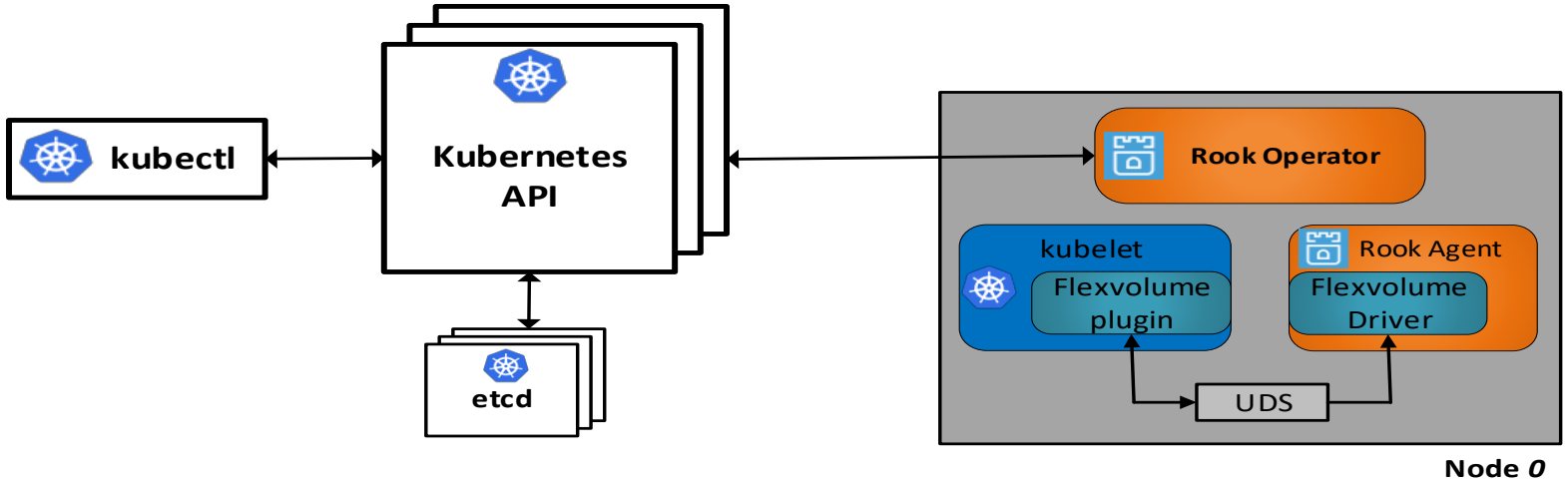
Nov 15, 2018

KubeCon+CloudNativeCon, Shanghai, 2018

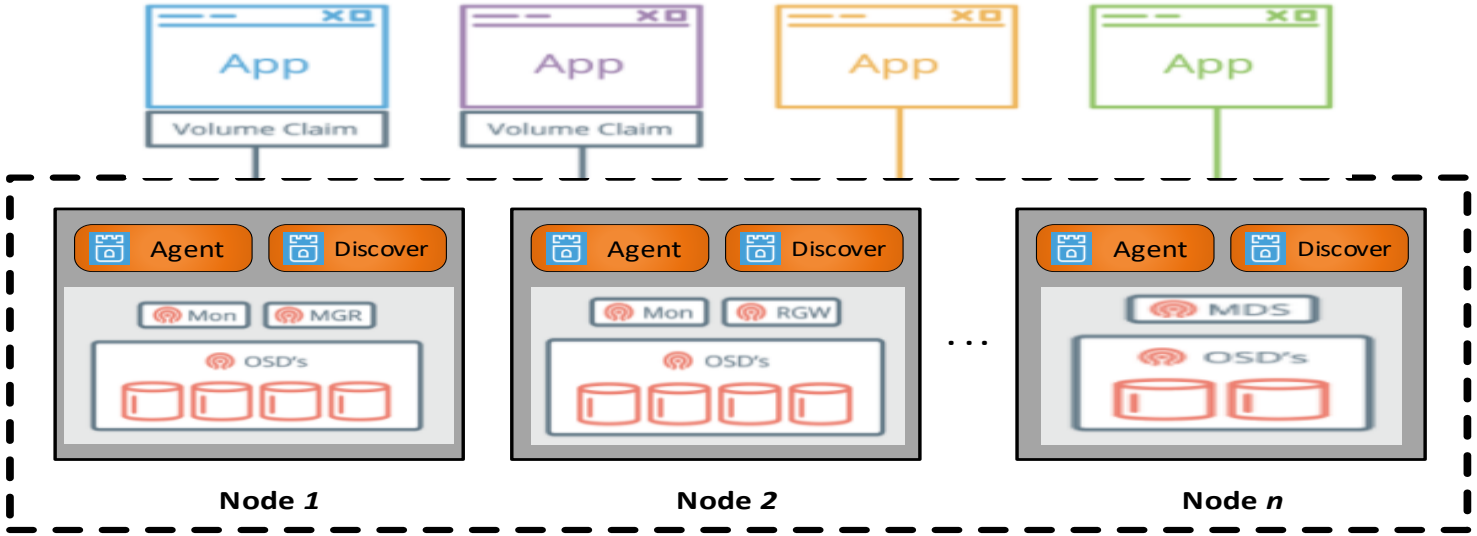
Agenda

- Overview of Design
- Operator PoD
- Agent and Discover PoD
- Ceph cluster deployment with Rook
- Flex Volume Driver -- rookflex
- Ceph RBD volume operations
- Status update on AArch64

Overview of Design



- Rook cluster is on top of k8s cluster
- 3 key elements of Rook: Operator, Agent and Discover
- Now Rook is based on out-of-tree Flexvolume machenism
- Rook doesn't change the data plane of the Ceph cluster



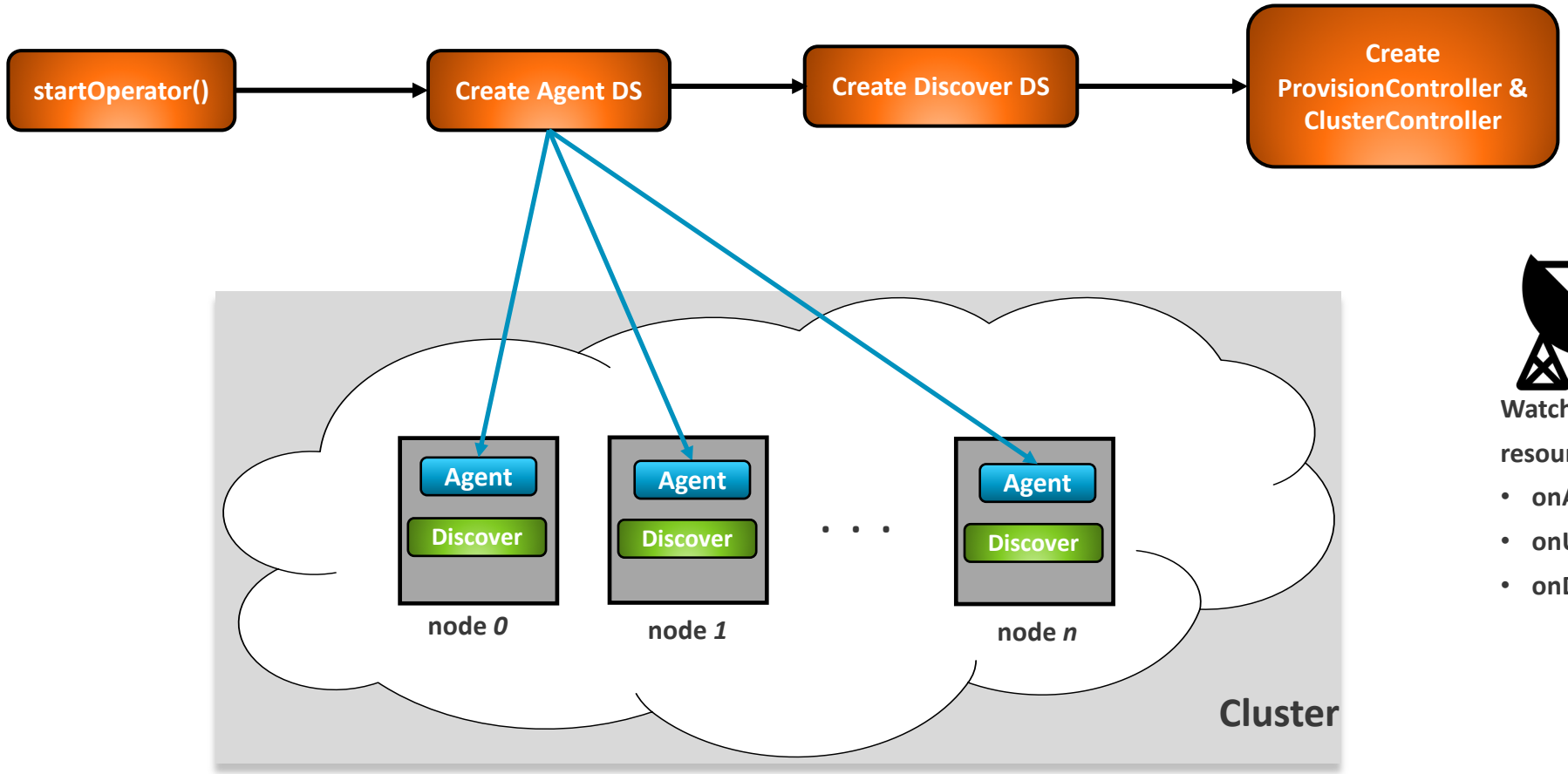
arm

Operator PoD

- A simple container functions at cluster level : bootstrap and monitor the storage cluster
- `kubectl create –f operator.yaml` to create the Rook Operator PoD.
- Rook Operator PoD is created through k8s' Deployment obj with `replicas: 1` (only one instance running in the cluster), it's mechanism of k8s not Rook's.
- The arguments to startup the operator pod container- args: ["ceph", "operator"], triggering the `/usr/local/bin/rook ceph operator` command line inside container.
- So RunE = startOperator() is invoked to bootstrap the `Agent` and `Discover` PoDs in form of DaemonSet.

Operator PoD

- startOperator() workflow



Watch the cluster's resource changes :

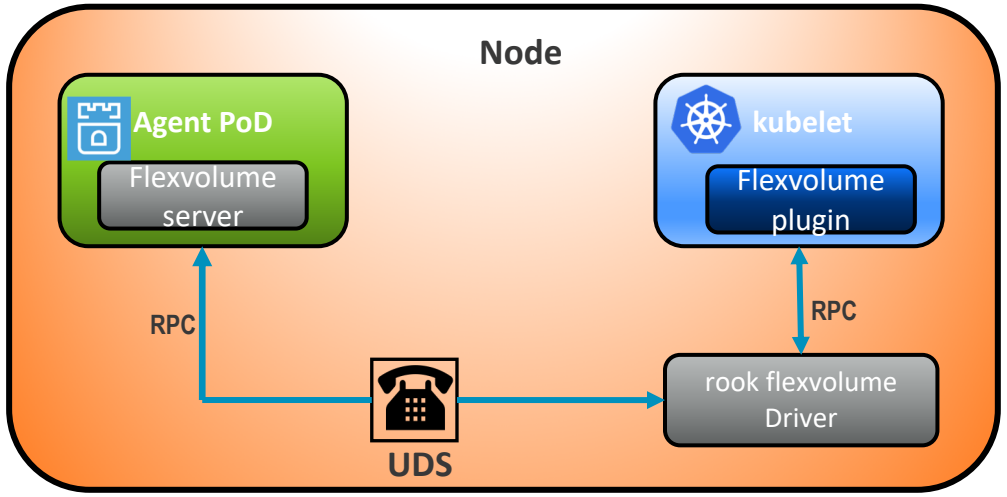
- onAdd()
- onUpdate()
- onDelete()

*Operator includes the provisioning component: ProvisionController

arm

Agent and Discover PoDs

- A mini operator that functions at the node level
- Agent PoD startups with args: “ceph agent” -- `/usr/local/bin/rook ceph agent` command line inside container, will call RunE=startAgent()
- Install the Rook Flexvolume driver to the volume-plugin-dir directory on every node



- Perform storage operations on behalf of the Flexvolume driver, such as attaching/detaching, mount/unmount via UDS
- Discover PoD is used to discover the available devices on the node periodically

arm

Ceph cluster deployment with Rook

- Rook ***clusterController*** is watching the resources changes within the cluster, so:
- `kubectl create –f cluster.yaml` will trigger the Ceph cluster bootstrap workflow.
- Rook configures the PoD spec of the `mon` and launch the PoD with `makeDeployment()`
- Launch the ceph `mgr` and `osd` PoD with similar method.
- Finally we'll get a Ceph cluster like this(on a 2-node cluster):

```
root@huawei-d05:~# kubectl -n rook-ceph get pod
NAME                                  READY     STATUS      RESTARTS   AGE
rook-ceph-mgr-a-9d4994847-rg47n       1/1       Running     0          2d
rook-ceph-mon-a-6c485ffd74-sc6t8      1/1       Running     0          2d
rook-ceph-mon-b-766ff4c757-lwp56      1/1       Running     0          2d
rook-ceph-mon-c-5b4fc4bf8f-4nf99      1/1       Running     0          2d
rook-ceph-osd-0-7b4bc6d7fd-4k4fd      1/1       Running     1          2d
rook-ceph-osd-1-779cf9c575-tslrn      1/1       Running     0          2d
```


arm

Flex Volume Driver -- rookflex

- `rookflex` exists in form of a binary file and has been deployed into volume-plugin-dir by Rook Agent on each node.
- `rookflex` implements ‘mount’ and ‘umount’ methods required by FlexVolume Spec
- For a specific YAML file of a workload, the storage related part looks like:

Storage Provisioning

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: rook-ceph-block
provisioner: ceph.rook.io/block
parameters:
  pool: replicapool
```

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
spec:
  storageClassName: rook-ceph-block
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
---
```

Storage Consuming

```
    spec:
      containers:
      - image: wordpress:4.6.1-apache
        name: wordpress
        env:
        - name: WORDPRESS_DB_HOST
          value: wordpress-mysql
        - name: WORDPRESS_DB_PASSWORD
          value: changeme
        ports:
        - containerPort: 80
          name: wordpress
        volumeMounts:
        - name: wordpress-persistent-storage
          mountPath: /var/www/html
      volumes:
      - name: wordpress-persistent-storage
        persistentVolumeClaim:
          claimName: wp-pv-claim
```


arm

Flex Volume Driver -- rookflex

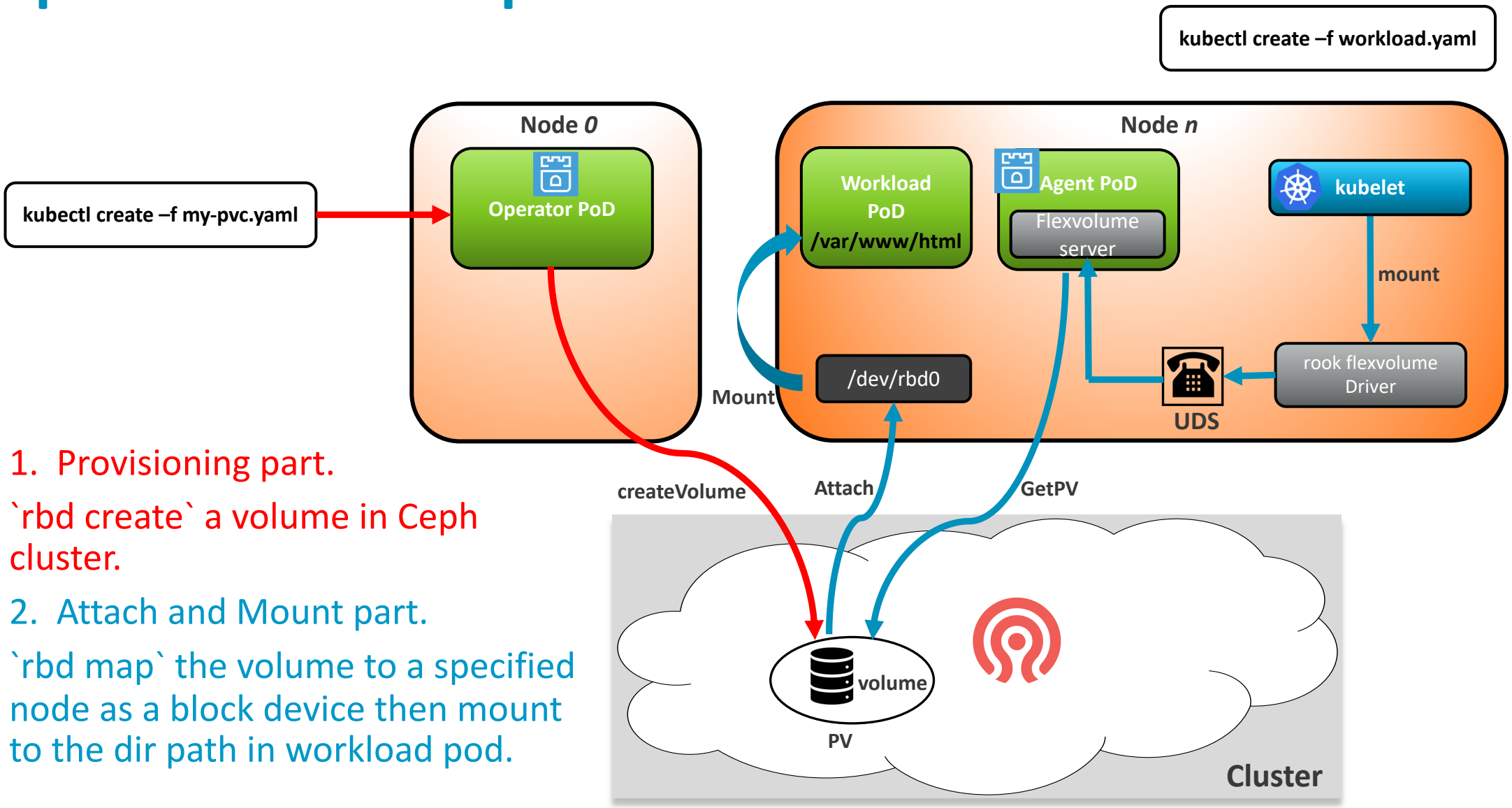
- When that workload PoD is scheduled to one node and begin to run, the kubelet will interacts with the driver to mount the volume into the `mountPath` in the YAML. To do so, kubelet needs to:

  1. Lookup the right Flexvolume driver.

     The look up flow is: PVC name → StorageClass → provisioner name: ceph.rook.io/block → Flex volume vendor name: "ceph.rook.io“ → figure out the driver folder and driver name: rookflex

  2. Call `mount` method of rookflex like: `$(volume-plugin-dir)/rookflex mount`
  3. The above `mount` will call the corresponding function in Rook Agent via UDS.
  4. Local Rook Agent will attach the volume into its node(a ‘rbd map’ operation).

Ceph RBD volume operations



1. Provisioning part.

`rbd create` a volume in Ceph cluster.

2. Attach and Mount part.

`rbd map` the volume to a specified node as a block device then mount to the dir path in workload pod.

arm

Status Update on AArch64

- Participating the Rook community (https://github.com/rook/rook) actively
- 14 patches from Arm have been merged into the upstream mainline code so far
- Rook can be built and run smoothly on AArch64 now
- CSI support in the future (csi volume driver, csi provision controller, etc)

Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!