NVMeoF Storage Volumes for Containers

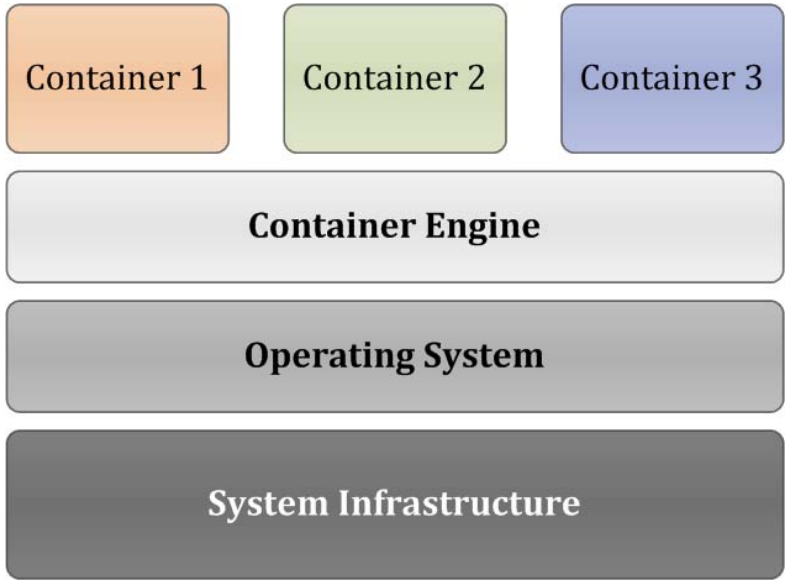
203-C

Gunna Marripudi, Principal Architect
Ming Lin, Sr. Architect

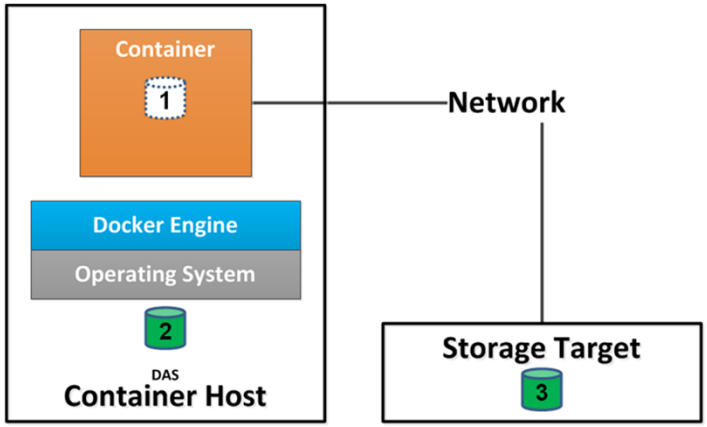
Samsung Semiconductor Inc.

Containers Overview

- Container holds everything an application needs to run
- Isolates applications from cross library dependencies
- Enables portability and scale



Stateful Containers



Storage Type	State	Comments
1. Local to container	Ephemeral	Data loss when container stops
2. Local to container host	Persistent	Loss of access to data when container host fails OR container moves to another host
3. Network Storage	Persistent	Continuous access to data

Kubernetes Persistent Volumes*

- Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.
- A PersistentVolume (PV) is a piece of networked storage in the cluster that has been provisioned by an administrator.
- It is a resource in the cluster just like a node is a cluster resource.
- PVs are volume plugins like Volumes, but have a lifecycle independent of any individual pod that uses the PV.

* *http://kubernetes.io/docs/user-guide/persistent-volumes/*

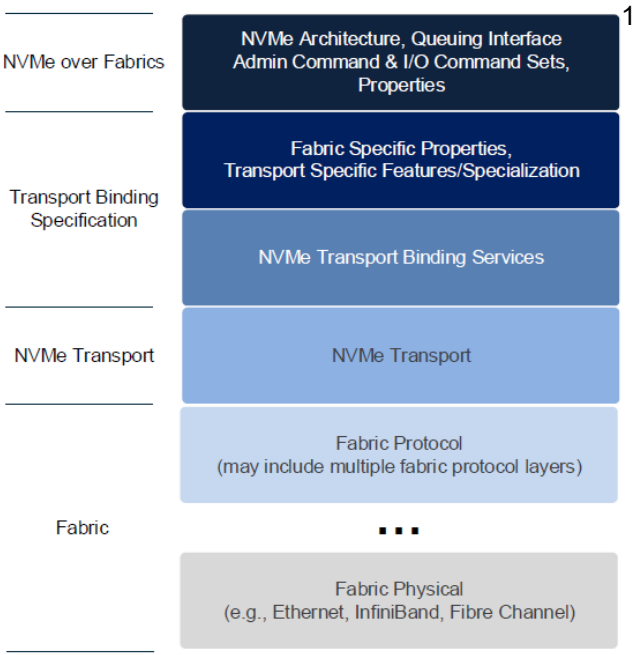
Types of Persistent Volumes*

- PersistentVolume types are implemented as plugins. Kubernetes currently supports the following plugins:
  - GCEPersistentDisk
  - AWSElasticBlockStore
  - AzureFile
  - FC (Fibre Channel)
  - NFS
  - iSCSI
  - RBD (Ceph Block Device)
  - CephFS
  - Cinder (OpenStack block storage)
  - Glusterfs
  - VsphereVolume
  - HostPath (single node testing only – local storage is not supported in any way and WILL NOT WORK in a multi-node cluster)

* *http://kubernetes.io/docs/user-guide/persistent-volumes/*

NVMe over Fabrics Overview

- Maintains consistency with base NVMe definition but for fabrics support[1]
- Support for multiple transport types
- Exposes NVMe parallelism to host
- Performance closer to local attached NVMe devices



[1] *http://www.nvmexpress.org/wp-content/uploads/NVMe_over_Fabrics_1_0_Gold_20160605.pdf*

Kubernetes PV Plugin: NVMf

- api/swagger-spec/v1.json
  - Specifies the fields to identify NVMf volume in pod's yaml file
- cmd/kubelet/app/plugins.go
  - Add nvmef as a supported volume plugin
- pkg/api/types.go
  - Specifies the fields to define NVMf volume source.
- pkg/api/validation/validation.go
  - Adds validation checks to verify NVMf volume specification in pod's yaml file.
- pkg/kubectl/describe.go
  - Format helper to print NVMf volume descriptions.

Kubernetes PV Plugin: NVMf

- pkg/volume/nvmef/nvmef.go
  - Implements Kubernetes volume plugin interfaces.
- pkg/volume/nvmef/disk_manager.go
  - Attaches and mounts the specified volume to the kubelet host.
- pkg/volume/nvmef/nvmef_util.go
  - Implements NVMe specific operations to support disk_manager operations.
- pkg/volume/nvmef/doc.go
  - Placeholder file.

NVMf Target Configuration

NVMf Subsystem Port Configuration

NVMf Subsystem NQN and Namespace Configuration

```
$ tree /sys/kernel/config/nvmet
├── hosts
├── ports
│   └── 1
│       ├── addr_adrfam
│       ├── addr_traddr
│       ├── addr_treq
│       ├── addr_trsvcid
│       ├── addr_trtype
│       ├── referrals
│       └── subsystems
│           └── nqn.testiqn -> ../../../../nvmet/subsystems/nqn.testiqn
└── subsystems
    └── nqn.testiqn
        ├── allowed_hosts
        ├── attr_allow_any_host
        └── namespaces
            └── 1
                ├── device_nguid
                ├── device_path
                └── enable
```

NVMf Target Configuration Example

- Port configuration
  - ports/1/addr_trtype:**rdma**
  - ports/1/addr_trsvcid:**1023**
  - ports/1/addr_traddr:**40.10.10.114**
  - ports/1/addr_treq:not specified
  - ports/1/addr_adrfam:ipv4

- Subsystem configuration
  - subsystems/**nqn.testiqn**/namespaces/1/enable:1
  - subsystems/nqn.testiqn/namespaces/1/device_nguid:00000000-0000-0000-0000-000000000000
  - subsystems/nqn.testiqn/namespaces/**1**/device_path:**/dev/nvme3n1**
  - subsystems/nqn.testiqn/attr_allow_any_host:1

Pod using NVMf Volume

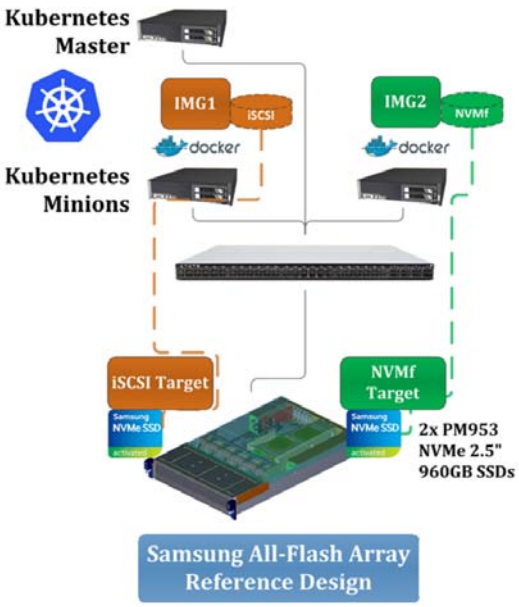
- Specifies NVMf volume name: ***nvmf-volume***
- Specifies volume type as ***nvmef***
- Identifies the NVMf volume by
  - Subsystem: ***nqn.testnqn***
  - Subsystem port: ***40.10.10.114:1023, rdma***
  - Namespace: ***1***

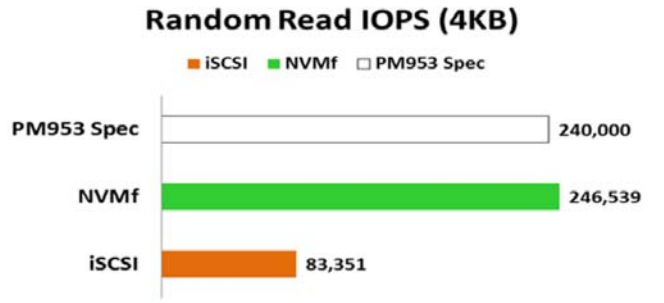
- Specifies NVMf volume mount points into container namespace

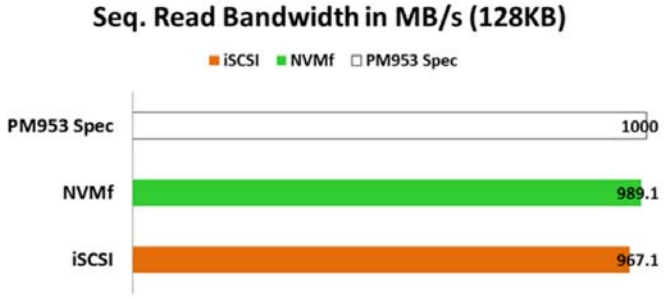
```
gpiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nvmeof-nginx
spec:
  template:
    metadata:
      labels:
        run: nvmeof-nginx
    spec:
      volumes:
      - name: nvmf-volume
        nvmef:
          # Address of the NVMeoF target portal
          targetPortal: "40.10.10.114:1023"
          # Subystem NQN of the portal
          nqn: "nqn.testiqn"
          # Namespace we want to mount
          ns: 1
          # transport
          transport: "rdma"
          # Filesystem on the LUN
          fsType: ext4
          readOnly: false

      containers:
      - name: my-nginx
        image: nginx
        ports:
        - name: web
          containerPort: 80
          protocol: TCP
        volumeMounts:
        # 'name' must match the volume name above.
        - name: nvmf-volume
          # Where to mount the volume.
          mountPath: "/usr/share/nginx/html/"
```

Kubernetes PV Performance Comparison: NVMf vs. iSCSI







PVs based on NVMf delivers native NVMe performance to Containers!

Call for Action

- NVMf plugin integration into Kubernetes
- Linux NVMe stack and CLI enhancements needed for Kubernetes PV support
- Support for additional attributes in NVMf volume specification

Thanks

References

- Kubernetes Persistent Volumes
  - http://kubernetes.io/docs/user-guide/persistent-volumes/
- Kubernetes Volume Plugins
  - https://github.com/kubernetes/kubernetes/tree/master/pkg/volume
- Samsung All-Flash Reference Design
  - http://www.samsung.com/semiconductor/support/tools-utilities/All-Flash-Array-Reference-Design/