Persistent Storage for Containerized Applications on Kubernetes with PowerMax SAN Storage

Abstract
This white paper gives an overview of storage provisioning and management for containerized applications on Kubernetes platform with PowerMax as the back-end storage. The paper also presents the Container Storage Interface (CSI) driver for PowerMax, key concepts of storage provisioning as well as some best practices of storage management for containerized applications.

October 2019
Revisions

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>October 2019</td>
<td>Initial release</td>
</tr>
</tbody>
</table>

Acknowledgements

This paper was produced by the following:

Author: TME for Data Essentials - DevOps

Support:

Other:

The information in this publication is provided “as is.” Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

Copyright © <10/14/2019> Dell Inc. or its subsidiaries. All Rights Reserved. Dell, EMC, Dell EMC and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be trademarks of their respective owners.
# Table of contents

1 Kubernetes Storage in Nutshell ......................................................................................................................................................................................................................................................................................................................5  
   1.1 StorageClass ..................................................................................................................................................................................................................................................................................................................5  
   1.2 PersistentVolume ......................................................................................................................................................................................................................................................................................6  
   1.3 PersistentVolumeClaim .................................................................................................................................................................................................................................6  
2 Container Storage Interface .......................................................................................................................................................................................................................................................................................8  
   2.1 CSI Driver for PowerMax .................................................................................................................................................................................................................................8  
3 Why using SAN storage with Kubernetes .........................................................................................................................................................................................................................................9  
4 Architecture .........................................................................................................................................................................................................................................................................................10  
5 Quotas ......................................................................................................................................................................................................................................................................................11  
   5.1 Quotas per Persistent Volume ........................................................................................................................................................................................................................................11  
   5.2 Quotas per Storage Class .................................................................................................................................................................................................................................12  
   5.3 Quota per namespace .........................................................................................................................................................................................................................13  
A Support .........................................................................................................................................................................................................................................................................15  
   A.1 Main links .........................................................................................................................................................................................................................................................15  
   A.2 Tools .........................................................................................................................................................................................................................................................15
Executive summary

As applications are increasingly deployed as containers, infrastructure owners are being asked to effectively manage these new workloads and enable modern DevOps workflows. The recent introduction of Container Storage Interface (CSI) driver standard brought much needed standardization for both the storage vendors, to develop plugins for their respective storage platforms, and the infrastructure and DevOps leads to easily provision and manage storage for containerized applications.

In this context, the role of storage administrators as key enablers of operational agility is more relevant than ever. In this whitepaper we will show some of the common storage workflows in a Kubernetes setup with Dell PowerMax storage platform. This document also provides a good overview of the concepts, architecture and best practices to implement and use the CSI Driver for the PowerMax array as persistent storage within a Kubernetes cluster.
1 Kubernetes Storage in Nutshell

This section describes the main concepts to know when dealing with related to persistent storage in Kubernetes.

1.1 StorageClass

A StorageClass defines the type of storage that the infrastructure offers to the users. Each class has different characteristics which we can associate to different service levels and data services provided by the storage platform.

The CSI Driver for PowerMax comes with two Storage Classes. The default powermax class creates volumes and formats them in ext4 and powermax-xfs for xfs volumes.

For example, we can get the details of the Storage Class with: `kubectl get storageclass powermax-xfs -o yaml`

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  creationTimestamp: "2019-09-04T14:49:25Z"
  name: powermax-xfs
parameters:
  FsType: xfs
  SRP: SRP_1
  SYMID: "000000000000"
  ServiceLevel: Bronze
provisioner: csi-powermax
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

The key attributes that can be modified to be tuned are:

- **parameters**: sets the driver parameters (file system type and attached storage pool)
- **reclaimPolicy**: to configure the whether the volumes will be retained or deleted when the assigned pod is destroyed

For more details on how to configure a new StorageClass for PowerMax see [Quotas per Storage Class](#).
1.2 PersistentVolume

A **PersistentVolume** (PV) is a logical volume that is provisioned to containerized applications that require persistent storage. A PV usually corresponds to a single volume on the physical storage platform.

A PersistentVolume can be **statically provisioned** by the storage administrator or **dynamically provisioned** by the application owner in a PersistentVolumeClaim.

It is the role of the CSI Driver for PowerMax, to create, format and bind the volume.

```
kubectl get pv
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
STORAGECLASS   REASON   AGE
k8s-3401e3645a   8Gi        RWO          Delete           Bound
default/pgdata-pgset-2 powermax 36d
k8s-792f388c66   8Gi        RWO          Delete           Bound
test/pvol0       powermax 21d
test/pvol1       powermax-xfs 21d
k8s-c71377055a   16Gi       RWO         Delete           Bound
default/pgdata-pgset-0 powermax 36d
k8s-c9c63c835a   8Gi        RWO          Delete           Bound
default/pgdata-pgset-1 powermax 36d
```

1.3 PersistentVolumeClaim

A **PersistentVolumeClaim** (PVC) is a request to provision storage. A PersistentVolume with the right size and StorageClass is assigned to fulfil the request.

Here is an example of a PVC definition we can get with **kubectl get pvc pgdata-pgset-0 -o yaml**

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pgdata-pgset-0
spec:
  storageClassName: powermax
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```
To obtain the mapped PV to a PVC you can run `kubectl get pvc`:

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>VOLUME</th>
<th>CAPACITY</th>
<th>ACCESS MODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>pgdata-pgset-0</td>
<td>Bound</td>
<td>k8s-demo-c71377055a</td>
<td>8Gi</td>
<td>RWO</td>
</tr>
<tr>
<td>pgdata-pgset-1</td>
<td>Bound</td>
<td>k8s-demo-c9c63c835a</td>
<td>8Gi</td>
<td>RWO</td>
</tr>
<tr>
<td>pgdata-pgset-2</td>
<td>Bound</td>
<td>k8s-demo-3401e3645a</td>
<td>8Gi</td>
<td>RWO</td>
</tr>
</tbody>
</table>

The key attributes that can be modified are:

- **accessModes**: [Access Modes](#) to set how the volume will be read or written per the nodes
- **resources.requests.storage**: sets the capacity claimed by the PVC

**Note**: ReadWriteMany access mode should not be used with the PowerMax Driver because neither ext4 nor xfs are a clustered file system.
2 Container Storage Interface
With the Kubernetes 1.13 release, the Container Storage Interface (CSI), has been promoted to GA status. The CSI specification standardizes the persistent storage management (Block or File) within a container orchestrator.

The advantage of the CSI specification over “in-tree” driver is the driver development is now independent of Kubernetes releases. It gives more flexibility and agility to driver roadmap.

2.1 CSI Driver for PowerMax
The CSI Driver for PowerMax offers native integration between a Kubernetes cluster with PowerMax backend storage for:

- Static & Dynamic Persistent Volume Provisioning
- Delete & Retain Persistent Volume Reclaiming
3 Why use SAN storage with Kubernetes

The [CSI Driver for PowerMax](#) and other drivers offer a native integration between [Kubernetes](#) and the backend storage.

All the tasks related to the storage provisioning of a [PersistentVolume](#) and backend block device are delegated to [PersistentVolumeClaim](#) from the POD definition.

Moreover, by using or reusing SAN storage you can leverage your investment in material and processes right away. It is possible to create a portfolio of [StorageClasses](#) with the service level you defined (type of storage, type of replication, type of backup, etc.).

The SAN allows organic consumption of the storage within the Kubernetes just like you do with VMware infrastructure and other compute platforms.

With a shared storage solution like SAN you can ensure that the persistent volume will be accessible from any node of the infrastructure. This allows for the workload mobility across the Kubernetes cluster.

SAN and shared storage offer rich data services for data protection like LUN level snapshots, remote replication and disaster recovery integration with industry standard tools like VMware Site Recovery Manager. When migrating legacy application to a container based deployment, you can ensure continuity of these services while modernizing your applications.

Even for cloud native applications with modern data store platforms like [Elasticsearch](#), [Cassandra](#) and [Couchbase](#), using SAN infrastructure can cuts down on the compute resources required for replication and take advantage of the mature data services of platforms like Dell EMC PowerMax. The SAN advantage becomes more obvious as the size of the datastores grows.
4 Architecture

CSI Drivers are application that implements gRPC services as described in the CSI specification.

Deployment of CSI is done through Sidecars containers:

- The provisioner which watches the Kubernetes API server for PersistentVolumeClaim objects. If a claim appears the registrar will create the volume, if a claim disappear the registrar will remove the volume.
- The attacher which watches the Kubernetes API server for VolumeAttachment objects and then Publish/Unpublish the volumes
- The registrar (in this case node-registrar) to register the driver to each node.
5 Quotas

With shared infrastructure like a Kubernetes cluster, quotas are a useful way to make sure workloads have the appropriate extent of shared resources.

Kubernetes offers several mechanisms to limit storage usage. These mechanisms are defined and enforced at different layers of the infrastructure (POD, definition, storage class, namespace).

5.1 Quotas per Persistent Volume

To paraphrase Kubernetes official documentation, PVs are resources in the cluster. PVCs are requests for those resources and also act as claim checks to the resource.

This means that within the pod definition the PersistentVolumeClaim is just a request and it will be fulfilled only if there is an available PersistentVolume that satisfies the requirement.

LimitRange sets at the namespace level it is possible to enforce a minimum and maximum request size. It prevents a pod to bloat all the storage resources an impact the future claims.

```yaml
cat <<EOF > limit-pvc.yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: storagelimits
spec:
  limits:
    - type: PersistentVolumeClaim
      max:
        storage: 100Gi
      min:
        storage: 8Gi
EOF

kubectl create -f ./limit-pvc.yaml --namespace=test
```
5.2 Quotas per Storage Class

One easy way to manage and segregate storage consumption within the cluster is to create different storage pools in the array and map these pools to different storage class.

Indeed, with PowerMax CSI driver, each pool can be associated with a different storage class.

```yaml
cat <<EOF > storage-class-my-other-pool.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
    name: powermax-my-other-pool
annotations:
    storageclass.beta.kubernetes.io/is-default-class: "false"
provisioner: csi-powermax
reclaimPolicy: Delete
parameters:
    storagepool: my-other-pool
EOF

kubectl create -f ./storage-class-my-other-pool.yaml
```

The CSI driver itself doesn’t give visibility into subscription rates and utilization rates. Therefore, it is important for the PowerMax storage administrator to have proper monitoring in place for the Storage Pool.
5.3 Quota per namespace

Kubernetes allows defining resource quotas to limit the usage of compute and storage resources of the cluster. The quotas are given per namespace. The storage resource quotas can limit the amount of storage to be used and the number of persistent volume claim per namespace.

It is important to define a ResourceQuota attached to a StorageClass because on the rate PersistentVolume can be created.

It is also possible to scope the quota to a particular storage class like in the example below:

```
cat <<EOF > storage-resources.yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: storage-resources
spec:
  hard:
    powermax.storageclass.storage.k8s.io/requests.storage: "500Gi"
    powermax.storageclass.storage.k8s.io/persistentvolumeclaims: 10000
EOF
```

```
kubectl create -f ./storage-resources.yaml --namespace=test
```

Check the quotas definitions for a namespace with:

```
kubectl describe quota storage-resources --namespace=test
```

<table>
<thead>
<tr>
<th>Resource</th>
<th>Used</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>powermax.storageclass.storage.k8s.io/persistentvolumeclaims</td>
<td>2</td>
<td>10k</td>
</tr>
<tr>
<td>powermax.storageclass.storage.k8s.io/requests.storage</td>
<td>16Gi</td>
<td>500Gi</td>
</tr>
</tbody>
</table>
6 Conclusion

The Kubernetes on-premise infrastructures must have a shared storage solution to run stateful workloads. SAN and NAS storage are more than ever a vital part of these infrastructures. In addition to the data mobility, these solutions can bring data services that cloud-hosted solution might not be available to deliver to Kubernetes, such as features like deduplication, high IOPS and bandwidth volumes, and more.
Support

A  Support

A.1  Main links

- [https://github.com/dell/csi-powermax](https://github.com/dell/csi-powermax): Homepage for the PowerMax CSI Driver
- [https://hub.docker.com/r/dellemc/csi-powermax](https://hub.docker.com/r/dellemc/csi-powermax): Container image for the Driver
- [https://www.dell.com/community/Containers/bd-p/Containers](https://www.dell.com/community/Containers/bd-p/Containers): Dell EMC community for the driver and more
- [https://kubernetes.io/blog/2019/01/15/container-storage-interface-ga/](https://kubernetes.io/blog/2019/01/15/container-storage-interface-ga/): CSI GA announcement, has interesting configuration
- [https://www.dell.com/community/Containers/FAQ-CSI-Drive](https://www.dell.com/community/Containers/FAQ-CSI-Drive): FAQ on PowerMax CSI Driver

A.2  Tools

- [https://github.com/coulof/vxflexos-csi-examples](https://github.com/coulof/vxflexos-csi-examples): Videos and recipes of CSI Driver usage
- [https://github.com/coulof/kubespray](https://github.com/coulof/kubespray): Deploy a Production Ready Kubernetes Cluster with CSI and PowerMax provisioner
- [https://kubernetes-csi.github.io/docs/troubleshooting.html](https://kubernetes-csi.github.io/docs/troubleshooting.html): debug CSI driver issues