Integrating Kubernetes Persistent Volumes into a Composable Infrastructure Platform

Brian Pawlowski
Jean-François Remy
DriveScale Inc.
The DriveScale Composable Infrastructure platform works seamlessly with Kubernetes to provide performant dynamic volumes allowing you to bring data intensive scale-out applications under this emerging data center orchestration standard without compromise.
Agenda

- Kubernetes
- Container Storage Interface (CSI)
- Scale-out Applications
- DriveScale Composable Infrastructure
- DriveScale CSI Plug-in for Kubernetes
- Future
- Questions

[Terminology] and [References]
Technology Trends
Three Technology Trends

- Commodity Virtualization
  - 2006

- Scale-out Applications
  - 2008

- Containers
  - 2013

- Post-VM Infrastructure
Kubernetes
Journey to Containers

Scale-up

Traditional Deployment

Operating System

Hardware

Virtualized Deployment

App

Operating System

Hypervisor

Operating System

Hardware

Container Deployment

App

App

App

Bin/ Library

Bin/ Library

Virtual Machine

Virtual Machine

From https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
Containers

- An operating-system level virtualization method to run multiple isolated lightweight (Linux) systems (containers) on a single physical host
- Containers are lightweight
  - MBs for a containers vs. GBs for VMs
  - Only one copy of the host operating system
  - Common binaries/libraries can be shared by multiple applications
- Containers are highly scalable
- Containers are both hardware-agnostic and platform-agnostic.
  - They can run on your laptop or on a bare metal platform or an EC2 instance in exactly the same way
- Containers can simplify scale-out application deployment – but don’t provide the means to manage at scale
Kubernetes

- An open-source container-orchestration platform for automating deployment, scaling and management of containerized applications.
  - Started at Google, now maintained by Cloud Native Computing Foundation
  - The name Kubernetes originates from Greek, meaning helmsman or pilot. (It is also the root of cybernetics)
- Manage clusters of hosts (reminds one of scale-out apps)
  - Deploy, maintain, and scale applications based on CPU, memory
  - Provides grouping, load balancing, auto-healing & scaling features
  - The basic scheduling unit in Kubernetes is a pod – which co-locates containers on a host machine to share resources
Container Storage Interface (CSI)

- Consistent, orchestrator-independent volume management API
  - CSI is the preferred volume storage provider API in Kubernetes
    - Enables a wide variety of storage plug-ins
  - Enables Kubernetes to flexibly support apps requiring persistent storage.
- Supports dynamic provisioning and deprovisioning of a volume.
- Exists outside of containers (Docker)
  - Attaching/detaching a volume from a node.
  - Mounting/unmounting a volume from a node.
- Kubernetes is evolving as the de facto orchestration for all applications
  - Containers originally only supported stateless applications, in part because of lack of *volume management*.
  - CSI allows enterprises to use a single framework to manage stateless and stateful applications
Scale-Out Applications
### Evolution of Workloads

#### Virtualized Workloads

Many apps managed on each server with shared enterprise storage

- Solves the unused CPU problem by providing secure sandboxes for each application
- Each VM contains an entire copy of the OS – as if an application library
- First real standard software orchestration to application deployment

#### Horizontal Scale-out Data Intensive Workloads

Apps running on clusters of commodity servers with local storage

- Commodity platforms lowers cost
- Local storage configuration and management (compression/replication) done by application
- VMs and NAS introduce I/O performance bottlenecks and cost
- Traditional SAN does not scale, adds cost
Scale-out Applications

- Applications embed high availability and resiliency
  - Usually triple replication
  - Recovery - rebuild
- Performance achieved with local Direct-Attached Storage (DAS)
  - Bring compute to storage
  - Scale compute/storage together

- **Kubernetes does not change the basic requirements and challenges to scale-out applications**
  - But it does promise better management at scale
DriveScale in a Few Slides
Static data center architecture

Horizontal Scale-out Data Intensive Workloads

Apps running on clusters of commodity servers with local storage

- Hadoop
- Cassandra
- Spark
- Aerospike
- Kubernetes

Server with local storage

- Tightly coupled, fixed resources
- Commonly overprovisioned
- Stranded compute and storage resources
- Server SKU sprawl
- Lifecycles tied together
Composable Infrastructure

Captive, fixed DAS

Disaggregate

Composable

Purchase Time Defined Infrastructure

Right sized

Software Defined Infrastructure

2019 Storage Developer Conference. © DriveScale Inc. All Rights Reserved.
The DriveScale Composable Platform

DriveScale Composer

- Policy
- Orchestration
- Monitoring

- Automated creation of dynamic servers
- Automated end-to-end set up for NVMe/TCP, NVMe/RDMA or iSCSI
- Patented load balancing
- Vendor mix-and-match

ToR Ethernet switches
10G, 25G, 100G, 400G

Diskless servers (boot drive)
DriveScale Server Agent

eBODs, JBODs (Flash, HDD)*
DriveScale Adapter Software

Each resource treated as independent from its enclosure

* eBOD – ethernet-attached Bunch Of Drives
Transformative economics of DriveScale Composable Infrastructure

- 50% lower cost than legacy and cloud
- 69% less data center footprint
- 44% savings for upgrades
- 2 minutes to deploy infrastructure
- No more overprovisioning
DriveScale CSI Driver
Kubernetes on DriveScale

DriveScale Domain

Kubernetes on DriveScale

DriveScale Domain

Kubernetes

Inventory
Kubernetes on DriveScale

- DriveScale Composer Platform
  - RESTful API
  - DriveScale Engine (mgmt_pyro_server)
  - ZooKeeper
  - MongoDB

- DriveScale Domain

- Kubernetes
  - Kubernetes Master
    - etcd
    - API Server
    - Scheduler
    - Controller Manager

- Inventory

- etcd API Server
- Scheduler
- Kubernetes Master
- Controller Manager
- Kubernetes Master
- Controller Manager
- Kubernetes Master
- Controller Manager

2019 Storage Developer Conference. © DriveScale Inc. All Rights Reserved.
Kubernetes on DriveScale

- A Kubernetes cluster maps to a single DriveScale cluster
  - Kubernetes is the software/container orchestrator
  - DriveScale is the hardware/physical orchestrator
  - DriveScale server agent runs on all nodes
- Multiple Kubernetes clusters can run in a DriveScale domain (data center)
  - DriveScale clusters for Kubernetes are created dynamically
  - Kubernetes clusters can exist outside of DriveScale
- Kubernetes and DriveScale both approach configuration management from a desired state approach.
  - etcd and zookeeper (respectively) are the distributed key-value stores.
  - Highly available, durable and consistent in face of controller node failures and network partitions
Configuration Link to Kubernetes

drivescale-secret.yaml:

```yaml
apiVersion: v1
type: Opaque
data:
  dmsUser: <base64 encoded DriveScale admin username>
  dmsPassword: <base64 encoded DriveScale admin password>
  dmsServer: <base64 encoded DriveScale Management server>
  dmsClusterName: <base64 encoded cluster name to create - if not set, "CSI_Cluster" will be used>
```

• Plus the (user provided) storage class referencing provisioner csi.drivescale.com
Dynamic PV Lifecycle

CreateVolume

Controller
PublishVolume

Node
Stage Volume

Node
Publish Volume

DeleteVolume

Controller
UnpublishVolume

Node
Unstage Volume

Node
Unpublish Volume
Creating a DriveScale Volume

<table>
<thead>
<tr>
<th>Kubernetes CSI</th>
<th>DriveScale API*</th>
<th>Result</th>
</tr>
</thead>
</table>
| CreateVolume   | \[Create Logical Cluster\]  
Add Logical Node  
Add Drive(s)  
Map Drive(s) to Logical Node | 1 Logical Cluster ↔ 1 Kubernetes cluster  
1 Logical Node ↔ 1 PV |
| Controller PublishVolume | Add Server  
Map Server to Logical Node  
Configure RAID and encryption  
[create filesystem] | CSI driver mount operation of block device to node |
| Node Stage Volume | | CSI driver bind mounts the node directory to the pod/container |
| Node Publish Volume | | |

* Items in brackets ‘[ ]’ done on first PV in cluster or first reference of a PV for existing cluster.
DriveScale Drive Types (PVs)

Single

Redundant (RAID 1 mirror)

Striped* (RAID 10)

* If a request is greater than the size of an SSD or HDD, whole drives are allocated in a RAID 10 stripe. SSD slices are dynamically allocated as a right-sized portion of an SSD.
SSD Slicing

- To support Kubernetes, the DriveScale Composer was extended to automatically create SSD Slices on the fly.
- SSD can be carved into 1 GB-aligned chunks to serve multiple clients with right-sized allocations
  - The high performance (both IOPS and throughput) of SSDs make this feasible
  - Cost effective SSD sizes are increasingly too large for particular applications
- Slicing is never enabled for HDDs (sequential performance collapse)
Persistent Data Secure by design

✓ All filesystem and data in-flight from the server, and at-rest on the drives are encrypted
✓ Fully automated encryption: scales seamlessly, keys follow drive within security domain
✓ DriveScale Composer and Server Agents automatically create a Drive Encryption Key derived from a customer-supplied secret
✓ Key exchange between DriveScale Composer and Server Agents performed over a secure Key Distribution Channel
✓ DriveScale Server Agent plumbs Linux dm-crypt to deliver data encryption
# DriveScale CSI key options

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>type (string)</strong></td>
<td>hdd / ssd / slice - <strong>default hdd</strong>: type of volume (hard drive, ssd drive, slice of ssd drive)</td>
<td>Type of volume (hard drive, ssd drive, slice of ssd drive)</td>
</tr>
<tr>
<td><strong>rpm (int)</strong></td>
<td>minimum RPM value of drives used in the cluster (ignored for ssd / slice)</td>
<td>Minimum RPM value of drives used in the cluster (ignored for ssd / slice)</td>
</tr>
<tr>
<td><strong>fsType (string)</strong></td>
<td>ext4 / xfs - <strong>default ext4</strong>: the filesystem type to use on the volume</td>
<td>The filesystem type to use on the volume</td>
</tr>
<tr>
<td><strong>striping (bool)</strong></td>
<td><strong>default true</strong>: if the plugin cannot fit the volume size on one drive, it will try create a RAID10 array that can accommodate the total size requested. This is not supported for slices.</td>
<td>Whether to use striping (RAID10)</td>
</tr>
<tr>
<td><strong>redundancy (bool)</strong></td>
<td><strong>default false</strong>: the plugin will create the volume as a RAID1 (mirror) array to ensure the volume can survive a drive failure</td>
<td>Whether to enable redundancy (RAID1)</td>
</tr>
<tr>
<td><strong>raidMaxDrives (int)</strong></td>
<td><strong>default 16</strong>: how many drives to use in a RAID array at most</td>
<td>Maximum number of drives to use in a RAID array</td>
</tr>
<tr>
<td><strong>sgResiliency (bool)</strong></td>
<td><strong>default true</strong>: should the plugin ensure that RAID volumes will survive a storage group failure (volume creation will fail if not possible). If false, the plugin will still try to provide storage group failure resiliency if it can</td>
<td>Whether to enable storage group resiliency</td>
</tr>
<tr>
<td><strong>jbodResiliency (bool)</strong></td>
<td><strong>default true</strong>: should the plugin ensure that RAID volumes will survive a JBOD failure (volume creation will fail if not possible). If false, the plugin will still try to provide JBOD failure resiliency if it can</td>
<td>Whether to enable JBOD resiliency</td>
</tr>
</tbody>
</table>
## DriveScale CSI add’l options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>encrypt (bool)</td>
<td><strong>default false:</strong> should the volume be encrypted</td>
</tr>
<tr>
<td>maxVolumeSize (int)</td>
<td>the maximum size allowed for a volume</td>
</tr>
<tr>
<td>minVolumeSize (int)</td>
<td>the minimum size allowed for a volume</td>
</tr>
<tr>
<td>softDomains (bool)</td>
<td><strong>default false:</strong> should drives meet the bandwidth domains requirements</td>
</tr>
<tr>
<td>requiredTags (string)</td>
<td>comma separated list of all the tags that the drives / slices used in the volume must have set</td>
</tr>
<tr>
<td>excludedTags (string)</td>
<td>comma separated list of any tag that would exclude a drive from being used in the volume</td>
</tr>
<tr>
<td>storageGroup (string)</td>
<td>comma separated list of storage groups to choose the drives from</td>
</tr>
<tr>
<td>networkTransports Allowed (string)</td>
<td>comma separated list of network transports allowed for the volume (allowed values are <strong>iscsi, nvmetcp, roce</strong>). If left empty, the system will use the most performant transport available.</td>
</tr>
</tbody>
</table>
Generic CSI Reclaim Policy

- Default CSI behavior is to destroy the storage on delete of the PVC
  - Can be overridden to retain on PVC delete
Podwatcher (pw)

- Single podwatcher instance in Kubernetes cluster
  - podwatcher uses Kubernetes API to listen for pod changes
  - Annotates the DriveScale Composer with
    - pod → server/node mappings
    - PVC/Logical Node → pod bindings
- Else Logical Nodes are attached to a server with no idea why!
GUI display of podwatcher info
DriveScale CSI Driver Recap

- DriveScale CSI driver available (see resources at end of talk). It is GA.
- Scalable Persistent Storage for Containers
  - Up to 10,000 compute and 100,000 drives currently
- Shared Nothing – focus on scale-out apps
  - Deliver (local) native performance of the drives to containers
  - Equivalent to applications running in Bare Metal servers
- Data locality for Containers
  - I/O scalability/performance
  - Failure domain optimization
- Logical connection between drives and containers
  - Container mobility critical to deliver efficiency gains – avoid copies or rebuilds
  - Transparent to applications running in containers
Future Work
Future Work

▪ Automatic move of pod/PVC after node HW failure
▪ podwatcher enhancements to provide more useful descriptions of PVCs (besides their UID)
▪ Additional RAID types (besides 1 and 10) as needed
▪ Volume expansion for SSD slices possible
▪ Block volumes will be supported in the future.
Support Failure/Performance Domains

- Kubernetes weakness: requires manual specification of failure and performance domains
  - DriveScale currently automatically spreads drive allocations across failure domains
  - DriveScale also automatically determines bandwidth domains
- Want to extend these capabilities to Kubernetes in future
Questions?
Thank you

Feel free to email me!
beepy@drivescale.com
Backup
# History of Kubernetes

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old Days</td>
<td>Google Borg, etc.</td>
</tr>
<tr>
<td>2008</td>
<td>cgroups introduced to mainstream Linux</td>
</tr>
<tr>
<td>2013</td>
<td>Docker first released</td>
</tr>
<tr>
<td>2014</td>
<td>Google open-sources Kubernetes, world rejoices</td>
</tr>
<tr>
<td>2015</td>
<td>• Kubernetes v1.0 released July 21, 2015</td>
</tr>
<tr>
<td></td>
<td>• CNCF launches, Google’s managed Kubernetes GKE opn K8s to a wider community</td>
</tr>
<tr>
<td></td>
<td>• Red Hat’s OpenShift launches</td>
</tr>
<tr>
<td>2016</td>
<td>• Focus moves to other clouds or on prem to adopt Kubernetes</td>
</tr>
<tr>
<td>2017</td>
<td>• FlexVolumes released</td>
</tr>
<tr>
<td></td>
<td>• Managed Kubernetes begins to appear on AWS and Azure</td>
</tr>
<tr>
<td></td>
<td>• Docker and Mesosphere announce support for Kubernetes</td>
</tr>
<tr>
<td></td>
<td>• Kubernetes becomes the de facto standard</td>
</tr>
<tr>
<td></td>
<td>• Production deployments at scale begin</td>
</tr>
<tr>
<td>2018</td>
<td>• CSI GA</td>
</tr>
<tr>
<td></td>
<td>• Additional managed services (DigitalOcean, Oracle)</td>
</tr>
<tr>
<td></td>
<td>• Investment shifts to lifecycle management in Kubernetes</td>
</tr>
<tr>
<td>2019</td>
<td>Kubernetes V1.16 released</td>
</tr>
</tbody>
</table>
## Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Similar to a VM, a container has its own filesystem, CPU, memory, process space, but shares the operating system on a node and therefore is considered much lighter weight.</td>
</tr>
<tr>
<td>Container Orchestrator (CO)</td>
<td>Automate the provisioning of containerized infrastructure and provide load balancing for the services that containers are used to create.</td>
</tr>
<tr>
<td>Container Storage Interface (CSI)</td>
<td>The Container Storage Interface provides a standard interface for any Container Orchestration systems (like Kubernetes) to expose arbitrary storage systems to their container workloads via Out-of-tree plug-ins.</td>
</tr>
<tr>
<td>Volume</td>
<td>A unit of storage made available inside of a CO-managed container, via the CSI. “Volume” or Persistent.</td>
</tr>
<tr>
<td>Persistent Volume (PV)</td>
<td>Storage that persists beyond the lifetime of a container.</td>
</tr>
<tr>
<td>Persistent Volume Claim (PVC)</td>
<td>A PVC abstracts the storage request from a pod to allow dynamic volumes.</td>
</tr>
<tr>
<td>Block Volume</td>
<td>A volume that will appear as a block device inside the container.</td>
</tr>
<tr>
<td>Mount Volume</td>
<td>A volume that will be mounted as a file system and appears as a directory inside the container.</td>
</tr>
<tr>
<td>FlexVolume</td>
<td>Earlier Kubernetes-specific volume API preceding CSI. (Out-of-tree) &lt;well known location, etc.&gt;</td>
</tr>
<tr>
<td>Out-of-tree</td>
<td>A plug-in, such as a CSI driver, that ships separately from the Kubernetes distribution. Versus built in storage drivers (in-tree)</td>
</tr>
<tr>
<td>SP</td>
<td>Storage Provider, the vendor of a CSI plugin implementation.</td>
</tr>
<tr>
<td>Drive</td>
<td>Any of HDD, SSD, or SSD slice</td>
</tr>
<tr>
<td>HDD</td>
<td>Hard Disk Drive (spinning disk)</td>
</tr>
<tr>
<td>SSD</td>
<td>Solid State Drive/Device</td>
</tr>
<tr>
<td>SSD slice</td>
<td>A right-sized virtual slice of an SSD</td>
</tr>
<tr>
<td>RAID</td>
<td>Redundant Array of Independent Drives (RAID 0 – striped, RAID 1 – mirror, RAID 5 and 6 – parity)</td>
</tr>
<tr>
<td>Pod</td>
<td>Smallest deployable unit of computing created and managed by Kubernetes. One or more containers (such as Docker containers) exist in a pod, share an IP address, have a Kubelet agent.</td>
</tr>
<tr>
<td>Kubelet</td>
<td>An agent that runs on each node in the cluster. It makes sure that containers are running in a pod</td>
</tr>
<tr>
<td>gRPC</td>
<td><a href="https://developers.google.com/protocol-buffers/">Google Remote Procedure Call</a></td>
</tr>
<tr>
<td>Node</td>
<td>A host where the user workload will be running, uniquely identifiable from the perspective of a Plugin by a node ID.</td>
</tr>
<tr>
<td>Plugin</td>
<td>Aka “plugin implementation”, a gRPC endpoint that implements the CSI Services.</td>
</tr>
<tr>
<td>Plugin Supervisor</td>
<td>Process that governs the lifecycle of a Plugin, MAY be the CO.</td>
</tr>
<tr>
<td>Workload</td>
<td>The atomic unit of “work” scheduled by a CO. A container or a collection of containers.</td>
</tr>
</tbody>
</table>
References

- What is Kubernetes?
- CSI Specification
- Kubernetes CSI Introduction
- DriveScale CSI plug-in
- Volumes, plus see also CSI
- Persistent Volumes
- Kubernetes Pods
- gRPC Principles
- Introduction to the DriveScale Architecture and API (PDF link at bottom of page)
- Children’s Guide to Kubernetes
- Omega: flexible, scalable schedulers for large computer clusters
- Kubernetes (Wikipedia)
- Persistent volumes by example
- DriveScale is certified OpenShift and now published in the Red Hat Container Catalog: https://access.redhat.com/containers/
- DriveScale Kubernetes Solution Brief