Above the clouds with container-native storage

Ryan Cook, Senior Software Engineer
Davis Phillips, Principal Software Engineer
Glenn West, Principal Software Engineer
Annette Clewett, Senior Storage Architect

May 10, 2018
Container-native Storage
Openshift Persistent Storage Framework
Openshift Persistent Storage Framework

A request for storage
Provider: ABC
Capacity: 10 GiB
Features: XYZ

“submits”
Openshift Persistent Storage Framework

**PersistentVolumeClaim**
- "A request for storage"
- Provider: ABC
- Capacity: 10 GiB
- Features: XYZ

**StorageClass**
- "A provider of storage"
- Provider URL: ...
- Credentials: ...
- Options: ...

**DEVELOPER**
- "submits"

**OPERATIONS**
- "sets up"
Openshift Persistent Storage Framework

A request for storage
Provider: ABC
Capacity: 10 GiB
Features: XYZ

A provider of storage
Provider URL: ...
Credentials: ...
Options: ...

Storage Backend

DEVELOPER

OPERATIONS

sets up

submits

submitted to

instructs
Openshift Persistent Storage Framework

A request for storage
Provider: ABC
Capacity: 10 GiB
Features: XYZ

A provider of storage
Provider URL: ...
Credentials: ...
Options: ...

Provisioned Storage
Name: ...
Size: ...
AccessMode: ...

sets up
creates
provisions

DEVELOPER

submit
submit to
instructs
provisions
Openshift Persistent Storage Framework

OPERATIONS

“sets up”

DEVELOPER

“submits”

PersistentVolumeClaim

“A request for storage”
Provider: ABC
Capacity: 10 GiB
Features: XYZ

“submitted to”

StorageClass

“A provider of storage”
Provider URL: …
Credentials: …
Options: …

“creates”

PersistentVolume

“Provisioned Storage”
Name: …
Size: …
AccessMode: …

“provisions”

Storage Backend

“instructs”

APPLICATION POD(S)

“mounted by”
GlusterFS - Distributed File Storage

Single, Global namespace

- Deploys on Red Hat-supported servers and underlying storage: DAS, JBOD
- Scale-out linearly
- Replicate synchronously and asynchronously
Federating Local Storage

CLIENT MOUNT

GLUSTERFS VOLUME

server1:/exp1
server2:/exp2

BRICK

FILE 1
FILE 2

FILE 1
FILE 2
Container-Native Storage on Openshift
Container-Native Storage on Openshift
Container-Native Storage on Openshift
Container-Native Storage on Openshift

1. PERSISTENT VOLUME CLAIM

2.)
Container-Native Storage on Openshift

1.) Persistent Volume Claim

2.)

3.) Gluster Volume

#redhat #rhsummit
Container-Native Storage on Openshift

AVAILABILITY ZONE A

AVAILABILITY ZONE B

AVAILABILITY ZONE C
VMware vSphere Platform
OpenShift Container Platform on VMware w/CNS

Storage types in vSphere

- NFS
- vSphere-volume
- CNS
vSphere Storage - NFS

Pros

- Easy to use and setup

Cons

- No ReadWriteMany or ReadOnlyMany support
- Limits POD scaling for services that would benefit from it
- Not supported for Registry, Metrics or Logging
vSphere Storage - vSphere-Volume

New in Kubernetes 1.6 using vSphere Cloud Provider

Pros

- Makes provisioning storage a streamlined process
- Supports multiple datastores in vSphere
- Dynamic Provisioning using Storage Classes

Cons

- No ReadWriteMany support
- Tied to Node
- Registry is not Supported
vSphere Storage - CNS Gluster File Storage

Pros

- ReadWriteMany and ReadOnlyMany Support
- Replication
- Highly available
- Dynamic Provisioning using Storage Classes
- Support for Registry, Logging and Metrics

Cons

- May require additional licenses
## vSphere Plugin - Why CNS?

<table>
<thead>
<tr>
<th>Topic</th>
<th>CNS/CRS</th>
<th>vSphere Plugin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Provisioning</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>RWO (Block), RWX (File), Object</td>
<td>✔ (All three in one)</td>
<td>✗ only one option (RWO)</td>
</tr>
<tr>
<td>Hybrid Deployment VMware, Bare Metal, AWS, ...</td>
<td>✔ (Dev, deploy, use anywhere)</td>
<td>✗</td>
</tr>
<tr>
<td>Online Resize Support</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Volume Metrics/PV Consumption</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Custom Volume Naming</td>
<td>✔ (Use business Rules)</td>
<td>✗</td>
</tr>
<tr>
<td>Integration with OCP</td>
<td>Install, Management, Monitoring</td>
<td>✗ (PV Allocation only)</td>
</tr>
<tr>
<td>High Availability</td>
<td>✔ 3 Failure Domains</td>
<td>✗ 2 Failure Domains</td>
</tr>
<tr>
<td>Mounts</td>
<td>✔ ~7-25 seconds</td>
<td>Slow ~ 1 min – 2 min</td>
</tr>
<tr>
<td>Failover</td>
<td>Transparent</td>
<td>Slow ~ 3-5 minutes (App outage)</td>
</tr>
</tbody>
</table>

Mounts: Slow ~ 1 min – 2 min
Failover: Slow ~ 3-5 minutes (App outage)
Sample Architecture - CNS Registry only
Sample Architecture - CNS App & Registry
Azure
OCP on Azure w/CNS

Storage Options

Existing Storage Choices

- Azure Disk
- CNS

AZURE DISK Profile

kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: fast
provisioner: kubernetes.io/azure-disk
parameters:
  skuName: Standard_LRS
  location: eastus
  storageAccount: azure_storage_account_name

CNS Profile

kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: slow
provisioner: kubernetes.io/glusterfs
parameters:
  resturl: "http://127.0.0.1:8081"
  restuser: "admin"
  secretName: "heketi-secret"
  secretNamespace: "default"
  gidMin: "40000"
  gidMax: "50000"
Azure Disk Storage

Pros
- Native to Azure
- Azure Underlying Redundancy Used
- Direct cost to usage per PV
- Direct use of Premium / SSD Drives

Cons
- ReadWriteOnce only supported model
- Number of drives per host severely limited (Limits containers per host)
- Very Slow PV Creation (Multiple Minutes)

Note
Make sure vmname and hostname the same
CNS Gluster File Storage vs Azure Disk

Pros

- Fast PV Create Time
- ReadWriteMany and ReadOnlyMany Support
- Highly available
- Dynamic Provisioning using Storage Classes
- Support for Registry, Logging and Metrics
- Ability to do very small to very large PV

Cons

- May require additional licenses
Azure with CNS Storage
# Azure Why CNS?

<table>
<thead>
<tr>
<th></th>
<th>Azure - File</th>
<th>CNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Provisioning</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Max File Size Supported</td>
<td>1 TB</td>
<td>Limited only by available resources</td>
</tr>
<tr>
<td>Max Size of a File Share</td>
<td>5 TB</td>
<td>Limited only by available resources</td>
</tr>
<tr>
<td>Client Support</td>
<td>Windows only with SMB and limited CIFS from Linux</td>
<td>POSIX compatible with Fuse</td>
</tr>
<tr>
<td>Availability zone resiliency</td>
<td>X, LRS, GRS only</td>
<td>HA across Availability Sets</td>
</tr>
<tr>
<td>Region resilience / availability</td>
<td>X, No RA-GRS</td>
<td>Geo-replicate across regions with read-only access to slave</td>
</tr>
<tr>
<td>Poor I/O, Unless $$</td>
<td>Extra capacity needed for Perf ex 512 GB disks for 2300 IOPS</td>
<td>Consolidate under CNS. CNS can divide and use Azure Data Disks efficiently</td>
</tr>
<tr>
<td>Vendor Lock-in</td>
<td>Yes</td>
<td>X</td>
</tr>
</tbody>
</table>
AWS
OCP on AWS w/CNS

Storage Types on AWS

- EFS
- EBS
- S3
AWS Storage - EFS

Pros

- Not AZ specific
- ReadWriteMany support
- Automatically scaled storage

Cons

- No guaranteed performance
- Not suitable for all application workloads
- Cannot be used for logging and metrics
- Requires a specific playbook to be ran
AWS Storage - EBS

Pros
- Guaranteed performance

Cons
- AZ specific
- ReadWriteMany support requires all pods must run on a single node
AWS Storage - S3

Pros

- Automatically scaled
- Perfect for the image registry

Cons

- Object based storage cannot be used for persistent container storage
AWS - CNS Gluster File Storage

Pros

● ReadWriteMany and ReadOnlyMany Support
● Replication
● Highly available regardless of AZ
● All workloads supported (logging, metrics, prometheus)

Cons

● May require additional licenses
● Requires large instances
## AWS - Why CNS?

<table>
<thead>
<tr>
<th></th>
<th>AWS - EBS</th>
<th>CNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Provisioning</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hybrid Cloud (Dev, Deploy, Use anywhere)</td>
<td>✗</td>
<td>public, bare-metal, virtual</td>
</tr>
<tr>
<td>Availability zone resiliency</td>
<td>✗  AZ failure domain</td>
<td>✓  Stretch across AZ’s,</td>
</tr>
<tr>
<td>Region resilience / availability</td>
<td>✗  one off snap copy</td>
<td>✓  Geo-replicate</td>
</tr>
<tr>
<td>Mounts</td>
<td>Slow 1 min – 2 min</td>
<td>✓ ~18-25 seconds</td>
</tr>
<tr>
<td>Failover</td>
<td>Slow ~ 3-5 minutes (App outage)</td>
<td>Transparent</td>
</tr>
<tr>
<td>Poor I/O, Unless $$</td>
<td>Extra capacity needed for Perf example- 500GB for 1500 IOPS</td>
<td>Consolidate under CNS. CNS can divide and use EBS efficiently</td>
</tr>
<tr>
<td>Shared Storage across nodes (RWX)</td>
<td>✗, use EFS, slow &amp; not across AZ’</td>
<td>✓  Same Cluster</td>
</tr>
<tr>
<td>Vendor Lock-in</td>
<td>Yes</td>
<td>✗</td>
</tr>
<tr>
<td>Volume Metrics, Consumption</td>
<td>No</td>
<td>✓ Yes</td>
</tr>
<tr>
<td>Online Volume resize</td>
<td>No (Need App to be restarted)</td>
<td>✓ Yes (in Progress)</td>
</tr>
<tr>
<td>Service Catalog, OCP Console, Prometheus</td>
<td>No</td>
<td>✓ Yes (in Progress)</td>
</tr>
</tbody>
</table>
Q&A
Links

- Ref archs
  - https://goo.gl/BhvCHm
- https://access.redhat.com/products/red-hat-container-native-storage