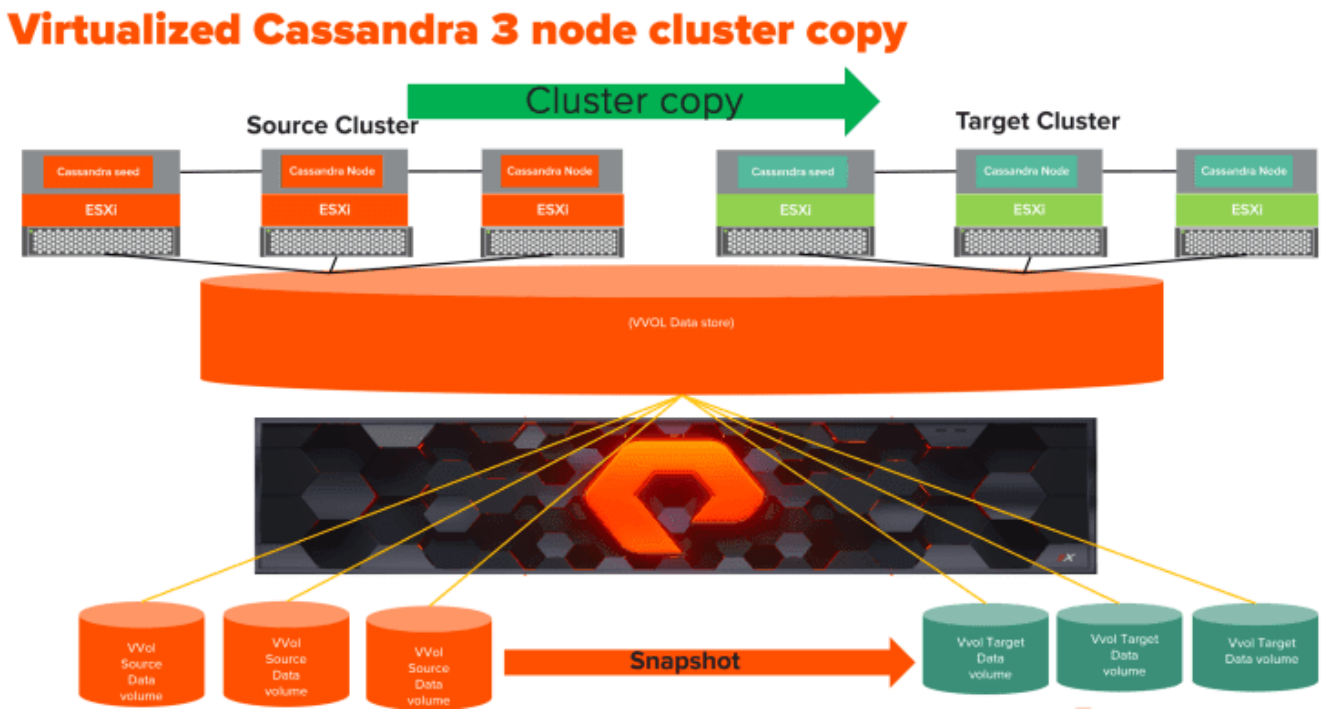
Copying Apache Cassandra Cluster Running On VVols Datastore



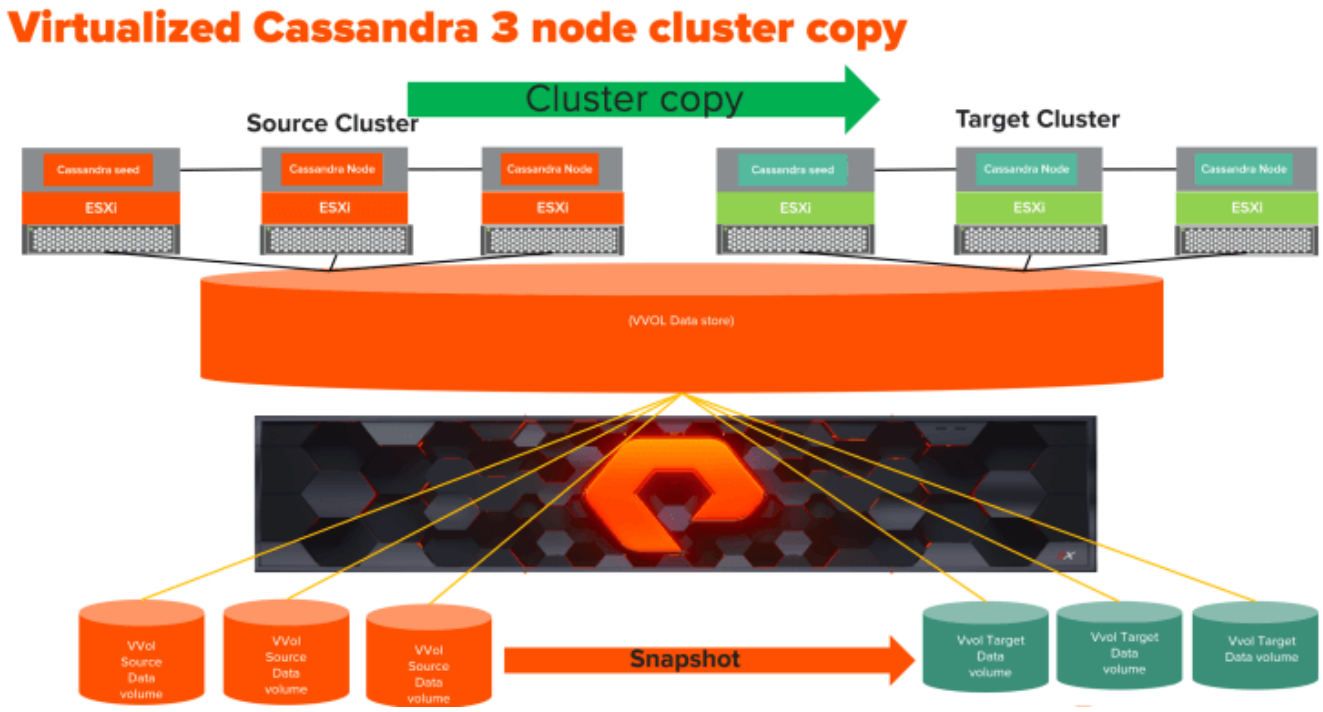
Copying Apache Cassandra Cluster Running On VVols Datastore Pure Storage FlashArray//X Snapshots-II

This is the second part of the two blog series. Here I would like to show how to copy virtualized Apache Cassandra to an existing Cassandra cluster using Pure Storage snapshots. In the previous blog, we used VMware VMFS datastore whereas, here we will cluster copy Cassandra with the database on VVols. It is interesting to see how much easier is Apache Cassandra cluster copy process using VVols compared to VMFS deployment of Cassandra cluster with Pure Storage snapshots.

Apache Cassandra is an open-source, distributed, wide column store, NoSQL database management

system designed to handle large amounts of data across many servers, providing high availability with no single point of failure.

Pure storage FlashArray//X snapshots are instantaneous and initially do not consume any additional disk space. The whole cluster copy process is accelerated using Pure Storage FlashArray//X snapshots.



As you can see in the diagram above, I have built two three-node Apache Cassandra clusters and each cluster has one seed node.

```
[root@Cassandra1SVVOL ~]# nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address       Load       Tokens       Owns    Host ID                               Rack
UN  10.21.211.29  259.41 KiB  256          ?       2c70ca55-0bea-4afa-8278-6c51274e0125  rack1
UN  10.21.211.28  316.57 KiB  256          ?       557b794e-5142-4607-a459-5c85edf409cc  rack1
UN  10.21.211.27  181.93 KiB  256          ?       0dfd86f6-e70e-4a60-a218-ab004ffad851  rack1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
[root@Cassandra1SVVOL ~]#
```

Each Cassandra node has a data volume (VVol) which corresponds to a volume on FlashArray. This simple, one-to-one relationship, makes it really easy to snapshot each volume and copy it to its respective volume (VVol) in the target cluster. Hence, the VVols based Cassandra cluster copy Cassandra cluster copy process is much simpler than a Cassandra cluster deployed using VMFS based data volumes. For more information please look in the following blog which talks about performance advantage with VVOls and also different ways we can take VVol snapshot.

Cody Hosterman Blog: VVol Snapshots

Now let us look into the cluster copy process in detail

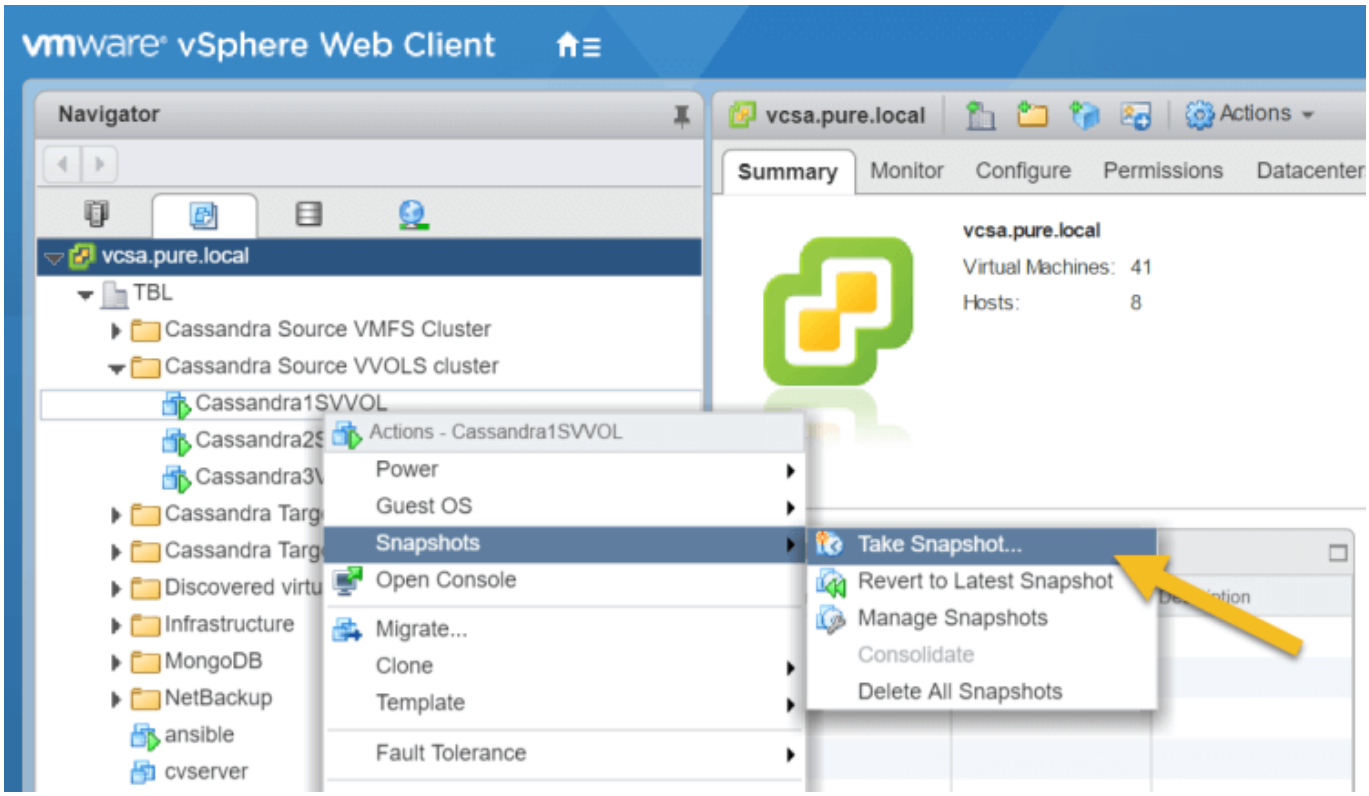
1. Run nodetool flush on all the Cassandra source nodes at the same time. This causes flush from the memtable to SSTables on disk and clears the logs.

→ *nodetool flush*

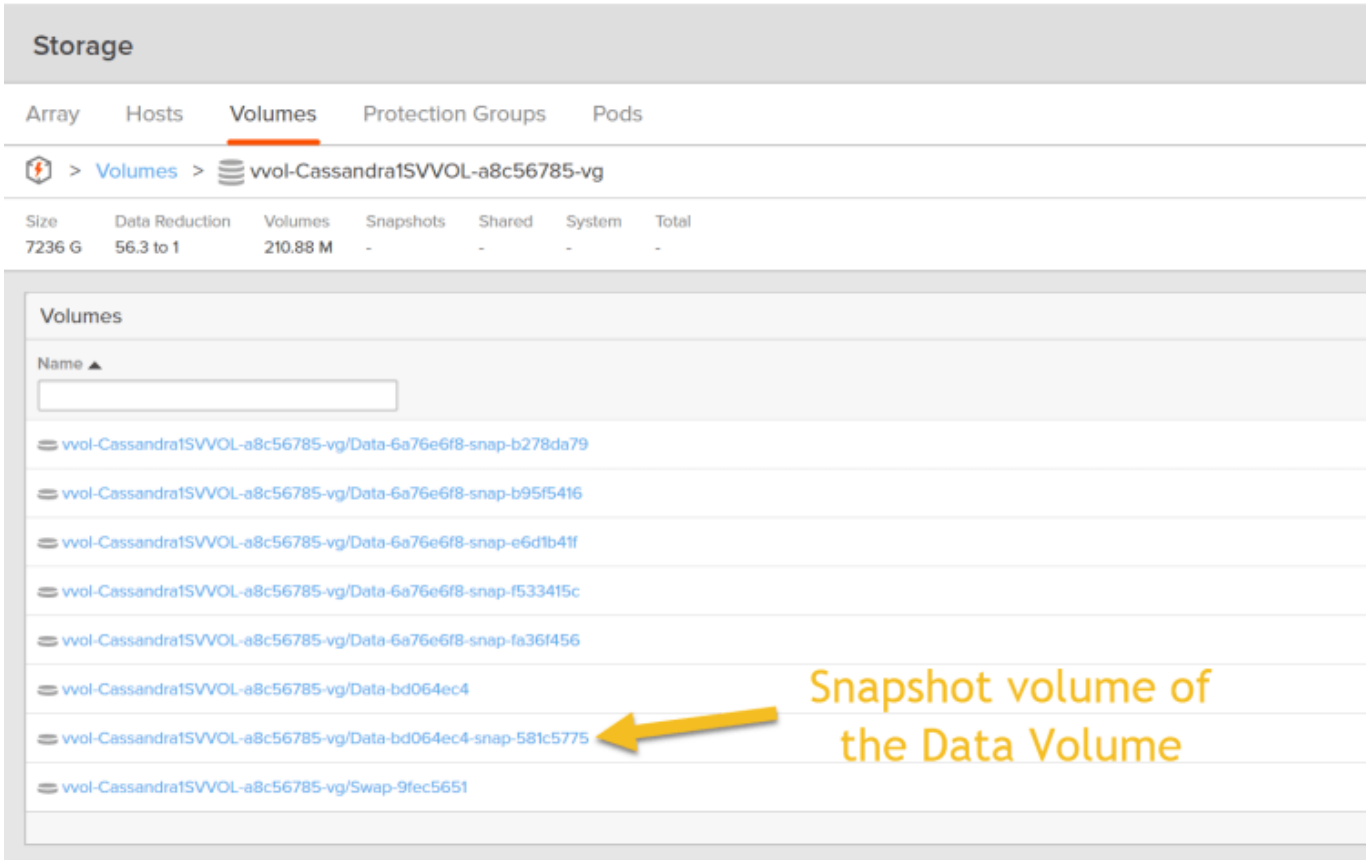
2. Take the Vmware snapshot on each of the Cassandra nodes: Here we are taking a managed snapshot where we Login to the vSphere client and select each of the nodes and open the context menu to take the snapshot.

“Node”->”Actions”->”Take Snapshot…”.

This creates a storage array-based snapshot which VMware is aware of.

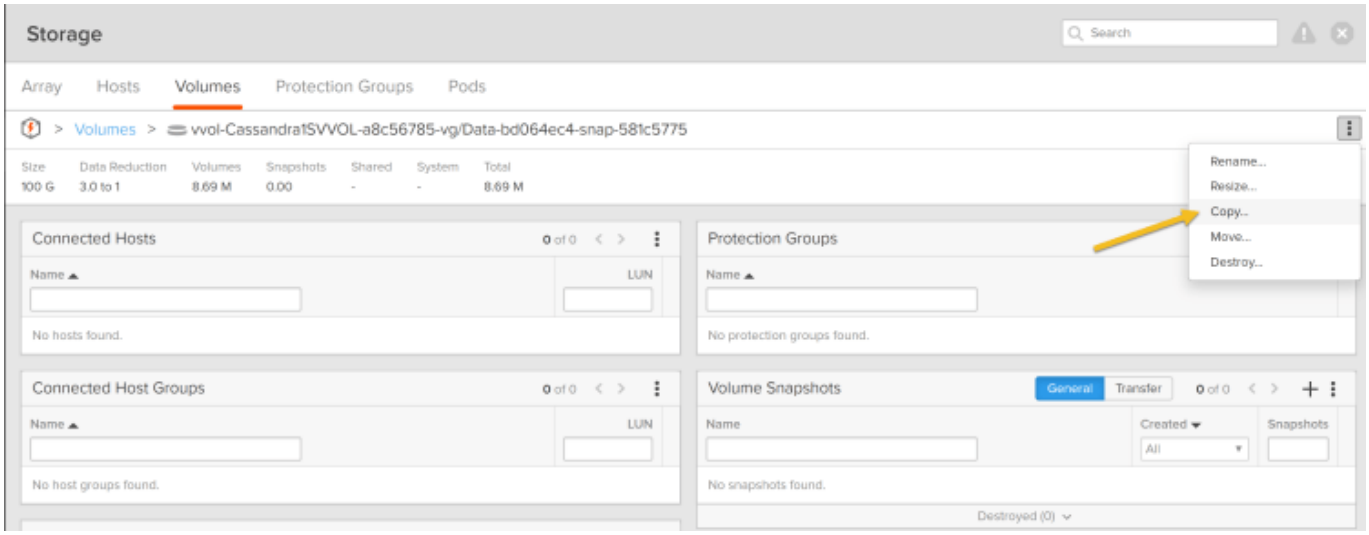


This creates the array based snapshot volume as shown below for a given Cassandra data volume:



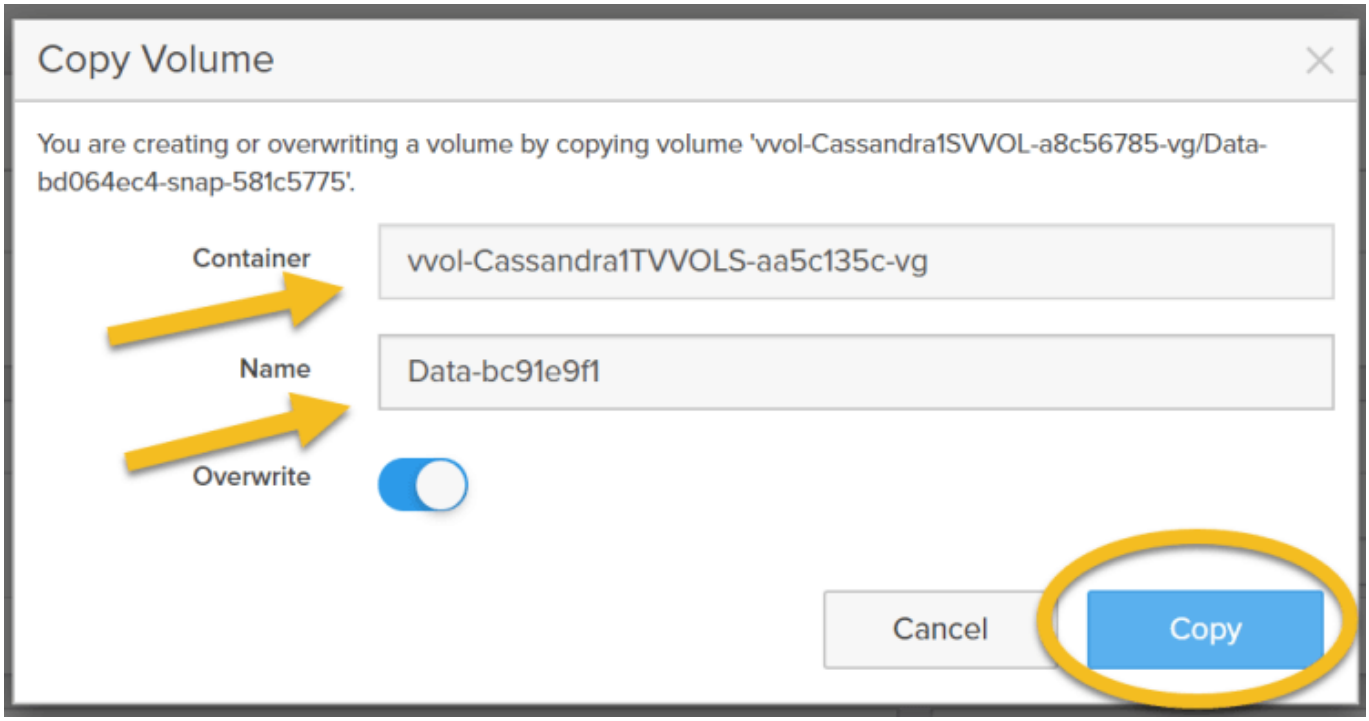
3. Copy the Pure Storage FlashArray//X snapshot volume to the target volume where target VVol is present. Before that make sure target Cassandra cluster is stopped and the data disk (/var/lib/cassandra/data) of cassandra is unmounted. Login to the Pure Storage FlashArray//X GUI

Goto Storage tab→ Go to Volumes→ Select the Source VVol Data snapshot volume → click Copy as shown below, overwrite the target VVol data volume.



Then provide the container name and the name of the target volume as shown below. This will overwrite

the target VVol from the source VVol.



1. Mount the volumes back on the target cluster using the mount command:

mount /dev/sdb /var/lib/cassandra/data

1. Important prerequisites: Source and target token ranges will not match, because the token ranges cannot be exactly the same in the other cluster. You need to specify the tokens for the target cluster that were used in the source cluster.

a. From the source cluster, retrieve the list of tokens associated with each node's IP:

nodetool ring | grep ip_address_of_node | awk '{print $NF ","}' | xargs

b. In the cassandra.yaml file for each node in the target cluster, add the list of tokens you obtained in the previous step to the initial_token parameter using the same num_tokens setting as in the source cluster.

c. Make any other necessary changes in the new cluster's cassandra.yaml and property files so that the source nodes match the target cluster settings. Make sure the seed nodes are set for the target cluster.

d. Clear the system table data from each new node:

*sudo rm -rf /var/lib/cassandra/data/system/**

This allows the new nodes to use the initial tokens defined in the cassandra.yaml when they restart.

e. Make sure the Cassandra commit logs are also cleared in the target nodes. Otherwise, it will cause inconsistencies.

```
sudo rm -f /var/lib/cassandra/commitlog/
```

1. Restart Cassandra on all nodes, you should have the latest data from the source cluster.

This concludes the first part where we refreshed a target virtualized Cassandra cluster using source virtualized Cassandra cluster using VVols and Pure Storage FlashArray//X snapshots. As seen here this process is extremely simple and does not require additional steps like creating a new data store, assigning a new signature, etc which is needed for VMFS based deployment.