Technical Documentation
StorPool Volume Management

StorPool version 16.01

Document version 2016-06-07

Volumes

This is a volume

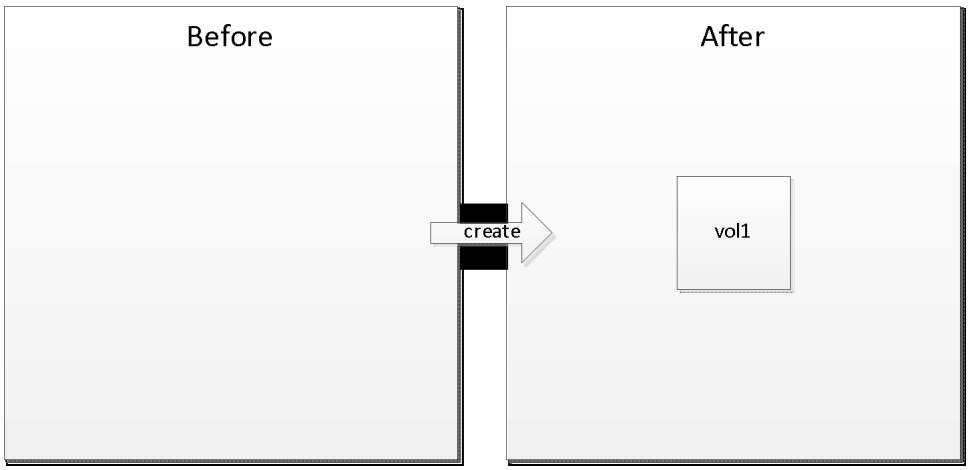
Volumes are the basic service of the StorPool storage system. They have a name and a certain size. They can be read from and written to. They can be attached to hosts as read-only or read-write block devices under /dev/storpool . The volume name is a string consisting of one or more of the following characters:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
. : - _
```

These are lower- and upper-case Latin letters, numbers and the delimiters dot(.), colon(:), dash(-) and underscore(_) .

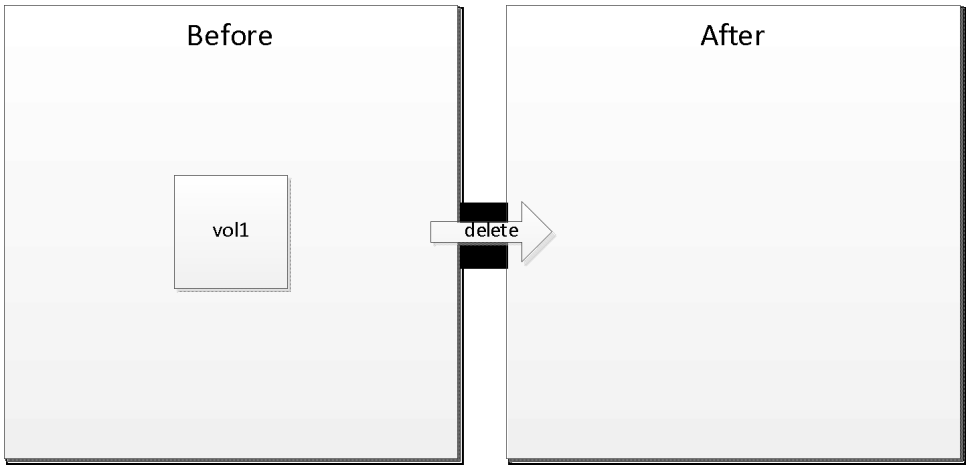
Creating a volume

```
StorPool> volume vol1 size 10G replication 3
API: POST /ctrl/1.0/VolumeCreate + json
```



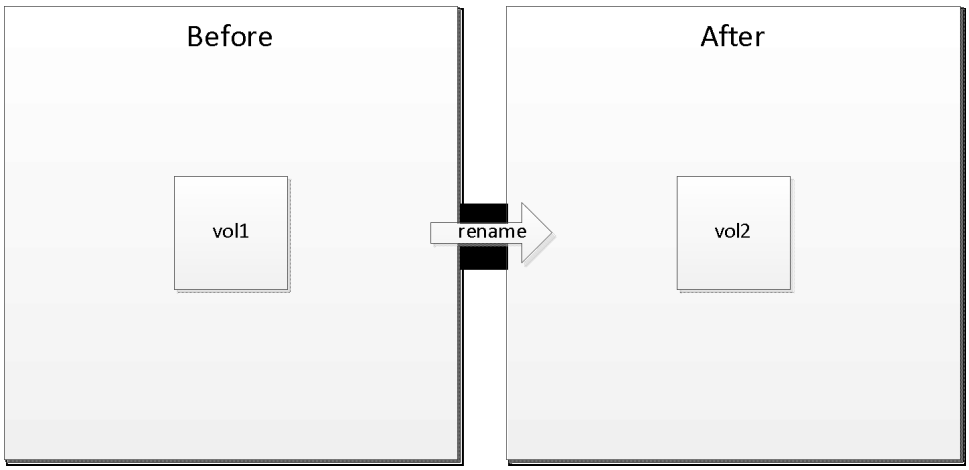
Deleting a volume

```
StorPool> volume vol1 delete vol1
API: POST /ctrl/1.0/VolumeDelete/vol1
```



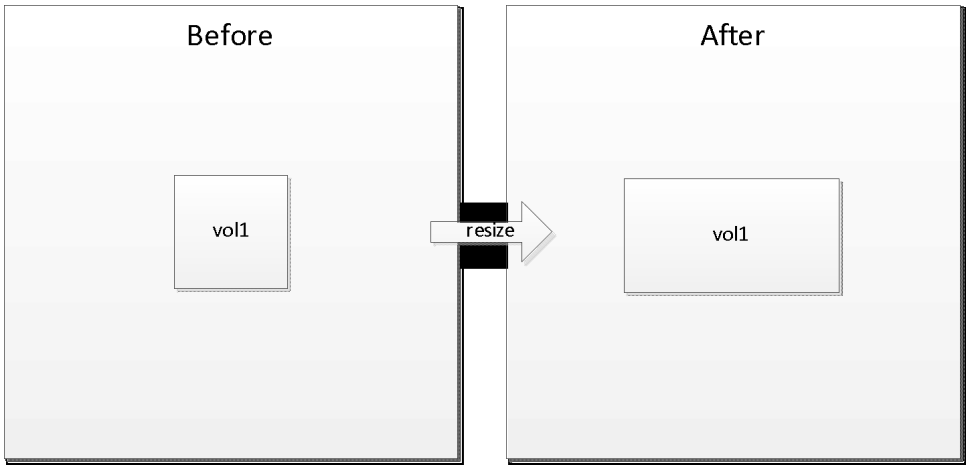
Renaming a volume

```
StorPool> volume vol1 rename vol2
API: POST /ctrl/1.0/VolumeUpdate/vol1 + json
```

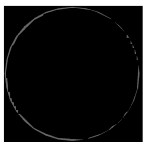


Resizing a volume

```
StorPool> volume vol1 size +10G
API: POST /ctrl/1.0/VolumeUpdate/vol1 + json
```


Before
vol1
resize
After
vol1

Snapshots

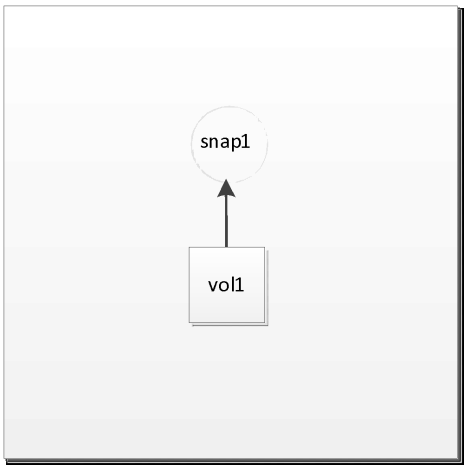


This is a snapshot

Snapshots are read-only point-in-time images of volumes. They are created once and cannot be changed. They can be attached to hosts as read-only block devices under /dev/storpool/

Volumes and Snapshots (name-space):
vol1
vol2
snap1
snap2

Volumes and snapshots share the same namespace. Names of volumes and snapshots are unique within a StorPool cluster.

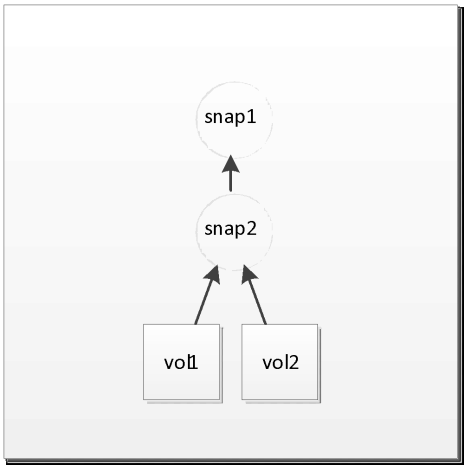


This diagram illustrates the relationship between a snapshot and a volume.

Volume vol1 is based on snapshot snap1. Vol1 contains only the changes since snap1 was taken. In the common case this is a small amount of data.

Arrows indicate a child-parent relationship. Each volume or snapshot may have exactly one parent which it is based upon.

Writes to vol1 are recorded within the volume. Reads from vol1 may be served by vol1 or by its parent snapshot – snap1, depending on whether vol1 contains changed data for the read request or not.

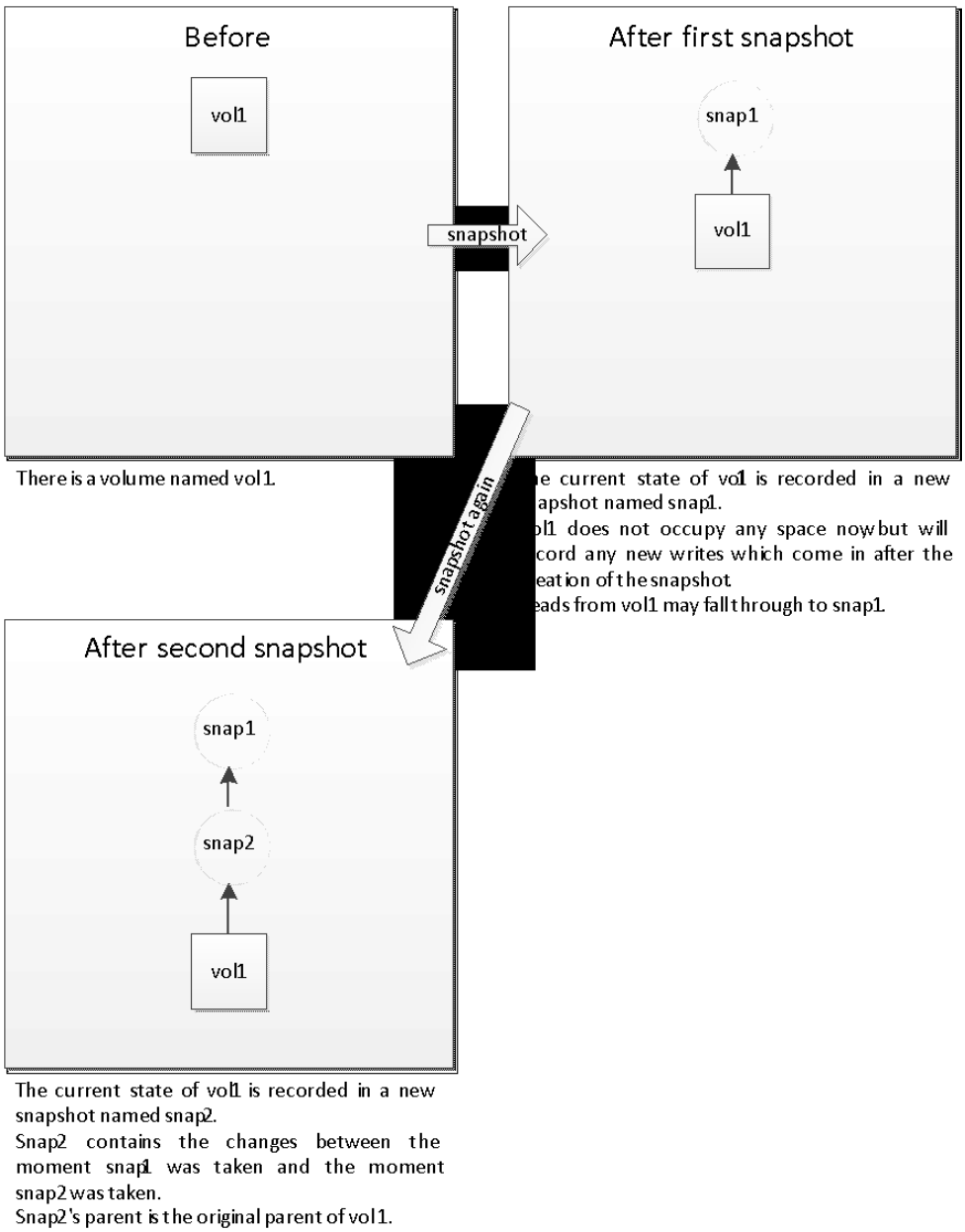


Snapshots and volumes are completely independent. Each snapshot may have many children (volumes and snapshots). Volumes cannot have children.

Snap1 contains a full image. Snap2 contains only the changes since snap1 was taken. Vol1 and vol2 contain only the changes since snap2 was taken.

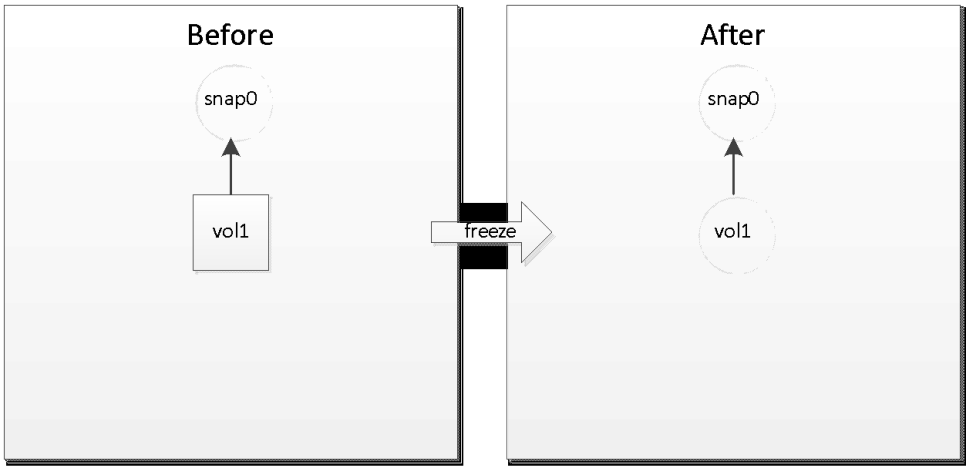
Creating a snapshot of a volume

```
StorPool> volume vol1 snapshot snap1
API: POST /ctrl/1.0/VolumeSnapshot/vol1 + json
```



Converting a volume to a snapshot (freeze)

```
StorPool> volume vol1 freeze
API: POST /ctrl/1.0/VolumeFreeze/vol1
```

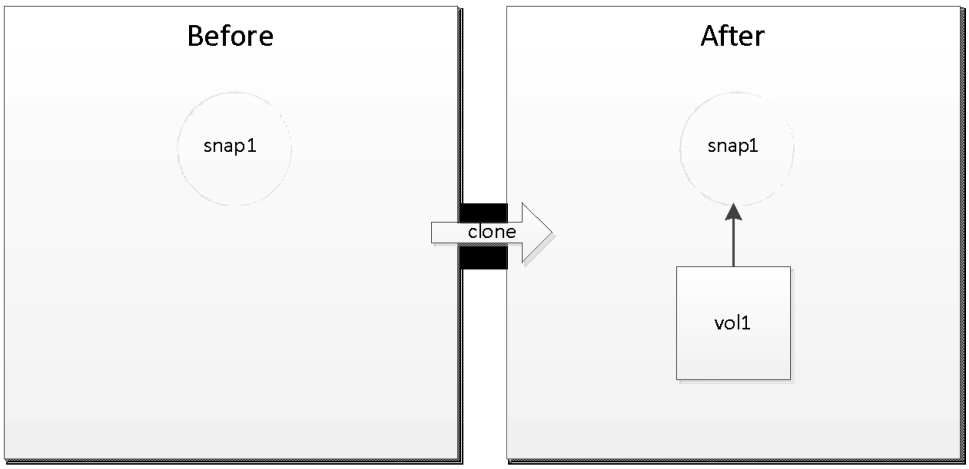


There is a volume named vol1, based on a snapshot snap0.
Vol1 contains only the changes since snap0 was taken.

The current state of vol1 is recorded in a new snapshot of the same name.
The Snapshot vol1 contains the changes between the moment snap0 was taken and the moment vol1 was frozen.

Creating a volume based on an existing snapshot (aka cloning)

```
StorPool> volume vol1 parent snap1
API: POST /ctrl/1.0/VolumeCreateFromSnapshot/snap1 + json
```

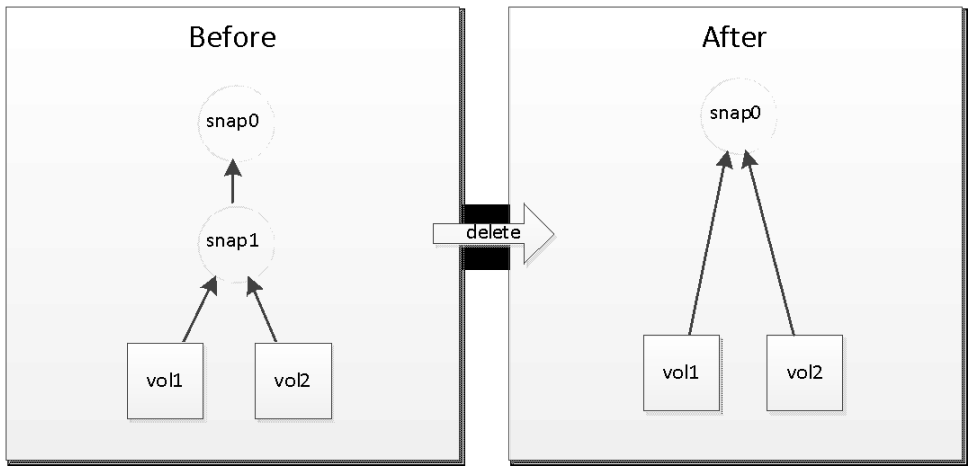


Before the creation of vol1 there is a snapshot named snap1.

A new volume, named vol1 is created. Vol1 is based on snap1.
The newly created volume does not occupy any space initially. Reads from the vol1 may fall through to snap1 or to snap1's parents (if any).

Deleting a snapshot

```
StorPool> snapshot snap1 delete snap1
API: POST /ctrl/1.0/SnapshotDelete/snap1
```



Vol1 and vol2 are based on snap1. Snap1 is based on snap0.

Snap1 contains the changes between the moment snap0 was taken and when snap1 was taken.

Vol1 and Vol2 contain the changes since the moment snap1 was taken.

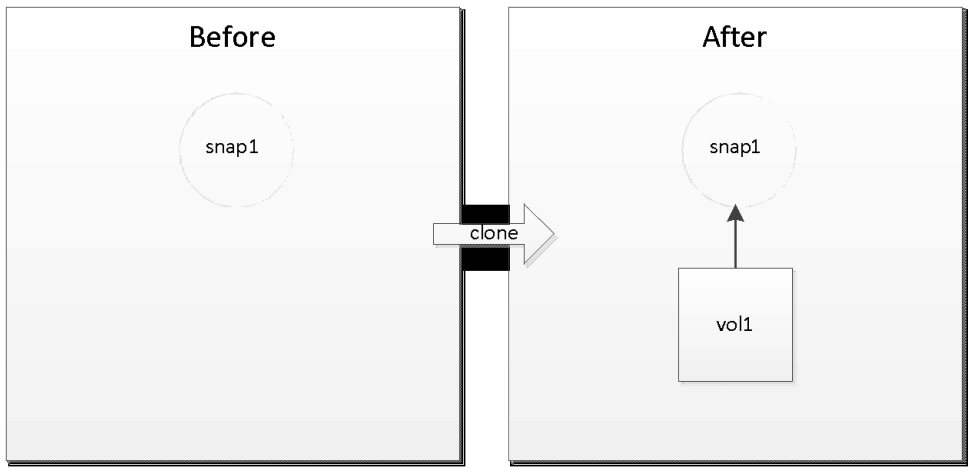
After the deletion, vol1 and vol2 are based on snap1's original parent (if any). In the example they are now based on snap0.

When deleting a snapshot, the changes contained therein are propagated to all its children (if any). In this case vol1 and vol2 now contain all changes since snap0 was taken, not just since snap1.

Although StorPool tries to minimize the harmful effects of this, deleting a snapshot with many children may lead to an explosion of disk space usage.

Creating a volume based on an existing snapshot (aka cloning)

```
StorPool> volume vol1 parent snap1
API: POST /ctrl/1.0/VolumeCreateFromSnapshot/snap1 + json
```

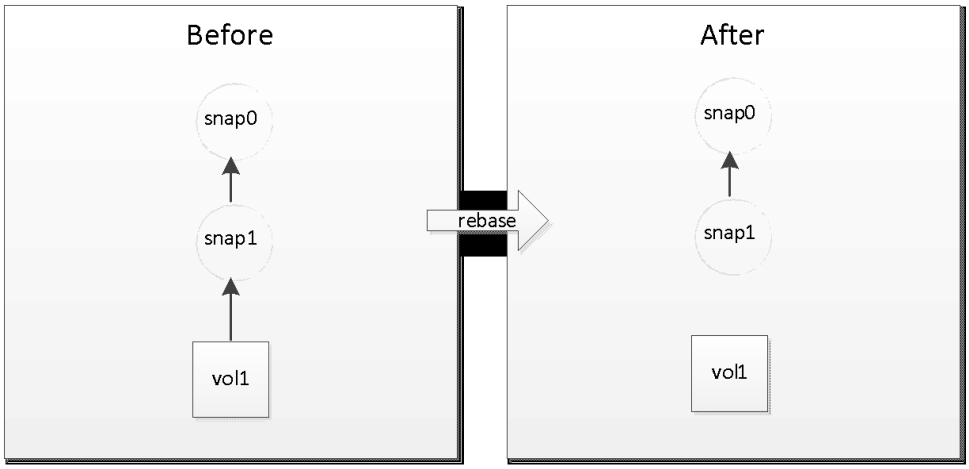


Before the creation of vol1 there is a snapshot named snap1.

A new volume, named vol1 is created. Vol1 is based on snap1.
The newly created volume does not occupy any space initially. Reads from the vol1 may fall through to snap1 or to snap1's parents (if any).

Rebase to null (aka promote)

```
StorPool> volume vol1 rebase
API: POST /ctrl/1.0/VolumeRebase/vol1
or
StorPool> snapshot snap2 rebase
API: POST /ctrl/1.0/SnapshotRebase/snap2
```

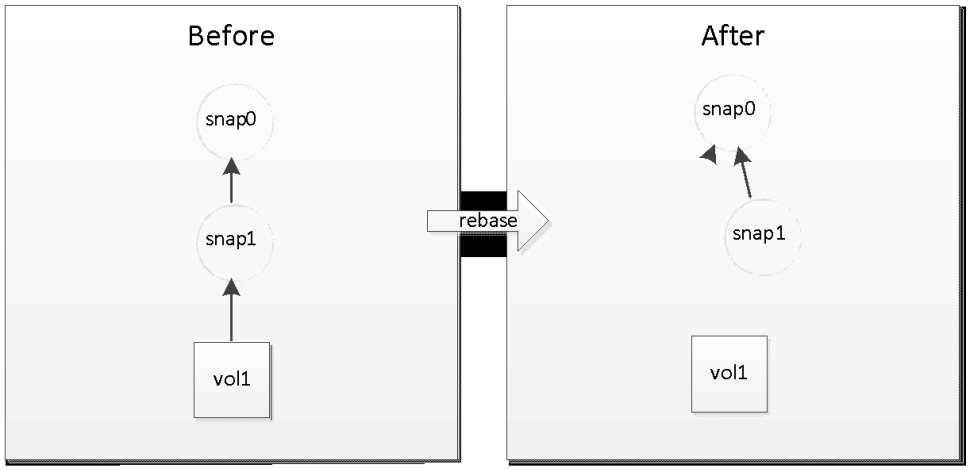


Vol1 is based on snap1. Snap1 is in turn based on snap0.
Snap1 contains the changes between the moment snap0 was taken and up to the moment snap1 was taken.
Vol1 contains the changes from the moment snap1 was taken until now.

After promotion vol1 is not based on a snapshot.
Vol1 now contains all data, not just the changes since snap1 was taken.
Snap1 and snap0 are unaffected.

Rebase

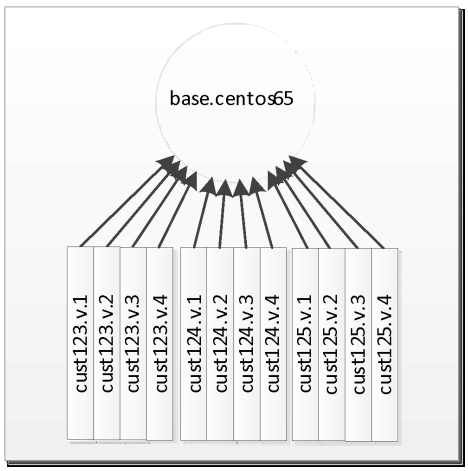
```
StorPool> volume vol1 rebase snap0
API: POST /ctrl/1.0/VolumeRebase/vol1 + json
or
StorPool> snapshot snap2 rebase snap0
API: POST /ctrl/1.0/SnapshotRebase/snap2 + json
```



Vol1 and vol2 are based on snap1. Snap1 is based on snap0.
Snap1 contains the changes between the moment snap0 was taken and when snap1 was taken.
Vol1 contains the changes since the moment snap1 was taken and now.

After rebase vol1 is based on snap0.
Vol1 now contains all changes since snap0 was taken, not just since snap1.
Snap1 is unchanged.

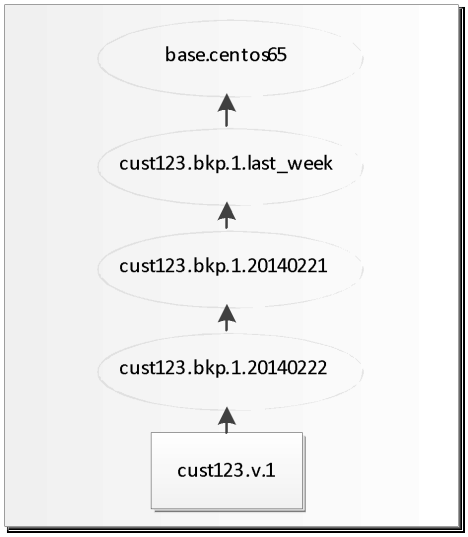
Example use of snapshots



This is a semi-realistic example of how volumes and snapshots may be used.

There is a snapshot called base.centos65. This snapshot contains a base CentOS 6.5 VM image, which was prepared carefully by the service provider.

There are 3 customers with 4 virtual machines each. All virtual machine images are based on CentOS 6.5, but may contain custom data, which is unique to each VM.



This example shows another typical use of snapshots – for backup.

In the example, there is one base image for Centos 6.5, three backup snapshots and one live volume "cust123.v.1"

Attaching/detaching volumes and snapshots

Attaching a volume

```
StorPool> attach volume vol1 client 1
API: POST /ctrl/1.0/AttachmentUpdate + json
```

Attaches volume vol1 on client 1.

This creates a block device named /dev/storpool/vol1 on the host with id 1.

You can attach volumes as read-only or read-write.

Detaching a volume

```
StorPool> detach volume vol1 client 1
API: POST /ctrl/1.0/AttachmentUpdate + json
```

Detaches vol1 from the host with id 1.

This removes the block device /dev/storpool/vol1.

Attaching/detaching a snapshot

```
StorPool> attach volume snap1 client 1 mode ro
StorPool> detach volume snap1 client 1
API: POST /ctrl/1.0/AttachmentUpdate + json
```