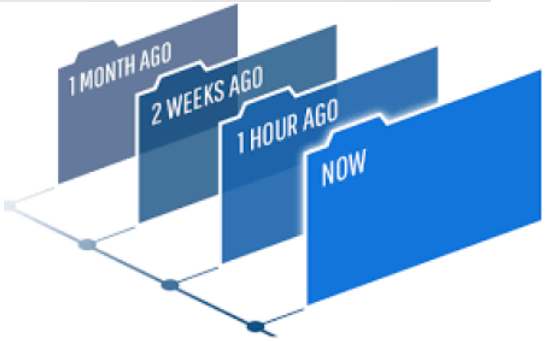
Openmediavault

HOW-TO: Setup automated, Self Rotating and Purging ZFS snapshots

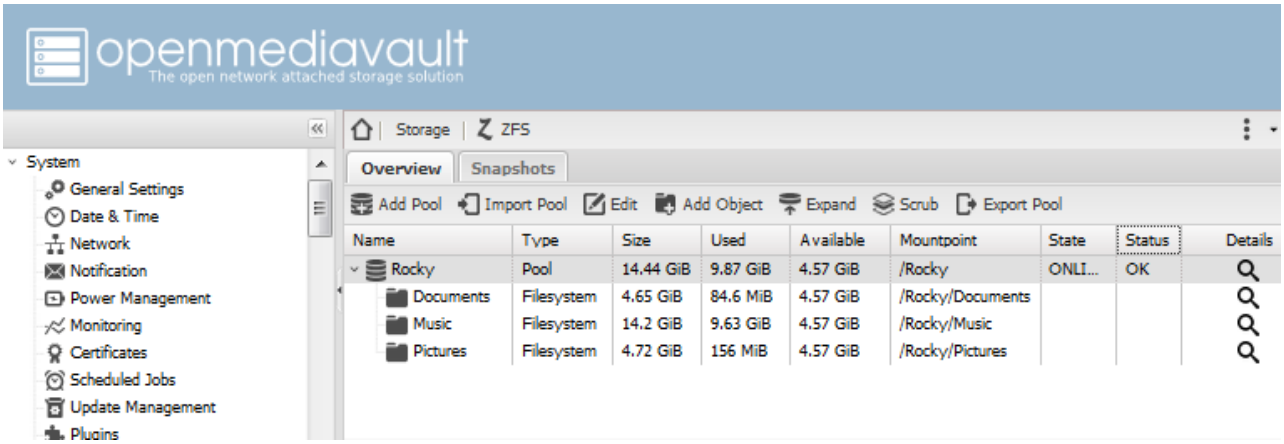


For ZFS users:
This guide will show how to setup and take advantage of one of ZFS' most valuable features for restoration, SNAPSHOTS. **zfs-auto-snapshot**, automatically creates, prunes, and destroys periodic snapshots. Other than ZFS utilities which are installed with the ZFS filesystem, and cron which is part of most Linux installations, there are no specific dependencies.

General:
By the very nature and function of a CoW (copy on write) filesystem, ZFS gives users the ability to "capture" the state of their file system at a given moment in time and preserve it using snapshots. Since only changes to ZFS filesystems are maintained between current filesystem states and historic snapshots, snapshoting is very efficient. For largely static pools, maintaining the cumulative differences between snapshots, shouldn't require more than an extra 25% disk space.
Having the ability to "roll back" the pool, individual filesystems in the pool, or retrieve individual files from previous snapshots has obvious advantages. These abilities would allow an entire pool to be restored to a time before a virus infection, a ransomware infection, and allow the retrieval of deleted files or previous versions of files.

An excellent overview of how ZFS snapshots work is available on Youtube. ZFS snapshots. While the presentation is 46 minutes, the video can be skimmed for the highlights.

For the purpose of illustration - the following pool and child filesystems were created:



To install **zfs-auto-snapshot**, from the command line as **root**:

```
wget https://github.com/zfsonlinux/zfs-auto-snapshot/archive/master.zip
unzip master.zip
```

If the output of the above is; "`-bash: unzip: command not found`**" do **`apt install unzip`** and rerun the unzip command.**

```
cd zfs-auto-snapshot-master
make install
```

By default the installed scripts will run the following snapshot jobs separately on the pool and on all individual child filesystems.

frequent snapshots run every 15 mins, keeping 4 snapshots
hourly snapshots run every hour, keeping 24 snapshots
daily snapshots run every day, keeping 31 snapshots
weekly snapshots run every week, keeping 7 snapshots
monthly snapshots run every month, keeping 12 snapshots

With default settings, all of these jobs can preserve previous states of the pool and child filesystem(s) for up to a year.

For the purpose of understanding what is captured in a Snapshot:

A "pool snapshot" captures the state of the parent pool, in this example "Rocky", and all Linux folders, sub-directories, and files contained in the parent pool. Child filesytems are separate ZFS objects and are not included in snapshots of the parent pool.

****In the interests of clarity:****
A ZFS "filesystem" is interchangeable with a standard Linux folder at the root of the pool and is navigable, on the command line, in the same manner. The difference is a standard Linxu folder at the root of the pool, is part of the parent pool. A child "filesystem" is created by ZFS and is assigned editable ZFS properties. A "filesystem snapshot" is taken independent of the parent pool and is independently restorable.

Customizing Snapshot jobs

By default all snapshot time intervals are set to "true". Changes are required only if a specific time interval is not desired.

Pool Snapshots:

The following command lines can be used to selectively turn snapshot intervals **ON** or **OFF** for the **pool** named **Rocky**. In this example, the first line enables or disables snapshots of the pool "Rocky". The lines, after the first line, enable specific snapshot time intervals.

zfs set com.sun:auto-snapshot=**true** Rocky
zfs set com.sun:auto-snapshot:frequent=true Rocky
zfs set com.sun:auto-snapshot:hourly=true Rocky
zfs set com.sun:auto-snapshot:daily=true Rocky
zfs set com.sun:auto-snapshot:weekly=true Rocky
zfs set com.sun:auto-snapshot:monthly=**false** Rocky

Time intervals where a snapshot is desired are set to **true**. Intervals that are not desired are set to **false**. In this example, with all intervals up to "weekly" set to "true", the pool "Rocky" will have a snapshot recorded 4 times an hour (4 total), once per hour (24 total), once a day (31 total), and once a week (7 total). In this case, it would be possible roll this pool back to any one of several snapshots taken during the past 7 weeks.
Again, in the example above, **true** is assumed on all lines. To turn off monthly snapshots for the pool named Rocky, a single command is required:

```
zfs set com.sun:auto-snapshot:monthly=false Rocky
```

Filesystem Snapshots:

The following command lines can be used to selectively turn snapshots on or off for a **filesystem**. In this example, the first line enables or disables snapshots of the **Rocky/Pictures** filesystem. The following lines enable specific snapshot time intervals.

zfs set com.sun:auto-snapshot=**true** Rocky/Pictures
zfs set com.sun:auto-snapshot:frequent=**false** Rocky/Pictures
zfs set com.sun:auto-snapshot:hourly=**false** Rocky/Pictures
zfs set com.sun:auto-snapshot:daily=true Rocky/Pictures
zfs set com.sun:auto-snapshot:weekly=**false** Rocky/Videos_TV
zfs set com.sun:auto-snapshot:monthly=**false** Rocky/Videos

Again, time intervals where a snapshot is desired are set to **true**. Periods that are not desired are set to **false**. In this example, with "**daily=true**" , the filesystem "Rocky/Pictures" will have a snapshot recorded once a day, everyday, retaining 31 snapshots totaling 1 month. In this case, it would be possible roll this file system back to a snapshot taken on any day, in the prior 31 days.

Again, because **true** is the default for all intervals, to disable the unneeded intervals, the following commands on the Command Line Interface (CLI) would be required:

```
zfs set com.sun:auto-snapshot:frequent=false Rocky/Pictures
zfs set com.sun:auto-snapshot:hourly=false Rocky/Pictures
zfs set com.sun:auto-snapshot:weekly=false Rocky/Videos_TV
zfs set com.sun:auto-snapshot:monthly=false Rocky/Videos
```

Looking at and Organizing Snapshots:

All snapshots taken by zfs-auto-snapshot are viewable in the OMV GUI. OMV provides a tab for looking at and sorting snapshots under **Storage, ZFS, Snapshots**. (On the CLI **zfs list -t snapshot** will provide a list of snapshots but the shear length of the list makes it unwieldy.)

Rolling Back

While rolling back a file system or the parent pool is relatively easy, if going significantly back in time (beyond the most recent snapshot) the roll back feature will not work in the OMV GUI. However, a roll back can be done with the following command line:

(In this example, the roll back was done to **the parent "Rocky" pool**.)
zfs rollback **Rocky**@zfs-auto-snap_hourly-2018-01-14-1617 -r

(In this example – the roll back would be done to **the "Rocky/Pictures" filesystem**.)
zfs rollback **Rocky/Pictures**@zfs-auto-snap_hourly-2018-01-14-1617 -r

Using the exact name of the snapshot provided in the OMV GUI (examples in red above) and the above rollback command with the -r suffix, the pool or child filesystems can be rolled back to the date / time of the specified snapshot.

Notes:

1. For maximum snapshot flexibility, creating child filesystems on the parent pool is essential.

2. Think of the roll back feature as going back in time, in the ZFS "time line". When rolling back to a specific date/time snapshot, ZFS destroys all snapshots (all file changes, additions and deletions) between the present state and the past state of the filesystem. This is necessary, using a time travel analogy, to avoid a "Paradox". (Where two different time lines of the file system would exist.) Rolling back is permanent. It's not possible to undo a roll back or "spring forward".

   The practical implications are:
   - In order to not lose file changes, deletions, etc., it's better to roll back the shortest interval possible.
   - By extension, it's better to roll back a single filesystem (the ZFS equivalent of a root folder), than it is to roll back an entire pool without child filesystems.
   - Due to the minimal overall impact, to prevent the loss of file versions, etc; selective restoration(s) of files and folders is preferred and is "best practice".
   - Roll backs are a disaster recovery option that, if applied, should but be as narrow as possible (a single filesystem) and limited to the shortest time interval possible.

3. If a particular file system experiences a high rate of file turnover or file versioning (databases, etc.), snapshot retention periods may need to be shortened to prevent excessive use of disk space. Given this consideration, a dedicated filesystem could be used to isolate frequently changing, transactional data sets, from relatively static file storage.
4. While it is an advanced technique; it is possible to create a file system "clone" from a past snapshot, without rolling back. The clone could be used as a source of folders and files as they existed when the snapshot was taken, without sacrificing the snapshots in between. (See references for further details.)
5. While **zfs-auto-snapshot** creates considerable flexibility in file, filesystem, and pool restorations, it is NOT BACKUP. If a ZFS pool is lost, local snapshots are lost as well.
6. If more advanced features are needed, such as offloading snapshots to an external host, ZnapZend might be a more appropriate solution.

Additional Information:

zfs-auto-snapshot source:
https://github.com/zfsonlinux/zfs-auto-snapshot/
A Comprehensive and well written ZFS reference:
https://pthree.org/2012/12/19/zfs-administration-part-xii-snapshots-and-clones/

ZFS Video Tutorial:
Part 1: https://sysadmincasts.com/episodes/35-zfs-on-linux-part-1-of-2
Part 2: https://sysadmincasts.com/episodes/37-zfs-on-linux-part-2-of-2
** Due to a difference in repositories, do not install ZFS and use a pool created in Tutorial, with OMV. It's best to install ZFS using OMV's GUI.
** While this tutorial is informative, it would be best done using a newly pool that can be destroyed when the tutorial is complete. Doing any roll back operation on an existing ZFS pool/filesystem comes with a risk of "fat finger" mistakes.