Aaron Toponce

{ 2012 12 19 }

ZFS Administration, Part XII- Snapshots and Clones

Table of Contents

Snapshots with ZFS are similar to snapshots with Linux LVM. A snapshot is a first class read-only filesystem. It is a mirrored copy of the state of the filesystem at the time you took the snapshot. Think of it like a digital photograph of the outside world. Even though the world is changing, you have an image of what the world was like at the exact moment you took that photograph. Snapshots behave in a similar manner, except when data changes that was part of the dataset, you keep the original copy in the snapshot itself. This way, you can maintain persistence of that filesystem.

You can keep up to 2^64 snapshots in your pool, ZFS snapshots are persistent across reboots, and they don't require any additional backing store; they use the same storage pool as the rest of your data. If you remember our post about the nature of copy-on-write filesystems, you will remember our discussion about Merkle trees. A ZFS snapshot is a copy of the Merkle tree in that state, except we make sure that the snapshot of that Merkle tree is never modified.

Creating snapshots is near instantaneous, and they are cheap. However, once the data begins to change, the snapshot will begin storing data. If you have multiple snapshots, then multiple deltas will be tracked across all the snapshots. However, depending on your needs, snapshots can still be exceptionally cheap.

Creating Snapshots

You can create two types of snapshots: pool snapshots and dataset snapshots. Which type of snapshot you want to take is up to you. You must give the snapshot a name, however. The syntax for the snapshot name is:

- *pool/dataset@snapshot-name*
- *pool@snapshot-name*

To create a snapshot, we use the "zfs snapshot" command. For example, to take a snapshot of the "tank/test" dataset, we would issue:

```
# zfs snapshot tank/test@tuesday
```

Even though a snapshot is a first class filesystem, it does not contain modifiable properties like standard ZFS datasets or pools. In fact, everything about a snapshot is read-only. For example, if you wished to enable compression on a snapshot, here is what would happen:

```
# zfs set compression=lzjb tank/test@friday
cannot set property for 'tank/test@friday': this property can not be modified for snapshots
```

Listing Snapshots

Snapshots can be displayed two ways: by accessing a hidden ".zfs" directory in the root of the dataset, or by using the "zfs list" command. First, let's discuss the hidden directory. Check out this madness:

```
# ls -a /tank/test
./  ../  boot.tar  text.tar  text.tar.2
# cd /tank/test/.zfs/
# ls -a
./  ../  shares/  snapshot/
```

Even though the ".zfs" directory was not visible, even with "ls -a", we could still change directory to it. If you wish to have the ".zfs" directory visible, you can change the "snapdir" property on the dataset. The valid values are "hidden" and "visible". By default, it's hidden. Let's change it:

```
# zfs set snapdir=visible tank/test
# ls -a /tank/test
./  ../  boot.tar  text.tar  text.tar.2  .zfs/
```

The other way to display snapshots is by using the "zfs list" command, and passing the "-t snapshot" argument, as follows:

```
# zfs list -t snapshot
NAME                                  USED  AVAIL  REFER  MOUNTPOINT
pool/cache@2012:12:18:51:2:19:00         0      -   525M  -
pool/cache@2012:12:18:51:2:19:15         0      -   525M  -
pool/home@2012:12:18:51:2:19:00       18.8M      -  28.6G  -
pool/home@2012:12:18:51:2:19:15       18.3M      -  28.6G  -
pool/log@2012:12:18:51:2:19:00         184K      -  10.4M  -
pool/log@2012:12:18:51:2:19:15         184K      -  10.4M  -
pool/swap@2012:12:18:51:2:19:00          0      -    76K  -
pool/swap@2012:12:18:51:2:19:15          0      -    76K  -
pool/vmsa@2012:12:18:51:2:19:00          0      -  1.12M  -
pool/vmsa@2012:12:18:51:2:19:15          0      -  1.12M  -
pool/vmsb@2012:12:18:51:2:19:00          0      -  1.31M  -
pool/vmsb@2012:12:18:51:2:19:15          0      -  1.31M  -
tank@2012:12:18:51:2:19:00               0      -  43.4K  -
tank@2012:12:18:51:2:19:15               0      -  43.4K  -
tank/test@2012:12:18:51:2:19:00          0      -  37.1M  -
tank/test@2012:12:18:51:2:19:15          0      -  37.1M  -
```

Notice that by default, it will show all snapshots for all pools.

If you want to be more specific with the output, you can see all snapshots of a given parent, whether it be a dataset, or a storage pool. You only need to pass the "-r" switch for recursion, then provide the parent. In this case, I'll see only the snapshots of the storage pool "tank", and ignore those in "pool":

```
# zfs list -r -t snapshot tank
NAME                                  USED  AVAIL  REFER  MOUNTPOINT
tank@2012:12:18:51:2:19:00               0      -  43.4K  -
tank@2012:12:18:51:2:19:15               0      -  43.4K  -
tank/test@2012:12:18:51:2:19:00          0      -  37.1M  -
tank/test@2012:12:18:51:2:19:15          0      -  37.1M  -
```

Destroying Snapshots

Just as you would destroy a storage pool, or a ZFS dataset, you use a similar method for destroying snapshots. To destroy a snapshot, use the "zfs destroy" command, and supply the snapshot as an argument that you want to destroy:

```
# zfs destroy tank/test@2012:12:18:51:2:19:15
```

An important thing to know, is if a snapshot exists, it's considered a child filesystem to the dataset. As such, you cannot remove a dataset until all snapshots, and nested datasets have been destroyed.

```
# zfs destroy tank/test
cannot destroy 'tank/test': filesystem has children
use '-r' to destroy the following datasets:
tank/test@2012:12:18:51:2:19:15
tank/test@2012:12:18:51:2:19:00
```

Destroying snapshots can free up additional space that other snapshots may be holding onto, because they are unique to those snapshots.

Renaming Snapshots

You can rename snapshots, however, they must be renamed in the storage pool and ZFS dataset from which they were created. Other than that, renaming snapshots is pretty straight forward:

```
# zfs rename tank/test@2012:12:18:51:2:19:15 tank/test@tuesday-19:15
```

Rolling Back to a Snapshot

A discussion about snapshots would not be complete without a discussion about rolling back your filesystem to a previous snapshot.

Rolling back to a previous snapshot will discard any data changes between that snapshot and the current time. Further, by default, you can only rollback to the most recent snapshot. In order to rollback to an earlier snapshot, you must destroy all snapshots between the current time and that snapshot you wish to rollback to. If that's not enough, the filesystem must be unmounted before the rollback can begin. This means downtime.

To rollback the "tank/test" dataset to the "tuesday" snapshot, we would issue:

```
# zfs rollback tank/test@tuesday
cannot rollback to 'tank/test@tuesday': more recent snapshots exist
use '-r' to force deletion of the following snapshots:
tank/test@wednesday
tank/test@thursday
```

As expected, we must remove the "@wednesday" and "@thursday" snapshots before we can rollback to the "@tuesday" snapshot.

ZFS Clones

A ZFS clone is a writeable filesystem that was "upgraded" from a snapshot. Clones can only be created from snapshots, and a dependency on the snapshot will remain as long as the clone exists. This means that you cannot destroy a snapshot, if you cloned it. The clone relies on the data that the snapshot gives it, to exist. You must destroy the clone before you can destroy the snapshot.

Creating clones is nearly instantaneous, just like snapshots, and initially does not take up any additional space. Instead, it occupies all the initial space of the snapshot. As data is modified in the clone, it begins to take up space separate from the snapshot.

Creating ZFS Clones

Creating a clone is done with the "zfs clone" command, the snapshot to clone, and the name of the new filesystem. The clone does not need to reside in the same dataset as the clone, but it does need to reside in the same storage pool. For example, if I wanted to clone the "tank/test@tuesday" snapshot, and give it the name of "tank/tuesday", I would run the following command:

```
# zfs clone tank/test@tuesday tank/tuesday
# dd if=/dev/zero of=/tank/tuesday/random.img bs=1M count=100
# zfs list -r tank
NAME           USED  AVAIL  REFER  MOUNTPOINT
tank           161M  2.78G  44.9K  /tank
tank/test     37.1M  2.78G  37.1M  /tank/test
tank/tuesday   124M  2.78G   161M  /tank/tuesday
```

Destroying Clones

As with destroying datasets or snapshots, we use the "zfs destroy" command. Again, you cannot destroy a snapshot until you destroy the clones. So, if we wanted to destroy the "tank/tuesday" clone:

```
# zfs destroy tank/tuesday
```

Just like you would with any other ZFS dataset.

Some Final Thoughts

Because keeping snapshots is very cheap, it's recommended to snapshot your datasets frequently. Sun Microsystems provided a Time Slider that was part of the GNOME Nautilus file manager. Time Slider keeps

snapshots in the following manner:

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

Unfortunately, Time Slider is not part of the standard GNOME desktop, so it's not available for GNU/Linux. However, the ZFS on Linux developers have created a "zfs-auto-snapshot" package that you can install from the project's PPA if running Ubuntu. If running another GNU/Linux operating system, you could easily write a Bash or Python script that mimics that functionality, and place it on your root's crontab.

Because both snapshots and clones are cheap, it's recommended that you take advantage of them. Clones can be useful to test deploying virtual machines, or development environments that are cloned from production environments. When finished, they can easily be destroyed, without affecting the parent dataset from which the snapshot was created.

Posted by Aaron Toponce on Wednesday, December 19, 2012, at 6:00 am. Filed under Debian, Linux, Ubuntu, ZFS. Follow any responses to this post with its comments RSS feed. You can post a comment or trackback from your blog. For IM, Email or Microblogs, here is the Shortlink.

{ 12 } Comments