

# BTRFS snapshots with snapper

Picked from an [article here](#).

Set up automatic snapshots of a BTRFS root subvolume, add these snapshots to the GRUB boot menu, and gain the ability to rollback an [Arch Linux](#) system to an earlier state.

## Let's go!

See "[A\(rch\) to Z\(ram\)](#)" for my step-by-step install of Arch, where I created:

- `@` subvolume, mounted to `/`. Create snapshots of this root subvolume.
- `@snapshots` and other subvolumes, which are *excluded* from root snapshots.

## 1. Install Snapper and snap-pac

**Snapper** is a tool for managing BTRFS snapshots. It can create and restore snapshots, and provides scheduled auto-snapping. **Snap-pac** provides a Pacman hook that uses Snapper to create `pre-` and `post-` BTRFS snapshots triggered by use of the package manager.

Install ...

```
$ sudo pacman -S snapper snap-pac
```

## 2. Create snapshot configuration for root subvolume

Snapper's `create-config` command assumes:

- Subvolume `@` already exists and is mounted at `/`.
- `/.snapshots` directory is *not* mounted and doesn't exist.

During my Arch install, I created the `@` and `@snapshots` subvolumes, and `/.snapshots` mountpoint. Before letting Snapper do its config thing, I need to move my earlier snapshot setup out of the way.

Unmount the subvolume and remove the mountpoint ...

```
$ sudo umount /.snapshots
$ sudo rm -rf /.snapshots
```

Create a new `root` config ...

```
$ sudo snapper -c root create-config /
```

This generates:

- Configuration file at `/etc/snapper/configs/root`.
- Add `root` to `SNAPPER_CONFIGS` in `/etc/conf.d/snapper`.
- Subvolume `.snapshots` where future snapshots for this configuration will be stored.

## 3. Setup `/.snapshots`

List subvolumes ...

```
$ sudo btrfs subvolume list /
ID 256 gen 199 top level 5 path @
ID 257 gen 186 top level 5 path @home
ID 258 gen 9 top level 5 path @snapshots
[...]
ID 265 gen 199 top level 256 path .snapshots
```

Note the `@snapshots` subvolume I had created earlier, and the `.snapshots` created by Snapper.

I prefer my `@snapshots` setup over `.snapshots`, so I delete the Snapper-generated subvolume ...

```
$ sudo btrfs subvolume delete .snapshots
Delete subvolume (no-commit): '//.snapshots'
```

Re-create and re-mount `/.snapshots` mountpoint ...

```
$ sudo mkdir /.snapshots
```

```
$ sudo mount -a
```

This setup will make all snapshots created by Snapper be stored outside of the `@` subvolume. This allows replacing `@` without losing the snapshots.

Set permissions. Owner must be `root`, and I allow members of `wheel` to browse through snapshots ...

```
$ sudo chmod 750 /.snapshots
```

```
$ sudo chown :wheel /.snapshots
```

## 4. Manual snapshot

Example of taking a manual snapshot of a fresh install ...

```
$ sudo snapper -c root create -d "***Base system install**"
```

## 5. Automatic timeline snapshots

Setup timed auto-snapshots by modifying `/etc/snapper/configs/root`.

Allow user (example: `foo`) to work with snapshots ...

```
ALLOW_USERS="foo"
```

Example: Set some timed snapshot limits ...

```
TIMELINE_MIN_AGE="1800"
```

```
TIMELINE_LIMIT_HOURLY="5"
```

```
TIMELINE_LIMIT_DAILY="7"
```

```
TIMELINE_LIMIT_WEEKLY="0"
```

```
TIMELINE_LIMIT_MONTHLY="0"
```

```
TIMELINE_LIMIT_YEARLY="0"
```

Start and enable `snapper-timeline.timer` to launch the automatic snapshot timeline, and `snapper-cleanup.timer` to periodically clean up older snapshots...

```
$ sudo systemctl enable --now snapper-timeline.timer
```

```
$ sudo systemctl enable --now snapper-cleanup.timer
```

## 6. Pacman snapshots

Pacman `pre-` and `post-` snapshots are triggered before and after a significant change (such as a system update).

Example: I install `tree`, which triggers a `pre` and `post` install snapshot.

List configs ...

```
$ snapper list-configs
```

```
Config | Subvolume
```

```
-----+-----
```

```
root | /
```

List snapshots taken for `root` ...

```
$ snapper -c root list
```

#	Type	Pre #	Date	User	Cleanup	Description	Userdata
0	single			root	current		
1	single		Sat 20 Aug 2022 11:21:53 AM	root		**Base system install**	
2	pre		Sat 20 Aug 2022 11:22:39 AM	root	number	pacman -S tree	
3	post	2	Sat 20 Aug 2022 11:22:39 AM	root	number	tree	
4	single		Sat 20 Aug 2022 12:00:04 PM	root	timeline	timeline	

List updated subvolumes list, which now includes the snapshots ...

```
$ sudo btrfs subvolume list /
```

```
ID 256 gen 270 top level 5 path @
```

```
ID 257 gen 270 top level 5 path @home
```

```
ID 258 gen 257 top level 5 path @snapshots
```

```
[...]
```

```
ID 266 gen 216 top level 258 path @snapshots/1/snapshot
```

```
ID 267 gen 218 top level 258 path @snapshots/2/snapshot
```

```
ID 268 gen 219 top level 258 path @snapshots/3/snapshot
```

```
ID 269 gen 237 top level 258 path @snapshots/4/snapshot
```

## 7. Updatedb

If `locate` is installed, skip indexing `.snapshots` directory by adding to `/etc/updatedb.conf` ...

```
PRUNENAMES = ".snapshots"
```

## 8. Grub-btrfs

Include the snapshots as boot options in the GRUB boot loader menu.

Install ...

```
$ sudo pacman -S grub-btrfs
```

Set the location of the directory containing the `grub.cfg` file in `/etc/default/grub-btrfs/config`.

Example: My `grub.cfg` is located in `/efi/grub` ...

```
GRUB_BTRFS_GRUB_DIRNAME="/efi/grub"
```

## 9. Auto-update GRUB

Enable `grub-btrfs.path` to auto-regenerate `grub-btrfs.cfg` whenever a modification appears in `/.snapshots` ...

```
$ sudo systemctl enable --now grub-btrfs.path
```

At the next boot, there is an submenu in GRUB for `Arch Linux snapshots`.

## 10. Read-only snapshots and overlays

Booting on a snapshot is done in *read-only* mode.

This can be tricky:

An elegant way is to boot this snapshot using **overlayfs** ... Using overlayfs, the booted snapshot will behave like a live-cd in non-persistent mode. The snapshot will not be modified, the system will be able to boot correctly, because a writeable folder will be included in the RAM ... Any changes in this system thus started will be lost when the system is rebooted/shutdown.

Add the hook `grub-btrfs-overlayfs` at the end of `HOOKS` in `/etc/mkinitcpio.conf` ...

```
HOOKS=(base ... fsck grub-btrfs-overlayfs)
```

Re-generate initramfs ...

```
$ sudo mkinitcpio -P
```

Note: Any snapshots that do not include this modified initramfs will not be able to use overlayfs.

## 11. System rollback the 'Arch Way'

Snapper includes a rollback tool, but on Arch systems the preferred method is a manual rollback.

After booting into a snapshot mounted `rw` courtesy of overlayfs, mount the toplevel subvolume (subvol=5). That is, omit any subvolid or subvol mount flags (*example*: an encrypted device map labelled `cryptdev`) ...

```
$ sudo mount /dev/mapper/cryptdev /mnt
```

Move the broken `@` subvolume out of the way ...

```
$ sudo mv /mnt/@ /mnt/@.broken
```

Or simply delete the subvolume ...

```
$ sudo btrfs subvolume delete /mnt/@
```

Find the number of the snapshot that you want to recover ...

```
$ sudo grep -r '<date>' /mnt/@snapshots/*/info.xml
[...]  
/.snapshots/8/info.xml: <date>2022-08-20 15:21:53</date>  
/.snapshots/9/info.xml: <date>2022-08-20 15:22:39</date>
```

Create a read-write snapshot of the read-only snapshot taken by Snapper ...

```
$ sudo btrfs subvolume snapshot /mnt/@snapshots/number/snapshot /mnt/@
```

Where `number` is the snapshot you wish to restore as the new `@`.

Unmount `/mnt`.

Reboot and rollback!

---

Revision #3

Created 21 September 2024 20:21:40 by Thorgan

Updated 21 September 2024 20:26:59 by Thorgan