ZFS Pools and snapshots

START HERE

`zpool status`
Is the pool healthy?

`zfs snapshot <ds>@<snap>`
Freeze a point in time

`zpool list`
Pool size and usage

`zfs rollback <ds>@<snap>`
Go back to a snapshot

`zfs list`
Show datasets

`zfs send <snap> | zfs recv <ds>`
Copy a snapshot

`zfs create <pool>/<name>`
New dataset

`zpool scrub <pool>`
Check data integrity

Disks vdevs: mirror, raidz → `zpool create` → **Pool** shared storage → `zfs create` → **Dataset** filesystem or zvol → `zfs snapshot` → **Snapshot** read-only, cheap → `zfs clone / send` → **Clone / stream** writable copy or backup

CONCEPTS

Two commands: zpool manages disks and pools, zfs manages datasets (filesystems, volumes, snapshots) inside them.

Datasets are cheap and inherit properties from their parent: make one per thing you want to snapshot, quota or tune separately.

Redundancy lives in the vdev: losing one non-redundant top-level vdev loses the whole pool, and a raidz vdev cannot simply be removed later.

POOLS

`zpool status`
Health, errors and scrub state

`zpool status -x`
Show only unhealthy pools

`zpool list`
Size, used, free and health per pool

`zpool create <pool> mirror <disk1> <disk2>`
Create a mirrored pool
Use stable /dev/disk/by-id paths. Destroys existing data on the disks.

`zpool create <pool> raidz <disk1> <disk2> <disk3>`
Create a raidz pool

`zpool create -o ashift=12 <pool> <disk>`
Force 4K sector alignment
ashift is fixed per vdev at creation time.

`zpool scrub <pool>`
Verify all checksums, repair from redundancy

`zpool scrub -p <pool>`
Pause a running scrub

`zpool import`
List pools available to import

`zpool import <pool>`
Import a pool

`zpool import -d /dev/disk/by-id <pool>`
Import searching a given directory

`zpool export <pool>`
Cleanly detach a pool

`zpool iostat -v <pool> 5`
Per-vdev I/O every 5 seconds

`zpool history <pool>`
Show commands run against the pool

POOLS (cont.)

`zpool clear <pool>`
Reset error counters after a fault

`zpool replace <pool> <old> <new>`
Swap a failed or old disk

`zpool offline <pool> <disk>`
Take a disk offline

`zpool attach <pool> <disk> <new>`
Add a mirror side to a disk

`zpool add <pool> mirror <d1> <d2>`
Add a top-level vdev
Permanent for raidz pools. Preview with -n first.

`zpool add -n <pool> <vdev> <disks>`
Dry-run adding a vdev

`zpool destroy <pool>`
Destroy a pool

DATASETS

`zfs list`
List filesystems and volumes

`zfs list -r <pool>`
List a pool's datasets recursively

`zfs create <pool>/<name>`
Create a filesystem

`zfs create -p <pool>/<a>/<b>`
Create with missing parents

`zfs create -V <size> <pool>/<vol>`
Create a block volume (zvol)

`zfs rename <old> <new>`
Rename a dataset or snapshot

`zfs destroy <dataset>`
Delete a dataset
Irreversible. Add -r to include children and snapshots.

`zfs destroy -rn -v <dataset>`
Dry-run a recursive destroy

`zfs mount`
List mounted datasets

`zfs mount -a`
Mount all datasets

`zfs unmount <dataset>`
Unmount a dataset

`zfs list -o space`
Space broken down by use

PROPERTIES

`zfs get all <dataset>`
Show every property

PROPERTIES (cont.)

`zfs get <prop> <dataset>`
Show one property

`zfs get -r -s local all <pool>`
Show locally set properties only

`zfs set compression=lz4 <dataset>`
Enable compression
Applies to newly written data only.

`zfs set mountpoint=<path> <dataset>`
Change where it mounts

`zfs set quota=<size> <dataset>`
Cap a dataset's size

`zfs set reservation=<size> <dataset>`
Guarantee space for a dataset

`zfs set atime=off <dataset>`
Skip access-time writes

`zfs set recordsize=<size> <dataset>`
Tune block size (databases, media)

`zfs set readonly=on <dataset>`
Make a dataset read-only

`zfs inherit <prop> <dataset>`
Drop local value, inherit again

SNAPSHOTS AND CLONES

`zfs snapshot <dataset>@<name>`
Create a snapshot

`zfs snapshot -r <pool>@<name>`
Snapshot a pool and all children

`zfs list -t snapshot`
List snapshots

`zfs list -t snapshot -o name,used,creation -s creation`
Snapshots oldest first

`zfs rollback <dataset>@<snap>`
Revert to a snapshot
Discards changes since the snapshot. -r also destroys newer snapshots.

`zfs diff <dataset>@<snap>`
Files changed since the snapshot

`zfs destroy <dataset>@<snap>`
Delete a snapshot

SNAPSHOTS AND CLONES (cont.)

`zfs clone <dataset>@<snap> <pool>/<new>`
Writable copy of a snapshot

Browse old versions without rolling back: read-only copies sit under <mountpoint>/.zfs/snapshot/<name>/.

SEND AND RECEIVE

`zfs send <dataset>@<snap> | zfs recv <pool>/<name>`
Copy a snapshot locally

`zfs send <dataset>@<snap> | ssh <host> zfs recv <pool>/<name>`
Replicate to another machine

`zfs send -i <old> <new> | zfs recv <ds>`
Send only changes between snapshots

`zfs send -R <pool>@<snap> | zfs recv -d <pool2>`
Replicate a whole tree with properties

`zfs send -w <dataset>@<snap>`
Send encrypted data as-is (raw)

`zfs send -nv <dataset>@<snap>`
Dry run with size estimate

ENCRYPTION

`zfs create -o encryption=on -o keyformat=passphrase <pool>/<name>`
Create an encrypted dataset
Encryption can only be set at creation.

`zfs load-key <dataset>`
Unlock a dataset

RECIPES

Mirrored pool with a compressed dataset

```
zpool create -o ashift=12 tank
mirror /dev/disk/by-id/<a> /
dev/disk/by-id/<b>
zfs set compression=lz4
atime=off tank
zfs create tank/data
zpool status tank
```

Incremental backup over ssh

```
zfs snapshot tank/data@mon
zfs send tank/data@mon | ssh
<host> zfs recv -u backup/data
zfs snapshot tank/data@tue
zfs send -i @mon tank/data@tue
| ssh <host> zfs recv -u
backup/data
```