The Storage Hygiene Playbook

A procedure for reclaiming disk space on a Mac without buying anything.

Companion to *The Drive I Didn't Need* · wazirigaruba.com/the-drive

---

Before you start: three things that fail silently

Every one of these reports success while doing nothing. Check them first or you will waste an afternoon chasing numbers that were never real.

1. Local snapshots preserve everything you delete

macOS takes an hourly local Time Machine snapshot, and a snapshot's job is to remember — including every file you just removed. Free space will not move, and can *fall*, while you delete.

```
tmutil listlocalsnapshots /
```

If anything is listed, your deletions are being held. Purge them last, after all deleting is done:

```
for s in $(tmutil listlocalsnapshots / | grep TimeMachine | sed 's/com.apple.TimeMachine.
```

Paste that into **Terminal.app directly**. It needs a real password prompt, which most automation environments cannot provide. It is slow — a blinking cursor is progress, not a hang.

These are *not* your external Time Machine backups. Those are untouched.

2. macOS can silently revoke folder access

If your terminal loses permission to read Desktop or Documents, scans do not fail — they return *empty*, which reads exactly like "nothing found." A folder count can read 2 when the truth is 40.

```
ls ~/Desktop >/dev/null && echo OK || echo BLOCKED
ls ~/Documents >/dev/null && echo OK || echo BLOCKED
```

If blocked:

```
tccutil reset SystemPolicyDesktopFolder com.apple.Terminal
tccutil reset SystemPolicyDocumentsFolder com.apple.Terminal
```

Then restart the terminal. Do not try to force it with `sudo`.

3. Verify writes actually landed

While access is blocked, some tools report a successful save for a file that was never created. Confirm with `stat`, not by asking the tool that wrote it.

Step 1 — Caches and build artifacts

Nothing here is yours. All of it regenerates.

```
npm cache clean --force
pnpm store prune
brew cleanup --prune=all
rm -rf ~/Library/Caches/*
rm -rf ~/.Trash/*
```

Dependency folders. If you write code, this is usually the largest single category. Exempt anything you might need to run on short notice; everything else costs one reinstall.

```
EXEMPT="project-i-still-need|another-live-one"
find ~/code -maxdepth 4 -type d -name node_modules -prune | grep -vE "$EXEMPT" | while re
```

Docker. Its virtual disk is a single enormous file that never shrinks on its own. Check `~/Library/Containers/com.docker.docker/Data/vms/0/data/Docker.raw`. Mine was 14 GB and had been idle for four months.

Old model weights, simulators, DerivedData. Anything downloaded once for an experiment you finished.

This step alone typically returns more than everything below it combined.

Step 2 — The question that beats archiving

Before moving any file to an external drive, ask whether a copy already exists in your cloud storage.

Most people upload things and forget. Recordings, exports, client deliverables — uploaded the day they were made, then duplicated locally forever.

Find the big ones:

```
find ~/Desktop ~/Documents ~/Downloads ~/Movies -type f \
  \( -iname '*.mp4' -o -iname '*.mov' -o -iname '*.m4v' \) -size +250000k \
  -not -path '*/node_modules/*' -print0 | xargs -0 stat -f '%z%t%N' | sort -rn
```

Then search your cloud storage for each name and **compare the exact byte count.** Not "about the same size" — the exact number. An exact match under your own account is the bar.

The pattern to look for: one large raw recording surrounded by many small finished cuts. The value was already extracted; you are storing the quarry after the statue is done. Video call recordings are the worst offenders — cloud services often keep several renditions plus audio and a transcript while the raw sits on your disk.

Log before you delete. One line per file: path, byte count, where the copy lives, and a direct link to it. Write it before removing anything, so an interrupted cleanup still leaves a record. This is what makes deletion reversible, and it is the difference between confidence and hoping.

Step 3 — Judgment calls

Item	Call
Spotlight index (`~/Library/Metadata`)	**Skip.** Reindexing costs more than it returns.
Toolchains for hardware you still use	**Trim, never delete.** Remove old board or SDK versions only.

Item	Call
Downloaded AI model weights	**Delete.** Re-downloadable, and usually forgotten.
Anything you cannot name the purpose of	**Leave it.** Come back when you can.

Never delete

- Time Machine backup volumes
- System photo libraries
- Any credential, key, or environment file
- **Any photo or video without a verified second copy**

That last one is the only irreversible category on the list. Flag them in a list instead, with paths and sizes, and deal with them deliberately.

Before you copy anything to a drive

Check what the drive actually is:

```
diskutil info /Volumes/YOUR-DRIVE | grep -Ei 'file system|uuid'
```

If it says FAT32, it cannot hold a file larger than 4 GB. Some tools fail silently at that boundary rather than erroring. Reformat to exFAT — after moving whatever is already on it somewhere safe.

Identify drives by their **volume UUID**, not by name. macOS appends suffixes when names collide, so a second drive can silently make every path you recorded point somewhere wrong.

Closing out

1. Purge snapshots again — anything deleted after your first purge is pinned by a new one.
2. Write down what you removed and what you deliberately kept.

3. Note the number you finished at, so next time you know what normal looks like.

The part that isn't a command

I got to 78 GB reclaimed without moving a single file to external storage, and without buying anything. The tooling mattered less than the sequence, and the sequence came from being asked questions before executing — *what is this for, do you already have it, does it need to move at all.*

A drive in a drawer has no redundancy and fails silently. Verify a second copy exists, then delete. Do not relocate by reflex.
