Git stash recovery

GIT

When a stash goes missing

`git stash drop` removes a stash entry from the reflog but does **not** garbage-collect the underlying commit immediately. As long as you act before the next `git gc`, the stash is recoverable.

Find dangling stash commits

```
git fsck --unreachable --no-reflogs | grep commit
```

This lists every commit that isn't pointed at by any ref. Stashes are commits, so a dropped stash shows up here.

For each candidate SHA, peek at the message:

```
git show --stat <sha>
```

Stash commits have messages like `WIP on main: ...` or `On branch foo: ...`. That's the one.

Re-apply the recovered stash

Once you've found the SHA:

```
git stash apply <sha>
```

Or to put it back into the stash list with its original metadata:

```
git update-ref --create-reflog refs/stash <sha>
```

How long do you have?

Default `git gc` runs every ~14 days OR when loose-object count crosses a threshold. So practically:

TRIGGER	LIKELIHOOD
Immediate (within minutes)	Safe — fsck finds it
Same day	Almost always safe
Days to weeks	Usually safe
After explicit `git gc --prune=now`	Gone forever

The "I already ran gc" case

If `git fsck` shows nothing and you ran an explicit prune: there's nothing left to recover from your local clone. Last hopes:

- Your editor or IDE might keep file backups (`.swp`, `.autosave`, JetBrains local history)
- macOS Time Machine / Windows File History / etc. may have a recent file snapshot
- If you'd pushed to a remote at any point: check the remote's reflog and dangling refs

Prevention

```
# Soft alias — never drop without showing what you'd lose first
git config --global alias.stash-drop '!f() { git stash show -p "$@" && read -p "Drop this stash? [y/N] " ok && [
```

> Stashes are commits. Commits don't disappear unless you tell git to forget them.