

Git Internals, Content-Addressable DAG & CI/CD Release Engineering

Blobs, trees, commits, rebase vs merge, interactive bisect, reflog recovery, and immutable artifact pipelines.

EXECUTIVE SUMMARY

Under-the-hood reference for Git's Merkle-DAG object model (`.git/objects`), surgical history recovery with `git reflog`, automated regression hunting with `git bisect run`, and trunk-based CI/CD.

KEY EQUATIONS & INVARIANTS

* Git Object Content-Addressed Hash:

LaTeX:
$$\text{OID} = \text{SHA-1}(\text{type} \parallel \text{' '} \parallel \text{size} \parallel \text{'\textbackslash 0'} \parallel \text{payload})$$

Note: Every blob, tree, commit, and annotated tag is immutably addressed by the hash of its header + content.

1. Git Object Model & Emergency Recovery Commands

Because Git commits are immutable snapshots in a Directed Acyclic Graph (DAG), a 'hard reset' or botched rebase never immediately deletes committed data--it only moves branch pointer refs (`HEAD`).

Scenario | Command | Mechanism

Recover lost commit after git reset --hard | `git reflog && git reset --hard HEAD@{2}`
 | Reads local reference transaction log
 Find exact commit that broke tests | `git bisect start HEAD v1.2.0 && git bisect run pytest` | Binary searches commit DAG in $O(\log N)$ steps
 Safe force-push on rebased feature branch | `git push --force-with-lease` | Rejects push if remote ref advanced since last fetch
 Cleanly undo a public commit on main | `git revert -m 1 <commit-sha>` | Appends inverse diff commit without rewriting history

COMMON PRODUCTION & INTERVIEW PITFALLS

- Running `git push --force` on a shared branch overwrites teammates' commits; always use `--force-with-lease`.

FREQUENTLY ASKED QUESTIONS

Q: How does git store files: as diffs or as full snapshots?

A: Logically, every Git commit points to a full tree snapshot of blobs (unchanged files share existing blob SHA pointers). Physically, `git gc` later delta-compresses similar blobs inside `.pack` files.