October 23, 2025

Jeevan

What we'll explore:

- Git internals and architecture
- Life lessons from Git's design principles

▓▓ Is Git Just This?

```
git add .
git commit -m "fix stuff"
git push
```

Is .git folder size same as the working directory files size?

▓▓ Git = Key-Value Store

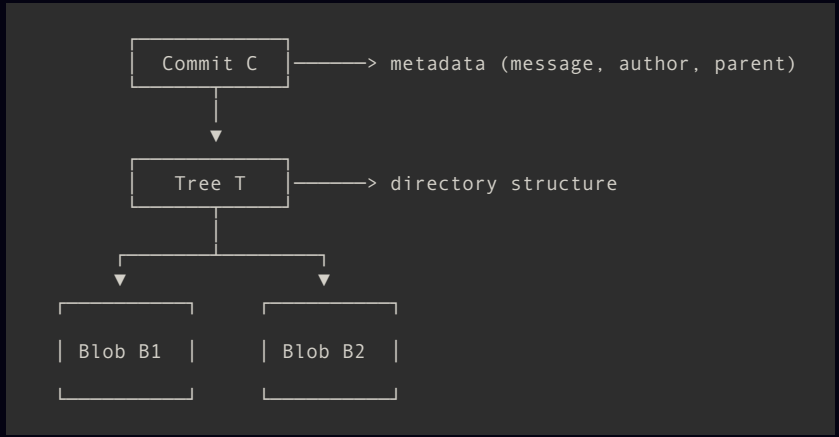
```
echo "hello world" | git hash-object -w --stdin
# Returns: 95d09f2b10159347eece71399a7e2e907ea3df4f

git cat-file -p 95d09f2b
# Returns: hello world
```

Your file content → SHA hash → stored forever To note: git doesn't store things by filenames but by hash

Three Building Blocks

- blob - file content
- tree - directory structure
- commit - snapshot + parent + metadata

Everything else is just pointers

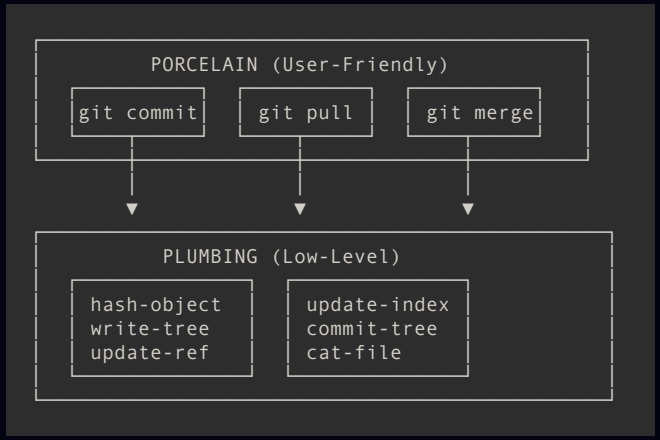


Visualize this stucture:

```
git rev-parse HEAD              # commit hash
git cat-file -p <commit-hash>   # shows tree + parent
git cat-file -p <tree-hash>     # shows file pointers
git cat-file -p <blob-hash>     # shows content
```

Plumbing vs Porcelain

A bit of story on why Plumbing vs Porcelain



Lets Make a Commit Without git commit - Plumbing

```
# 1. Create a blob
echo "Hello Plumbing" | git hash-object -w --stdin
# Returns: <blob-hash>

# 2. Add blob to staging area (index)
git update-index --add --cacheinfo 100644 <blob-hash> hello.txt
# 100644 = regular file permissions

# 3. Create tree from staging area
git write-tree
# Returns: <tree-hash>

# 4. Create commit pointing to tree
git commit-tree <tree-hash> -m "My first plumbing commit"
# Returns: <commit-hash>

# 5. Update branch to point to new commit
git update-ref refs/heads/master <commit-hash>
```

`.git/objects` = Your Database

Query it like any database:

```
git cat-file -p <hash>    # SELECT
git ls-tree <hash>        # LIST
git rev-parse HEAD        # RESOLVE
```

Structure:

```
.git/objects/
├── 95/
│   └── d09f2b10159347eece71399a7e2e907ea3df4f  (blob)
├── ab/
│   └── c123...  (tree)
└── de/
    └── f456...  (commit)
```

Why the directories are named with 2 chars? Why not full hash as file name instead?

When Did This Bug Appear?

```
git log -S 'replicas' --stat
```

Find when any code was added or removed

▓▓ Binary Search Through History

```
git bisect start
git bisect bad
git bisect good <commit>
```

Git finds the breaking commit in O(log n)

The Real Lessons

Git's Design Principles as Life Philosophies

Immutability = Integrity

Once something is done, don't rewrite history — learn from it.

- In Git: Commits are immutable; history is never changed, only built upon
- In Life: You can't erase the past, but you can branch from it, improve, and merge lessons forward

Hash-Based Identity = Authenticity

You are defined by your content, not your label or location.

- In Git: Identity comes from content, not filename or folder
- In Life: Your substance (values, actions, consistency) defines you, not your titles or affiliations

▓▓ DAG Structure = Purposeful Progress

Move forward with causality, no endless loops.

- In Git: Directed Acyclic Graph — no cycles, every commit has clear ancestry
- In Life: Make intentional, traceable progress — each action should build upon something before it

▓▓ Branching = Exploration

Experiment safely; failure is just another branch.

- In Git: Branching is cheap — explore ideas freely without breaking the main line
- In Life/Work: Separate experimentation from production. You can always merge the good ideas back

▒▒ Distributed Model = Trust

Everyone has the full copy — collaboration through autonomy.

- In Git: Every clone has the full history — no single point of failure
- In Teams: Empower individuals with full context, context over control

▒▒ Commits = Conscious Checkpoints

Pause, reflect, and capture meaningful progress.

- In Git: A commit is a conscious snapshot — message, state, intent
- In Life: Journaling, retrospectives, or milestones are your commits — record intent, not just action

▒▒ Remote Sync = Connection

Push when ready, pull to stay in sync.

- In Git: Collaboration happens through pushes and pulls
- In Life: Share learning with others, and learn from others

▒▒ Garbage Collection = Letting Go

Even Git prunes unreachable commits.

- In Git: Unused objects are eventually GC'd to save space
- In Life: Holding onto every version of yourself is unsustainable — prune what no longer matters, keep what builds the story

T H A N K Y O U

Questions? Thoughts?

Author: Jeevan

> "Commit small, commit often — in code and in life"