Demystifying Git:

The Content-Addressable Filesystem

Dipanka Tanu Sarmah

January 20, 2026

1 Introduction

Most developers treat Git as a set of magical commands to save code. However, at its core, Git is not just a version control system but a sophisticated content-addressable filesystem built upon a Directed Acyclic Graph (DAG). Understanding the "background code" requires shifting focus from commands like `commit` to the underlying objects stored in the `.git` directory.

2 The Core Architecture: The Object Database

Git stores data as objects. When you perform a `git add`, Git takes the content of your file, hashes it, and stores it in the `.git/objects` directory.

2.1 The SHA-1 Hashing Mechanism

Every piece of data is identified by a 40-character SHA-1 hash. This hash is generated based on the content plus a small header. The mathematical formula for a blob hash can be represented as:

$$\text{Hash} = \text{SHA1}(\text{"blob "} + \text{size} + \text{"\0"} + \text{content})$$

> **Custom Insight: Immutability**
> Because the ID of a file is derived from its content, Git objects are immutable. If you change a single comma in a file, the content changes, the SHA-1 hash changes, and a brand-new object is created. Git never "edits" a file in the database; it only ever adds new ones.

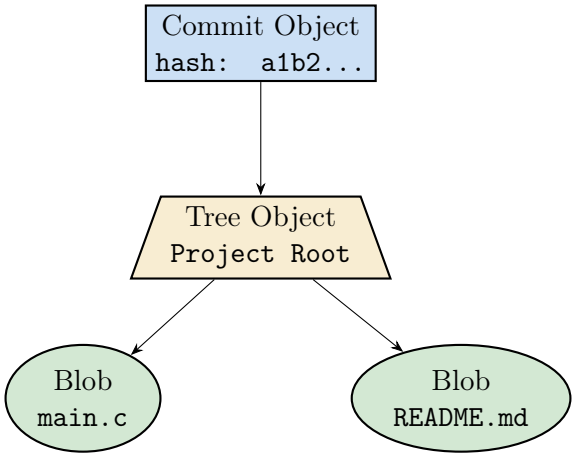
3 The Four Fundamental Objects

Git relies on four primary object types to reconstruct your project history:

1. **Blobs (Binary Large Objects):** These store only the file data. They do not store the filename or permissions.
2. **Trees:** These act like directories. A tree object contains a list of pointers (hashes) to blobs or other trees, along with their respective filenames.
3. **Commits:** A commit points to a specific root tree. It also contains the author, committer, timestamp, and a pointer to the parent commit(s).
4. **Tags:** A simple reference to a specific commit, often used for versioning.

4 Schematic: The Commit Graph

Below is a visual representation of how a commit object points to a tree, which in turn points to blobs (files).



Deep Dive: Delta Compression and Packfiles

While Git initially stores every version of a file as a full "loose" object, this would quickly consume gigabytes of space. To optimize this, Git uses **Packfiles**. During a process called garbage collection (`git gc`), Git searches for files with similar content across the history. It stores one version in full and then calculates the "deltas" (differences) for other versions. This allows Git to store thousands of versions of a text file in a fraction of the space.

5 The Staging Area: The Index File

The "Staging Area" is not a directory; it is a single binary file located at `.git/index`. This file acts as a map of the current working directory, listing every file path and the SHA-1 hash it should point to in the next commit. When you run `git add`, you are simply updating this map.

6 Conclusion

Git is built on the principle of simplicity and mathematical integrity. By using hashes to identify content, Git ensures that data cannot be corrupted or lost without the hash changing. It is a system of pointers where branches are just labels for commit hashes, and history is a chain of those hashes stretching back to the very first initialization.