Git uses three fundamental objects to represent the state and history of a repository: blobs, trees, and commits.

**Blob:**

A blob object stores the raw content of a file. When a file is added to the Git repository, its content is compressed and stored as a blob. Blobs are identified by their SHA-1 hash, which is generated from the content itself. If two files have identical content, they will share the same blob object, saving space.

**Tree:**

A tree object represents a directory or a subdirectory. It contains entries that map file names to blob objects (for files) and directory names to other tree objects (for subdirectories). This structure allows Git to represent the hierarchical organization of files and directories within a repository. Like blobs, tree objects are also identified by their SHA-1 hash.

**Commit:**

A commit object represents a snapshot of the entire repository at a specific point in time. It contains metadata about the commit, such as the author, committer, commit message, and timestamp. Crucially, each commit object also references a top-level tree object, which represents the complete directory structure and file contents of the repository at the time of that commit. Additionally, a commit object usually references one or more parent commit objects, forming the commit history and enabling tracking of changes over time.

In essence, when you perform a git commit, Git:

* Creates new blob objects for any modified or new files.
* Updates or creates tree objects to reflect the current directory structure and link to the relevant blob objects.
* Creates a new commit object that points to the top-level tree object representing the repository's state and links to its parent commit(s), thereby recording the snapshot and its place in the history.