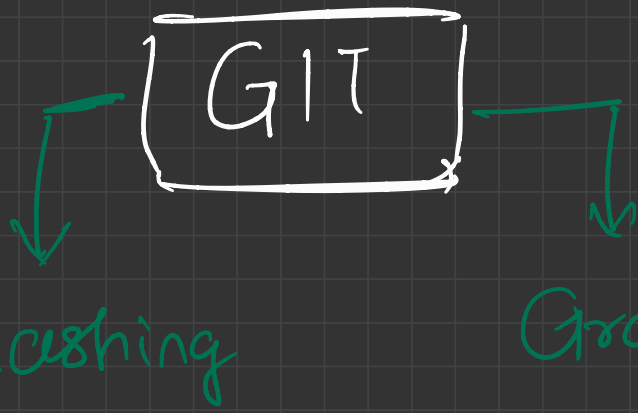
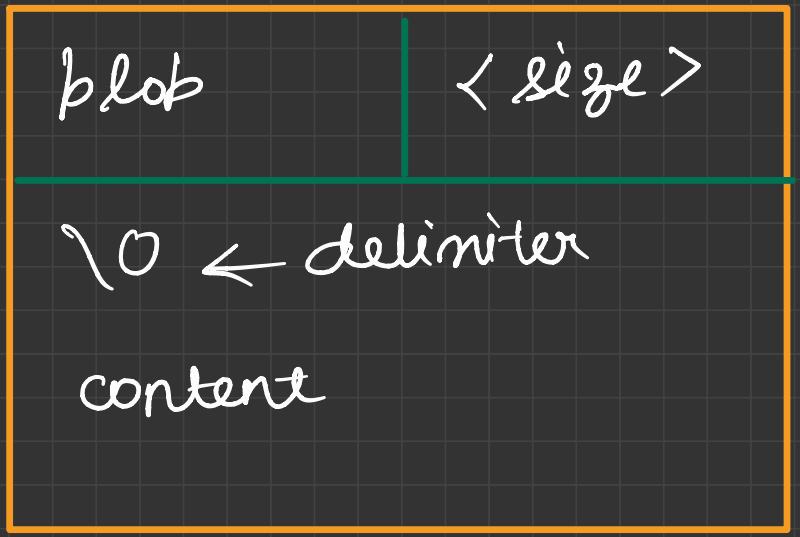
How GIT internally works?



At its core, GIT is like a key-value store. In the key, it stores the hash of the data. In the value, it stores the data.

GIT uses SHA1. (Cryptographic Algo.)
For a given data, it outputs 40 digit hexadecimal no.

GIT compresses data in a blob & stores some meta-data about it.

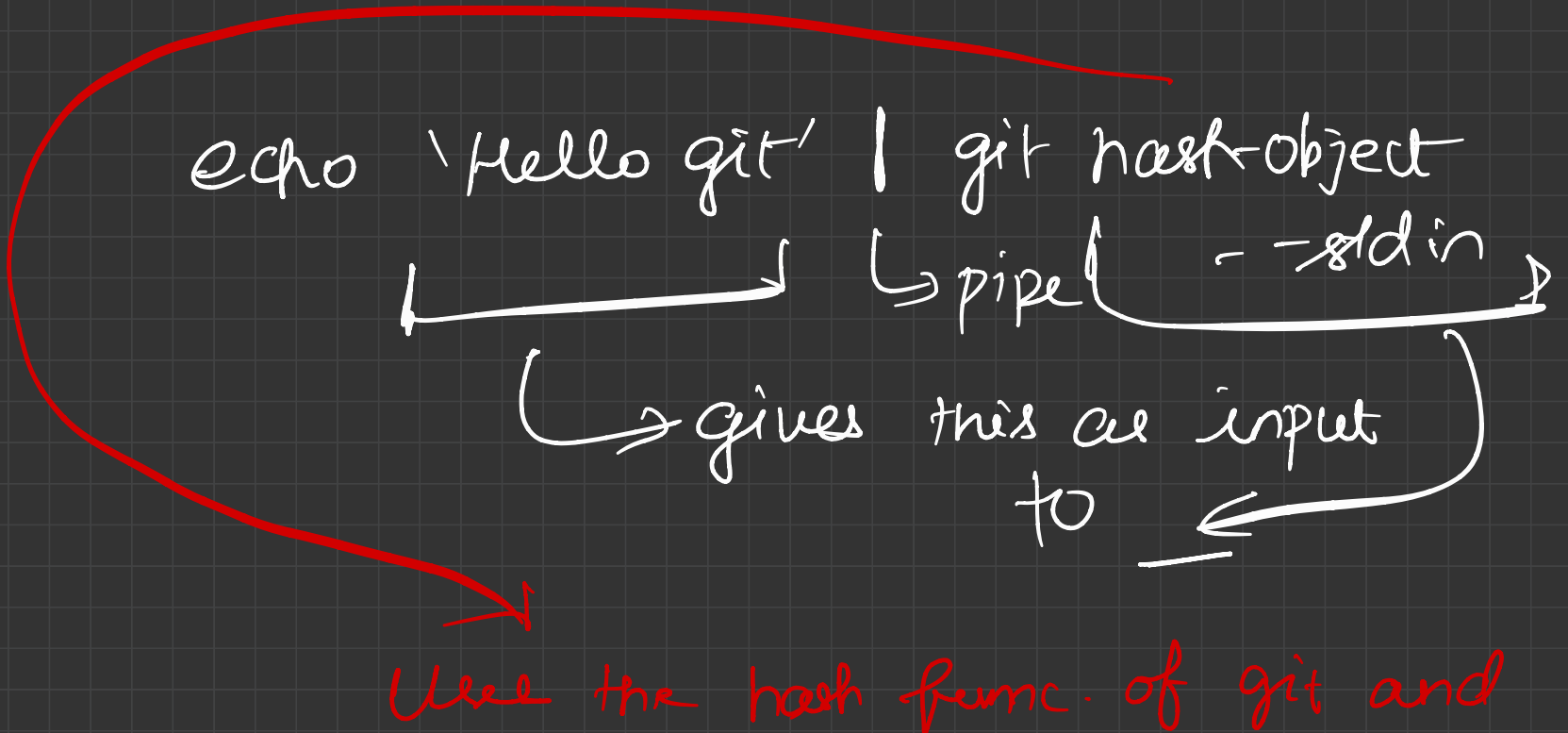


The value is wrapped inside a blob.

git init → Initializes the .git folder

tree .git → Visualize the whole git folder.

```
.git
├── HEAD
├── CONFIG
└── description
    :
    :
```



Uses the hash func. of git and takes input from stdin (terminal).

git add <some-file>

Under the objects folder in .git, it creates a directory.

The moment you did git add <some-file>, it created a 40-digit SHA-1 hash & a folder in the objects folder with first two digits from the SHA1 hash. Remaining digits → file name.

To check the contents of the file,

cat <file_name>

↳ you get a BLOB.

GIT provides a custom cat command.

git cat-file -p <whole-hexadecimal-hash>

Object

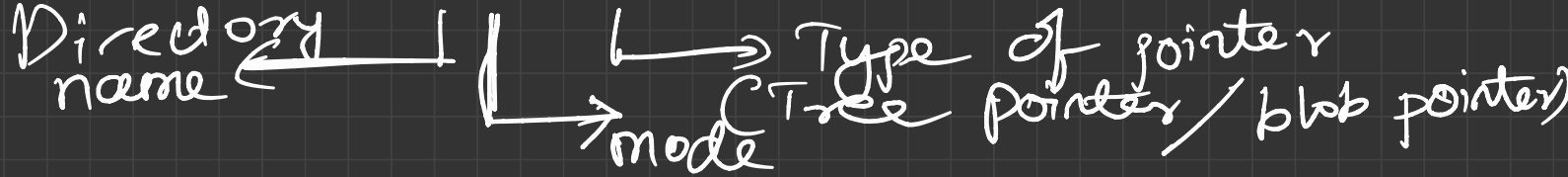
└ First two characters of Hash

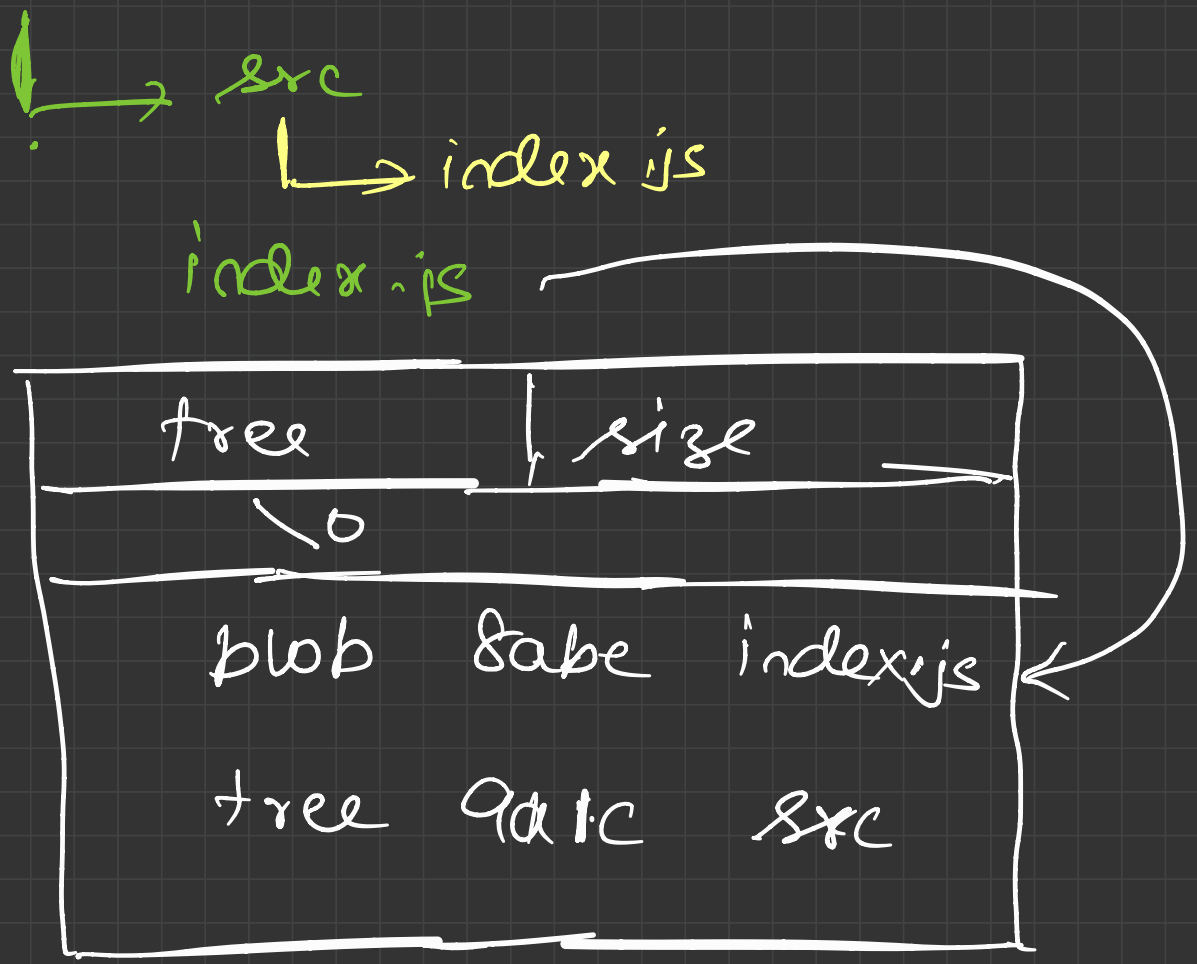
└ Name of file (Remaining 38 characters)

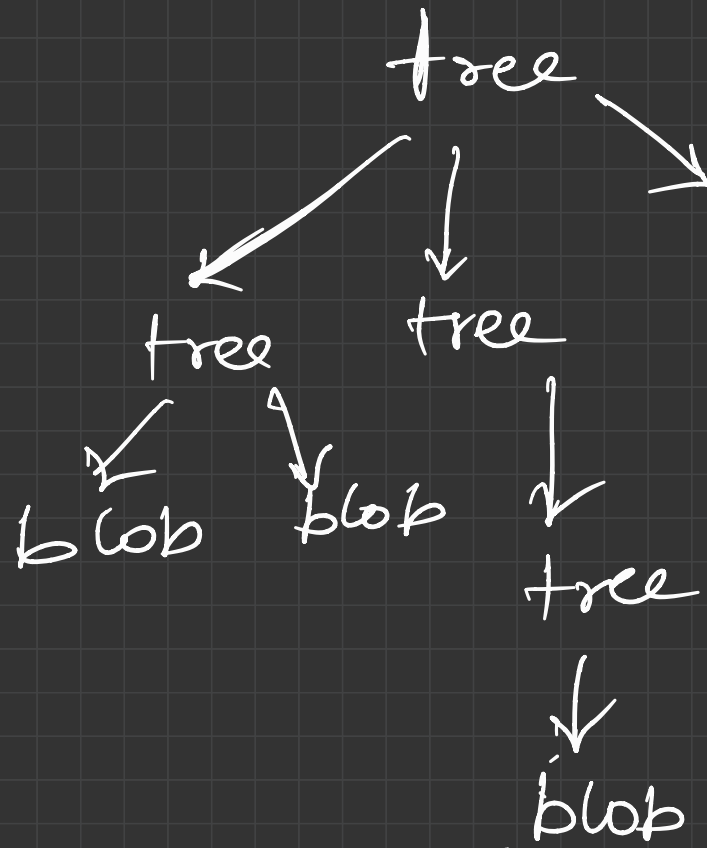
We are missing on details of filenames, directory details, sub-directory details etc.

tree → Directed tree

It contains pointers (using the hashes) to store blobs & store trees & it has meta-data.




src
index.js
index.js
tree
size
0
blob
&abe
index.js
tree
9d1c
src



git cat-file -t <hash_value>

↓

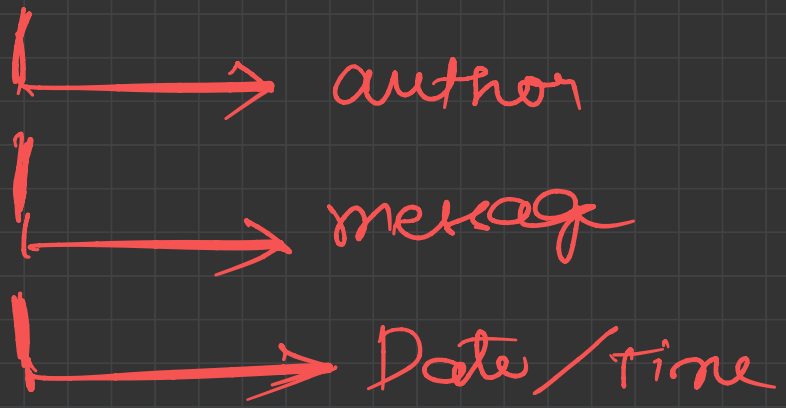
Prints the type of file

* TO manage directory, GIT uses tree.

Commit also gets stored as a hash.

What is actually a commit?

commit obj



You cannot have same piece of commit id for the same code change. This is due to the fact that commit has the timestamp property.

COMMIT ID WILL BE ALWAYS DIFFERENT.