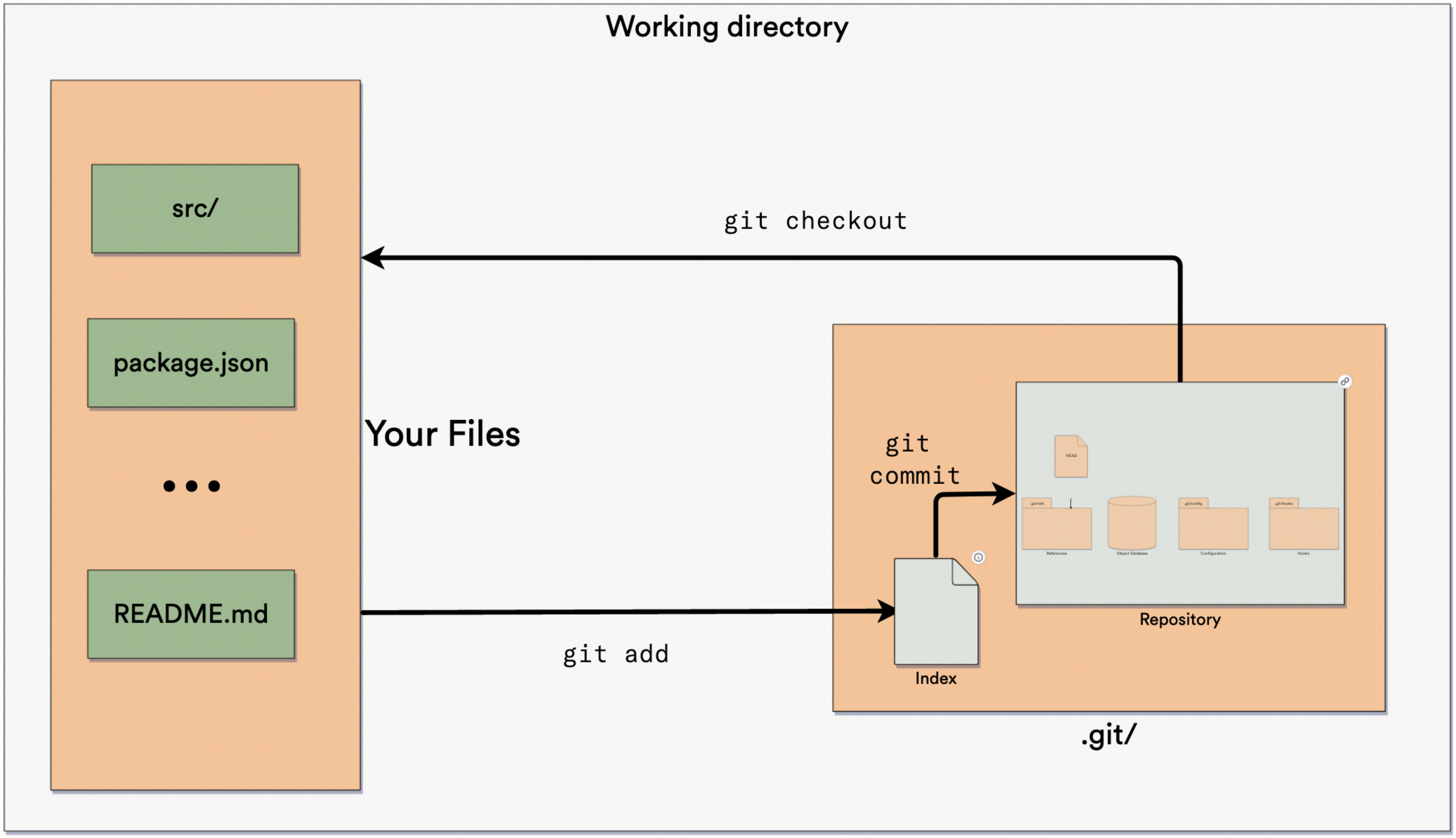
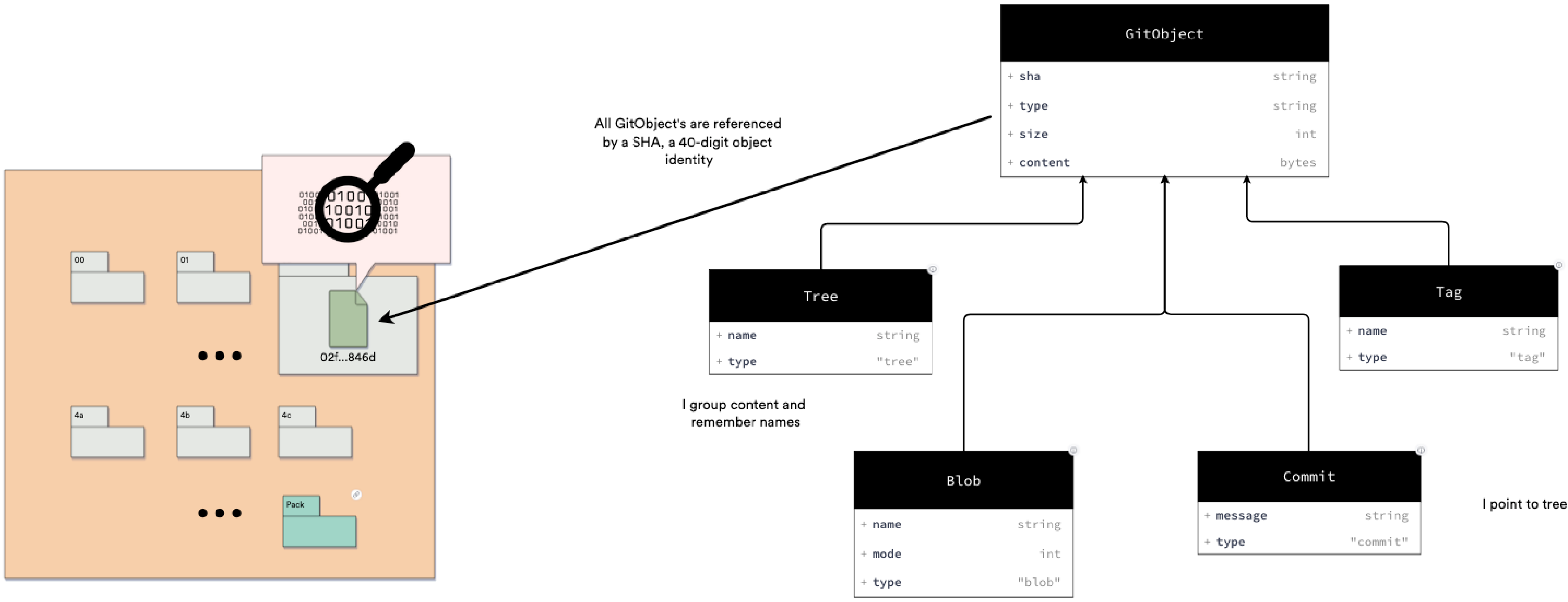
HOW DOES GIT WOKRS?

Git, created in 2005 by Linus Torvald, is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.



OBJECT DATABASE

Internally, all these files represent a GitObject, which can be one of four types



All GitObject's are referenced by a SHA, a 40-digit object identity

GitObject
+ sha string
+ type string
+ size int
+ content bytes

Tree
+ name string
+ type "tree"

I group content and remember names

Tag
+ name string
+ type "tag"

I add metadata

Blob
+ name string
+ mode int
+ type "blob"

I hold content

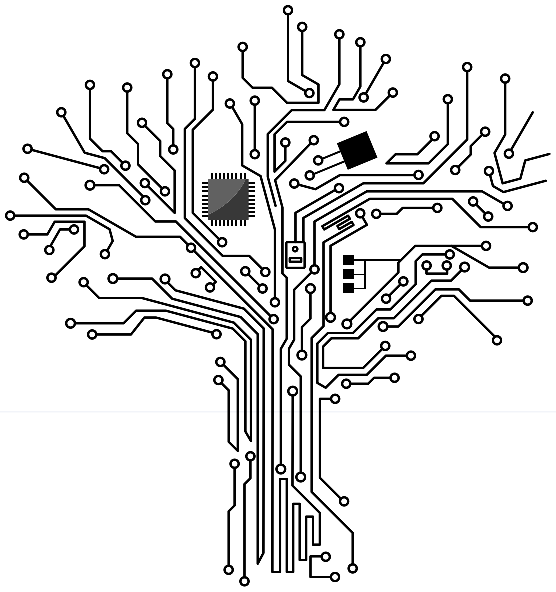
Commit
+ message string
+ type "commit"

I point to trees

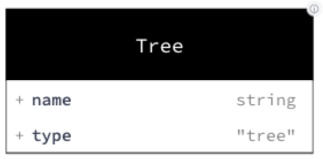
On your filesystem, the Object Database is arranged as a series of hexadecimal-ordered directories, each containing files of binary data.

PREMIUM PROGRAMMING COURSES

amigoscode.com

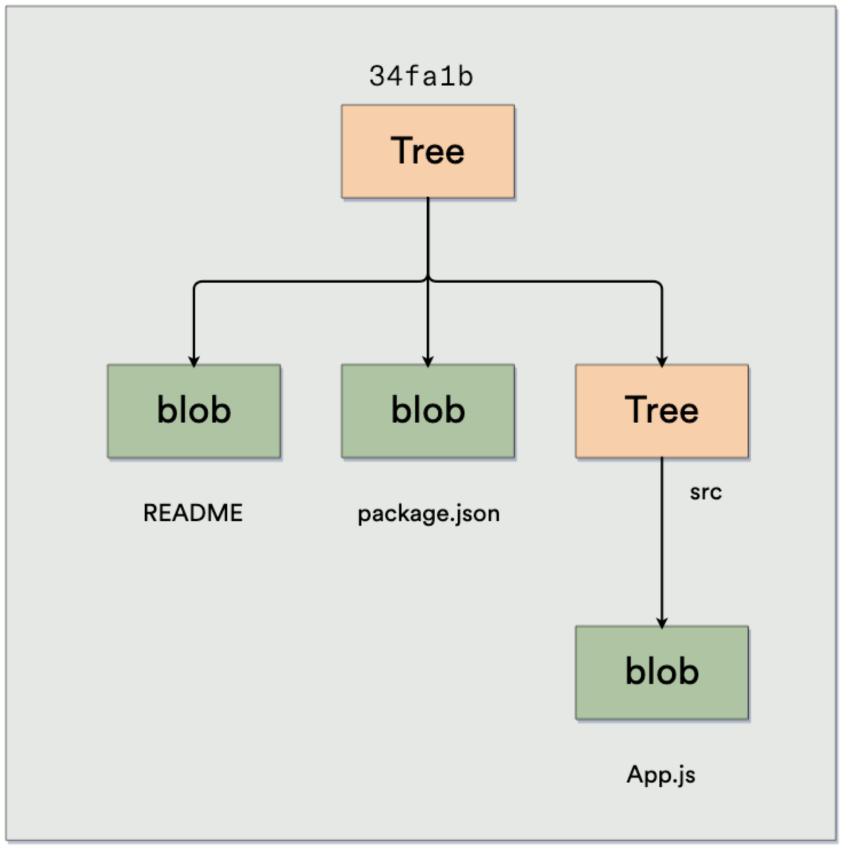


TREES

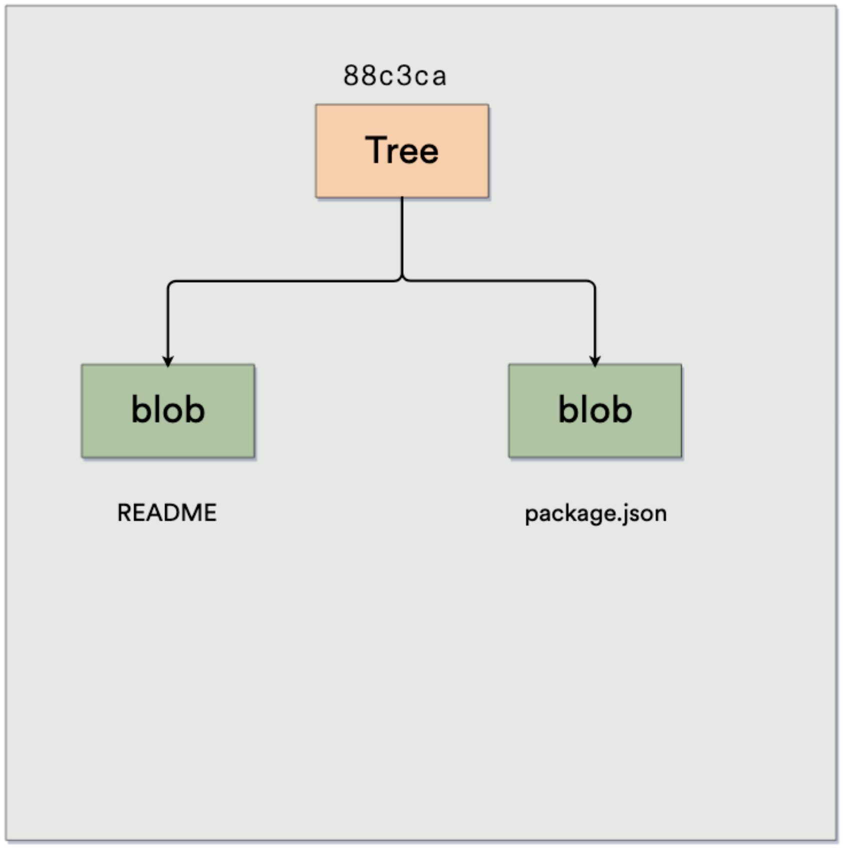


Trees group files together and stores filenames

Without trees, you would have to remember all the SHA's to know which blob to reference!



Snapshot of your files

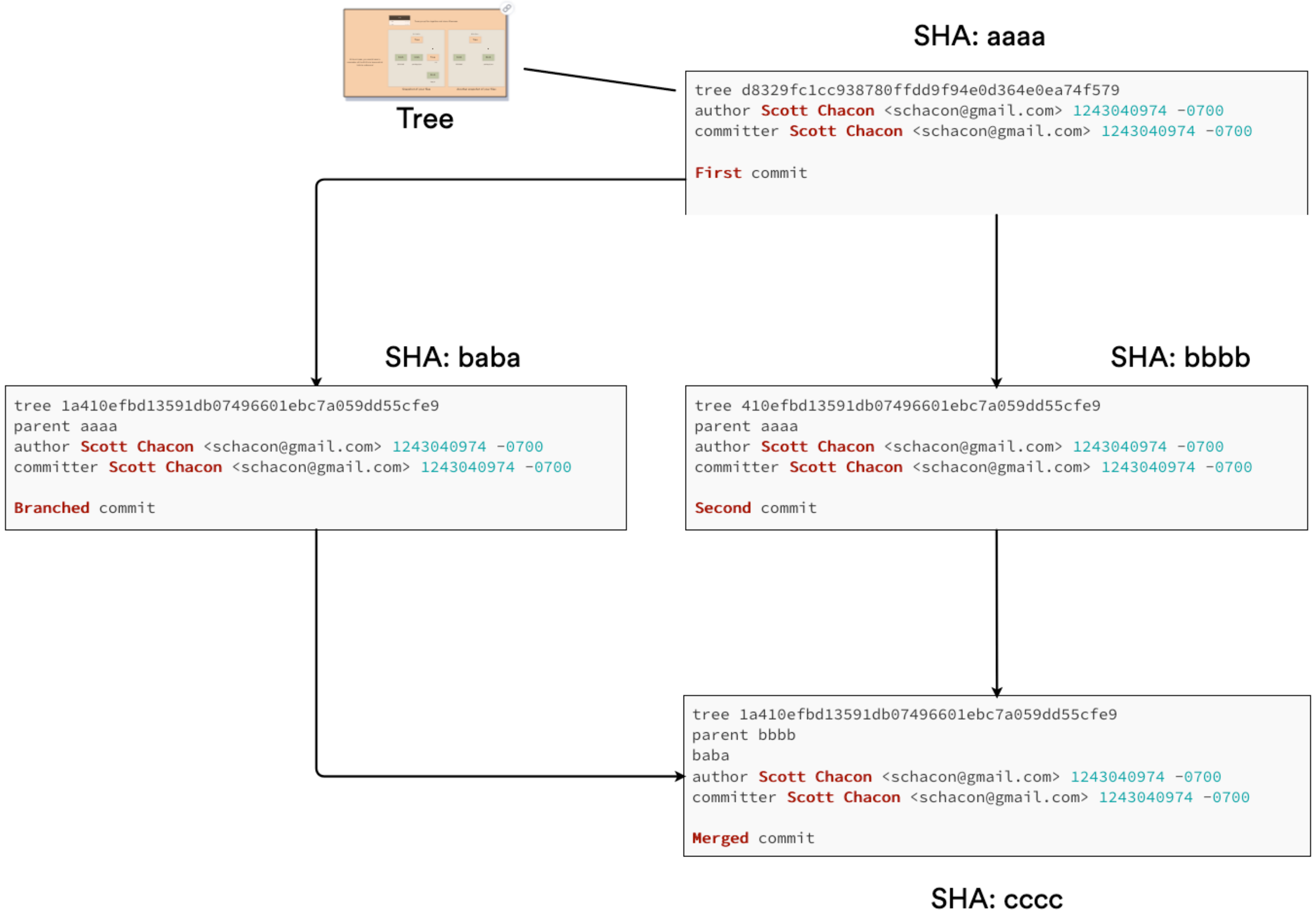


Another snapshot of your files

COMMITS

Commit	
+ message	string
+ type	"commit"

Commits give human readable names to trees



SHA: aaaa

```
tree d8329fc1cc938780ffdd9f94e0d364e0ea74f579
author Scott Chacon <schacon@gmail.com> 1243040974 -0700
committer Scott Chacon <schacon@gmail.com> 1243040974 -0700

First commit
```

SHA: baba

```
tree 1a410efbd13591db07496601ebc7a059dd55cfe9
parent aaaa
author Scott Chacon <schacon@gmail.com> 1243040974 -0700
committer Scott Chacon <schacon@gmail.com> 1243040974 -0700

Branched commit
```

SHA: bbbb

```
tree 410efbd13591db07496601ebc7a059dd55cfe9
parent aaaa
author Scott Chacon <schacon@gmail.com> 1243040974 -0700
committer Scott Chacon <schacon@gmail.com> 1243040974 -0700

Second commit
```

```
tree 1a410efbd13591db07496601ebc7a059dd55cfe9
parent bbbb
baba
author Scott Chacon <schacon@gmail.com> 1243040974 -0700
committer Scott Chacon <schacon@gmail.com> 1243040974 -0700

Merged commit
```

SHA: cccc

And answer questions like:
Who saved this snapshot?
When and why?

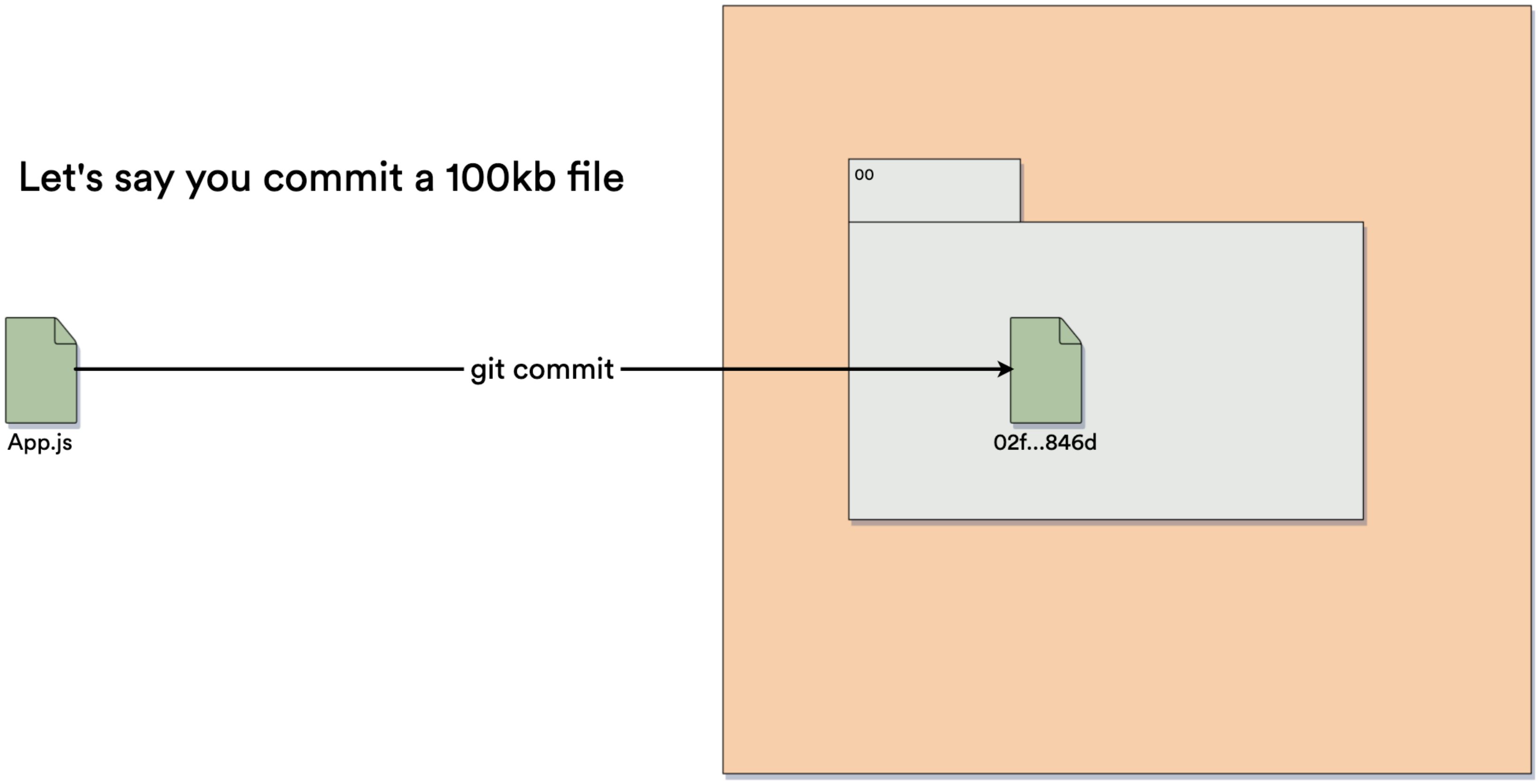
Commits can have parents. Those with 1 are ordinary commits.

But they can also have 2: merge commits.

COMPRESSION #1

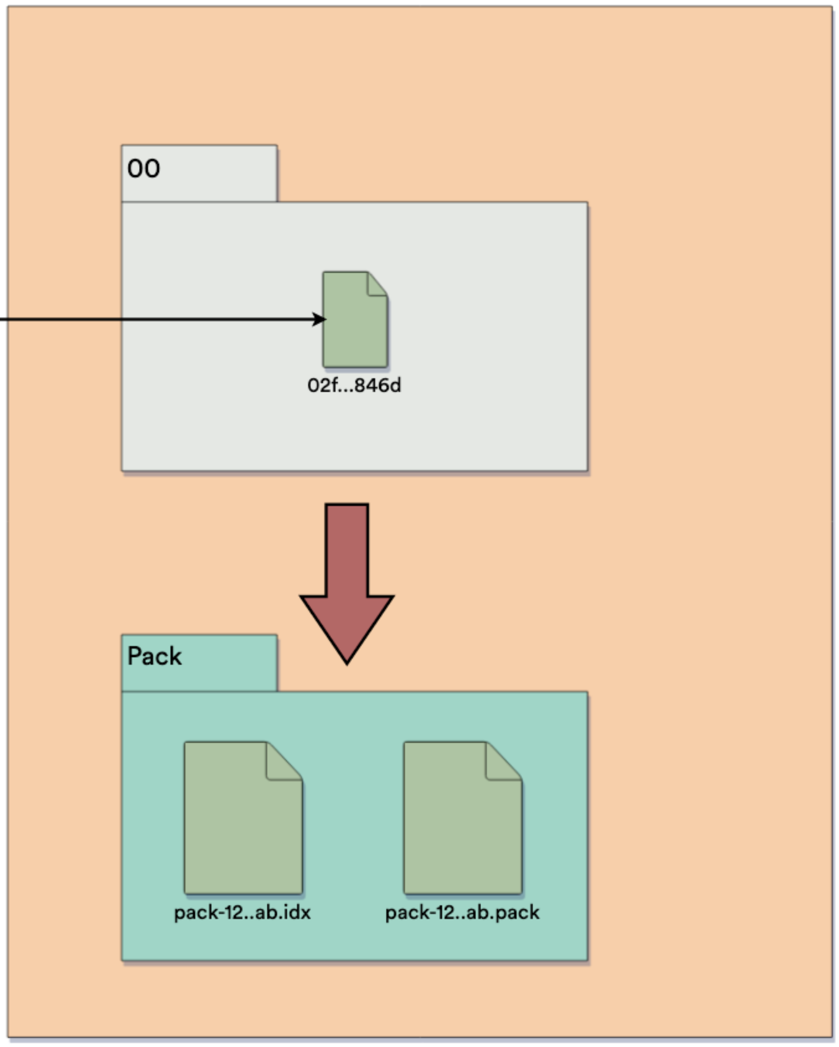
How does all the previous versions of my files fit into a Git repo??

Let's say you commit a 100kb file



COMPRESSION #2

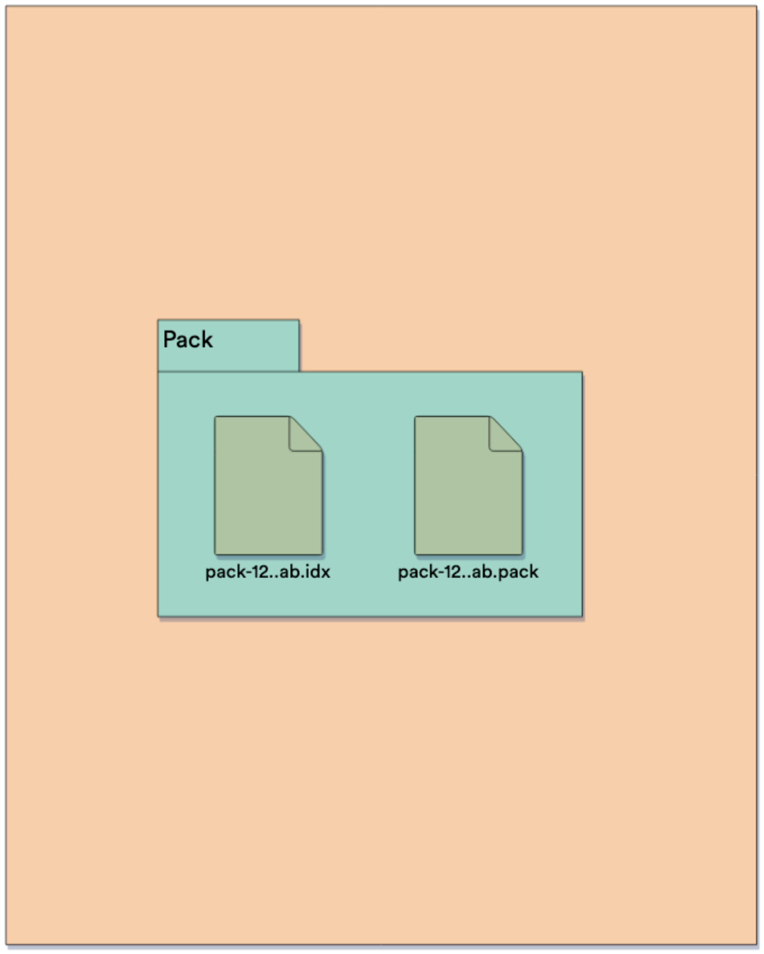
This is considered a "loose object".



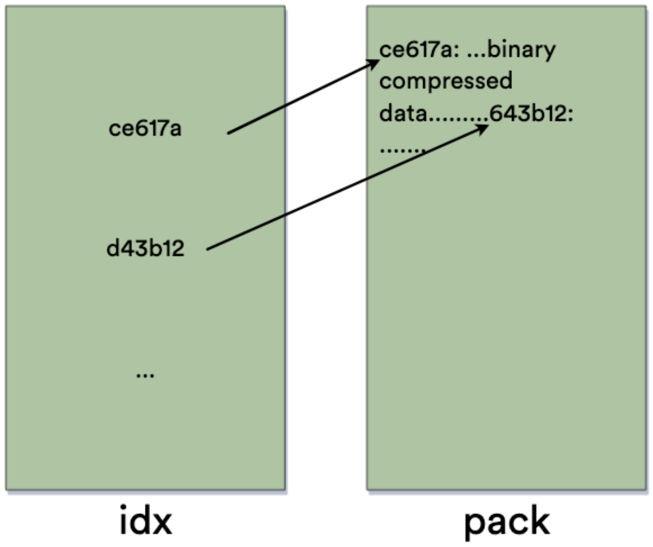
When you push to a remote server (or just have too many loose objects lying around), Git "packs" these.

COMPRESSION #3

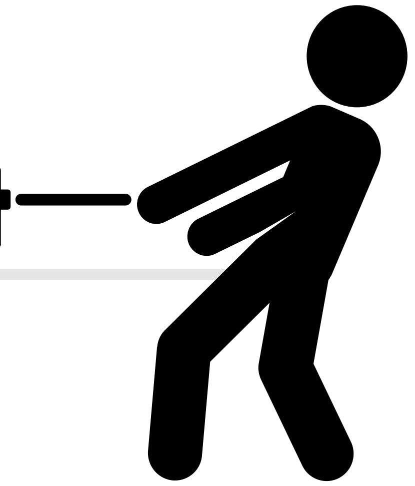
The loose object is then replaced with compressed data in a "pack" file.



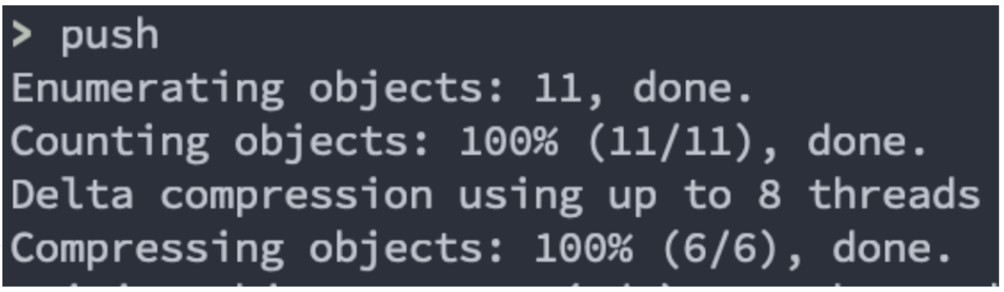
A pack can hold multiple objects, and the idx file stores the offsets to allow efficiently seeking the data.



COMPRESSION #4

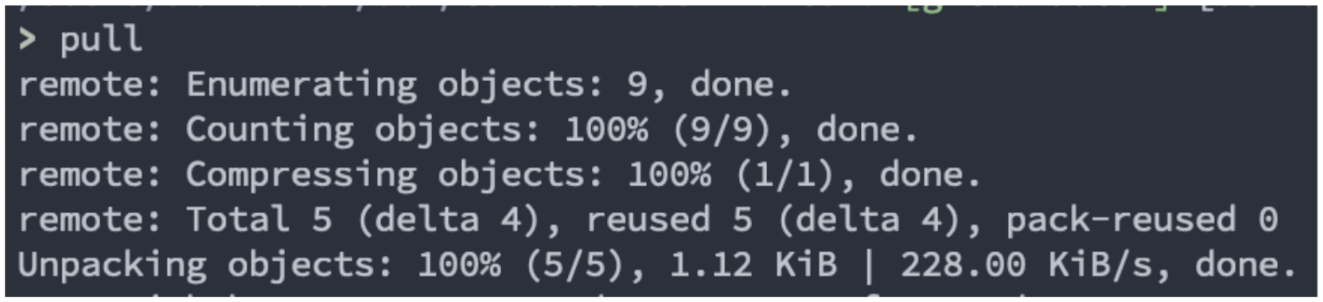


Every time you push, Git compresses



```
> push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
```

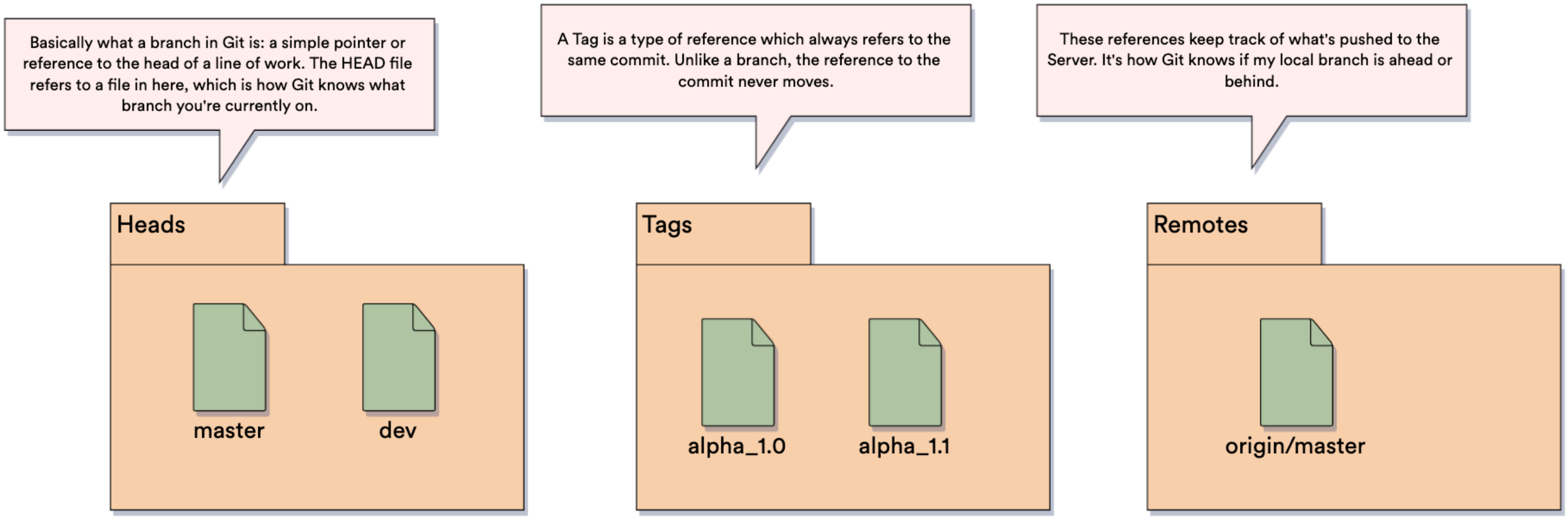
Every time you pull, Git unpacks



```
> pull
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 5 (delta 4), reused 5 (delta 4), pack-reused 0
Unpacking objects: 100% (5/5), 1.12 KiB | 228.00 KiB/s, done.
```

REFERENCES

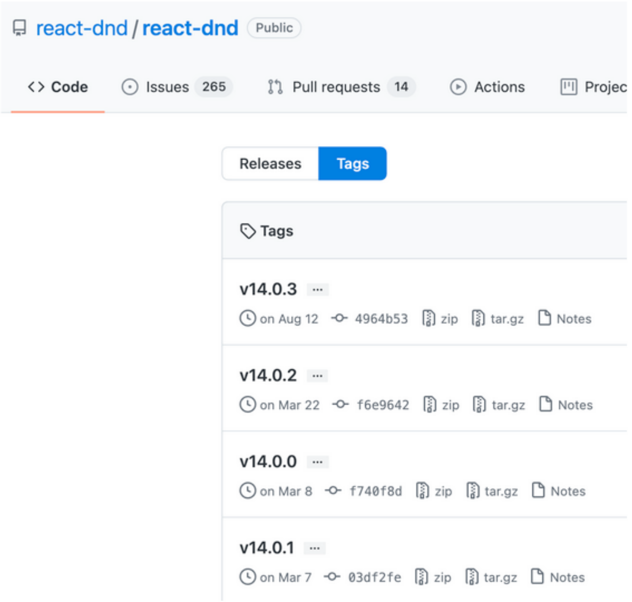
"References", or "refs", are simply aliases to commits.



Basically what a branch in Git is: a simple pointer or reference to the head of a line of work. The HEAD file refers to a file in here, which is how Git knows what branch you're currently on.

A Tag is a type of reference which always refers to the same commit. Unlike a branch, the reference to the commit never moves.

These references keep track of what's pushed to the Server. It's how Git knows if my local branch is ahead or behind.



react-dnd / react-dnd Public

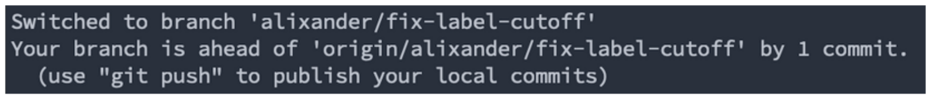
Tags

v14.0.3 ···
on Aug 12 4964b53 zip tar.gz Notes

v14.0.2 ···
on Mar 22 f6e9642 zip tar.gz Notes

v14.0.0 ···
on Mar 8 f740f8d zip tar.gz Notes

v14.0.1 ···
on Mar 7 03df2fe zip tar.gz Notes



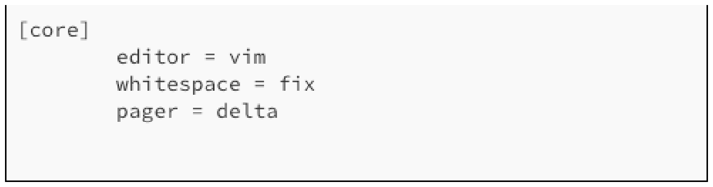
```
Switched to branch 'alixander/fix-label-cutoff'
Your branch is ahead of 'origin/alixander/fix-label-cutoff' by 1 commit.
  (use "git push" to publish your local commits)
```

CONFIGURATION

Git combines configurations from looking in multiple places and combining them, with later configs taking precedence

1 System config file

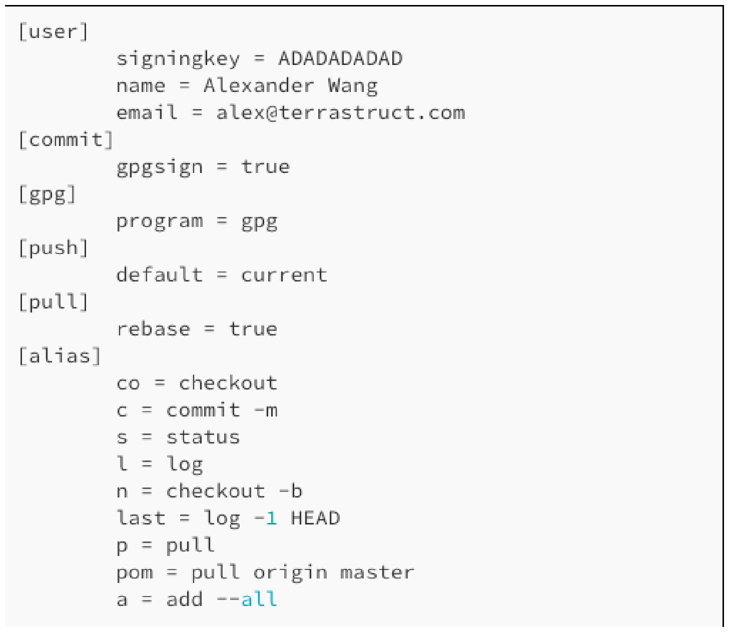
`/etc/gitconfig`



```
[core]
        editor = vim
        whitespace = fix
        pager = delta
```

2 User config file

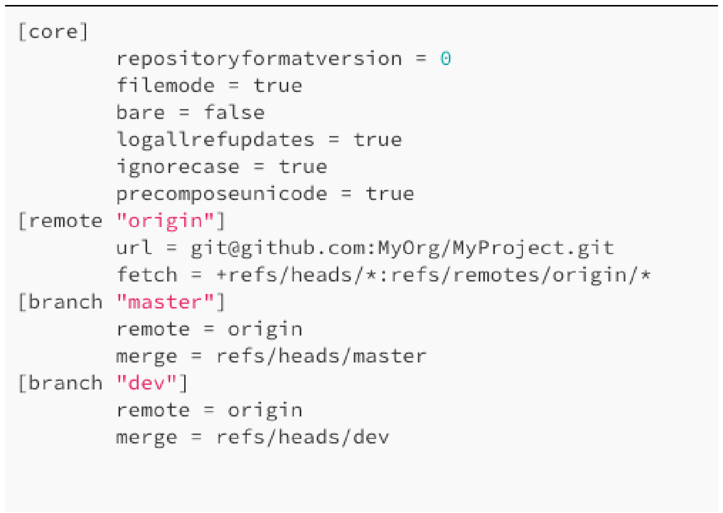
`~/.gitconfig`



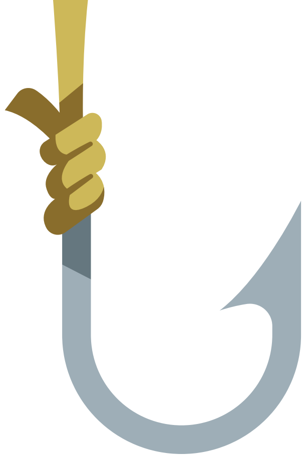
```
[user]
        signingkey = ADADADADAD
        name = Alexander Wang
        email = alex@terrastruct.com
[commit]
        gpgsign = true
[gpg]
        program = gpg
[push]
        default = current
[pull]
        rebase = true
[alias]
        co = checkout
        c = commit -m
        s = status
        l = log
        n = checkout -b
        last = log -1 HEAD
        p = pull
        pom = pull origin master
        a = add --all
```

3 Local config file

`./git/config`

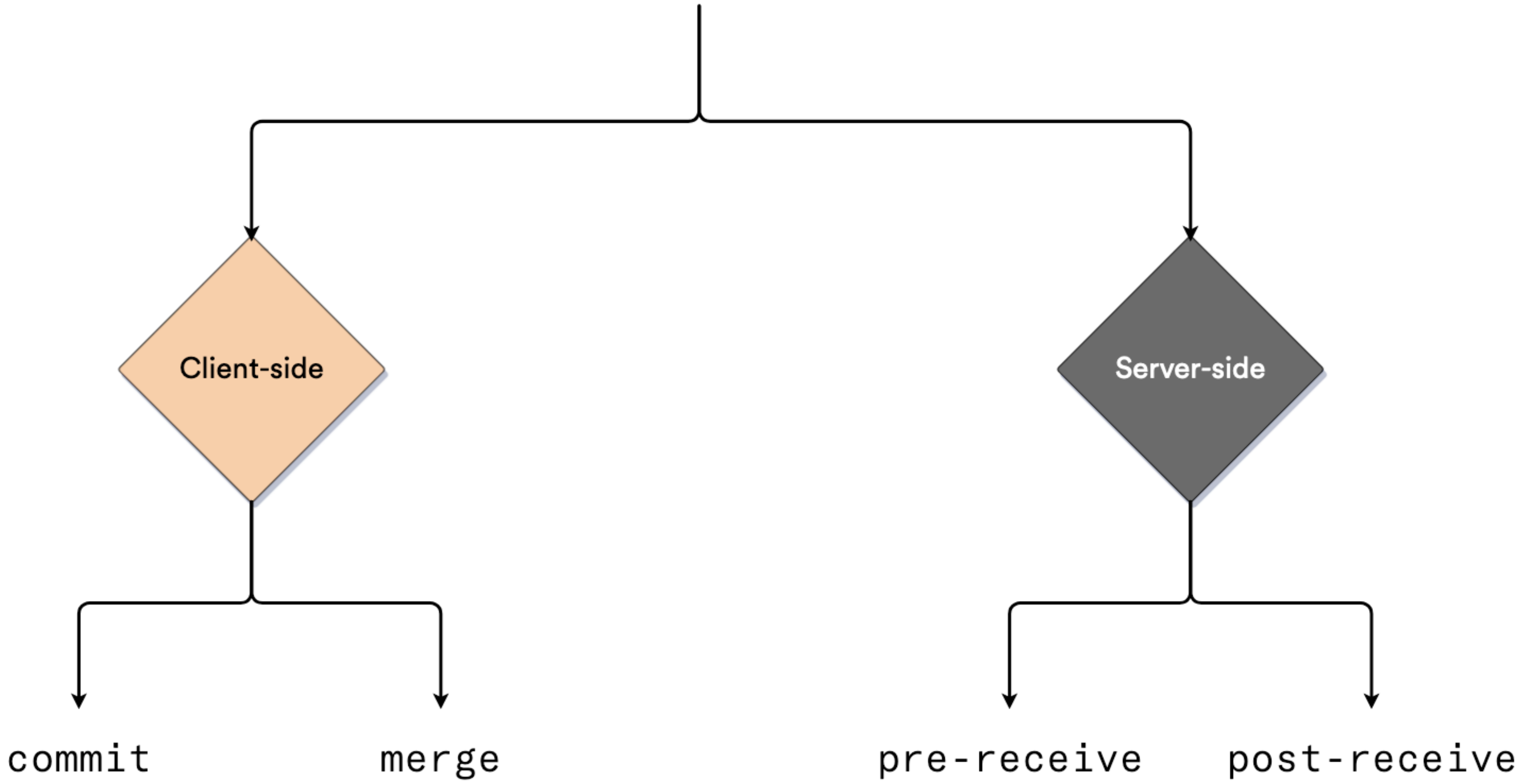


```
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
        ignorecase = true
        precomposeunicode = true
[remote "origin"]
        url = git@github.com:MyOrg/MyProject.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
        remote = origin
        merge = refs/heads/master
[branch "dev"]
        remote = origin
        merge = refs/heads/dev
```



HOOKS

Git lets you run scripts before or after certain actions. These scripts are called "hooks".



Client-side

Server-side

commit

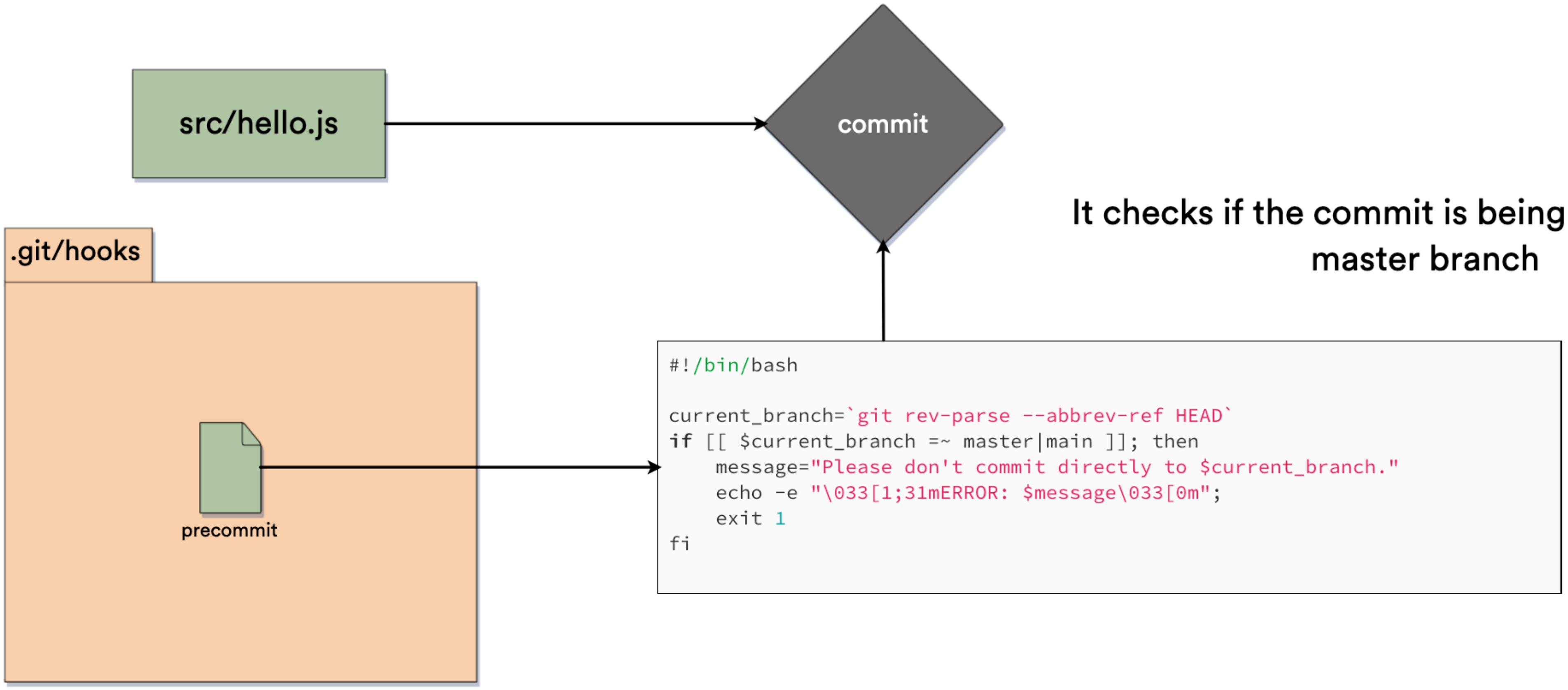
merge

pre-receive

post-receive

EXAMPLE HOOK

This precommit hook runs before every commit



```
#!/bin/bash

current_branch=`git rev-parse --abbrev-ref HEAD`
if [[ $current_branch =~ master|main ]]; then
    message="Please don't commit directly to $current_branch."
    echo -e "\033[1;31mERROR: $message\033[0m";
    exit 1
fi
```

It checks if the commit is being executed on master branch

AMIGOS CODE

PREMIUM PROGRAMMING COURSES
amigoscode.com