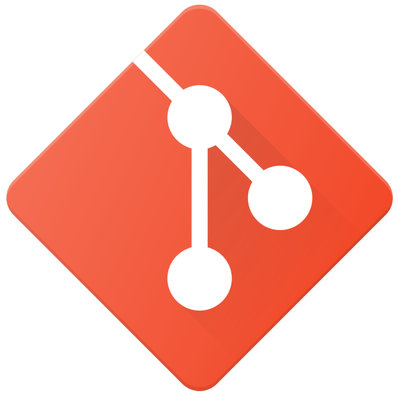
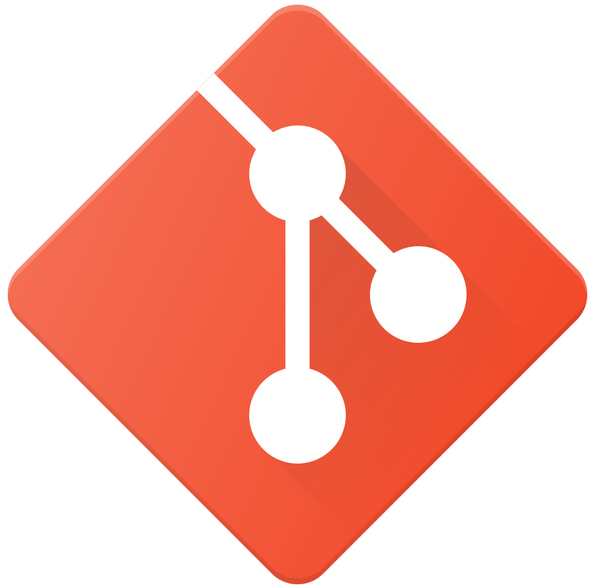


Git & GitHub

Complete Guide



What's **Git**?

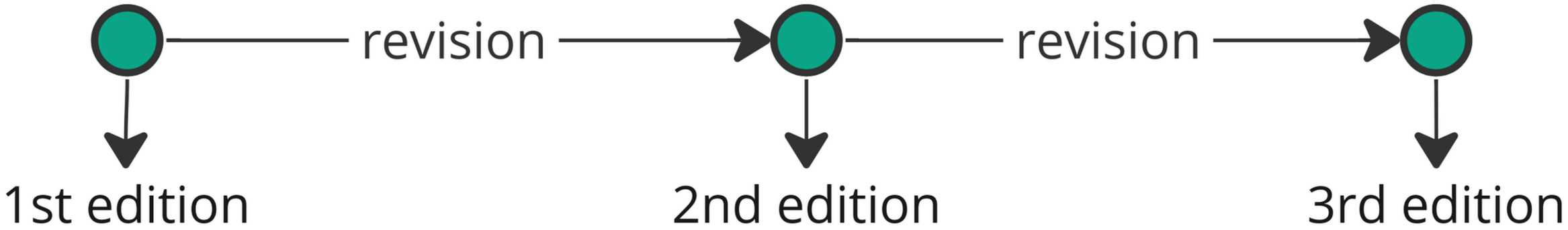
It's a **Distributed Version Control System**

DVCS

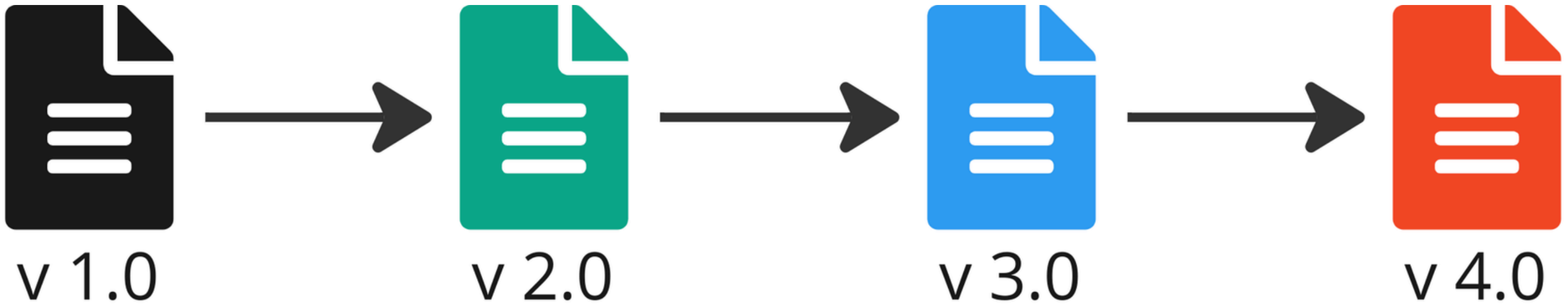
What's a **Version Control System**?

What's a **Version**?

A Version is a stable release of our work



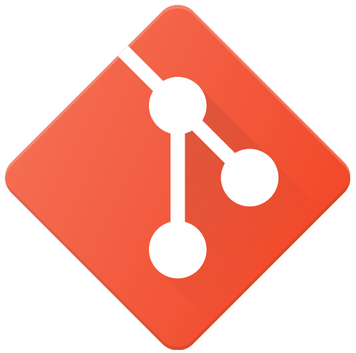
Renaming

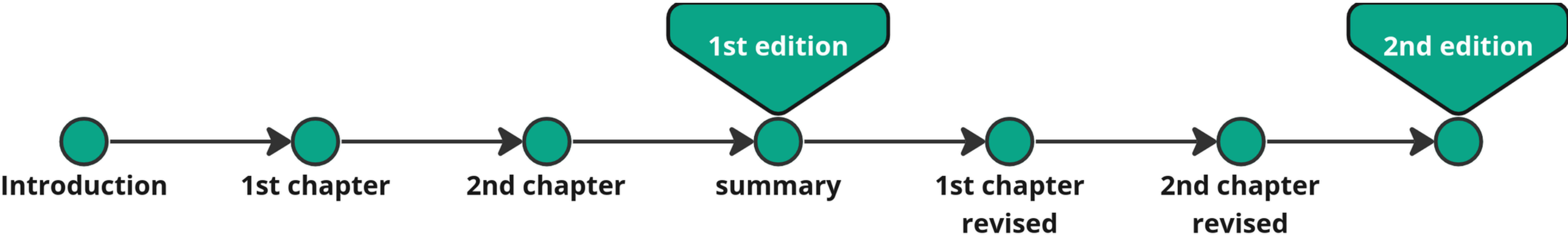


What's a **Version Control System**?

A **version control system** is a type of software that allows us to **track** changes, **collaborate**, and **undo** mistakes.

Some popular version control systems include:



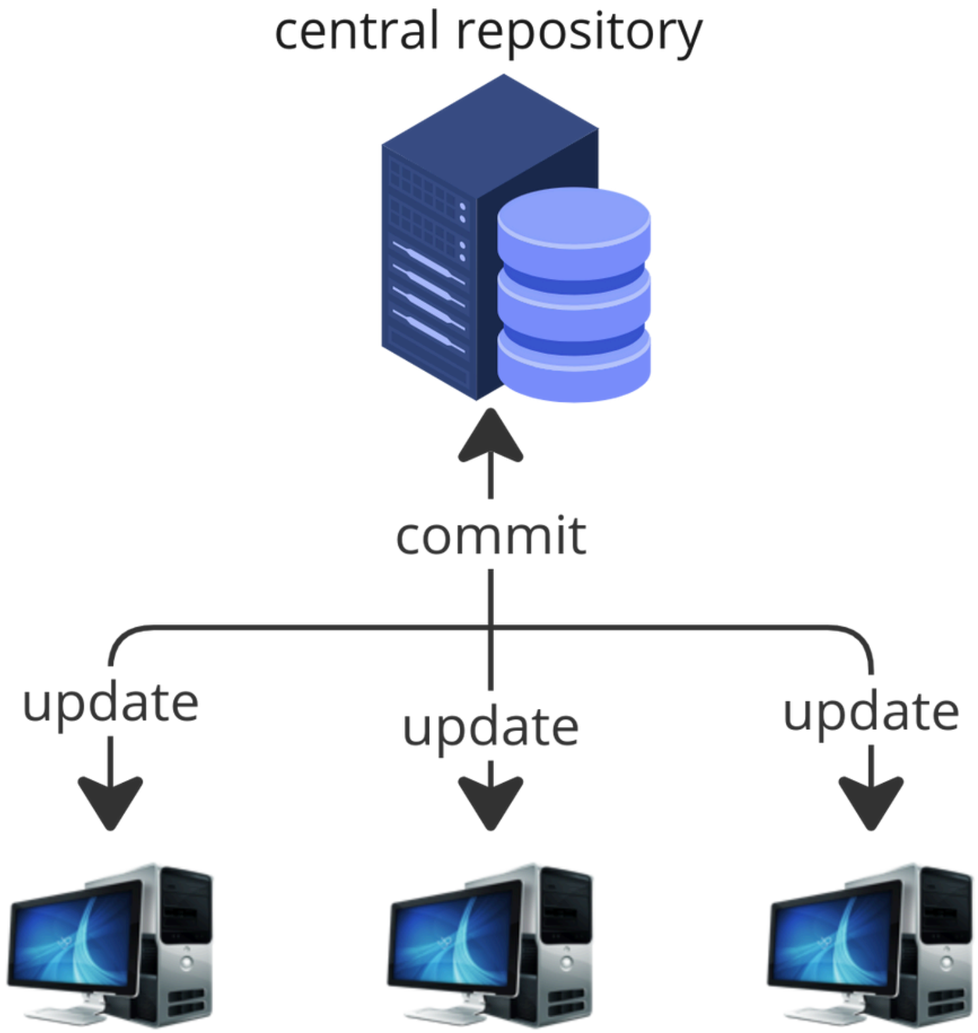

1st edition
2nd edition
Introduction
1st chapter
2nd chapter
summary
1st chapter revised
2nd chapter revised

It's a **Distributed Version Control System**

DVCS

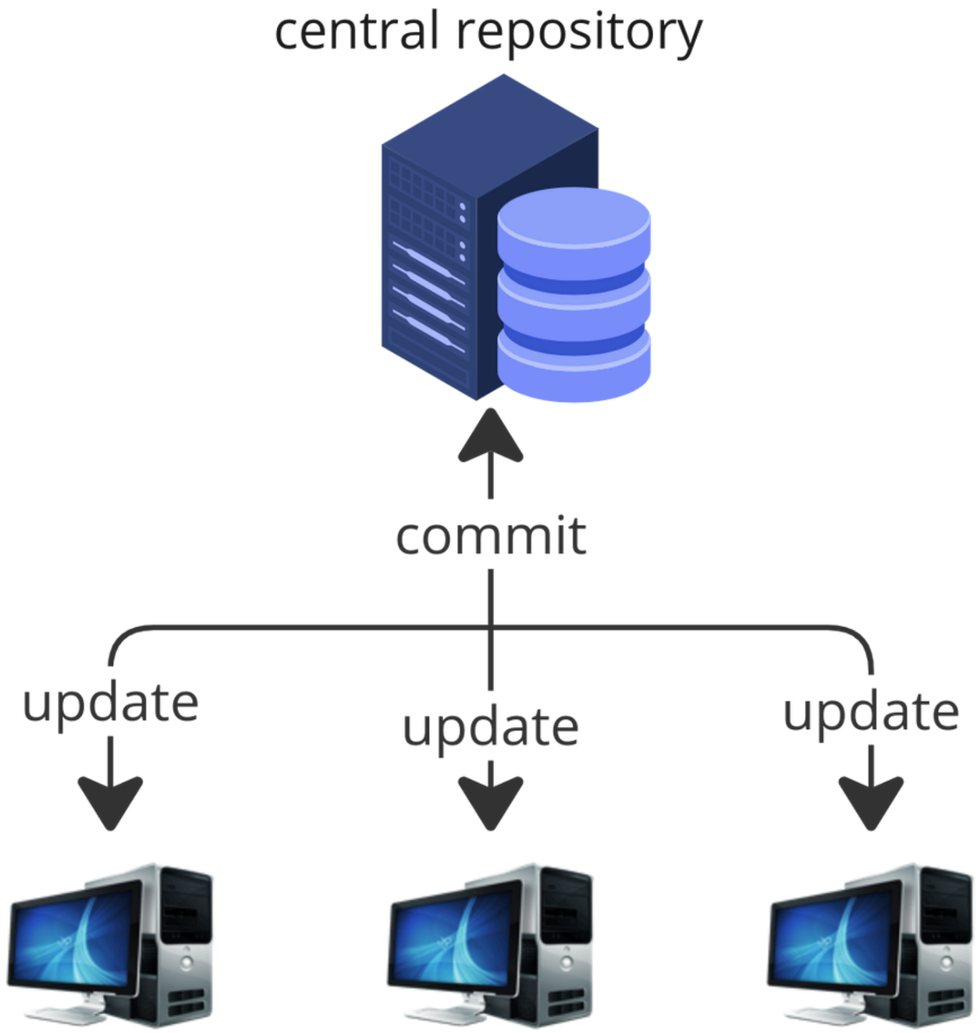
Centralized Vs Distributed VCS

Centralized CVCS



Centralized CVCS

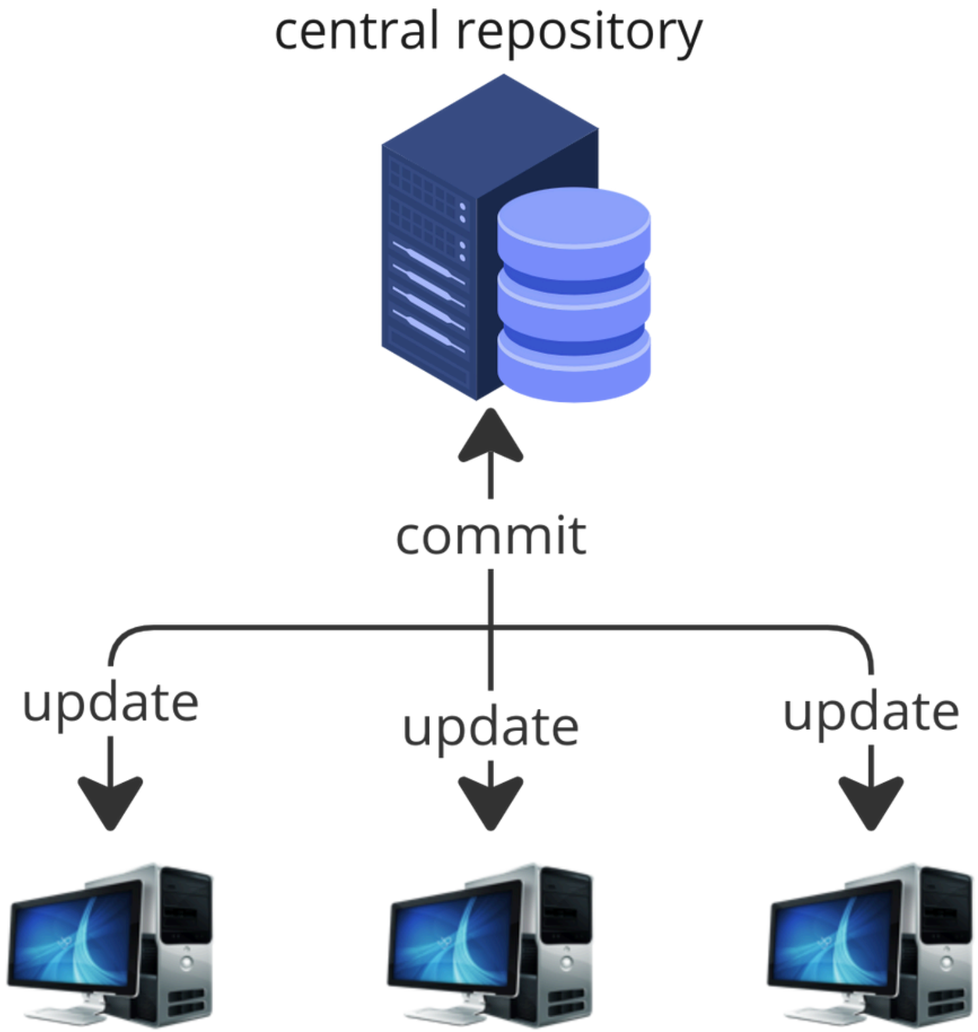
problems?



Centralized CVCS

problems?

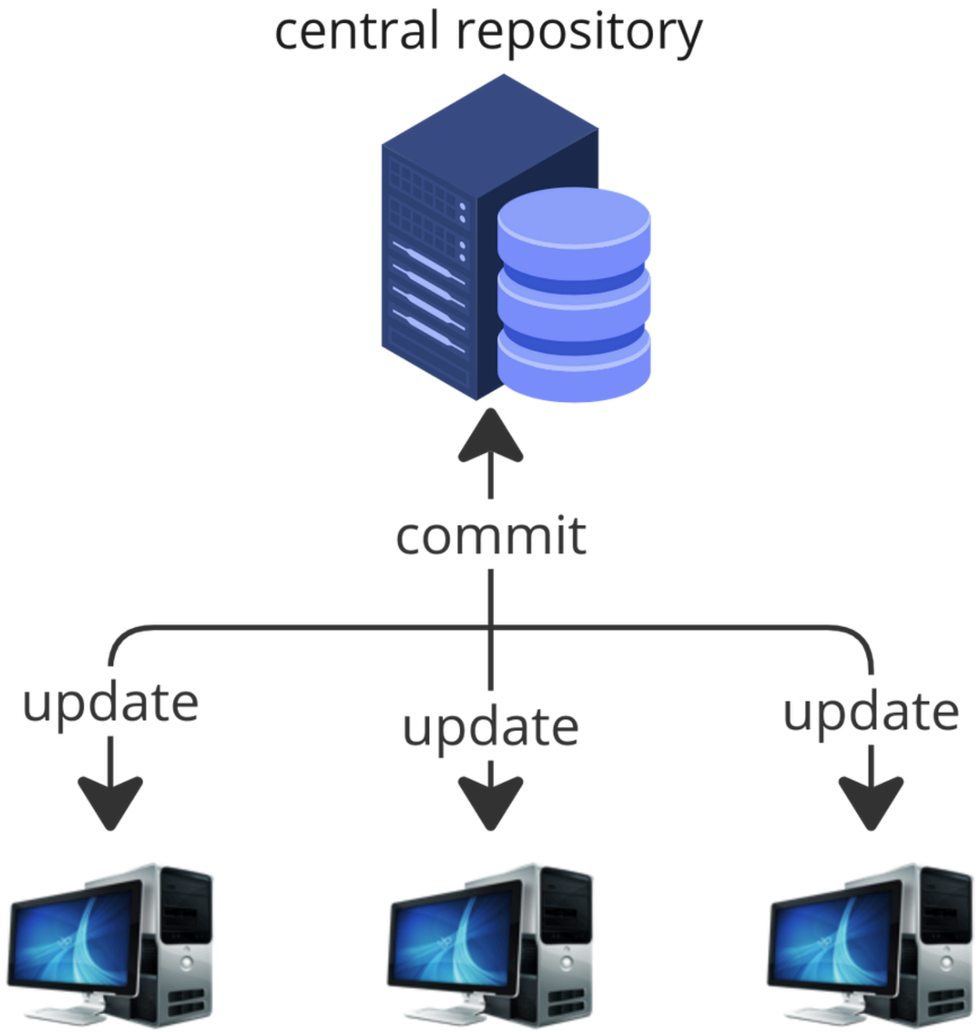
- no back-up
- slow
- no offline-work



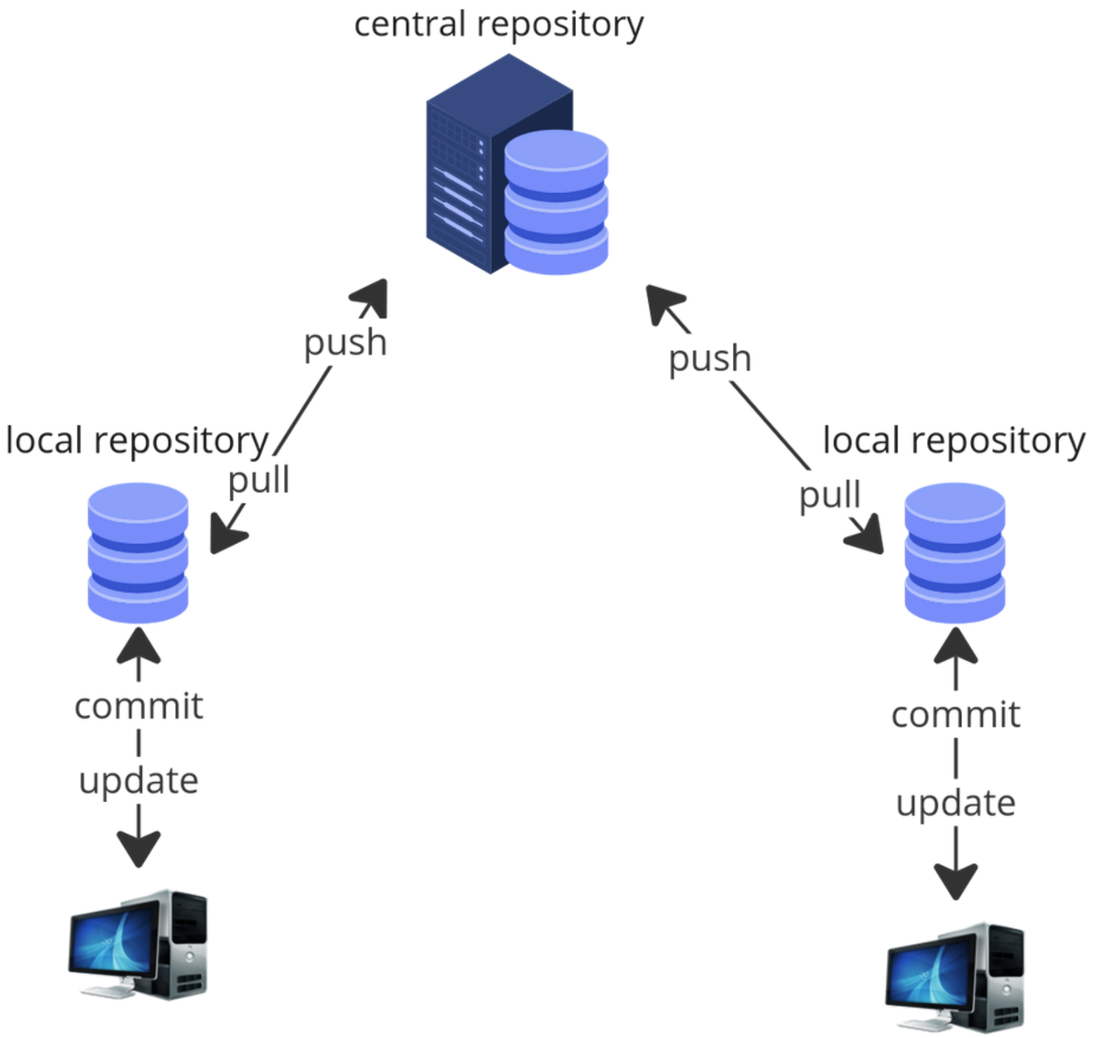
Centralized CVCS

problems?

- no back-up
- slow
- no offline-work

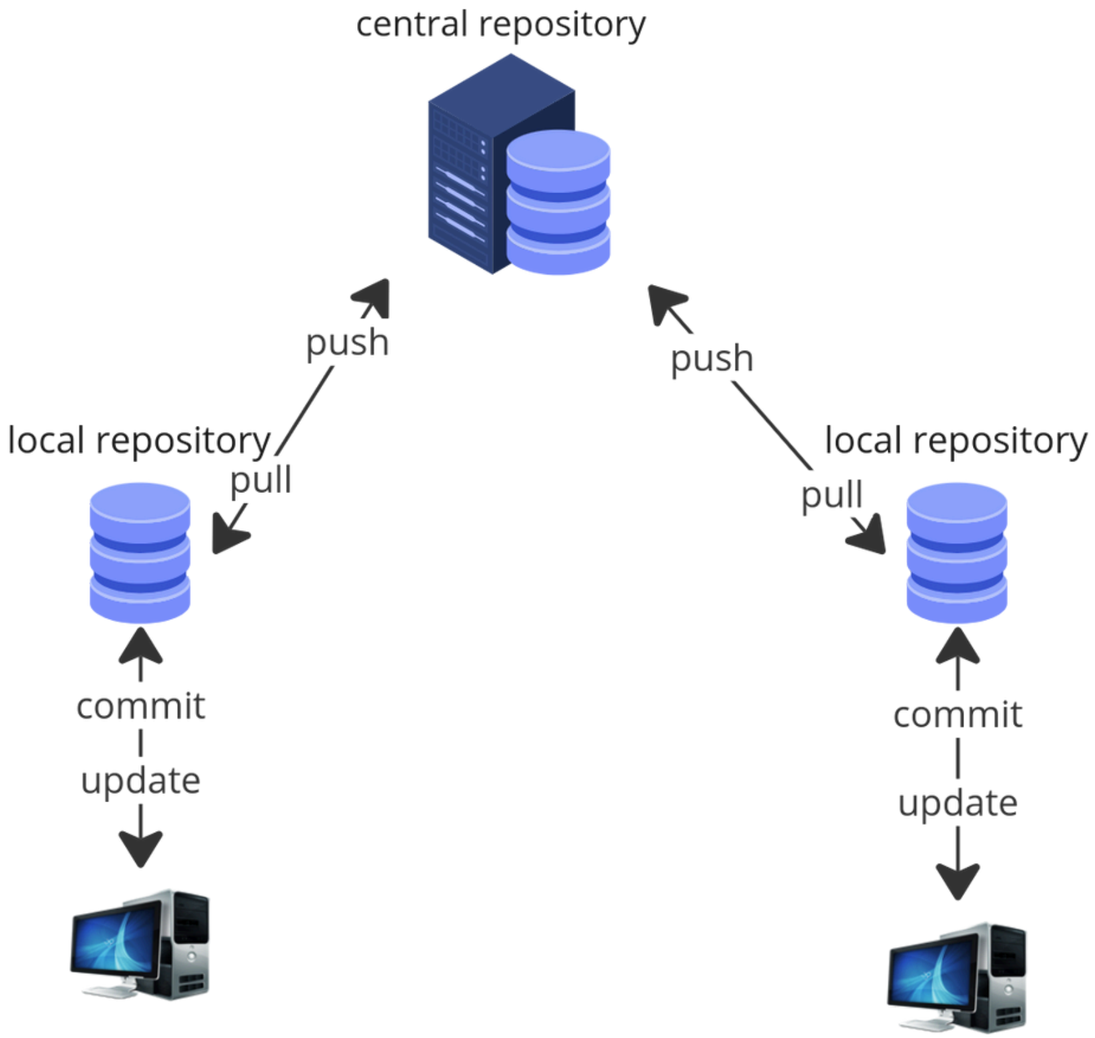


Distributed DVCS



Distributed DVCS

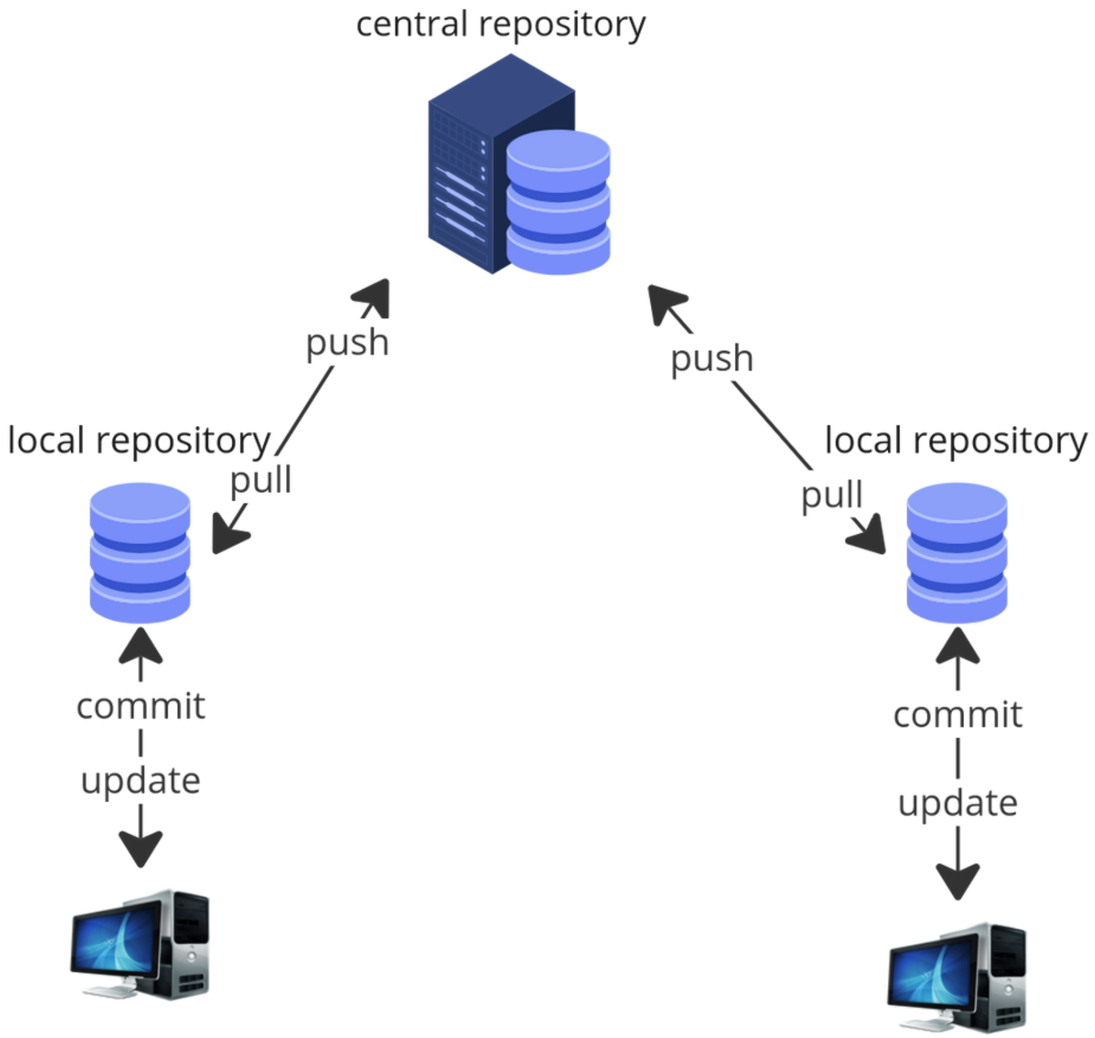
solutions:



Distributed DVCS

solutions:

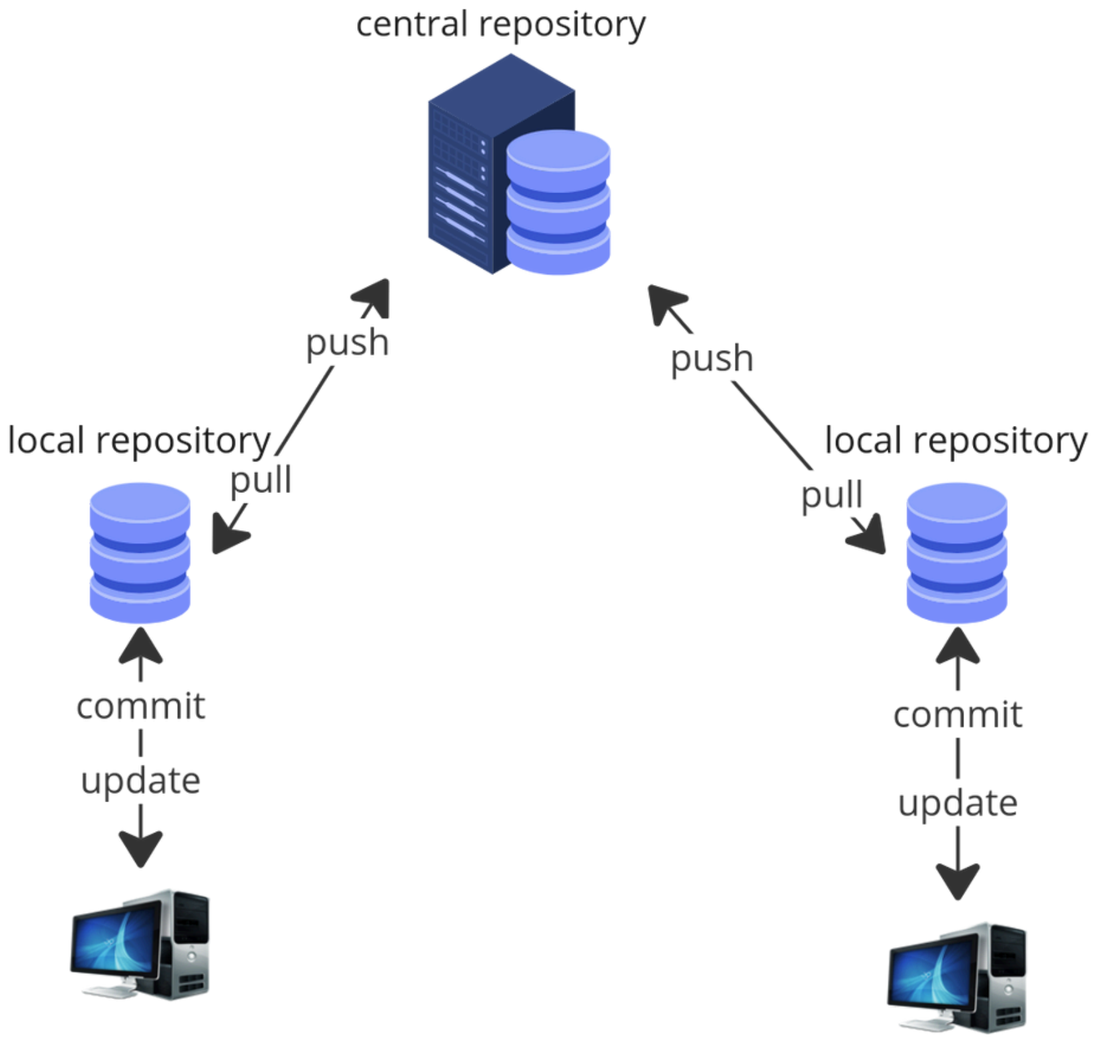
- history is shared
- extremely fast
- offline-work



Distributed DVCS

solutions:

- history is shared
- extremely fast
- offline-work



Short History of Git

BitKeeper
Scalable Version Control

2002 — Linux Team Started Using BitKeeper SCM from free version from BitMover

2005 — BitMover Stopped providing free version

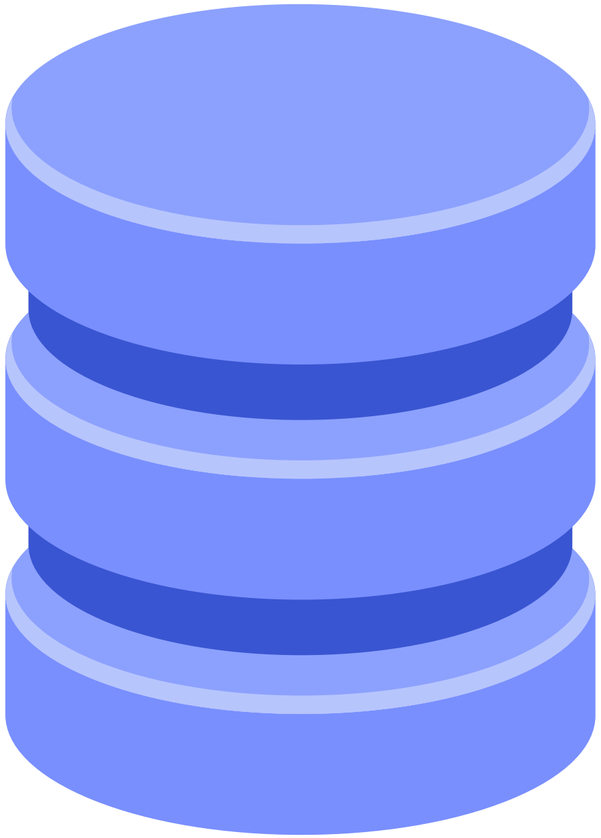
2005, April — Git Was Born

first commit

How Does Git Work?

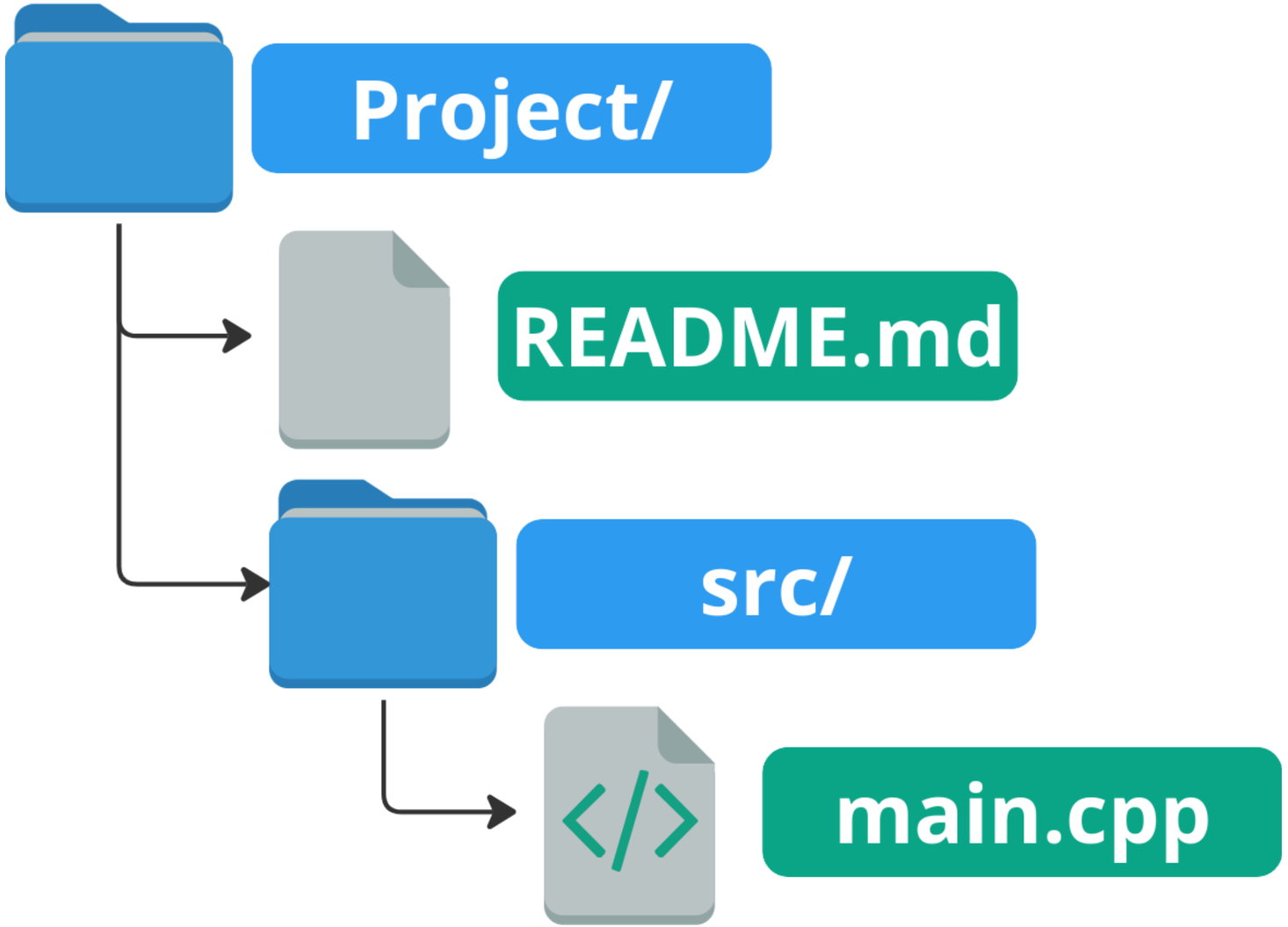
(internally)

Git needs a Place

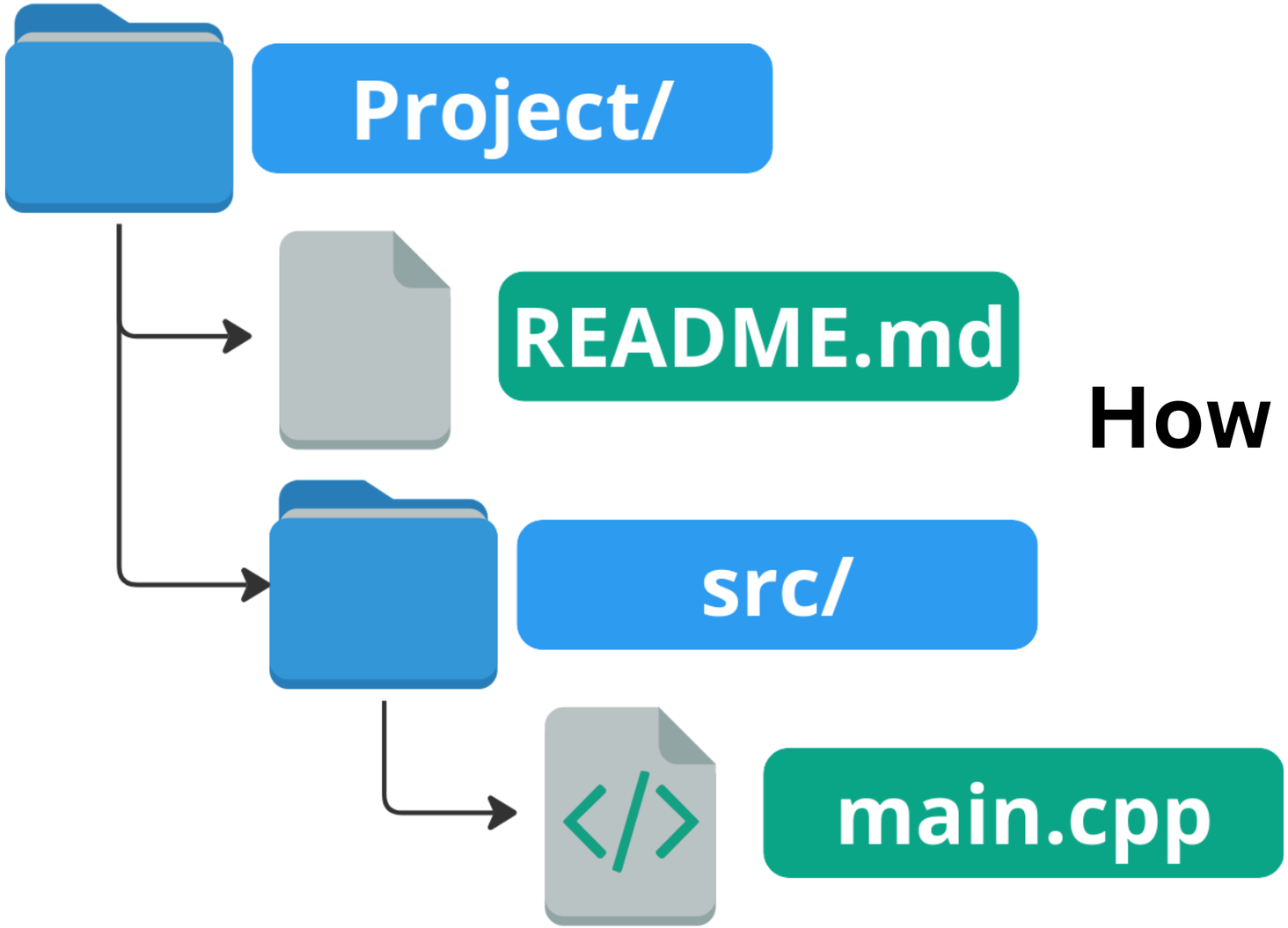


.git directory (repository)

Working Directory



Working Directory



How Does git see it?

Git Object Types

git objects are what git actually stores, we have four types, all of them are stored in **Git Object Database** which is stored in **.git/objects/**
each object is:

- **Hashed** using **SHA-1**
- **compressed** using **Zlib**

compressed file is stored at
SHA-1[1,2]**/**SHA-1[3:40]

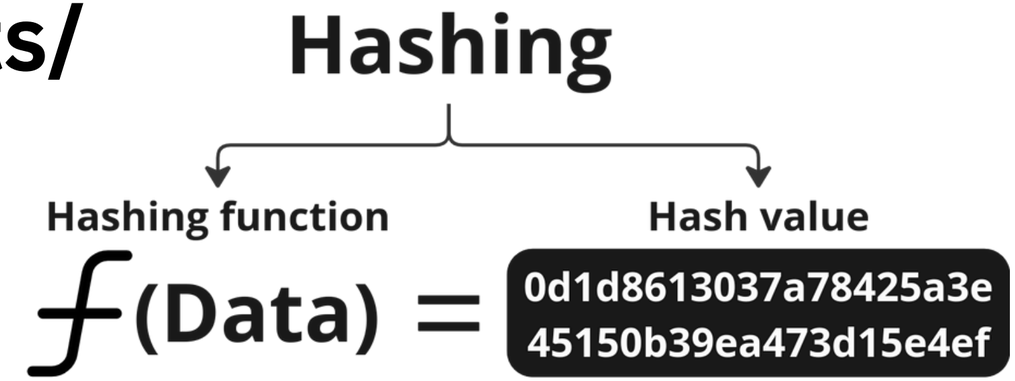
Git Object Types

git objects are what git actually stores, we have four types, all of them are stored in **Git Object Database** which is stored in **.git/objects/**

each object is:

- **Hashed** using **SHA-1**
- **compressed** using **Zlib**

compressed file is stored at SHA-1[1,2]**/**SHA-1[3:40]

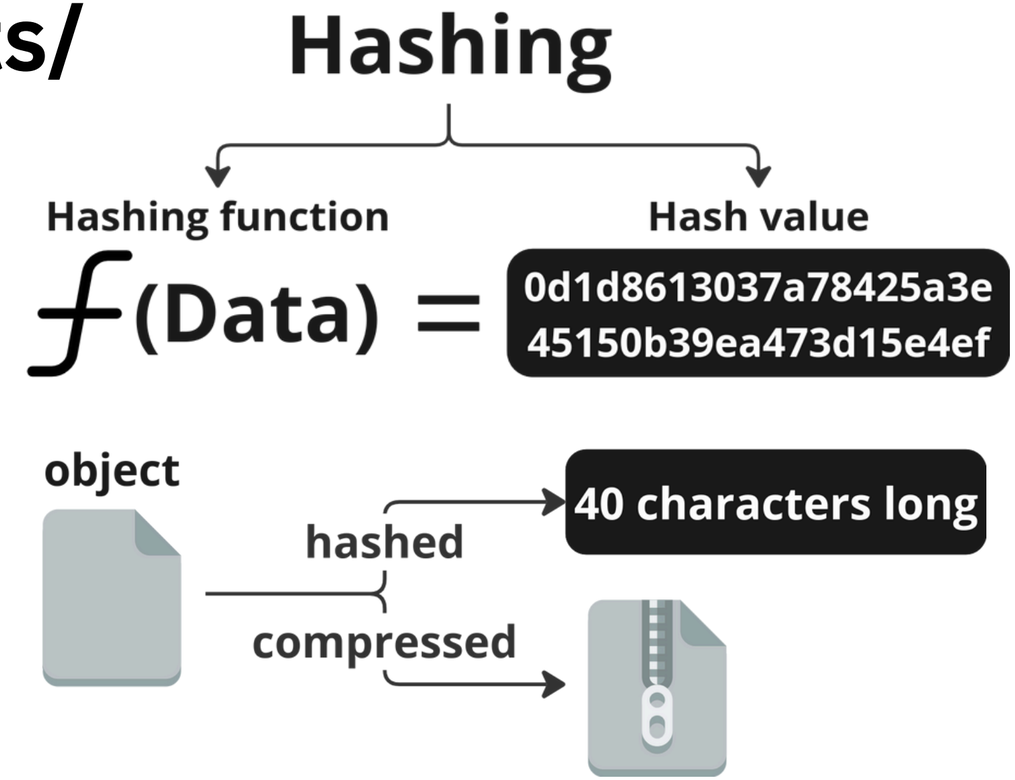


Git Object Types

git objects are what git actually stores, we have four types, all of them are stored in **Git Object Database** which is stored in **.git/objects/**
each object is:

- **Hashed** using **SHA-1**
- **compressed** using **Zlib**

compressed file is stored at SHA-1[1,2]**/**SHA-1[3:40]

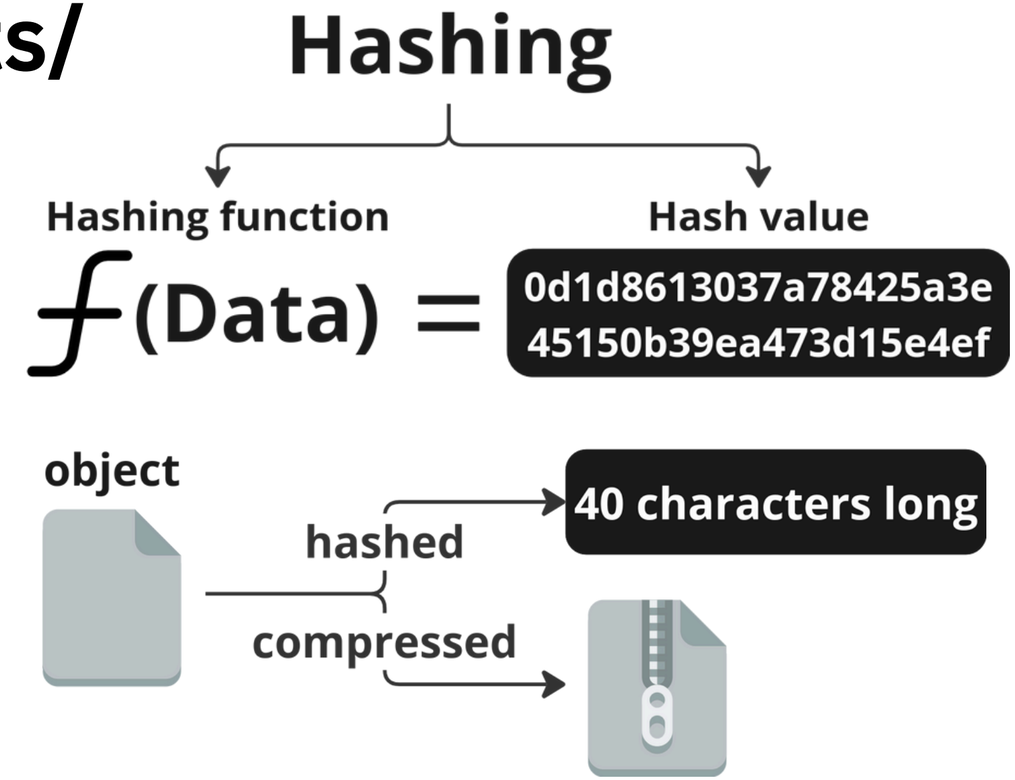


Git Object Types

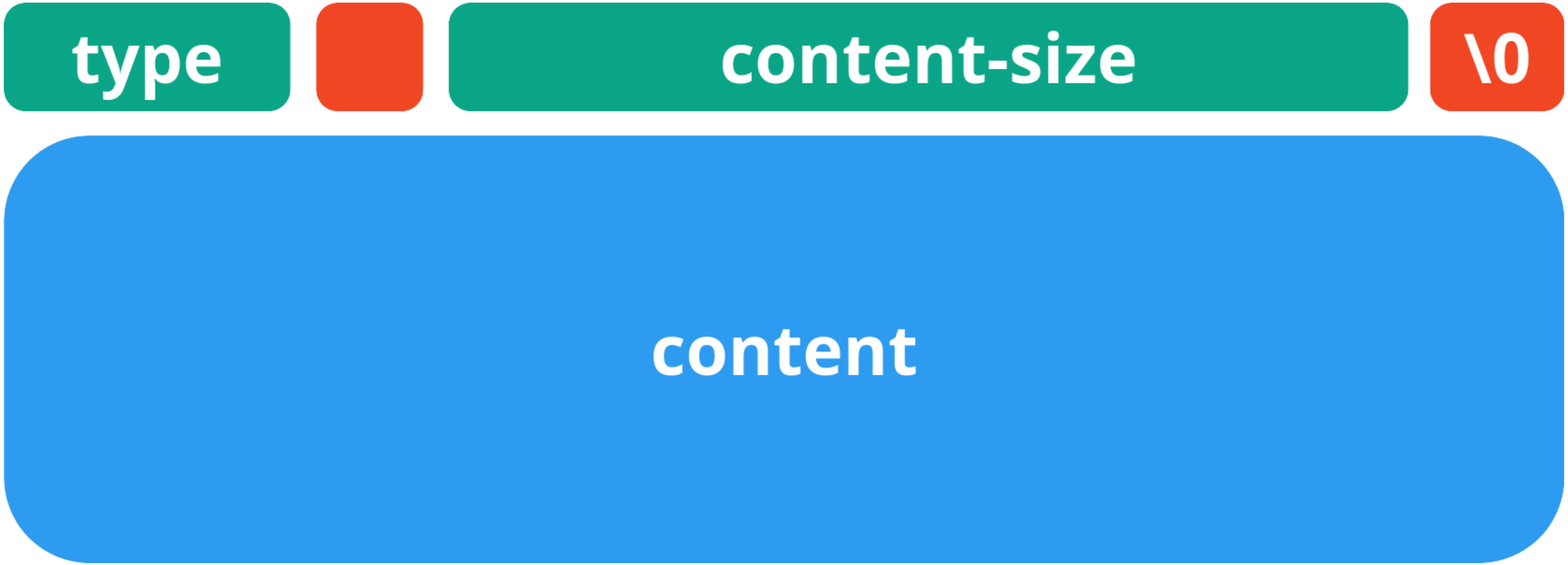
git objects are what git actually stores, we have four types, all of them are stored in **Git Object Database** which is stored in **.git/objects/**
each object is:

- **Hashed** using **SHA-1**
- **compressed** using **Zlib**

compressed file is stored at SHA-1[1,2]**/**SHA-1[3:40]



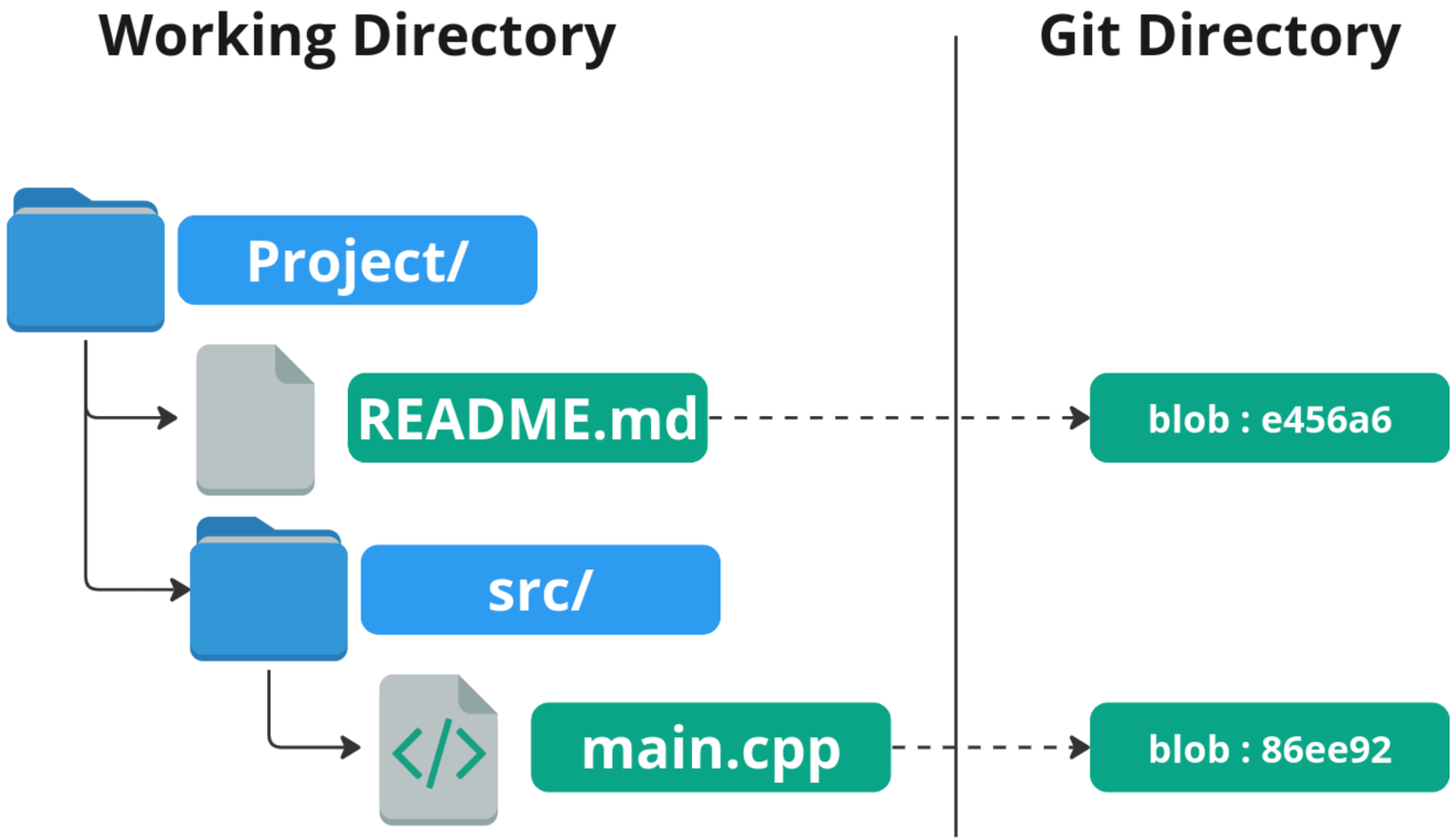
Object Structure



Note:object’s content depend on its type

The Blob

files-content are stored as blobs (Binary large Objects)



The Blob

The Blob

 # GitUnleashed

The Blob

blob

GitUnleashed

The Blob

GitUnleashed

The Blob

The Blob

The Blob

e456a6bbe37c808b34ed47e22eaaea668ac16f7d

The Blob

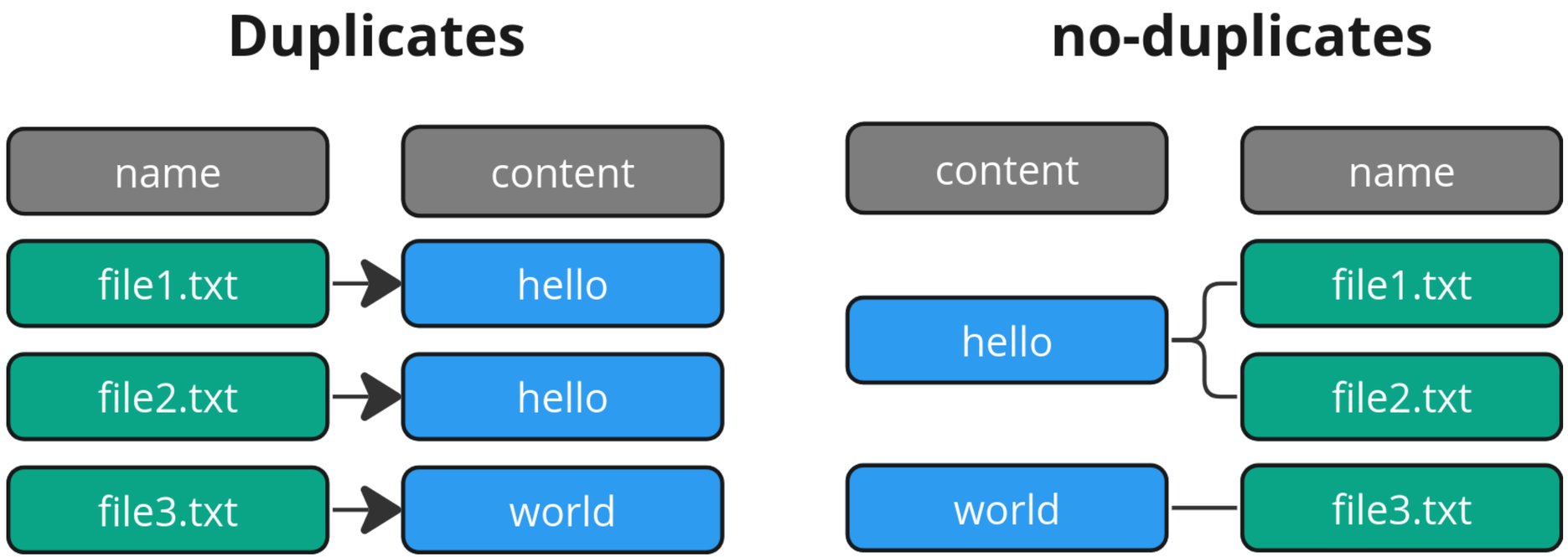
e456a6bbe37c808b34ed47e22eaaea668ac16f7d

The Blob

what if there’re multiple files with the same content? git will only stores one blob for that content, and share it among these files. **(doesn’t store duplicates)**

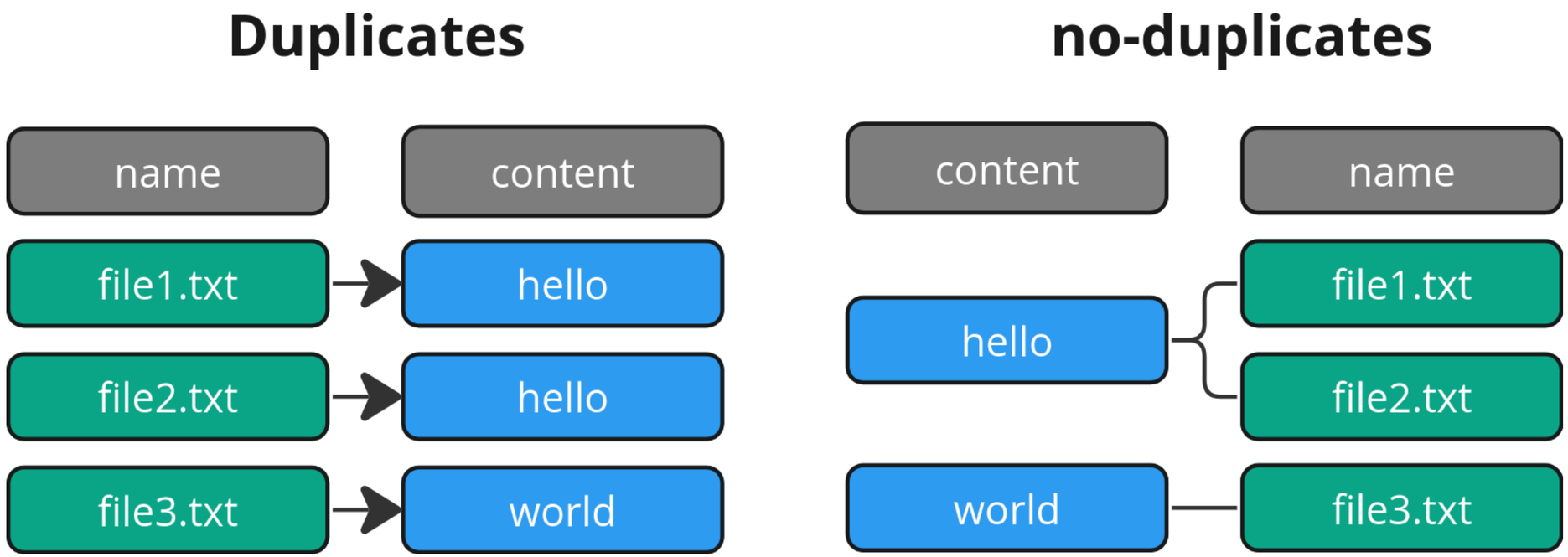
The Blob

what if there’re multiple files with the same content? git will only stores one blob for that content, and share it among these files. **(doesn’t store duplicates)**



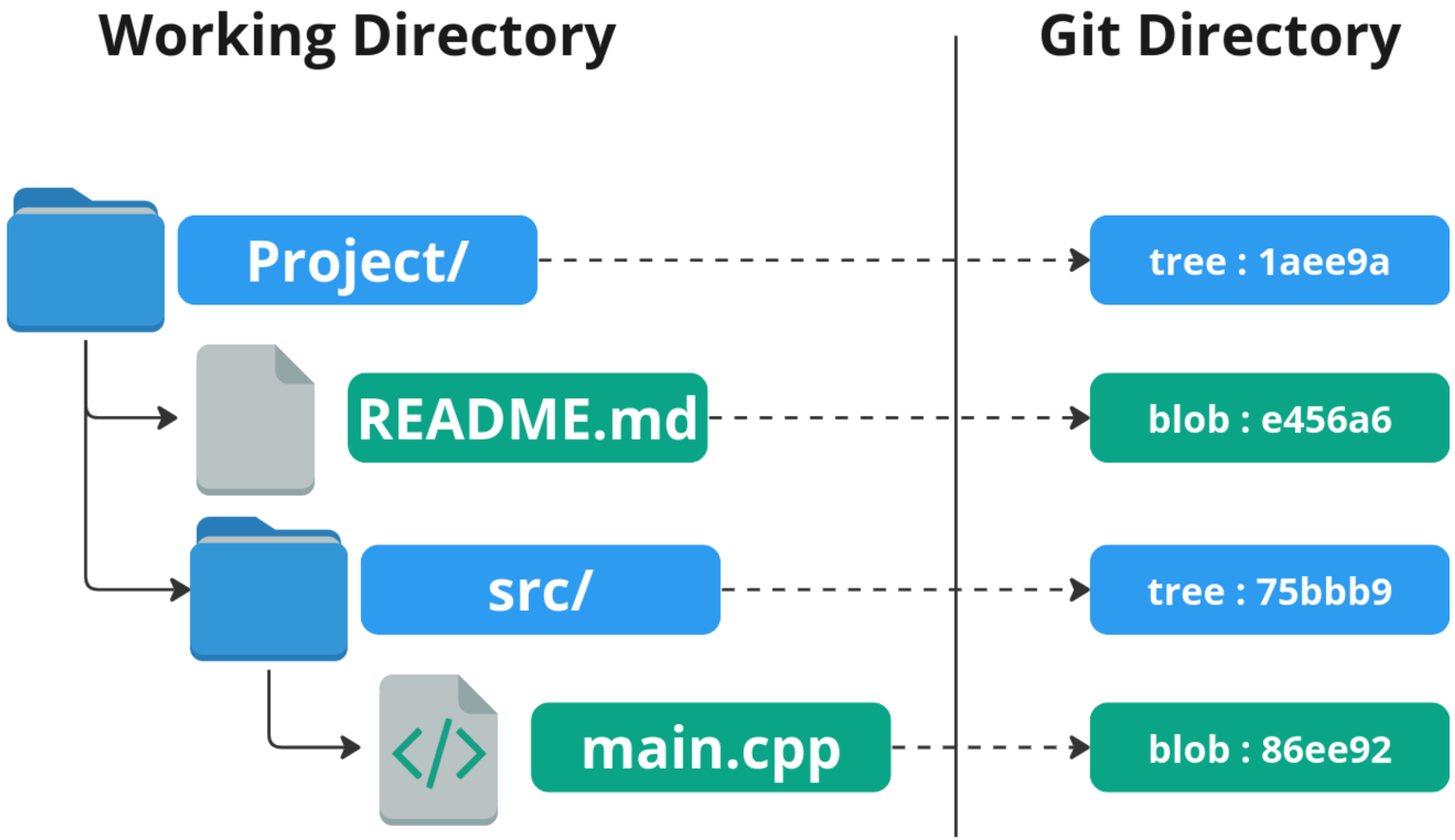
The Blob

what if there’re multiple files with the same content? git will only stores one blob for that content, and share it among these files. **(doesn’t store duplicates)**

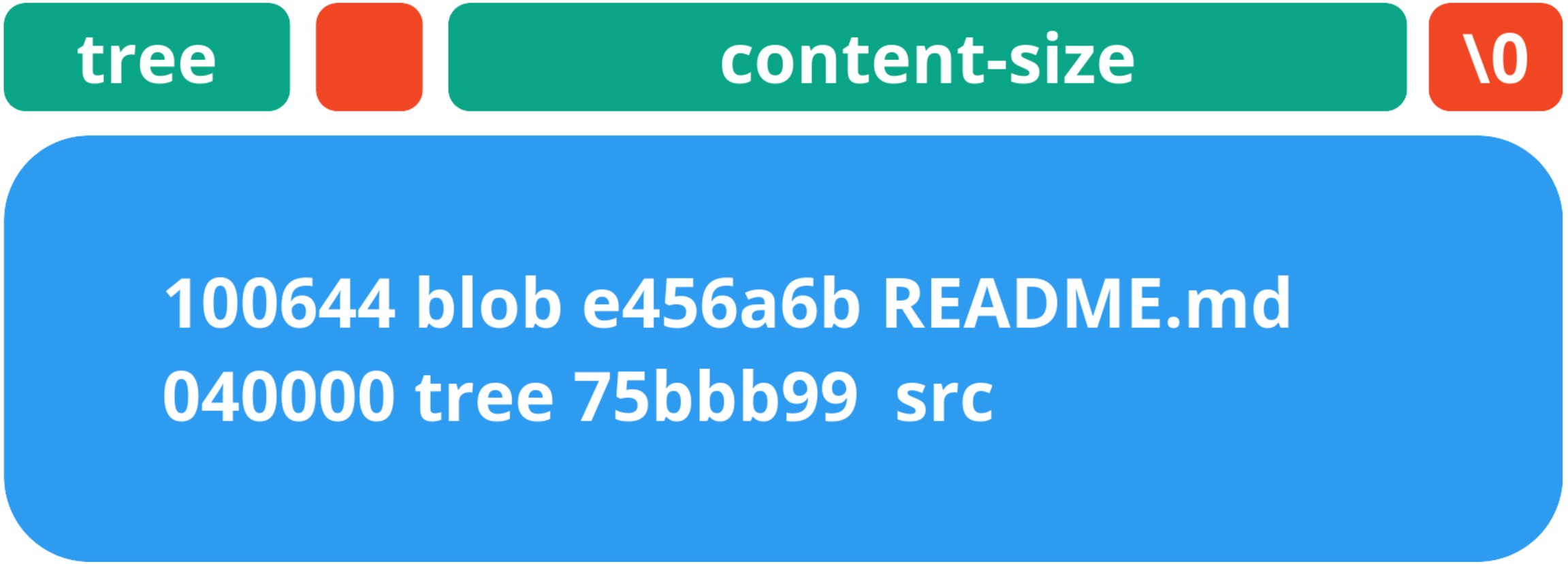


The Tree

Directories (folder) are just trees



The Tree


tree
content-size
\0
100644 blob e456a6b README.md
040000 tree 75bbb99 src

Tree Entry

Tree Entry

type is inferred by the mode

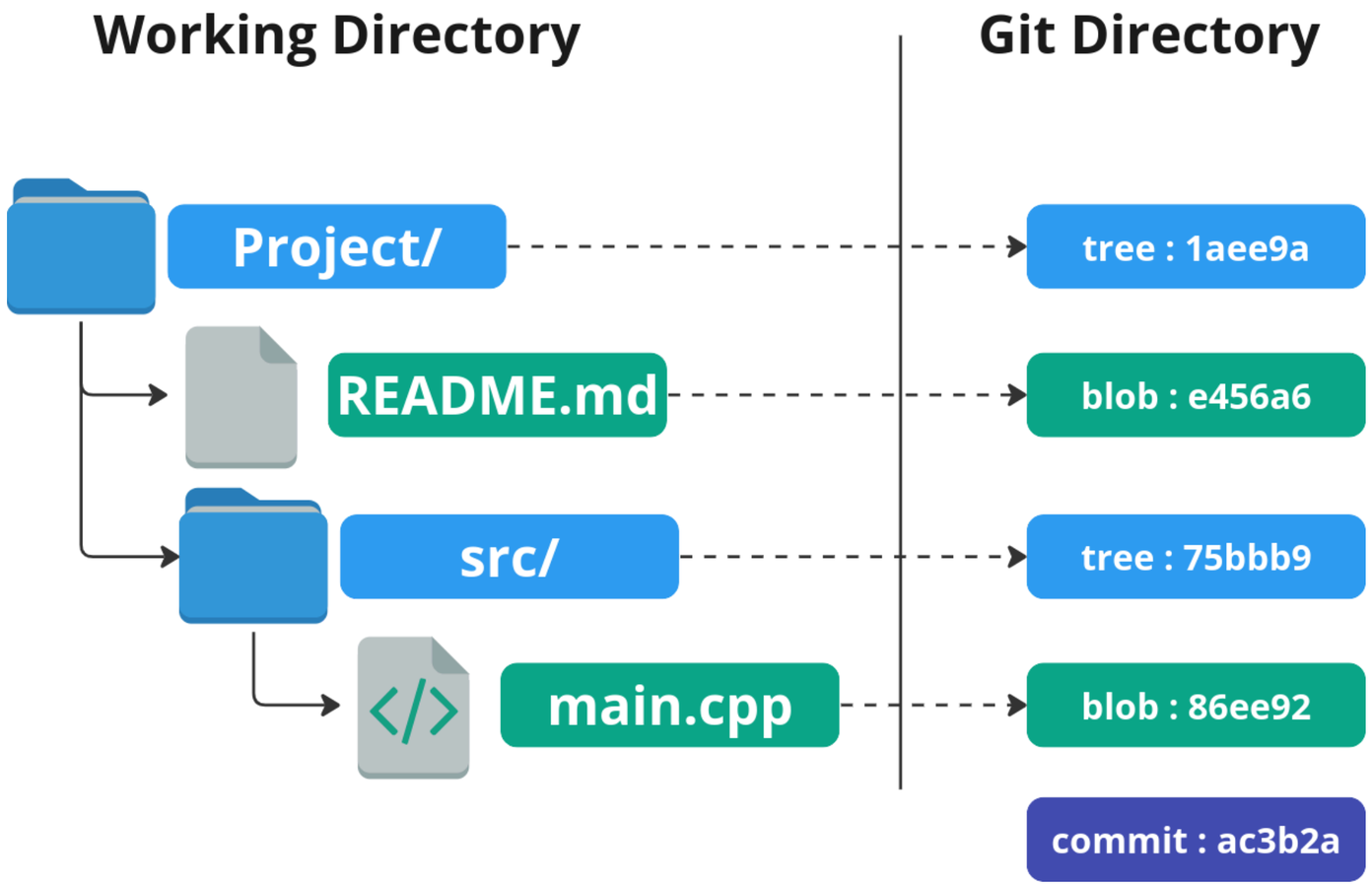
Tree Entry

type is inferred by the mode

Tree Entry Modes

The Commit

commits are the checkpoints



The Commit


commit
content-size
\0
tree 1aee9a
author Badr-1 <badrmohamedashraf@gmail.com> 1695662549 +0300
committer Badr-1 <badrmohamedashraf@gmail.com> 1695662549 +0300
init

The Commit


commit
content-size
\0
tree ecde1c
parent ac3b2a
author Badr-1 <badrmohamedashraf@gmail.com> 1695722752 +0300
committer Badr-1 <badrmohamedashraf@gmail.com> 1695722752 +0300
second commit

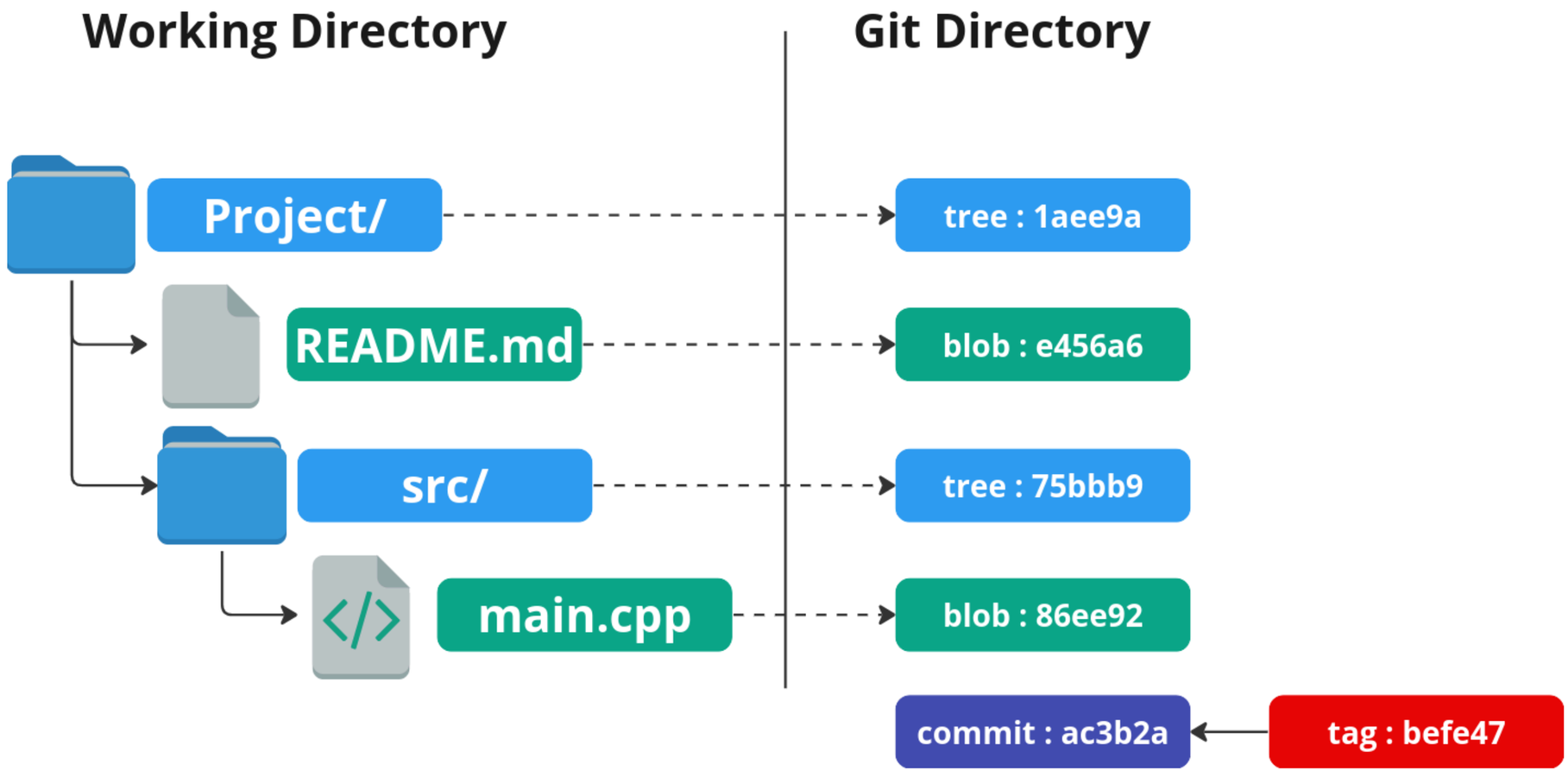
The Commit

content:

- tree
- parent commit(s)
- author
- committer
- data & time
- message

The Tag (annotated)

a short-hand for a particular commit



The Tag (annotated)

How Do We Know which is the Last?

References

they’re pointers (markers) to a particular commit in history.

There’re two references:

- branches **lives at** .git/refs/heads
- tags **lives at** .git/refs/tags

branches are timelines of the development, while tags are important points in the time line.

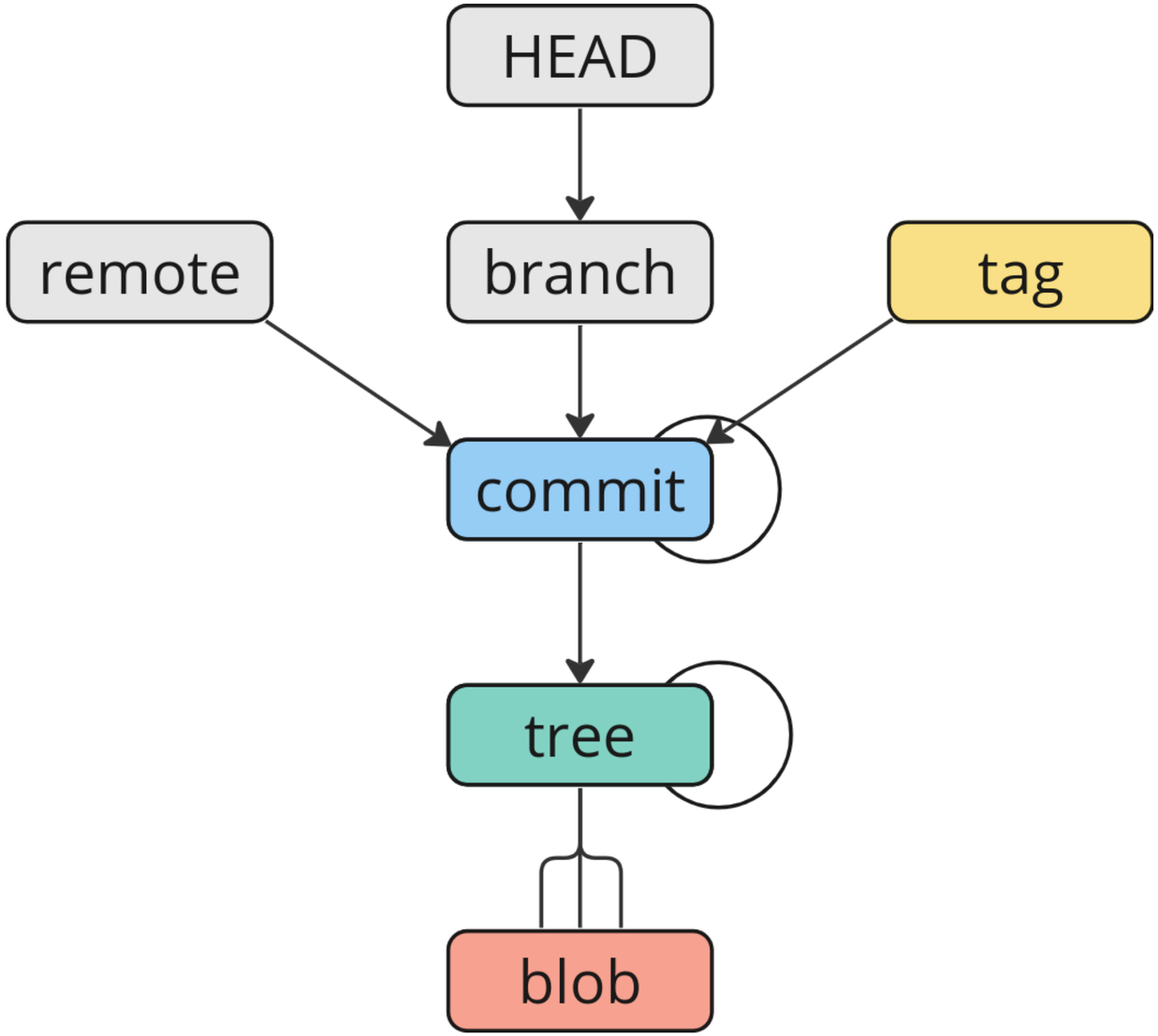
How Do We Know where are we?

HEAD

HEAD is a reference for the checkpoint (commit) we’re at currently, or the time line we’re on, it can hold two formats:

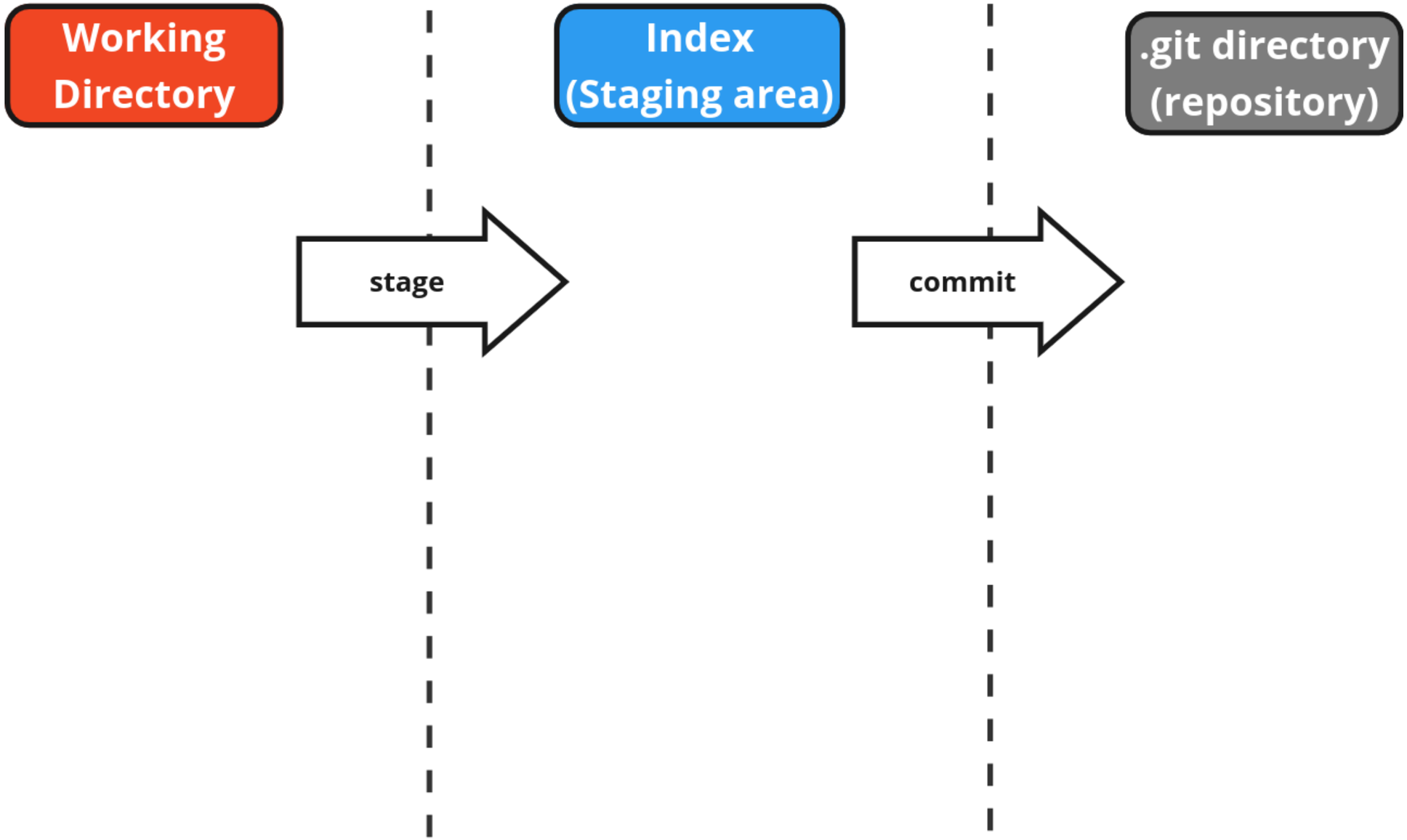
- refs: refs/heads/branch-name
- commit sha-1

The Bigger Picture



But How does it know what changed?

The Index

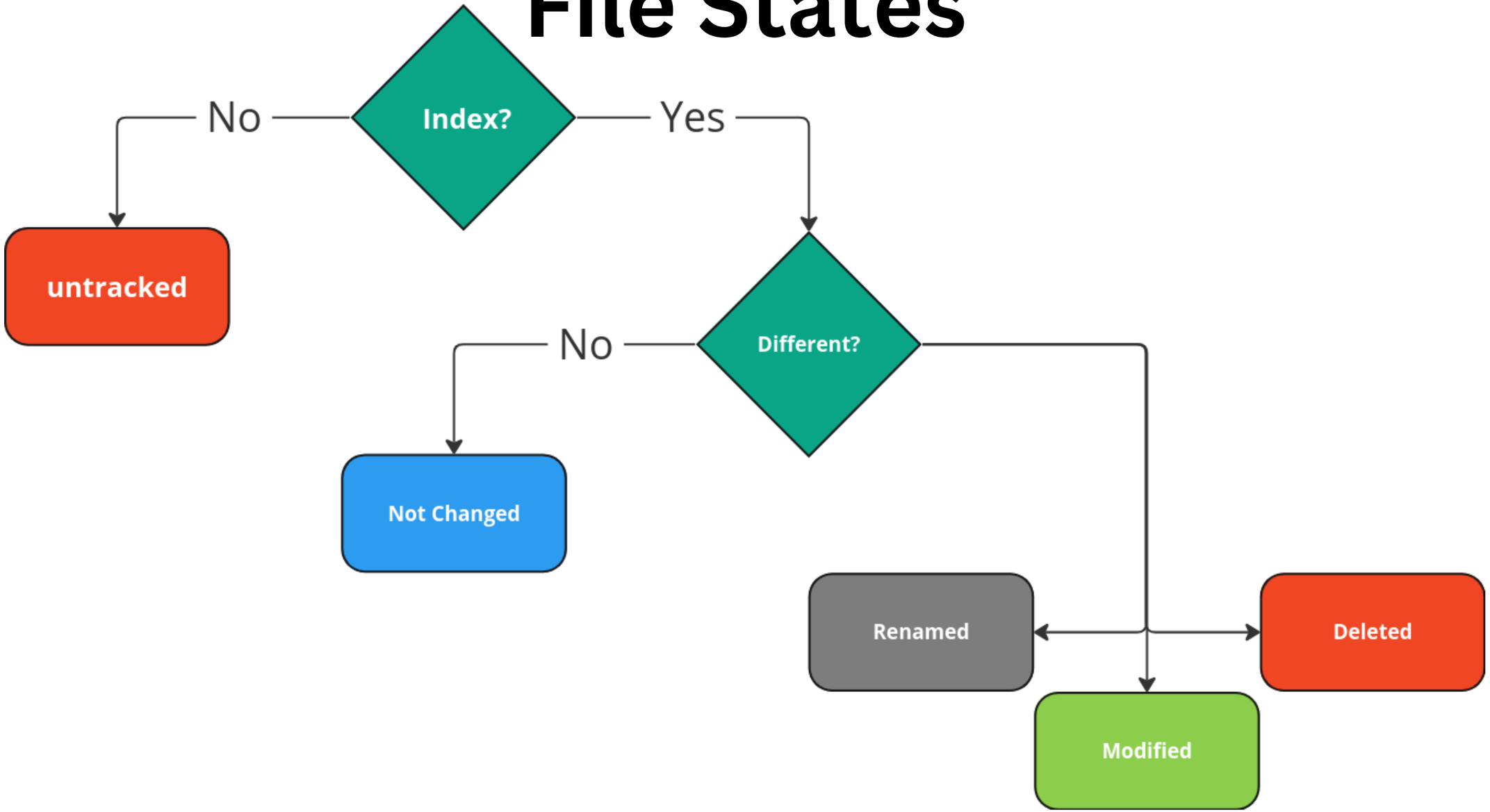

Working Directory
Index (Staging area)
.git directory (repository)
stage
commit

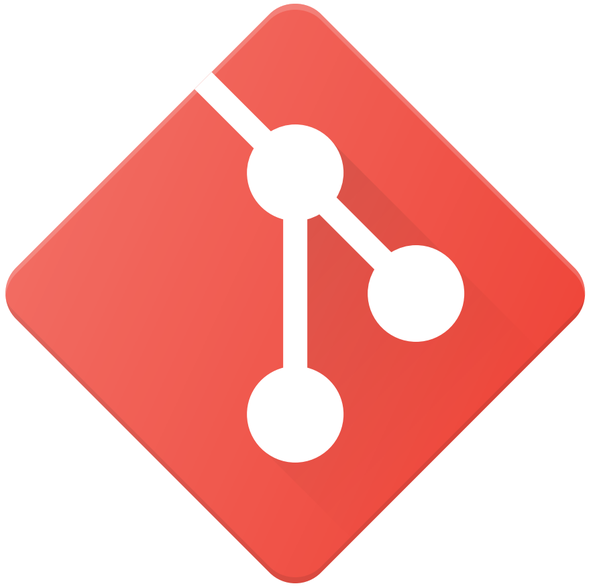
The Index

The index is a file in **.git/** which determine what changed as it stores the original state i.e. the state of the file before the change.
it stores multiple information but most importantly:

- file path
- file SHA-1
- file permissions

File States


Index?
No
Yes
untracked
Different?
No
Not Changed
Renamed
Deleted
Modified



Getting Started With **Git**