Git lab I

5 Initializing a local Git repository

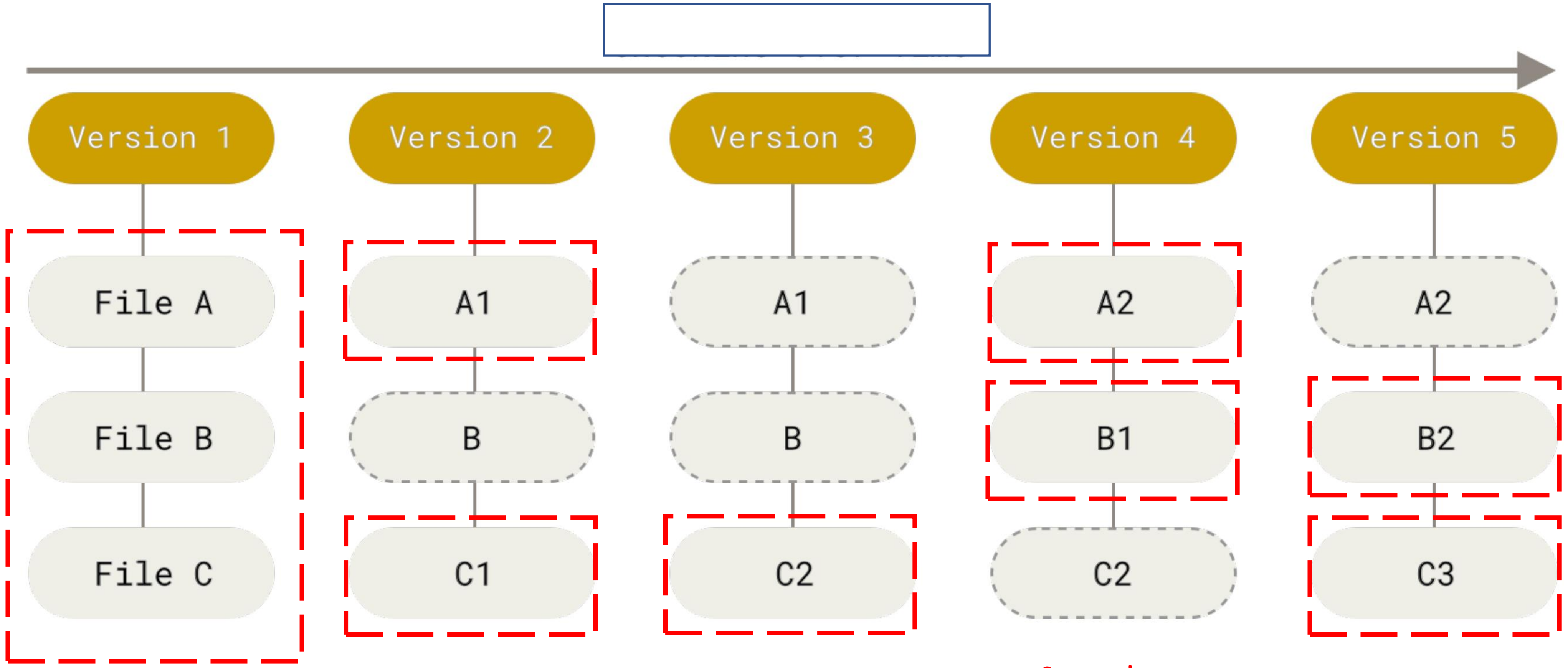
Git Project: 3 main components

- ❖Working directory / working tree
  - The directory holding the files whose revision history you want to track with Git
  - The root folder of your project where you initialize a Git repository
- ❖Git repository
  - Holds the Git objects, branch references and other related metadata
  - Stored in the .git folder in the working directory
- ❖Staging area / index
  - Temporary holding area for latest changes in the working directory that will go into the next commit

Commits / snapshots

- ❖Git allows devs to specify when the state of the project is to be saved
  - At specific points in code base lifecycle when a meaningful milestone has been reached
- ❖The stored state at specific point of time is called a commit / snapshot
  - Snapshots only contain project files whose content has changed since the previous snapshot
  - Annotated with additional metadata such as author, date and messages relevant to snapshot
- ❖Stream of commits over a duration of time is termed the commit history

Commit / snapshot history



Snapshot contains A, B and C

Snapshot contains A1 and C1

Snapshot contains C2

Snapshot contains A2 and B1

Snapshot contains B2 and C3

Branches

- Pointers to commits in the commit history
- When a repository is created, a default branch is provided
  - Typically called **master** (for local repo) or **main** (for remote repo)
- Every time a new commit is created
  - The branch pointing to the current commit is advanced to point to the new commit

git

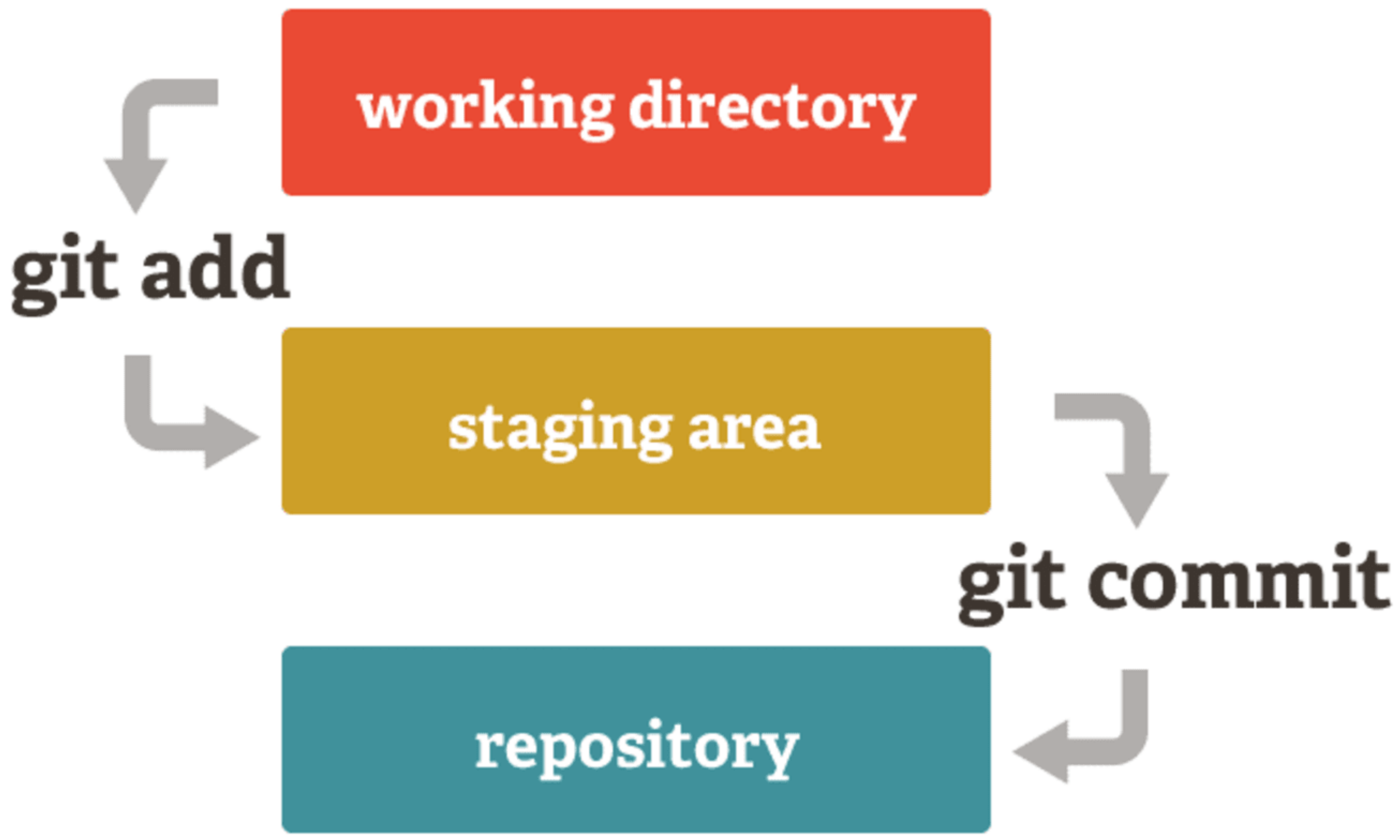
Git lab I

6 Staging and committing changes

File status in working directory

- Tracked
  - Changes to the file are tracked by the Git index
  - These changes can eventually be saved to a commit
- Untracked
  - Files for which we want Git to ignore as we do not wish to keep a commit history for them (e.g. executable binaries, config files, etc)
  - Ignored files (listed in .gitignore)
- Newly added files in the working directory are untracked by default
  - Can change them to tracked status by staging them

3 main components



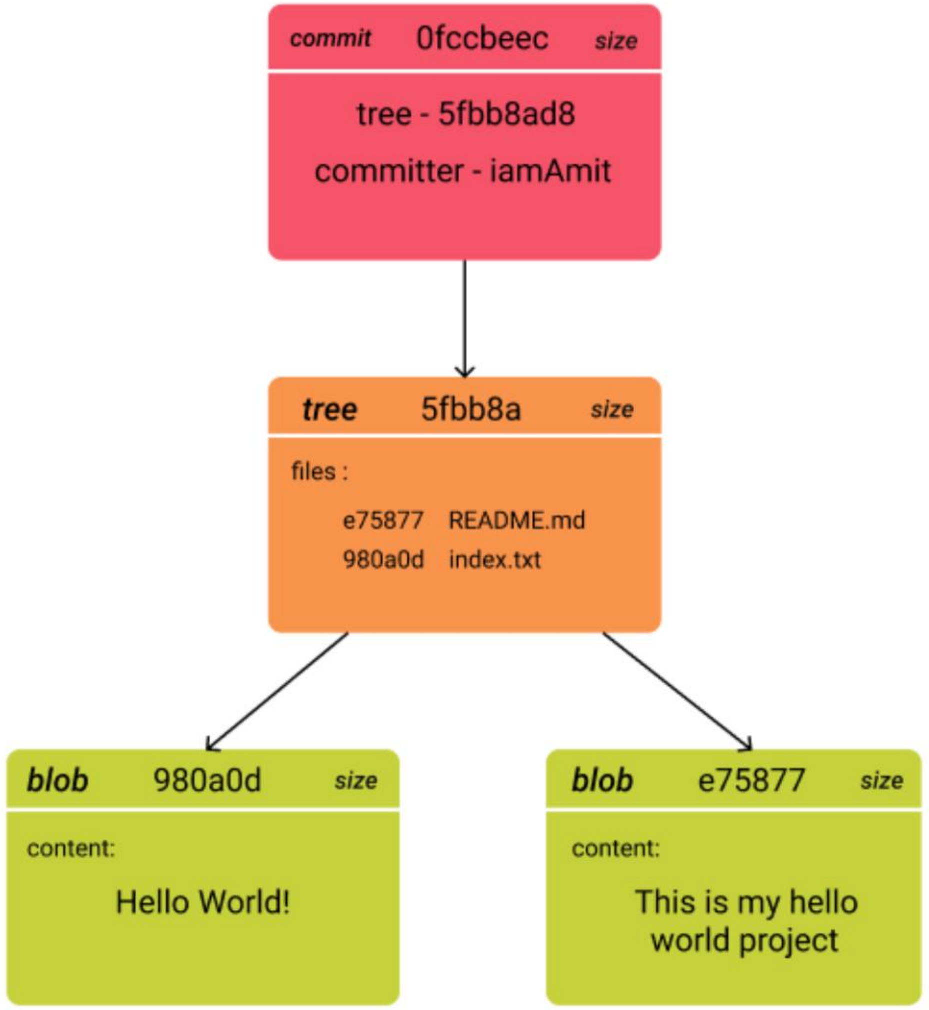
Git object model: Tree

- ❖ Represents the contents of a directory
  - references other trees (subdirectories) or blobs (files in the directory)
- ❖ Blobs within a tree have two modes
  - 100755 – File is executable by user
  - 100644 – File is not executable by user

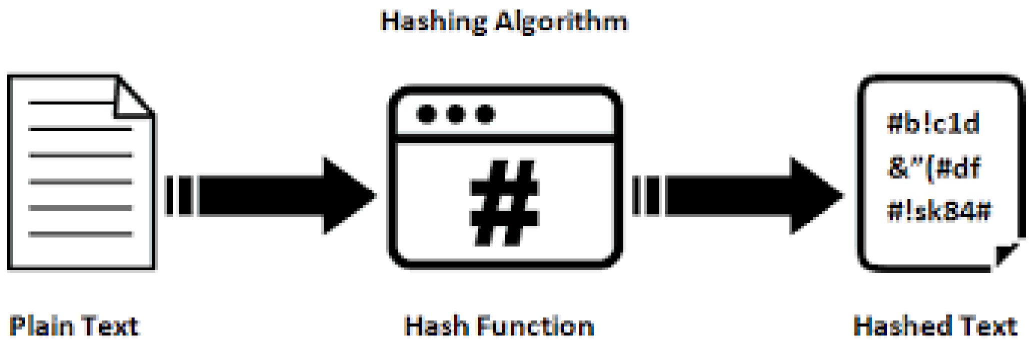
Git object model: Commit

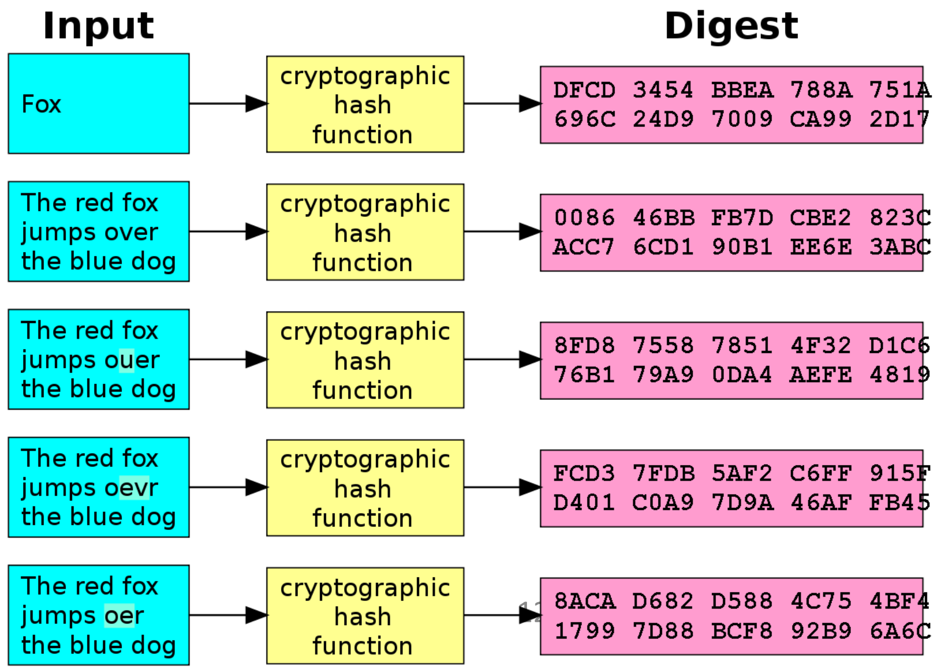
- ❖Snapshot of the project at a point in time
  - Pointer / reference to a tree
- ❖also includes additional metadata about the snapshot
  - Author – person responsible for the change
  - Committer – person who create the commit, typically the same but can be different from author
  - Commit creation date
  - Comment describing the commit

Git object model: Commit



Hash functions




Input
Digest
Fox
The red fox jumps over the blue dog
The red fox jumps ouer the blue dog
The red fox jumps oevr the blue dog
The red fox jumps oer the blue dog
cryptographic hash function
DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17
0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC
8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819
FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45
8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C

SHA-1 hash name

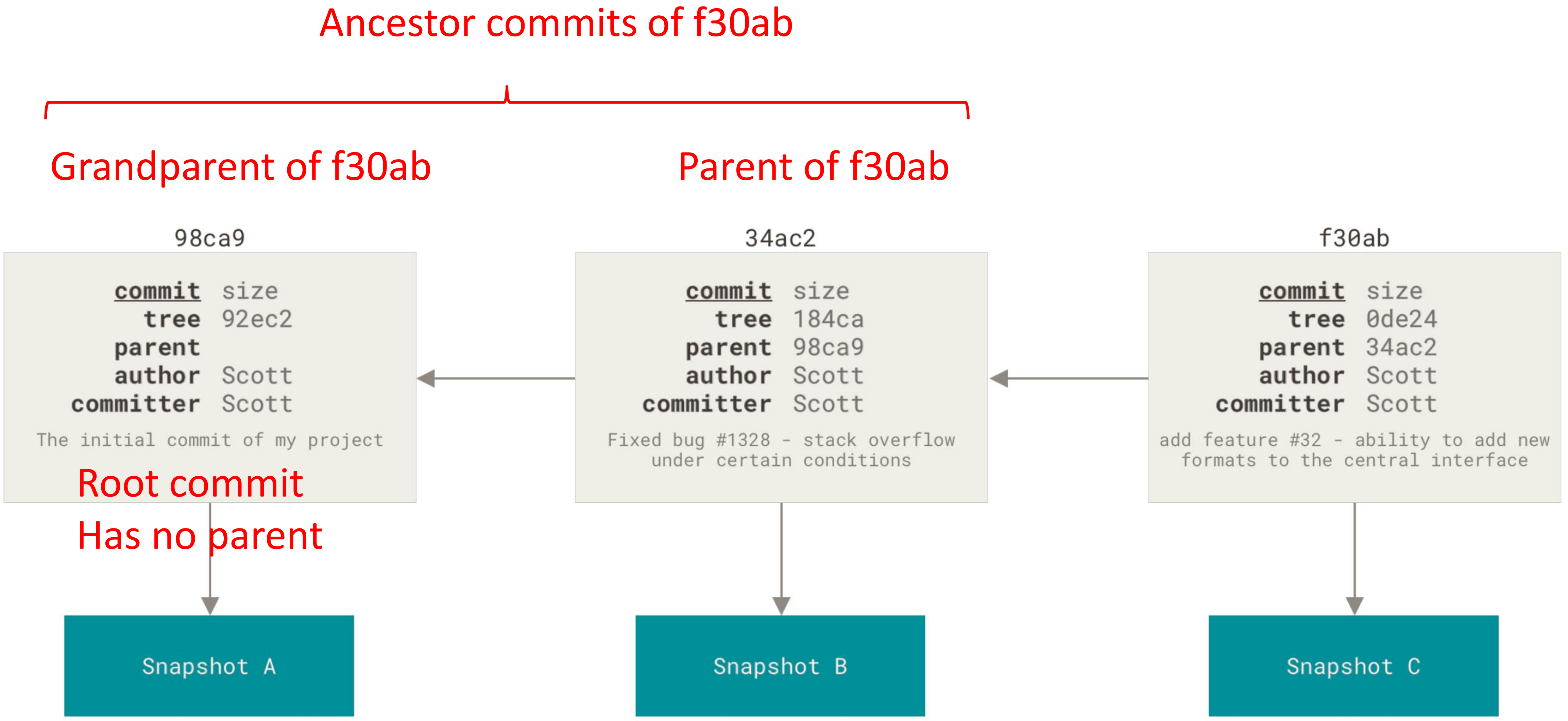
- All objects (blob, tree, commit, tag) have a unique name
  - This is a 40-digit SHA-1 hash of their contents
  - E.g. 6ff87c466…..
- Advantages
  - Determine whether two objects are identical or not by comparing their names - objects with identical content in different repositories have exactly the same name
  - Check for integrity by comparing a computed hash of object with its name

git

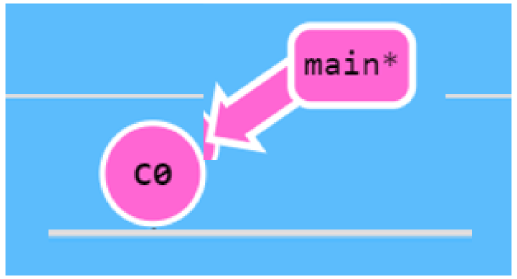
Git lab I

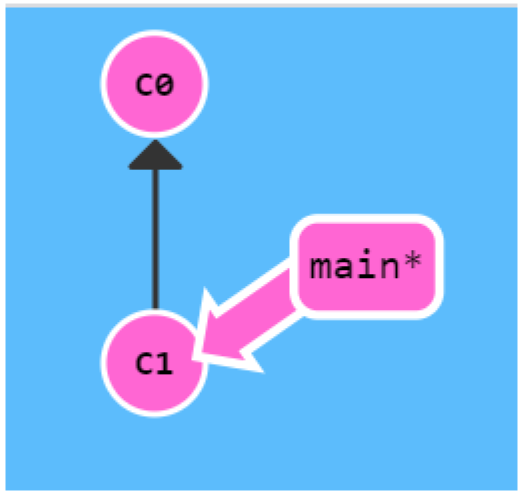
7 Tracking changes

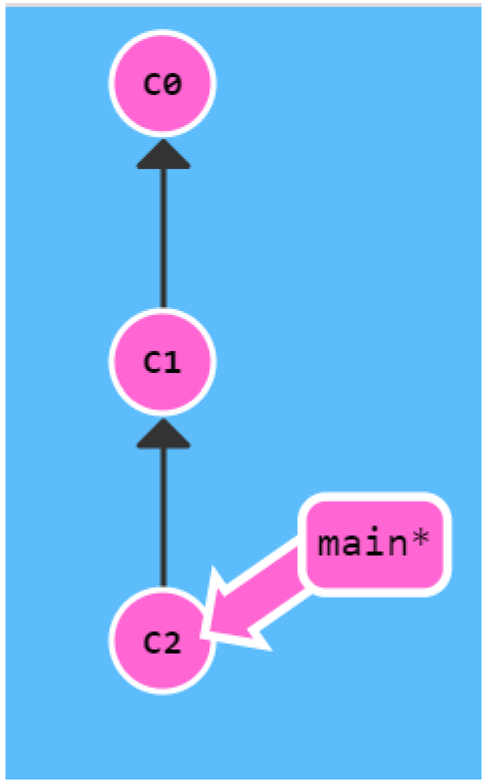
Commit history timeline



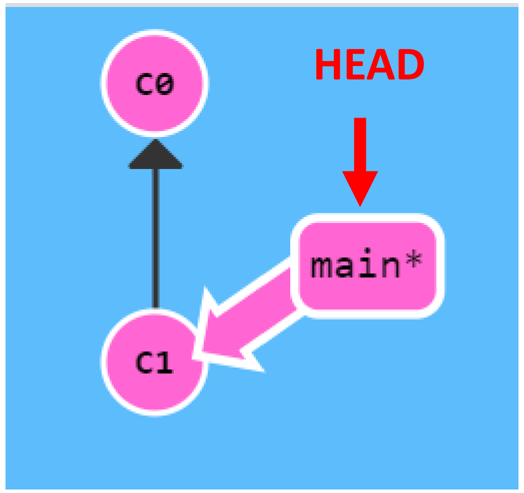
Moving branch to point to latest commit

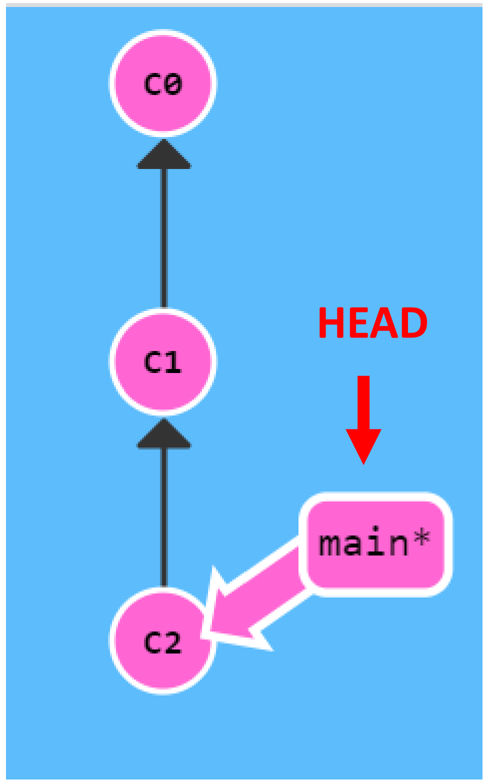


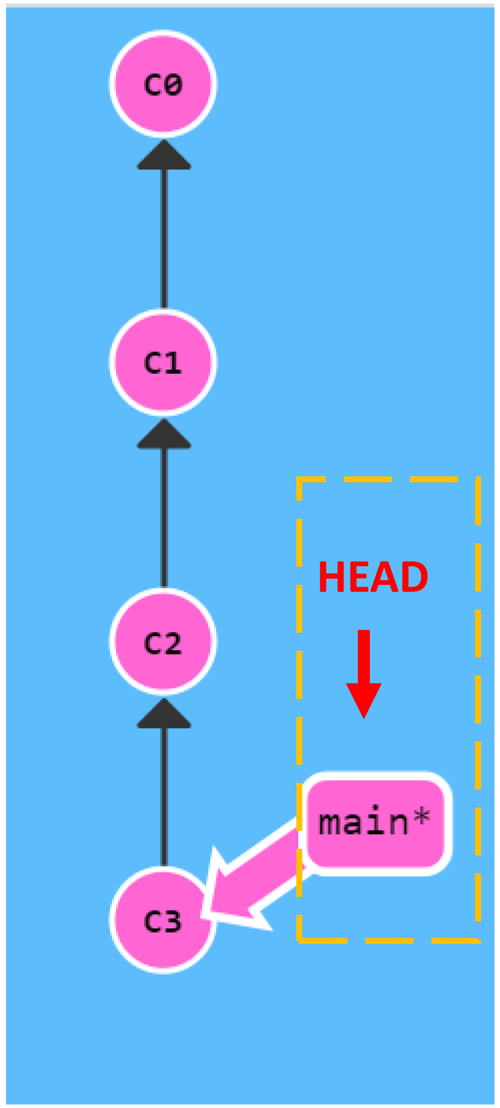




Using HEAD to track branch movement







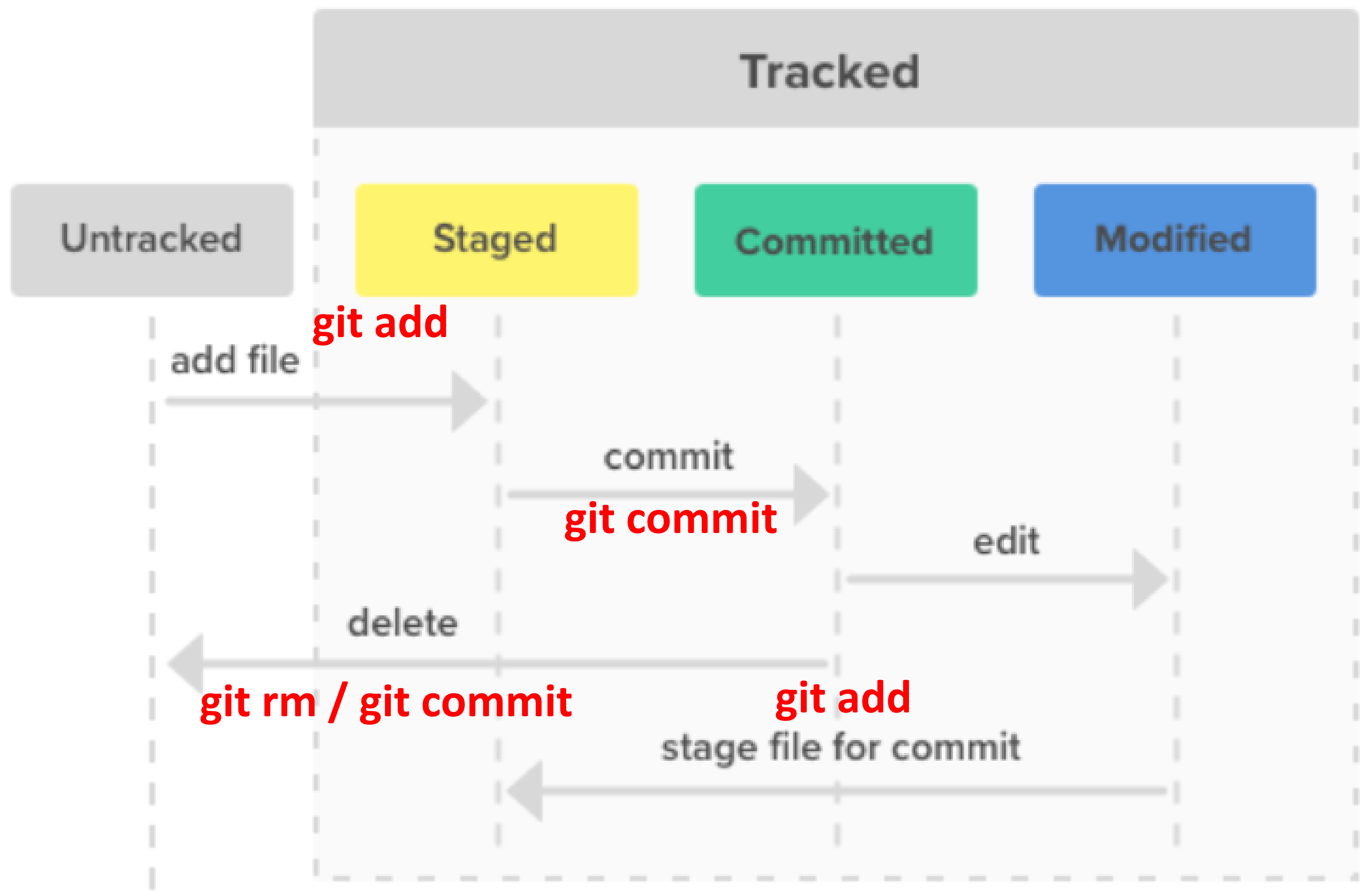
Git log

Victor Tan 2022

Git lab I

8 Deleting and renaming tracked files

git add, commit, rm, status



Check for status of staged / unstaged with

git status