

VCS-R

Version Control System in Rust

Lawson Oliveira Lima
Lucas Vitoriano De Queiroz Lira

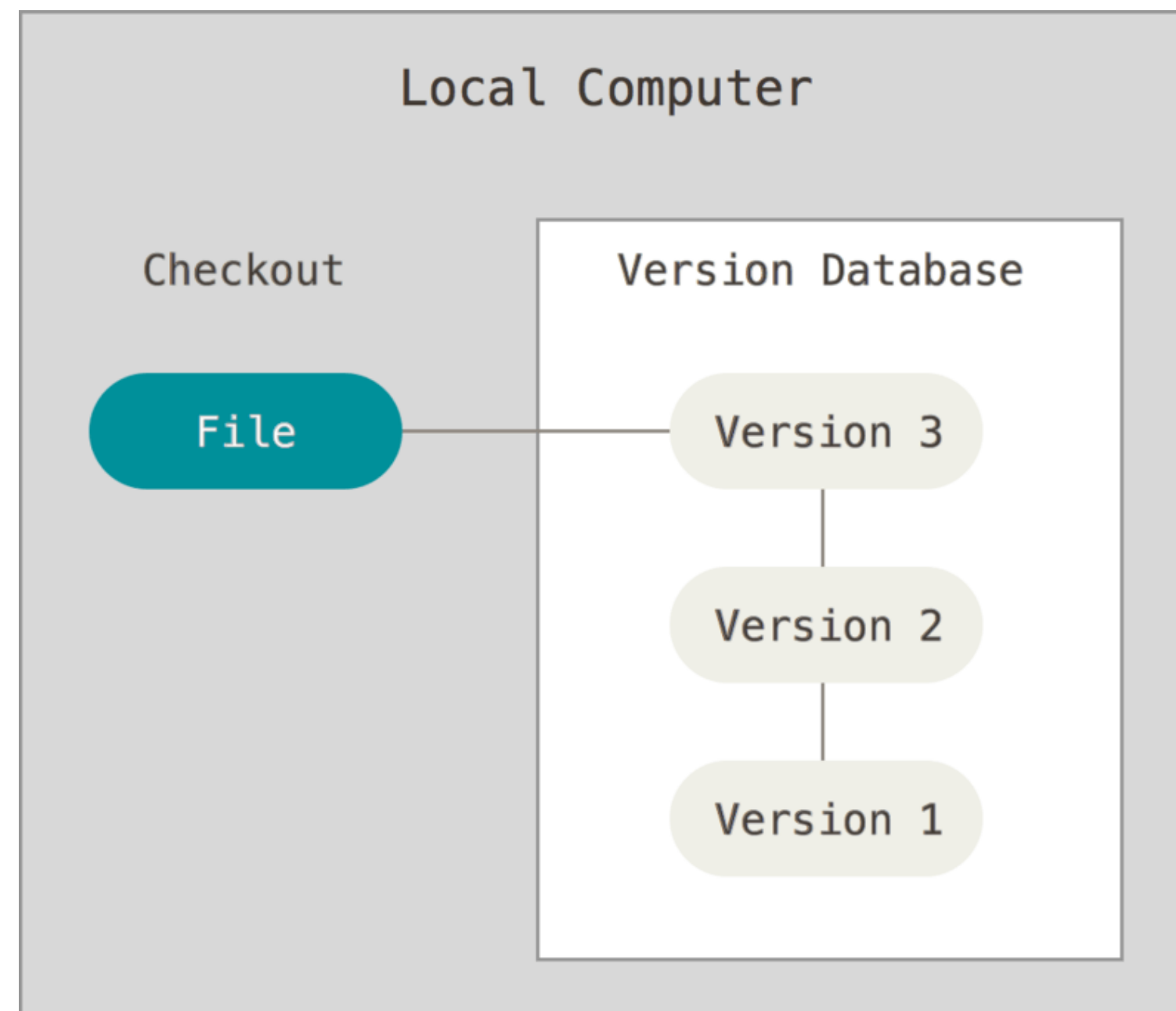
CONTENT

- 01 What it is
- 02 Objective
- 03 Why Rust
- 04 Structs
- 06 Add / Remove
- 07 Commit
- 08 Version
- 09 Branch
- 10 Demonstration

WHAT IS ?



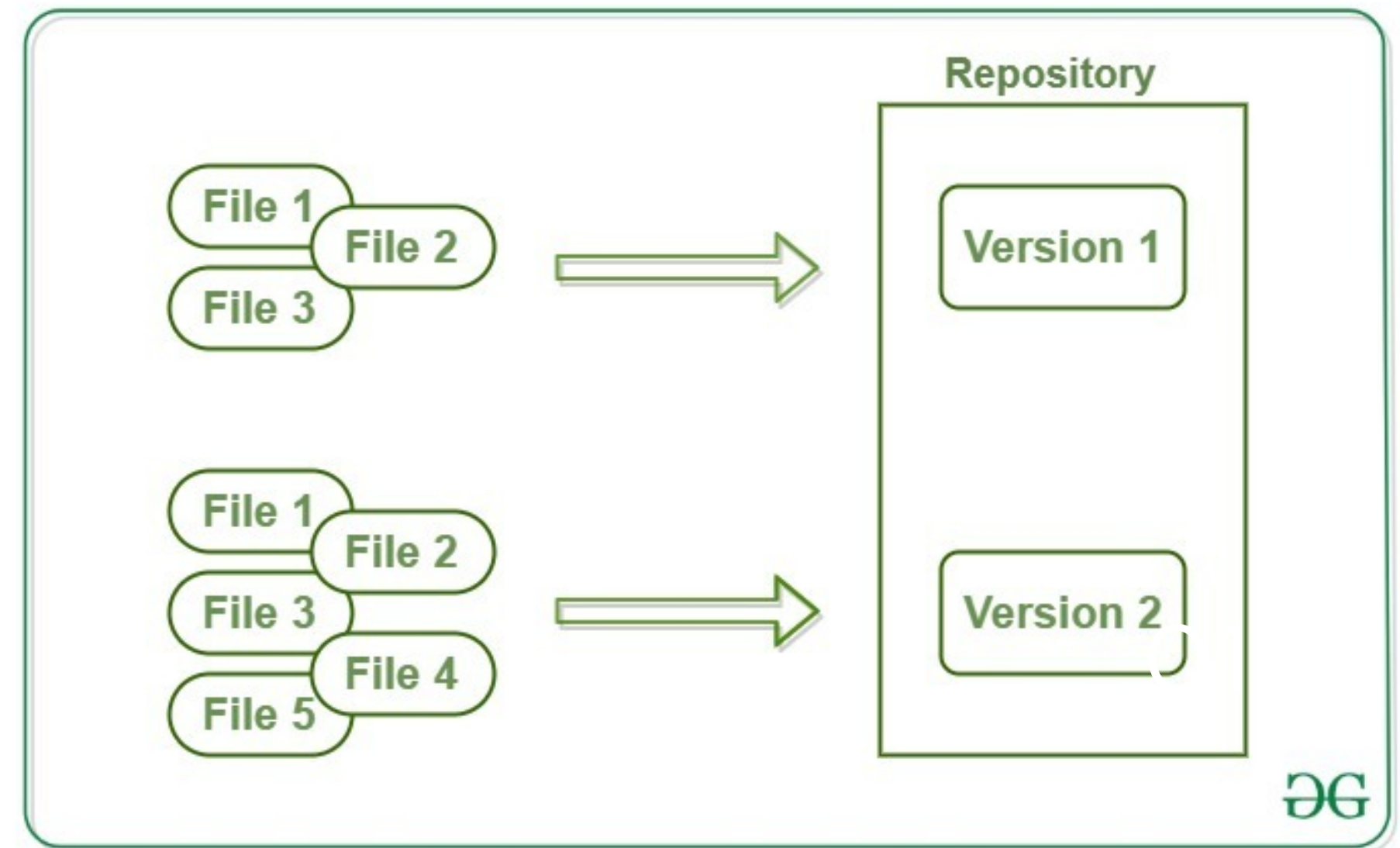
git



OBJECTIVE

SYSTEM CAPABLE OF GERENCIATE A REPOSITORY

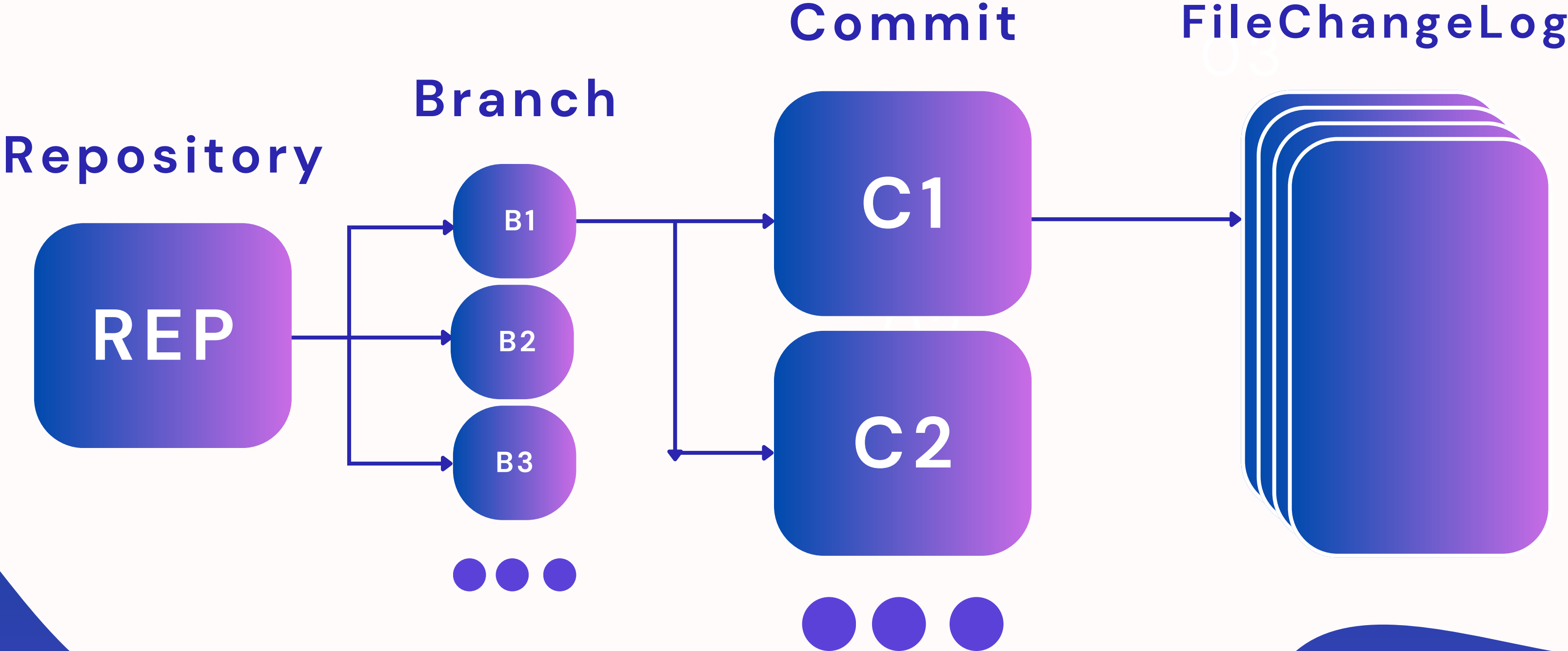
- Create a repository**
- Save a version of the file (commit)**
- Delete commit**
- Change the actual Version**
- Manage Branches**



WHY RUST ?



STRUCT



STRUCTS

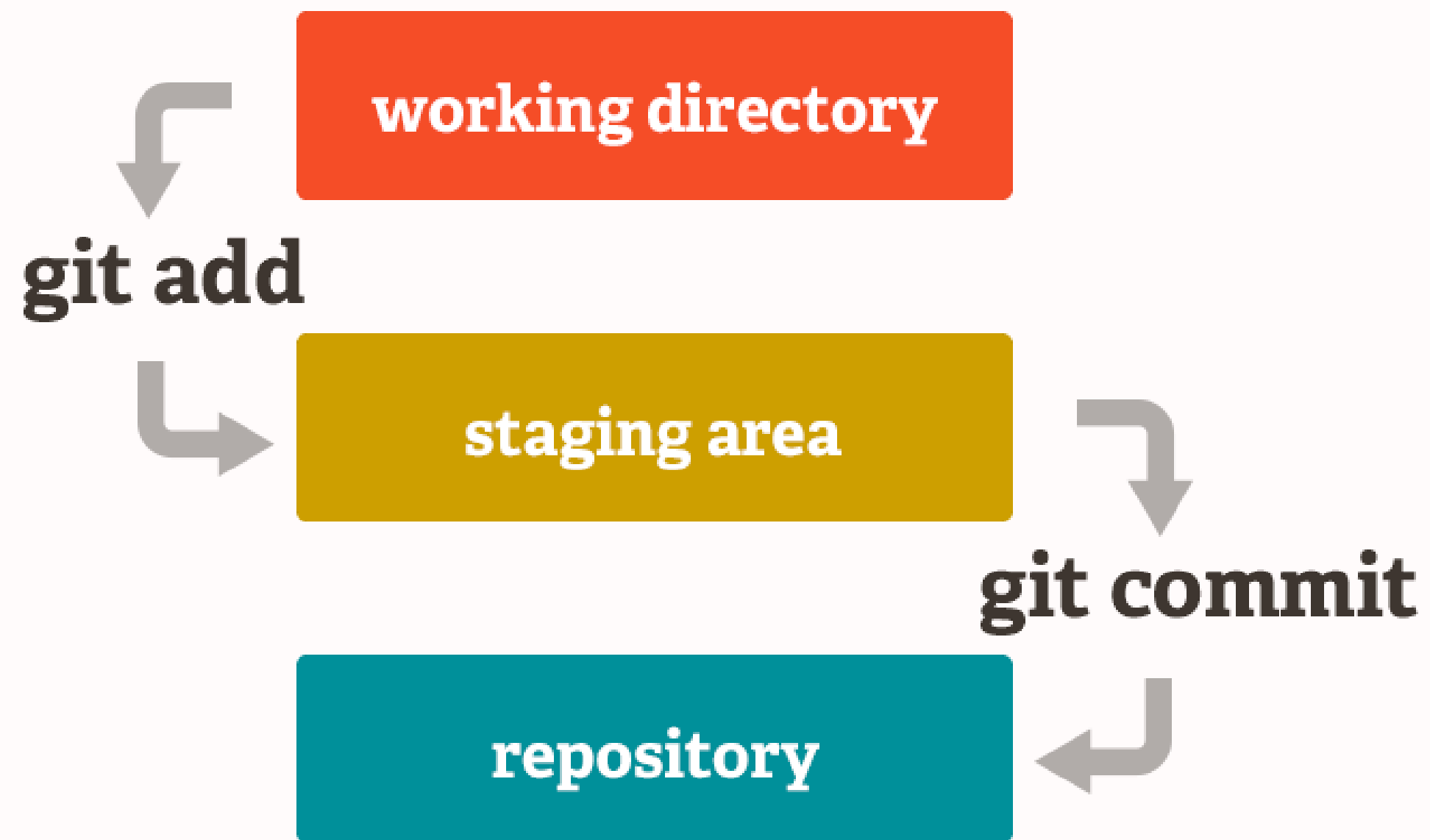
```
#[derive(Debug, Serialize, Deserialize)]
3 implementations
pub struct Branch {
    pub branch_name: String,
    pub head_commit_hash: String,
    pub commits: Vec<Commit>,
}

#[derive(Serialize, Deserialize)]
2 implementations
pub struct Repository {
    pub current_branch: String,
    pub branches: Vec<Branch>,
}
```

```
#[derive(Debug, Clone, Serialize, Deserialize)]
4 implementations
pub struct FileChangeLog {
    pub original_file_path: String,
    pub original_file: String,
    pub last_file: String,
    pub last_file_path: String,
    pub hash_changelog: String,
    pub hash_files_path: String,
    pub version: u32,
    pub parent_version: u32,
}

#[derive(Debug, Clone, Serialize, Deserialize)]
4 implementations
pub struct Commit {
    pub files_changelogs: Vec<FileChangeLog>,
    pub commit_hash: String,
    pub parent_commits: Vec<String>,
}
```

ADD / REMOVE



HOW IT WORKS ?

Add

Write the file name and file hash in staging_area.yml



Copy the file to add_contents

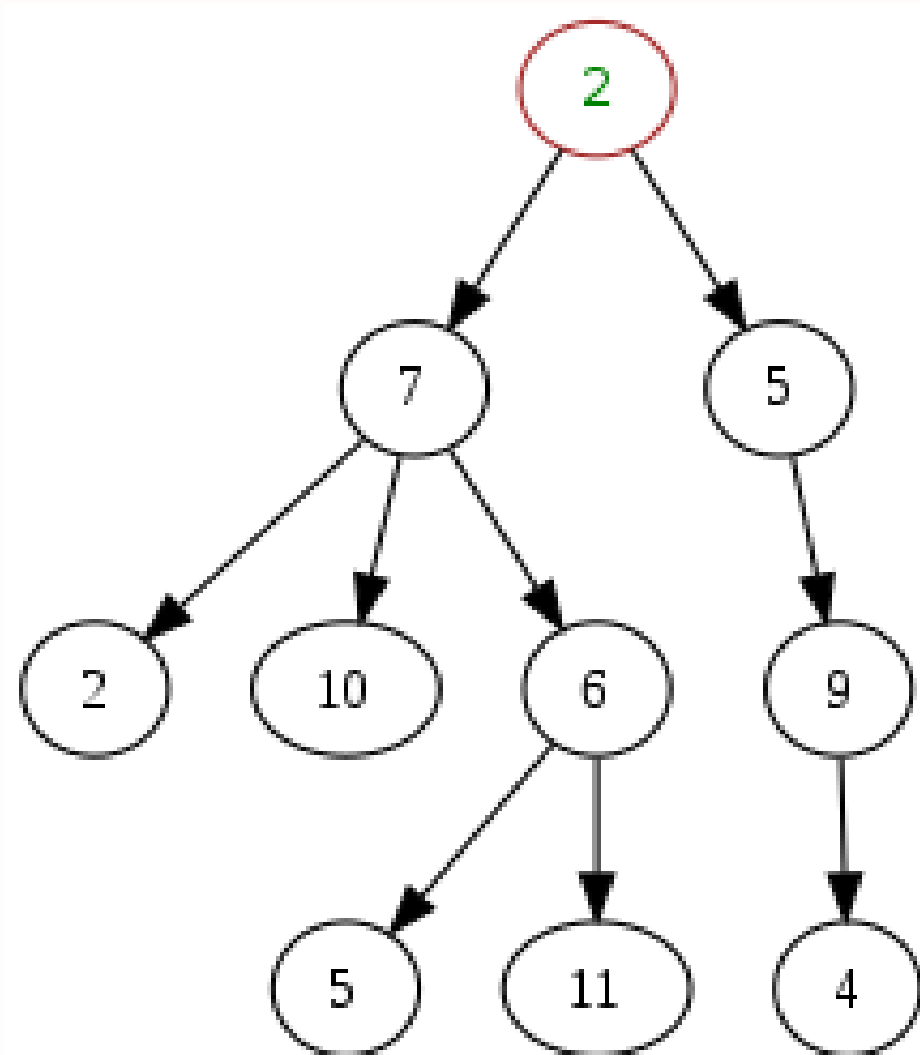
Remove

Erase the file name and file hash in staging_area.yml



Delete the file from add_contents

COMMIT



HOW IT WORKS ?

Commit

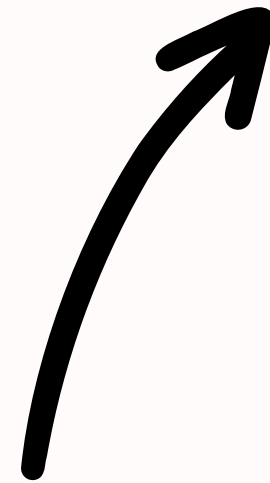
Hash of commit header



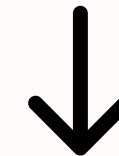
Build commit tree



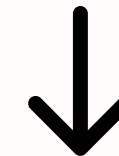
Build FileChangeLog tree for
each file in the staging area



Build a version tree for each file
in the staging area

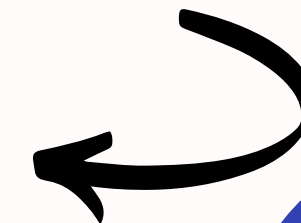


Apply each patch and finally
create the new diff file

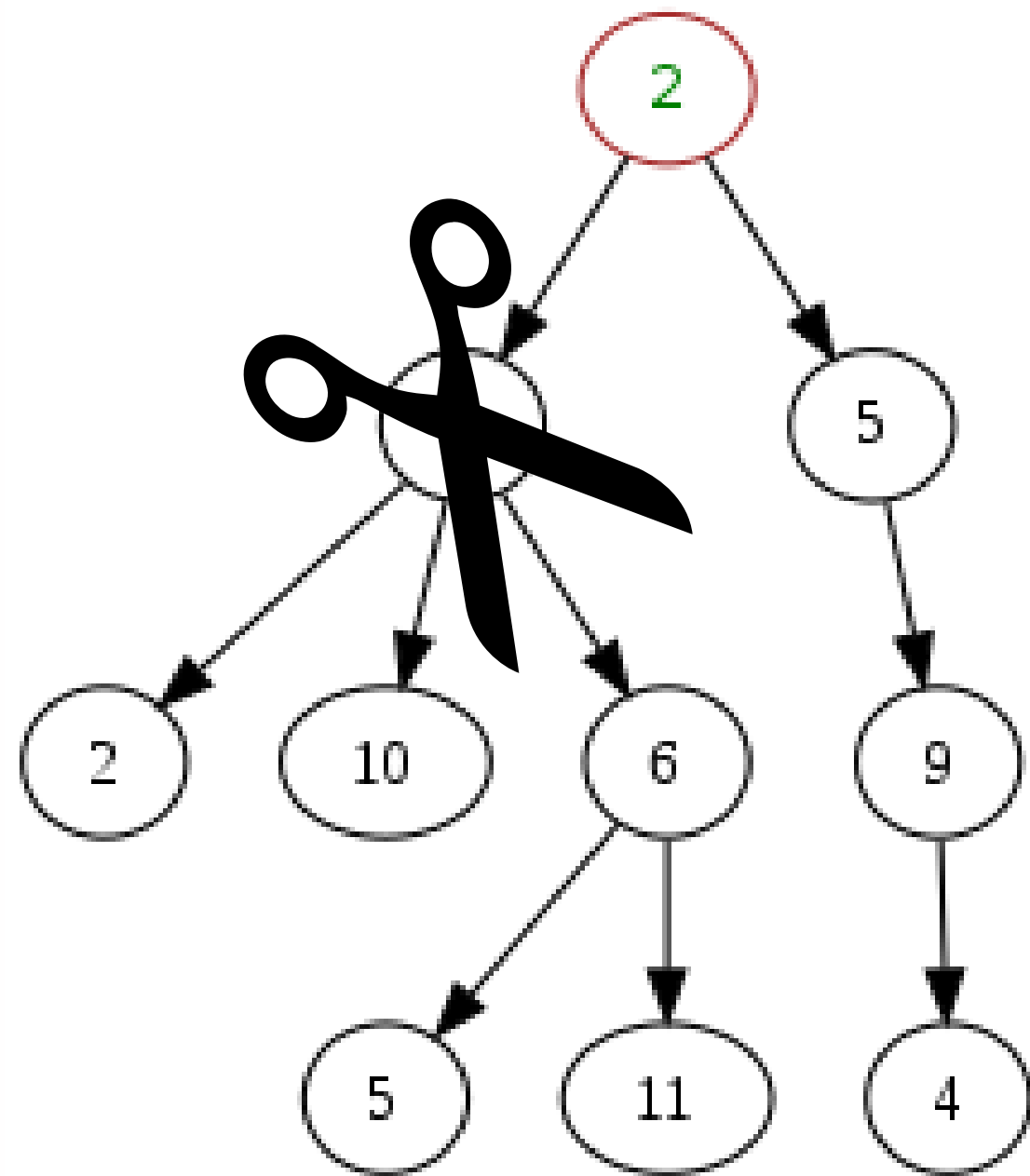


Save the diff in .saves using the
hash file as name

Remove files from staging area



DELETE COMMIT



HOW IT WORKS ?

Delete commit

Hash of the commit to delete



Build commit tree



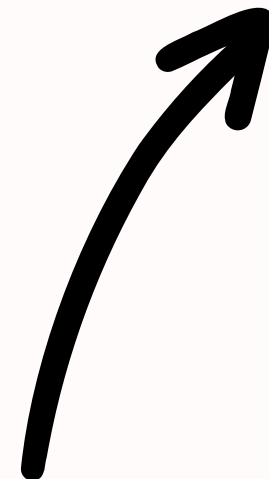
Delete the patches that are
after the desired commit



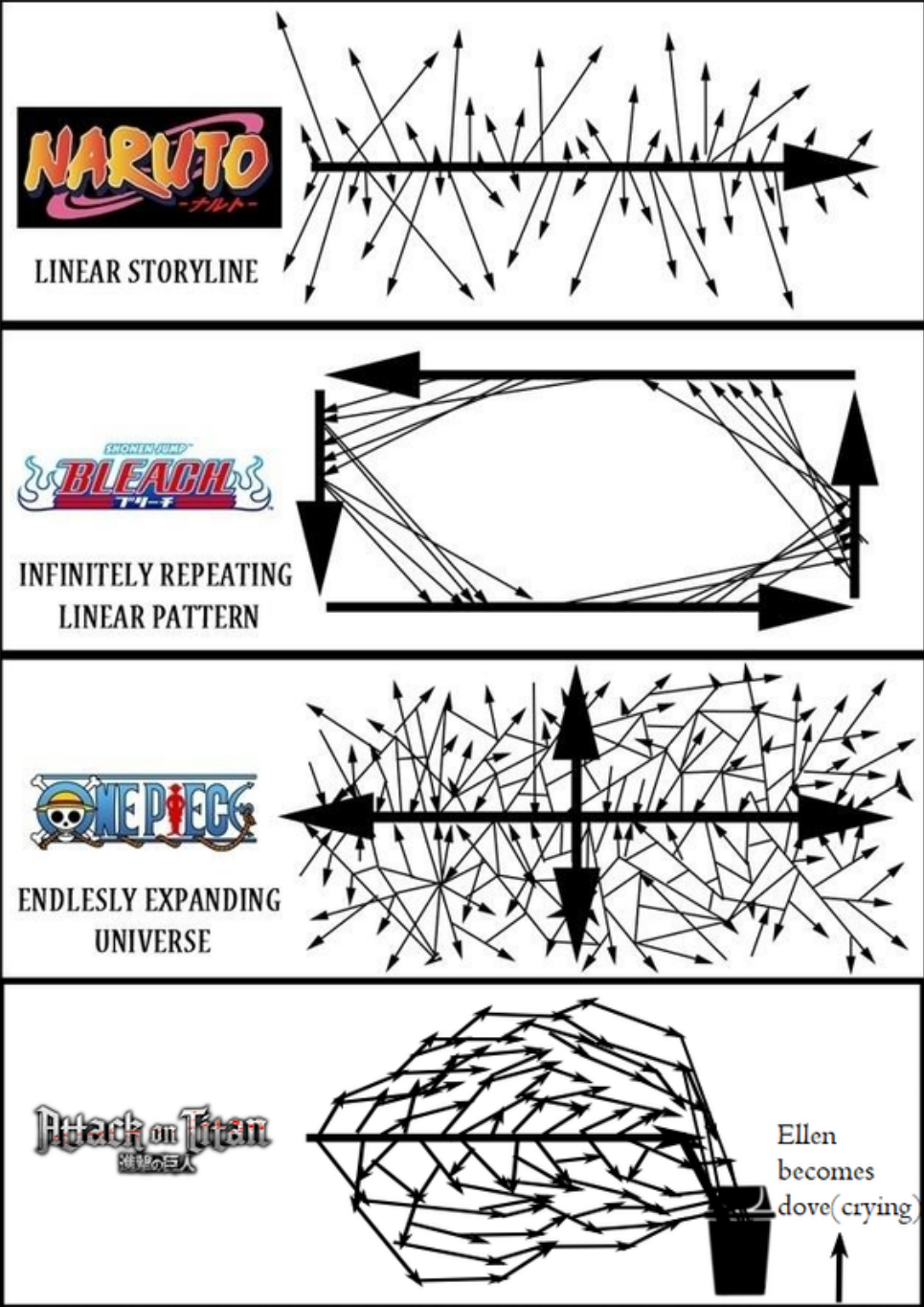
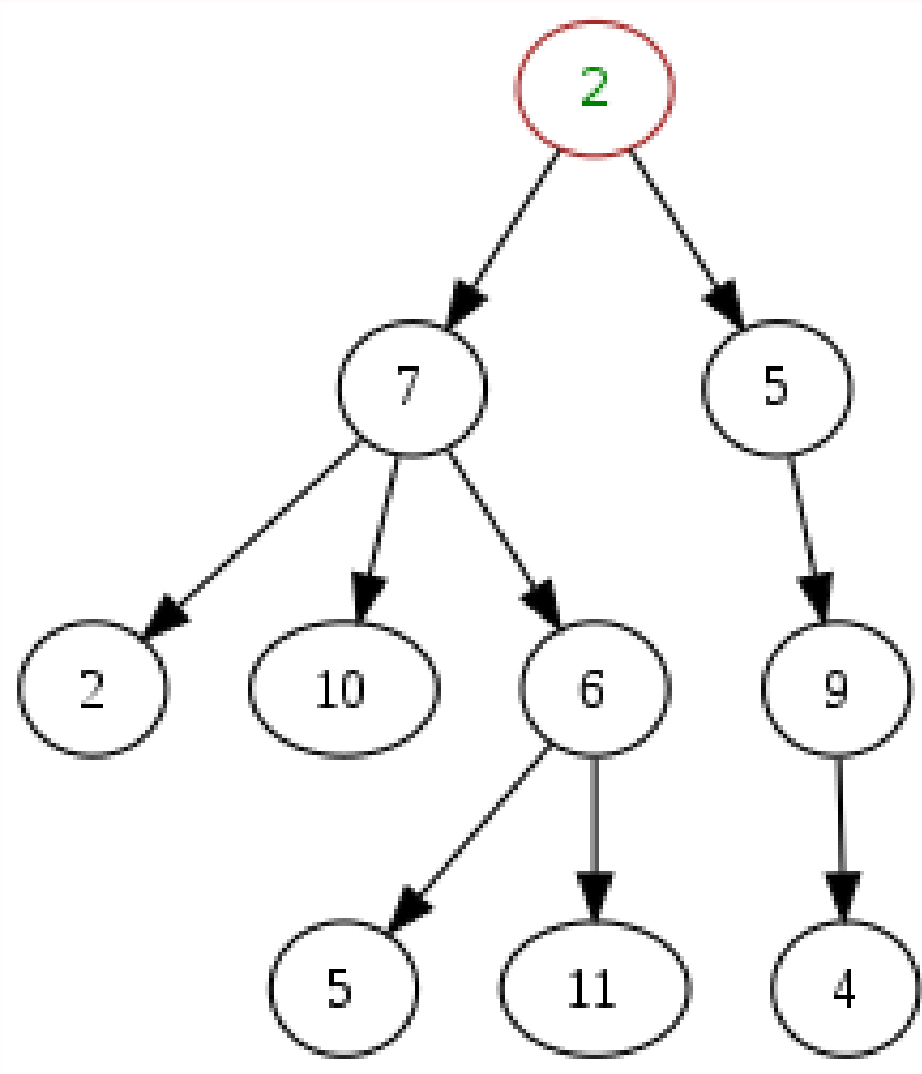
Replace the current commit tree



Update the struct file



CHANGE VERSION



CHANGE VERSION

Commit hash

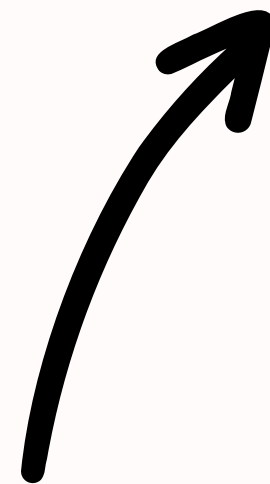
Desired version



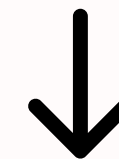
**Build commit tree until the
root version**



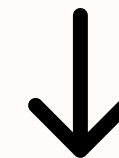
**Build FileChangeLog tree for
each file in commit tree**



**Build a version tree for each file
in the commit tree**

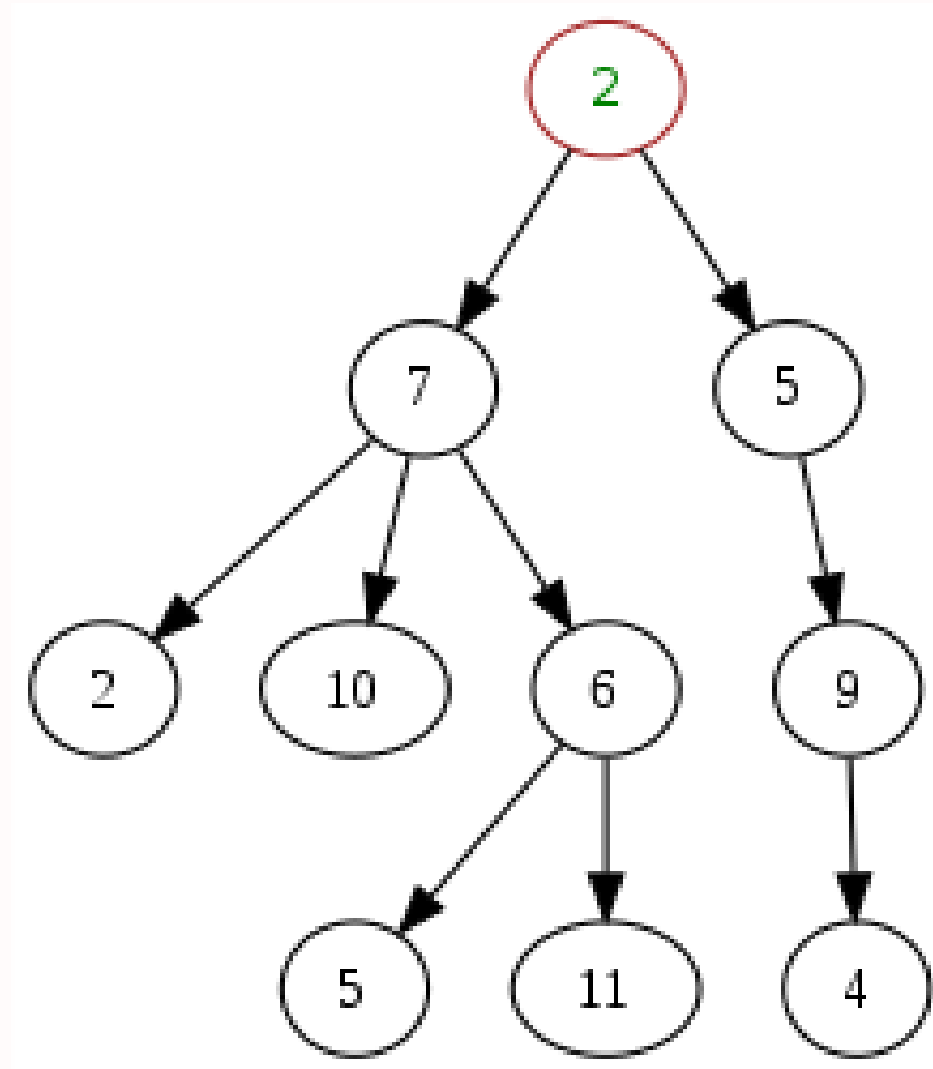


**Apply each patch to the
correspondent file**

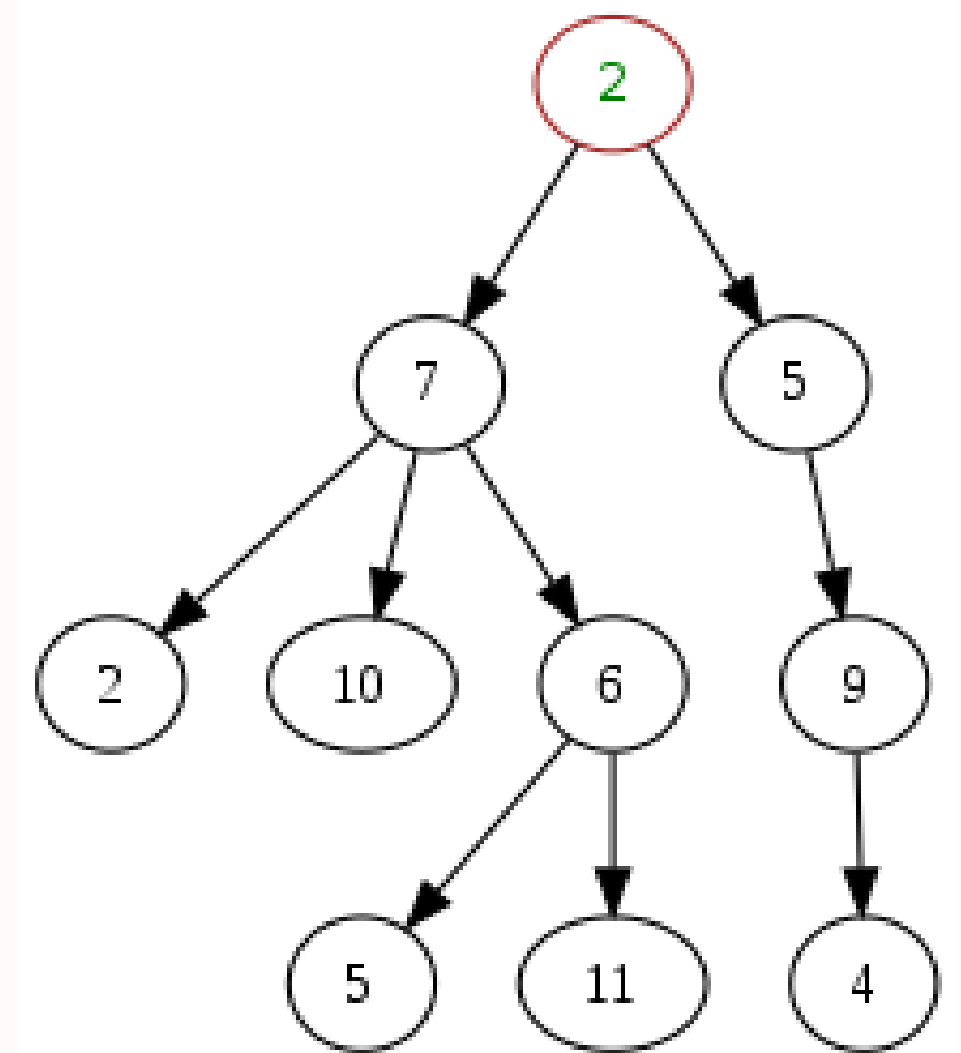
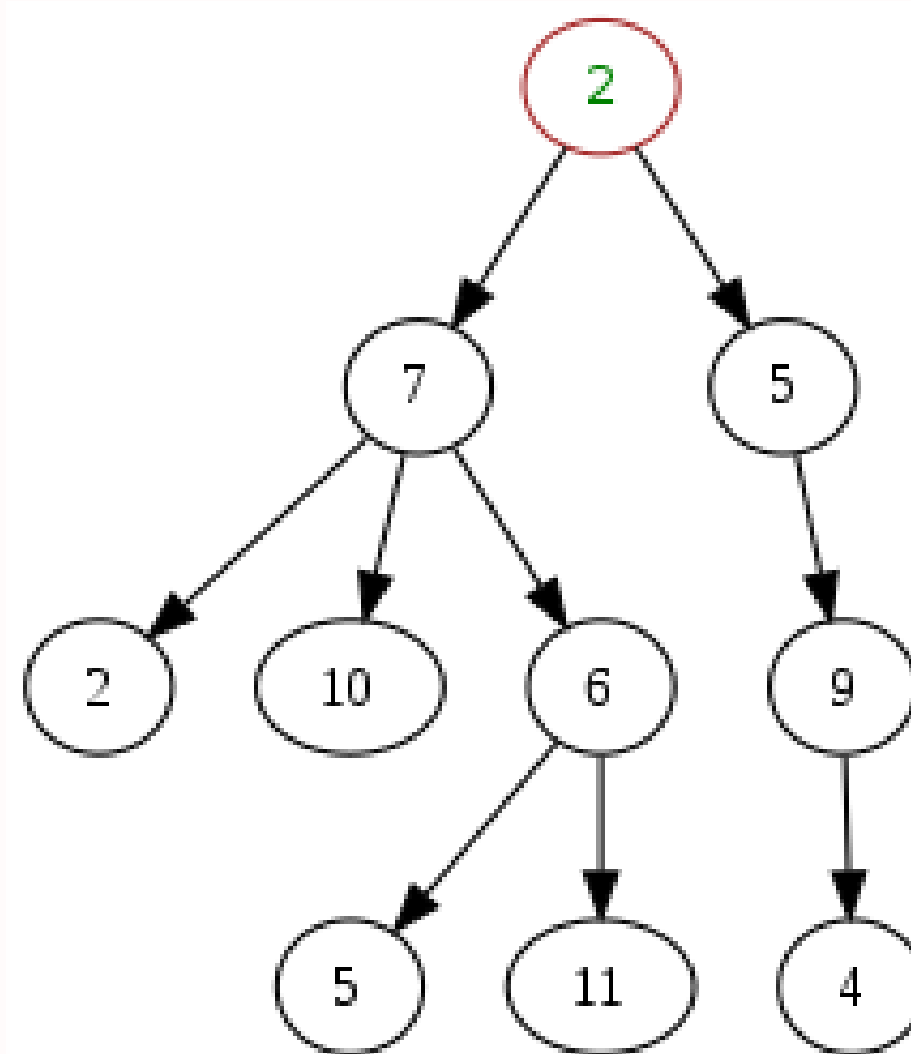
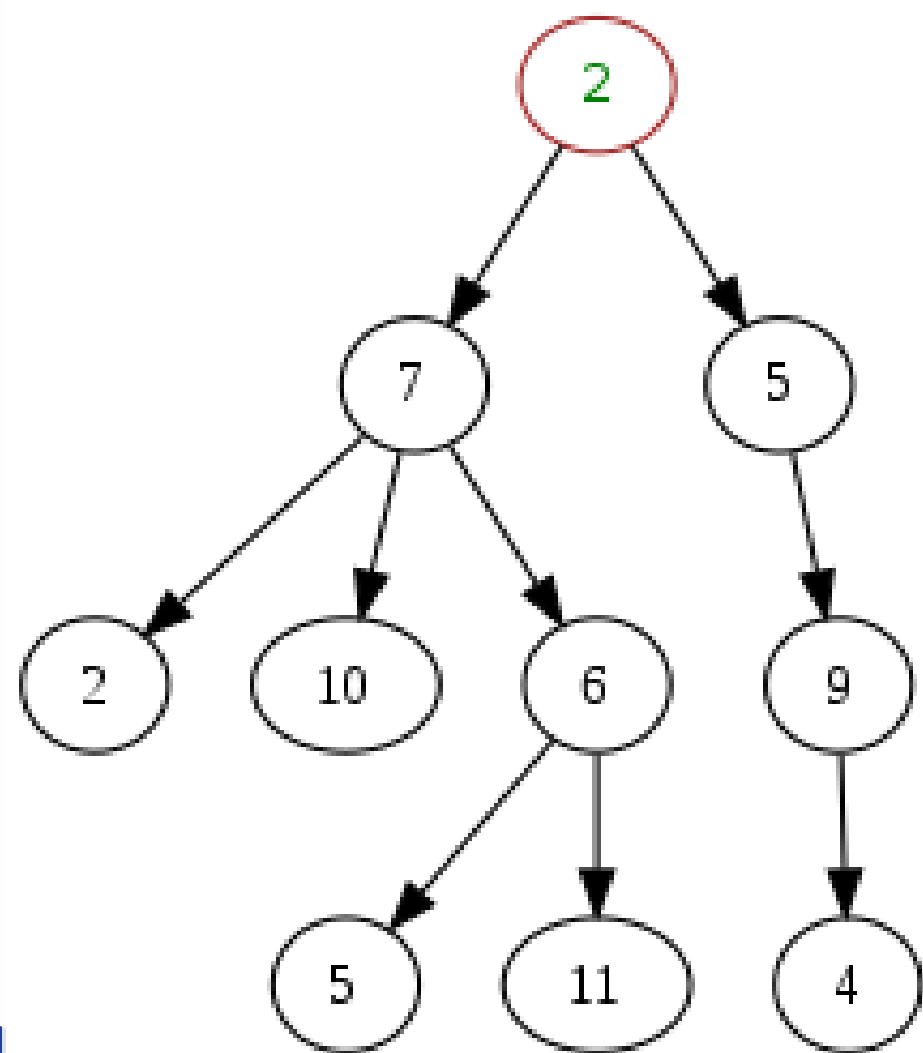


Apply the result for each file

CHANGE VERSION



BRANCH



CREATE BRANCH

```
#[derive(Debug, Serialize, Deserialize)]
3 implementations
pub struct Branch {
    pub branch_name: String,
    pub head_commit_hash: String, // Added: Hash
    pub commits: Vec<Commit>, // Added: List of c
}
```

```
#[derive(Serialize, Deserialize)]
2 implementations
pub struct Repository {
    pub current_branch: String,
    pub branches: Vec<Branch>,
}
mod structs mod
```

Duplicate the current branch

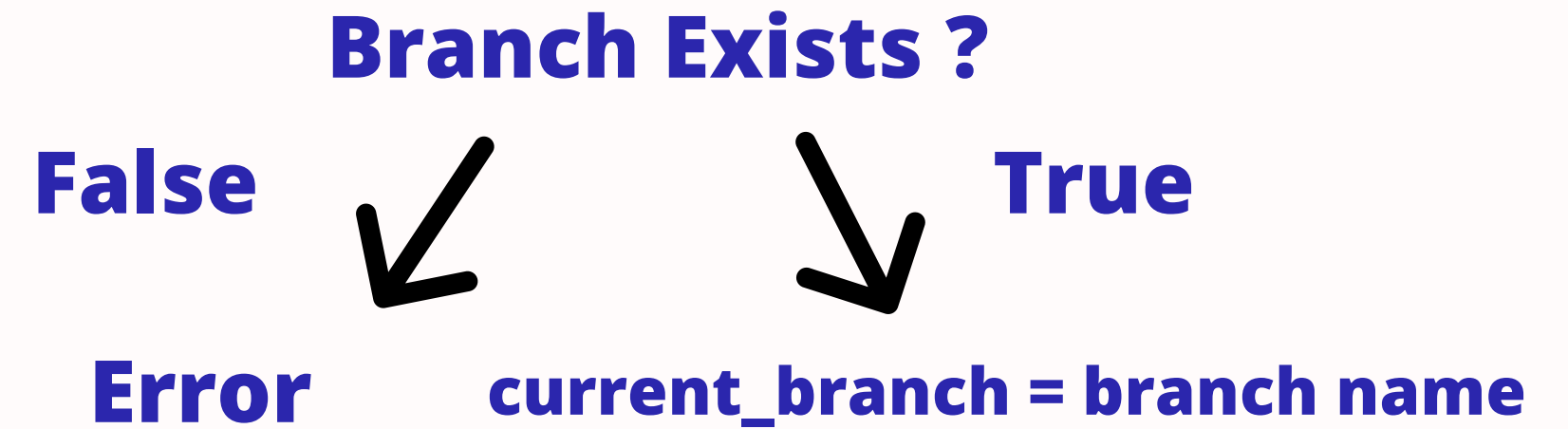


push in the branches vector

CHANGE BRANCH

```
#[derive(Serialize, Deserialize)]
2 implementations
pub struct Repository {
    pub current_branch: String,
    pub branches: Vec<Branch>,
}
```

mod structs mod



DELETE BRANCH

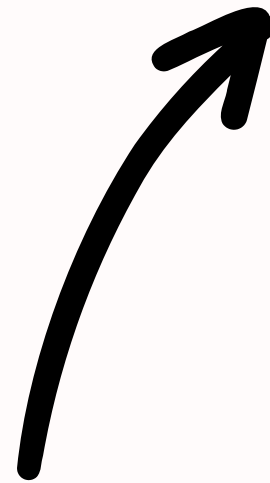
Runs over the branch to delete



for each file hash :
files_hashes.push(file_hash)



for file_hash in files_hash:
for each other branch:



if another branch contains file
hash

False



Delete file

True



nothing



delete branch struct from
repository

```
#[derive(Serialize, Deserialize)]
2implementations
pub struct Repository {
    pub current_branch: String,
    pub branches: Vec<Branch>,
}
```

The background is a solid blue color. It features several organic, wavy shapes in a bright yellow color. One large yellow shape is in the top-left corner, and another is in the bottom-right corner. There are also three small yellow circles: one in the top-left area, one in the bottom-center area, and one in the bottom-right area. A large, semi-transparent blue circle is located in the top-right corner.

Demonstration