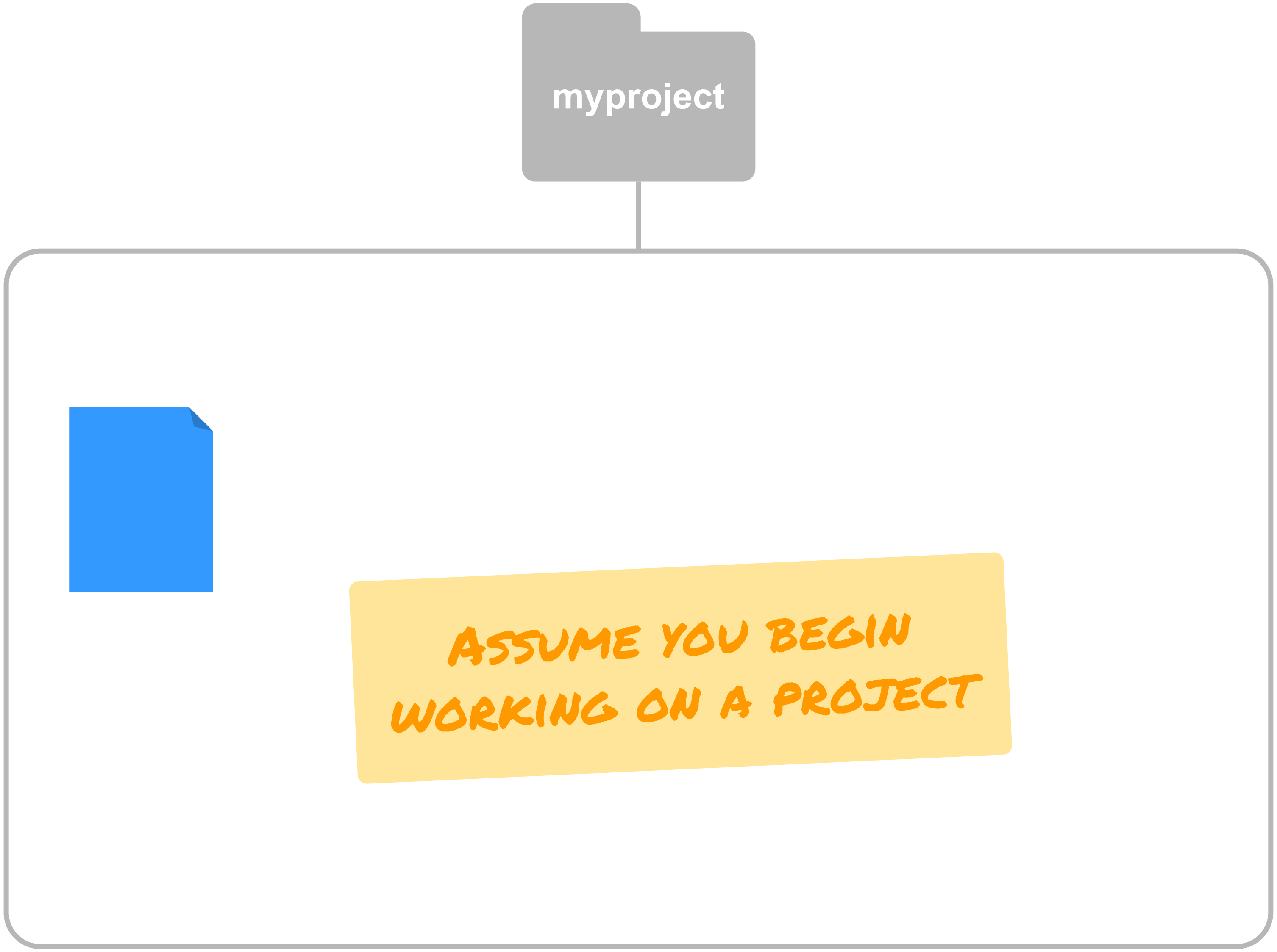
Introduction to Git

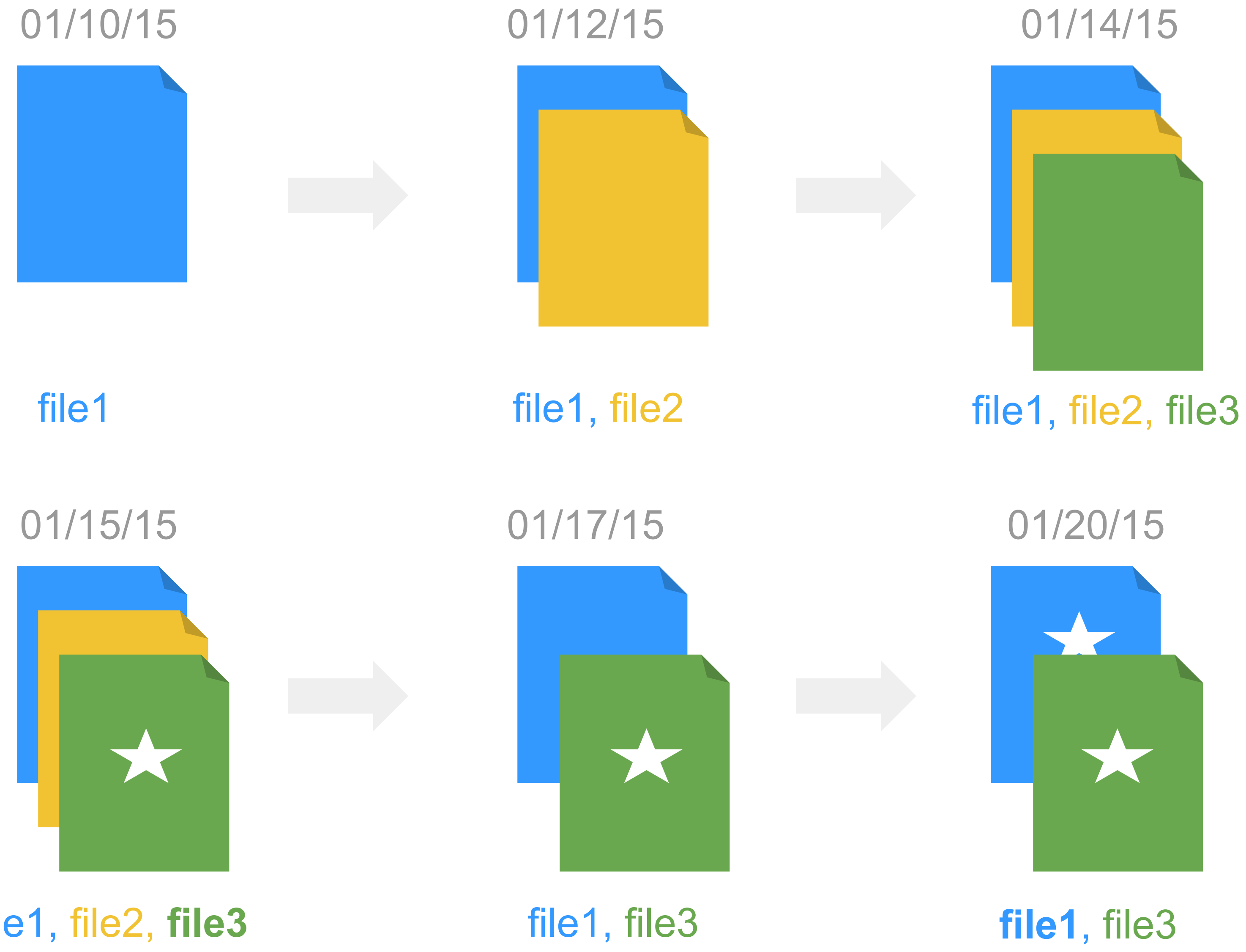
Stat 133 with Gaston Sanchez

project’s directory

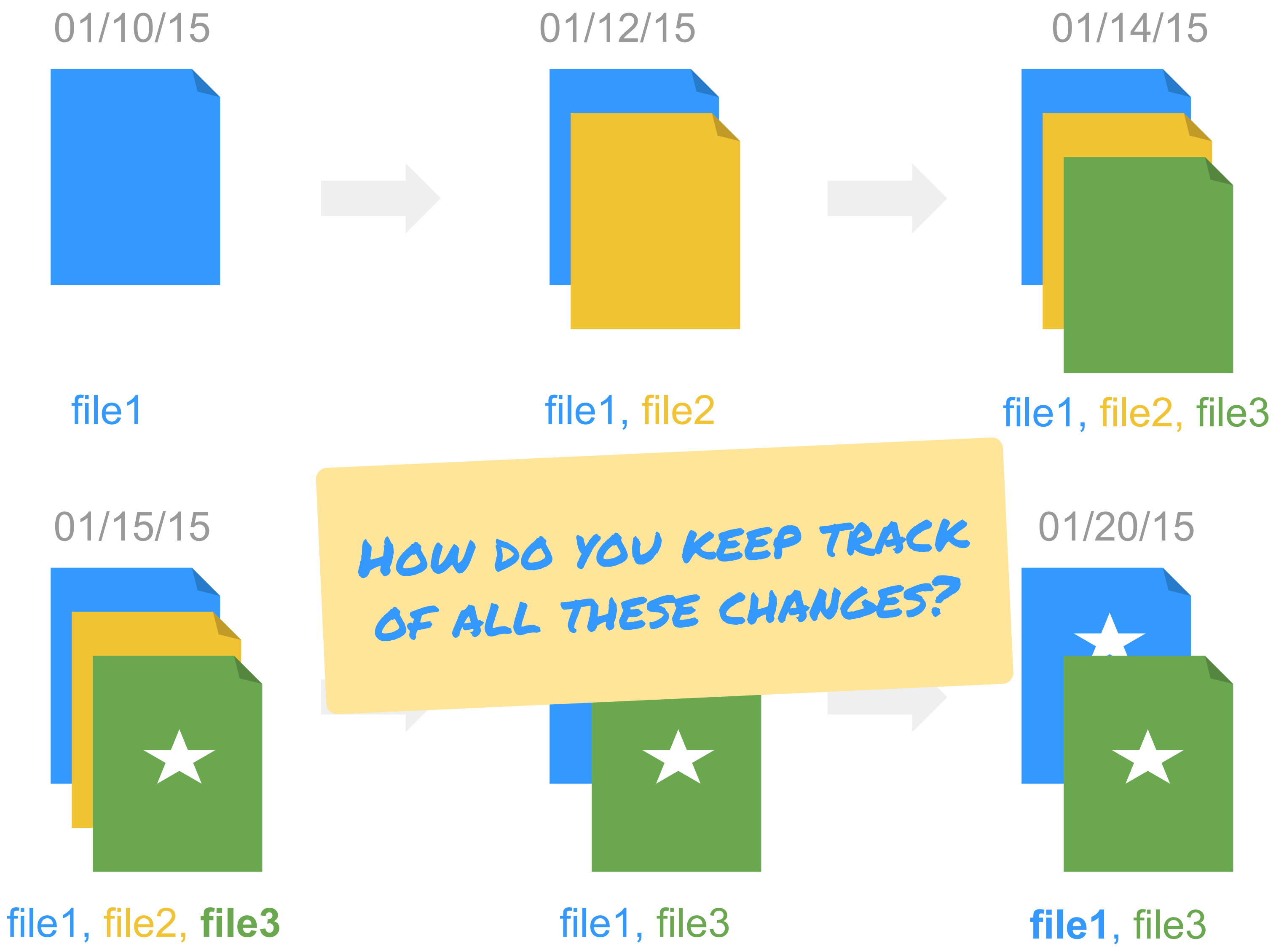


ASSUME YOU BEGIN WORKING ON A PROJECT

myproject

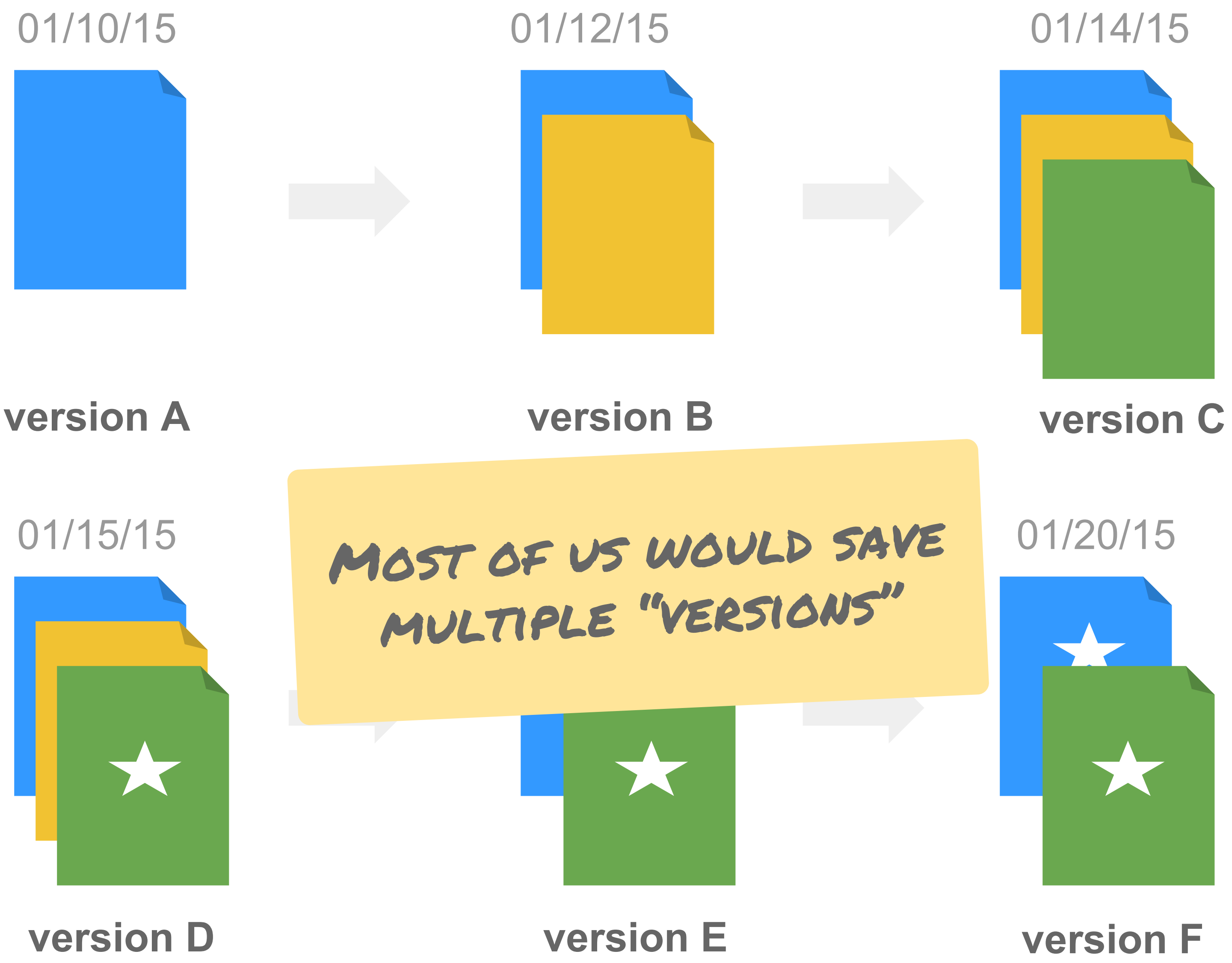

01/10/15
file1
01/12/15
file1, file2
01/14/15
file1, file2, file3
01/15/15
file1, file2, **file3**
01/17/15
file1, file3
01/20/15
file1, file3

myproject

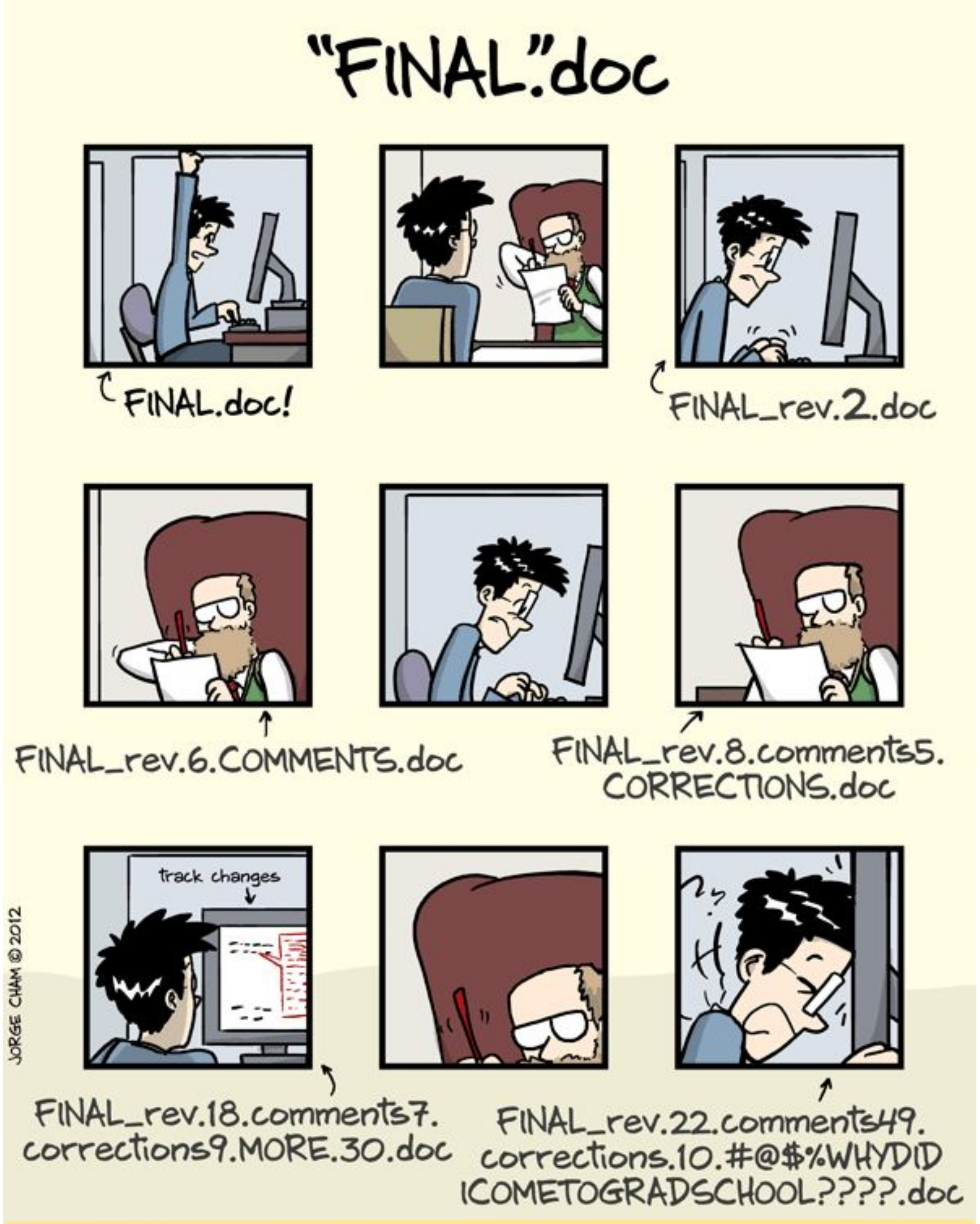


HOW DO YOU KEEP TRACK OF ALL THESE CHANGES?

myproject



MOST OF US WOULD SAVE MULTIPLE "VERSIONS"



Savings multiple "versions" is highly inefficient

Key Ideas

Keep a record of all the made changes

Storing changes of each version

Git is a Version Control System (VCS)

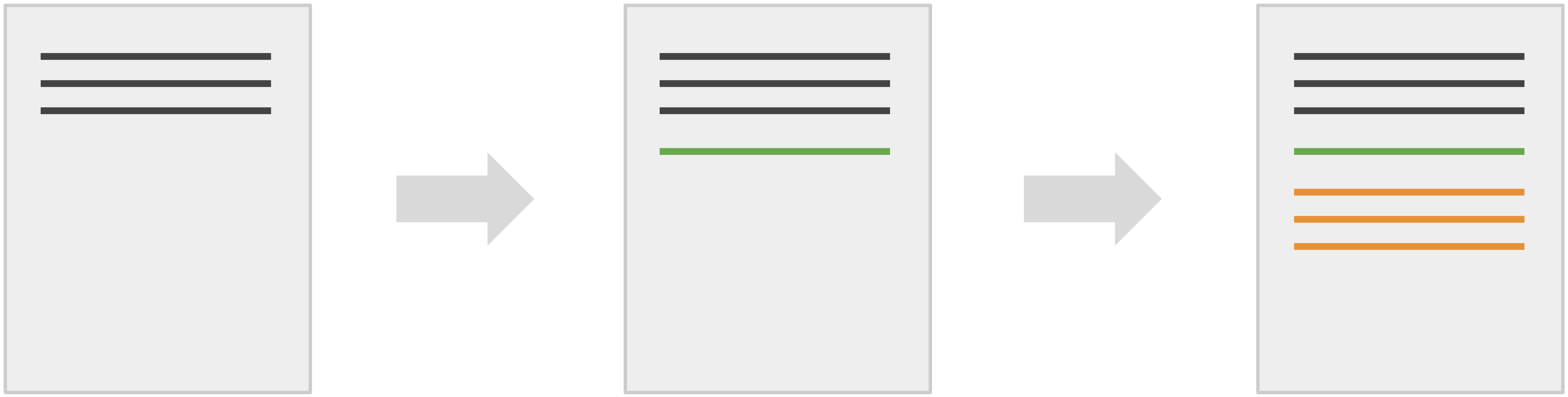
Version Control System

Keeps tracks of changes over time

Allows you track progress

Allows you to revert to earlier versions (dog can’t eat your homework)

Makes it easier to collaborate with others



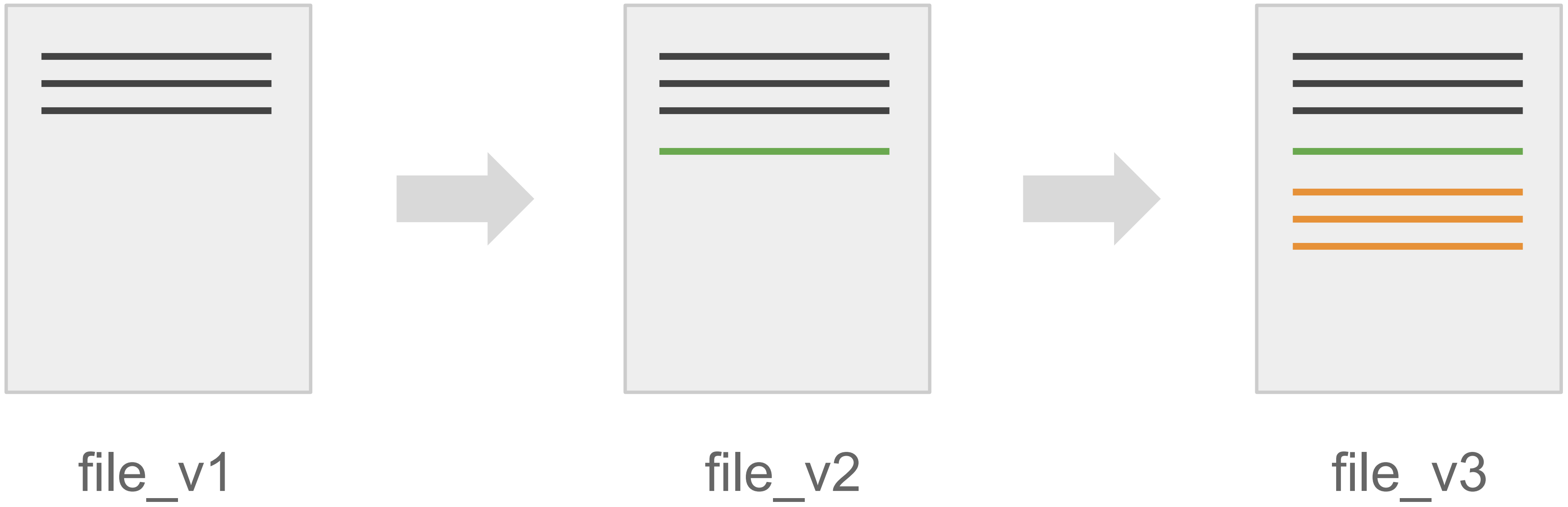
Consider some changes in a file

Saving a file 3 different times

-vs-

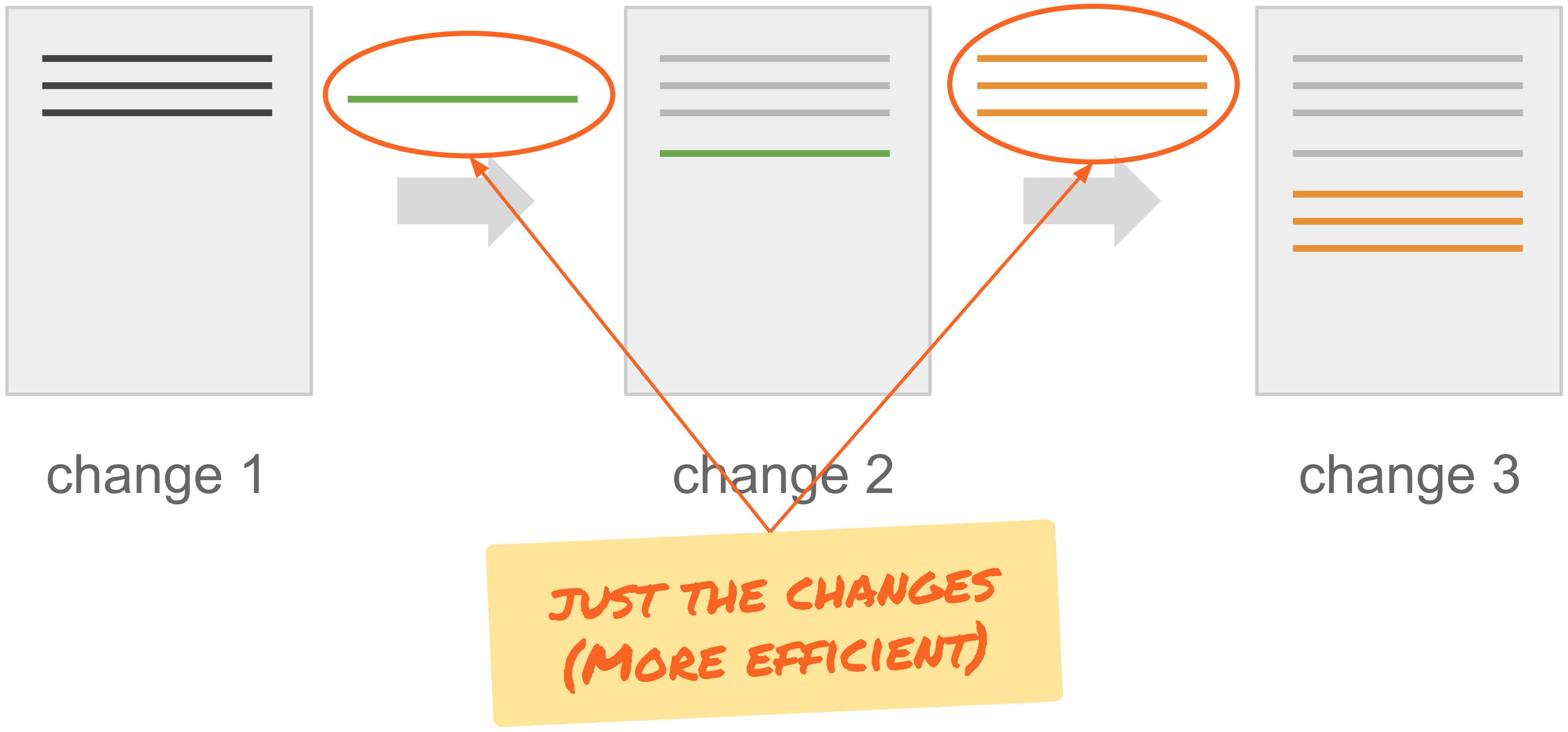
Saving snapshots of the changes

Saving a file 3 different times

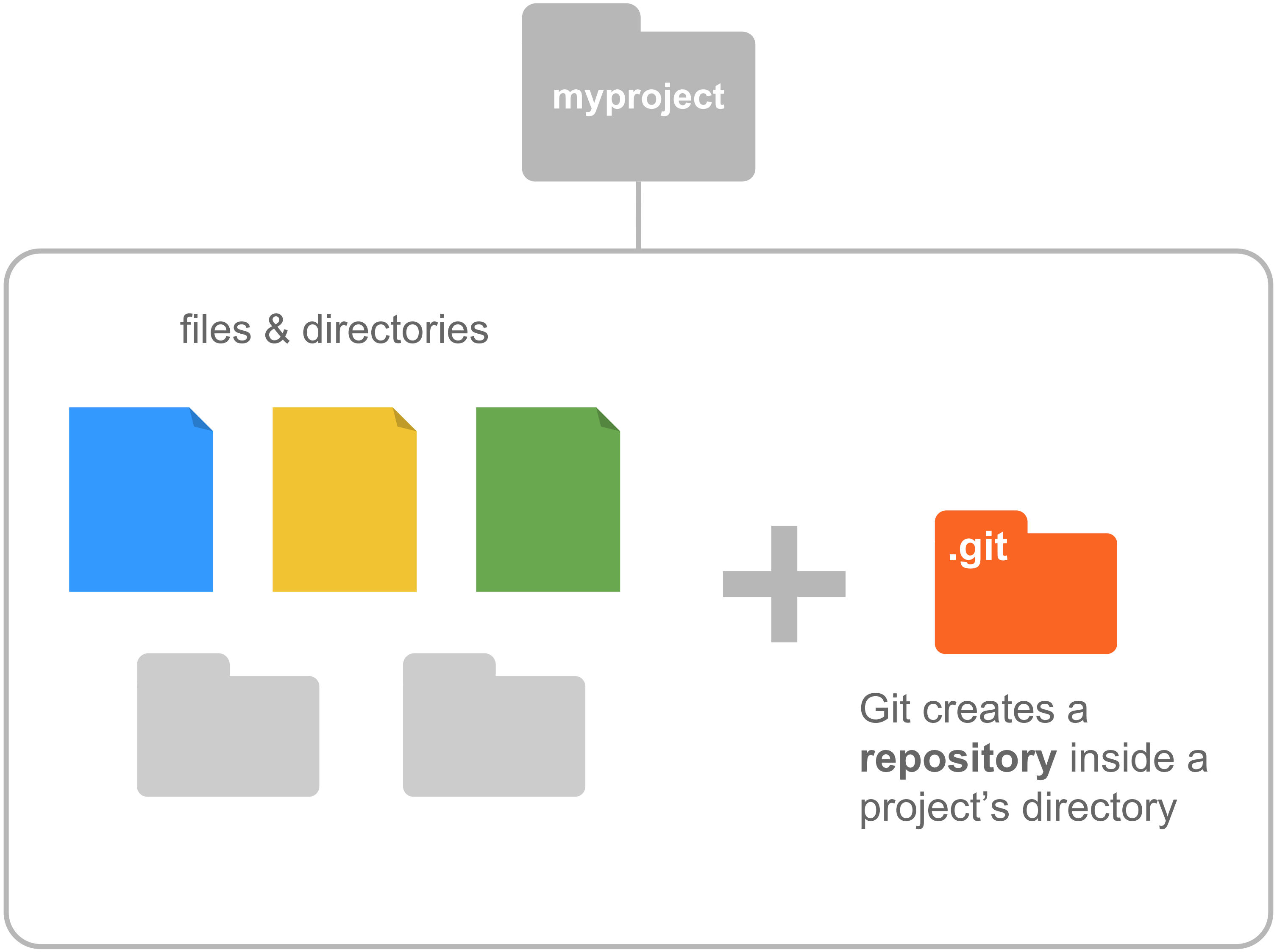


NOT VERY EFFICIENT

Saving snapshots of changes ...

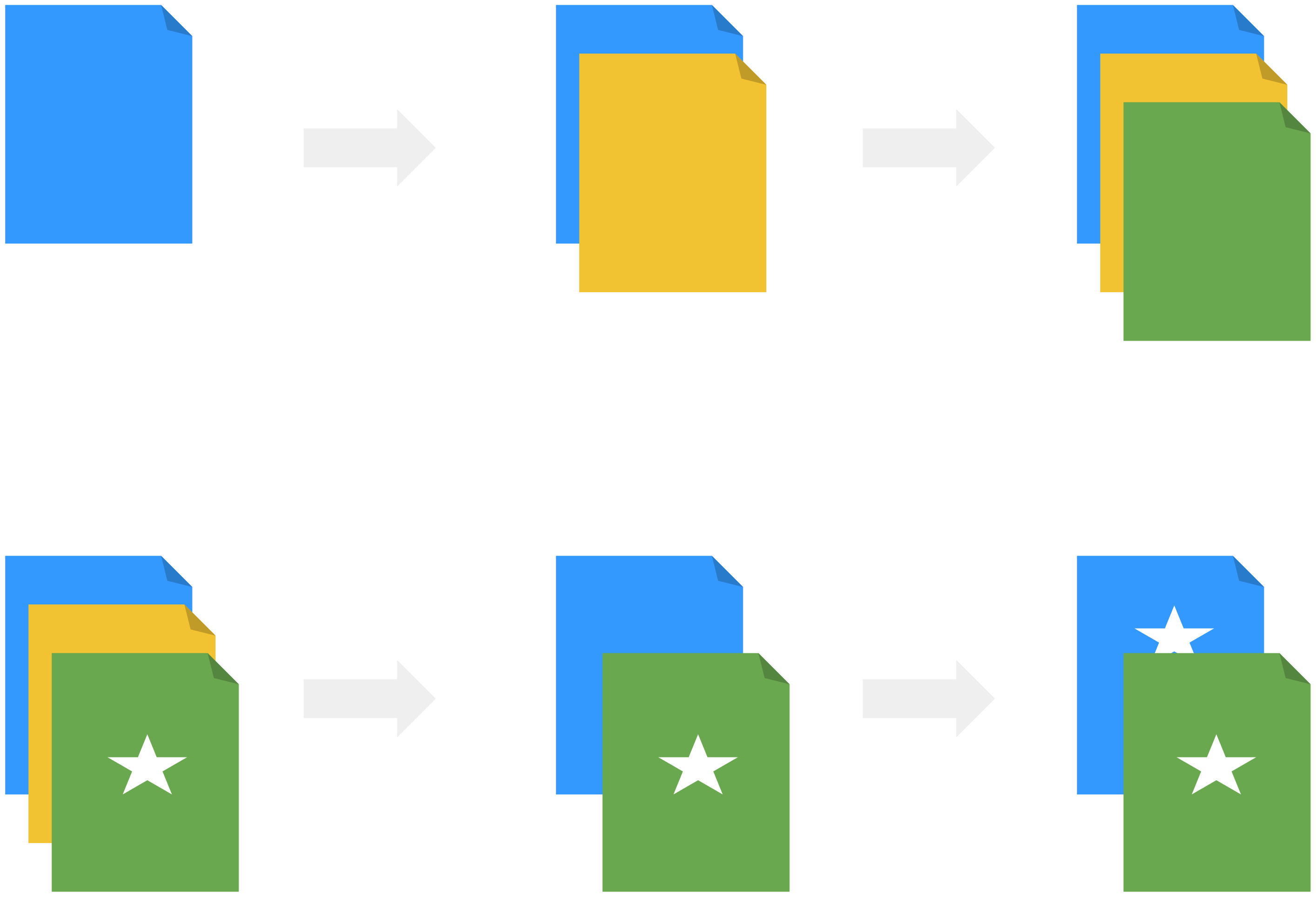


project's directory

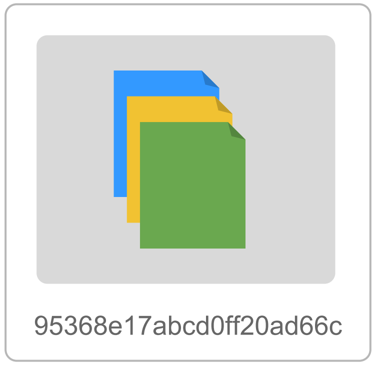


Git creates a **repository** inside a project's directory

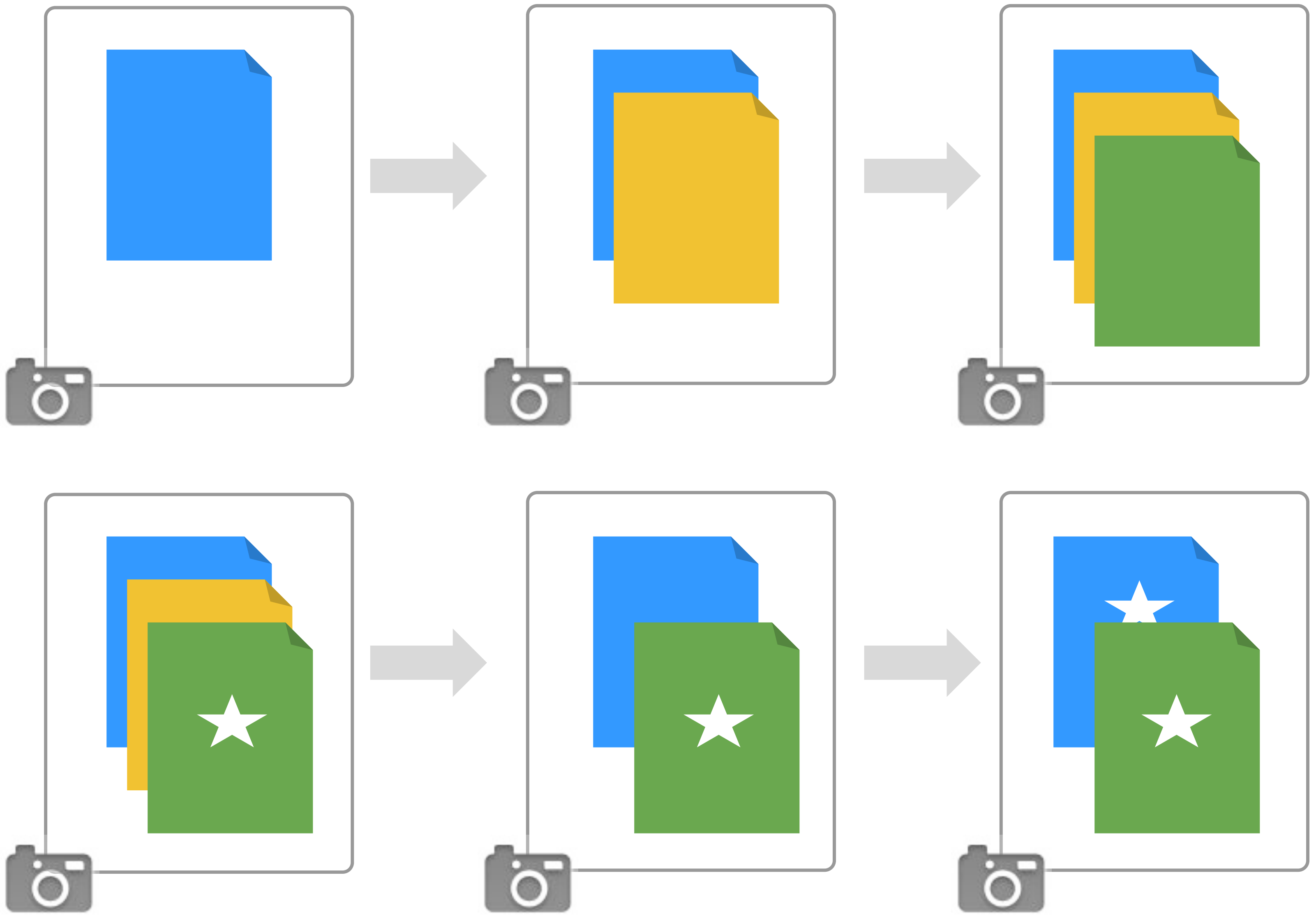
Project snapshots



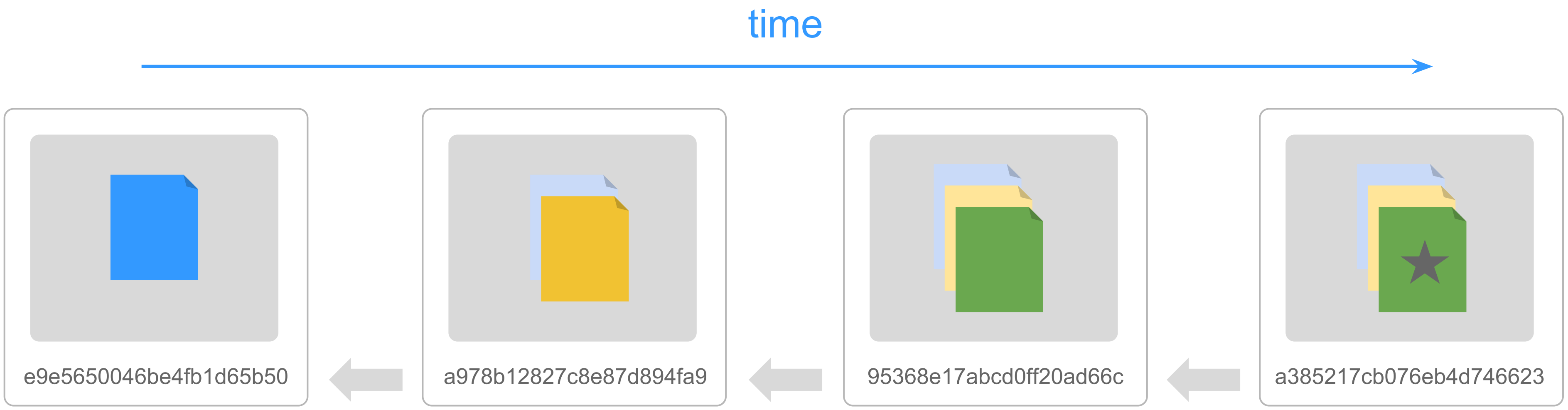
Git records the **changes** made on a project's files (not their versions) by taking "snapshots"



Project snapshots



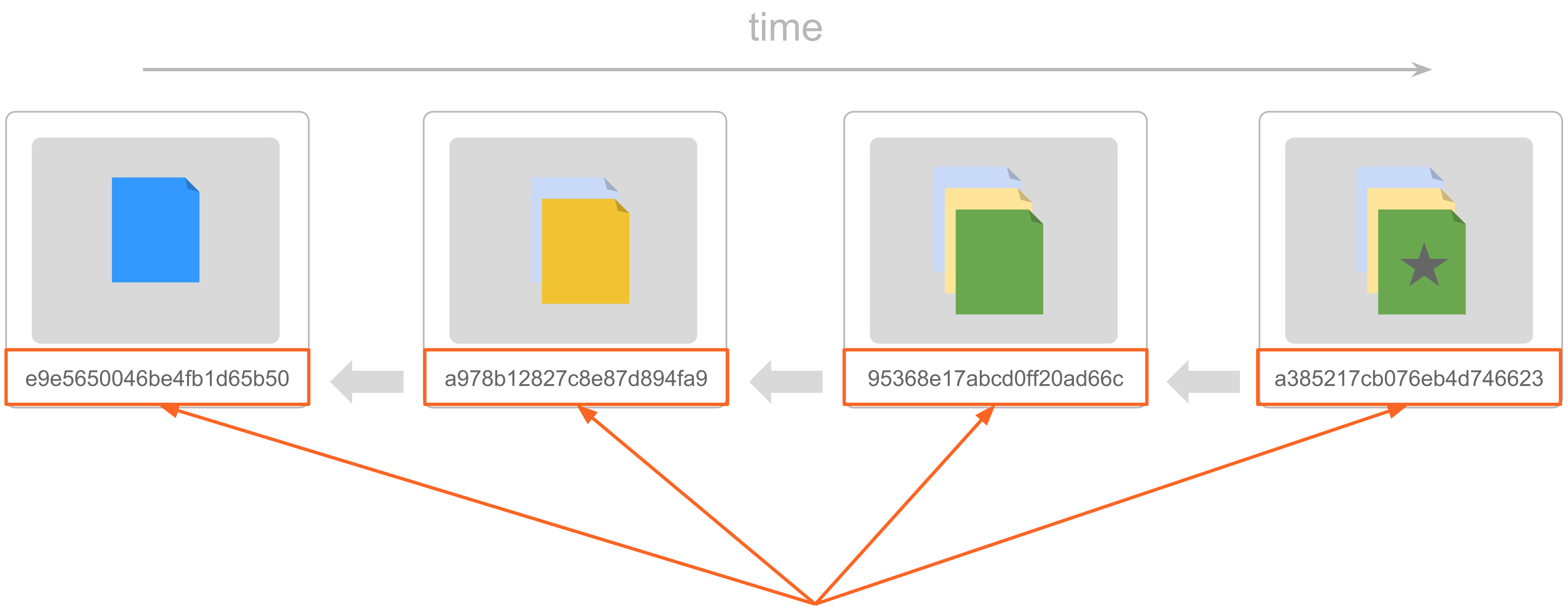
Git stores "snapshots"



A snapshot is a set of changes

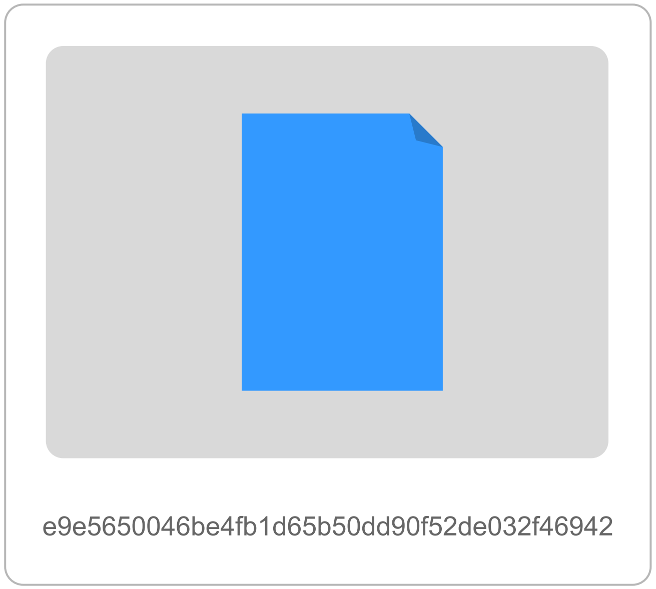
Each snapshot is known as a **commit**, i.e. a specific set of changes

Only new changes are tracked from one commit to the next one



Each commit (“snapshot”) has a unique ID or hash commit

SHA-1 values



`e9e5650046be4fb1d65b50dd90f52de032f46942`

SHA-1 value is 40-characters long

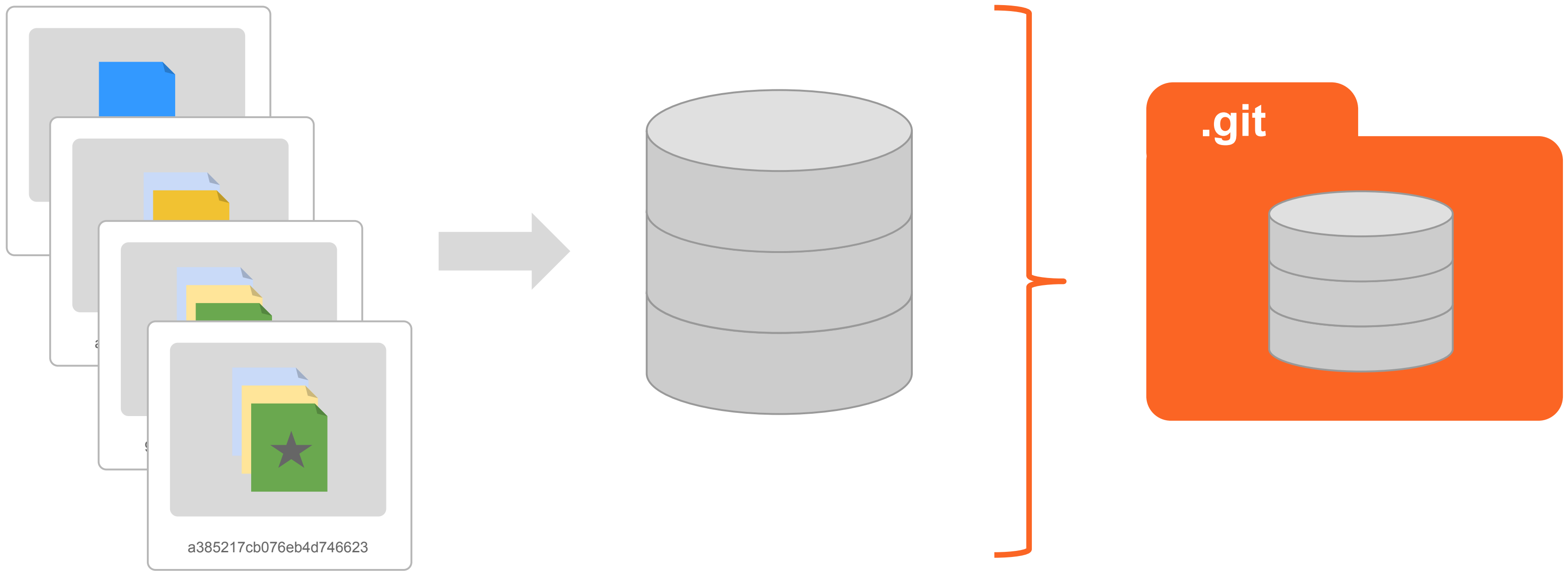
40 hexadecimal digits

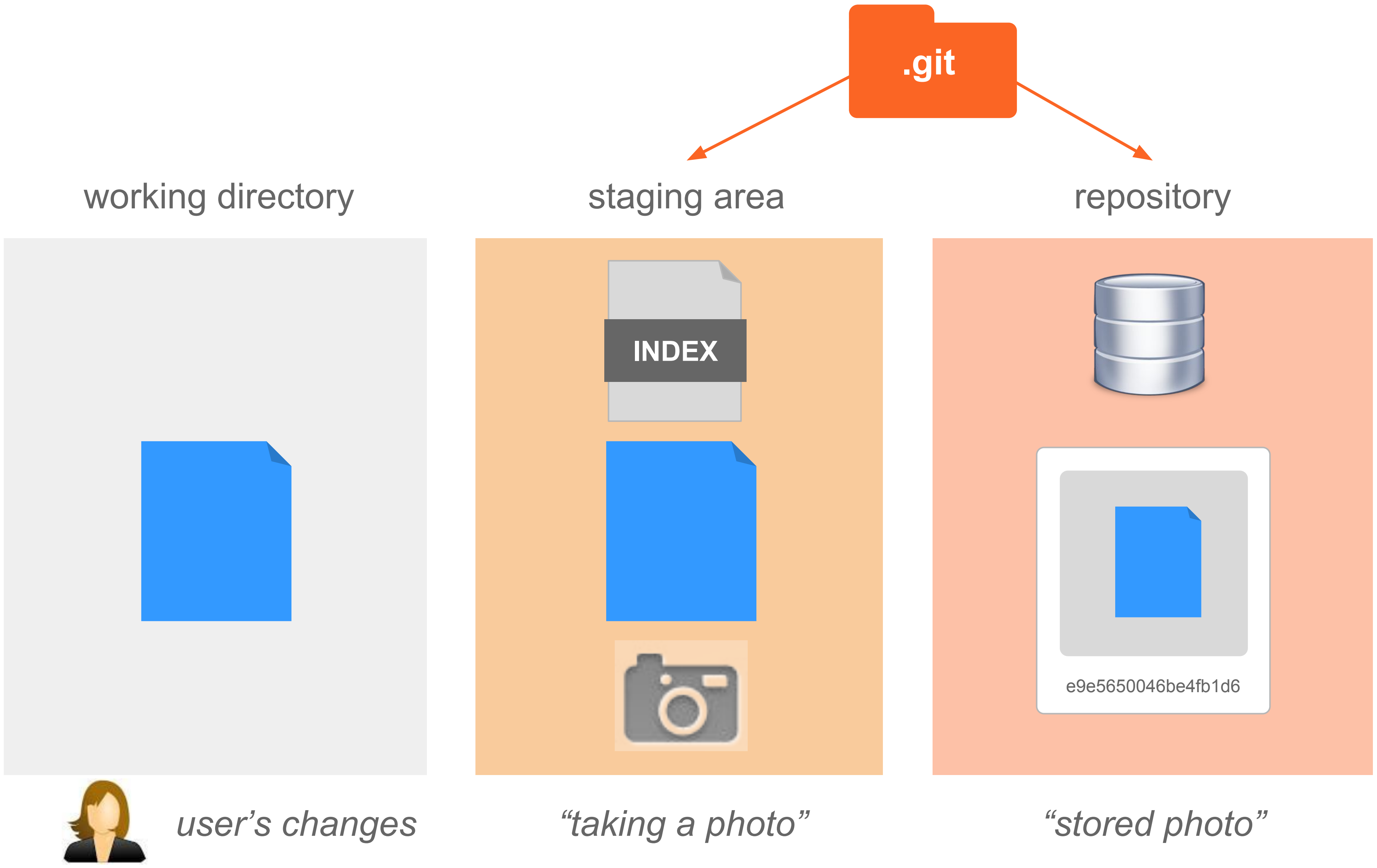
ID = hash commit

Determined by the SHA-1 algorithm https://en.wikipedia.org/wiki/SHA-1

How does Git 📷 “take and store snapshots”?

Git keeps information about all commits in its database (inside the .git directory)

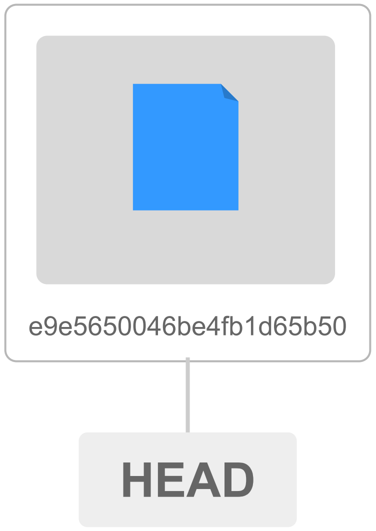




GIT USES A "THREE TREE" ARCHITECTURE

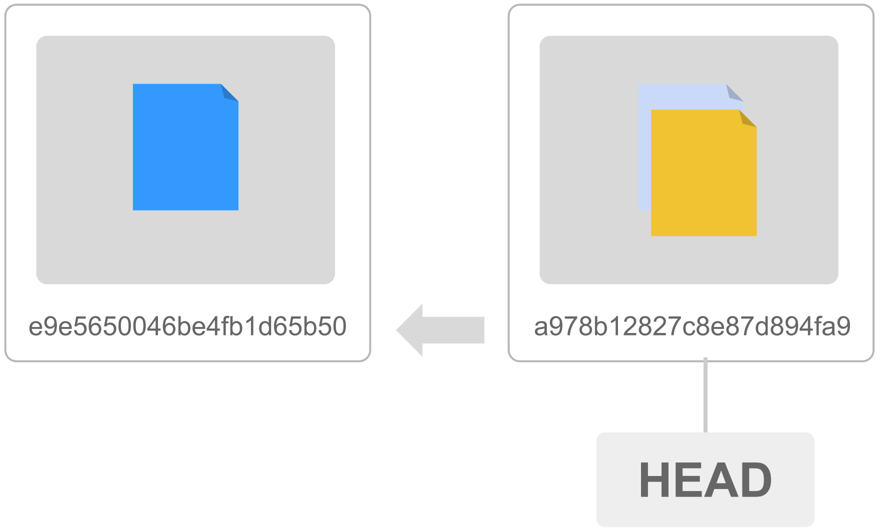
Basic Concept

HEAD



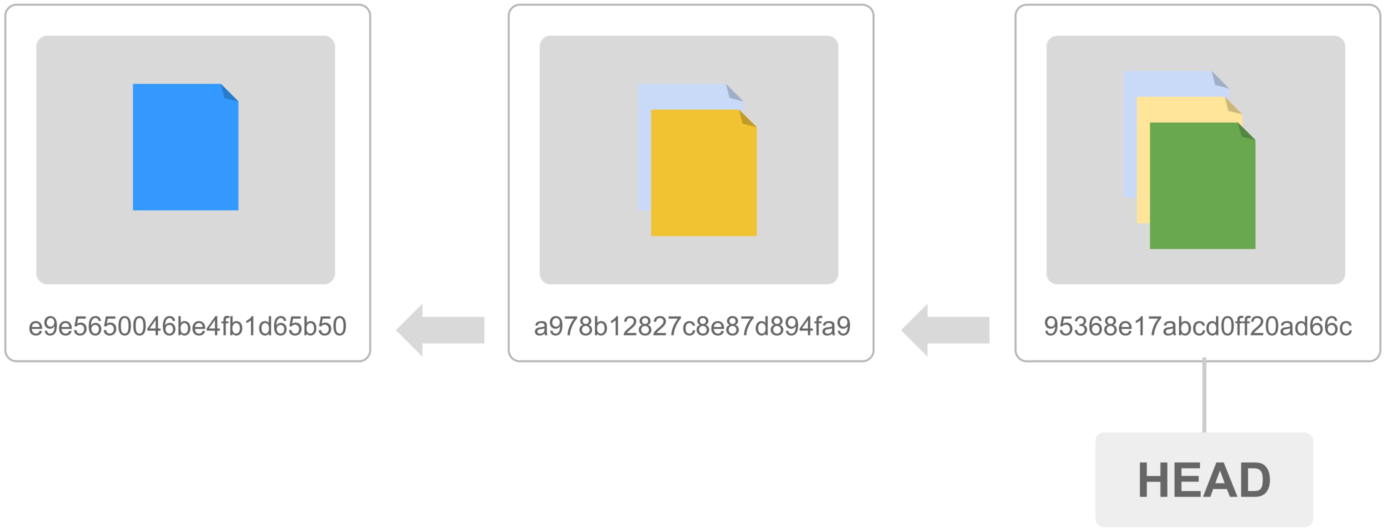
HEAD is a pointer

Typically, HEAD points to the last commit



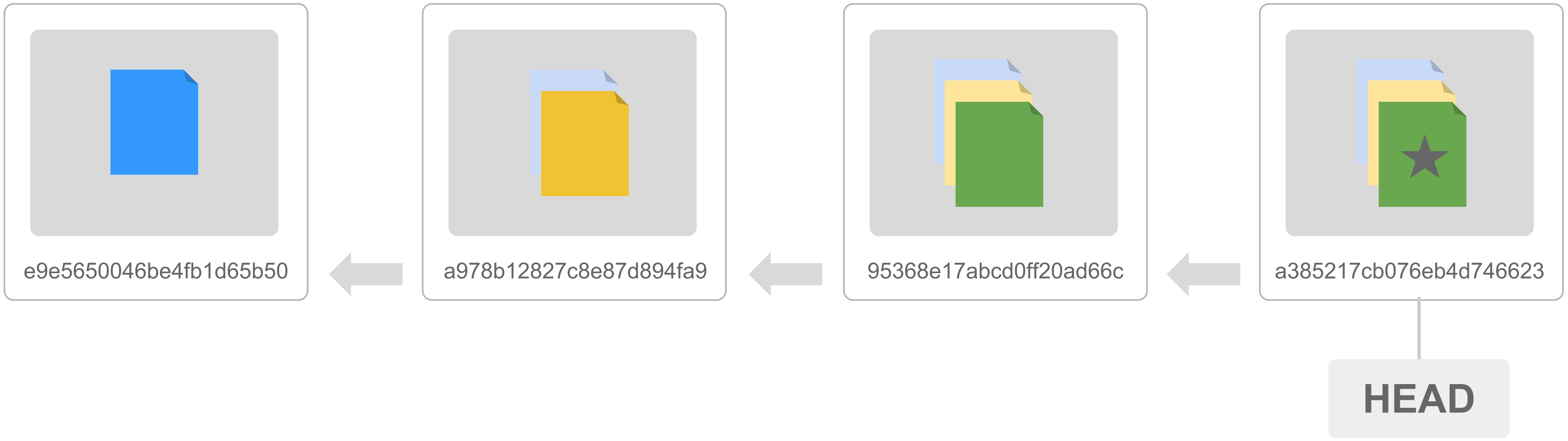
HEAD is a pointer

Typically, HEAD points to the last commit



HEAD is a pointer

Typically, HEAD points to the last commit



HEAD is a pointer

Typically, HEAD points to the last commit

Getting Started with Git

Installation & Configuration

Installation

Git is available for Mac, Windows, and Linux.

https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

I’m assuming you already have Git installed in your computer.

Configuration

After installing Git, the next step involves the so-called **Git Configuration**.

Think of the configuration steps as “introducing yourself to Git”.

The main command is **`git config`**

Global configuration

Tell Git who you are, e.g.:

```
git config --global user.name "Gaston Sanchez"

git config --global user.email "gasigiri@berkeley.edu"

git config --global color.ui "auto"
```

We recommend that you use your berkeley.edu email (which you should also use for your GitHub account)