Git Basic Concepts

Stat 133 by Gaston Sanchez

Git is a Version Control System (VCS)

VCS Key Idea

Keeping track of changes

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

01/10/15

file1

01/12/15

01/14/15

e1, file2, file3

How do you keep track of all these changes?

01/15/15

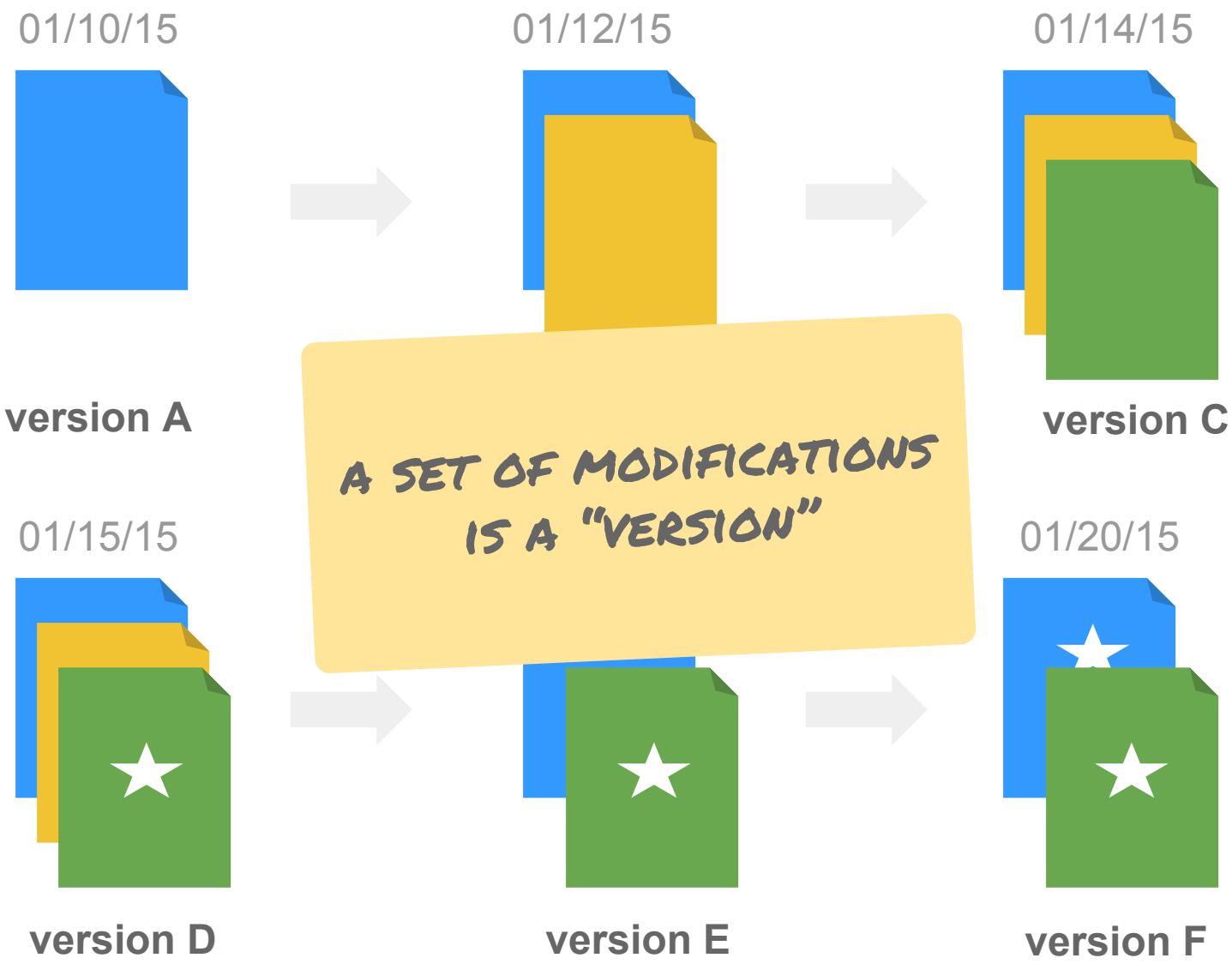
file1, file2, **file3**

file1, file3

01/20/15

file1, file3

myproject

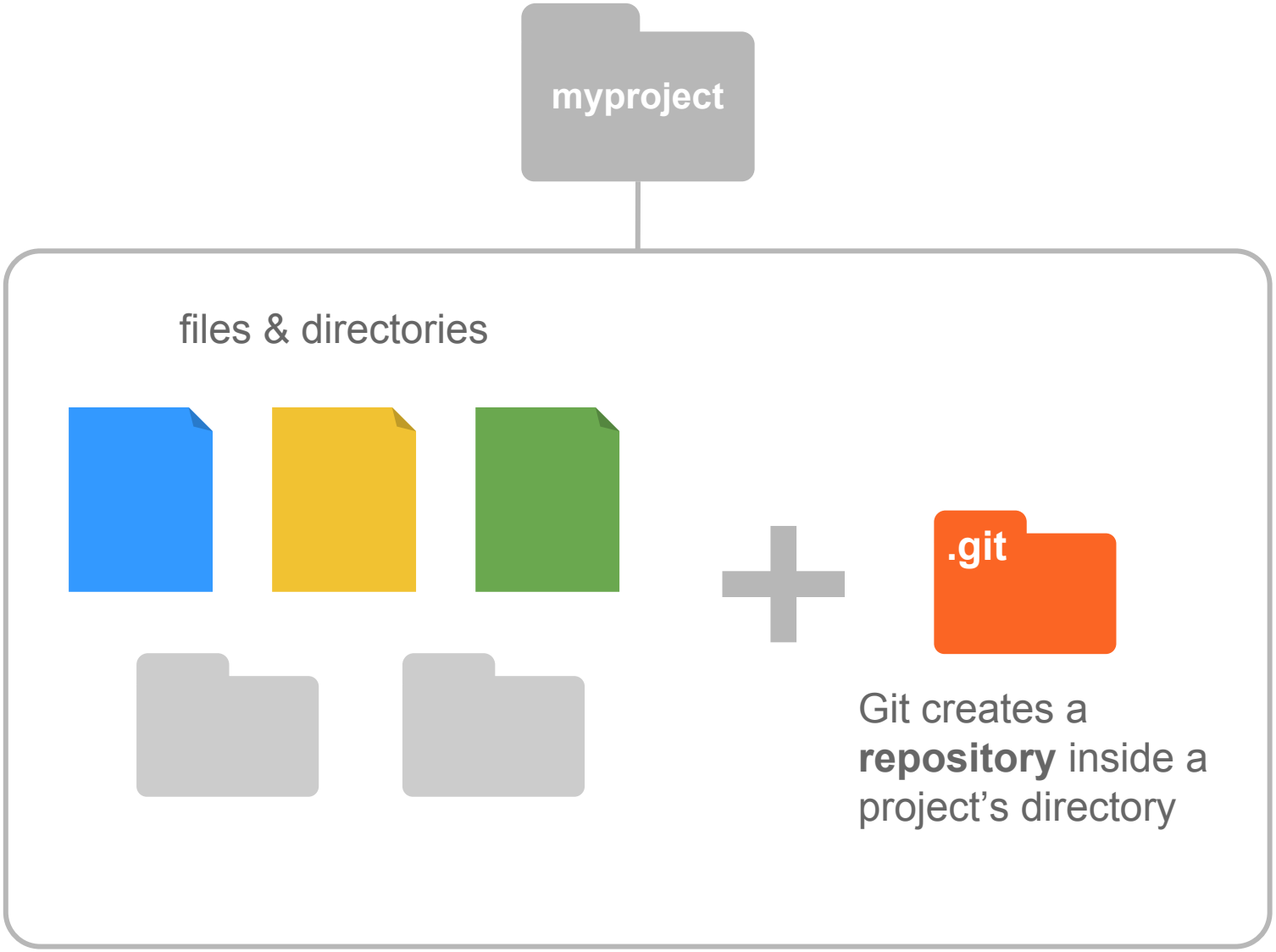

01/10/15
version A
01/12/15
version E
01/14/15
version C
01/15/15
version D
01/20/15
version F
A SET OF MODIFICATIONS
IS A "VERSION"

Key Ideas

Keep a record of all the made changes

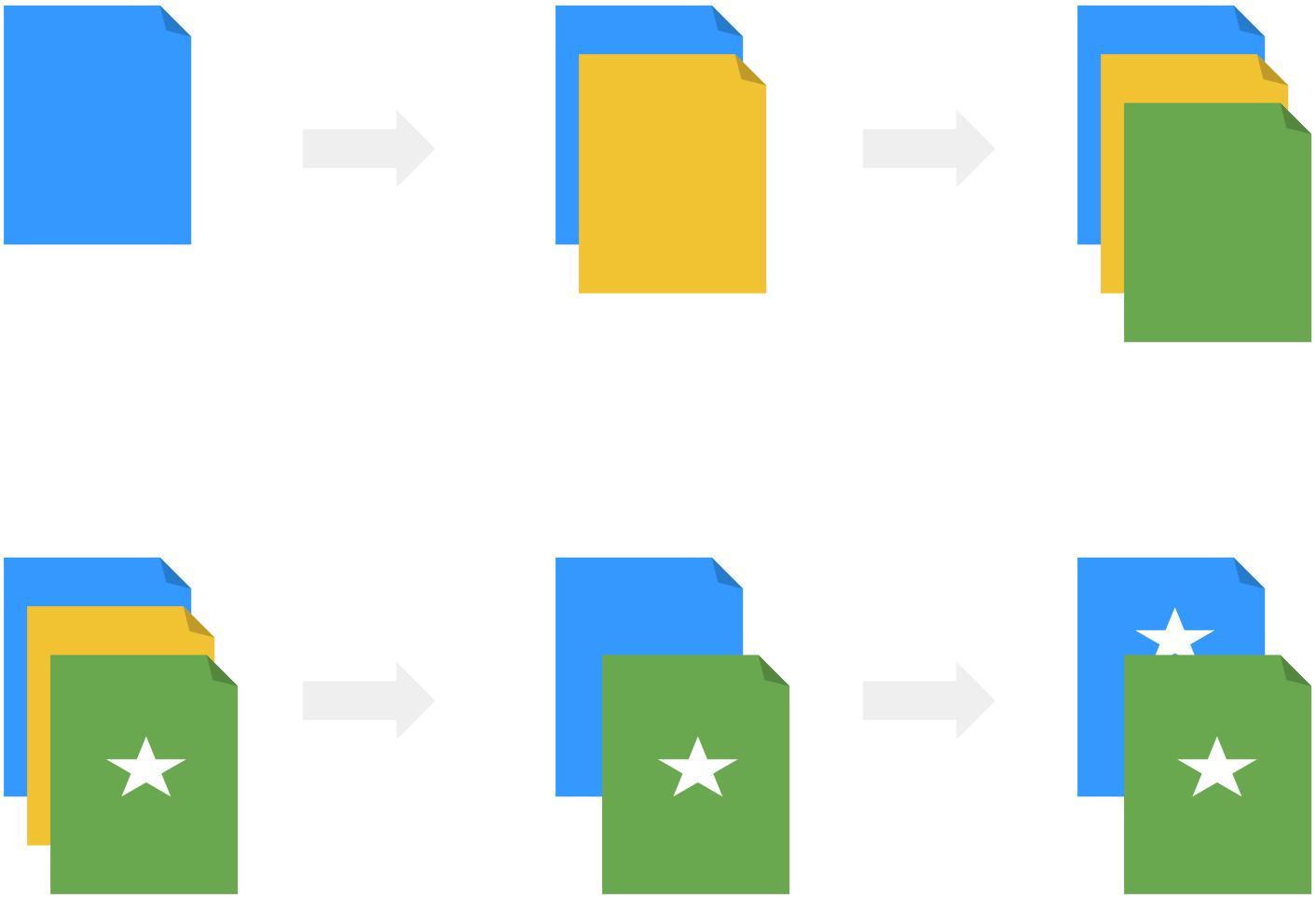
Storing changes of each version

project's directory

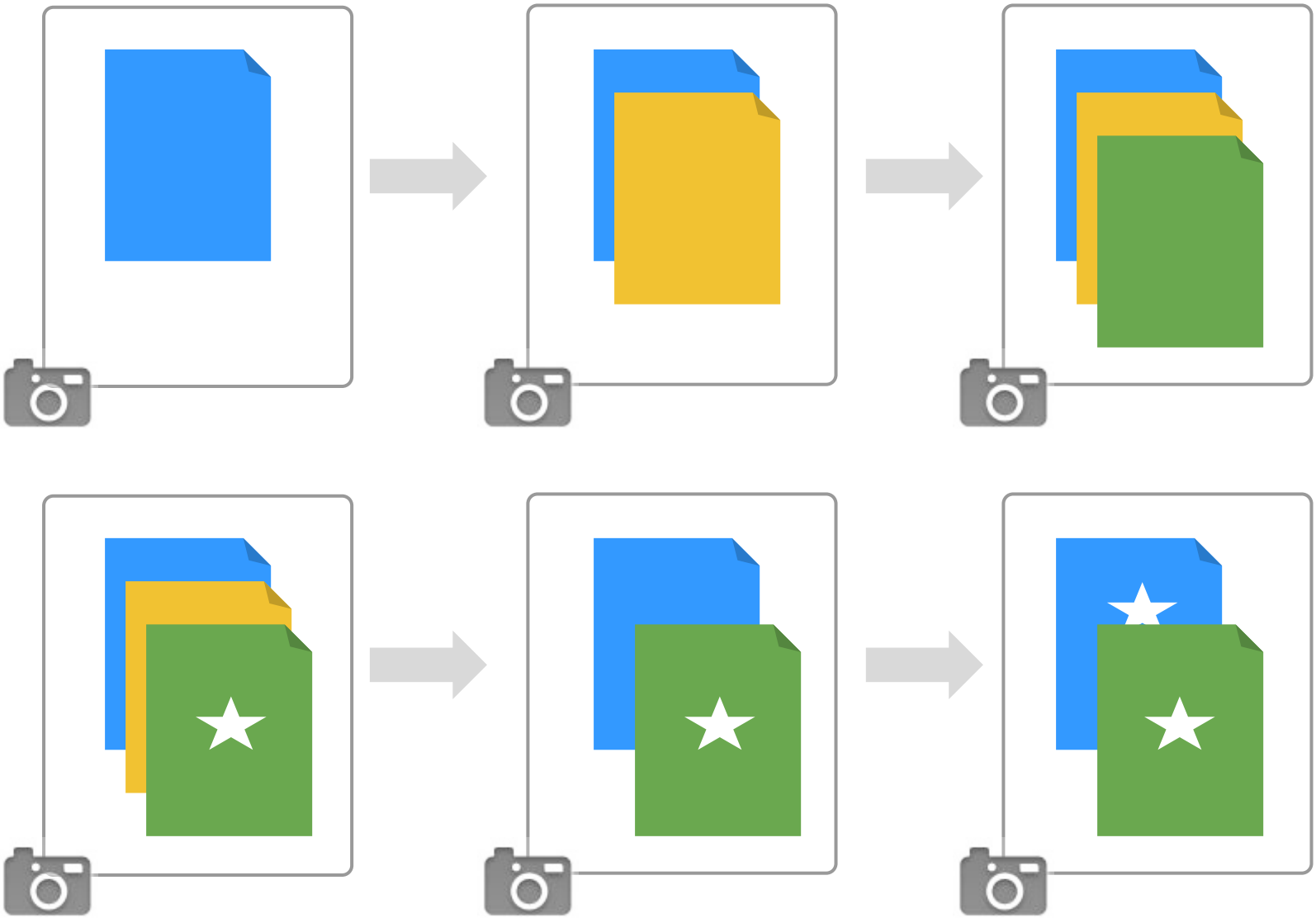


Git records the changes made on a project's files (not their versions)

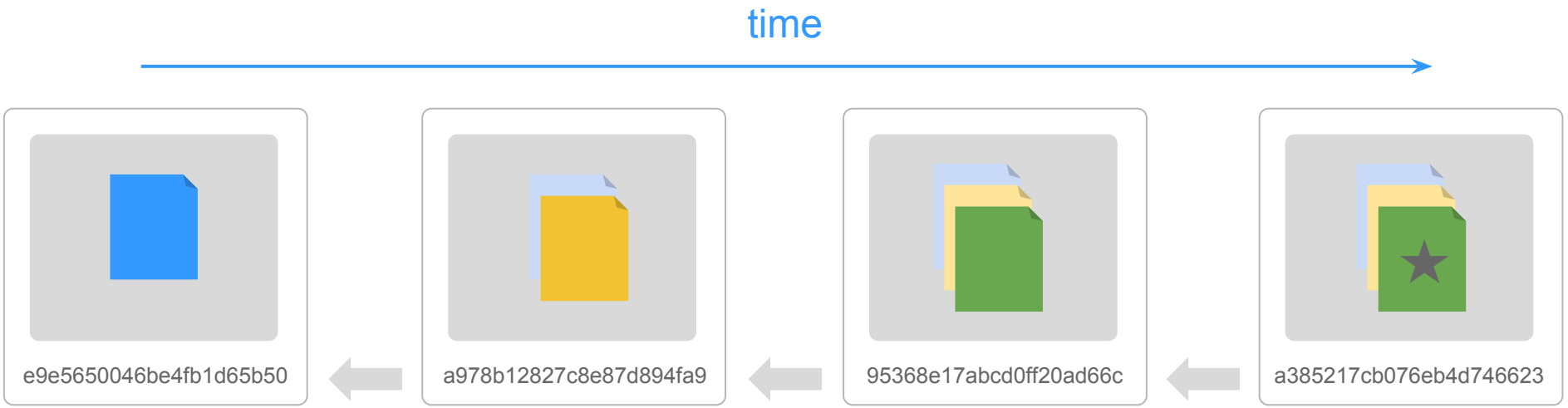
Project snapshots



Project snapshots

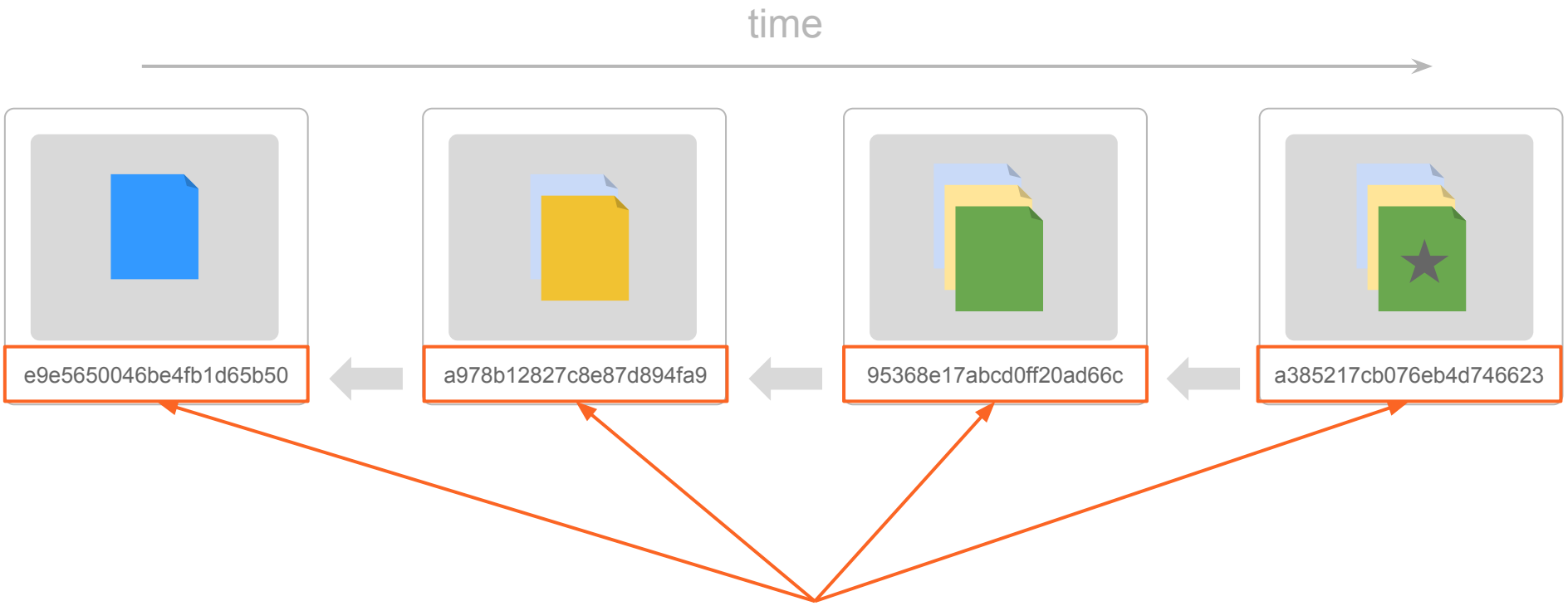


Git stores “snapshots”



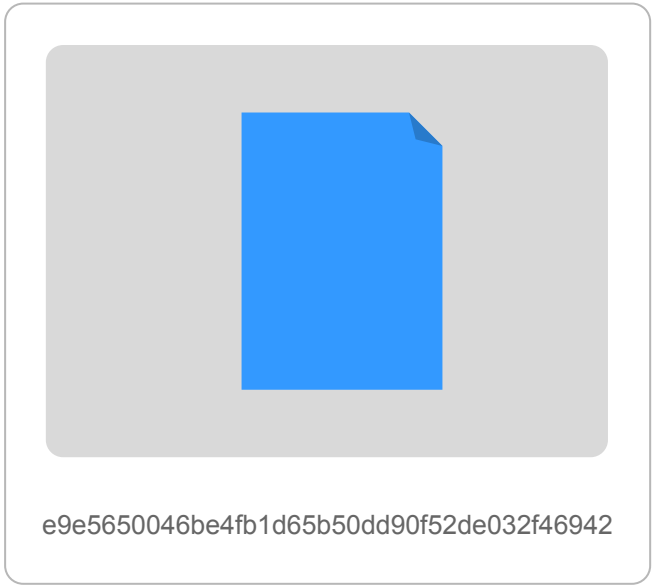
A snapshot is a set of changes
Each snapshot is known as a **commit**
Only new changes are tracked from one commit to the next one

Commit: a specific set of changes



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

SHA-1 values



`e9e5650046be4fb1d65b50dd90f52de032f46942`

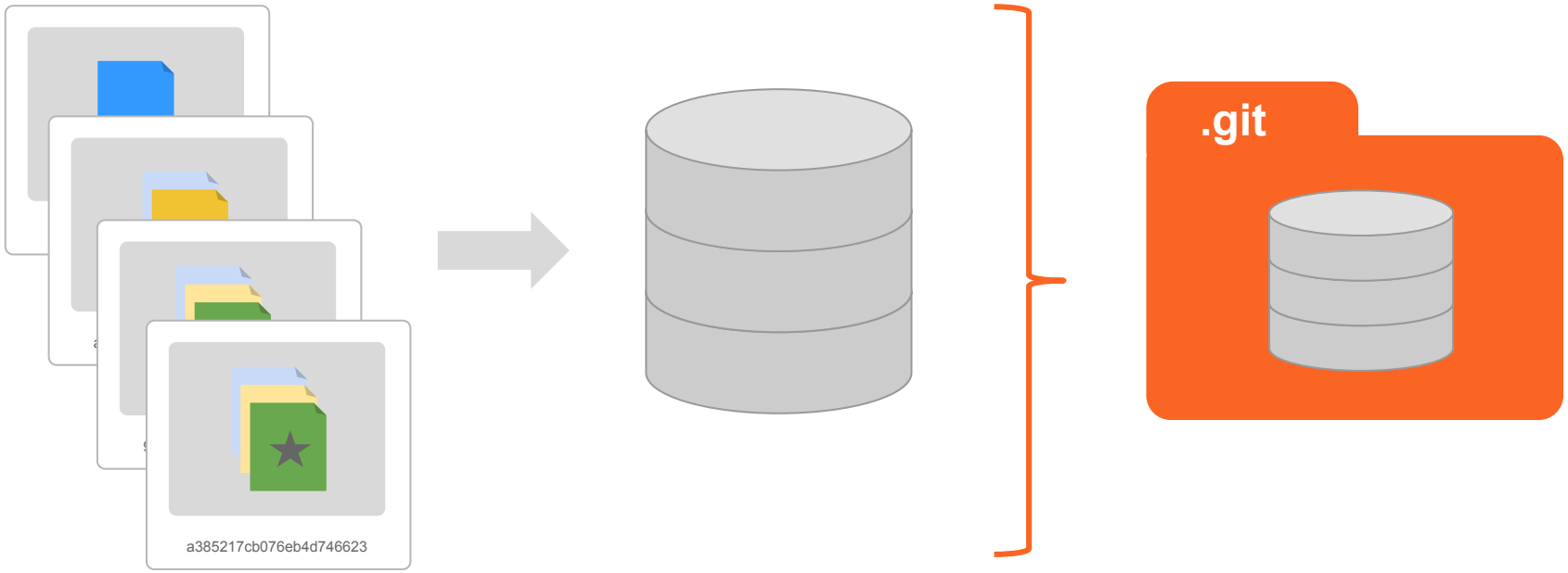
SHA-1 value is 40-characters long

40 hexadecimal digits

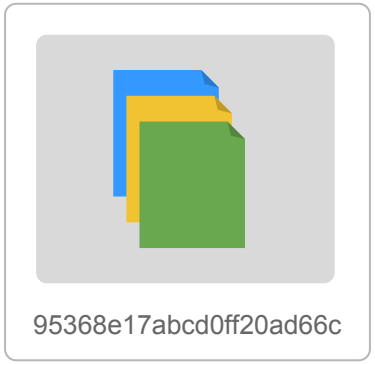
ID = hash commit

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

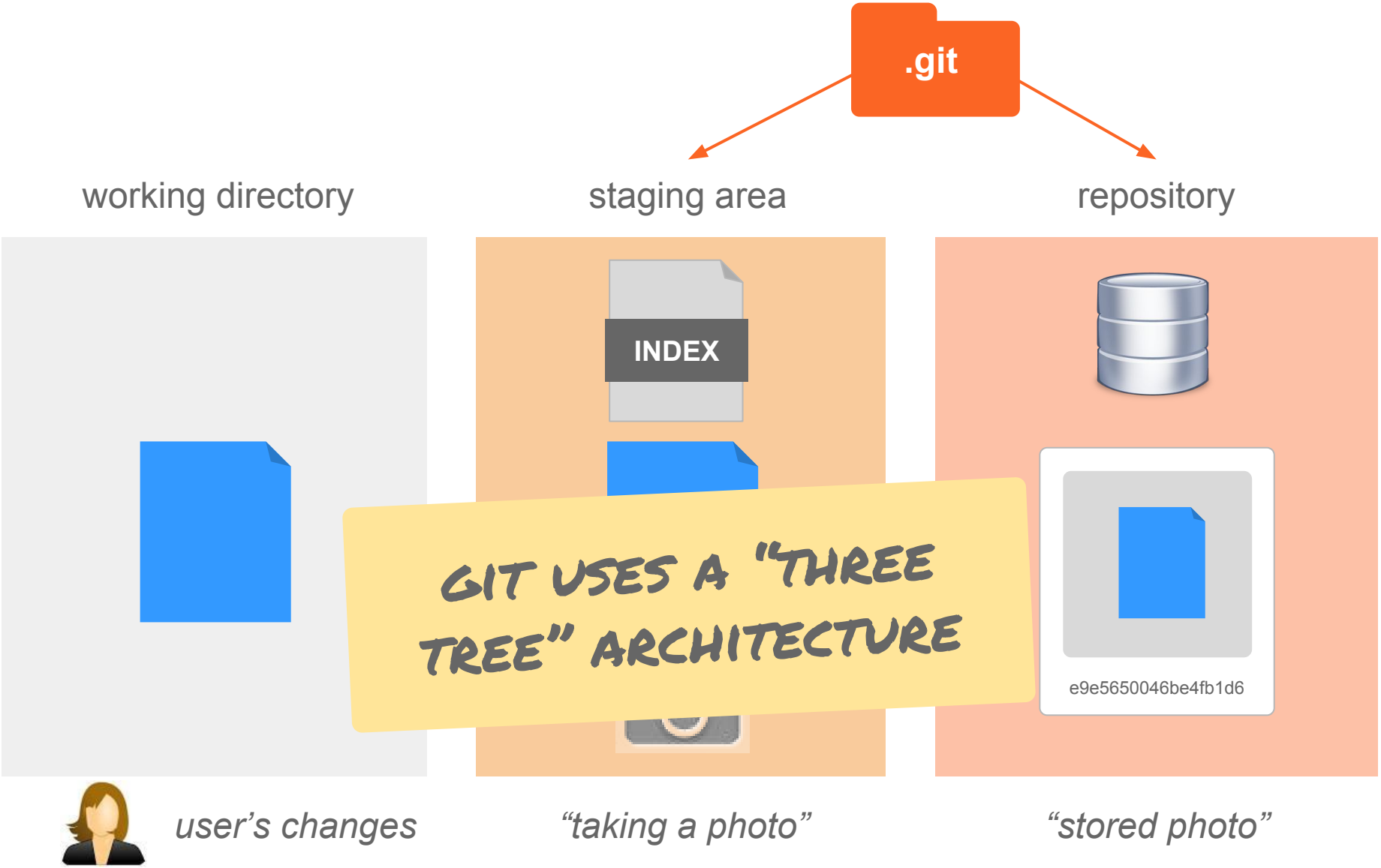
Git keeps information about all commits in its database (inside the .git directory)



Snapshots

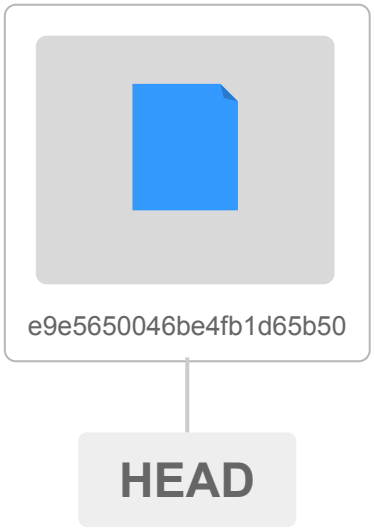


How does Git  “take a snapshot”?


.git
working directory
staging area
repository
INDEX
e9e5650046be4fb1d6
GIT USES A "THREE TREE" ARCHITECTURE
user's changes
"taking a photo"
"stored photo"

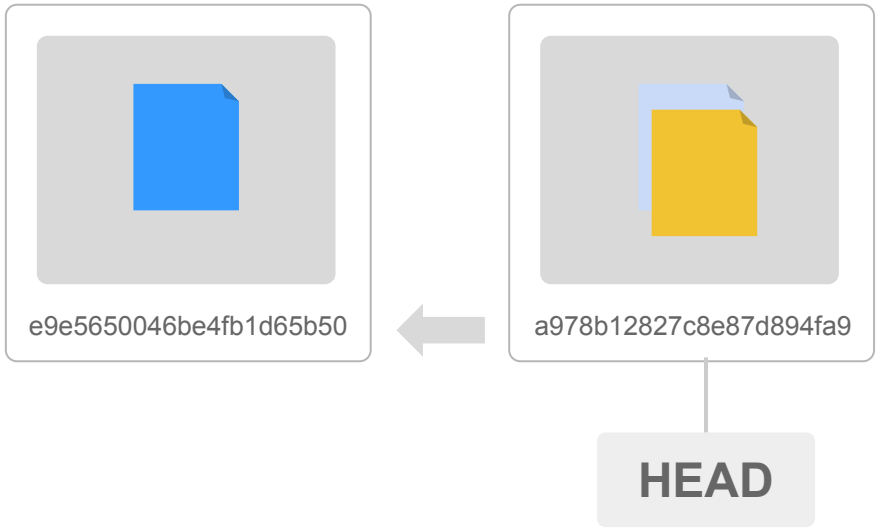
Basic Concept

HEAD



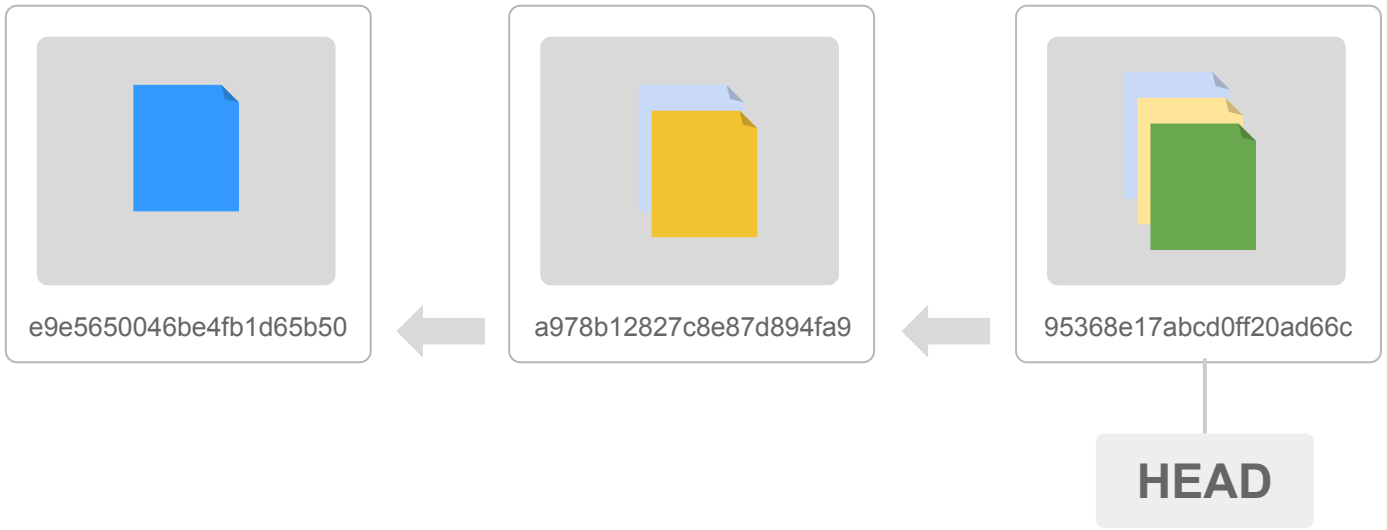
HEAD is a pointer

Typically, HEAD points to the last commit



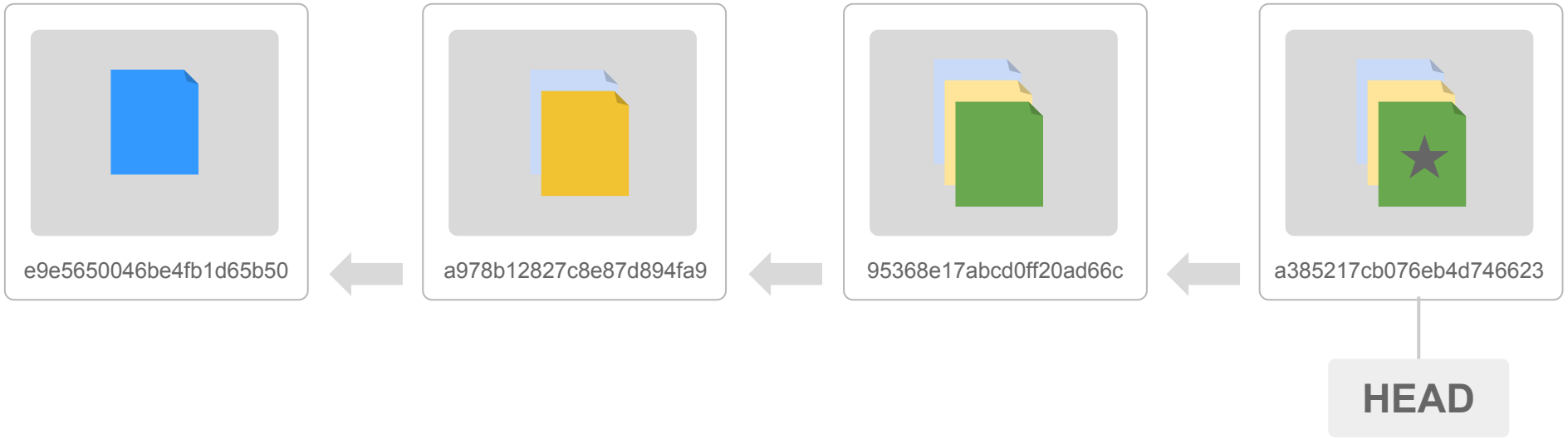
HEAD is a pointer

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

Commands

Basic Commands

git command

Basic Commands

`git help`	help documentation
`git config`	configuration
`git init`	initialize a repository
`git status`	status information
`git add`	add unstaged changes
`git commit`	commit staged changes
`git log`	see log (changes)
`git diff`	differences between files

Configuration

Configuration

3 types of configuration

System level

apply to every user of the computer

User level

apply to a single user

Project level

project to project configurations

System Level Configuration *(may not exist)*

Unix:

```
/etc/gitconfig
```

Windows:

```
Program Files\Git\etc\gitconfig
```

User Level Configuration

Unix:

```
~/.gitconfig
```

Windows:

```
$HOME\.gitconfig
```

Project Level Configuration

Unix:

```
my_project/.git/config
```

Windows:

```
my_project\.git\config
```

git config

command to configure git (depending on the level)

`git config --system`	system level
`git config --global`	user level
`git config`	project level

User Level Configuration

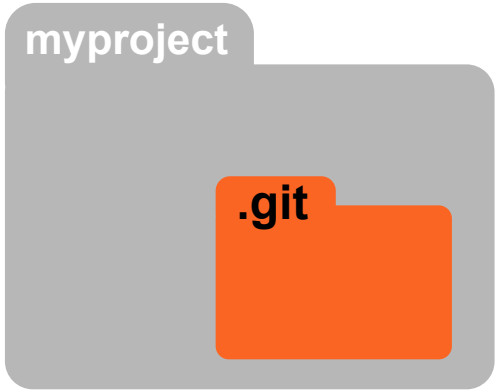
```
git config --global user.name "Jon Doe"

git config --global user.email "jondoe@email.com"

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

Repository Initialization

Repository



Repository:
Database (hooked to a project) where the VCS stores all the versions and metadata of the project.

a project's repository is the `.git` directory

git init

Initializes git on a project

Tell git to start tracking changes

Get everything ready to start doing its tracking

git init

```
Initialized empty Git repository in
/Users/gaston/Documents/git_project/.git/
```

INITIALIZATION MESSAGE

git init

```
Initialized empty Git repository in
/Users/gaston/Documents/git_project/.git/
```

this is where Git will be storing
information about its tracking

the **.git** directory

- `.git`
  - `.`
  - `..`
  - `HEAD` HEAD references
  - `branches`
  - `config` Configuration settings
  - `description`
  - `hooks`
  - `info`
  - `objects`
  - `refs` HEAD references

More git in the lab

Lab 03

You will have the chance to get a better feeling of how to use git in this week's lab.