UPskill

Digital Skills & Jobs

Version Control

Metodologias de Trabalho em Equipa

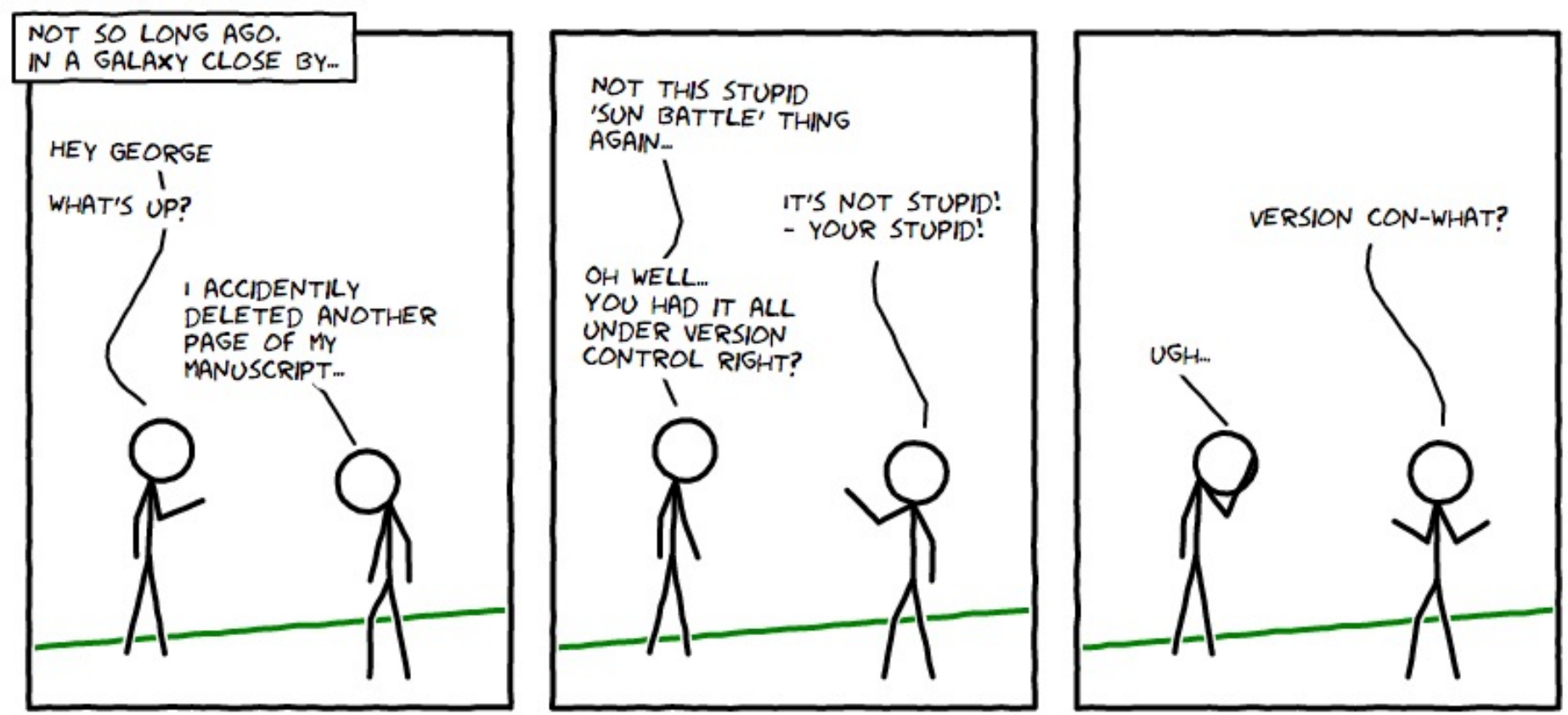
P.PORTO

apdc ASSOCIAÇÃO PORTUGUESA PARA O DESENVOLVIMENTO DAS COMUNICAÇÕES

INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL

CONSELHO COORDENADOR DOS INSTITUTOS SUPERIORES POLITÉCNICOS

Version Control



What is version control?

- System for recording and managing changes made to files and folders, so that you can recall specific versions later
- Used to manage source code
  - However, it is also well suited to tracking changes to any kind of file which contains mostly text.
- Used by a lone developer or as a means for many people to share and collaborate on projects efficiently and safely

What is version control?

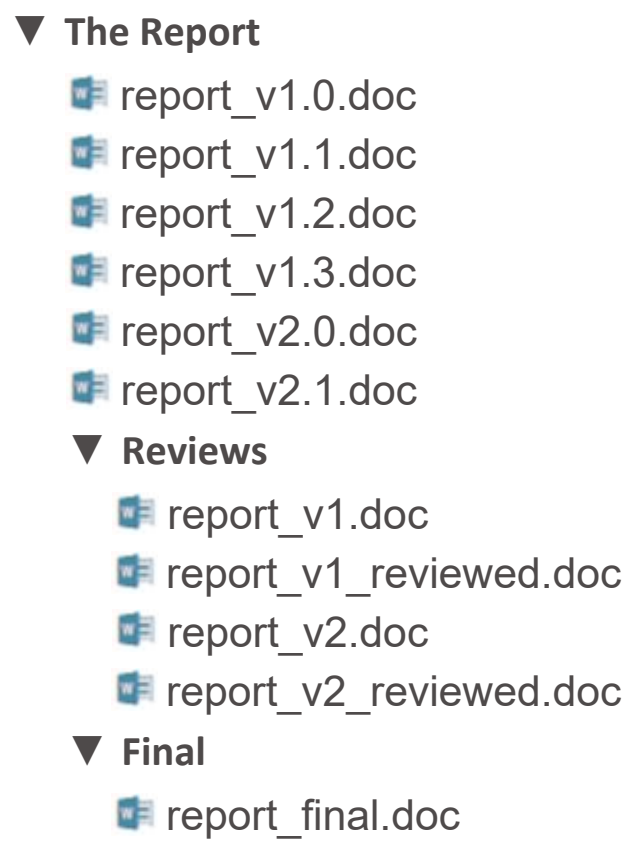
- Allows:
  - revert selected files back to a previous state;
  - revert the entire project back to a previous state;
  - compare changes over time;
  - see who last modified something that might be causing a problem;
  - who introduced an issue and when;
  - and more.
- Using a VCS also generally means that if you screw things up or lose files, you can easily recover.

What is version control?

- You probably already use a version control system, even without realizing it…
- This feature is built-in many modern editors such as:
  - Microsoft Word
  - Apple Pages
- Dropbox maintains a full history of all the files you have edited or deleted on the last month

What is version control?

- You have almost certainly employed your own simple form of a version control system in the past



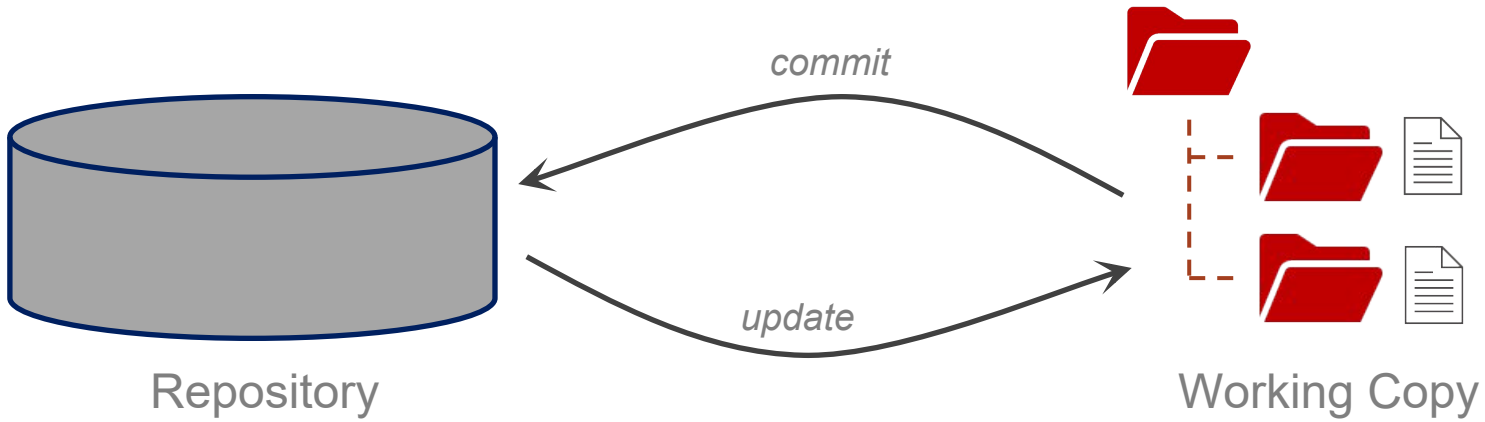
Why should you use it?

- **Reproducibility**
  - You should be able to replicate every figure you have ever published, even if you have significantly developed your codes and tools since;
- **Recoverability**
  - Bring back that snippet you accidentally deleted;
- **Experimentability**
  - Try different approaches and simply disregard them if you don't like it;
- **Collaborability**
  - Concurrently work on a project with a collaborator and then automatically merge all of your changes together.

> *In practice, everything that has been created manually should be put in version control, including programs, original field observations, and the source files for papers.*
>
> – Best Practices for Scientific Computing; Wilson et al. 2012 arXiv:1210.0530

Repositories and Working Copies

- Version control uses:
  - a Repository
    - Database of changes
  - a Working Copy (also called *checkout*)
    - Personal copy of the project files



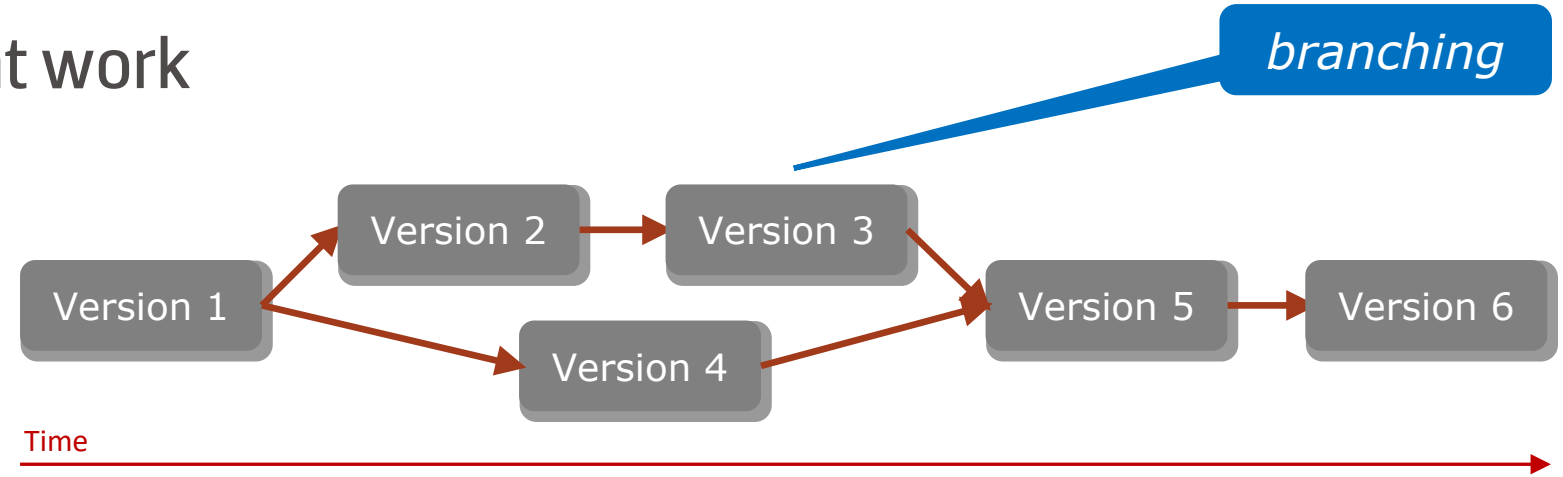
Repository

- Database of all the edits to, and/or historical versions (snapshots) of, your project.
- The repository can contain edits that have not yet been applied to your working copy.
- You can update your working copy to incorporate any new edits or versions that have been added to the repository since the last time you updated.

Repository – scenarios

- A linear history

- Concurrent work

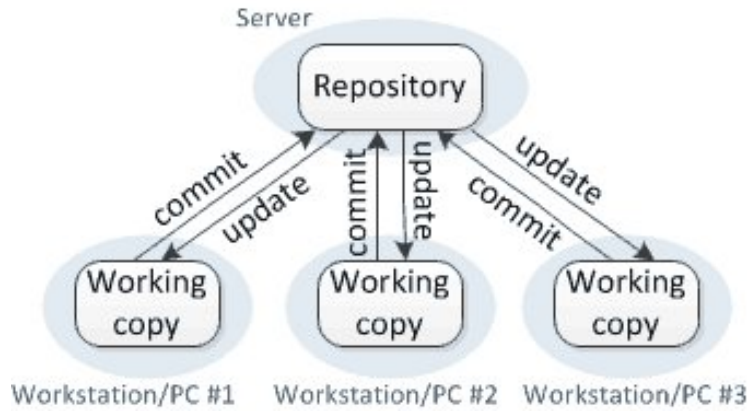


Version Control types

- Centralized
  - Slower
  - Easier to understand
  - Used by: *Subversion* (*SVN*)
- Distributed
  - Runs faster
  - Less prone to errors
  - Complex to understand
  - Used by: *Git*, *Mercurial*

Centralized version control

- Each user gets his own working copy, but there is just one central repository;
- As soon as you commit, it is possible for your co-workers to update and to see your changes.
- For others to see your changes, 2 things must happen:
  - You *commit*
  - They *update*



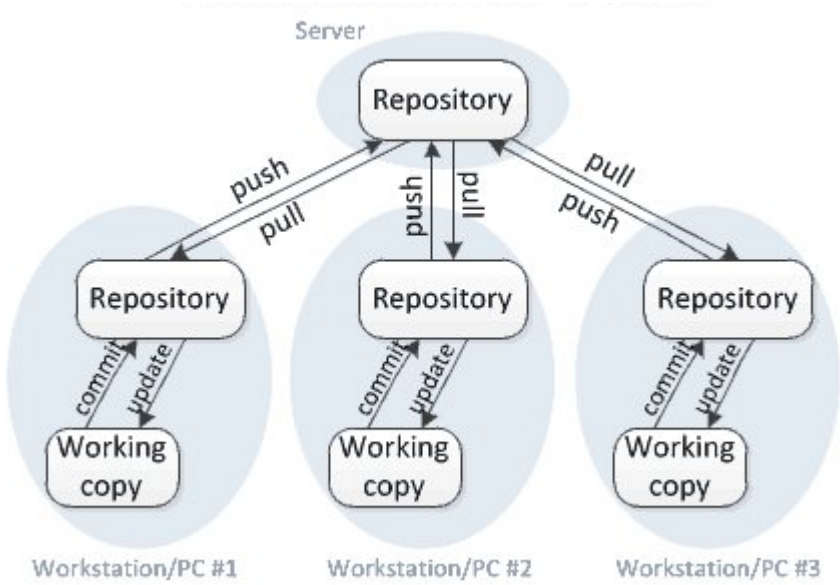
https://homes.cs.washington.edu/~mernst/advice/version-control.html

Distributed version control

- Each user gets his or her own repository and working copy.
- After you commit, others have no access to your changes until you *push* your changes to the central repository.
- When you update, you do not get others' changes unless you have first *pulled* those changes into your repository.

Distributed version control

- For others to see your changes, 4 things must happen:
  - You *commit*
  - You *push*
  - They *pull*
  - They *update*



https://homes.cs.washington.edu/~mernst/advice/version-control.html

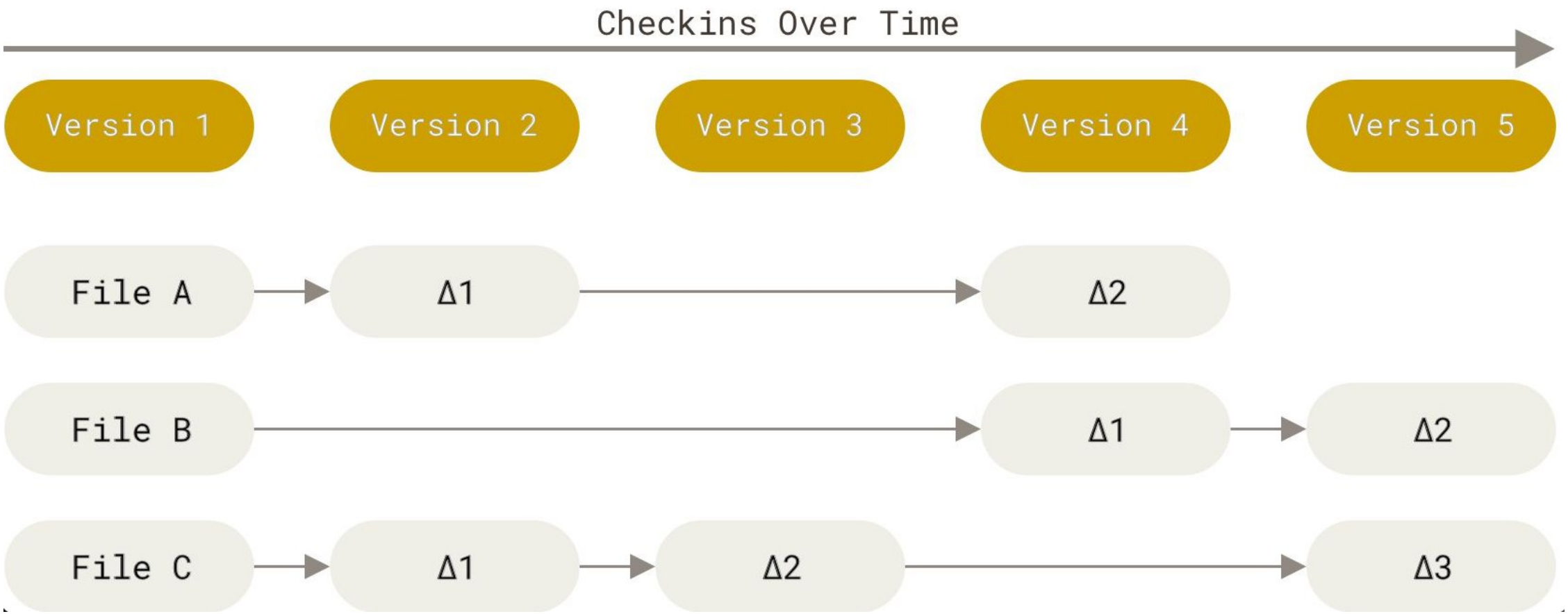
Introducing … Git!

> *Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.*
>
> *Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.*
>
> *https://git-scm.com/*

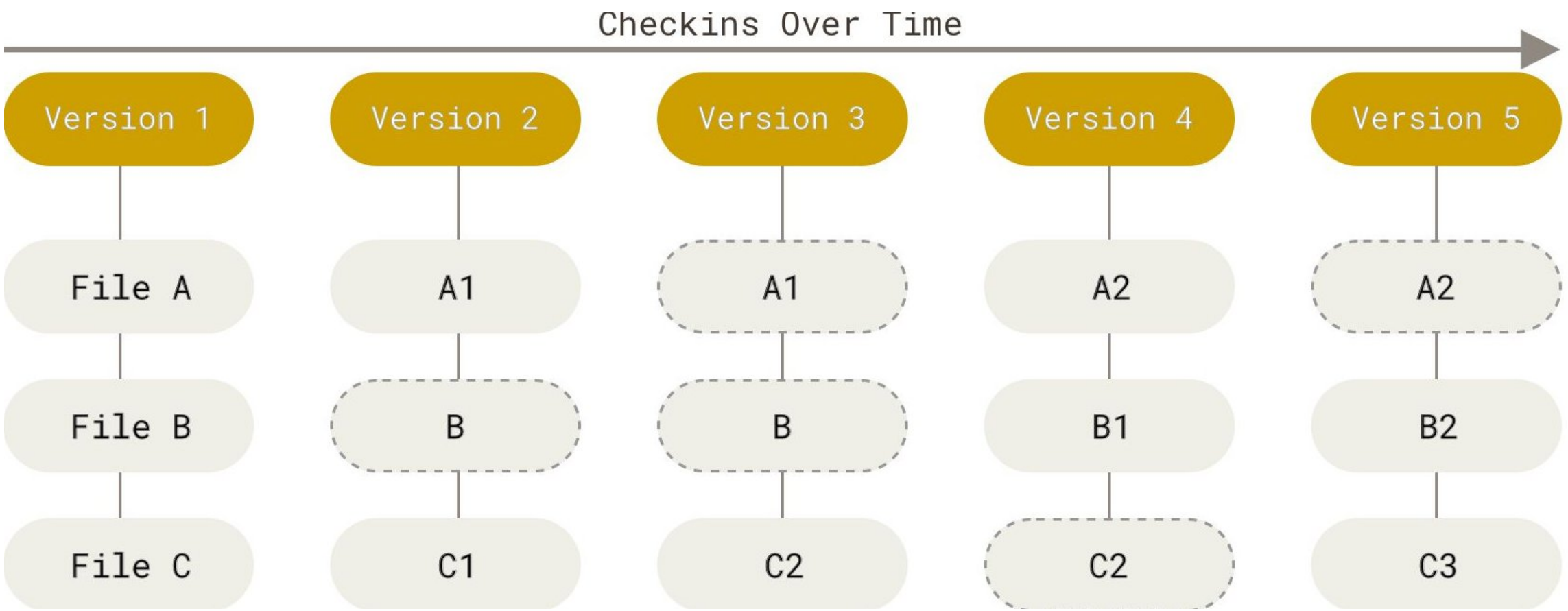
Global Information Tracker (Git)

- The major difference between Git and any other VCS is the way Git thinks about its data
  - Conceptually, most other systems store information as a list of file-based changes, and think of the information they store as a set of files and the changes made to each file over time (this is commonly described as delta-based version control).
  - Git thinks of its data more like a series of snapshots of a miniature filesystem. Every time you commit, or save the state of your project, Git stores a reference to that snapshot. To be efficient, if files have not changed, Git doesn't store the file again, just a link to the previous identical file it has already stored.

Most common VCS



Git


Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
File B
File C
A1
B
C1
A1
B
C2
A2
B1
C2
A2
B2
C3

Git

- Most operations in Git need only local files and resources to operate
  - For example, to browse the history of the project, Git doesn't need to go out to the server to get the history and display it for you, it simply reads it directly from your local database
  - If you want to see the changes introduced between the current version of a file and the file a month ago, Git can look up the file a month ago and do a local difference calculation

Git

- Everything in Git is checksummed before it is stored and is then referred to by that checksum.
  - This means it's impossible to change the contents of any file or directory without Git knowing about it.
- The mechanism that Git uses for this checksumming is called a SHA-1 hash. This is a 40-character string composed of hexadecimal characters (0–9 and a–f) and calculated based on the contents of a file or directory structure in Git.

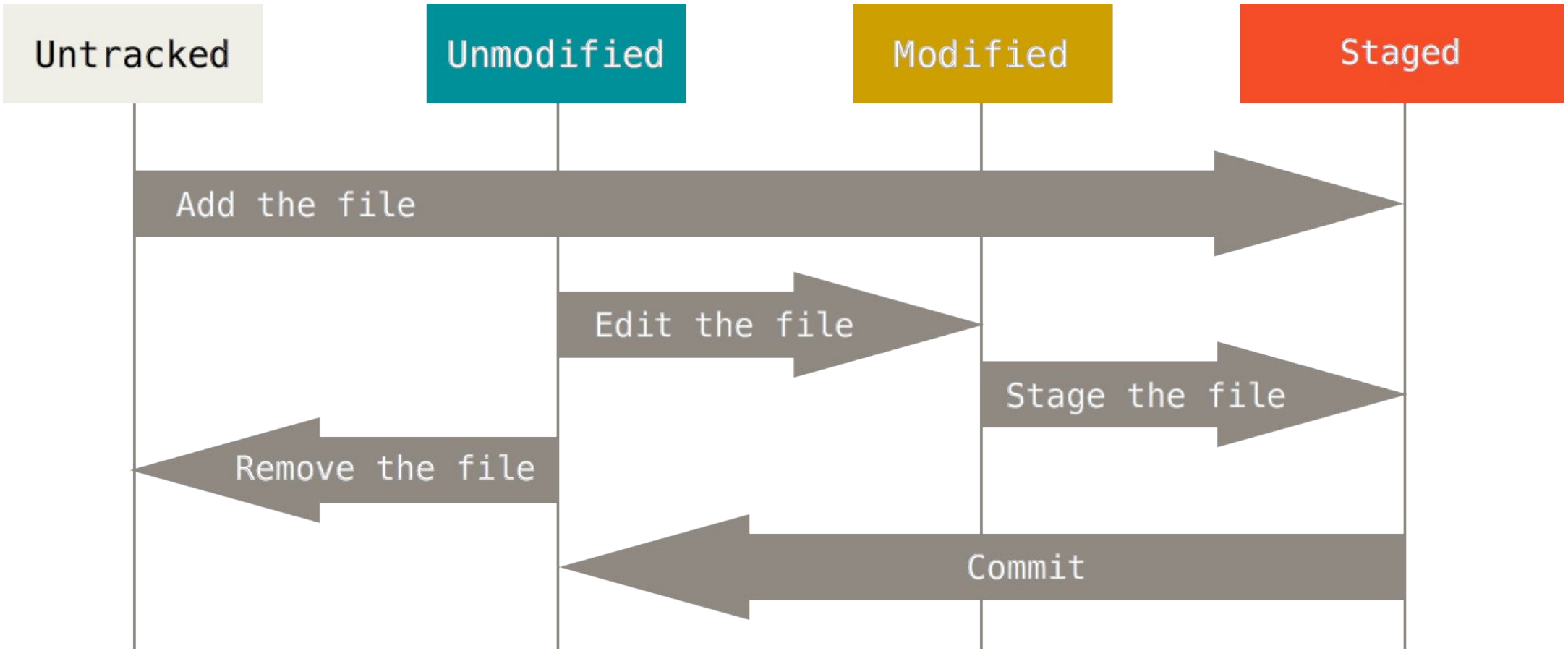
```
24b9da6552252987aa493b52f8696cd6d3b00373
```

Life Cycle

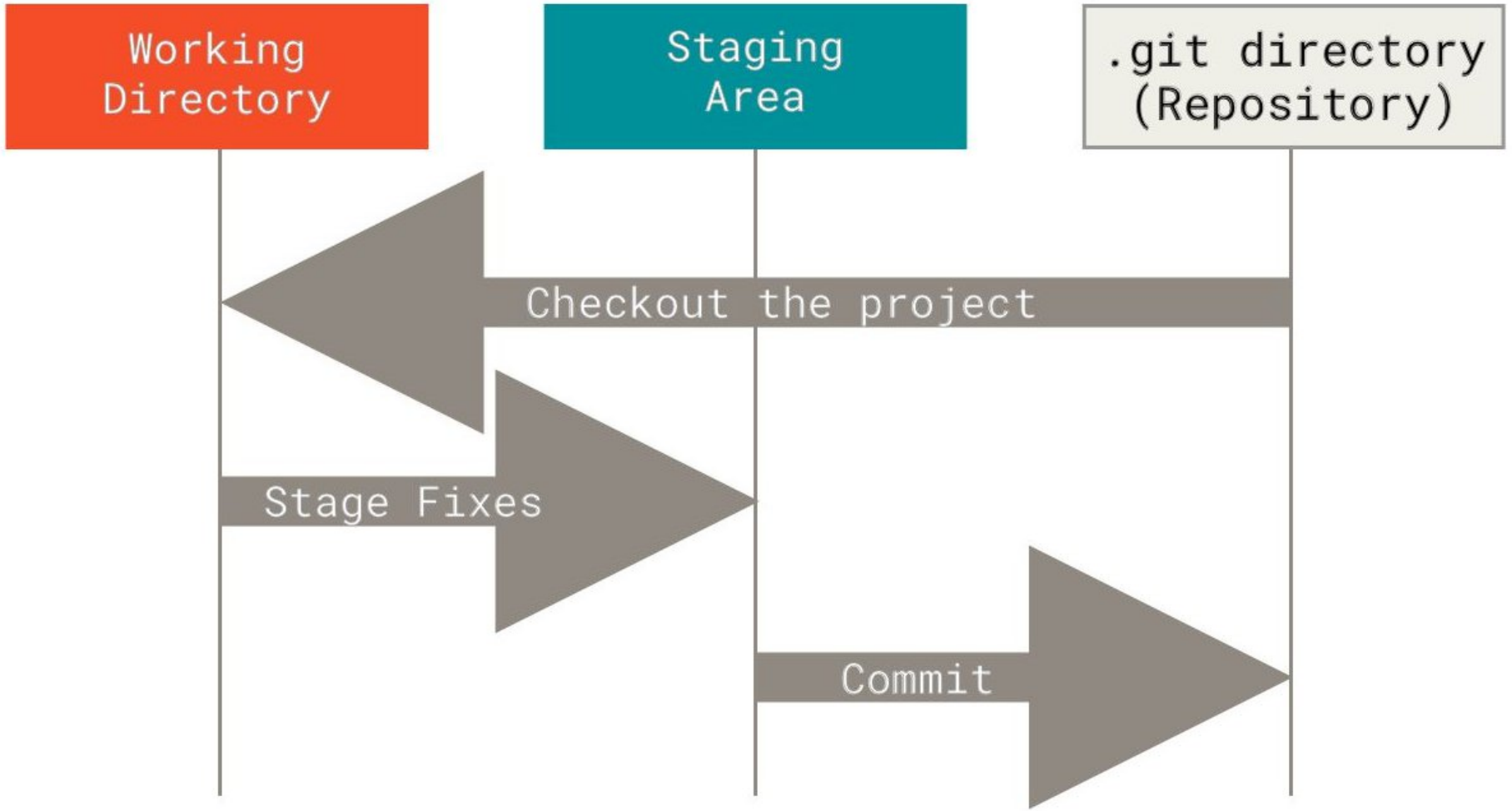
- Each file can have one of four different states:

  - Untracked: It's not listed in the last commit
  - Unmodified: It hasn't changed since the last commit
  - Modified: It has changed since the last commit
  - Staged: The changes will be recorded in the next commit made

Life Cycle


Untracked
Unmodified
Modified
Staged
Add the file
Edit the file
Stage the file
Remove the file
Commit

Main Sections of a Git Project



Main Sections of a Git Project

- The basic Git workflow goes something like this:

1. You modify files in your working directory (working tree).
2. You selectively stage just those changes you want to be part of your next commit, which adds only those changes to the staging area.
3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory (repository).

UPskill

Digital Skills & Jobs

Version Control

Metodologias de Trabalho em Equipa

P.PORTO

apdc ASSOCIAÇÃO PORTUGUESA PARA O DESENVOLVIMENTO DAS COMUNICAÇÕES

INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL

CONSELHO COORDENADOR DOS INSTITUTOS SUPERIORES POLITÉCNICOS