Git and Github

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

fit@hcmus

What is Version Control

- Version control systems (VCSs) are tools used to track changes to source code (or other collections of files and folders).
  - Help maintain a history of changes
  - Facilitate collaboration.
  - 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. In addition, you get all this for very little overhead.

What is Version Control

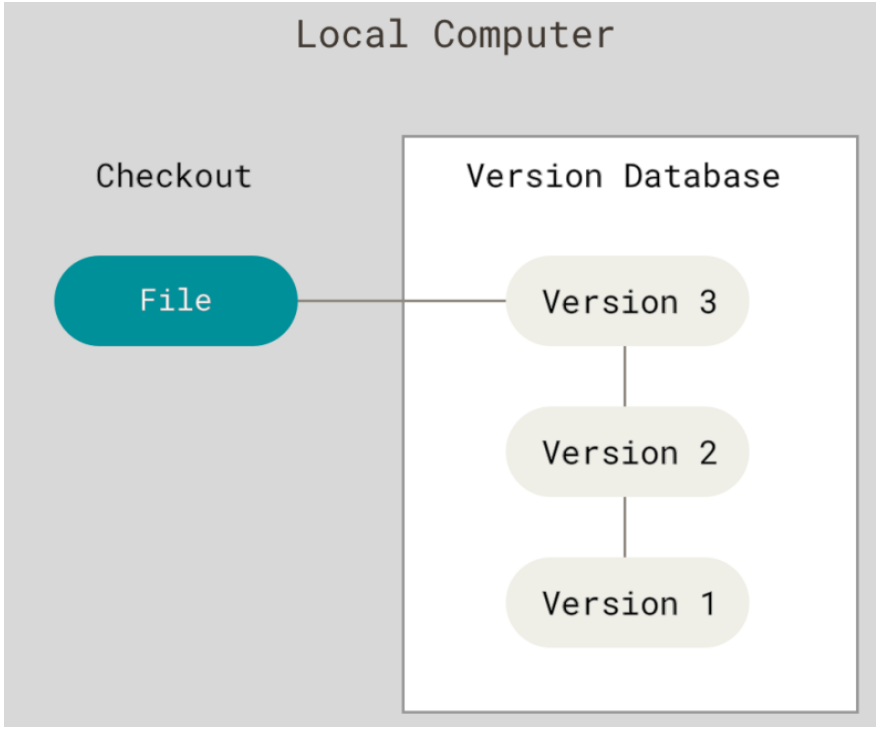
- Modern VCSs also let you easily (and often automatically) answer questions like:
  - Who wrote this module?
  - When was this particular line of this particular file edited? By whom? Why was it edited?
  - Over the last 1000 revisions, when/why did a particular unit test stop working?

Local Version Control Systems

- Many people's version-control method of choice is to copy files into another directory
  - Simple, but error prone
- Programmers long ago developed local VCSs that had a simple database that kept all the changes to files under revision control.

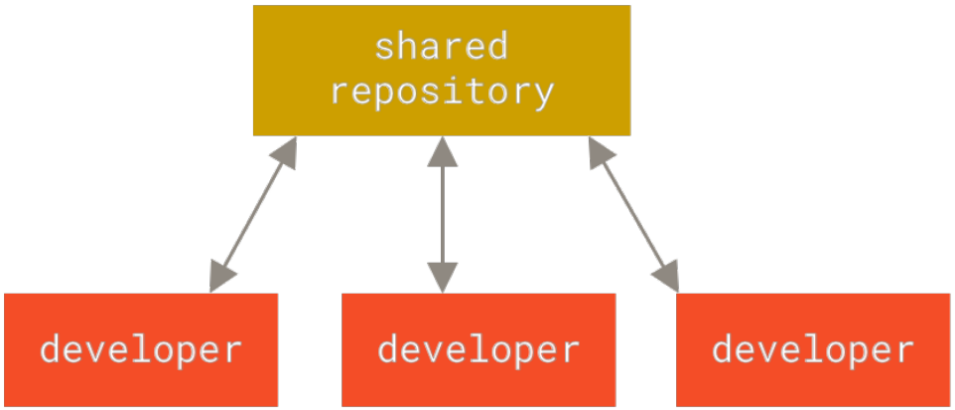
Local Version Control Systems

- Ex: RCS
- Keeping patch sets (the differences between files) in a special format on disk
- Can then re-create what any file looked like at any point in time by adding up all the patches



Centralized Version Control Systems

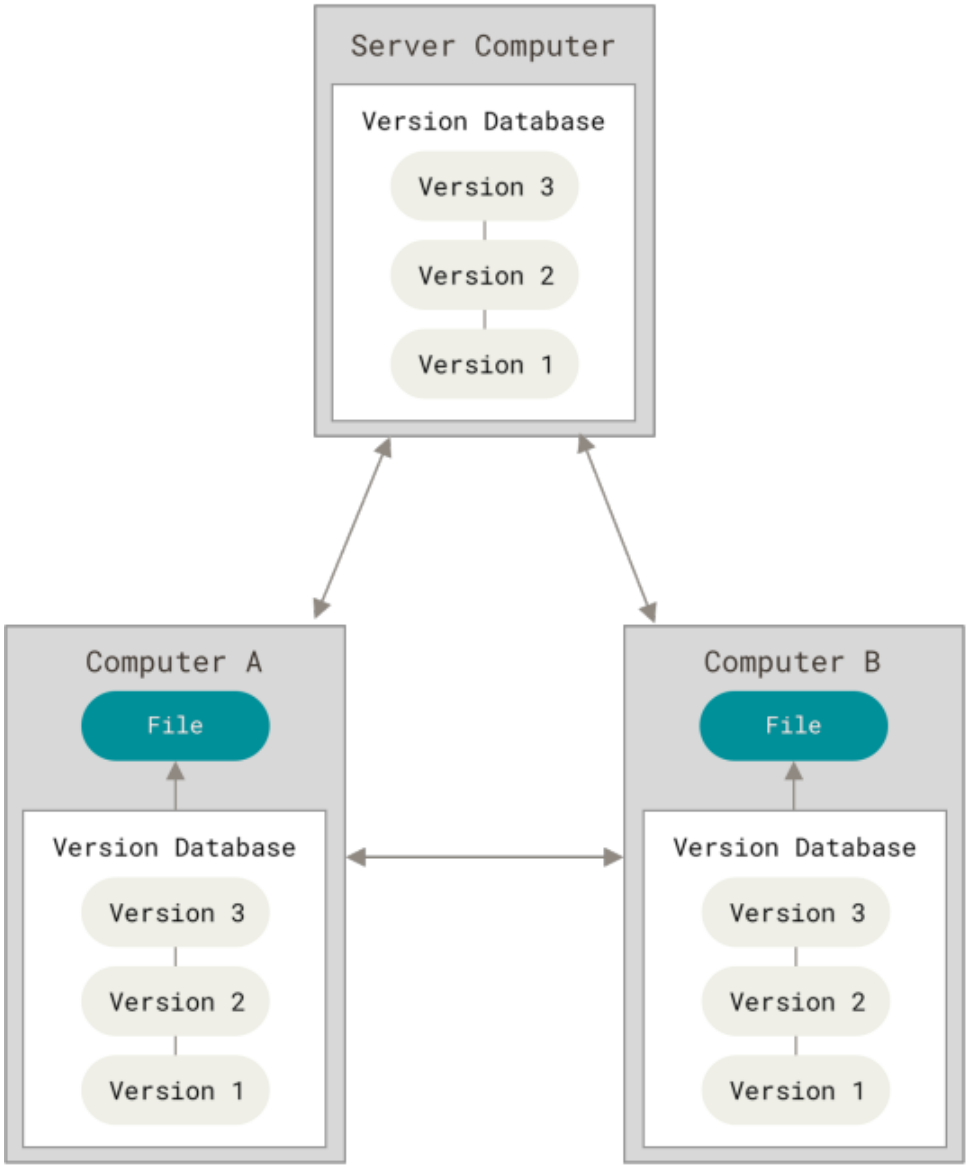
- To collaborate with developers on other systems
- These systems (CVS, Subversion, and Perforce) have a single server that contains all the versioned files, and a number of clients that check out files from that central place.



Centralized Version Control Systems

- **Advantages**
  - Everyone knows to a certain degree what everyone else on the project is doing.
  - Administrators have fine-grained control over who can do what, and it's far easier to administer a CVCS than local databases on every client.
- **Drawback**
  - Single point of failure: If that server goes down for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they're working on.
  - If the hard disk the central database is on becomes corrupted, and proper backups haven't been kept, you lose absolutely everything

Distributed Version Control Systems



Distributed Version Control Systems

- In a DVCS (such as Git, Mercurial, Bazaar or Darcs), clients don’t just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history
- Every clone is really a full backup of all the data.
- Can collaborate with different groups of people in different ways simultaneously within the same project - Hierarchical models

What is Git

- **It is a free, high-quality distributed version control system suitable for tracking modifications in source code in software development**
- Git is the tool created by Linux Torvalds to do version control and to support collaboration, originally for Linux source code when it was published and many people joined to contribute to it
- Now Git is used widely in companies; when you work for companies in the future, it's likely that you will have to use Git

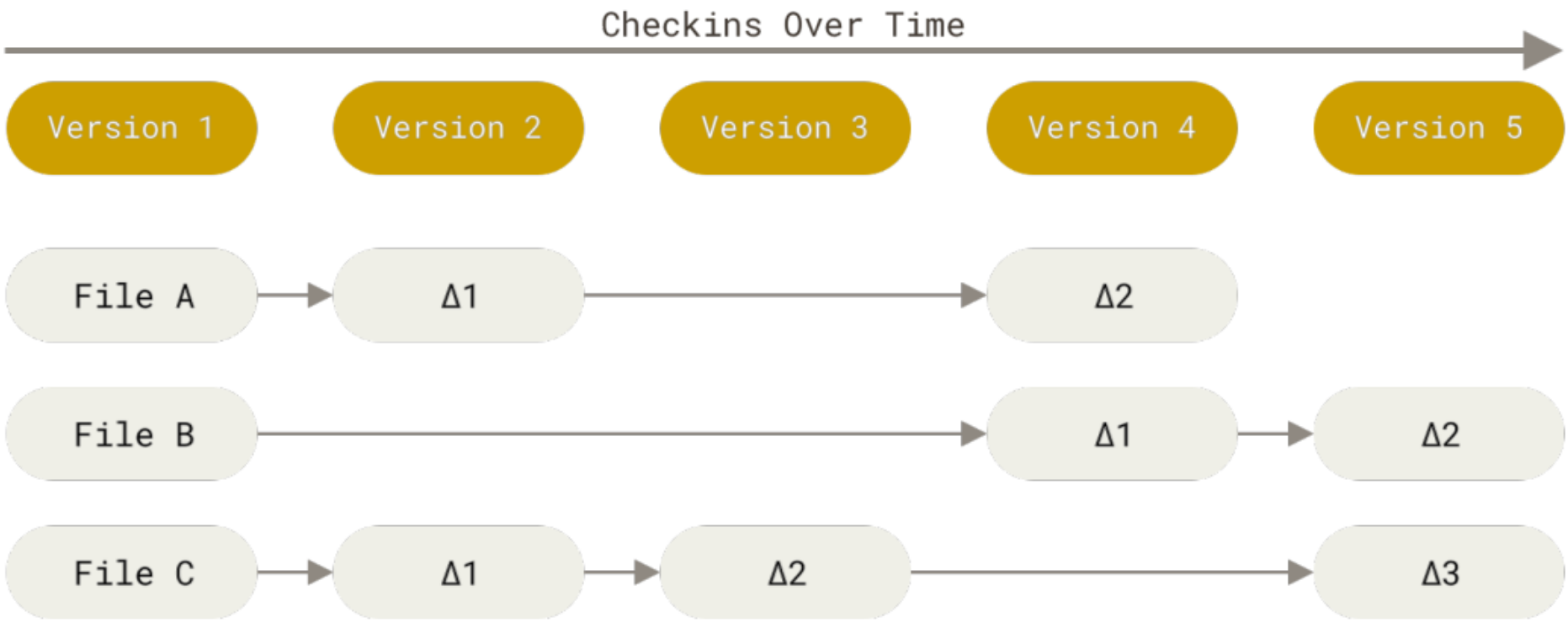
What is Git


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

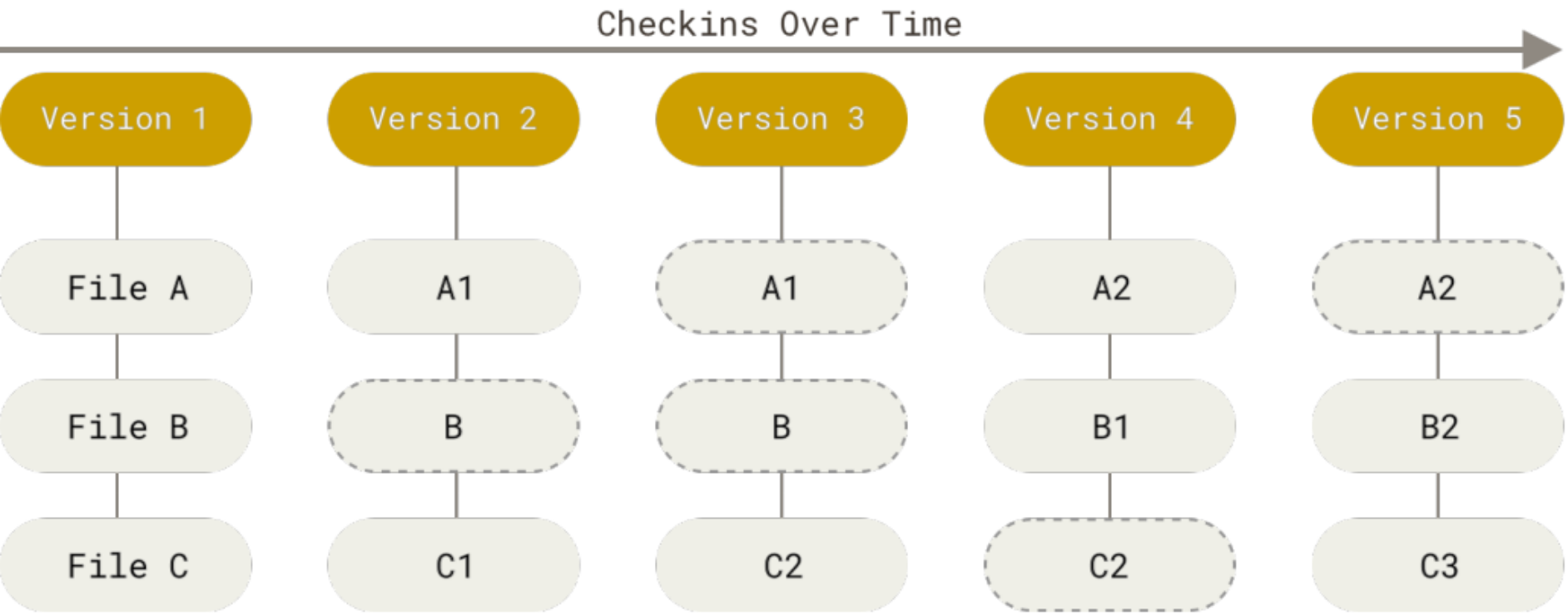
What is Github

- It is a web-based Git repository. This hosting service has cloud-based storage. GitHub offers all distributed version control and source code management functionality of Git while adding its own features. It makes it easier to collaborate using Git.
- Additionally, GitHub repositories are open to the public. Developers worldwide can interact and contribute to one another's code, modify or improve it, making GitHub a networking site for web professionals. The process of interaction and contribution is also called social coding.

Delta-based version control



Snapshot version control



Snapshot

- Git models the history of a collection of files and folders within some top-level directory as a series of snapshots.
- A file is called a “blob”
- A directory is called a “tree”, and it maps names to blobs or trees

```
<root> (tree)
|
+- foo (tree)
|  |
|  + bar.txt (blob, contents = "hello world")
|
+- baz.txt (blob, contents = "git is wonderful")
```

Snapshot

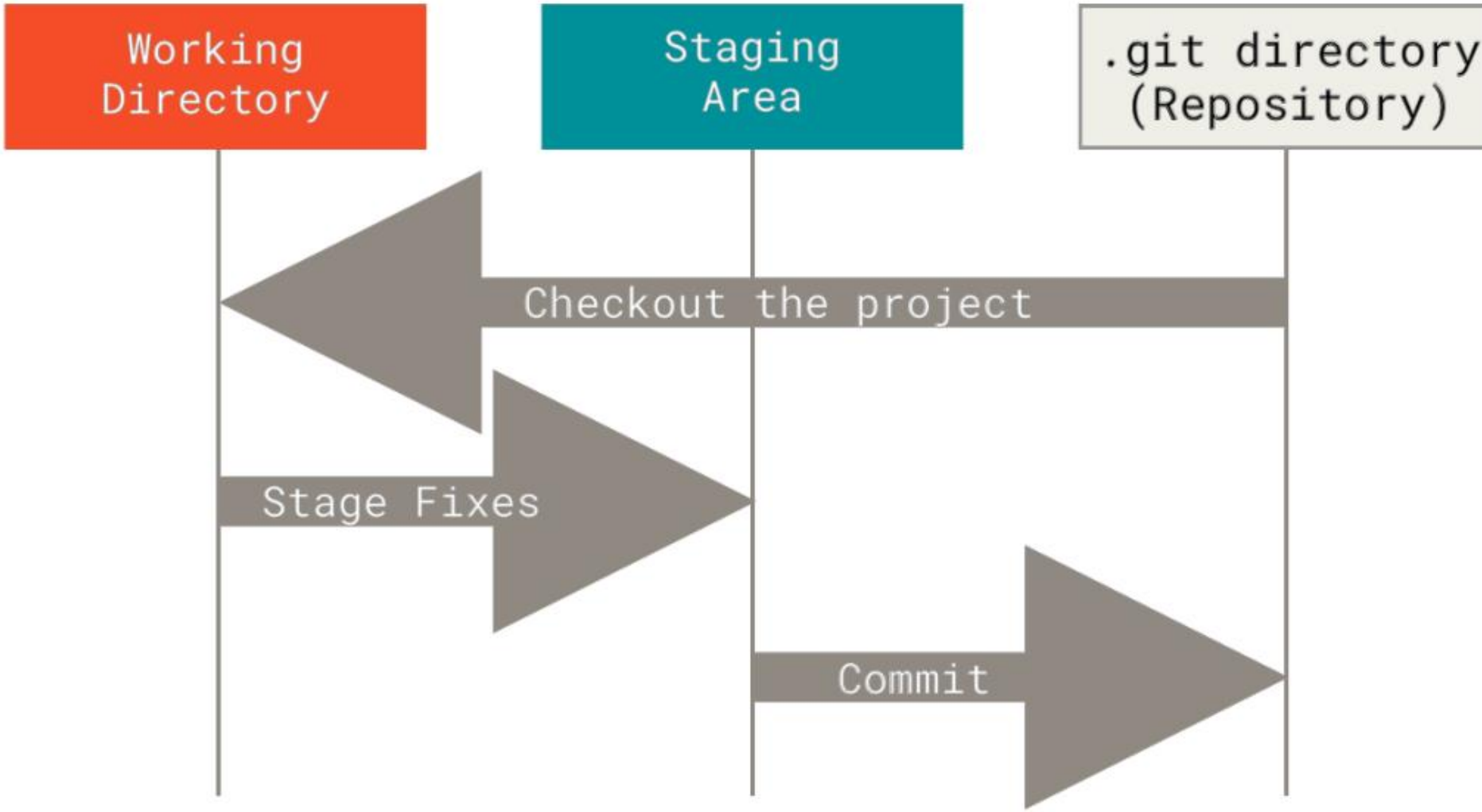
- Git models the history of a collection of files and folders within some top-level directory as a series of snapshots.
- A file is called a “blob”
- A directory is called a “tree”, and it maps names to blobs or trees

```
<root> (tree)
|
+- foo (tree)
|  |
|  + bar.txt (blob, contents = "hello world")
|
+- baz.txt (blob, contents = "git is wonderful")
```

Basic Git: Three stage

- Git has three main states that your files can reside in: modified, staged, and committed:
- **Modified** means that you have changed the file but have not committed it to your database yet.
- **Staged** means that you have marked a modified file in its current version to go into your next commit snapshot.
- **Committed** means that the data is safely stored in your local database.

Basic Git: Three stage



Basic Git: Three stage

- Git has three main states that your files can reside in: modified, staged, and committed:
- **Modified** means that you have changed the file but have not committed it to your database yet.
- **Staged** means that you have marked a modified file in its current version to go into your next commit snapshot.
- **Committed** means that the data is safely stored in your local database.
- Three main sections of a Git project: the **working tree**, the **staging area**, and the **Git directory**

Basic Git: Three stage

- The working tree is a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.
- The staging area is a file, generally contained in your Git directory, that stores information about what will go into your next commit.
- The Git directory is where Git stores the metadata and object database for your project. This is the most important part of Git, and it is what is copied when you clone a repository from another computer.

Basic Git workflow

- 1. You modify files in your 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.

BASIC GIT

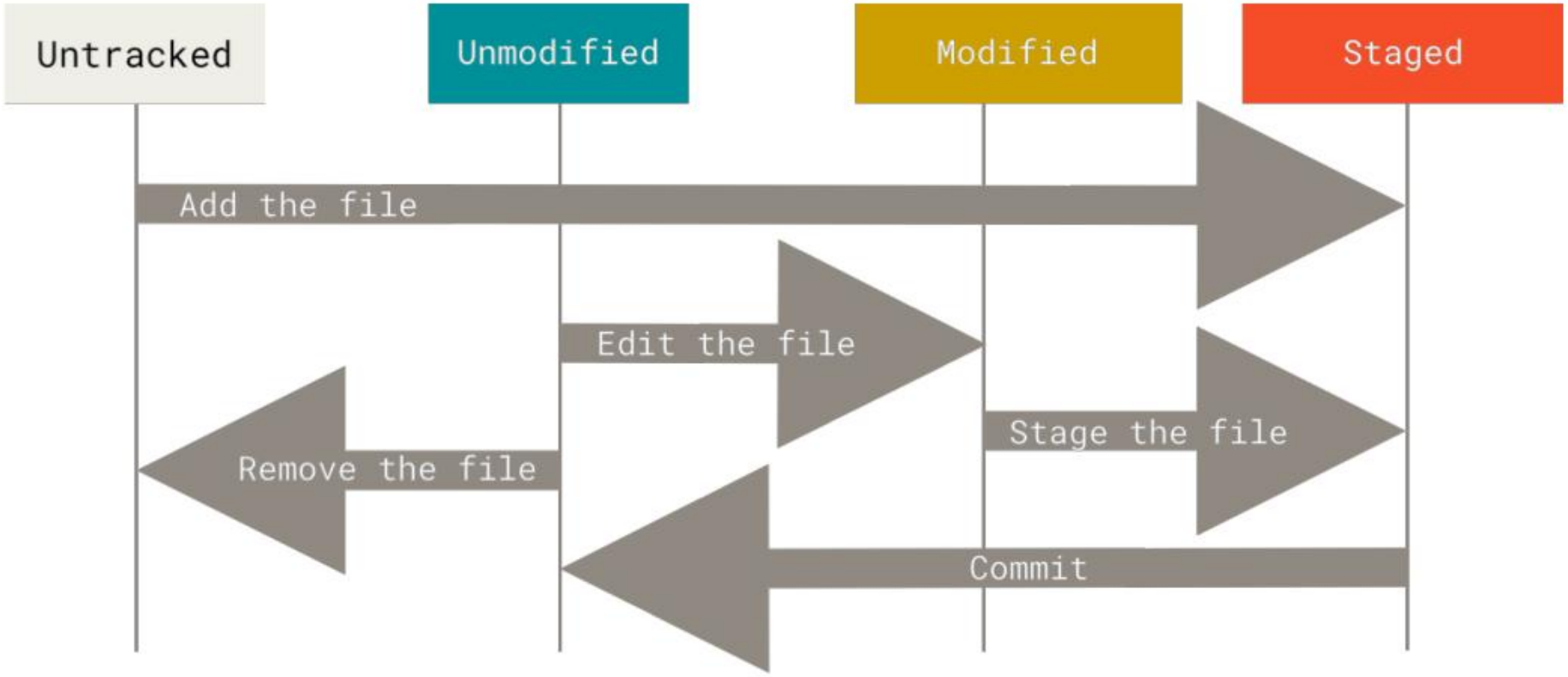
Git Repository

- You typically obtain a Git repository in one of two ways:
- 1. You can take a local directory that is currently not under version control, and turn it into a Git repository, or
- 2. You can clone an existing Git repository from elsewhere.

Initializing a Repository

- Existing Directory: go to that project's directory:
  - $ git init
- Cloning an Existing Repository:
  - $ git clone https://github.com/libgit2/libgit2

Initializing a Repository



Checking the Status of Your Files

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
```

Reference

- https://missing.csail.mit.edu/2020/version-control/
- https://git-scm.com/book/en/v2