Git and Github

Learning objectives

- Understand and explain what a `version control system` is
- Use basic Git commands such as `git add`, `git commit`, `git pull`, and `git push`.
- Pull and push between your local `repo` to a remote repo

What is version control?

`Version control` is a way for developers to manage large projects together to create stable version of software that can be run independently. For example, if you were working on the logging in functionality of a blog, and you finished the `login` part, but not the `logout` part, you might want to save your `work in progress (WIP)` but not want to publish that to the Internet.

What is version control? (cont.)

A `version control system` allows you do collaboratively program in small increments while maintaining an authoritative or stable version of your software for end-users that is separate from your WIP that other developers can contribute to while you roll out new features.

Version control helps you keep track of and manage changes to source code by multiple authors over time.

What is Git?

`Git` is a version control system. There are other competitors like `Apache Subversion (SVN)` and `Mercurial`.

Reference: `https://git-scm.com/doc`

History of Git

The original creator of Git is `Linus Torvalds`, the creator of `Linux`. Just about all server software today runs on Linux, and a ton of other software inherits or takes advantage of the free or `open source software` movement (including Android and Mac OS X).

Some argue that Linus' great contribution to the world was not actually Linux, his namesake for which he is famous, but Git, a tool he built to facilitate the development of Linux.

Why did Git get big?

Open source software is one of the main reasons why Git started to overtake other version control systems such as SVN and Mercurial.

What is Github?

`Github` is a web-based Git hosting service.

Competitors to it include `Gitlab`, `Bitbucket`, etc. These competitors offer different advantages over Github, but generally Github is the most well-known of the bunch and biggest host of open-source software.

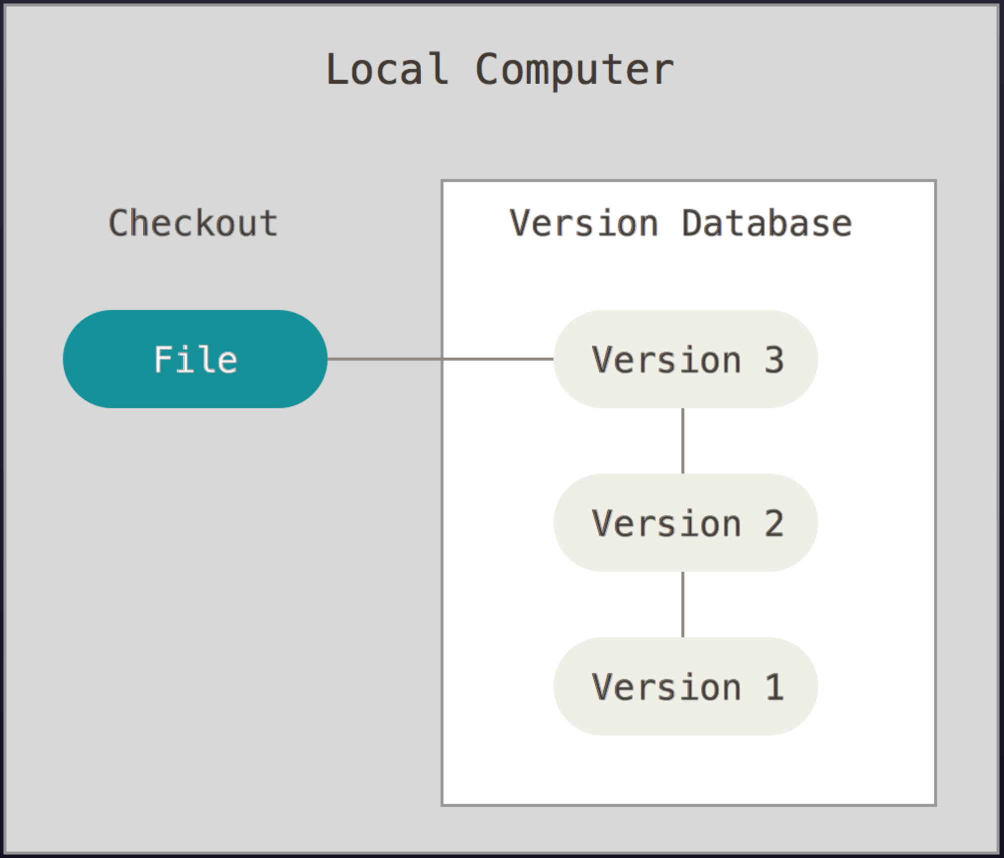
Version control overview

Let's take a look at how version control system evolved over time and how Git fits into that picture.

First, we have a `local version control system`, that looks like this:

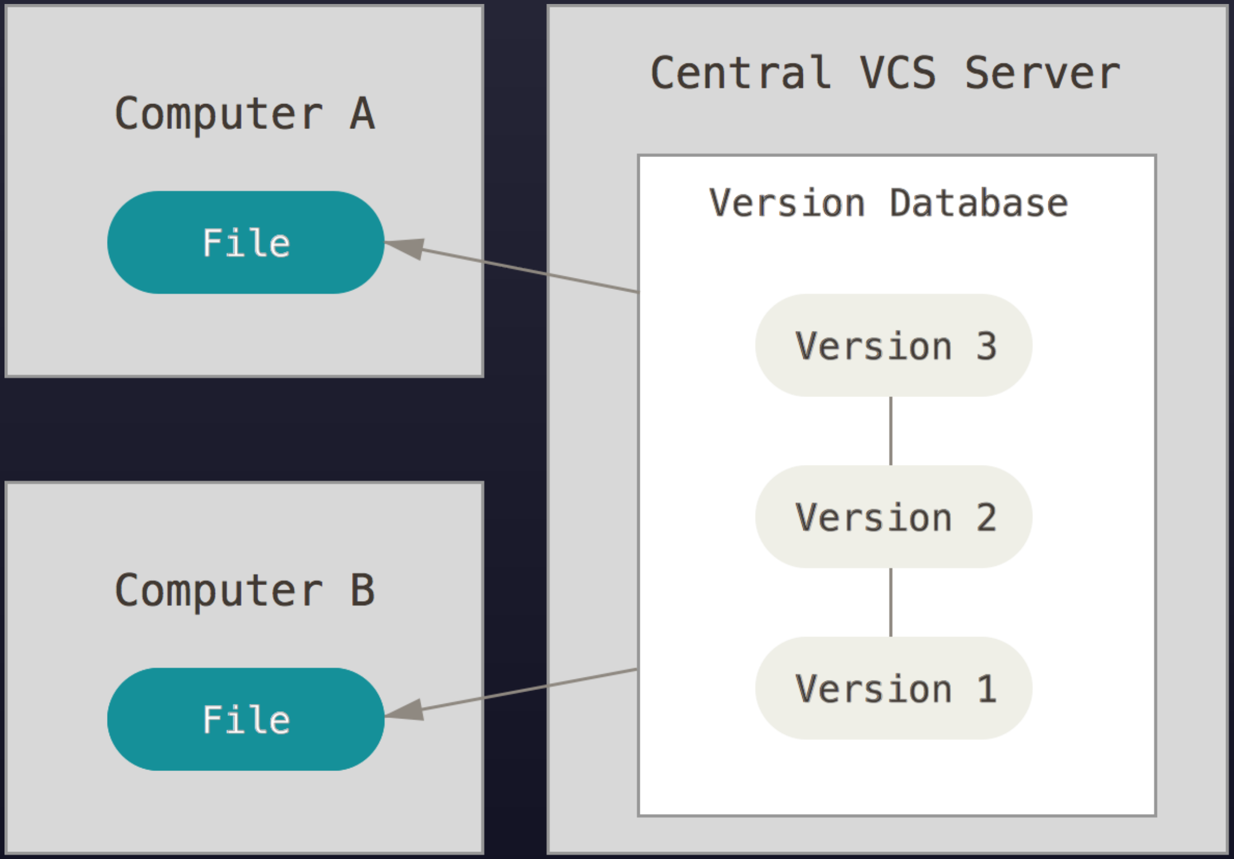
Local version control system

Manually create new directories on your computer to maintain separate versions of your software over time,

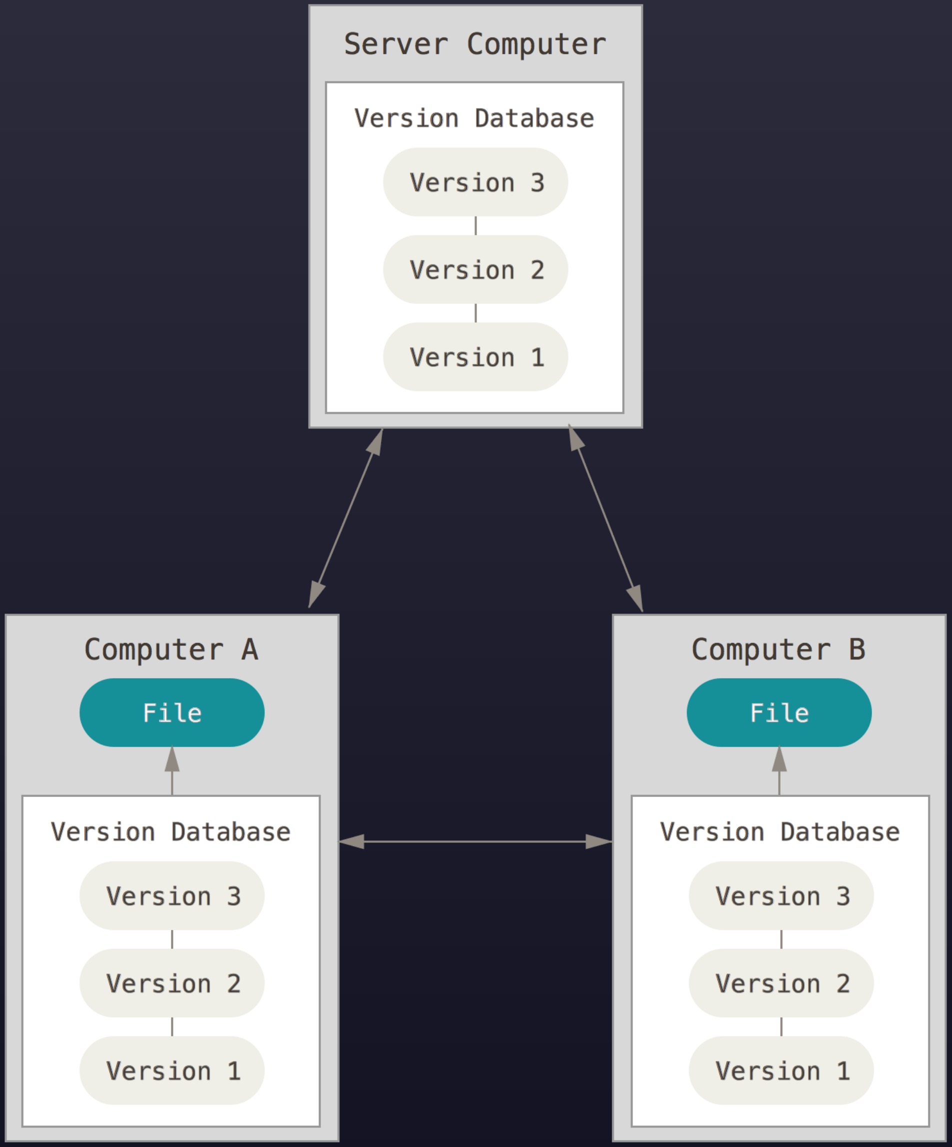


Centralized version control system

SVN is an example of this. Major downside is single point of failure,



Distributed version control system

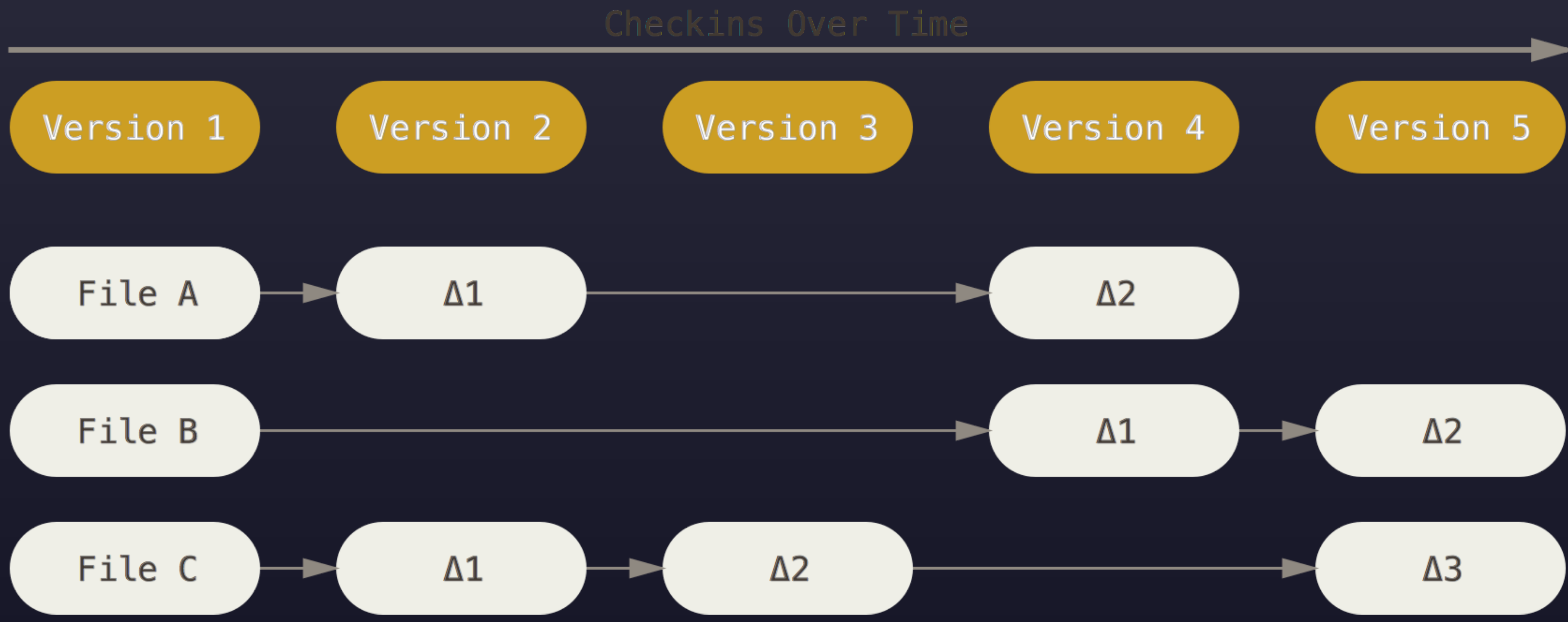


Break

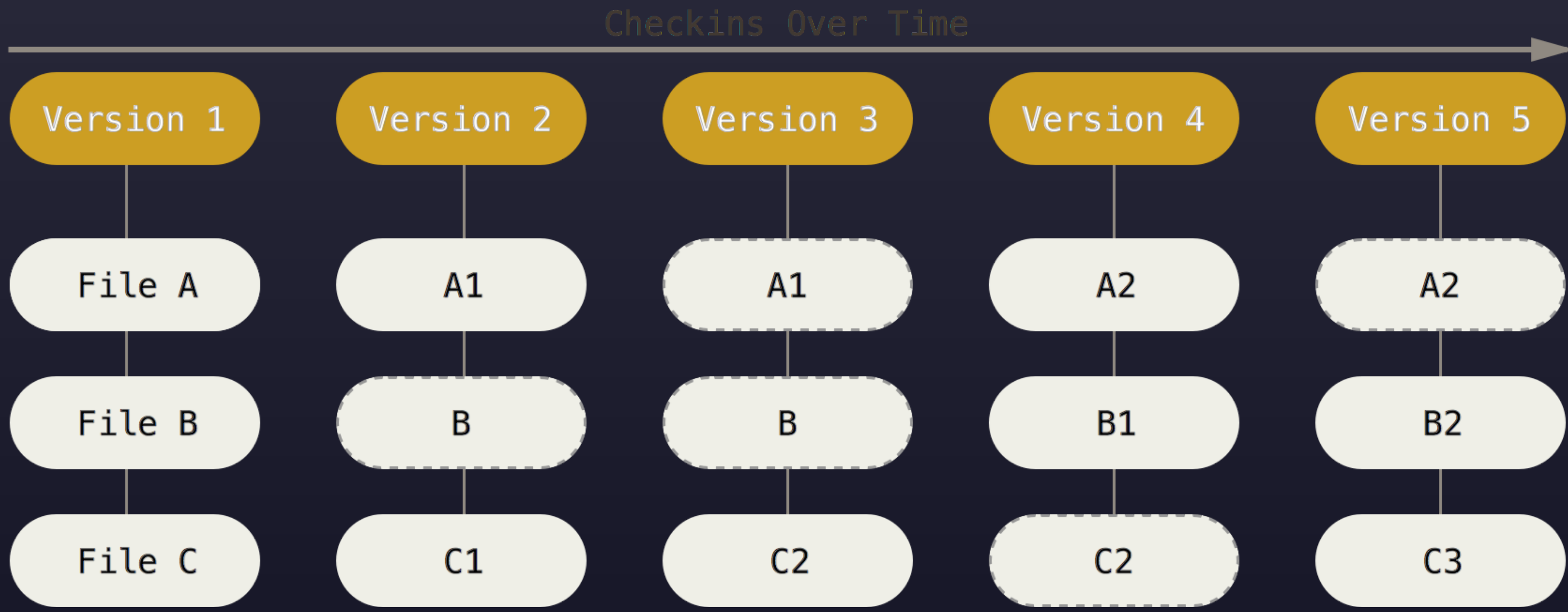
Make sure everyone is signed up for `Github`. Go there now and sign up.

Also, make sure you have Git installed locally. Go into terminal and run `$ git` and see if it prints out the help message.

Snapshots vs. Differences

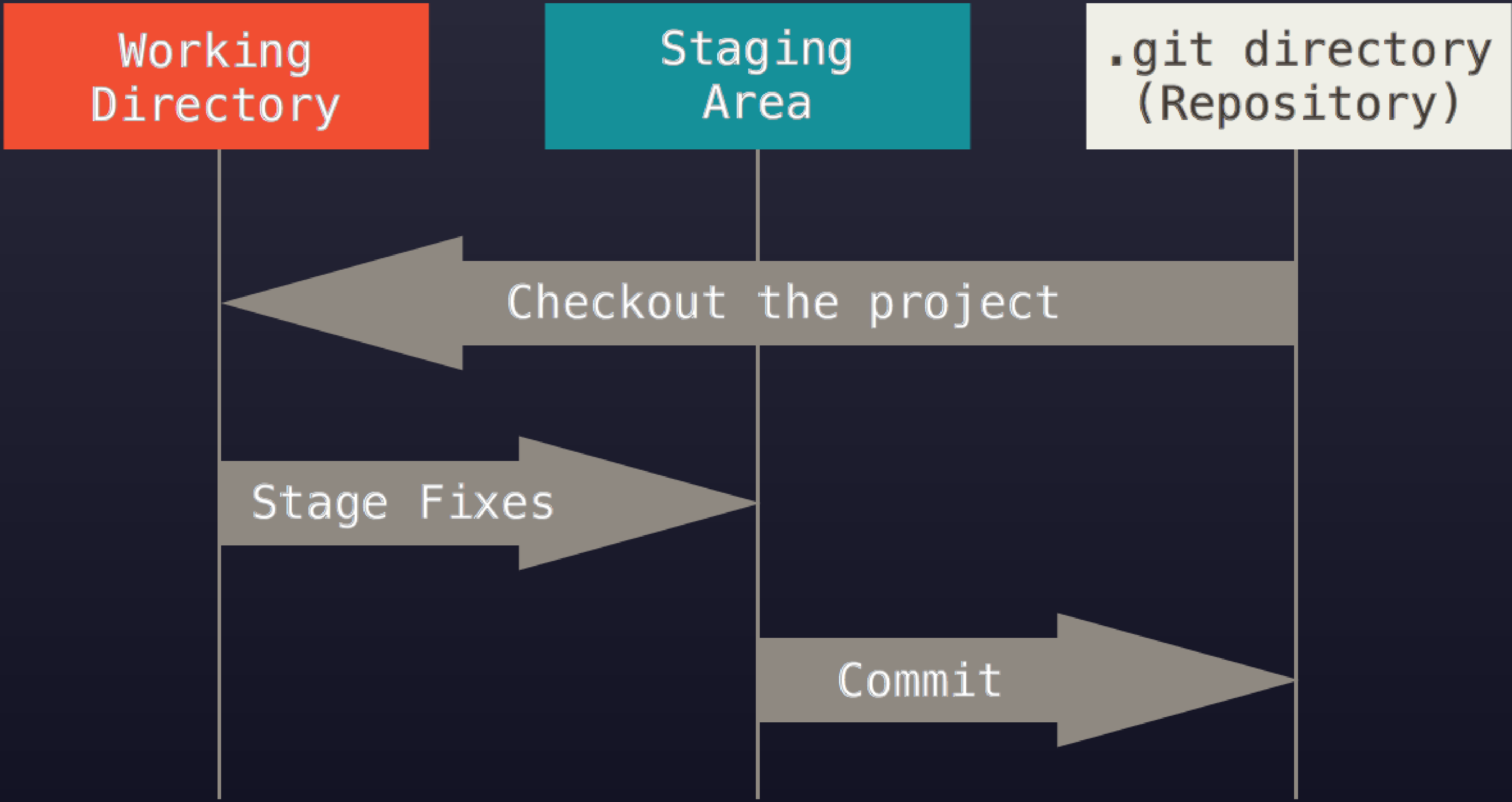


Snapshots vs. Differences (cont.)



The three states

State or context is **very** important in Git.



Let's learn some Git

Open up your cheatsheets.

Getting and creating projects

- `git init`
- `git clone`

Snapshots of your files

- `git add`
- `git status`
- `git diff`
- `git commit`
  - `-m <git message>`
  - `-- amend`
- `git reset`
- `git rm`
- `git mv`
- `git stash`

Branching and merging

- `git branch`
- `git checkout`
  - `-b`
- `git merge`
- `git log`
- `git cherry-pick <commit>`

Sharing and updating remote projects

- `git remote`
- `git fetch`
- `git pull <remote> <branch-in-remote>`
    - `-- rebase`
- `git push <remote> <branch-in-remote>`
    - `-f`

Inspection and comparison

- `git log`
- `git diff`