VERSION CONTROL

A Version Control System (VCS) maintains a record of the evolving state of a project.

Using a VCS you will be able to:

- Create ongoing, incremental **backup** of your code as you develop it.
- **Collaborate** in a straightforward way with other developers/teachers.
- Create a quasi-automatic **documentation** of the changes you or your collaborators make.
- **Revert** in a straightforward way to a previously working state of a of broken code.

We will make use of the Git VCS (https://git-scm.com).

There are multiple ways of using Git: local and online repositories, we will use the online github.com (github.com) to manage repositories.

You will use **Git** to **maintain** the code you develop and to **retrieve** and **submit** homework assignments.

You can find extensive documentation in the online GitPro book, available here (we will make extensive use of this book).

With Git you save a picture of what all your files look like at that moment.

If files have not changed, Git only stores a link to the previous identical version.

Git thinks about its data more like a **stream of snapshots**.

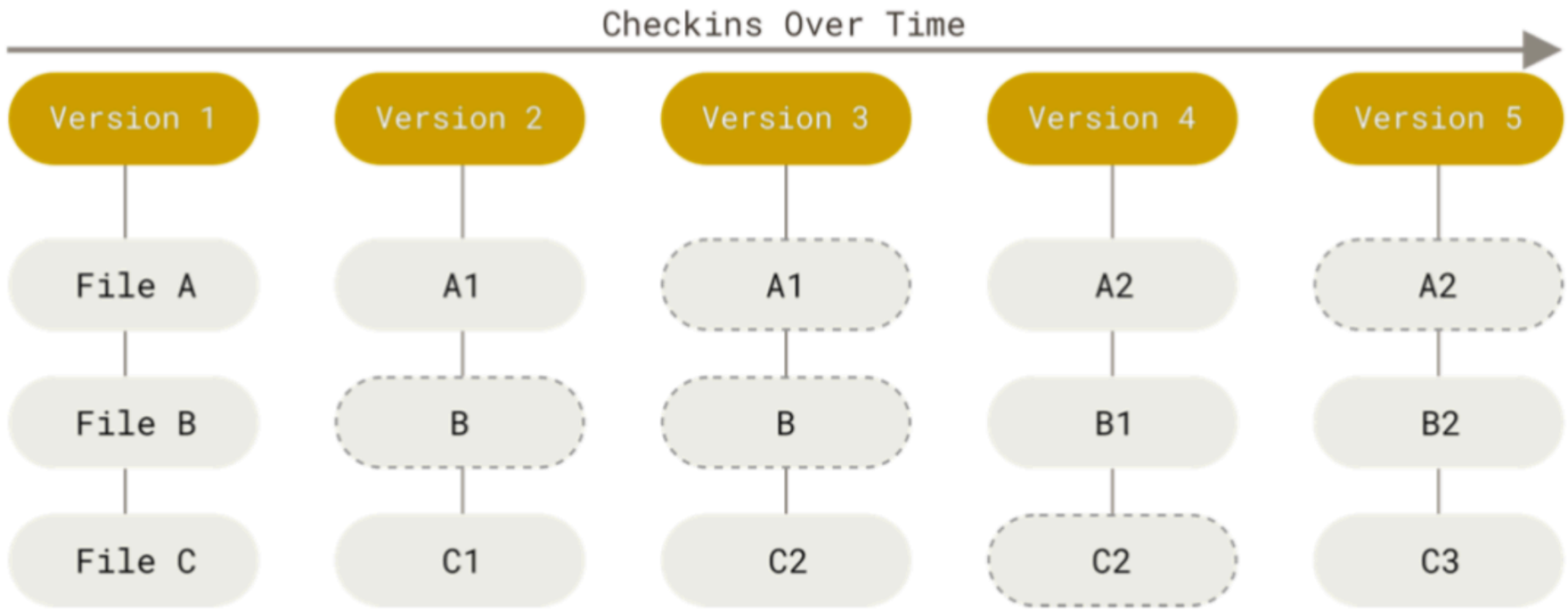


Figure 5. Storing data as snapshots of the project over time

Source: GitPro book

GIT BASICS

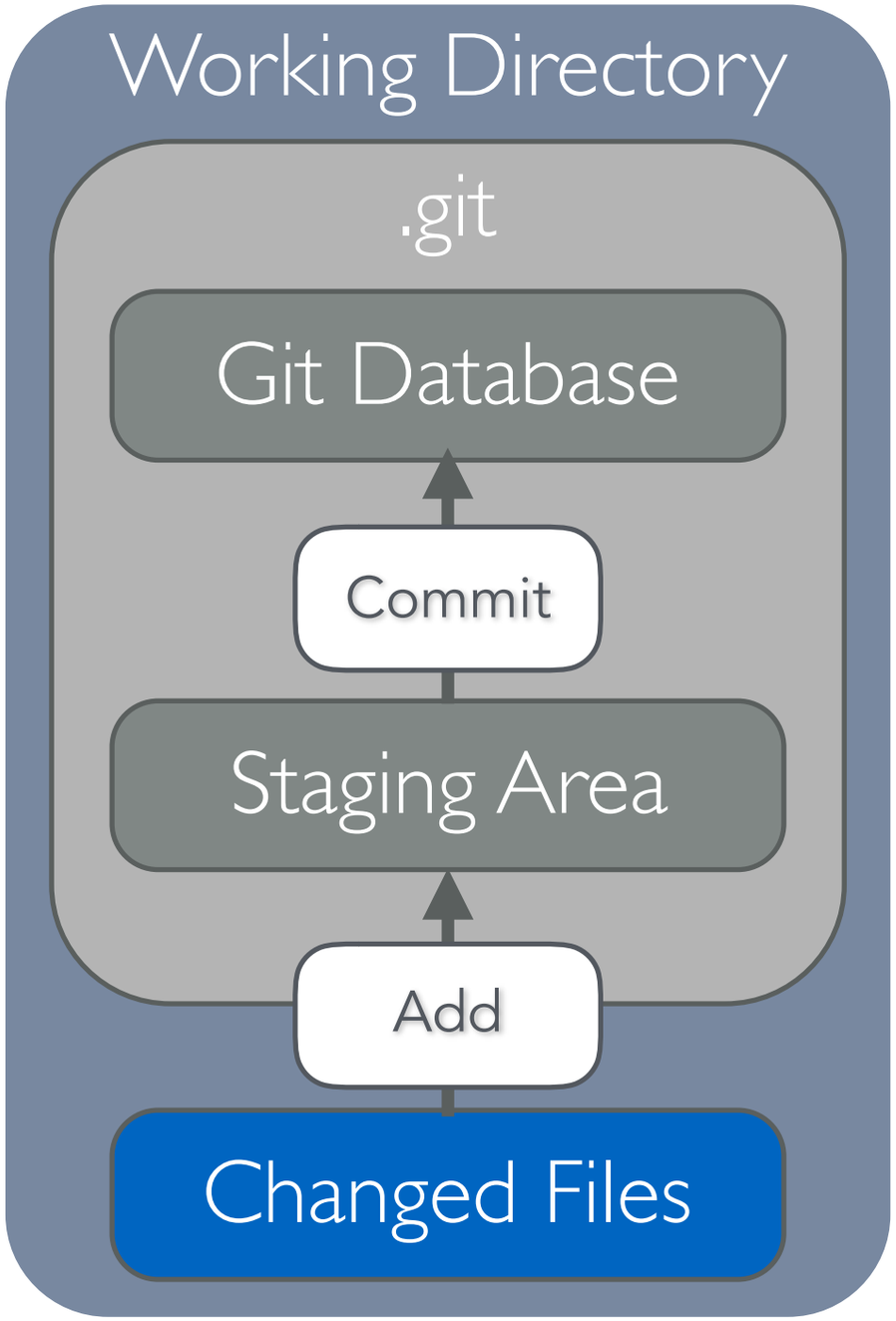
Git Pro Book
Chapter 1.3

- A directory containing files that are being monitored by Git is called a **WORKING DIRECTORY**.
- Every working directory contains a **hidden subdirectory** called `.git`.
  - A **HIDDEN DIRECTORY** or **HIDDEN FILE** has a name that begins with '`.`'. It will not appear in GUI windows.
  - Hidden files **can** be listed using the `ls` shell command with a special `-a` flag (> ls -a)
- The `.git` directory contains the **database of snapshots** representing previous states of the working directory.
- Most operations in Git need only local files: Git reads the history of your working directory directly from your local database.

THE THREE STATES

- For Git your files can reside in only three main states: *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.

SAVING LOCAL CHANGES



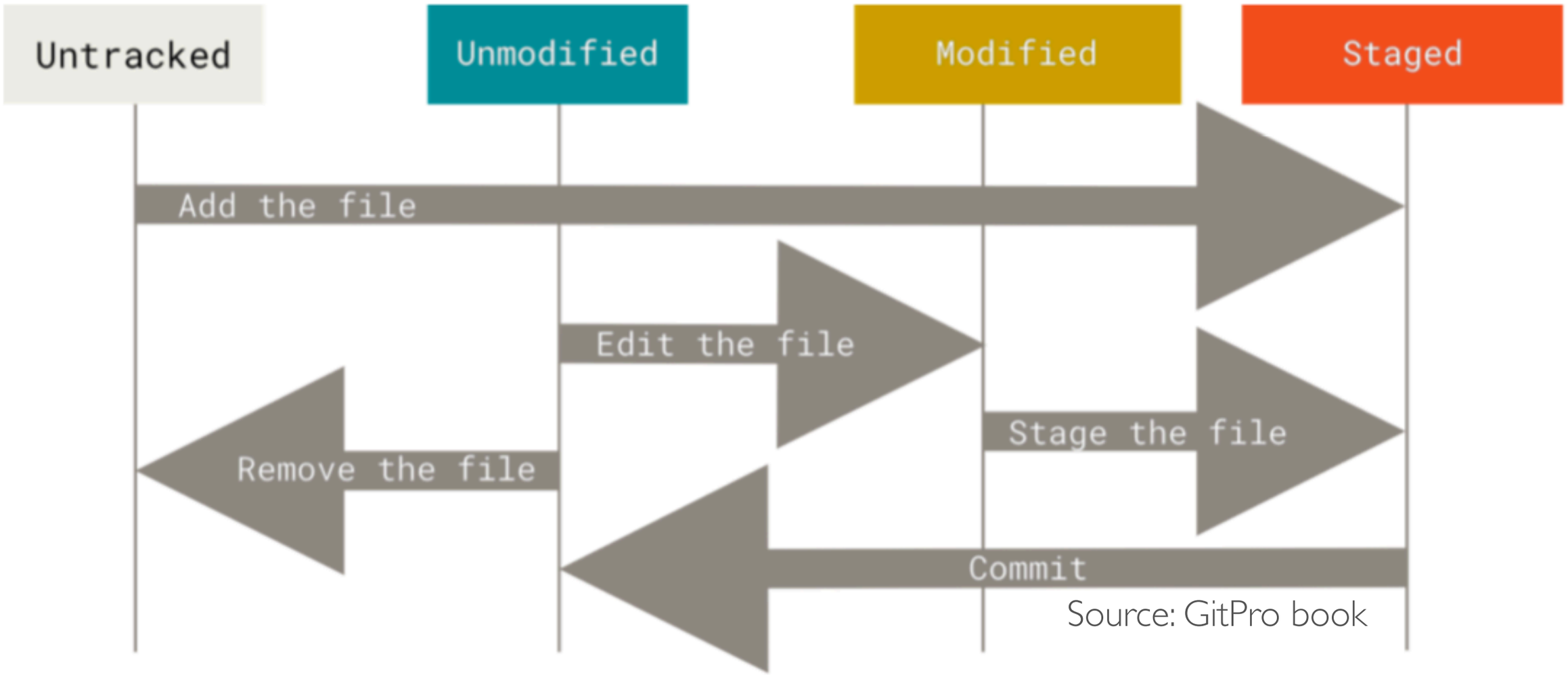
There are three main sections of a Git Project: the Working Directory, the staging area and .git repository.

- The working directory contains your files.
- The staging area is a file that stores information about what will go into your next commit.
- The Git directory is where Git stores the metadata (describing the change history of each file in the working directory) and object database for your project.

GIT BASICS

Git is installed in the CSE computer labs.

- Set your user name and email address.
  - > git config --global user.name “Claudia"
  - > git config --global user.email claudia.scarlata@gmail.com
- Create a working directory and go into it
  - > mkdir DSMMA
  - > cd DSMMA
- Initialize a git repository
  - > git init
- Add a README file to it
  - > touch README



Source: GitPro book

- Check the status of your files
  - > git status

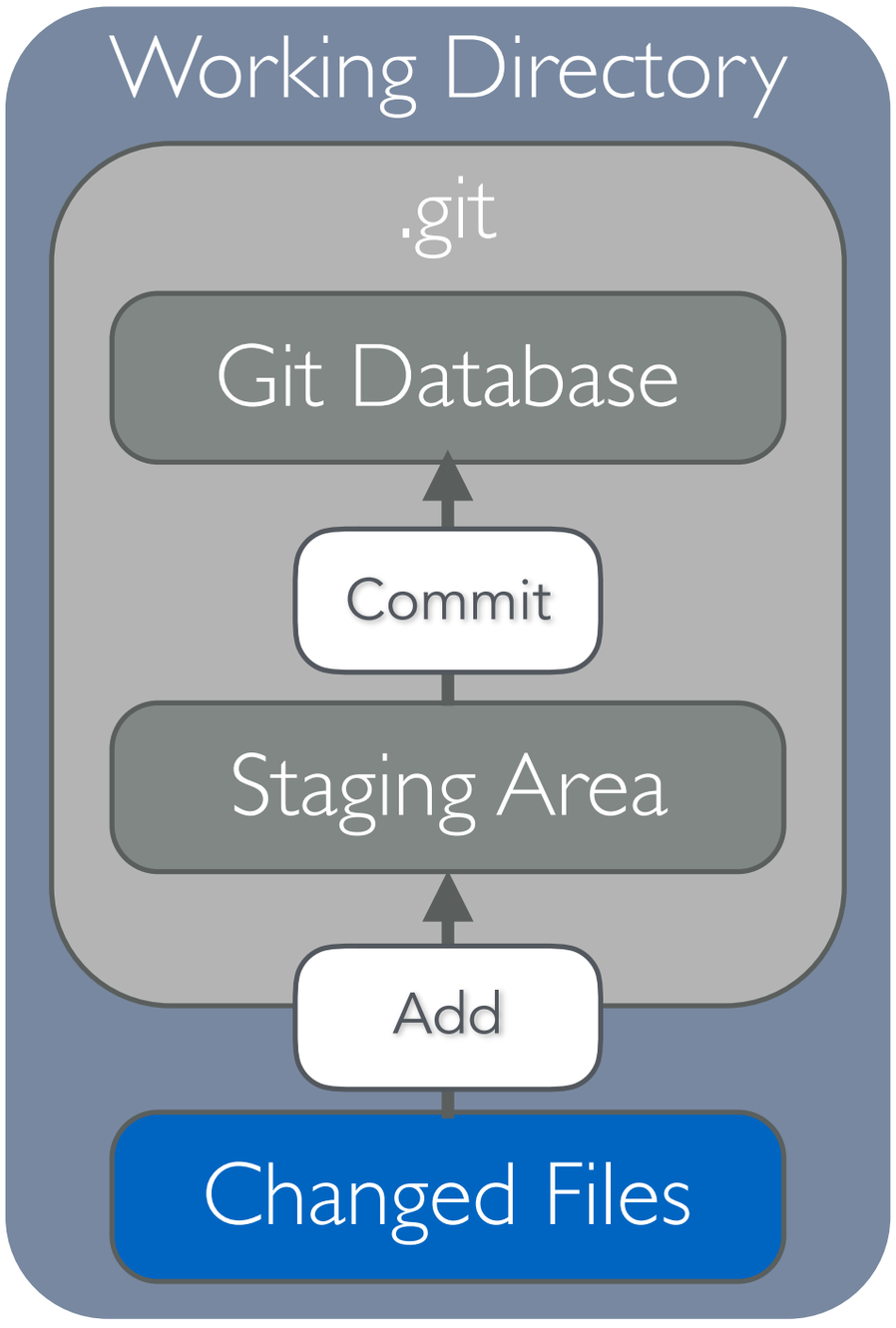
The README file is under the “Untracked files” heading in the status output.

- Tracking new files
  - > git add README

The README file is now “staged” because it appears in the changes to be committed

- Change the file and see how its status changes. It will now appear as modified.

GIT BASICS



After the staging area is set, we can commit our changes. The simplest way to commit is to type
> git commit -m ”comment"

Note : if you forger the comment git will open the default editor. The comment is very important! Document your changes. You can see the all the commit history with:

> git log

All steps up to here save local changes.

GITHUB

- **GitHub** provides online storage and management for git repositories.
- For this course, you will need a **GitHub account**.
- If you **do not** already have a GitHub account, you can **create one**. Start by navigating to:

https://github.com

CLONING REPOSITORIES

It is possible to create a local copy of an existing Git repository (for example a code you would like to contribute to).

> git clone url_of_repository local_path

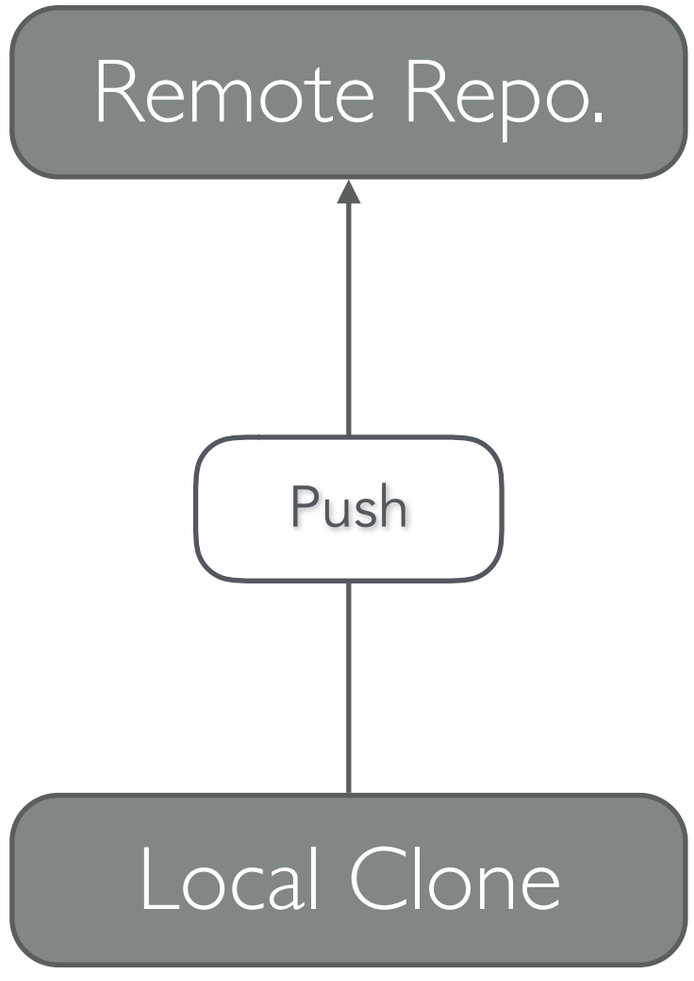
This will set up the directory specified by local_path as a Git working directory that is identical to the cloned repository's last committed state.

You can work in this directory, make changes, stage files and commit them.

UPDATING THE REMOTE REPO

- All the operations discussed so far only affect the **local** clone of the Git repository.
- After committing changes, you must **update the remote** repository that you originally cloned.
- To do this, simply invoke:

```
$ git push
```



ASSIGNMENT

- Here's a basic workflow:
  - Clone your personalized repository:

```
$ git clone https://github.com/claudiascarlata/DSMMA.git
```

  - Create a file and move it to the local repository.
  - Add and commit changes **locally**. It's good practice to add regular commits.

```
$ git add .
$ git commit -m "Completed some of the assignment."
```

  - Make sure you **push** your local changes back to GitHub before the deadline!

```
$ git push
```