Github, Text Editors, and Astropy

ASTR 400B Lecture 3, January 19

Hayden Foote

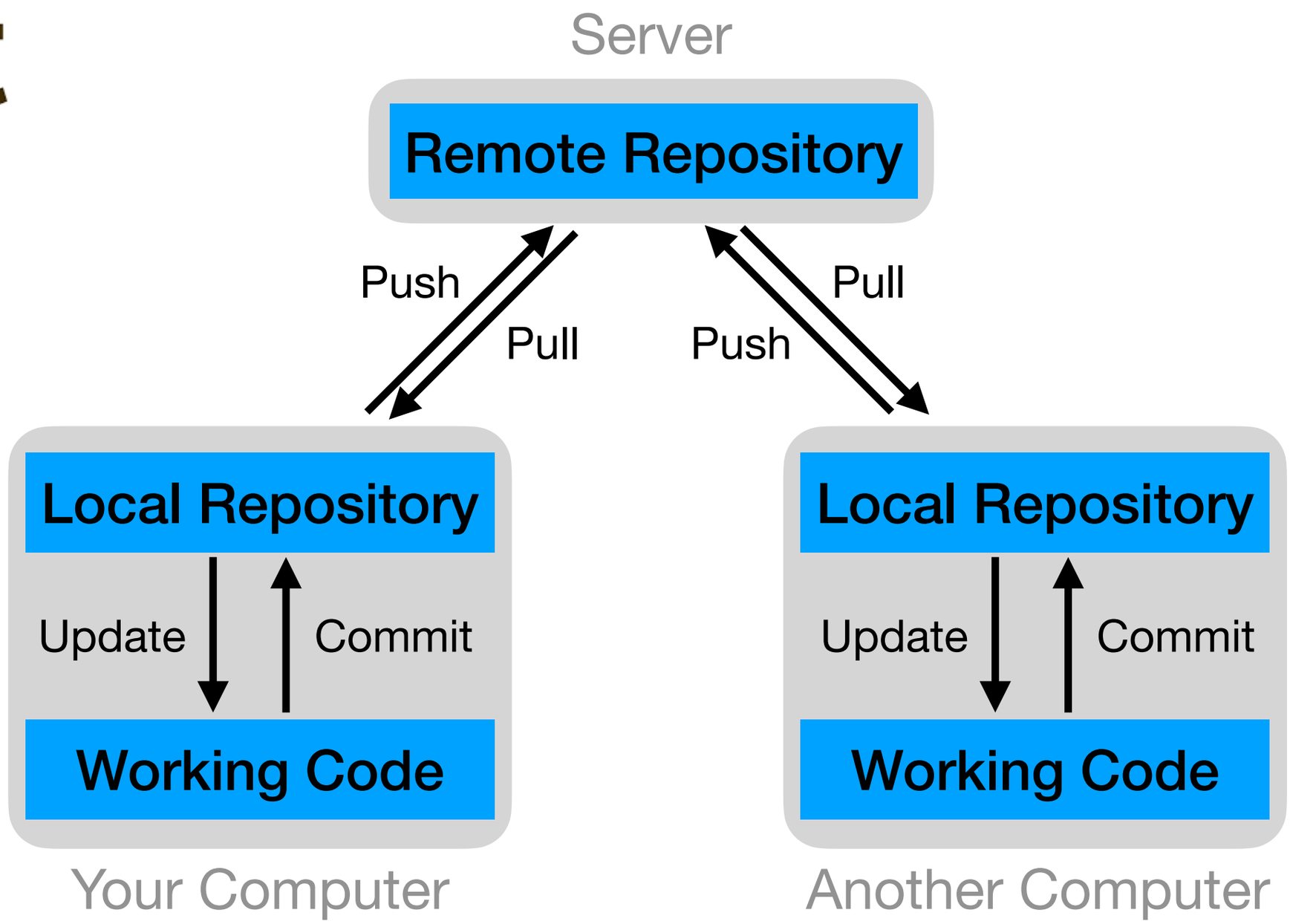
git

GitHub

What is version control?

- In short, version control means keeping track of changes to a code, as well as who made them and when.
- There's software to do this for you! Git is the most commonly-used version control software (VCS) in astronomy, and it's also very common in industry.

Distributed version control, and some lingo



Repository Hosting Services

GitHub

Bitbucket

GitLab

GitHub

- Find, contribute to, and create **open source software**
- Sign up for the **student pack** (https://education.github.com/pack) with your UA email for free private repositories and more!
- **Host webpages** at https://pages.github.com/
- Get **email notifications** about code changes made by collaborators
- **Issue tracking** allows you to set reminders for yourself to fix bugs, submit bug reports on others' code, make to-do lists, and more
- **Document your code** with README files and commit messages

Thanks to Dr. Ekta Patel for this list!

More resources for learning git, GitHub, and other repo hosting sites

- https://git-scm.com/doc Git documentation
- https://docs.github.com/en GitHub documentation
- https://www.atlassian.com/git/tutorials From the makers of bitbucket
- https://www.w3schools.com/git/ W3 schools' free tutorials on git, GitHub, Bitbucket, and GitLab with examples and exercises
- https://teamtreehouse.com/library/introduction-to-git A free video course on git and GitHub
- And More! As always, google is your friend!

Let’s try it!

- If you do NOT have a GitHub account, sign up here https://github.com/signup
- There’s two main ways to interact with repositories:
  - Using the web browser interface (you do not have to have a local copy of the repo), which can be more intuitive, but gives you less control.
  - Using the command line (this takes a bit of work to set up, but is the preferred method, as it gives you more complete tracking and control of changes).

Finding our class repo with a web browser

- Navigate to https://github.com/gurtina/ASTR400B_2023
- This is our class repository managed by Prof. Besla. Here you'll find all of the class content, notes, and assignments. Check it regularly!

Create your homework repository

- To submit your assignments, you'll also need a separate, personal repo for this class that I have access to.
- If you have not yet made your homework repository, navigate to your GitHub homepage, then click the 

button on the left to make a new repository.
- Call it 400B_2023_Yourlastname
- You can make it public or private, but if you make it private, you'll need to give me access so I can find and grade your homework. My GitHub name is **hfoote**.

An aside on text editors

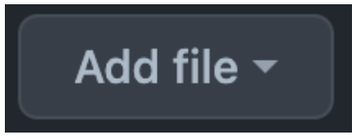
- Text editors allow you to edit files from your command line. Some examples are: vi/vim, emacs, nano, and gedit
- I'm most familiar with emacs, so that's what I'm going to teach you, but if you're already familiar with a different editor and would prefer to use it then please do!
- If you need to install emacs, do so here: https://www.gnu.org/software/emacs/
- In emacs, the command to open a file is

```
emacs file.txt
```

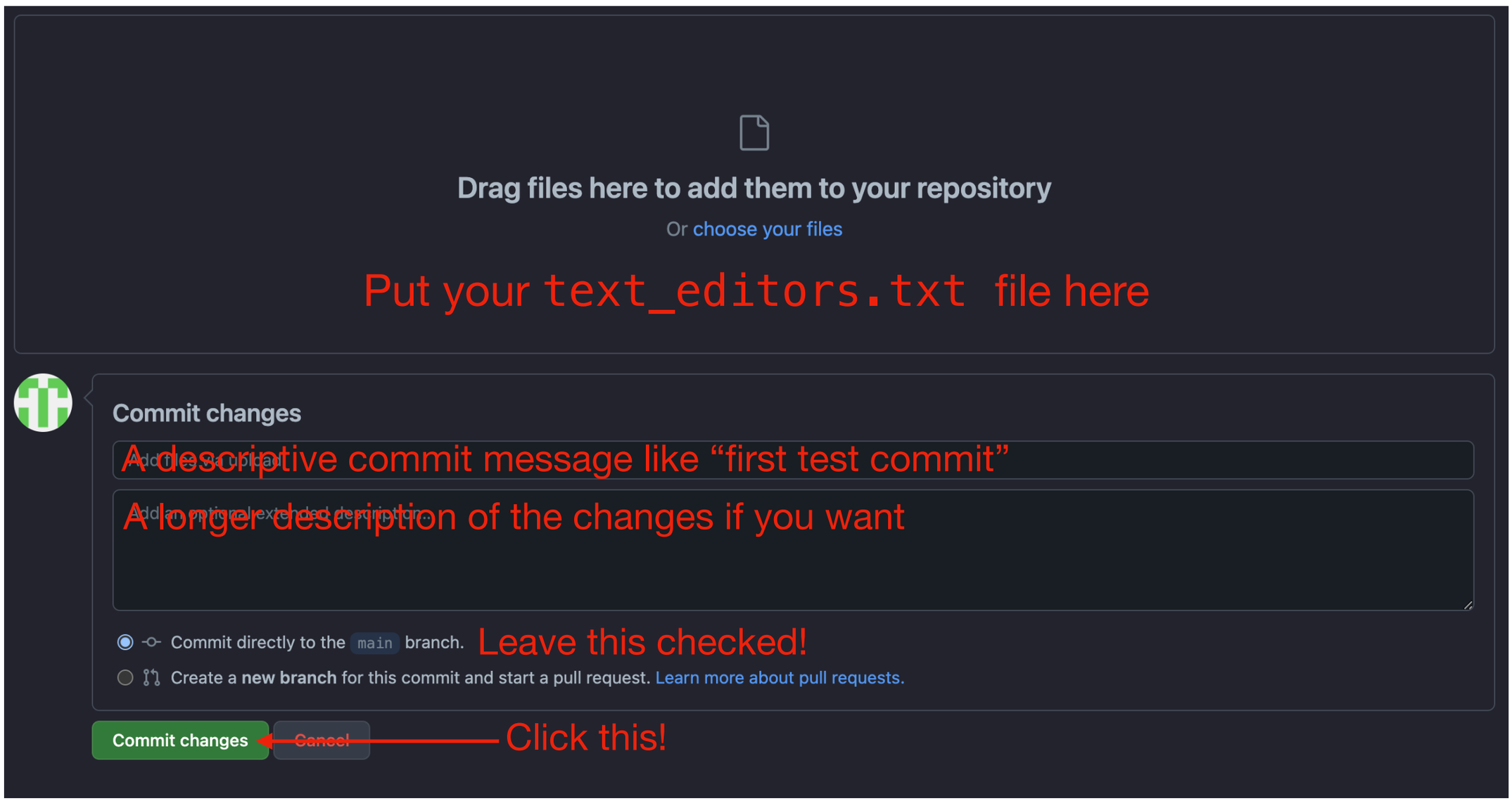
Using emacs

- In emacs, commands are typed by pressing the ESC key or Control (denoted C-) then another character which gives the command.
- A few of the commands I use most often:
  - **Help:** C-h; C-h t launches the tutorial; C-h i launches the manual page.
  - **Save:** C-x C-s
  - **Search:** C-s searches below cursor; C-r searches above cursor; ESC % for find and replace
  - **Exit:** C-x C-c (will ask you if you want to save unsaved changes)
- My favorite quick-reference for emacs commands: https://www.cs.colostate.edu/helpdocs/emacs.html
- **Exercise 1**: With a command line, navigate to the directory you use to store files for this class and use a text editor to create a file called `text_editor.txt`, type the name of the editor you're using into the file, then save it.

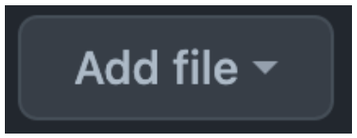
Let’s upload your file to GitHub!

- Go back to your homework repository on GitHub.
- Click the 

menu on the top-right, then select “upload files.” You should see this:



Let’s upload your file to GitHub!

- Go back to your homework repository on GitHub.
- Click the 

menu on the top-right, then select “upload files.”
- You should now see your file in your repository, and be able to click on it to read it.
- You can also edit files and commit the changes from this web browser interface!
- You can submit your homework using this method. However, doing so with the command line offers more control, more flexibility, and is preferred over the web interface, if possible.

Now, let’s try the command line

- If you have not installed git yet, visit https://git-scm.com/downloads and install the version appropriate for your machine.
- Open a command line and type `git`
- You should see something like on the right. If you don’t, let me know!

```
(base) haydenfoote@Haydens-MacBook-Pro ASTR400B_Foote % git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect    Use binary search to find the commit that introduced a bug
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
   show      Show various types of objects
   status    Show the working tree status

grow, mark and tweak your common history
   branch    List, create, or delete branches
   commit    Record changes to the repository
   merge     Join two or more development histories together
   rebase    Reapply commits on top of another base tip
   reset     Reset current HEAD to the specified state
   switch    Switch branches
   tag       Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch     Download objects and refs from another repository
   pull      Fetch from and integrate with another repository or a local branch
   push      Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

Useful git commands

- `git clone repo_url` **clones** a remote repository to your machine using HTTPS
- `git status` tells you the status of your local repo - which files are being tracked, if your repo is ahead of/behind the remote, etc. Use this often, especially before you start working!
- `git mv` (instead of only `mv`) moves files within a repo
- `git pull` **pulls** changes from the remote repository to your local repository
- `git add file(s)` tells git to start tracking changes made to a file or directory
- `git commit –m “commit message”` **commits** your tracked changes to your local repository. ALWAYS add a descriptive (but short) message so you can keep track of changes!
- `git push` **pushes** changes in your local repository to the remote repository on GitHub

Useful git commands (in order!)

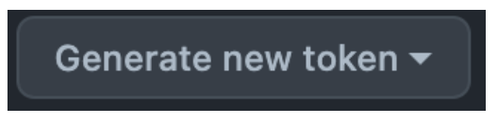
You'll use all of these four commands in this order every time you submit something!

- `git pull` **pulls** changes from the remote repository to your local repository
- `git add file(s)` tells git to start tracking changes made to a file or directory
- `git commit –m "commit message"` **commits** your tracked changes to your local repository. ALWAYS add a descriptive (but short) message so you can keep track of changes!
- `git push` **pushes** changes in your local repository to the remote repository on GitHub

Link your GitHub account to your computer

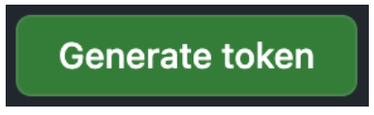
- Run the following commands:
  - `git config --global user.name your_username`
  - `git config --global user.email your@email`
- Each time you clone, push, or pull changes to/from GitHub, you'll need to authenticate your account. GitHub no longer allows you to use your account password for this. There are several ways of doing this, including a personal access token, and Git Credential Manager. You can also set up 2FA.
- Today, we'll set up a personal access token that works like your password.
- The instructions for this are here https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

Creating a personal access token

- From your GitHub page, click the dropdown menu by your profile picture in the top right  and choose “Settings”
- At the bottom of the left-hand sidebar, click “Developer Settings”
- From the left sidebar, click the “Personal access tokens” menu, then select “Tokens (classic)”
- Click 

on the top right, then select “Generate new token (classic)

Creating a personal access token

- Enter your account password if prompted, this will take you to the token creation page.
- Under “Note,” leave yourself a message about what this token is for.
- Under “Expiration” set an expiration date, if desired. Shorter expirations are more secure, but it’s up to you to decide how much security you want.
- Set the permissions, or scope, for the key using the checkboxes. This sets what operations the key will let you perform. For this class, I recommend everything under “repo”, “notifications”, “user”, and “delete_repo.”
- Click 

at the bottom.

Creating a personal access token

- GitHub will generate your token and display it for you. Make sure to copy this token to a *secure* place, I have mine in an encrypted folder that needs a separate password. After you close this page, you won't be able to see it again. If you lose your token, you'll have to generate a new one.

- Now that you have your token, you'll use it in place of your password any time Git prompts you for your account credentials on the command line.

- **IMPORTANT**: This is just one way of authenticating your GitHub account. It should be sufficient for this class since nothing we're doing is very sensitive. However, if you want to use a different or more secure method you're more than welcome to do so. See https://docs.github.com/en/authentication/keeping-your-account-and-data-secure for all of the different methods. Personally, I use Git Credential Manager (https://github.com/GitCredentialManager/git-credential-manager/blob/main/README.md), which stores my account info and tokens in a secure file on my computer so I don't have to type in a token every time I push something.

Cloning the class repository

- In a browser, visit https://github.com/gurtina/ASTR400B_2023
- Click the gree  button on the right, and copy the link.

- Using a command line, navigate to the directory where you want to store the files for this class.
- Type `git clone https://github.com/gurtina/ASTR400B_2023.git` (the link is what you copied from GitHub)
- If you see something like this:

```
(base) haydenfoote@Haydens-MacBook-Pro ASTR400B % git clone https://github.com/gurtina/ASTR400B_2023.git
Cloning into 'ASTR400B_2023'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 16 (delta 0), reused 7 (delta 0), pack-reused 0
Receiving objects: 100% (16/16), 218.85 KiB | 858.00 KiB/s, done.
(base) haydenfoote@Haydens-MacBook-Pro ASTR400B %
```

You now have a copy of our class repository on your computer! Use `git pull` regularly to update it as new content and assignments are added.

Exercise: put it all together

- Use `git clone` to clone your homework repository from GitHub onto your computer.
- Using `mkdir`, make a directory in your homework repo called Lecture3
- In your Lecture3 directory, use your favorite text editor to make a file called `test_commit.txt`
- In this file, type the name of your favorite astronomical object, then save it.
- Move the `text_editor.txt` file we created earlier into the Lecture3 directory.
- Use `git add` to start tracking changes to both of the text files.
- Use `git commit` to commit your new files to your local repo
- Use `git push` to push your changes to your remote repo on GitHub
- Go to your repository on GitHub, you should see the Lecture3 directory with both `test_commit.txt` and `text_editor.txt`. You might have to refresh the page.
- **This is the procedure you will follow to submit your homework!** If you're ever having trouble with this, I encourage you to come to my office hours for help. You can also upload your homework using the web interface in a pinch.

jupyter

Jupyter Notebooks and Astropy

- Jupyter is a browser-based application that creates notebooks for your code.
- If you installed anaconda, you already have Jupyter!
- To launch jupyter:
  - Use the anaconda app, OR
  - Type `jupyter notebook &` in a command line (the & allows you to keep using the terminal window while the notebook is running)
- Navigate to our class repository, and open the astropy_tutorial.ipynb notebook in the Lecture 3 folder.