Week 8: Version Control with Git and GitHub

Learning Objectives

This module introduces the fundamental principles of version control, focusing on Git and GitHub. It aims to equip students with the skills needed to manage coding projects, track changes in their work, and collaborate with others in a distributed environment. By the end of this module, you should be able to complete the following tasks:

- create and manage a Git repository
- Use basic Git commands (clone, status, add, commit, push, pull)

Introduction to version control

Programming is a constantly evolving process where you make changes to your code to improve the function or even add entirely new ideas. Because of this, it becomes important to save versions of your code so that you always have a working version of the code to fall back on if you make a mistake in your current version. This is the practice of **version control**, saving copies of your work in an organized fashion so you can refer to it later. Version control systems are softwares that can help you version control with less hassle. An example of a version control system is **Git**. You may have even heard of **GitHub**, which is a web-based platform that uses Git to version control and host code for collaborative projects.

Git and GitHub setup

As mentioned above, GitHub is a platform that allows programmers to publicly share their code and work on projects collaboratively. Projects in GitHub are organized as **repositories**, where a repository is designed to store all the code and data contents of a project into an organized space.

But before any sharing can be done, first follow the steps below to set up your computers with git and create a new repository to test with.

Step 1: Install Git

1. On your computer, open the *Anaconda Prompt* (Windows) or *Terminal* (macOS). Please note that all steps provided in this module should work for both operating systems and CLIs. Screenshots provided throughout the module are based using macOS but should look fairly similar to what is expected on windows.

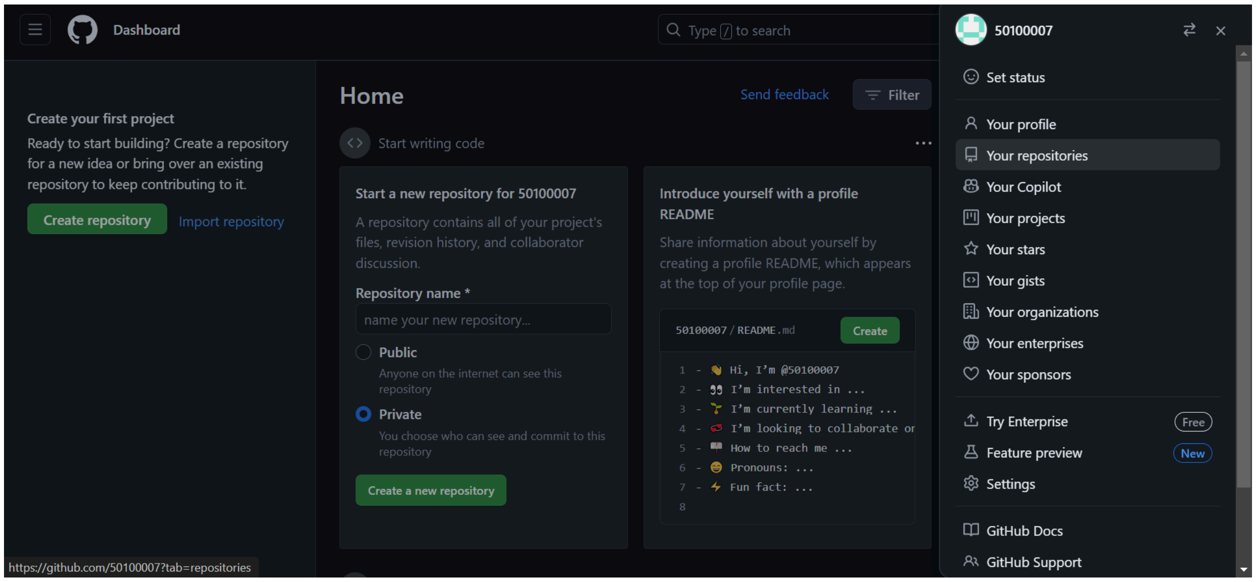
2. Type `conda install git`, then press enter. This will install Git on your computer. Halfway through the process of creating an environment, you may be prompted on whether you want to install other required packages. Type `y`, then press enter.

3. Type `git --help`, then press enter. This is to check if Git was successfully installed on your computer. You should get a list of functions you can use through Git.

Step 2: Create a Github repository

1. Follow the instructions provided through the link to create a free GitHub account (https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home). After signing up, log into your account.

2. Go to your profile page by clicking the profile picture icon in the top right and open the *Repositories* section.



Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

Start a new repository for 50100007

A repository contains all of your project's files, revision history, and collaborator discussion.

Repository name *

name your new repository...

Public — Anyone on the internet can see this repository

[✓ Private] — You choose who can see and commit to this repository

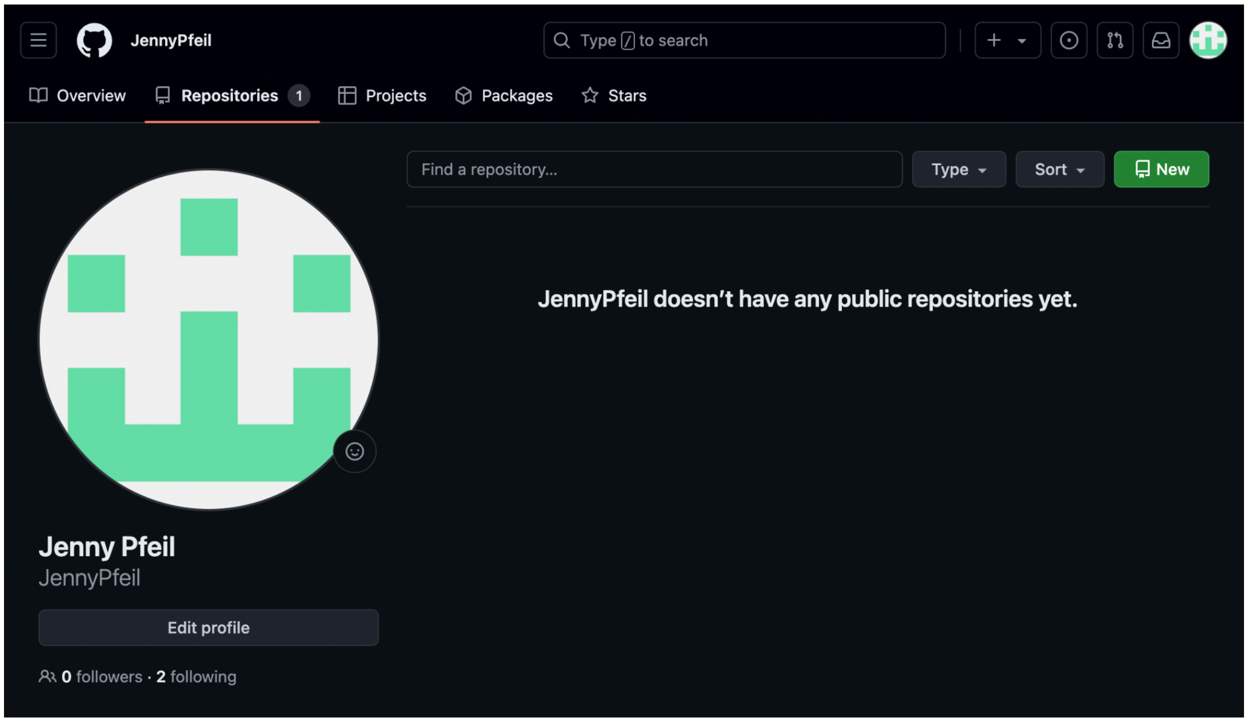
Introduce yourself with a profile README

Share information about yourself by creating a profile README, which appears at the top of your profile page.

50100007 / README.md

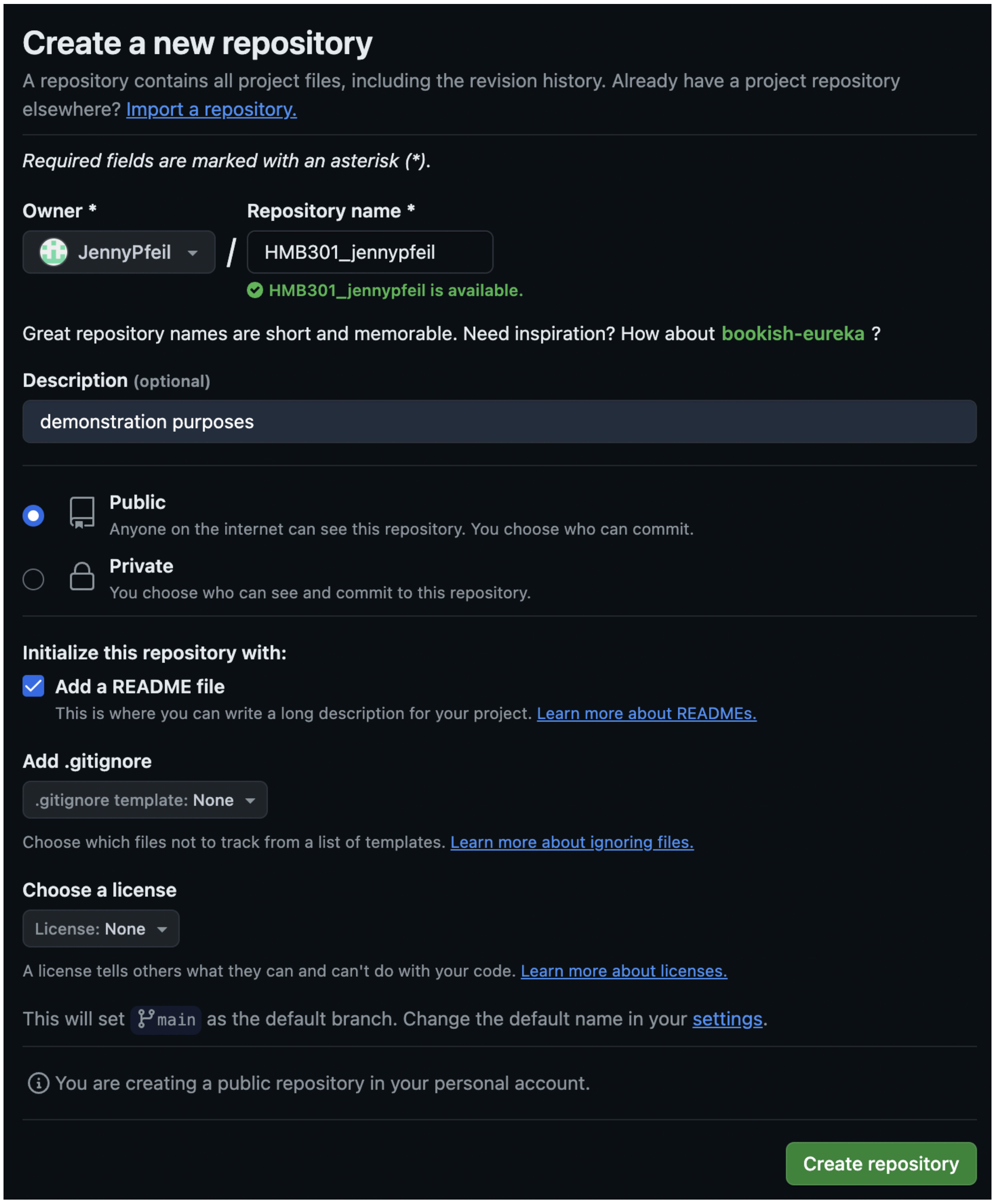
```
- 👋 Hi, I'm @50100007
- 👀 I'm interested in ...
- 🌱 I'm currently learning ...
- 💞️ I'm looking to collaborate on
- 📫 How to reach me ...
- 😄 Pronouns: ...
- ⚡ Fun fact: ...
```

3. Click New to create a new repository



JennyPfeil doesn't have any public repositories yet.

4. Configure your new repository based on the settings shown to you in the image below, then press **Create repository**. Note that the name of the repository should include your username or be unique to you in some way, otherwise the name may not be available. The configuration settings we have provided here are recommended for the completion of this module, however for your own work outside of this class you may want to adjust these settings based on the project.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * JennyPfeil / **Repository name *** HMB301_jennypfeil

HMB301_jennypfeil is available.

Great repository names are short and memorable. Need inspiration? How about **bookish-eureka** ?

Description (optional)

demonstration purposes

● **Public**
Anyone on the internet can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☑ **Add a README file**
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

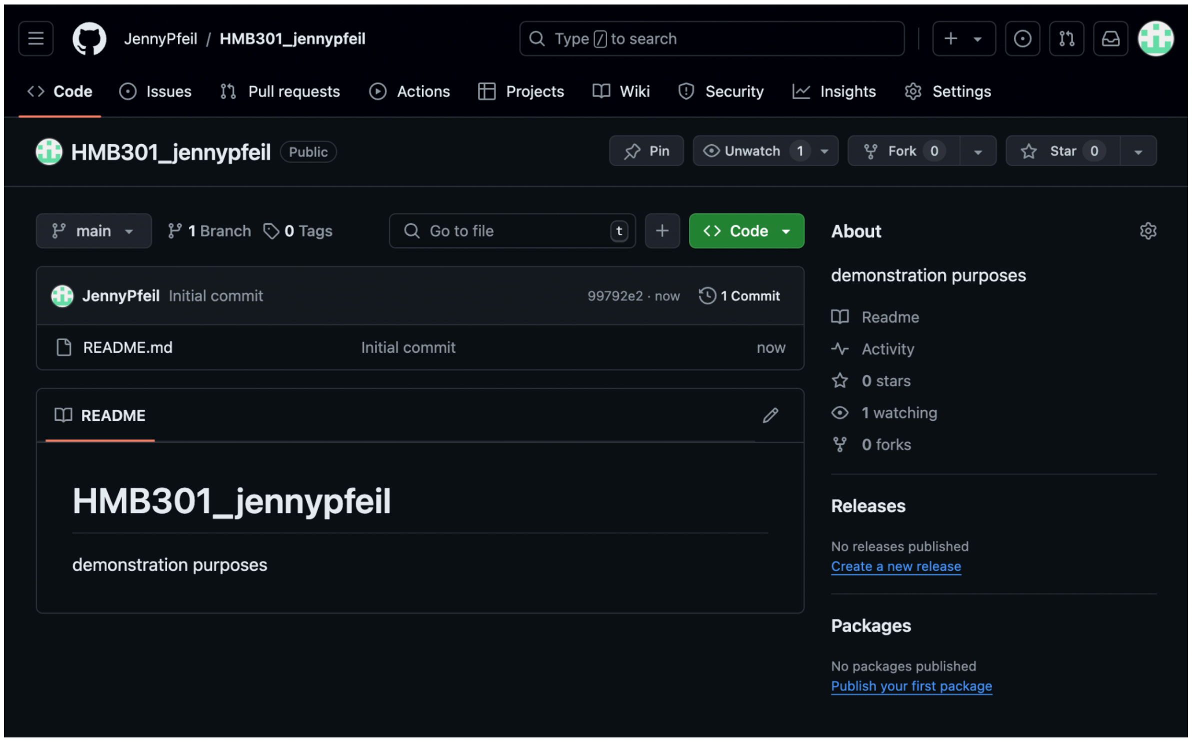
License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

This will set `main` as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

5. A new repository should now appear on your account.



JennyPfeil Initial commit 99792e2 · now 1 Commit

README.md	Initial commit	now

README

HMB301_jennypfeil

demonstration purposes

About

demonstration purposes

Readme
Activity
0 stars
1 watching
0 forks

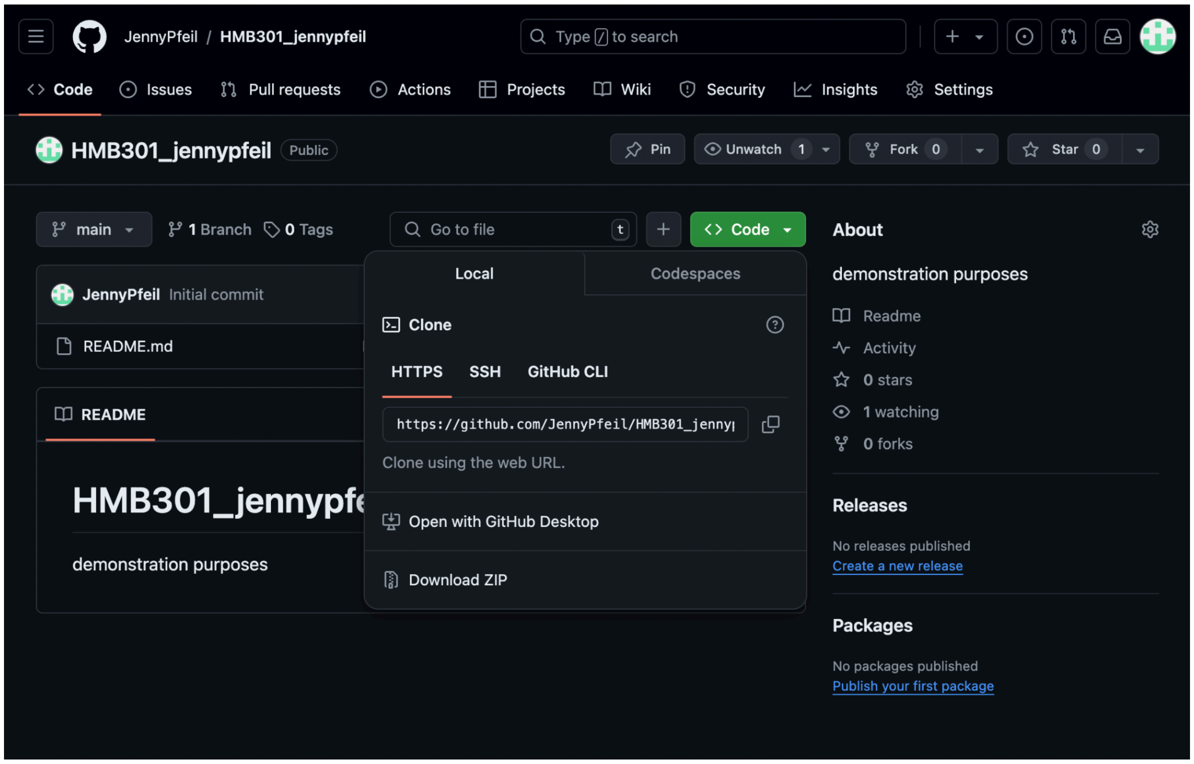
Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

6. Press <> Code ▾. This should open up a window where you can copy a URL. Save this URL for later.



JennyPfeil Initial commit

README.md

README

HMB301_jennypfe

demonstration purposes

```
https://github.com/JennyPfeil/HMB301_jenny
```

Clone using the web URL.

About

demonstration purposes

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Manage repositories with Git

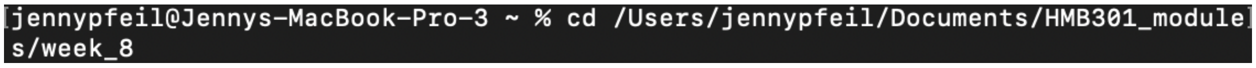
Git provides a variety of functions for users to manage change and track changes of their projects, as well as others' projects. One common function is to share changes from your local computer with your publicly accessible repository, called **pushing**. Another common function is to save changes to your local computer from publicly accessible repositories, called **pulling**. There are also many other functions of Git, such as creating multiple versions of a project simultaneously, which will not be covered in the context of this module but can be found at https://git-scm.com/doc if you are interested. Git can be used directly through a CLI or through added extensions in VSCode. For this module we will be using Git through a CLI.

Follow the steps below to make changes to your newly created repository and save changes from others' repositories.

Step 1: Clone a repository

1. On your computer, open *Anaconda Prompt* (Windows) or *Terminal* (macOS).

2. Navigate into the week 8 folder you created in the previous module. Type `cd Documents/HMB301_modules/week_8` , then press enter.



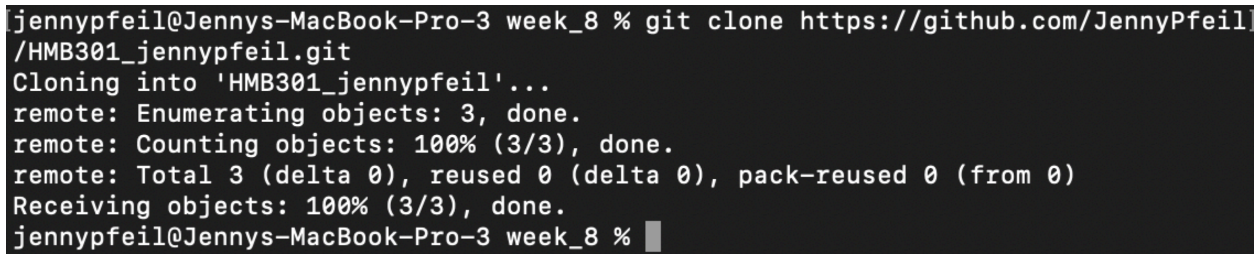
```
[jennypfeil@Jennys-MacBook-Pro-3 ~ % cd /Users/jennypfeil/Documents/HMB301_module]
s/week_8
```

3. If this is your first time using git commands through your CLI on your computer, you may need to perform a couple set up steps.

For Windows first type `git init config -- global <username>`, then press enter. Then type `git init config -- global <email used to set up your github account>`, then press enter.

For macOS first type `git config -- global <username>`, then press enter. Then type `git config -- global <email used to set up your github account>`, then press enter.

4. Using the URL that you saved previously, type `git clone <URL>` then press enter. You should see something like this if your repository was successfully cloned



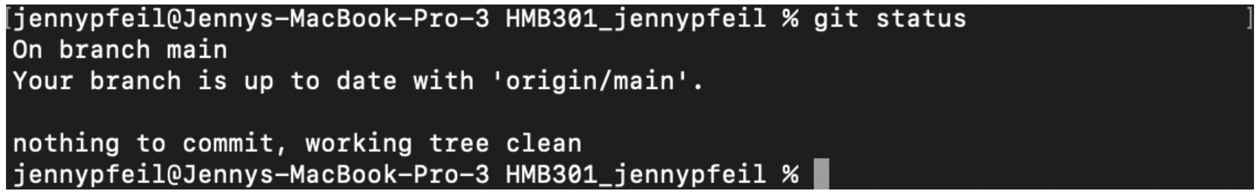
```
[jennypfeil@Jennys-MacBook-Pro-3 week_8 % git clone https://github.com/JennyPfeil]
/HMB301_jennypfeil.git
Cloning into 'HMB301_jennypfeil'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
jennypfeil@Jennys-MacBook-Pro-3 week_8 %
```

You should also now be able to check that the repository has been correctly cloned by navigating through the *Finders* application to find new directories added to the week_8 directory.

Step 2: Push changes to a repository

1. Navigate into the cloned repository, Type `cd <repository name>` , then press enter.

2. To first check the status of your repository before we make any changes. Type `git status`, then press enter.



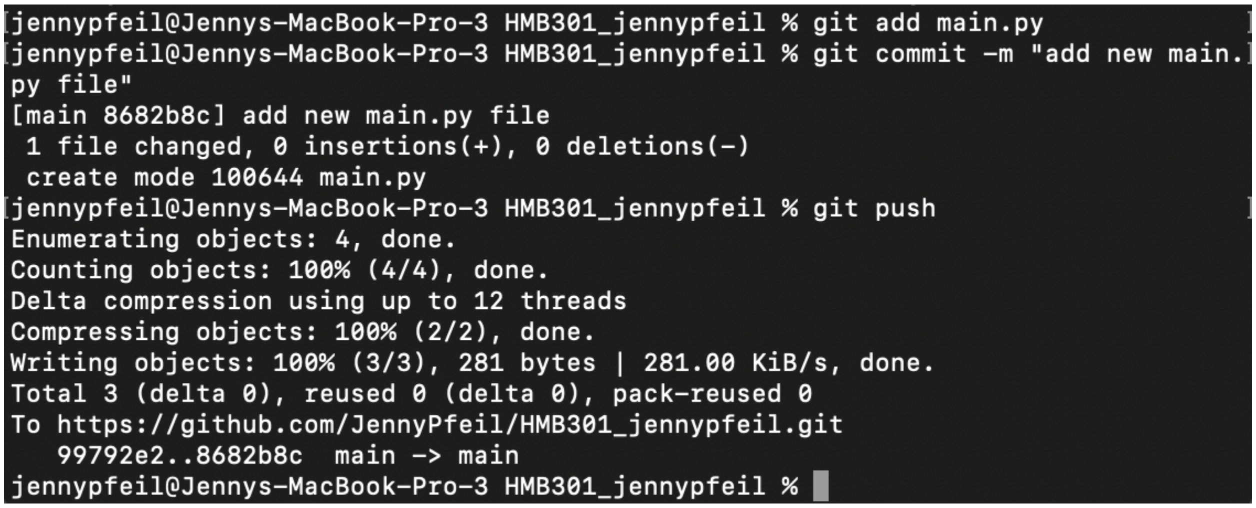
```
[jennypfeil@Jennys-MacBook-Pro-3 HMB301_jennypfeil % git status                    ]
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
jennypfeil@Jennys-MacBook-Pro-3 HMB301_jennypfeil %
```

3. Now we will add a new file to the repository on our computers locally, type `touch main.py`, then press enter. If you check the status of your repository again. We should see that there are now changes that need to be committed to the repository.

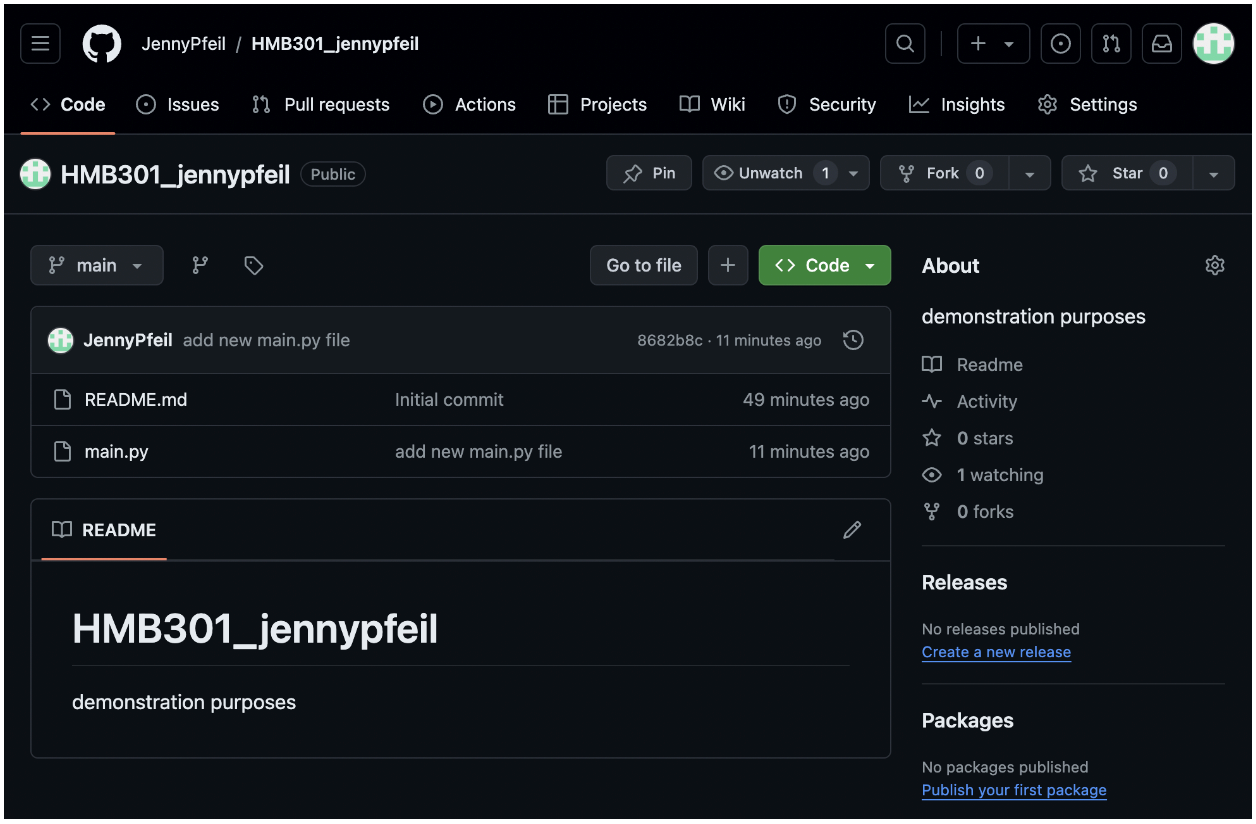
4. There are a series of steps that need to be completed in order to make local changes to your repository appear on the GitHub repository for the public, also called pushing. First type `git add main.py`, then press enter. Now type `git commit -m "add new main.py file"`. The -m option allows you to add a message to the commit so that you can later refer to what you changed in your repository. Lastly type, `git push`.

You should see something like this if changes to your repository were successfully committed and pushed



```
jennypfeil@Jennys-MacBook-Pro-3 HMB301_jennypfeil % git add main.py
jennypfeil@Jennys-MacBook-Pro-3 HMB301_jennypfeil % git commit -m "add new main.
py file"
[main 8682b8c] add new main.py file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 main.py
jennypfeil@Jennys-MacBook-Pro-3 HMB301_jennypfeil % git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 281 bytes | 281.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/JennyPfeil/HMB301_jennypfeil.git
   99792e2..8682b8c  main -> main
jennypfeil@Jennys-MacBook-Pro-3 HMB301_jennypfeil %
```

You should also now be able to check on your browser that the repository has changed for the public.



JennyPfeil add new main.py file 8682b8c · 11 minutes ago

README.md	Initial commit	49 minutes ago
main.py	add new main.py file	11 minutes ago

README

HMB301_jennypfeil

demonstration purposes

About

demonstration purposes

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Step 3: Pull changes to a repository (The following steps require you to have a partner. Please make your own partners within your tutorial or you can ask your TA for assistance.)

1. On your computer, open *Command Prompt* (Windows) or *Terminal* (macOS)

2. Navigate into the week 8 folder you created in the previous module.

3. Attain the URL for your partner's repository, similar to how you found the URL for your own repository. You can do this either by sharing the URL through an external messaging system or by searching GitHub for the name of their repository. Using this URL from your partner, clone their repository to your week_8 directory.

4. Now navigate back to your own cloned repository.

5. Both you and your partner should now add a new file to the repository on your computers locally, naming it `extra.py`.

6. Push this change to your own repository similar to as you have previously done. Double check the change is also reflected in GitHub on your browser.

7. Now navigate back to your partner's cloned repository.

8. Type `git pull`, then press enter. You should be able to see a new file be added to your local version of your partner's directory. Remember that it is always a good practice to pull the latest changes from the remote repository you are working on before making any of your own changes to it, otherwise conflict between versions can occur.

Now you have successfully performed some of the most common tasks in managing a repository.

Conclusion

This brings us to the end of the content for module week 8. This week, we learned about version control with Github. Specifically, we learned about setting up a new repository and how to push and pull from public repositories. Although the process of version control can be tedious at first, especially when you are working on projects on your own. The benefits of version control will become more obvious when you work on projects collaboratively, as you use the skills you have learnt throughout all these modules in a setting where multiple individuals are working on different but dependent aspects of a project together.

Graded exercises

Take a screenshot of your newly created github repository and upload the image to Quercus for grading.