GitHub

SURP 2023 Python Bootcamp

Ohio State Astronomy

Slides by: James W. Johnson

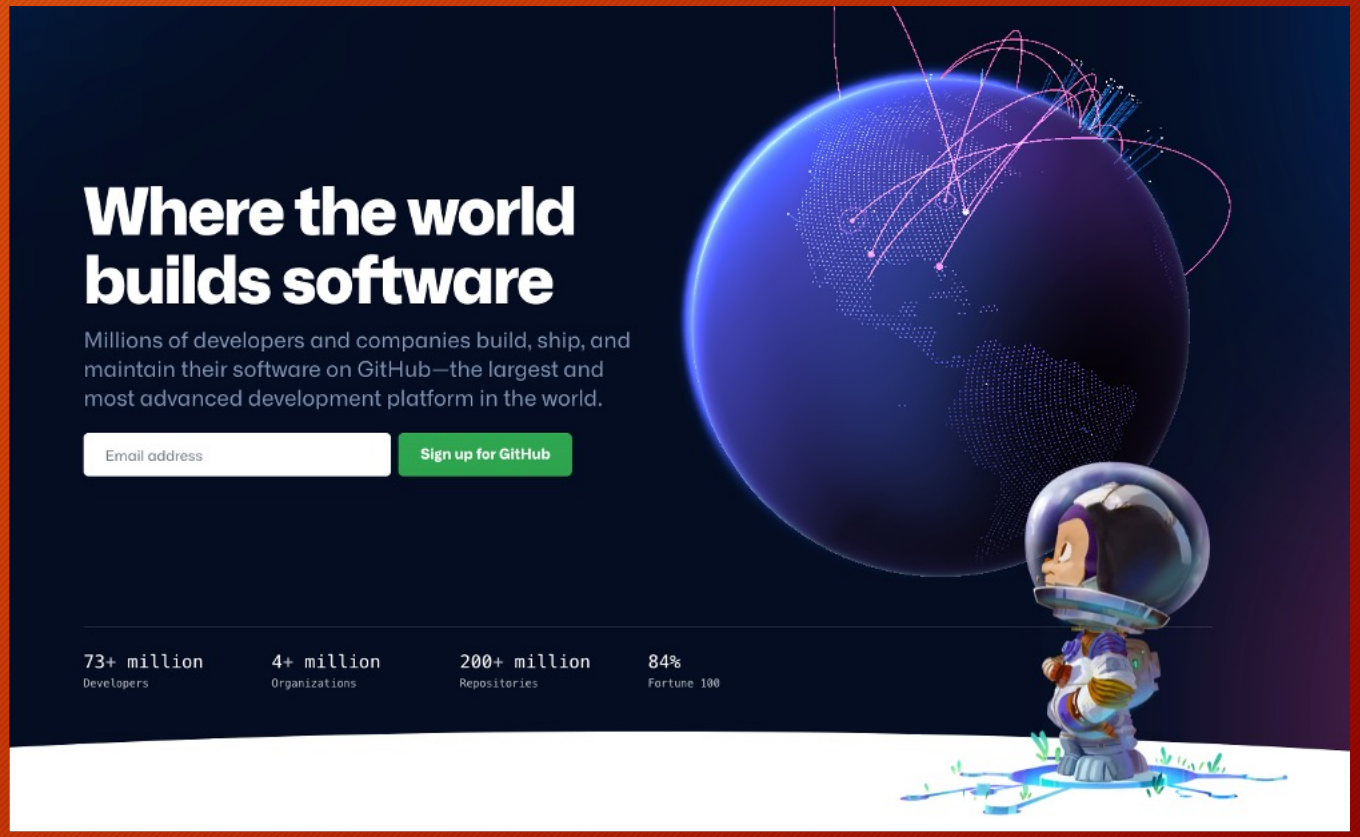
What is GitHub?

A website for hosting files in the cloud

More fundamentally: version control

- Tracks changes to your files over time
- Can host multiple versions of the same file(s) under the same name(s)

Free to use, with some features enabled with a premium account



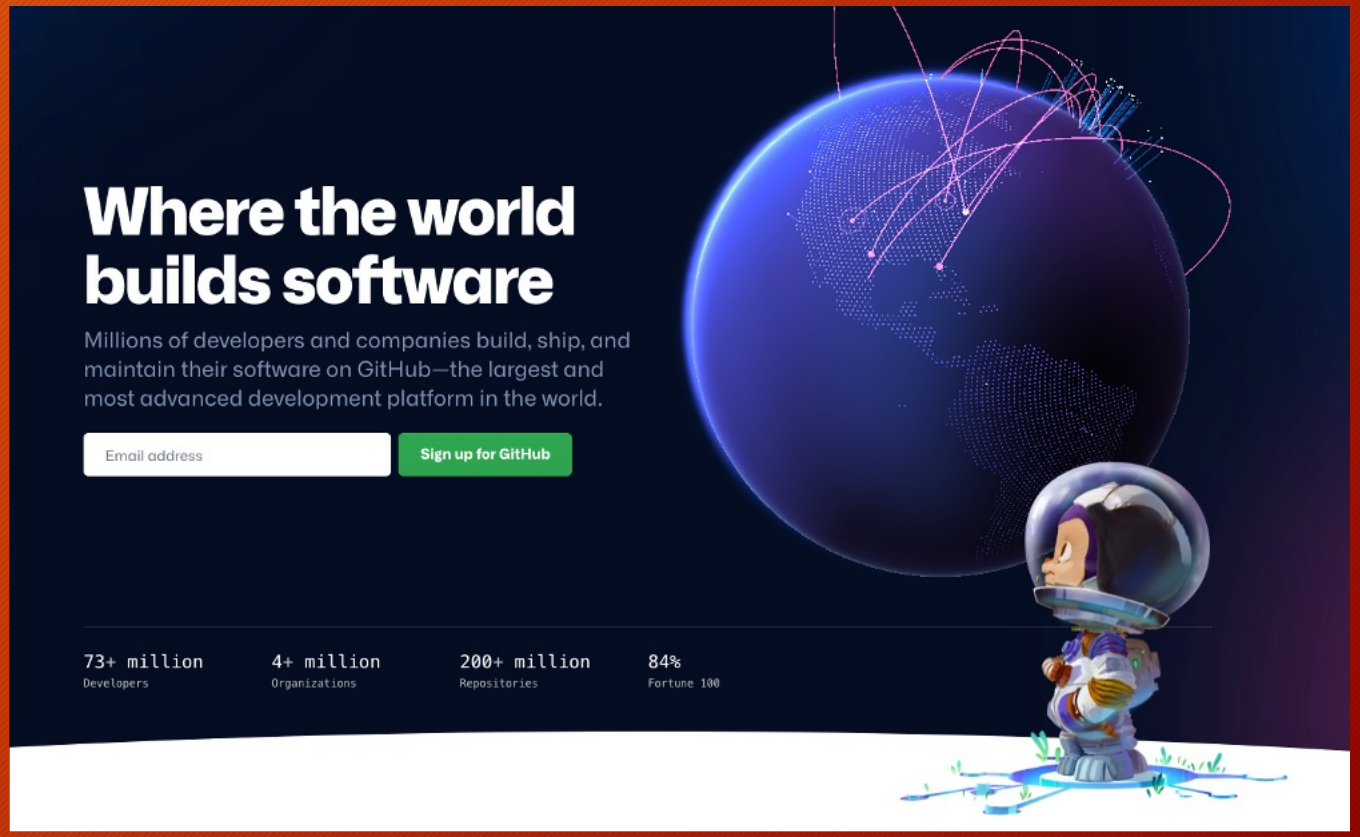
What can you do with GitHub?

For researchers:

- Share files with collaborators
- Manage copies between multiple computers
- Backup copy of your work

For software developers:

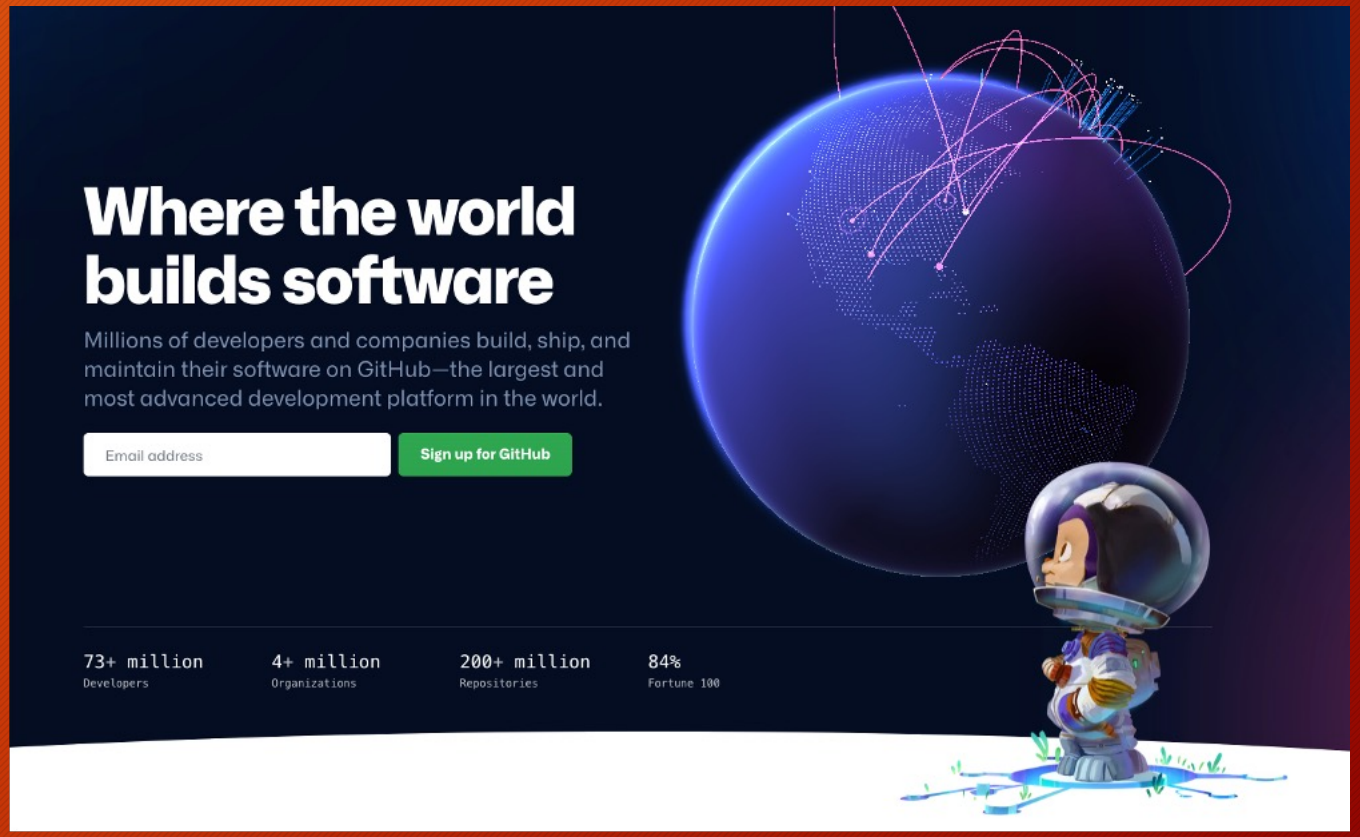
- Track changes to code over time
- Collaborate with other developers seamlessly



Why learn it now?

A majority of astronomers and physicists eventually leave academia for data science or other private sector jobs

In these positions, knowledge of version control tools like GitHub is *expected*.



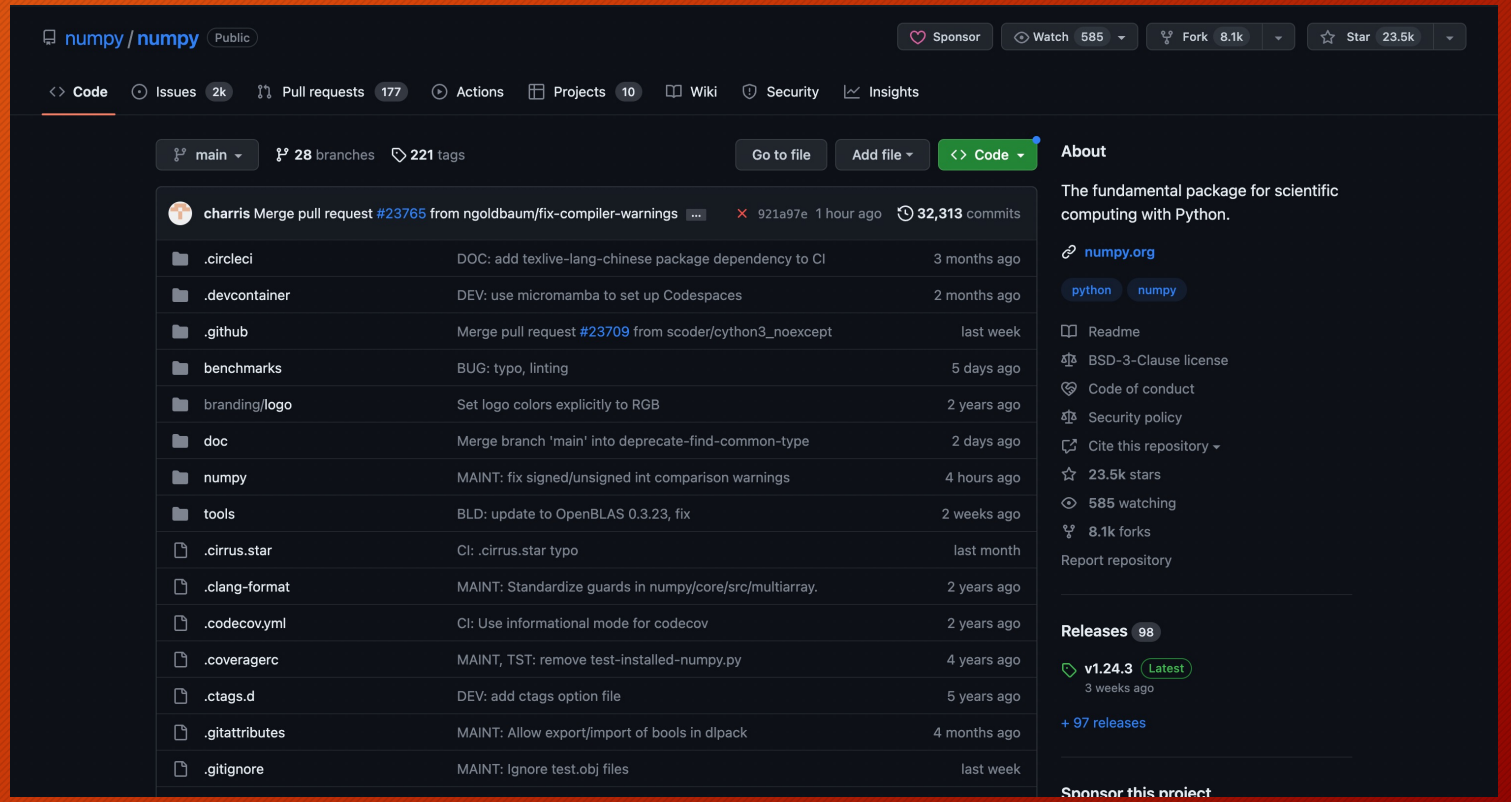
Repositories

A directory containing files and other directories that is stored on GitHub

Example: NumPy, SciPy, and matplotlib each host their source code on GitHub

Files can be anything

- Code (including jupyter notebooks)
- Documents
- Images

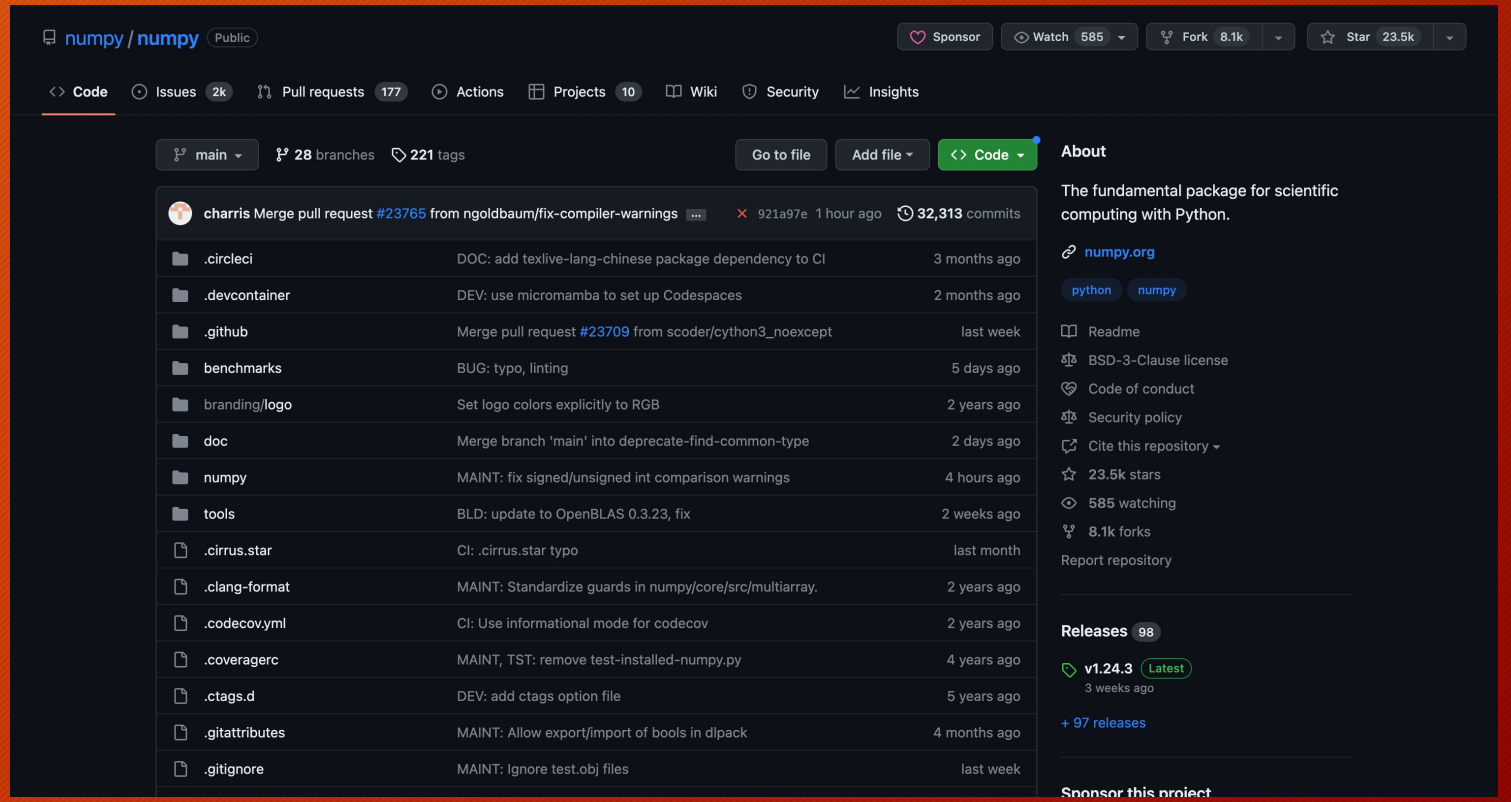


Repositories: Local vs. Remote

Local: this file system *on your computer*

Remote: this file system *on GitHub's website*

Remote will always have the url format *github.com/<user>/<repo>*



Objectives

- Commits: how to make changes to a remote repository
- *git*: terminal application for interacting with a repository on GitHub
- Branching: how to store the same file(s) in different states
- GitHub websites
- Setting up a repository: GitHub itself walks you through this

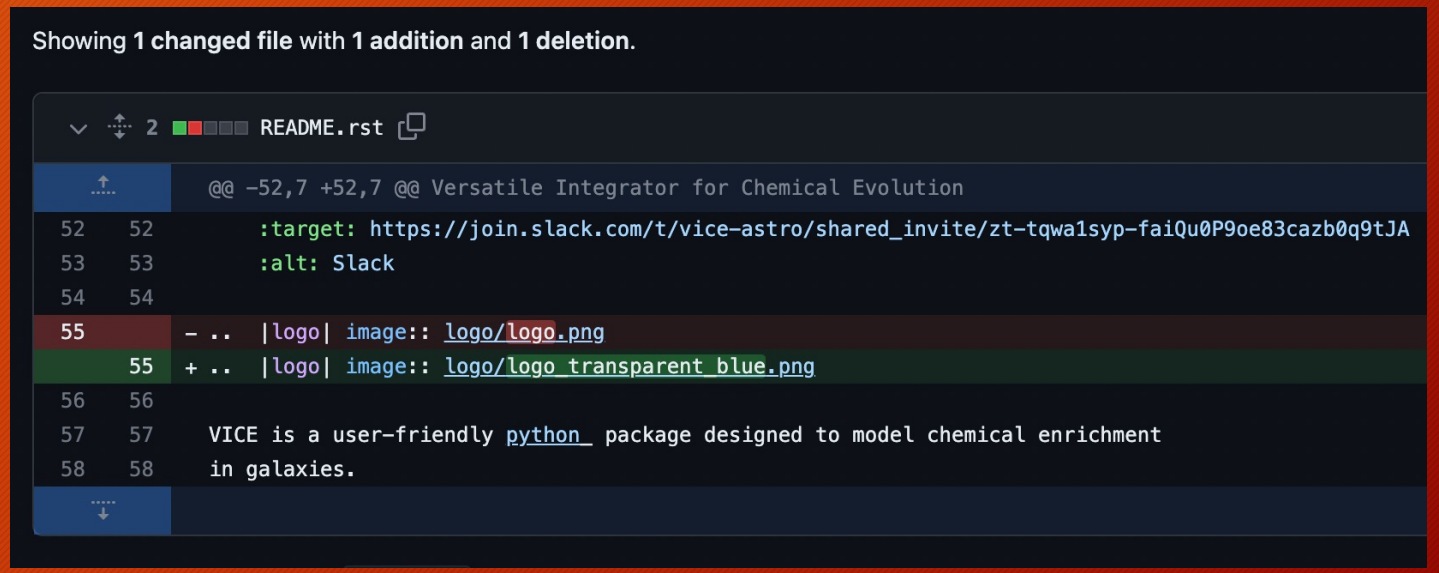
Commits

Bread and butter for making changes to a repository

A series of +’s and –’s within files

- Current state of repository = sum of all commits across all files

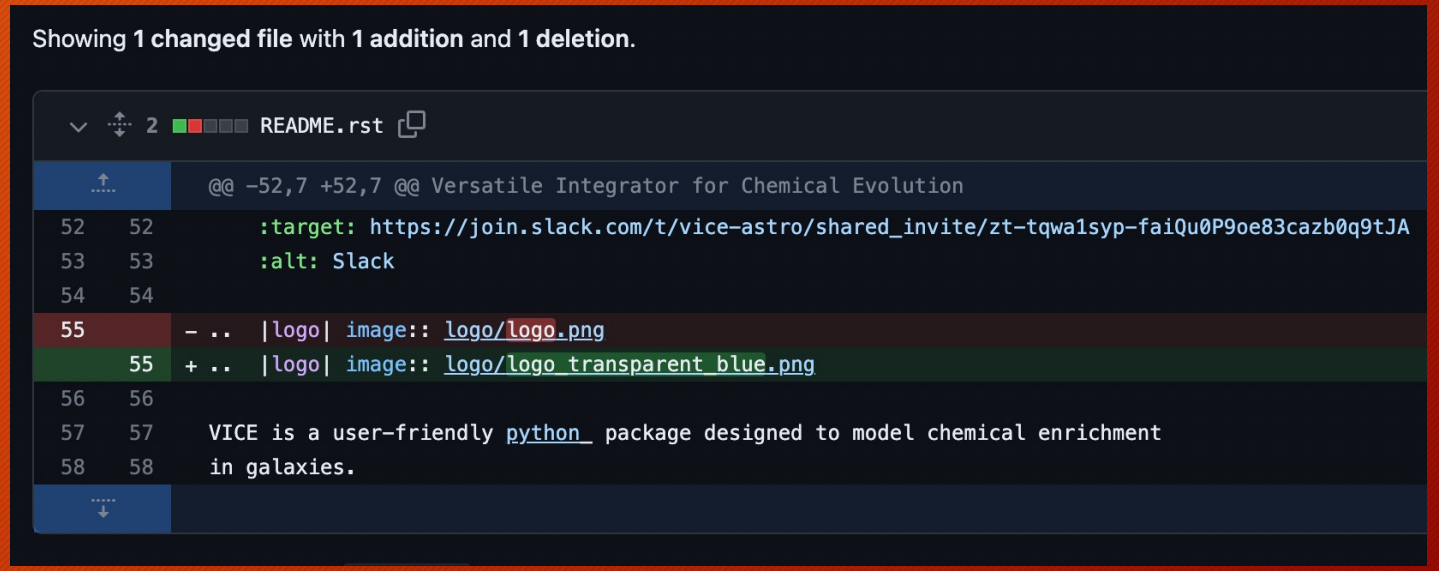
Tracks bit-by-bit changes to binary files (e.g., PDFs)



Commits

That’s about all there is to it!

All of this happens *after* you’re ready to upload changes to your files



git: Managing Repositories

Terminal application for managing a repository, including making commits

- Generally pre-installed
- Uploading: *git add*, *git commit*, *git push*
- Downloading: *git pull*

git --help prints an API reference in the terminal

No connection to GitHub as a website : *git* is an open-source version control program

- Tracks changes in any set of computer files for projects of any size
- Other websites built around *git*: GitLab, Bitbucket

git mv and *git rm*

Both fulfill the same functionality as *mv* and *rm* otherwise do

git mv: rename the file

- A bare *mv* without *git* will make the new version show up as an untracked file
- The file takes its commit history with it
- If you mess up, *mv* followed by *git mv* is fine

git rm: delete the file

- Change will be reflected when others run *git pull*

git status: Which files have updates?

Displays the relative paths to files that

- are new and not yet stored within the repository (i.e., “untracked” files)
- are already stored within the repository, but have local updates not yet added to the next commit
- have local updates that have been added to the next commit

```
(python3.10) jamesjohnson@Binary slides % git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   github.pptx

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ~$github.pptx

no changes added to commit (use "git add" and/or "git commit -a")
(python3.10) jamesjohnson@Binary slides % _
```

The *.gitignore* file

git looks for a plain-text file called *.gitignore* in the top-level directory of all repositories

- Contains paths and glob patterns for files that it should skip

```
1
2  *.so
3  *.pyc
4  *.pdf
5  *.sublime-workspace
6  *.sublime-project
7
```

git add and *git restore*

git add: tells *git* which file(s) to include in the next commit

- Can be as many files as you want/need
- Doesn't have to be *every* file with changes

git restore [*--staged*]: the "undo" button

```
(python3.10) jamesjohnson@Binary slides % git add github.pptx
(python3.10) jamesjohnson@Binary slides % git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   github.pptx

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        images/github.gitignore.png
        images/github.gitstatus.png
        ~$github.pptx
```

git commit

Creates the commit itself

Command will open the *vi* text-editor and prompt you for a message to attach to the commit

- Press *A* to enter edit mode
- Press *Esc* to enter command mode, then ":wq" and *Enter* will save and quit

git commit –m "message here" takes the message in quotes without opening *vi*

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Your branch is up to date with 'origin/main'.
#
# Changes to be committed:
#       modified:   github.pptx
#
# Changes not staged for commit:
#       modified:   github.pptx
#
# Untracked files:
#       images/github.gitadd.png
#       images/github.gitignore.png
#       images/github.gitstatus.png
#       ~$github.pptx
#
~
~
~
~
~
~
~
~
```

git push and *git pull*

git push: uploads the new commit to the remote repository on GitHub

- The first time you do this, it will likely prompt you to run with extra flags to tell it the branch you're pushing to
  git push --set-upstream origin main

git pull: downloads new commits from the remote repository

Display remote URL: *git remote –v*

Set remote URL: *git remote set-url origin* [*url*]

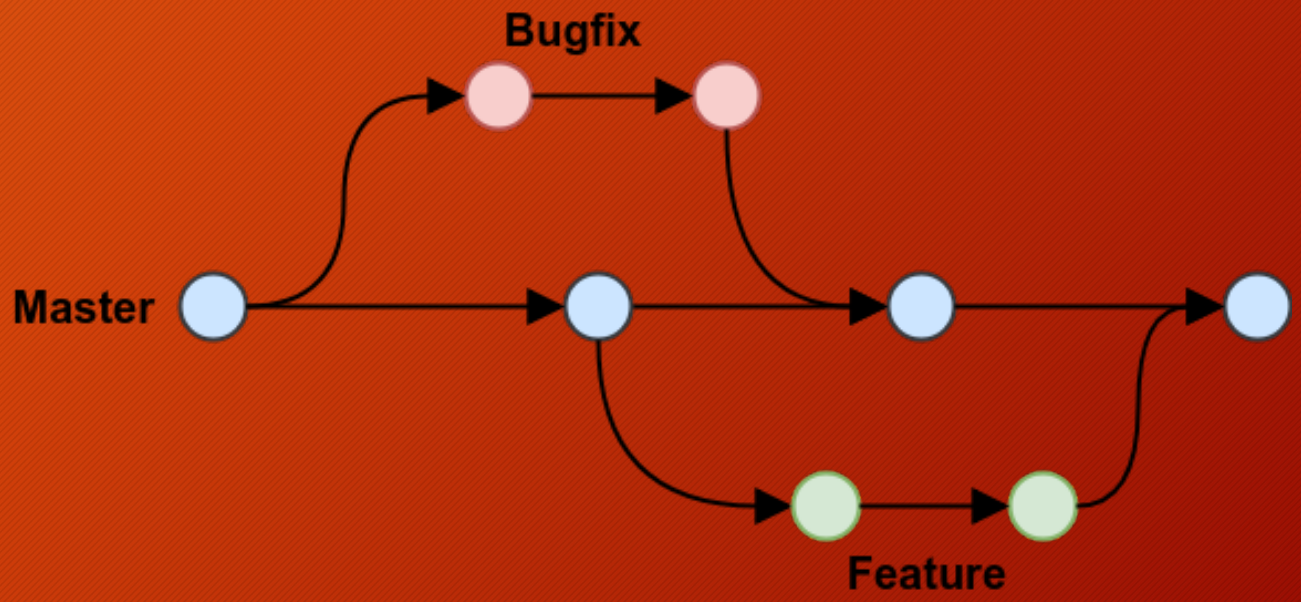
git gud: for when coding is as hard as Dark Souls

Branches

Different branches track different commit histories

- Current state of file(s) = sum of commits *on a given branch*

Of particular importance if you are developing software that will be distributed/released



Note: default name of the “primary” branch x to be *master* but is now *main*

git branch: Displays which branch you're on

Branching in production-level code:

- Development
- Different features on their own branches
- Recently released versions
- Main branch usually latest release

Branching for researchers:

- Can be used to store the same files in different states
- More useful if your code base gets large

```
(python3.10) jamesjohnson@Binary VICE % git branch
* development
  development-openmp
  development-openmp-modeling
  development-openmp-windowscompat
  main
  v1.1.x
  v1.2.x
  v1.3.x
  v1.3.x-patch
```

git switch

Changes between branches locally

Can also be done with *git checkout*

- *checkout* can also be used to copy file(s) from another branch

Will prompt you to commit your changes if it detects updates in tracked files.

```
(python3.10) jamesjohnson@Binary VICE % git branch
* development
  development-openmp
  development-openmp-modeling
  development-openmp-windowscompat
  main
  v1.1.x
  v1.2.x
  v1.3.x
  v1.3.x-patch
(python3.10) jamesjohnson@Binary VICE % git switch main --quiet
(python3.10) jamesjohnson@Binary VICE % git branch
  development
  development-openmp
  development-openmp-modeling
  development-openmp-windowscompat
* main
  v1.1.x
  v1.2.x
  v1.3.x
  v1.3.x-patch
(python3.10) jamesjohnson@Binary VICE % git checkout development --quiet
(python3.10) jamesjohnson@Binary VICE % git branch
* development
  development-openmp
  development-openmp-modeling
  development-openmp-windowscompat
  main
  v1.1.x
  v1.2.x
  v1.3.x
  v1.3.x-patch
```

git branch: A new branch

git branch <new_branch> will create a new branch with whatever name you give it

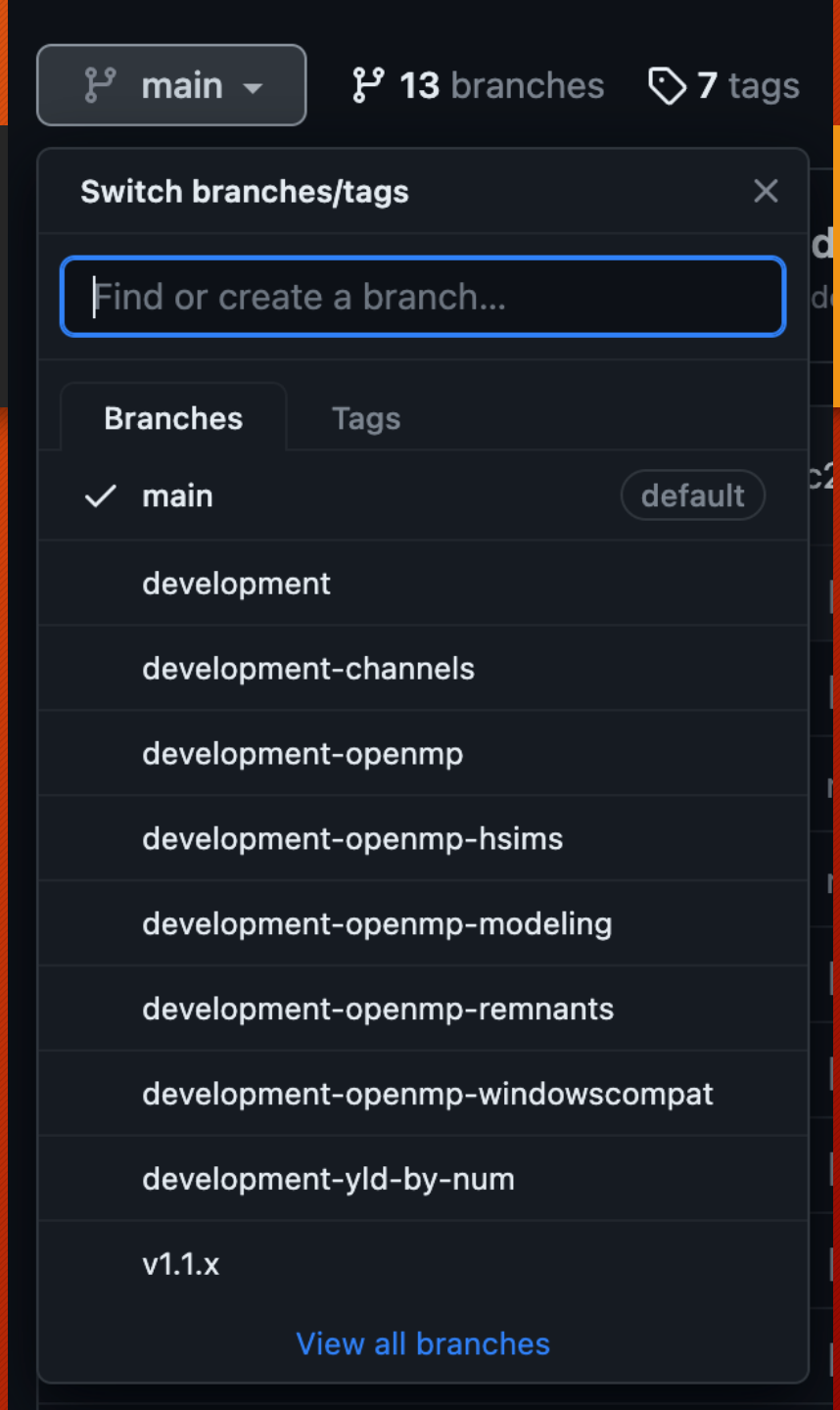
- *git push --set-upstream origin <new_branch>*

You can also create a new branch on GitHub's webpage for your repository, then

git pull

git switch <branch_name>

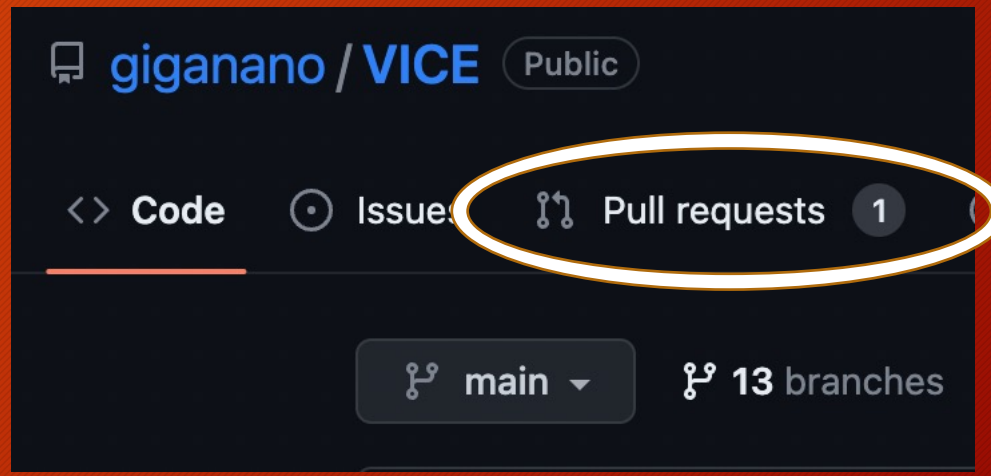
git branch --delete <branch_name> will delete the local copy of the branch



Merging branches

Copies the commits on one branch into another branch

When collaborating, open a *pull request*. Merge will be handled on GitHub's website.



Locally, you can use:

git switch <into_branch>

git merge <from_branch>

git push

How Researchers Benefit from GitHub

At the end of the day, or whenever you need it: *git add* + *git commit* + *git push*

Later, or simply on a different computer: *git pull*

- No need to handle transfers yourself!
- The other system *can* be a supercomputer (e.g., Ohio Supercomputer Center)
- If all you're looking for is managing files between systems, then *add*, *commit*, *push*, and *pull* are all you *really* need

You're backing up your work in the process

"Can you send me a copy of your code?"

"Sure, here's a link that won't change even if the code changes."

GitHub Pages: An easy way to deploy websites

GitHub will host *static* websites for free!

- *Static*: built using HTML/Markdown as opposed to JavaScript (*dynamic*)
- Popular among academics → more freedom than, e.g., google sites
- *https://username.github.io* → URLs of this format are GitHub websites

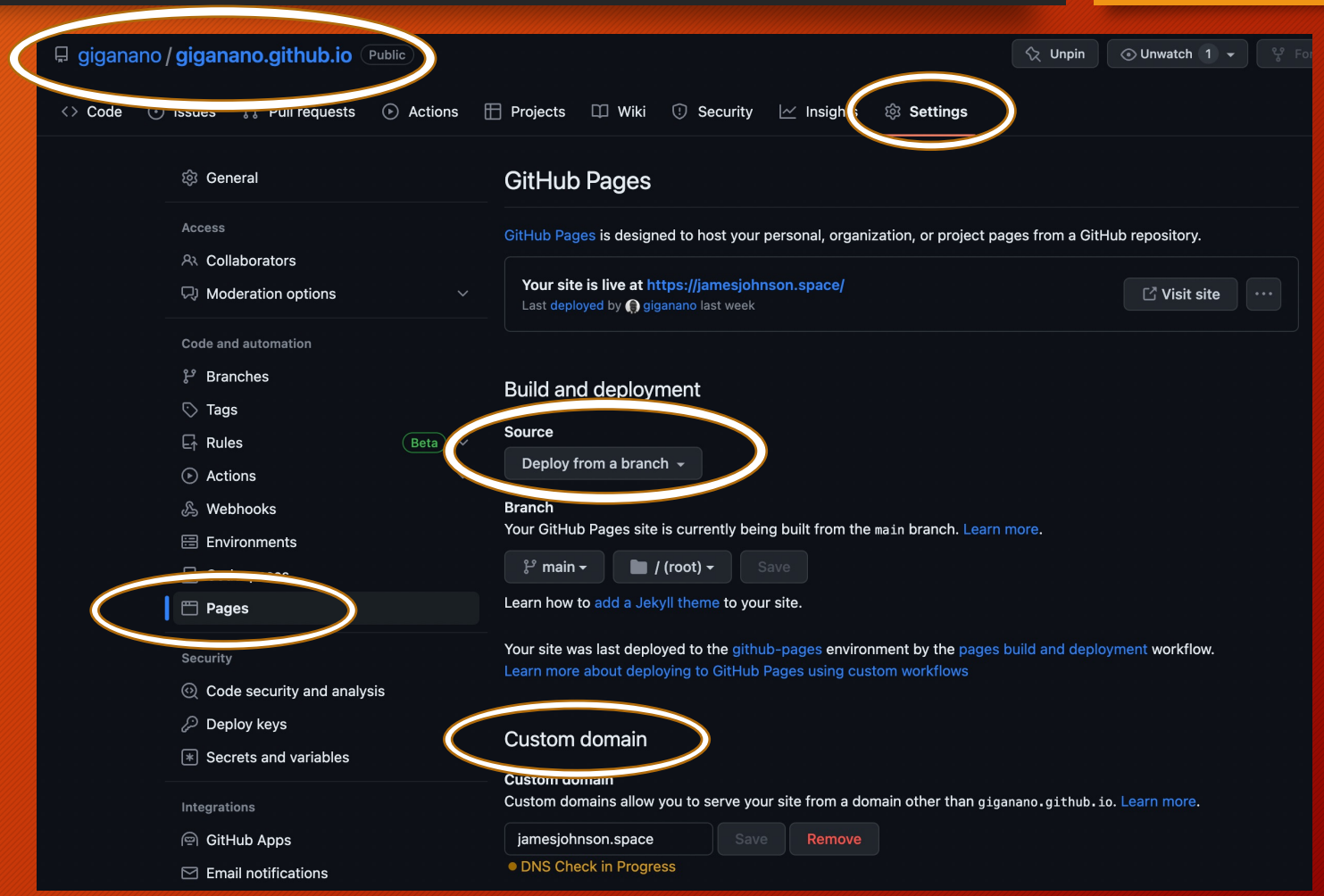
Can also be deployed at any URL that you own the rights to

- Example: *https://giganano.github.io* redirects to *https://jamesjohnson.space*
- The source material for this bootcamp is hosted on a GitHub website
    - *https://jamesjohnson.space/bootcamp* is really just *https://giganano.github.io/bootcamp*

GitHub Pages: An easy way to deploy websites

In repository, navigate to Settings, then Pages, and tell it which branch to deploy from (usually *main*)

In general, the name of the repo should end with *.github.io*



GitHub Pages: An easy way to deploy websites

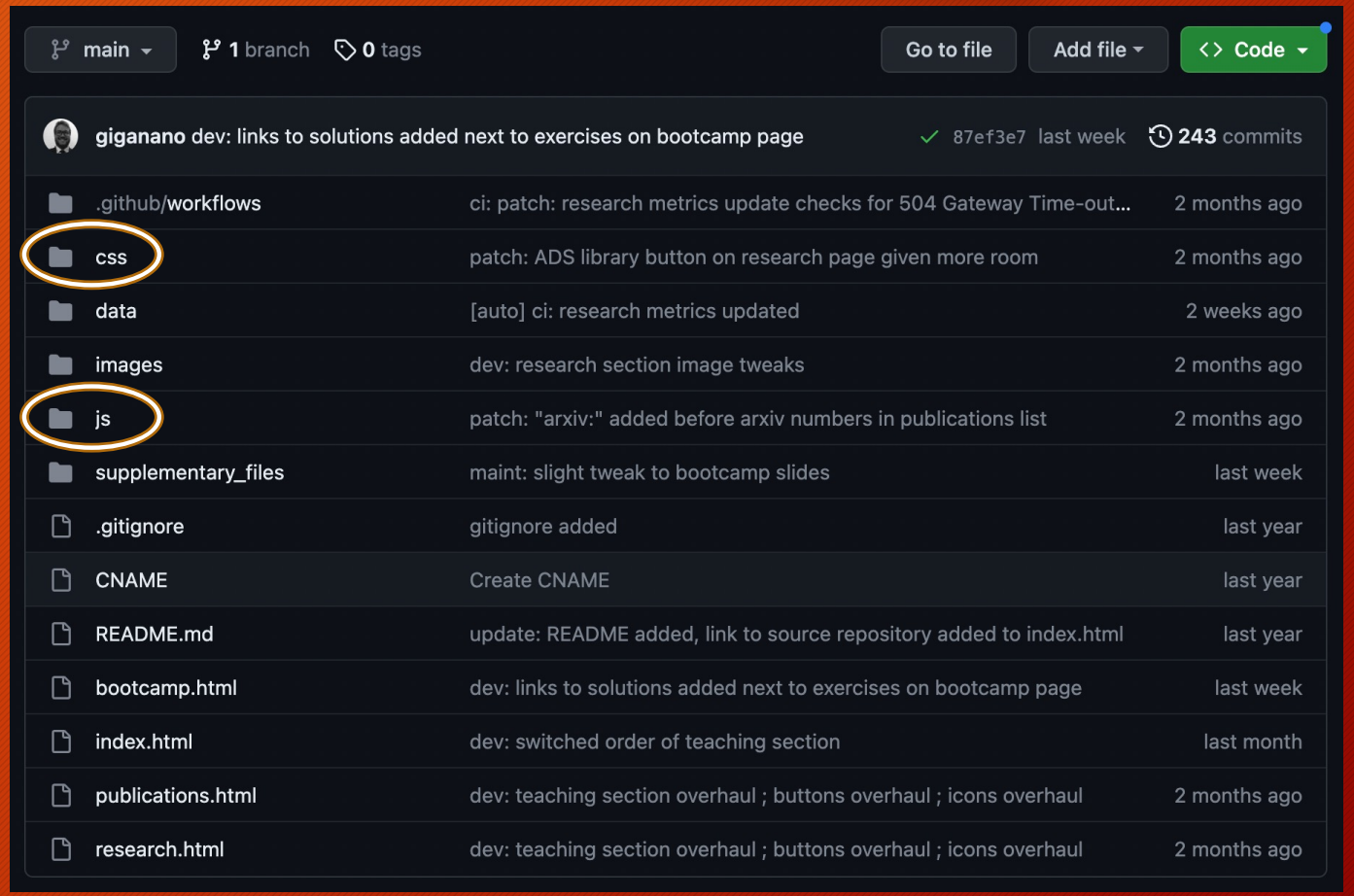
Built using HTML or Markdown

- *index.html* or *index.md* will be the website's front page

Can use your own JavaScript or CSS (Cascading Style Sheets)

Anatomy of a website:

- HTML = bones
- JavaScript = muscles
- CSS = skin



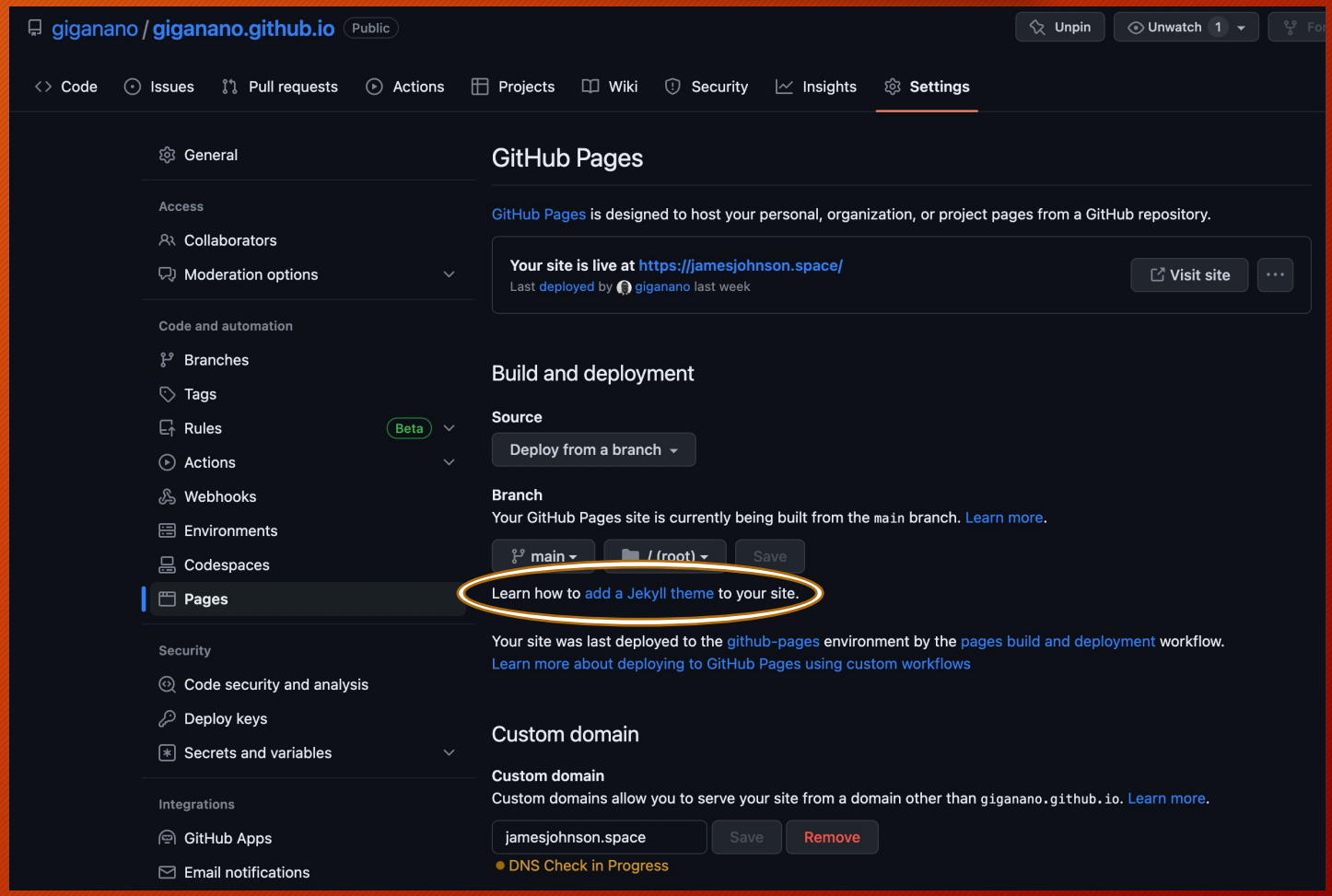
GitHub Pages: An easy way to deploy websites

Built using HTML or Markdown

- *index.html* or *index.md* will be the website's front page

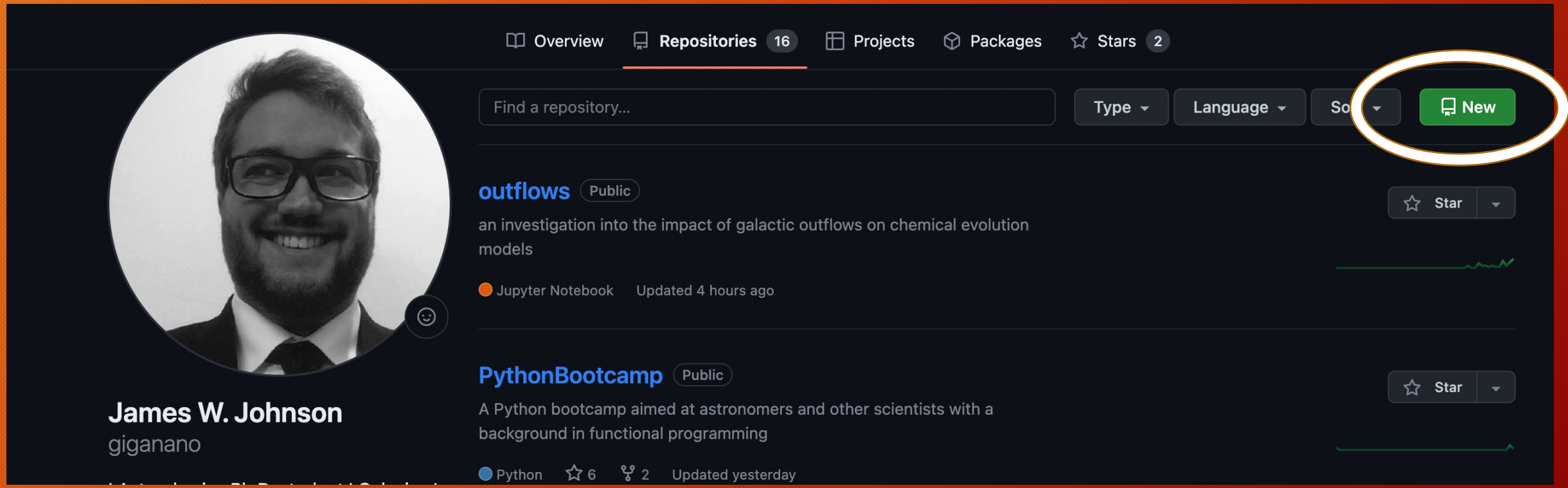
You can simply choose an "off-the-shelf" theme to expedite styling

- Regardless of HTML/Markdown



Creating a repository

Easy if you simply use GitHub's website



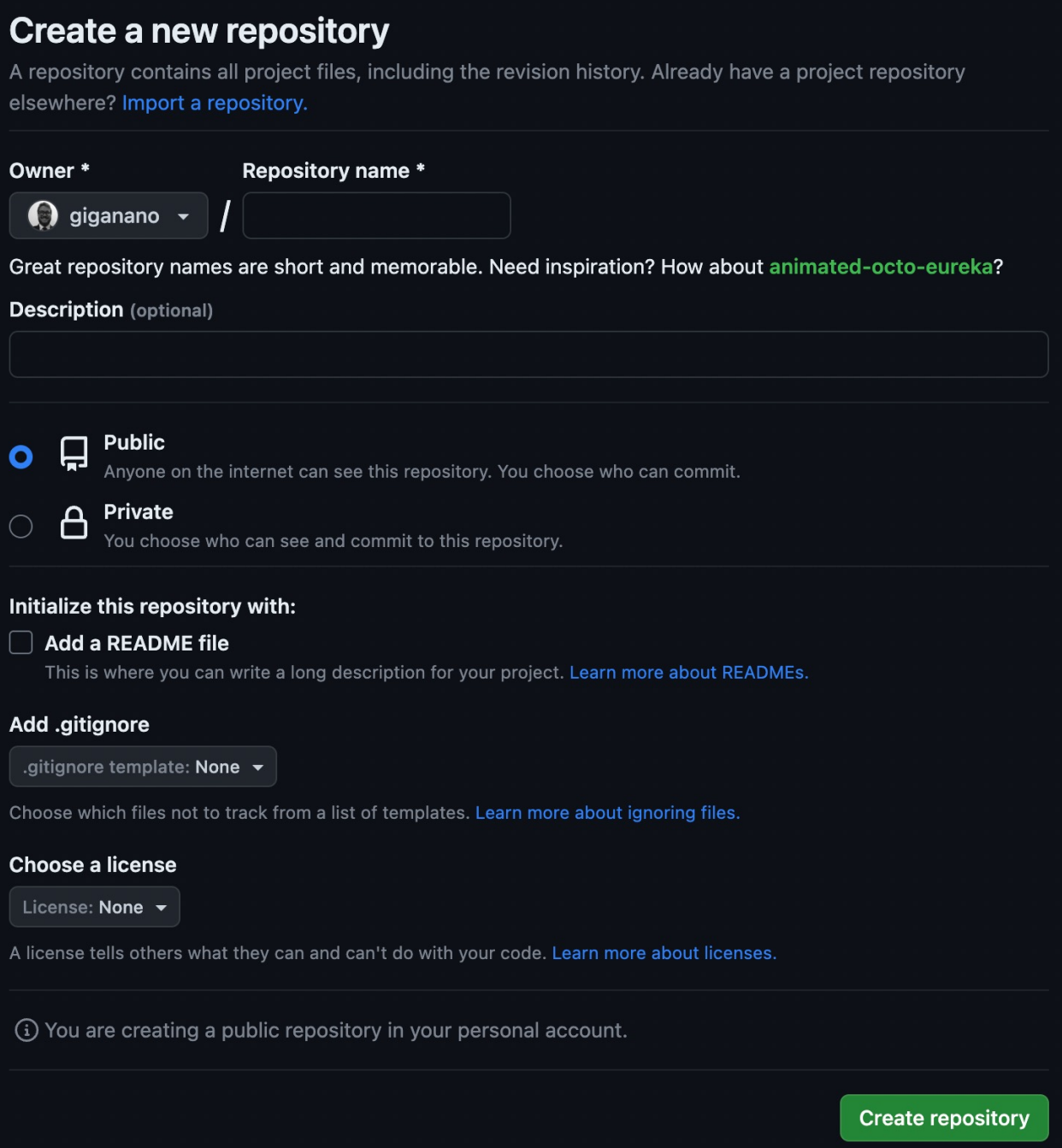
Creating a repository

You'll land at a page that looks like this

- Name it
- Private repos require premium

README, *.gitignore*, and *license* can be added at any time

- Leave them out if this is your first repository



Creating a repository

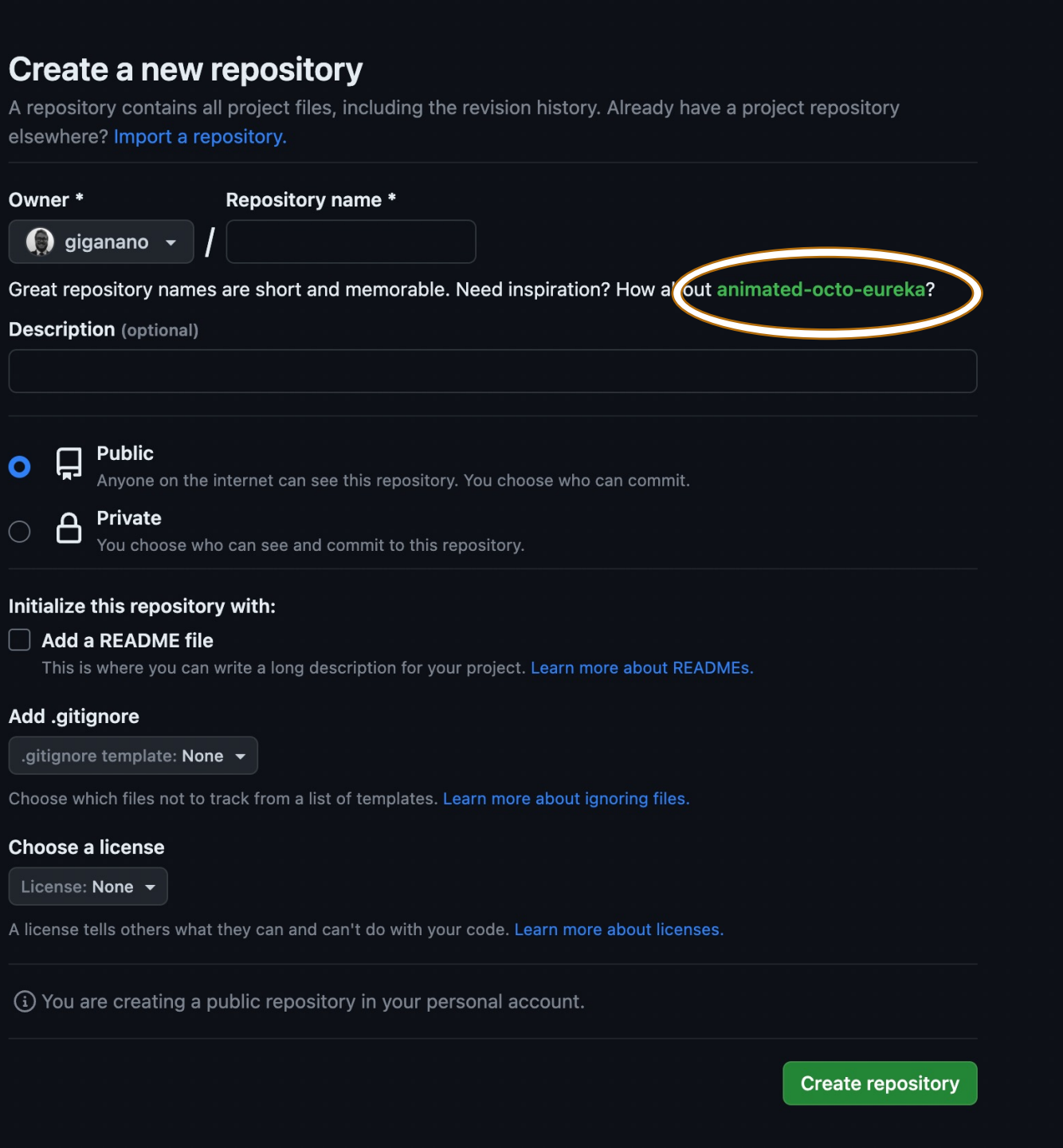
You'll land at a page that looks like this

- Name it
- Private repos require premium

README, *.gitignore*, and *license* can be added at any time

- Leave them out if this is your first repository

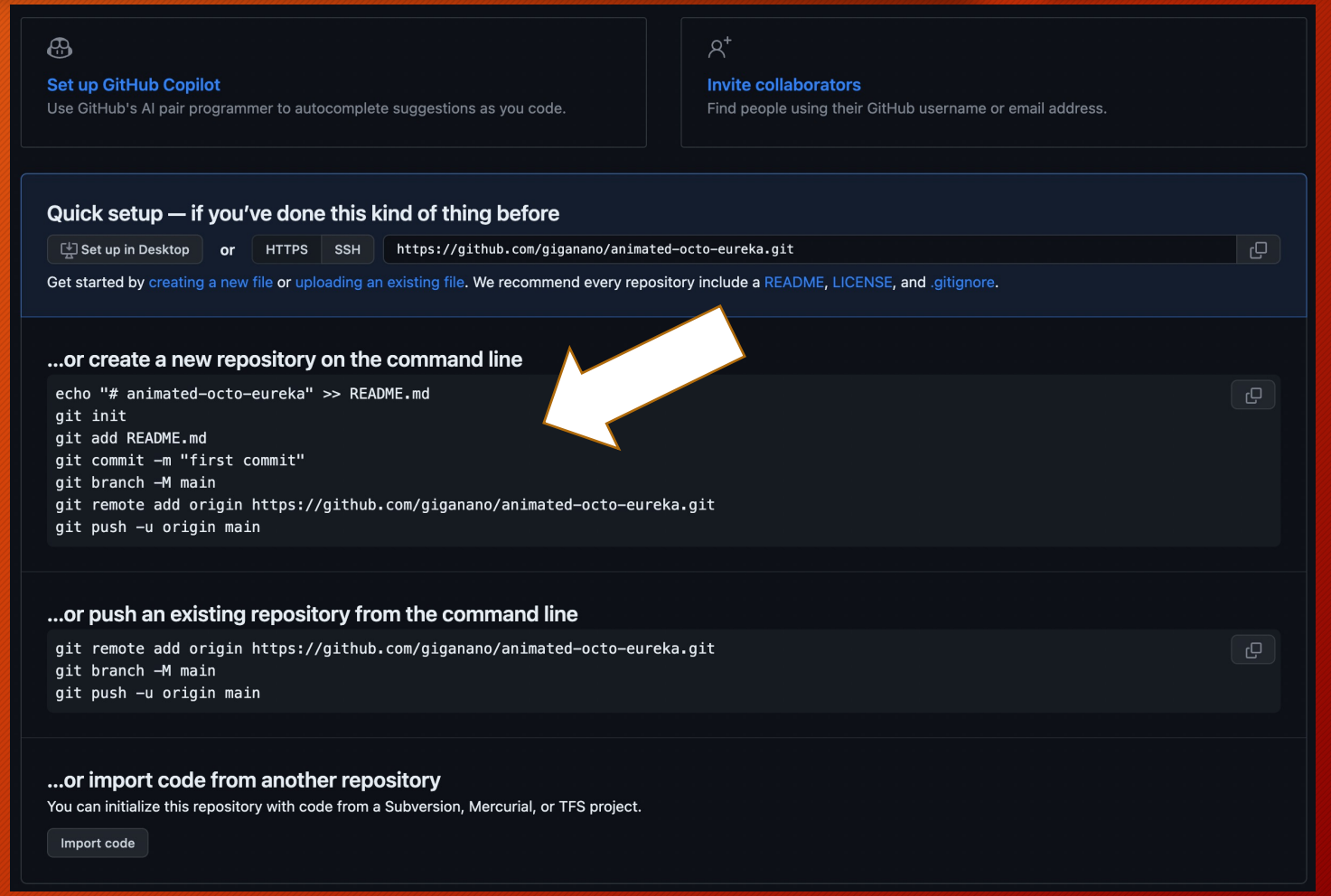
The suggested name is usually so random that it's funny…



Creating a repository

If you leave out the *README*, *.gitignore*, and *license*, then the landing page of an empty repository will walk you through how to make your first push

From here on, everything proceeds according to the earlier slides



Thank you!