Git Essentials Workshop

Use git and start programming more efficiently!

Agenda

- Reasons to use git
- Basic git Terminology + Logic
- Configuring git Settings
- Initializing a Repository
- Bringing it All Together
- Other Commands + Useful Tips!
- WiCS Announcements!

Why should you learn git?

Reasons to learn git!

- Makes version control SO easy!
- Github keeps track of how much you commit!
- Easy way to collaborate with others!
- Eventually use it in your classes (if not already!)
- Global use of git in industry

Basic git Terminology + Logic

git Terminology

- **git***: version control system used via terminal commands
- **Github**: online hosting service to manage repositories
- **repository**: .git/ folder inside a project; tracks all changes; directory for files
- **branch**: creates an independent line of development; for versions/errors
- **fork**: creates an independent copy of a repository
- **add:** adds changes in working directory to the staging area
- **commit**: saves any changes to the local repository
- **push**: adds all your local file changes to your remote branch
- **pull**: updates your local files with any remote changes
- **fetch**: downloads changes made to the remote repository to your local repository

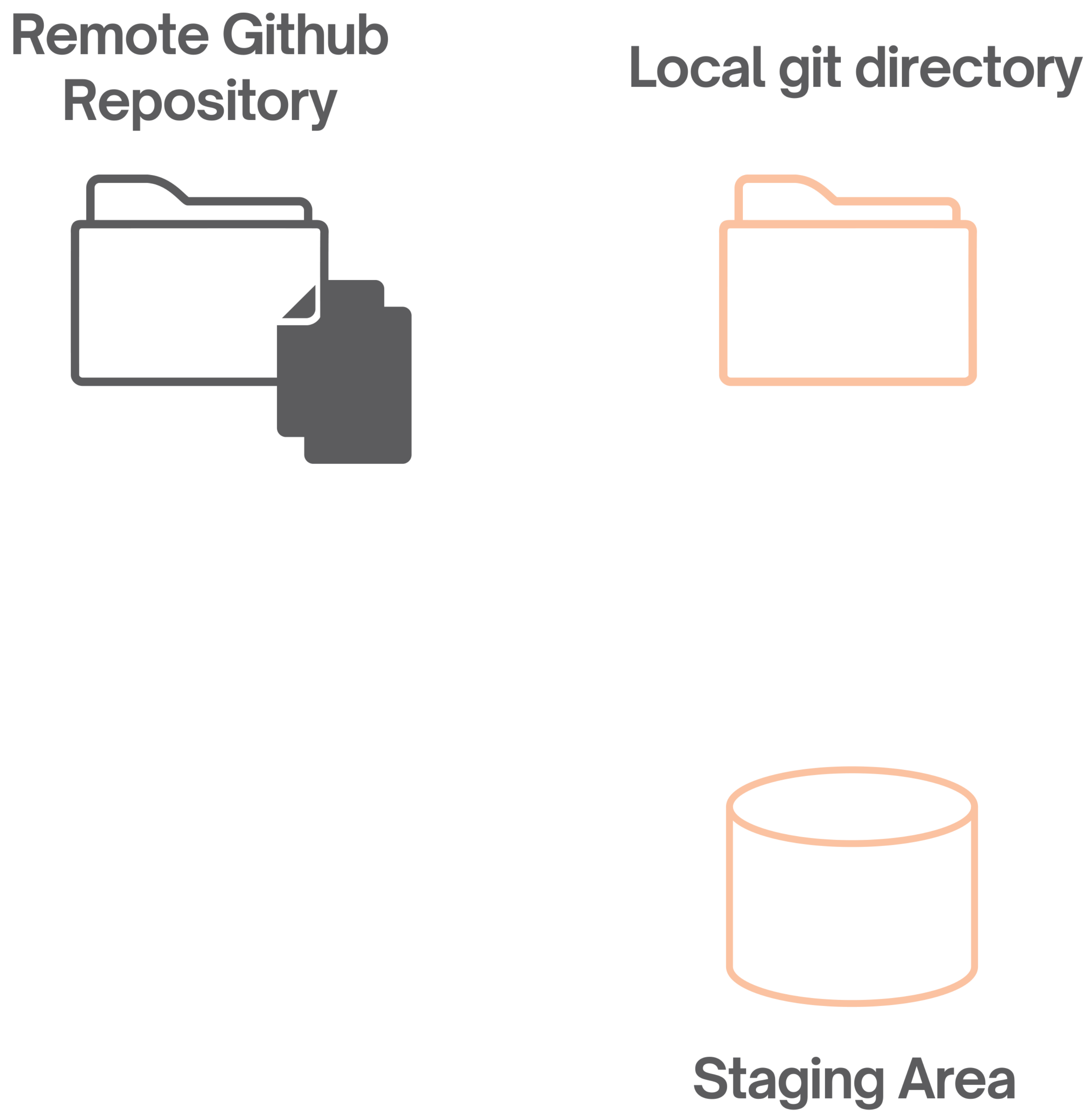
**git =/= Github!*

git Workflow Logic

- We will be discussing the general logic from a high-level
- The following workflow corresponds to terminal commands (i.e. git add, commit, push, pull) which we will demonstrate more in depth afterwards!

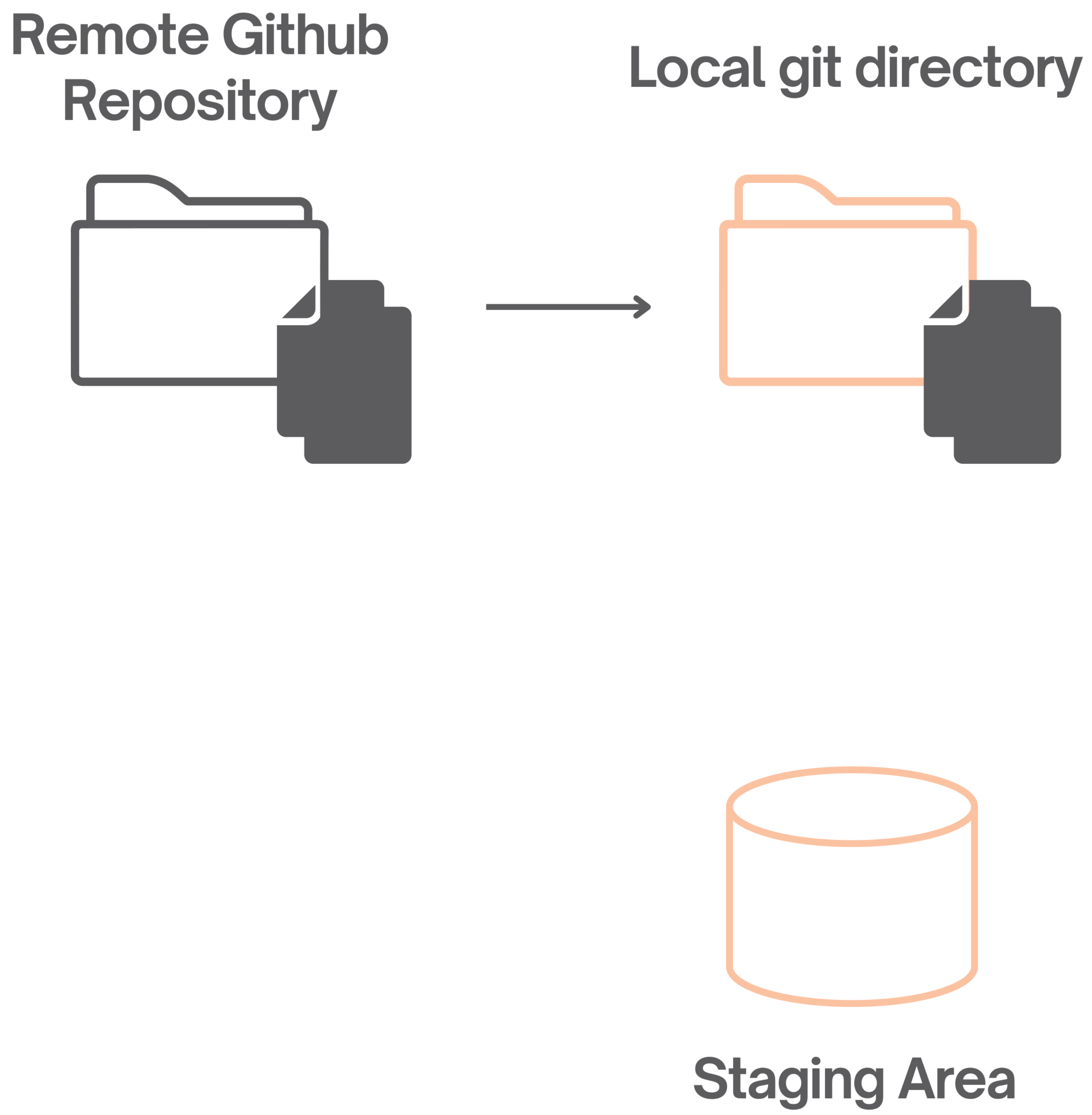
git Workflow Logic

- You have a folder (repository) on Github with all your project files
- You pull all changes to your local directory
- You make changes to your files via your local machine
- You add all the updated files into a "staging area" via your command line
- You update your folder on Github with the changes that you added to the "staging area"



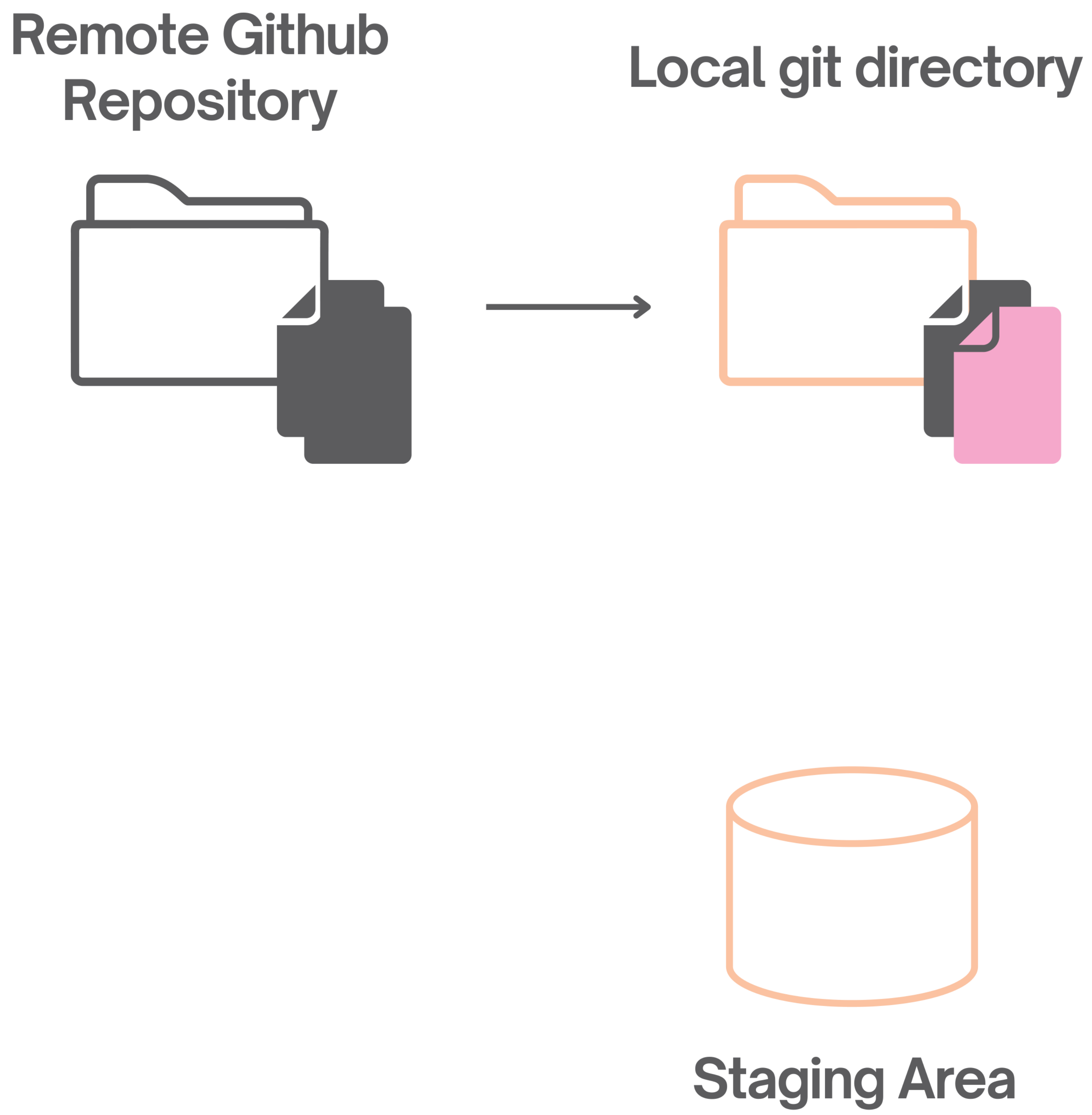
git Workflow Logic

- You have a folder (repository) on Github with all your project files
- You pull all changes to your local directory
- You make changes to your files via your local machine
- You add all the updated files into a "staging area" via your command line
- You update your folder on Github with the changes that you added to the "staging area"



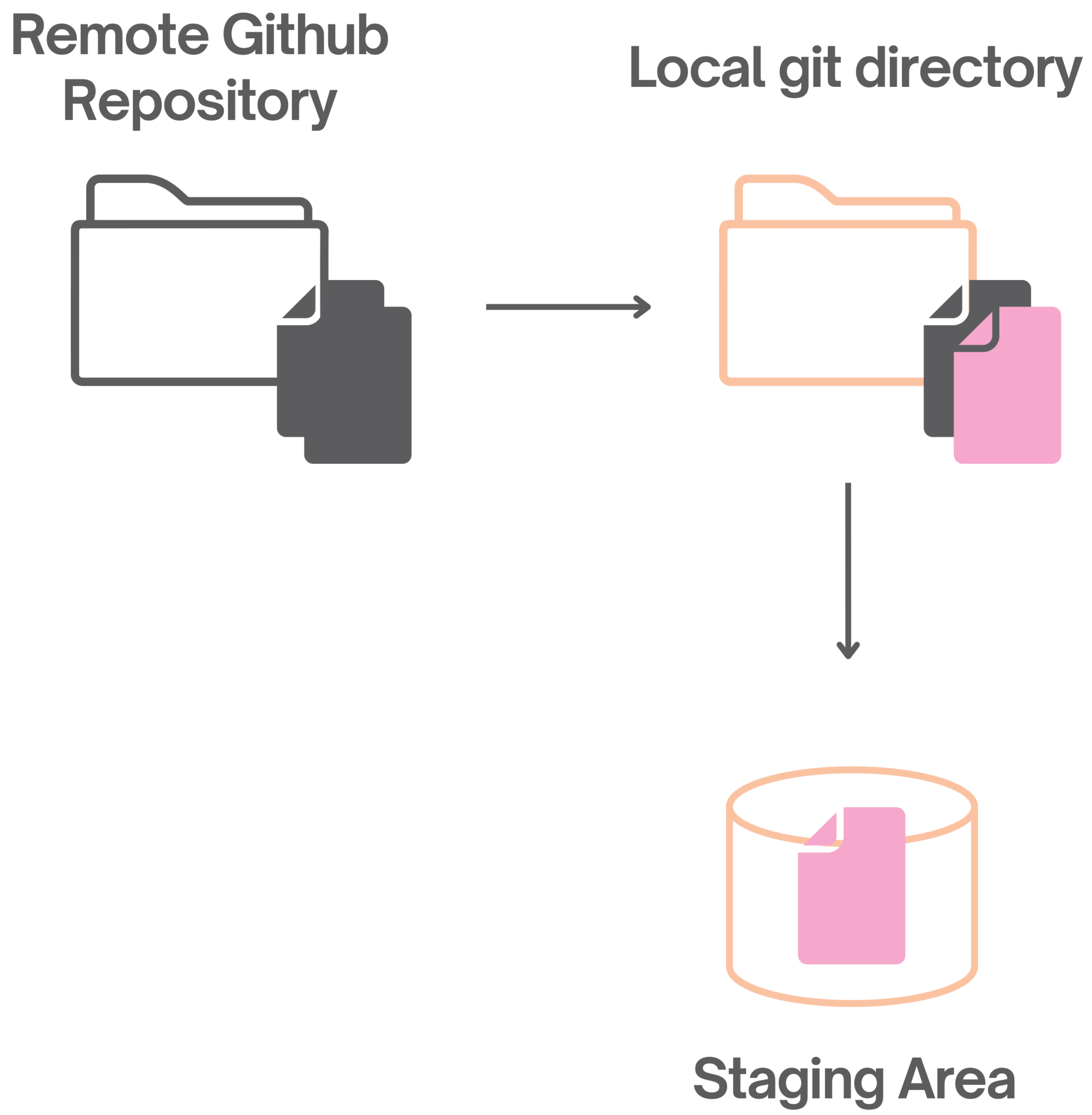
git Workflow Logic

- You have a folder (repository) on Github with all your project files
- You pull all changes to your local directory
- You make changes to your files via your local machine
- You add all the updated files into a "staging area" via your command line
- You update your folder on Github with the changes that you added to the "staging area"



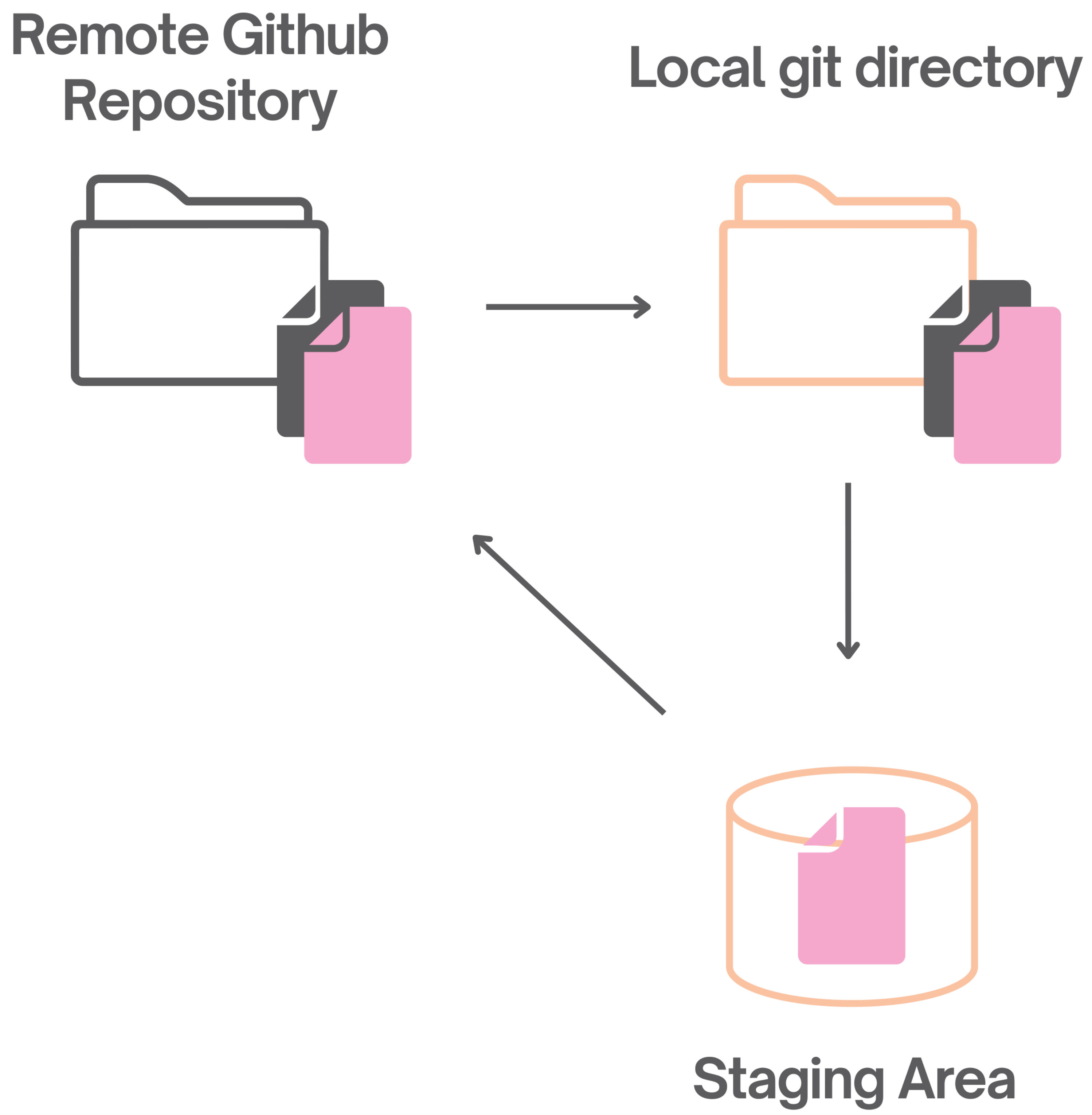
git Workflow Logic

- You have a folder (repository) on Github with all your project files
- You pull all changes to your local directory
- You make changes to your files via your local machine
- You add all the updated files into a "staging area" via your command line
- You update your folder on Github with the changes that you added to the "staging area"



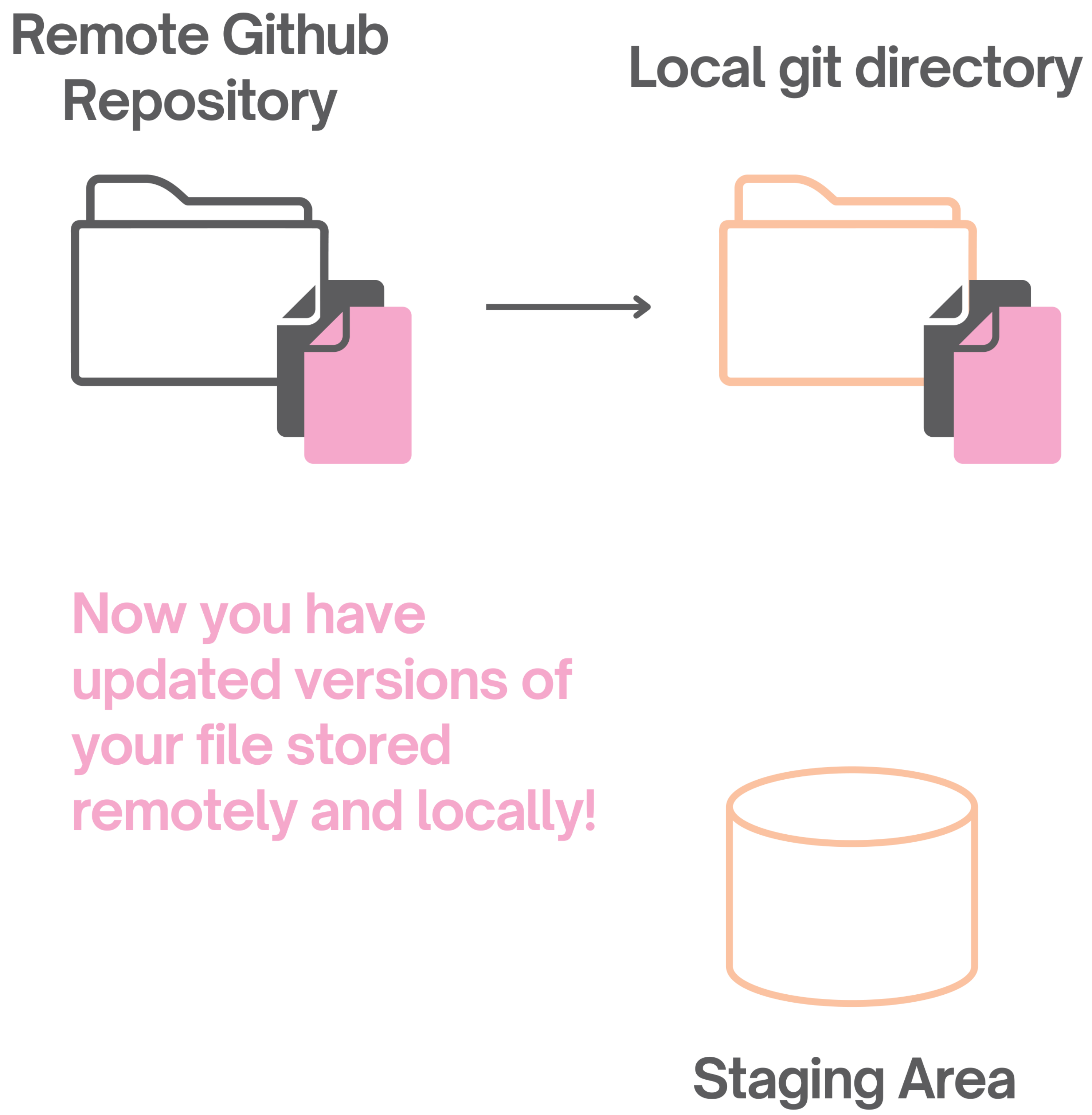
git Workflow Logic

- You have a folder (repository) on Github with all your project files
- You pull all changes to your local directory
- You make changes to your files via your local machine
- You add all the updated files into a "staging area" via your command line
- You update your folder on Github with the changes that you added to the "staging area"



git Workflow Logic

- You have a folder (repository) on Github with all your project files
- You pull all changes to your local directory
- You make changes to your files via your local machine
- You add all the updated files into a "staging area" via your command line
- You update your folder on Github with the changes that you added to the "staging area"

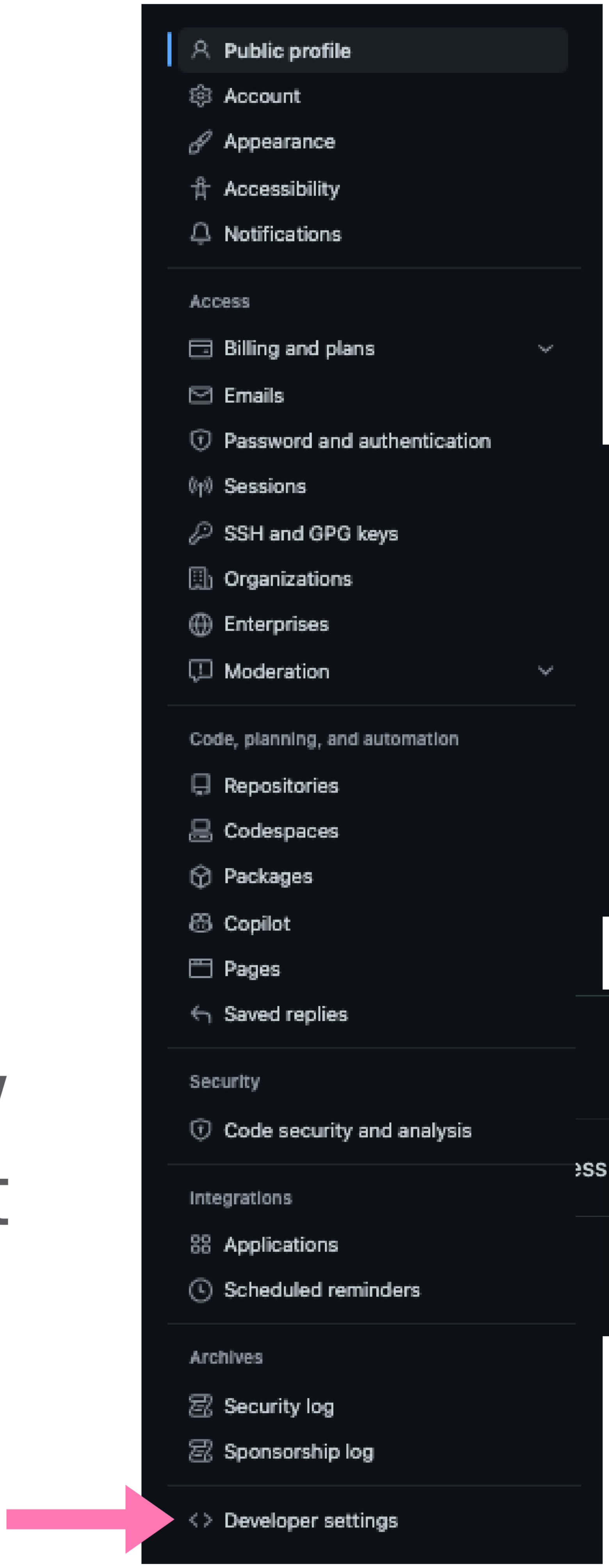

Remote Github Repository
Local git directory
Now you have updated versions of your file stored remotely and locally!
Staging Area

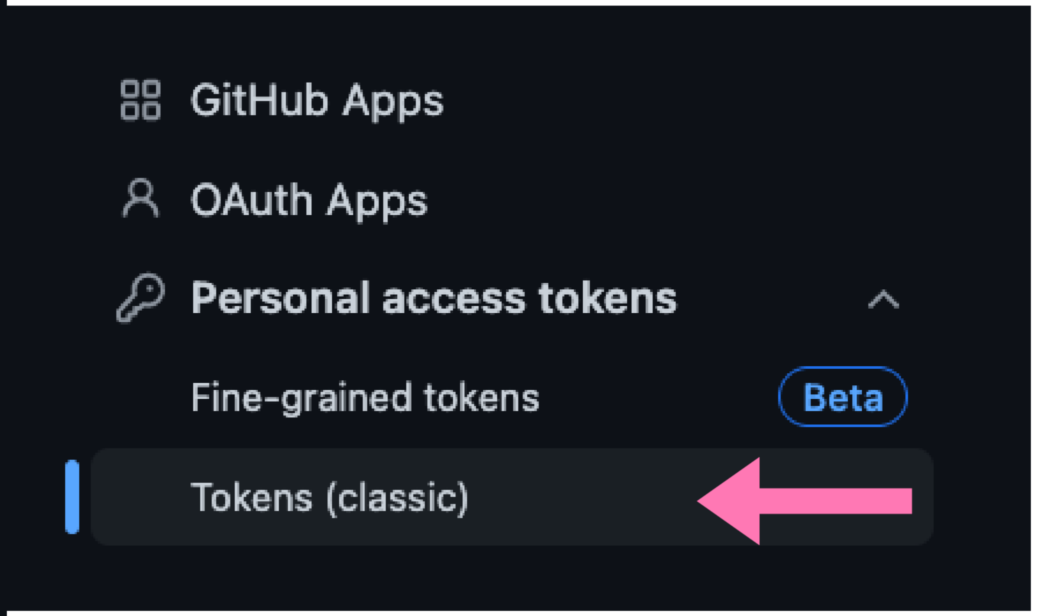
Configuring git Settings

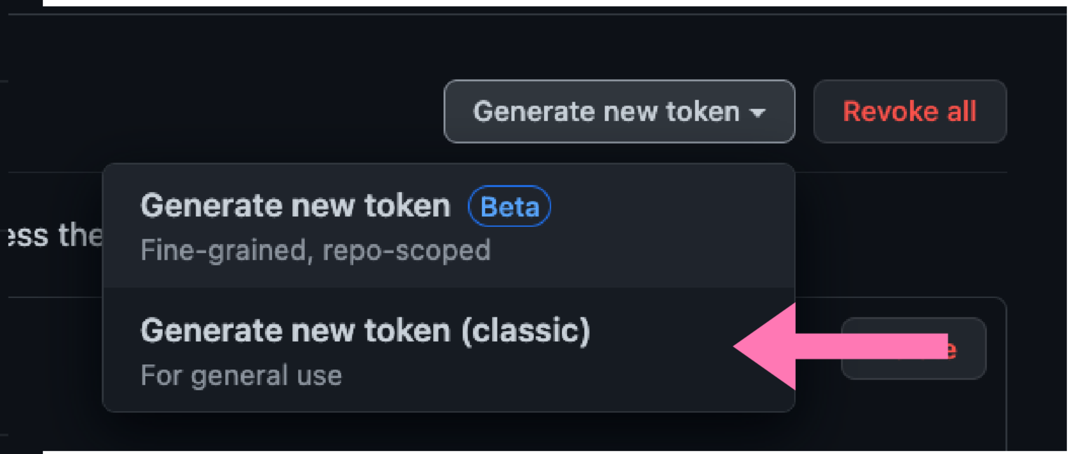
git Configuration

- Generate a personal access token via your settings
  - This will be your "password" when prompted to log in, keep your PAT secure, can't view it again after you've saved it once
- Check relevant permissions i.e. "repo"

Profile>Settings> Developer Settings







git Configuration

- Set your git settings via your terminal/CLI
  - `git config --global user.name <"your.username">`
  - `git config --global user.email <"your.email">`*

- Only have to do this once

make sure to omit brackets

Initializing a Repository

git First Steps

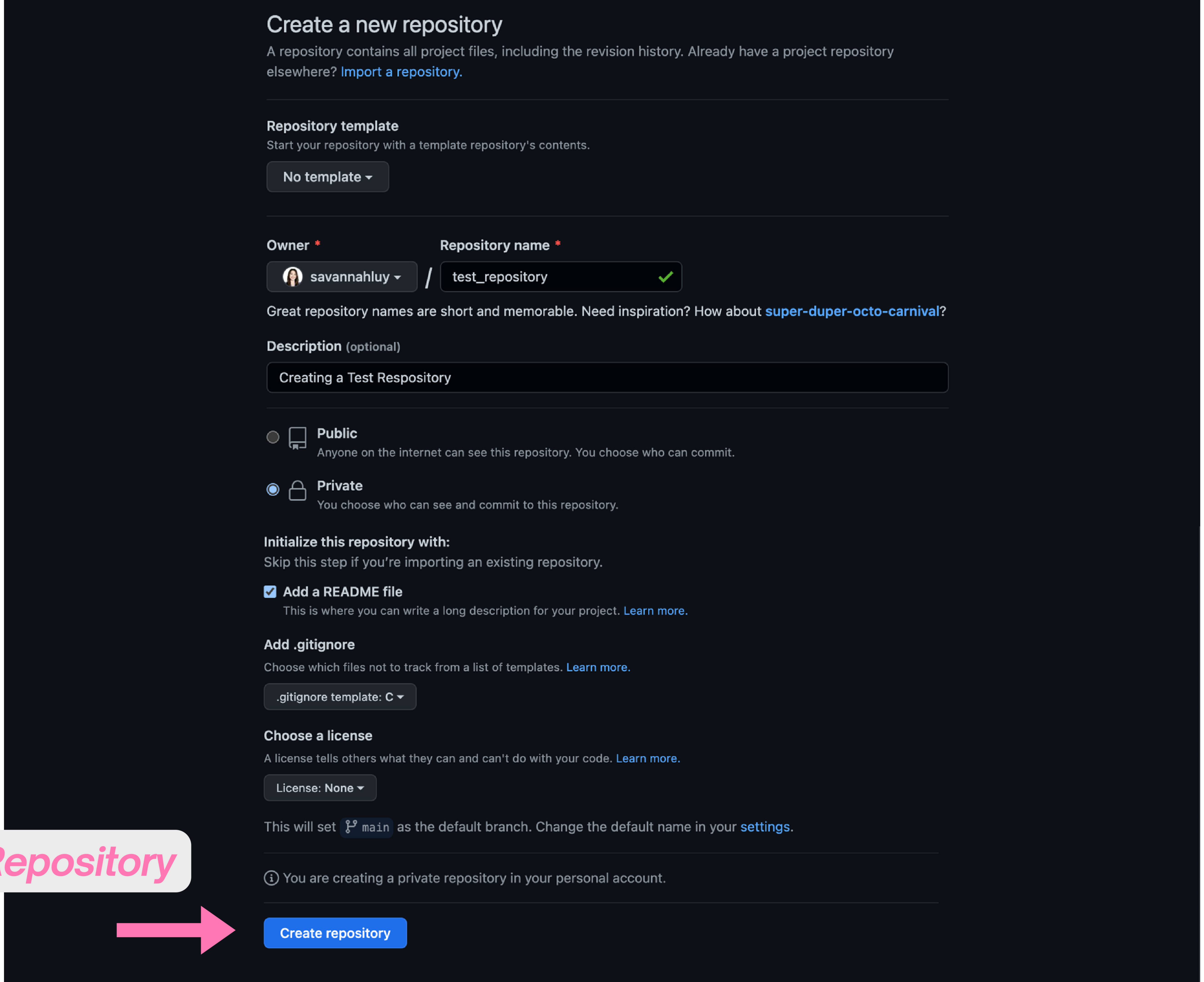
- Create a remote repository on Github
  - This is where all your up-to-date project files will live
- Create a local repository from your terminal/command line
  - This is the directory where you would usually have your files stored
- Connect your local repository to your remote one (or vice versa*)

**multiple ways to do this, discussed in next two slides*

git remote to local

- Create a new remote repository on Github
  - Upload all your project files using "Add File"
- Clone your remote repository via terminal from your directory

  -
    ```
    git clone <https://github.com/username/repo-name.git>
    ```
- Upon first clone, the terminal will prompt you for your username and password (which is the personal access token)



Create a new repository

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

Repository template

Start your repository with a template repository's contents.

No template

Owner * savannahluy / **Repository name** * test_repository ✓

Great repository names are short and memorable. Need inspiration? How about super-duper-octo-carnival?

Description (optional)

Creating a Test Respository

○ **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:

Skip this step if you're importing an existing repository.

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

Add .gitignore

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

.gitignore template: C

Choose a license

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

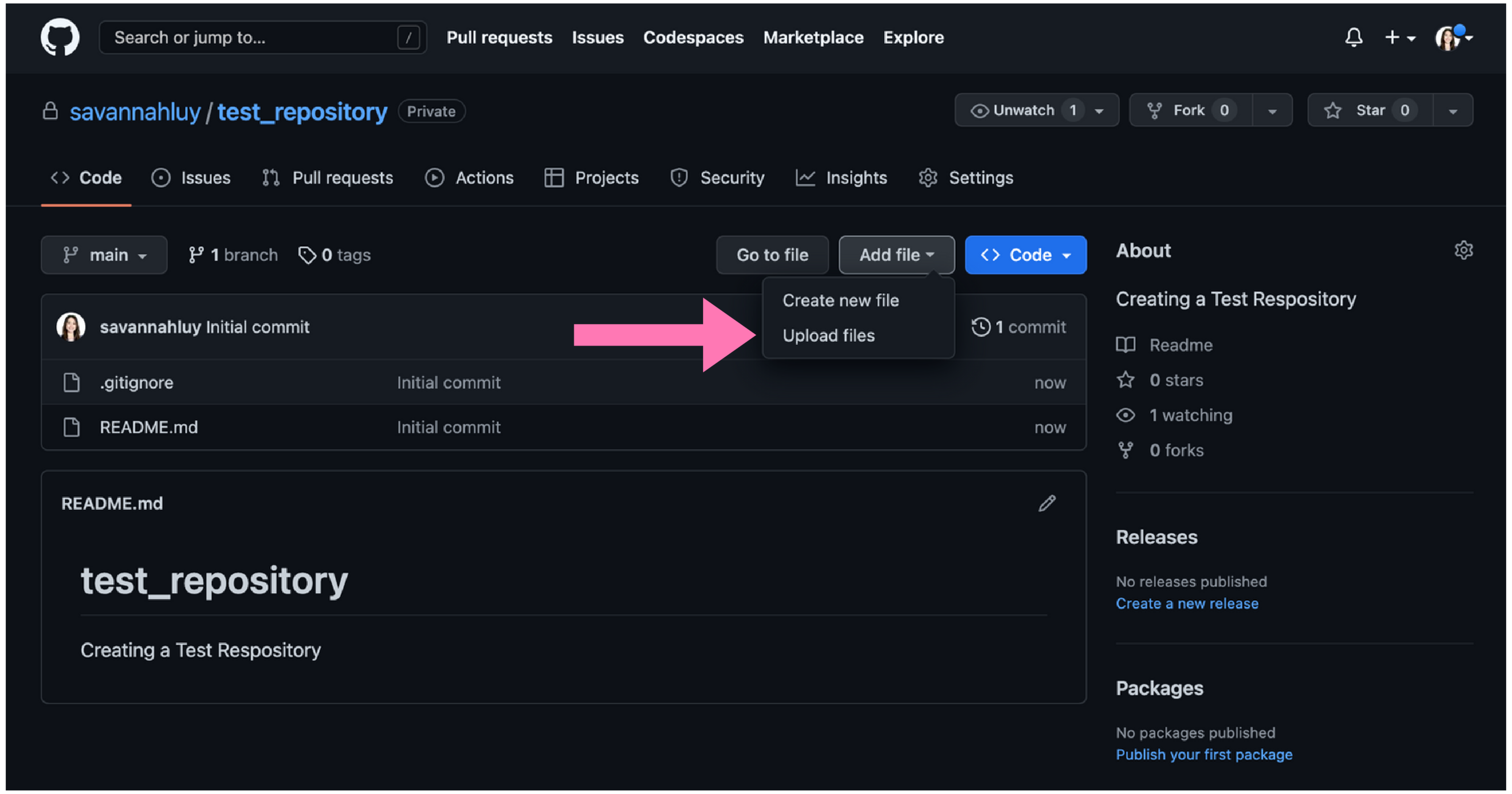
License: None

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

ⓘ You are creating a private repository in your personal account.

Create a Repository

Upload Files



savannahluy Initial commit · 1 commit

.gitignore	Initial commit	now
README.md	Initial commit	now

README.md

test_repository

Creating a Test Respository

Creating a Test Respository

Readme
0 stars
1 watching
0 forks

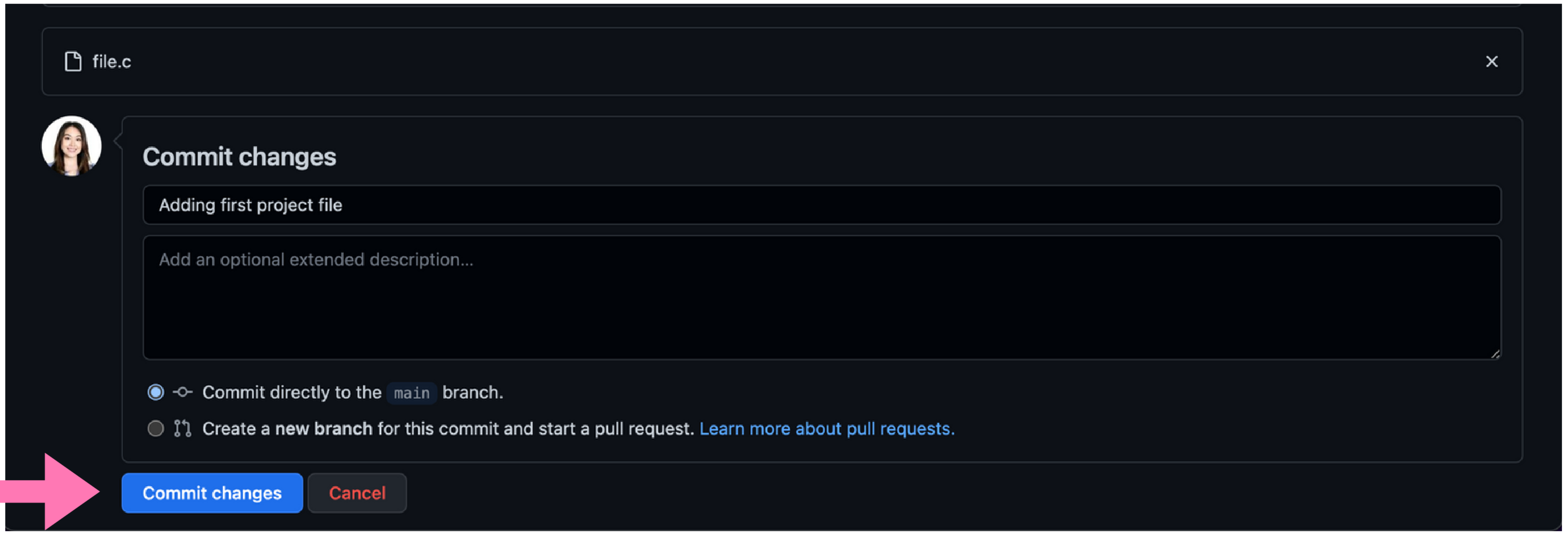
Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Commit Changes



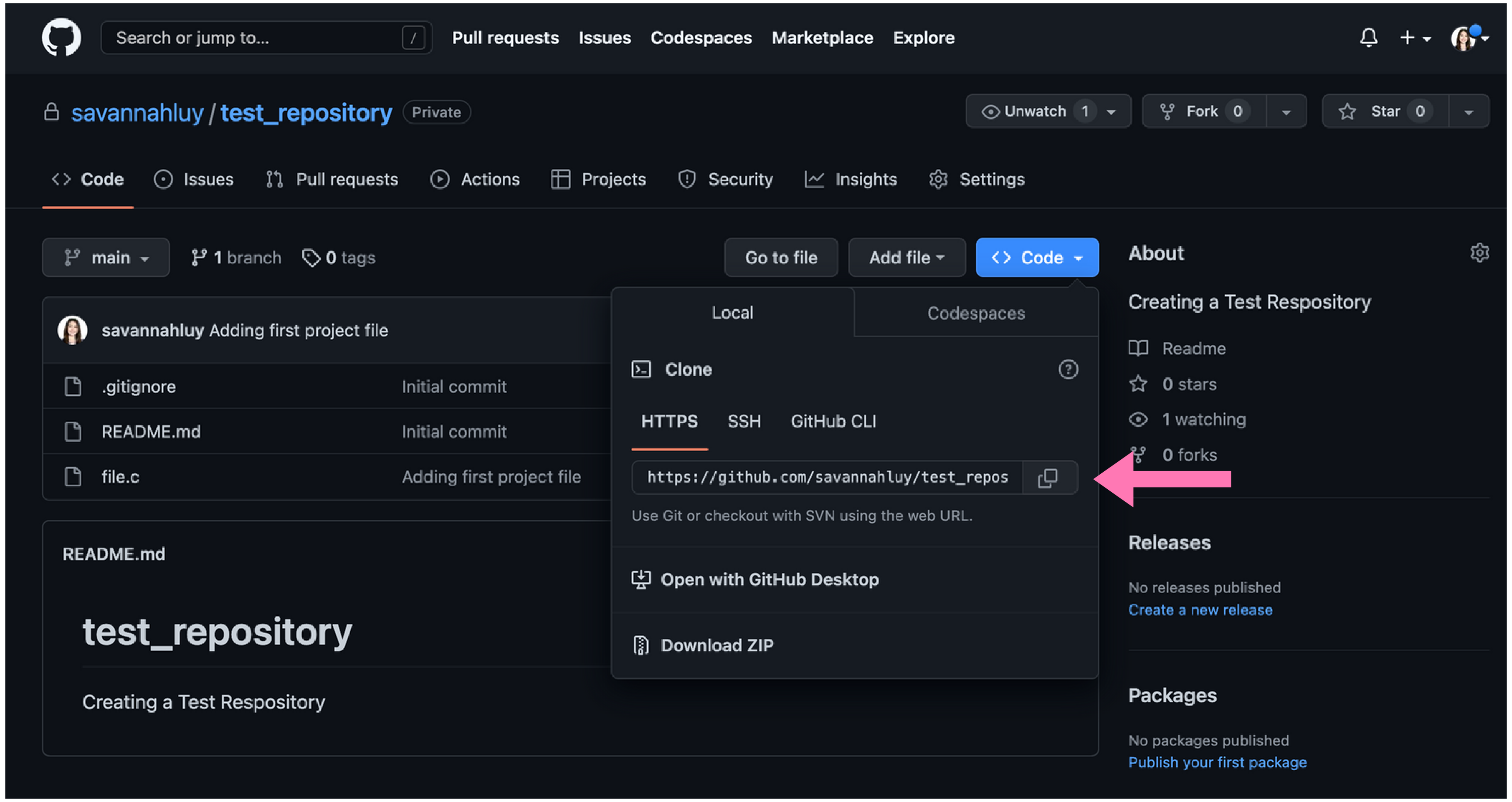
Adding first project file

Add an optional extended description...

Commit directly to the `main` branch.

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Copy the HTTPS line



savannahluy Adding first project file

.gitignore	Initial commit
README.md	Initial commit
file.c	Adding first project file

https://github.com/savannahluy/test_repos

Use Git or checkout with SVN using the web URL.

README.md

test_repository

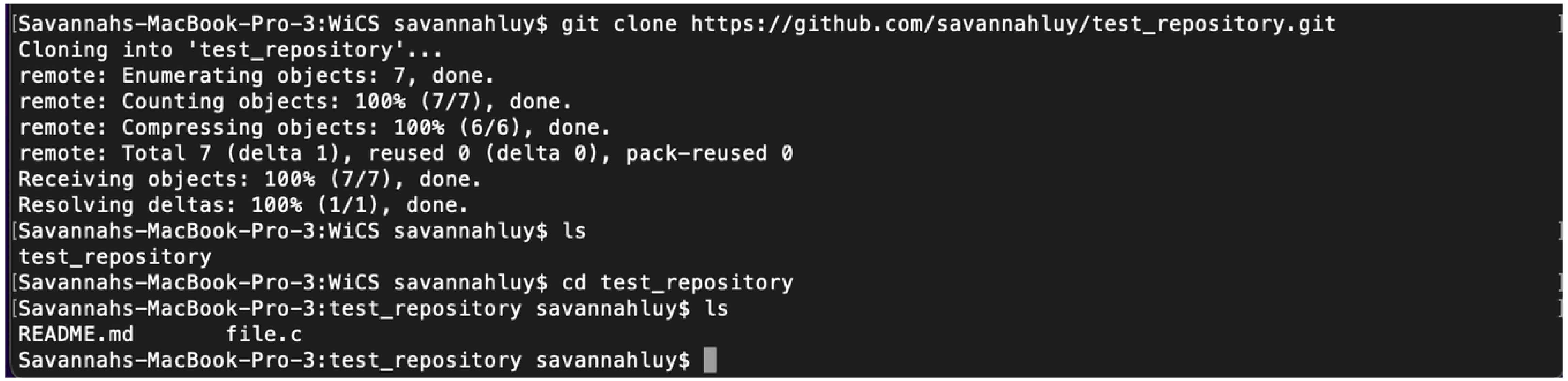
Creating a Test Respository

Creating a Test Respository

No releases published
Create a new release

No packages published
Publish your first package

Run the git clone <HTTPS> line in your working directory



```
Savannahs-MacBook-Pro-3:WiCS savannahluy$ git clone https://github.com/savannahluy/test_repository.git
Cloning into 'test_repository'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), done.
Resolving deltas: 100% (1/1), done.
Savannahs-MacBook-Pro-3:WiCS savannahluy$ ls
test_repository
Savannahs-MacBook-Pro-3:WiCS savannahluy$ cd test_repository
Savannahs-MacBook-Pro-3:test_repository savannahluy$ ls
README.md       file.c
Savannahs-MacBook-Pro-3:test_repository savannahluy$
```

git local to remote

- Create a new remote repository on Github without a README or .gitignore file (tells Git which files in the repository not to track)
- Initialize a local repository and commit files via terminal

```
○ git init
○ git add <filename>
○ git commit -m "Commit Message - Adding first file"
```

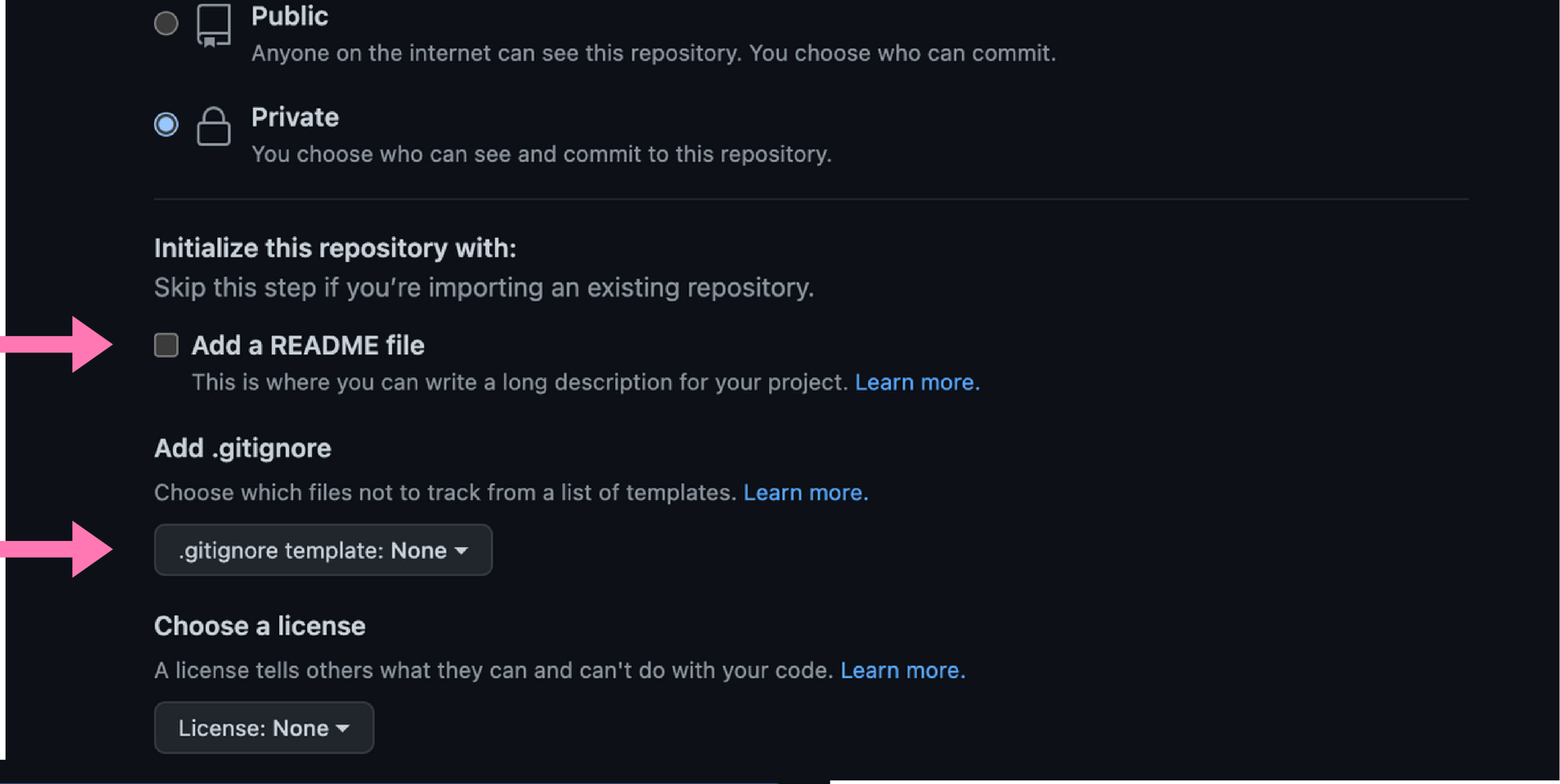
git local to remote

- Connect your remote repository via terminal

```
◦ git remote add origin
   https://github.com/username/repo_name.git
◦ git branch -M main         // setting default branch
◦ git push -u origin main    // push to main branch
```

- Refresh your remote repository on Github

Create new remote repo without README or .gitignore file


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:
Skip this step if you're importing an existing repository.

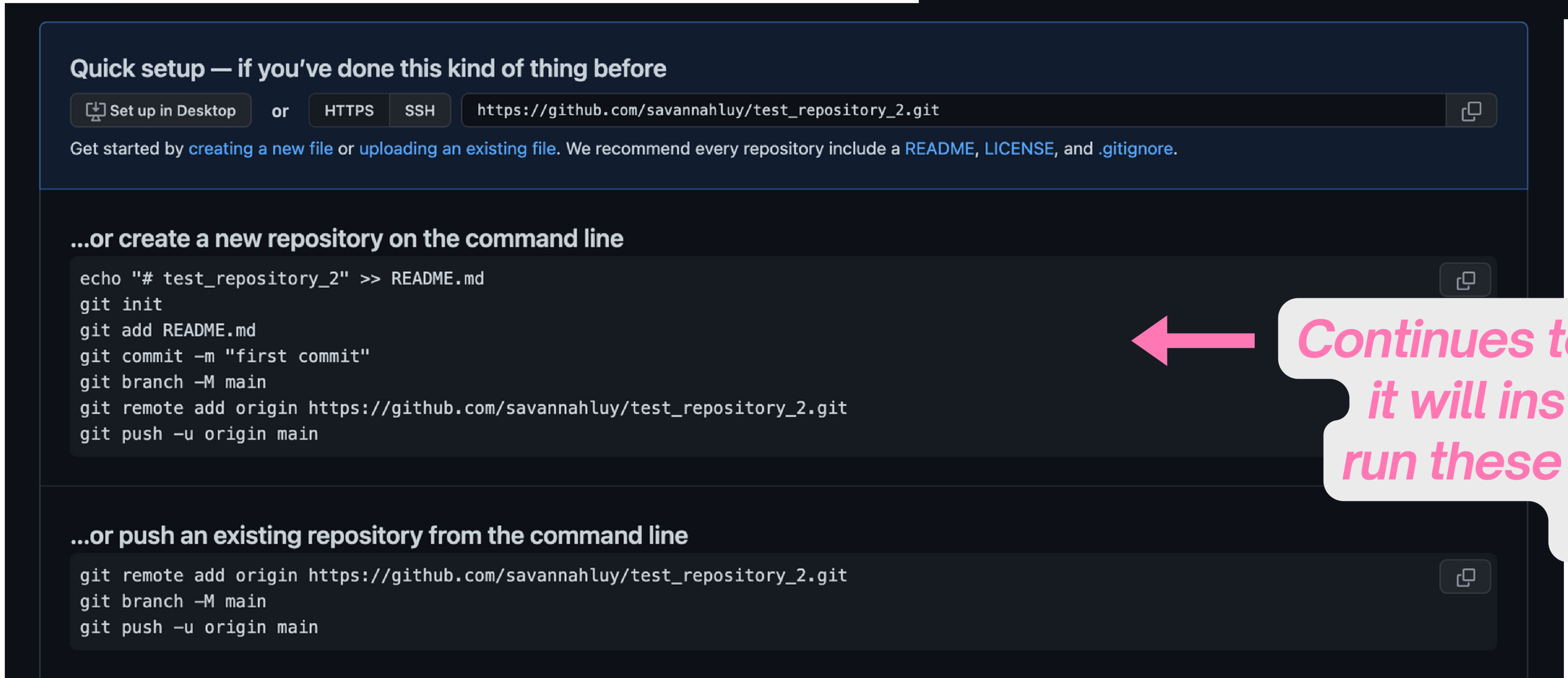
Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

.gitignore template: None

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

License: None


Quick setup — if you've done this kind of thing before

https://github.com/savannahluy/test_repository_2.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

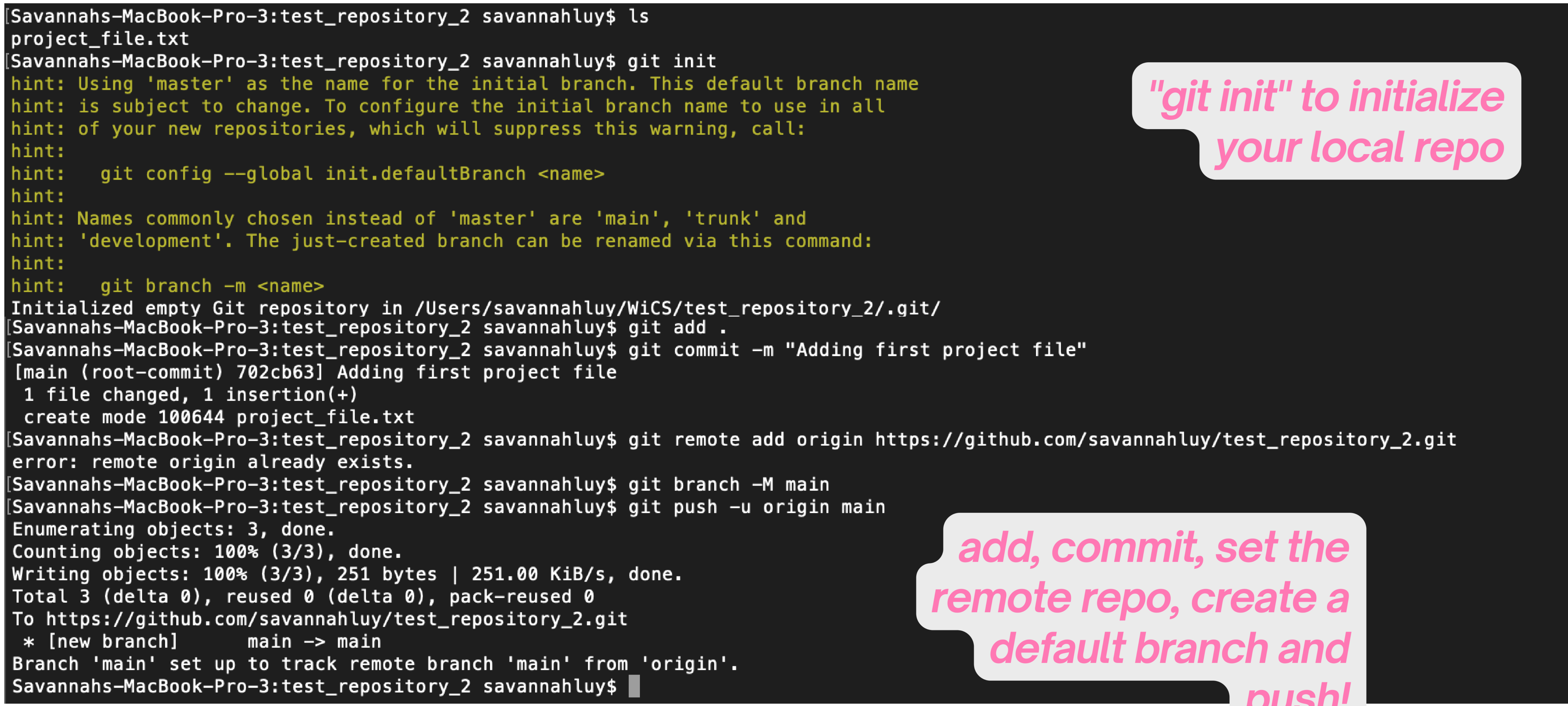
...or create a new repository on the command line

```
echo "# test_repository_2" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/savannahluy/test_repository_2.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/savannahluy/test_repository_2.git
git branch -M main
git push -u origin main
```

Continues to this page, it will instruct you to run these commands via terminal

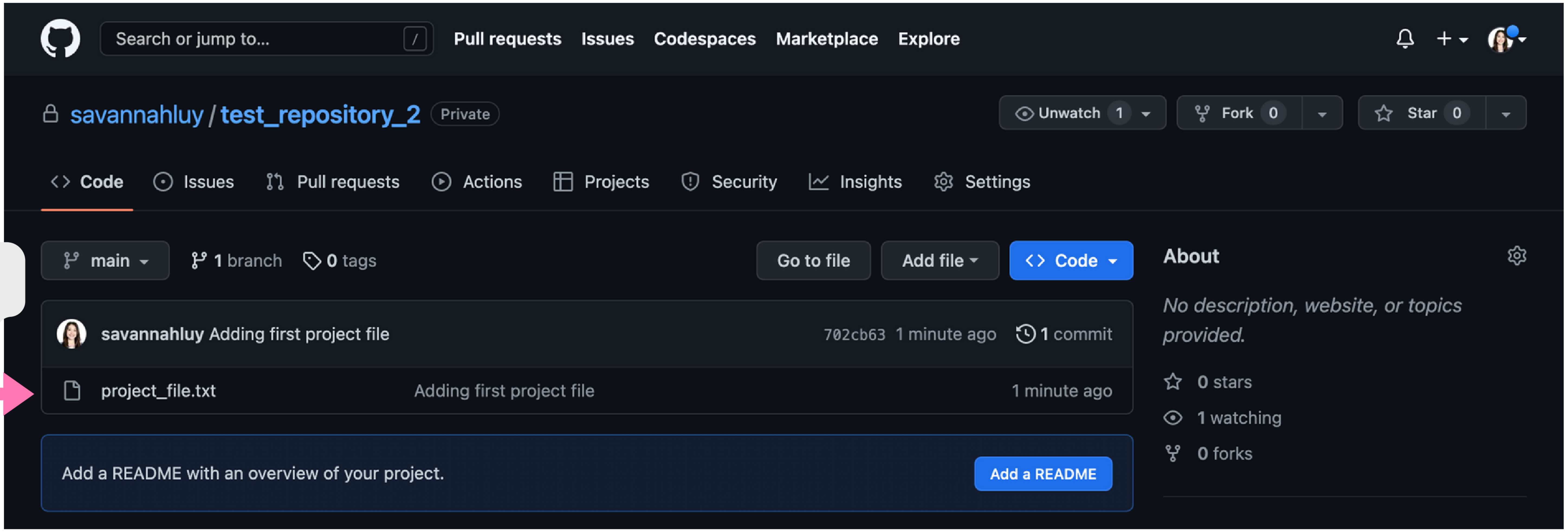


```
[Savannahs-MacBook-Pro-3:test_repository_2 savannahluy$ ls
project_file.txt
[Savannahs-MacBook-Pro-3:test_repository_2 savannahluy$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/savannahluy/WiCS/test_repository_2/.git/
[Savannahs-MacBook-Pro-3:test_repository_2 savannahluy$ git add .
[Savannahs-MacBook-Pro-3:test_repository_2 savannahluy$ git commit -m "Adding first project file"
[main (root-commit) 702cb63] Adding first project file
 1 file changed, 1 insertion(+)
 create mode 100644 project_file.txt
[Savannahs-MacBook-Pro-3:test_repository_2 savannahluy$ git remote add origin https://github.com/savannahluy/test_repository_2.git
error: remote origin already exists.
[Savannahs-MacBook-Pro-3:test_repository_2 savannahluy$ git branch -M main
[Savannahs-MacBook-Pro-3:test_repository_2 savannahluy$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 251 bytes | 251.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/savannahluy/test_repository_2.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
Savannahs-MacBook-Pro-3:test_repository_2 savannahluy$
```

"git init" to initialize your local repo

add, commit, set the remote repo, create a default branch and push!

Refresh and your files will appear!


savannahluy / test_repository_2 Private

main · 1 branch · 0 tags

savannahluy Adding first project file · 702cb63 1 minute ago · 1 commit

project_file.txt · Adding first project file · 1 minute ago

Add a README with an overview of your project.

About

No description, website, or topics provided.

0 stars

1 watching

0 forks

Bringing it All Together

git Workflow Logic Commands

- Pull* all and any changes:

```
◦ git pull
```

- Make changes to your files, save, then run either two cmds:

```
◦ git add <filename> // adds a specific file
◦ git add -A  // adds all modified and untracked
  files in the entire repository
```

**IMPORTANT: pull your changes from your remote repository every time you begin coding to avoid merge conflicts*

git Workflow Logic Commands

- Commit* your files to the staging area:
  - `git commit -m "Detailed Commit Message"`
- Push your commit to the remote branch:
  - `git push`

**Every logical change = 1 set of git add, commit, push commands*

git Branches

- Repositories have multiple branches (can be local and/or remote)
- The default branch is "main"
    - This branch typically contains your most up-to-date, polished, functioning code
    - Avoid pushing/merging to this branch if your code has bugs
- "checkout" a new branch to work these errors out

git branch + git checkout

- Show the current working branch:
  - `git branch`
- Move to a branch:
  - `git checkout <branch_name>`
- Create and move to a new branch:
  - `git checkout -b <new_branch_name> <branchToCopy>`

git Merges

- Merge your working branch into your main branch once you have solved the error
- Merge conflicts occur when project files on one branch have too many/arbitrary differences from project files on another
- Merge conflicts are inevitable, but it's important to reduce them as much as possible!
- Pushing and pulling your code often will help you avoid conflicts

git merge

- Merge* <branch_name> into current working branch
    - i.e. perform this command from main
    - `git merge <branch_name>`

- Return to the previous state after you git merged:
    - `git merge --abort`

**Remember to git push your merges!*

Other Commands + Useful Tips

Other Helpful git Commands

- **STATUS:** display current working branch + modified files

```
git status
```

- **DELETE:** delete a branch

```
git branch --delete <branch_name>
```

- **RESTORE:** restore file to state before previous pull

```
git restore <filename>
```

Useful Tips

- Create a README file to give collaborators configuration/project information
- .gitignore file is used to ignore files containing access keys, any other irrelevant files like .DS_Store
- Write meaningful commit messages like you would with comments!
- Remember to pull changes when collaborating + communicate with your team members!

That is all for now! Let us know if you need any help! Happy commiting! :)

WiCS Community Resources

- Now Available on Notion!
- Includes past slide decks, guides, interview prep materials, and (up and coming) informational databases!

Academic Planning Workshop

- Deepa is hosting a workshop with the CS department all about how to make an academic plan (three quarter plan)
- This Friday 11:00AM-11:45AM
- For CS and CSE majors!
- Register!

Workshop Registration

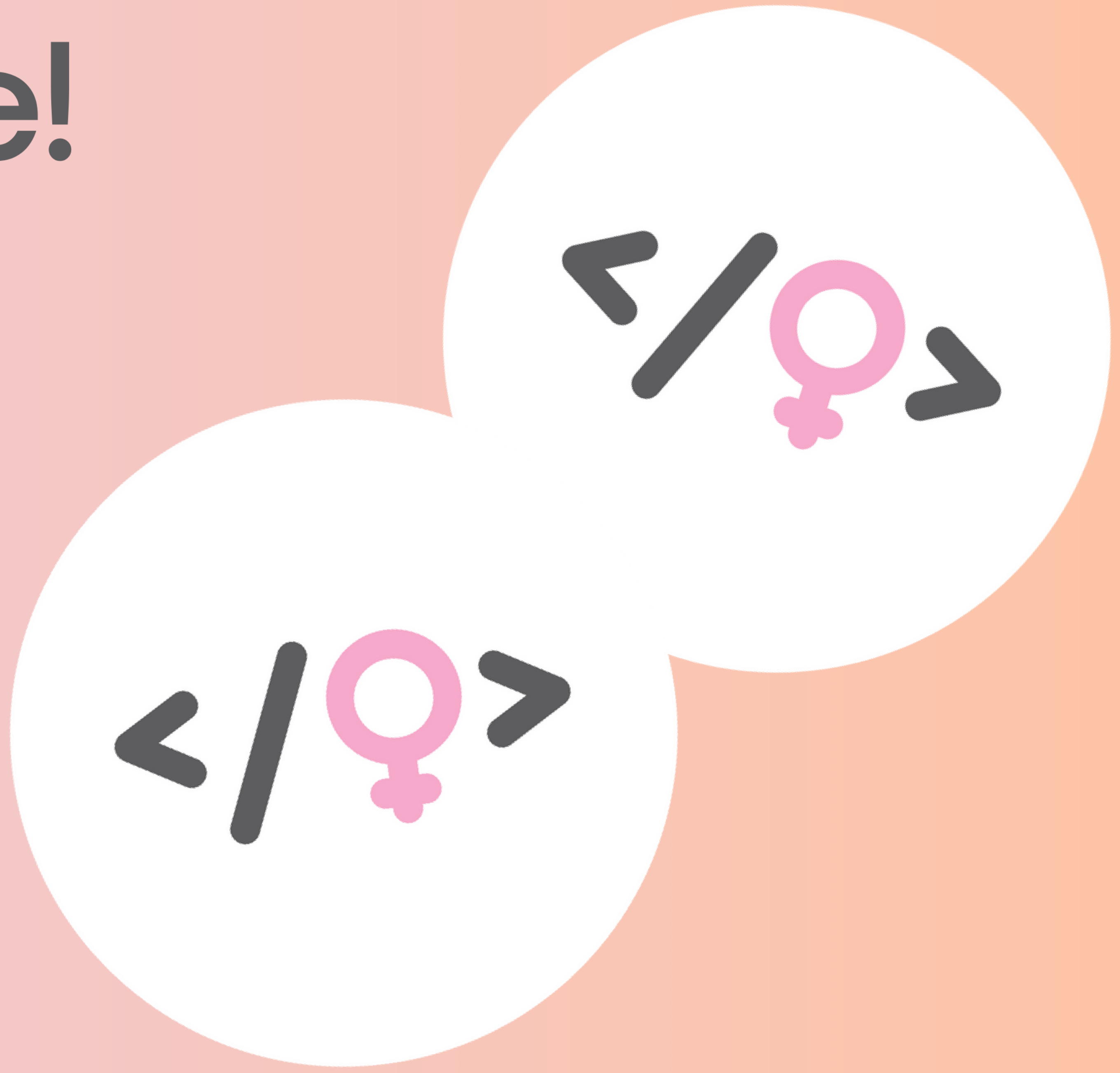
Board Applications - EXTENDED!!

- Fill out by Friday, April 21st @11:59 pm
- Role descriptions are attached to the form!
- If you are selected to interview, you will receive an email regarding availability in the week to follow
- Feel free to reach out to us via wicsdavis@gmail.com if you have any questions!

Board Application Form

WiCS Stickers are Here!

- Support WiCS by purchasing a sticker! Your contribution helps us provide resources to our community!
- Cost: $3 for 1, $4 for 2!
- Venmo: @britney-du-nguyen with the tagline, "WiCS Sticker"



Thank you for attending! See you next time!

Tech Talks with Sandia Labs

TLC 1010, Monday, April 24th, 6-7 pm