<repositories>

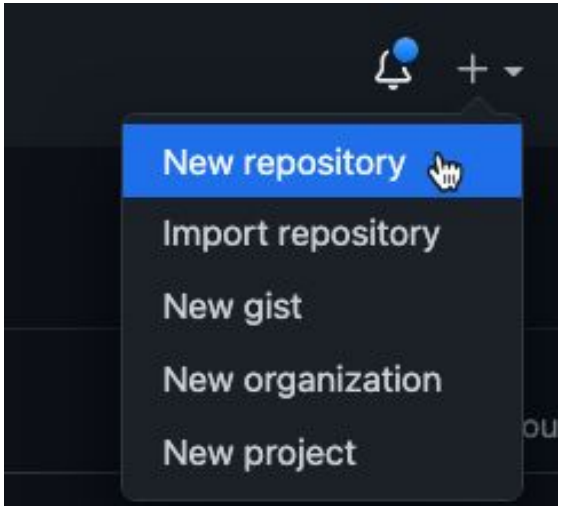
AKA: repos.

What is a repository?

- A repository is a collection of files that you've told Git to pay attention to
  - Usually, one project = one repository
  - Really large projects might have multiple repositories for different parts of their system (ie: front end vs back end)
  - Repositories can live on GitHub and/or your computer
- Let's make one now!

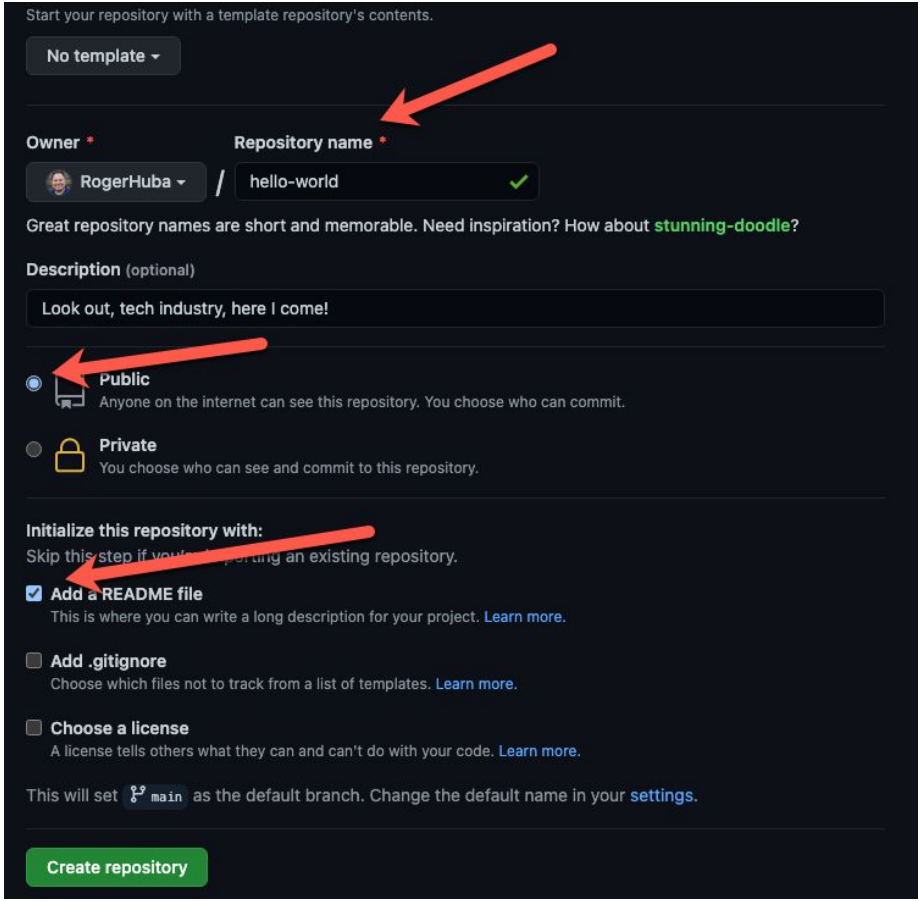
Creating a repo on GitHub

- Log in to GitHub.
- At the top right side of the window, look for your name and avatar.
- Next to it you'll find a small + sign. Click that.
- From the menu that opens, select *New repository.*



Creating a repo on GitHub

- Repos can be named anything…
- But pick something meaningful
- Add a description
- Make ‘Public’
- Check: *Add a README file.*
- Click: *Create repository*.

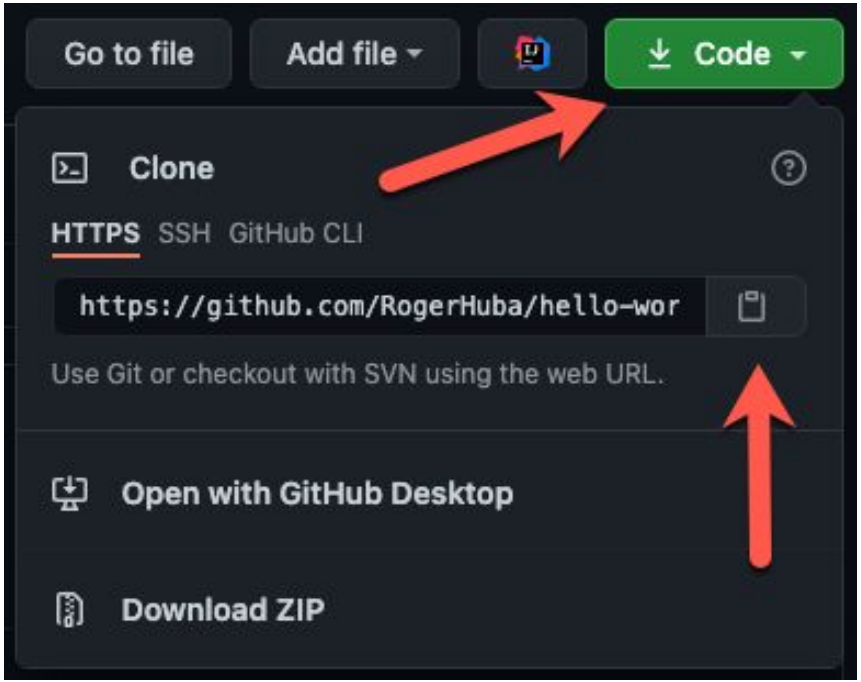


Linking repos

- Congrats! You just made a new repo.
- Now we need to copy this repo onto our computers, and connect the two repos to each other.
- If they’re connected, they can give and receive code from the other repo.
- We’ll do this by **cloning**: from the cloud, to our local machine

Clone that repo

- **1:** Click the big green button that says **Clone or download**. Ensure it says “HTTPS”.
- **2:** Then, copy the URL using the *Copy to clipboard* button shortcut.



Start in your `projects` folder

- Open up your terminal
- Use your **cd** and **ls** commands to navigate into your **projects** directory
  - If it doesn't exist, go to your home directory: **`cd ~`**
  - And make a folder to hold all your repos: **`mkdir projects`**
  - Then change into it: **`cd projects`**
- Check where you are with **pwd** ("print working directory")

```
> pwd
/Users/brookr/projects
```

Using `git clone`

In the **projects** directory:

- Type: **`git clone`**
- Follow that with a <space>
- Paste in the link you just copied
- It should look something like:

```
> git clone https://github.com/brookr/hello-world.git
```

- Ready? Hit Enter.

Using `git clone`

What just happened?

- You made a new folder, **`hello-world`** (see it with **`ls`**)
  - The new folder is a local repo in sync with the one on GitHub
- Change into that directory

```
> cd hello-world
```

- See the URL of the GitHub repo by typing **git remote -v**

```
> git remote -v
```

- It made a directory that has all the files in it you had online

<gitflow:acp>

Add, commit, push

Using `git status`

Now that your files are in your repo, we need to make a commit (take a snapshot of them).

- Review the current status of your files by typing **git status**

```
> git status
```

- It will tell you what files have changed since your last commit.
- Right now, git is paying attention to 1 file (README.md)

Using `git status`

- Edit that file with VS Code
  - Open the current folder in the editor

```
> code .
```

- Make changes
- Save your changes
- ...and see how **`git status`** reports the change

Using `git add`

Next: we need to tell git what changes to commit.

- This is done by typing **`git add`** and then a filename.

```
> git add README.md
```

- This tells git to include these changes in the next snapshot.
  - Think of it as placing items into a scene to photograph
- Type **`git status`** again to see the difference!

Using `git commit`

Finally, take that snapshot!

- Type: **`git commit -m “your message goes here”`**

```
> git commit -m “Adds initial greeting to the world”
```

- **git commit** is the shutter-button to take the snapshot
- **-m** specifies the message included with the commit
- Think of the message as being like a photo caption:
  - Why is this moment significant?

Using `git push`

Great! Now it's time to sync this code to your repo on GitHub.

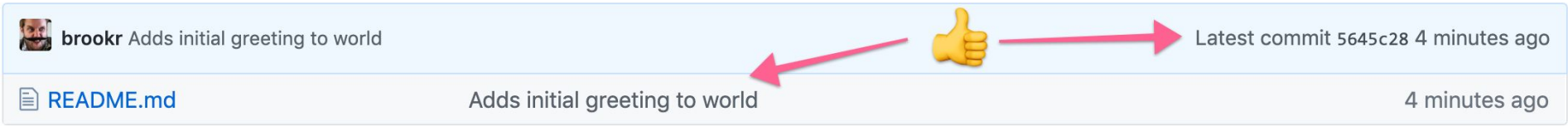
- Type: **`git push origin main`**

```
> git push origin main
Counting objects: 6, done.
…
To git@github.com:brookr/hello-world.git
 * [new branch]      master -> master
```

- **git commit** is what takes the snapshot
- This sends any new commits (the snapshots of your code) to GitHub.
- Go to your repo on GitHub, and look for your files!

Verify on GitHub

In your browser on GitHub you will see the changes that you pushed, as well as the commit message, and the commit id.



Agenda

1. Review of previous class
    - Share your learning
2. Sharing Code
    - Understanding git
    - GitHub
    - git-flow
3. **Deployment**
    - GitHub Pages
    - Demo
4. Assignments
    - Reading & Lab

<deployment>

Look at you, publishing your code

for the world to enjoy!

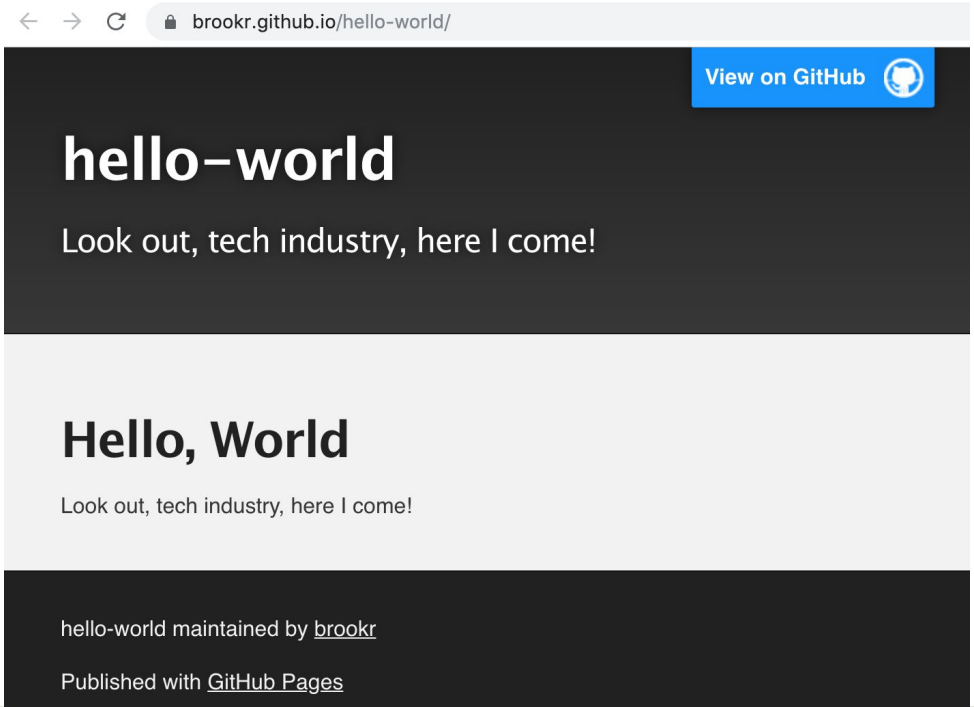
GitHub Pages

- Any repo can be published with GitHub Pages
- Each repo can be styled with a theme
- Whenever you push changes, the published site will update
- That's called "deployment." It turns this markdown...

```
1   # Hello, World
2
3   Look out, tech industry, here I come!
4
```

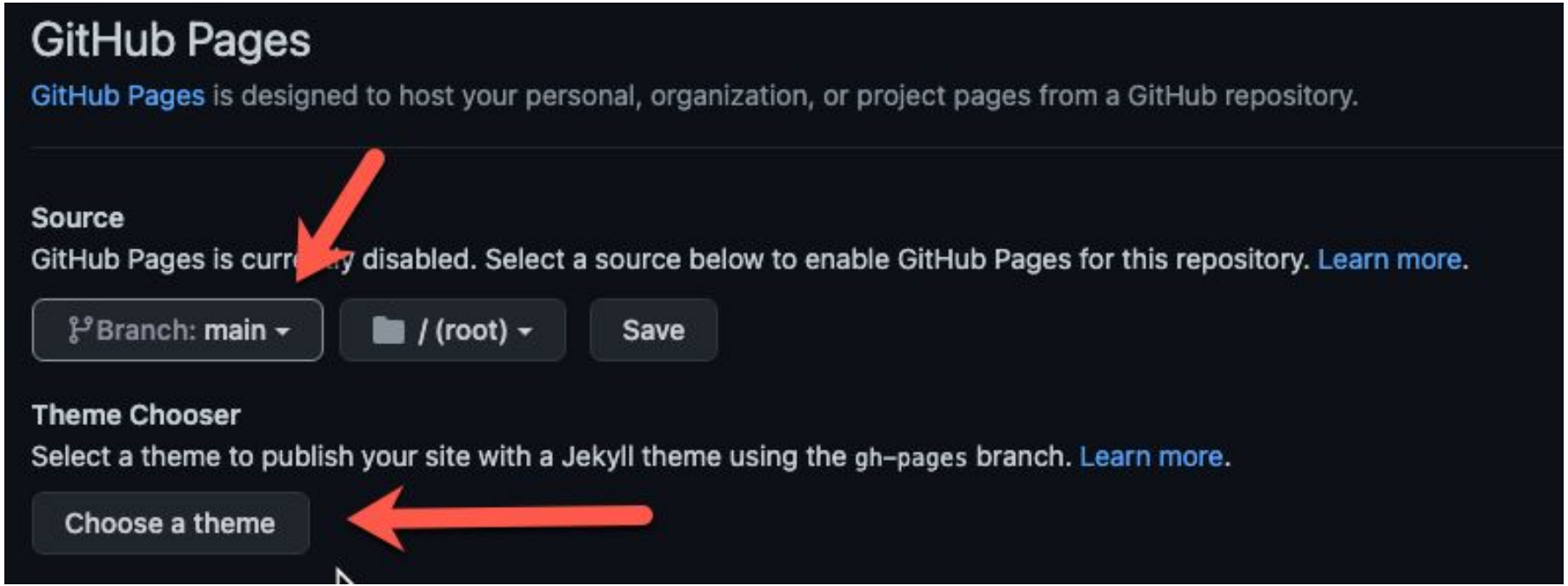
GitHub Pages

- ...into this web page
- Your web page URL is: ***USERNAME*.github.io/*REPO-NAME*/**



GitHub Pages

- Go to your GitHub repo settings, on the left menu click on pages
- Activate publication by selecting a source



Show off!

Congratulations! You have successfully deployed your awesome work on the world-wide web.

Now you can share that link with friends and family, and they can see what you did.

Tweet it! IG it! Fb it! Let the tech world know you are here!