Econ 280 Computation: Git and Github

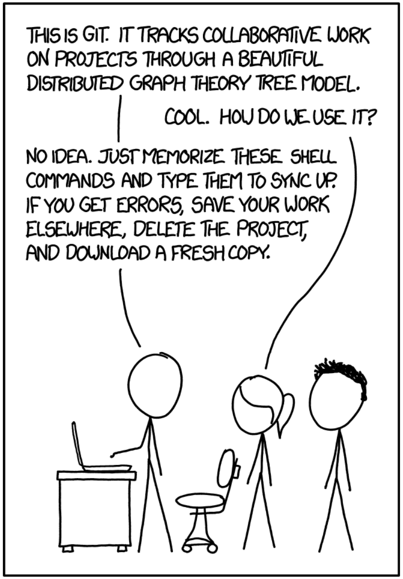
Version Control

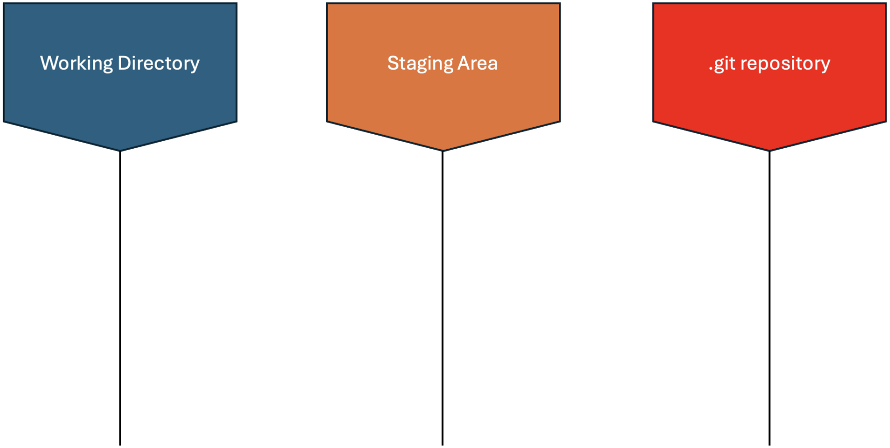
- Most folks in social sciences don't use a proper version control system
- Instead, use something like Dropbox, combined with naming files based on dates
  - `clean_09152024.do` vs. `clean_10012024.do`
  - This date-based naming is a nightmare
- No software on anything you own was written with this system of naming files

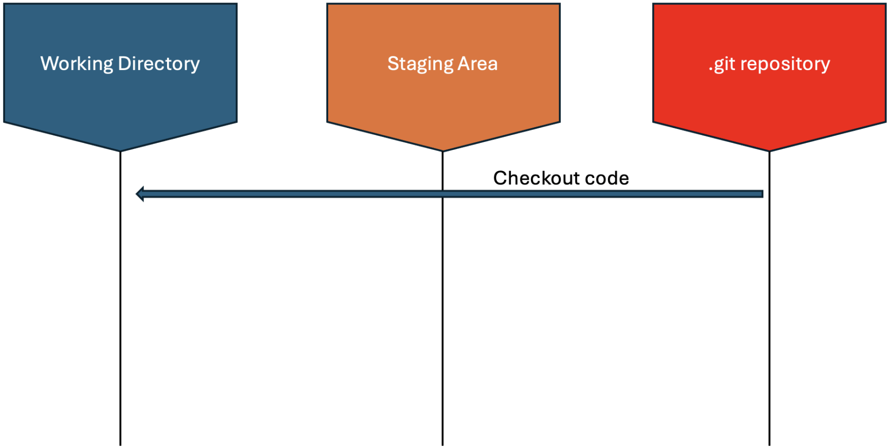
Version Control

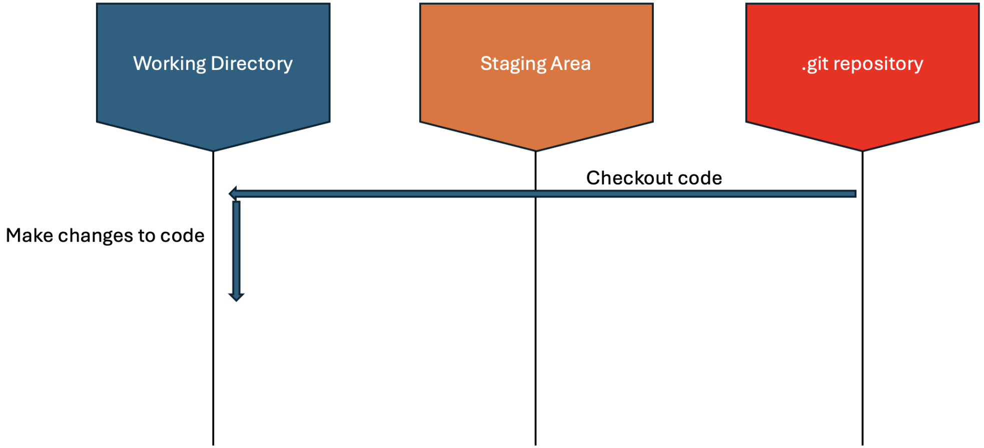
- Today we are going to go through the basics of Git, the most popular version control software currently used
  - Github is an online repository to store your projects, which is useful for collaborating, but you can use Git locally

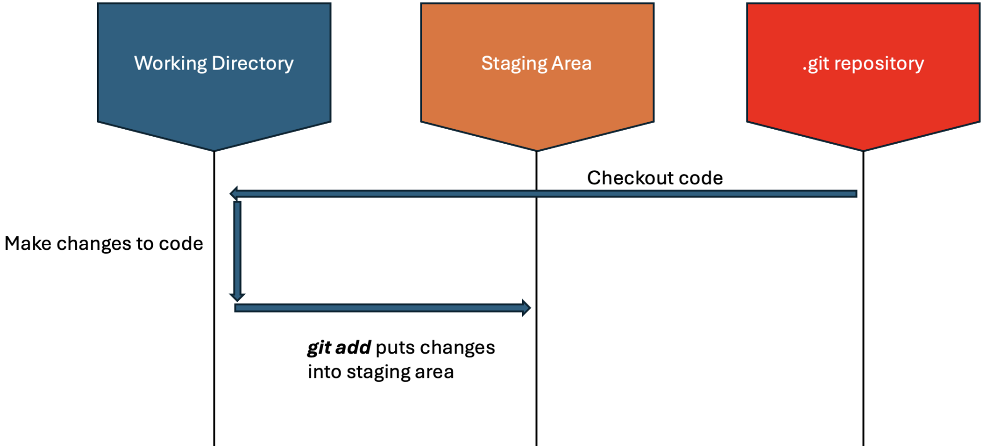
XKCD

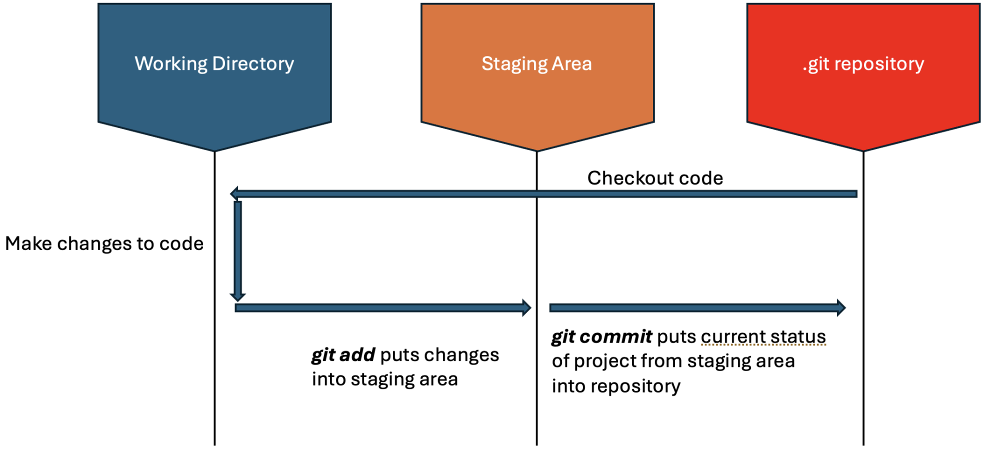

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

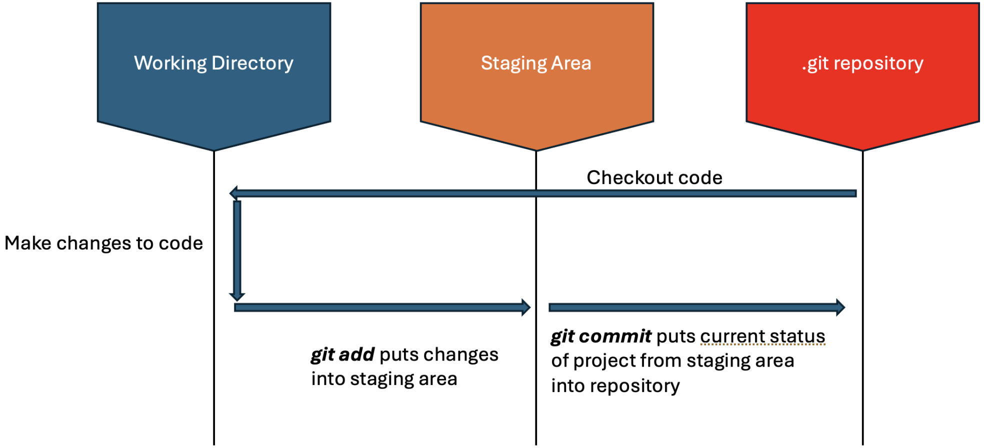

Working Directory
Staging Area
.git repository


Working Directory
Staging Area
.git repository
Checkout code


Working Directory
Staging Area
.git repository
Checkout code
Make changes to code


Working Directory
Staging Area
.git repository
Checkout code
Make changes to code
git add puts changes into staging area


Working Directory
Staging Area
.git repository
Checkout code
Make changes to code
git add puts changes into staging area
git commit puts current status of project from staging area into repository

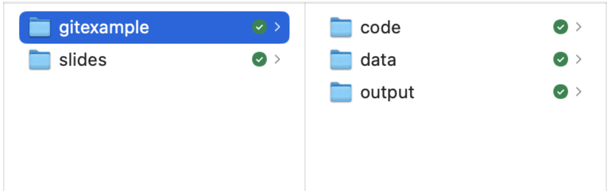

Working Directory
Staging Area
.git repository
Checkout code
Make changes to code
git add puts changes into staging area
git commit puts current status of project from staging area into repository

The git repository will contain two snapshots of the project, one before you checked out the project, and one after you committed any changes

Making a repository

We are going to make a git repository for a subset of the files we created in the Stata Application

Below is the directory `gitexample` that we would like to make into a git repository



Step 1: Install git if not installed

- Is git installed?
- Run this line in terminal/command prompt window

```
git --version
```

- Usually it will be installed

Step 2: Add your details

```
git config --global user.name "Your Name"
git config --global user.email "your_email@example.com"
```

- This will tell git who is making changes to the project
  - If you have a github account, you should just use the same username/email as on your github account (more on this later)

Let's create a repository

- Navigate to the directory you want to make a repository (in terminal/command prompt)

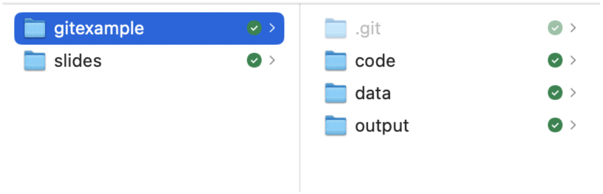
```
cd "/Users/davidarnold/Dropbox/Teaching/ECON280/Git/gitexample"
```

Then type:

```
git init
```

Initialized repository

- It will appear as if nothing has happened
- But if you reveal hidden files (Cmd + Shift + .) on Mac, you will see there is a directory `./git`. This folder will contain all of the version control information



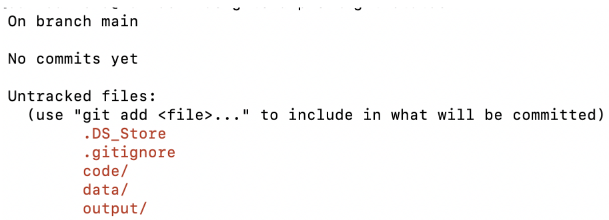
Ignoring files

- Often, not a good idea to use version control on large datasets
- You can tell git to ignore certain files by creating a `.gitignore` file (you can do this in any text editor)
  - Save the file in the root directory of your project and name it `.gitignore`
- The code below will ignore csv and dta file

```
*.dta
*.csv
```

git status

- `git status`



```
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .DS_Store
        .gitignore
        code/
        data/
        output/
```

- We have not added any files to the staging area, so everything in untracked, which also implies we have not committed anything

git add/commit

- `git add --all` will add all the files in the directory to the staging area (except those that are specified as ignored)
- `git commit -m "First committ, project creation"` will commit the changes to the git repository
  - You have now taken a snapshot of the current files in the staging area. This will allow you to revert to this snapshot in the future

Additional Changes

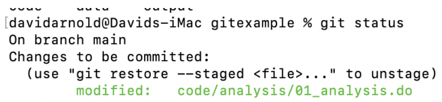
- Now let's make a change. For the purpose of this illustration, we are just going to change the title of the figure in `code/01_analysis.do`

Additional Changes

- Now let's make a change. For the purpose of this illustration, we are just going to change the title of the figure in `code/01_analysis.do`
- Next, add the change to the staging area by typing `git add --all`

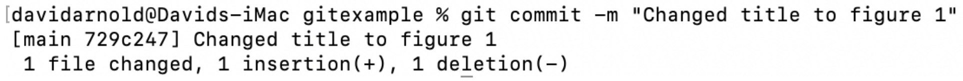
Additional Changes

- Now let's make a change. For the purpose of this illustration, we are just going to change the title of the figure in `code/01_analysis.do`
- Next, add the change to the staging area by typing `git add --all`
- Let's check the status with `git status`



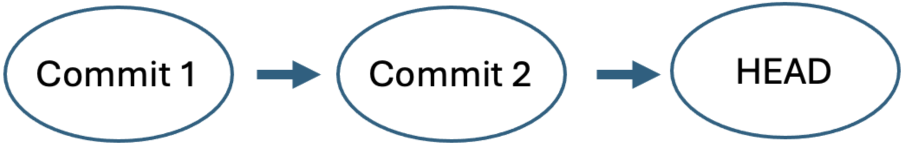
```
davidarnold@Davids-iMac gitexample % git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   code/analysis/01_analysis.do
```

Commit

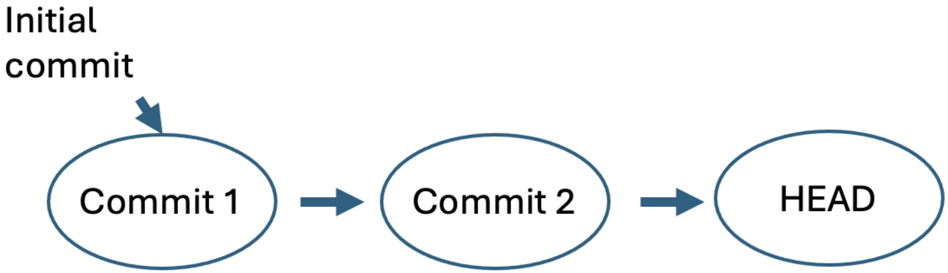


```
[davidarnold@Davids-iMac gitexample % git commit -m "Changed title to figure 1"
[main 729c247] Changed title to figure 1
 1 file changed, 1 insertion(+), 1 deletion(-)
```

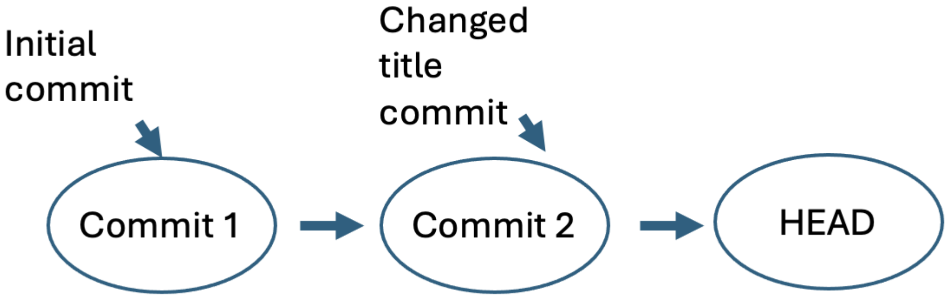
Structure so far



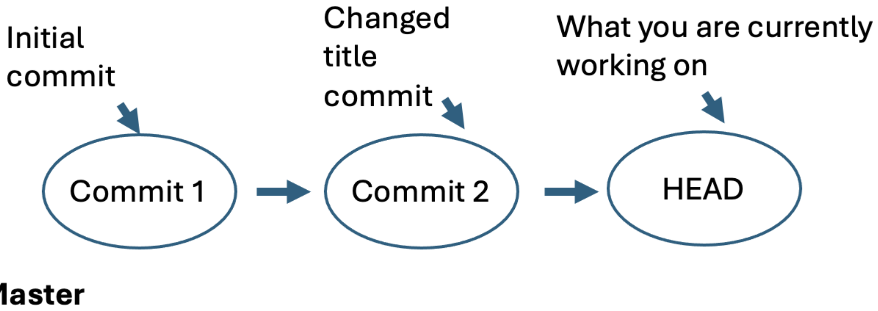
Structure so far



Structure so far



Structure so far



Difference between commits

- You can get a list of commits and identifiers by typing

```
git log
```

- To check what is different between two commits type

```
git diff commit1 commit2
```

- Where you replace commit1 and commit2 with the numbers that identify them

Branches

- A key feature in collaboration with git is branching
  - General use case: check out main branch, add some feature, merge back into main branch when code has been checked
- Let's check out our initial commit, and then create a branch from that commit

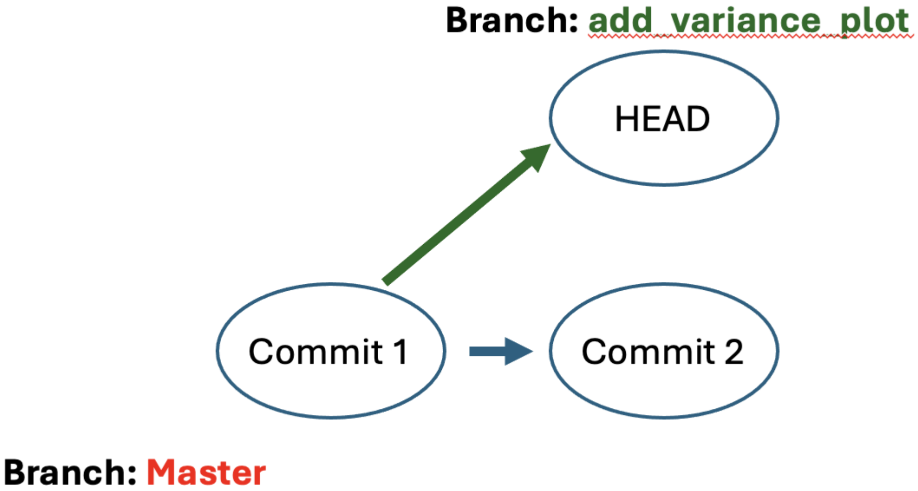
git checkout

- `git checkout <commit>` will revert your working directory to whatever commit you specify in `<commit>`
- We are also going to create a new branch using `-b`

```
git checkout 60aa9 -b "add_variance_plot"
```

- The string `60aa9` identifies the commit. The full hash is 60aa9db88d8c93348d5db11b37e72632834ffe7f, but you generally don't need the entire hash

New Structure



Add plot that shows variance of ROR gaps

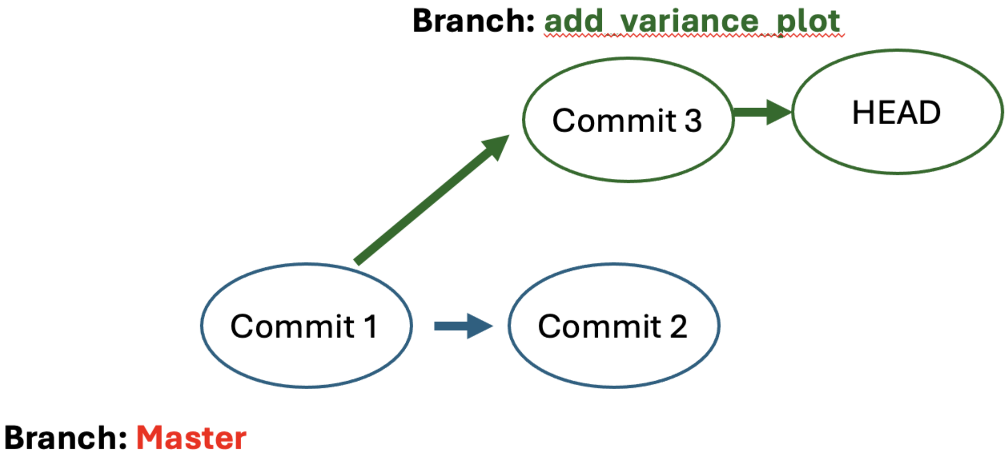
- Add a variance plot to 01_analysis.do

```
sort ror_gap

gen index = _n

gen ub = ror_gap + 1.96*se_ror_gap
gen lb = ror_gap - 1.96*se_ror_gap

twoway scatter ror_gap index, ///
    || rcap ub lb index, ///
    legend(off) ///
    xtitle(off)
```

Add and then commit

- `git add --all` will add the change to the branch `new_variance_plot`
- `git commit -m "Added variance plot"` will commit the change

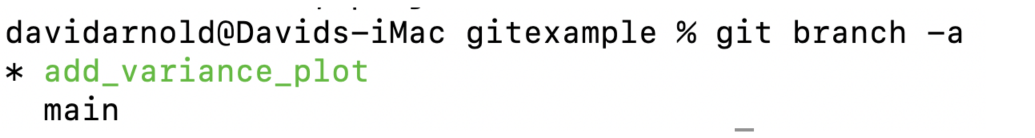
New Structure



Checking branches

- Sometimes useful to make sure you are on the branch you think you are on

```
git branch -a
```



```
davidarnold@Davids-iMac gitexample % git branch -a
* add_variance_plot
  main
```

- The asterisk tells you which branch you are on

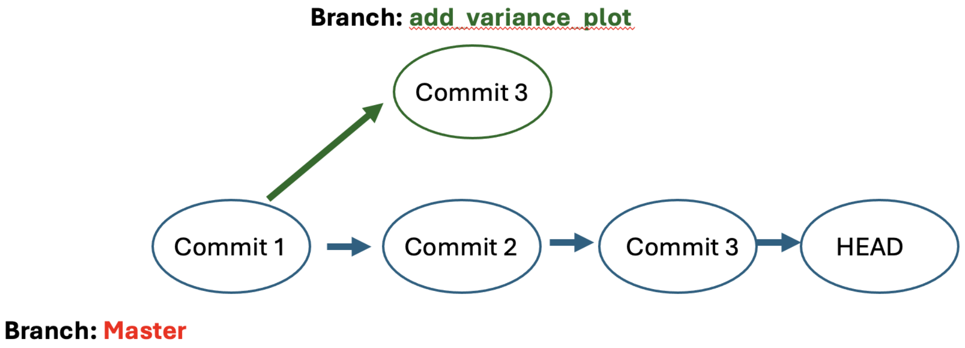
Merging back to main branch

- To add all of the changes you made on the branch, you just
  - Switch back to the main branch
  - Merge changes from previous branch

```
git checkout main
git merge add_variance_plot
```

Current structure

- `git merge` will add the changes made in the feature branch to the main branch

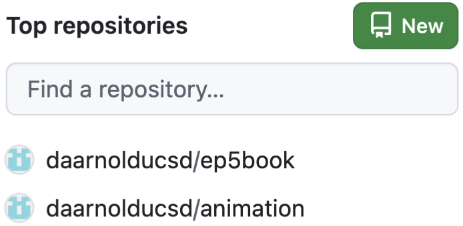


- Everything will be preserved unless you decide to delete the branch by typing `git branch -d add_variance_plot`

Github

- So far everything we have done has been local on our computer
- The most popular cloud hosting service for git is Github
- Next we will create a repository on Github to store this project

Create empty github repository



- The URL for the one I created is `https://github.com/daarnolducsd/econ280example.git`

Authentication

- You need some way to authenticate so that you can make changes to a github repo
- I use HTTPS. This process involves:
  - Click on your Github profile picture > Settings > Developer Settings > Personal Access Tokens
  - You can choose a fine grained token and set all permissions to Read/Write or Read only if that is all that is available
  - Copy your personal access token. It will only be displayed once to you

Communicating with Github

- In terminal tell your local machine about the git repo by typing

```
git remote set-url origin
https://username:personal_token@github.com/username/reponame.git
```

- Because we have specified the personal access token in the url, we can now upload our local files to this repo

Push

- Push sends files from your local machine to the cloud repository

```
git push --set-upstream origin main
```

- `git push` – upload your local git repository to the remote repository
- `--set-upstream` specifies that your local branch will be the upstream branch for your current local branch (you often see `git push origin main`) which means to push your main branch to the origin, which is the remote repository

Pull

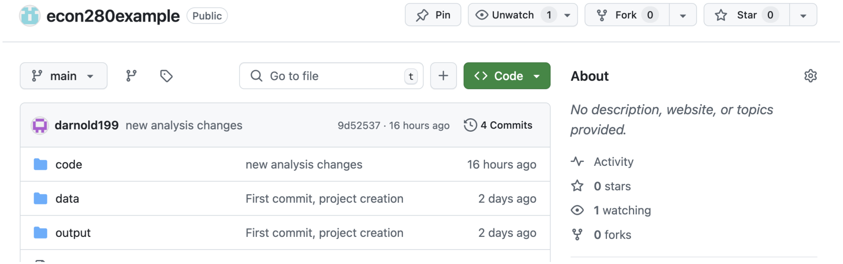
- You can also `pull` changes from the repository

```
git pull
```

- If you are collaborating with others you will want to pull their recent commits so that you are up to date

Git Clone

- You may be interested in downloading someone else's repository
- You can use `git clone` to accomplish this
- Go to the repository you want to clone and select `<code>` button


econ280example Public

darnold199 new analysis changes 9d52537 · 16 hours ago 4 Commits

code	new analysis changes	16 hours ago
data	First commit, project creation	2 days ago
output	First commit, project creation	2 days ago

No description, website, or topics provided.

Activity
0 stars
1 watching
0 forks

- Copy HTTPS url and type `git clone <copied url>` into terminal

Homework

- If you don't have one, make a Github account
- Make sure you can clone the repo from today's class
- In a future assignment, I will have you upload your own github repo