

GitHub Beginner Guide (See Appendix A for Quick Reference)

1. What is GitHub?

GitHub is an open-source repository hosting service. It is a version control, source code configuration, and project collaboration tool. If you use Google Drive, GitHub is like a drive for code.

2. Why GitHub?

You can use GitHub to track changes of your coding history, easily collaborate with your team, contribute to great programs, etc. It can also be used as a version control system.

3. Create Your First Repo

Repo is the abbreviation for repository. To integrate GitHub into your workflow, you need to first create a repo and connect that repo with your local works. There are many ways to establish this connection and the steps can be done in different orders.

However, as a beginner, I suggest the following way and order to start with:

3.0 Create the repo first **BEFORE** working with any code (including copying any starter code)

3.1 Create an empty repo on GitHub

Put in the name of your repo. The name should be easier for you to understand and recognize in the future. If you are working with a lab in your class, you can name it as “laboo_yourgithubID”.

Put in the description of your repo. Again, if it is a lab in your class, you can say “This is a repo for CS16 laboo”. To be more specific, you can say “This lab is an introduction to while loops and for loops”.

Choose if you want your repo to be public or private. Unless otherwise noted, choose private if you are working with a lab in your class.


If this is the very first time you are working with GitHub, I suggest you do not check the box for “initialize with a README” as it may confuse you.


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner

Repository name *

 ucsb-cs16-s20-nichols

 / create_repo_demo 

Great repository names are short and memorable. Need inspiration? How about [shiny-robot](#)?

Description (optional)

☐ Public

Anyone can see this repository. You choose who can commit.

☒ Private

You choose who can see and commit to this repository.

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

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer.

Add .gitignore: **None** | Add a license: **None** ⓘ

Create repository

Click “Create repository” and you will see the following page:

ucsb-cs16-s20-nichols / create_repo_demo Private Watch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Give access to the people you work with

You should give access to the collaborators and teams you need to work with.

Add teams and collaborators

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

Set up in Desktop or HTTPS SSH

https://github.com/ucsb-cs16-s20-nichols/create_repo_demo.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 "# create_repo_demo" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/ucsb-cs16-s20-nichols/create_repo_demo.git
git push -u origin master
```

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

```
git remote add origin https://github.com/ucsb-cs16-s20-nichols/create_repo_demo.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

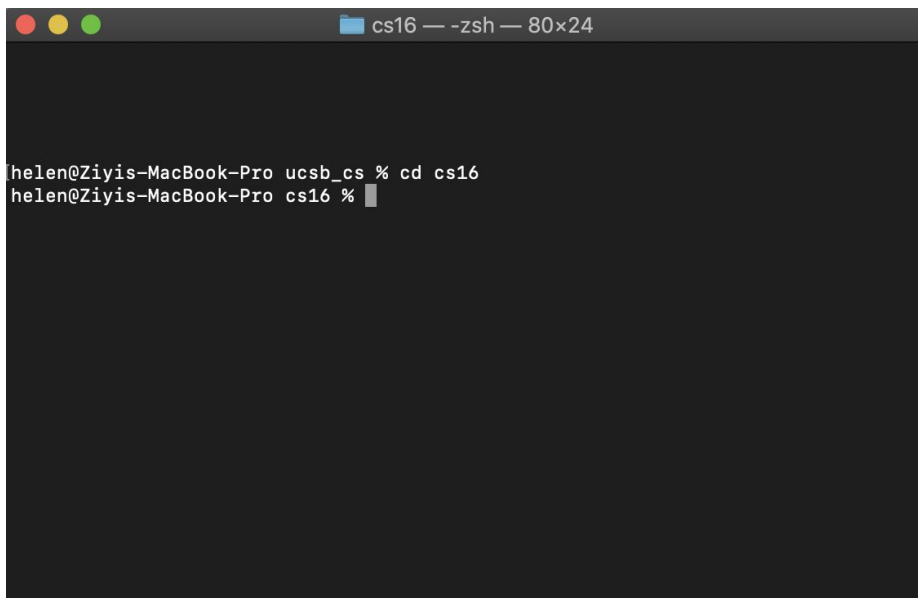
Import code

3.2 Clone your repo into your local working directory

This step will help you establish the connection between your local machine and the remote repo. “Clone” has a meaning of copy in this context. Your repo will be copied as a folder to your local working directory.

Go to (“cd” into) your current working directory. Remember that your repo will be copied as a folder, so you do NOT need to make a new directory for that repo before cloning. There are ways that you can create the directory first then clone the repo, but for now, let’s do with the simplest way.

To be clear, for example, if you are working with laboo in CS16 and you just create a repo called “laboo_myname” on GitHub, you will want to “cd” into the “CS16” directory on your local machine (NOT the “CS16/laboo” directory).

A terminal window titled "cs16 — -zsh — 80x24" is shown. The prompt is "helen@Ziyis-MacBook-Pro ucsb_cs %". The user has entered "cd cs16" and the prompt has changed to "helen@Ziyis-MacBook-Pro cs16 %".

```
helen@Ziyis-MacBook-Pro ucsb_cs % cd cs16
helen@Ziyis-MacBook-Pro cs16 %
```

While you are at the directory, type “git clone”, do not hit enter yet. Go to your repo page, copy the address of your repo. For the very first time, you want the “http” address.

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

Set up in Desktop

 or

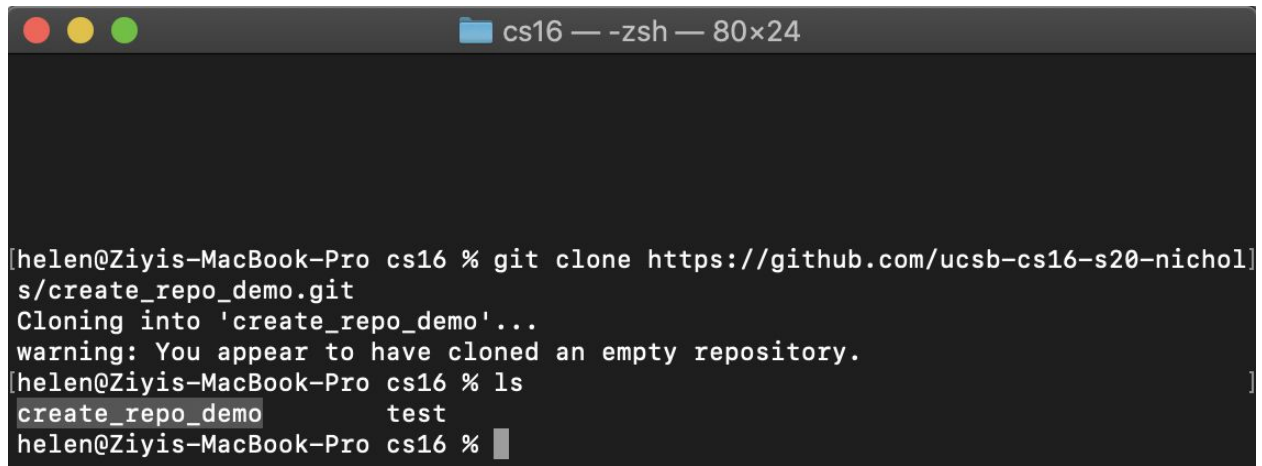
HTTPS

SSH

https://github.com/ucsb-cs16-s20-nichols/create_repo_demo.git

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

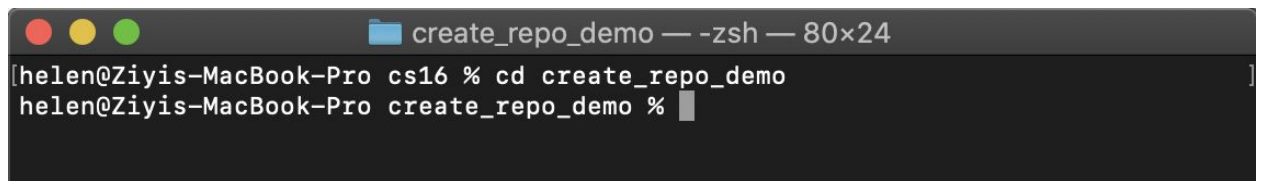
Paste the address after “git clone” in your command line.

A terminal window titled 'cs16 — -zsh — 80x24' on a macOS system. The user 'helen@Ziyis-MacBook-Pro' is in the 'cs16' directory. They run 'git clone https://github.com/ucsb-cs16-s20-nichol...s/create_repo_demo.git'. The output shows 'Cloning into 'create_repo_demo'...' and a warning: 'warning: You appear to have cloned an empty repository.' Then they run 'ls', which shows 'create_repo_demo' and 'test' directories. The prompt returns to 'helen@Ziyis-MacBook-Pro cs16 %' with a cursor.

```
[helen@Ziyis-MacBook-Pro cs16 % git clone https://github.com/ucsb-cs16-s20-nichol
s/create_repo_demo.git
Cloning into 'create_repo_demo'...
warning: You appear to have cloned an empty repository.
[helen@Ziyis-MacBook-Pro cs16 % ls
create_repo_demo  test
helen@Ziyis-MacBook-Pro cs16 % ]
```

Do a “ls” to see if you get the repo locally.

3.3 cd into the repo and start working with code

A terminal window titled 'create_repo_demo — -zsh — 80x24' on a macOS system. The user 'helen@Ziyis-MacBook-Pro' is in the 'cs16' directory. They run 'cd create_repo_demo'. The prompt changes to 'helen@Ziyis-MacBook-Pro create_repo_demo %' with a cursor.

```
[helen@Ziyis-MacBook-Pro cs16 % cd create_repo_demo
helen@Ziyis-MacBook-Pro create_repo_demo % ]
```

At this time, you can *start working with the code*.

If your instructor provides any files, you can copy those files into your directory at this time. Refer to other instructions if you do not know how to do that.

If no files are provided, you can create new files in your directory now (ex. “vim newfile.cpp”).

Use your editor to work with your code locally.

Now, you have the code on your local machine and you have the connection between your GitHub repo and your local directory. It is time to push (means upload) your code to GitHub.

4. Push code to GitHub

4.1 Check you GitHub repo status

In your command line, do a “git status”.

```
helen@Ziyis-MacBook-Pro create_repo_demo % git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .DS_Store
        README.md
        backslash.cpp
        starC.cpp
        starL.cpp
        starT.cpp
        starZ.cpp

nothing added to commit but untracked files present (use "git add" to track)
```

It shows you several information here. “On branch master” shows your current working branch on GitHub. We will not discuss the idea of branch here. We will just work on the “master” branch for now.

In the untracked files, it lists all the files that you have not put into the “connection” yet, including all the new files you’ve created. In this case, Simple.cpp and WordCount.cpp are all new files.

4.2 Add the files to the “connection”

```
helen@Ziyis-MacBook-Pro create_repo_demo % git add backslash.cpp starC.cpp ]
helen@Ziyis-MacBook-Pro create_repo_demo % git status ]
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   backslash.cpp
        new file:   starC.cpp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .DS_Store
        README.md
        starL.cpp
        starT.cpp
        starZ.cpp
```

Use “git add filename.cpp” to stage the files you want push to GitHub to the “connection”

Do another “git status” and you will see the files you just add in “Changes to be committed” and are ready to be committed.

4.3 Commit changes locally

Use “git commit -m “xxxxxxx”” to commit your changes. Put the messages in “xxxxxx” to explain what you’ve just done. For example, in this case, we can say we create new files and set up the starter code.

```
helen@Ziyis-MacBook-Pro create_repo_demo % git commit -m "create new files and set up starter code"
[master (root-commit) b198471] create new files and set up starter code
 2 files changed, 161 insertions(+)
 create mode 100755 backslash.cpp
 create mode 100755 starC.cpp
```

Do not put in random messages. Try to practice using accurate and explanatory messages so that it’s easier for you and others to understand. It will become helpful later when you are working with larger programs.

With “git commit”, the changes made since the last commit in the files that staged by “git add” are saved to the local repo, but not the remote repo on the server.

4.4 Push the changes to the remote repo (on GitHub in this case)

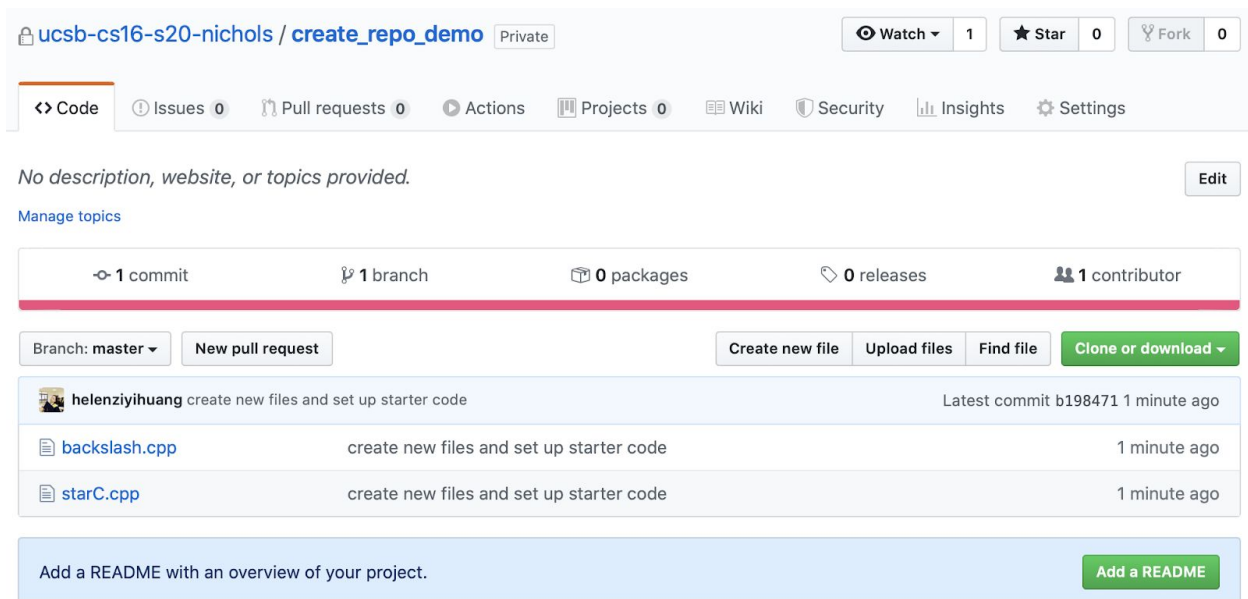
```
helen@Ziyis-MacBook-Pro create_repo_demo % git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.87 KiB | 1.87 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/ucsb-cs16-s20-nichols/create_repo_demo.git
 * [new branch]      master -> master
```

Use “git push origin master” to push your changes to the “master” branch.

It will ask you to enter your GitHub username and password. (You can avoid repeating this by setting up ssh keys. Refer to other instructions to set up that once you are familiar with the steps in this guide).

With “git push”, it pushed the changes from your local repo onto the remote repo.

After “git push”, you are almost done! Go to your GitHub repo page and refresh it, if you see all the files that you just add, commit and push, you are good!



Click on the “commit” tab at the upper left corner, you can review your commit history and see the difference in code between your commits.

4.5 Make changes to your code

Now, you just get familiar with the basic workflow of “git add, git commit, git push”. You can continue working with your code.

Once you have some changes in your code, you can do another “git status”. This time, your changed files will be listed in the “Changes NOT staged for commit” section.

```
helen@Ziyis-MacBook-Pro create_repo_demo % git status
On branch master
Your branch is up to date with 'origin/master'.

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:   starC.cpp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .DS_Store
        README.md
        starL.cpp
        starT.cpp
        starZ.cpp

no changes added to commit (use "git add" and/or "git commit -a")
```

Follow the previous steps to:

“git add”: add/stage the files so that it will be in the “Changes to be committed” section

“git commit”: add messages so that the changes are saved to the local repo

“git push”: push the changes to the remote repo on GitHub

4.6 Some clarification about different files

The new files you create will be in the “Untracked files” section and need to go to the “Changes to be committed” section by “git add ...”

The changes you make in existing files will be in the “Changes NOT staged for commit” section and need to go to the “Changes to be committed” section by “git add ...”

Only the files in the “Changes to be committed” section will go with the messages you enter in “git commit -m “xxx”” and be reflected on the GitHub repo page with “git push origin master”.

You may be wondering what is the “.DS_Store” file here. We will not discuss it in detail here. For now, you can just leave it in the untracked files. All the files that you don’t want to show on GitHub repo page, you can leave them locally in the untracked files.

5. Get code from GitHub

One great thing using GitHub is that you can access your code in any machine and start working on it whenever you want. There are two cases where you want to get your code from GitHub. First, you go to a new machine that never has this repo on it. Second, your machine already has this repo on it; however, you've made changes in this repo on other machines and you want those changes to be copied to your current machine.

5.1 Clone you entire repo

If you are on a new machine and this machine never has your repo on it, you can simply clone your entire repo onto this machine by “git clone ...”. Follow step 3.2, they are the same.

For example, you create the repo on your laptop with step 1-4 and you go to your lab section and want to continue working on your code on computers in the lab/CSIL. You can clone your repo on GitHub to the CSIL machine with this step.

5.2 Pull your changes into your local repo

If you already have this repo locally on your machine but has made changes elsewhere... For example, you create the repo on your laptop, and make some changes on CSIL machines and push the changes to GitHub using CSIL machines. You go back home and want to continue working on your laptop.

BEFORE making any changes, you will need to pull (copy) the new changes on GitHub to your laptop first so your code is up-to-date before continuing to work.

Use “git pull origin master” to get the new changes.

```
helen@Ziyis-MacBook-Pro create_repo_demo % git pull origin master
From https://github.com/helenziyihuang/create_repo_demo
* branch      master      -> FETCH_HEAD
Already up to date.
```

If no changes were made, you will see the message “Already up to date”.

5.2.1 Pull before anything

Note: It can be a good practice to do a “git pull” every time you open a new terminal and go to the directory before you start to work and change anything. Though you might not change anything, it can help you check so that things will not mess up.

Why pull before anything (2 scenarios):

If you make any changes locally and then do “git pull”, all the changes you make locally will go away. “git pull” will ignore your local change and put the code on GitHub to your local repo.

If you make any changes locally and try to push it to GitHub, but there are changes on GitHub that haven't in your local repo yet, GitHub will forbid you from pushing new changes.

Appendix A. Basic git Commands Summary

This table here summarizes all the basic git commands that are introduced in this beginner guide.

Command	Followed by	Explanation
git clone	+ repo address	Clone the remote repo to local directory
git status	/	Check the status of files
git add	+ files	Stages the files that are ready to be committed
git commit -m	+ “messages”	Add a message to explain what have been done for that commit
git push	+ origin master	Push the local changes to the remote repo
git pull	+ origin master	Get the latest version from GitHub to local machines