Getting Set Up with Git

CS/SE 3340

Dr. Karen Mazidi

Git

Git is a free, open-source, cross-platform version control system. Version control systems let you save your current code while retaining copies of previous code in case you need to roll back an update. A **repository** is a location that saves files for a given project. When you **commit** a change, a new version is saved but the older versions still exist. When programmers work together on projects they each may fork a branch of the repo to work on their own part of the project. At some point the branches will need to be merged so that all changes are reflected in the repo.

Git is most often used for code but it could be used for manuscripts, or any project that has incremental changes that you want to keep track of. By the way, git was created by Linus Torvalds and he called it git because he said he names all his projects after himself. “git” is British slang for a stupid or unpleasant person.

There are many version control systems but git is probably the most commonly used. The first thing to do is install Git.

Windows

1. Download from https://git-scm.com/
2. When installing, hit next on all screens except this one: “Adjusting your PATH environment”, choose “Use Git and optional Unix tools from the Windows Command Prompt”

Mac

1. Download from https://git-scm.com/ and follow the instructions.

Ubuntu

```
$sudo apt install git
```

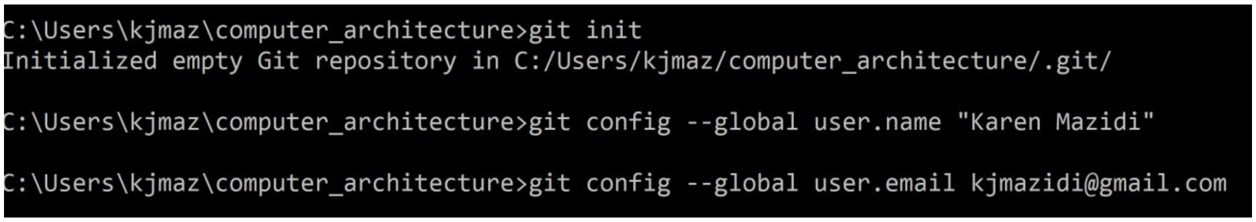
Start a git repository

Start a new terminal and navigate to the folder that holds your code. For my example, my directory is named computer_architecture. I will use this folder to store things I want to upload to my github.

First we need to set up a repo and configure our user information. Replace my info with your own for the second two lines:

```
$ git init
$ git config –global user.name "Karen Mazidi"
$ git config – global user.email kjmazidi@gmail.com
```

The "git init" command is something we do once per project. The other two commands are something we only do once. From now on, git knows who we are. Here's what this looked like on my Windows computer:



```
C:\Users\kjmaz\computer_architecture>git init
Initialized empty Git repository in C:/Users/kjmaz/computer_architecture/.git/

C:\Users\kjmaz\computer_architecture>git config --global user.name "Karen Mazidi"

C:\Users\kjmaz\computer_architecture>git config --global user.email kjmazidi@gmail.com
```

What did we just do? The last two lines are just storing our info. The first line creates a Git repository. The repository will store all the changes to the files in the folder (when we ask it to). We may not want to keep track of all files in the folder, however. In that case we can create a .gitignore file which lists things git should ignore.

You can easily create the .gitignore with "vim .gitignore". The dot in front of the file name is important. The .gitignore is not required if you don't want to ignore anything.

Before commit, it's usually a good idea to check the status:

```
$ git status
```

Git will tell you what files have not been committed yet.

Commit

To commit changes to the local repo, use the following commands:

```
$ git add --all .
$ git commit -m "First commit"
```

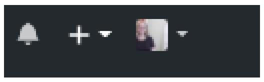
Pay attention to the first line. That is two dashes in front of all and a period after. The messages you get back explain what happened. The add command just gathered everything and the commit command made the commit, marking it with the label “First commit”.

Pushing to GitHub

GitHub is a free repo hosting service that lets you share your code with others. You can have private repos if you have a paid account but for the free account, your repos are public. It is a good idea for CS and SE students to have a GitHub where you can show off what you can do to potential employers.

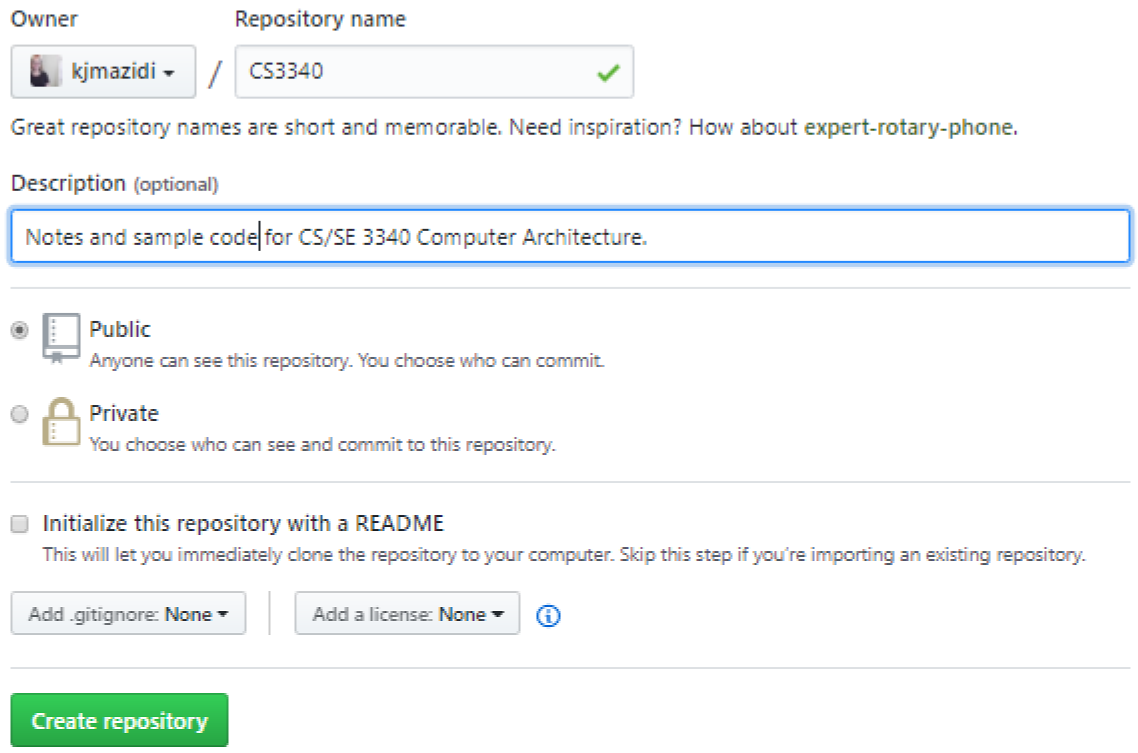
Before you can push your repo to GitHub, you have to have an account. Sign up for a free account at GitHub.com. You can add as much info on your profile as you want the world to know.

Notice the plus sign on the upper right of your github page. It should look something like this:



Click on the arrow to the right of the + and choose “Create new repository”. You will need to give it a name, and indicate whether it is public or private. Uncheck the “initialize with a README” box, and leave the .gitignore option blank.

Your screen should look something like the following screen shot. When you are ready, hit the green Create repository button.



Owner: kjmazidi / Repository name: CS3340

Great repository names are short and memorable. Need inspiration? How about expert-rotary-phone.

Description (optional): Notes and sample code for CS/SE 3340 Computer Architecture.

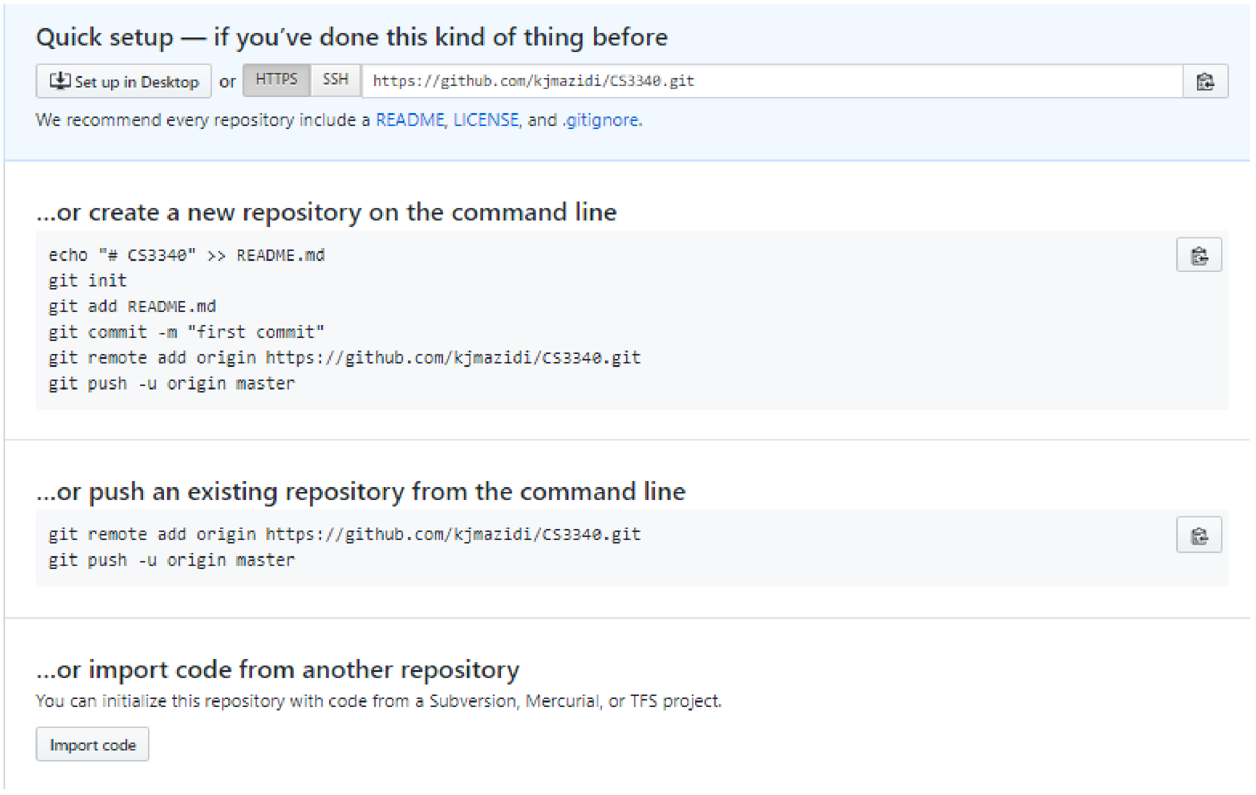
[✓ Public] Anyone can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

The next screen:



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

Set up in Desktop or HTTPS SSH https://github.com/kjmazidi/CS3340.git

We recommend every repository include a README, LICENSE, and .gitignore.

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

```
echo "# CS3340" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/kjmazidi/CS3340.git
git push -u origin master
```

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

```
git remote add origin https://github.com/kjmazidi/CS3340.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.

The HTTPS button should be selected by default and it shows the url to reach your repo.

We have set up a local repo and a spot for it on GitHub. Now we need to connect the two. We will do that back on our local computer at the command line, replacing my info with yours:

```
$ git remote add origin https://github.com/kjmazidi/CS3340.git
$ git push – origin master
```

You don't have to type the url because if you look back at the GitHub, you'll see a copy icon next to your url. When you push, git will ask you for your GitHub username and password, so have that handy.

If everything worked ok, you will see your files on GitHub.com.

Updates

Now that you have everything set up, pushing updates will involve the following steps:

```
$ git status
$ git add --all .
$ git commit -m "update documentation"
$ git push –u origin master
```

There is much more to Git and GitHub of course, but this has covered the basics and you are ready to go.