

Introduction to Git[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Introduction-to-Git)

*Author: Dave Yerrington, Sam Stack*

LEARNING OBJECTIVES[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#LEARNING-OBJECTIVES)

*After this lesson, you will be able to:*

* Use and explain common Git commands, including init, add, commit, push, pull, and clone.
* Distinguish between local and remote repositories.
* Create, copy, and delete repositories locally or on GitHub.
* Clone remote repositories.
* Establish Secure Shell connections to remote repositories.

STUDENT PRE-WORK[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#STUDENT-PRE-WORK)

*Before this lesson, you should already be able to:*

* Show that you've completed [Code Academy: Learn Git](https://www.codecademy.com/learn/learn-git).
* Install [Homebrew](http://brew.sh/).
* Install Git (after installing Homebrew, type "brew install git").
* Set up a GitHub account.

LESSON GUIDE[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#LESSON-GUIDE)

* [Introduction](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#introduction)
* [Git Basics](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#git_basics)
* [GitHub & GitHub Enterprise Accounts](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#gh_ghe_accounts)
* [Creating and Cloning Repos](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#making_cloning)
* [Pulling](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#pulling)
* [Secure Shell(SSH)](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#ssh)
* [Independent Practice](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#independent_practice)
* [Conclusion](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#conclusion)

Git vs. GitHub and Version Control, an Introduction[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Git-vs.-GitHub-and-Version-Control,-an-Introduction)

First things first — Git is not GitHub. This is a common mistake people make.

What is Git?[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#What-is-Git?)

[Git](https://git-scm.com/) is:

* A program you run from the command line.
* A distributed version control system.

Programmers use Git so that they can keep a history of all changes made to their code. This means that they can roll back changes (or switch to older versions) as far back as when they started using Git in their project.

A code base in Git is referred to as a **repository**, or **repo**, for short.

Git was created by [Linus Torvalds](https://en.wikipedia.org/wiki/Linus_Torvalds), the principal developer of Linux.

What is GitHub?[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#What-is-GitHub?)

[GitHub](https://github.com/) is:

* A hosting service for Git repositories.
* A web interface to explore Git repositories.
* A social network of programmers.

Some other points to note:

* We all have individual GitHub accounts and store our code bases there.
* You can follow users and star your favorite projects.
* Developers can access code bases on other public accounts.
* GitHub uses Git.

What is GitHub Enterprise (GHE)?[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#What-is-GitHub-Enterprise-(GHE)?)

[GitHub Enterprise](https://enterprise.github.com/home) Is:

* Professional application of GitHub
* All repository data is stored on private and/or local machines and networks.

Where GitHub is the *public*, 'Social Network' for programming and programmers, Github Enterprise is the *private*, professional application of GitHub. Because, GitHub and GitHub Enterprise have a similar structure and are based off the git language, interacting with the two is is almost identical.

**Additional information on** [**GitHub.com vs GitHub Enterprise**](https://enterprise.github.com/downloads/en/comvsenterprise-082415.pdf)

Over the course of this class we will be interacting with both GitHub and GitHub Enterprise(GHE). You will set up a GHE account for sharing course materials and as a private repo for you to store your own works-in-progress.

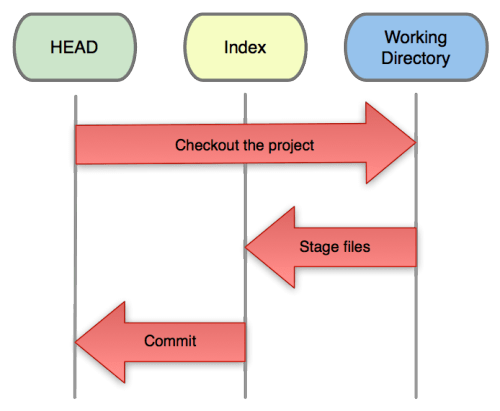
You will also setup a GitHub account as a location for you to host projects and work that you want to exhibit or share.

Can you use Git without GitHub?[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Can-you-use-Git-without-GitHub?)

Think about this quote: “Git is software. GitHub is a company that happens to use Git software.” So yes, you can certainly use Git without GitHub!

Your local repository consists of three "trees" that are maintained by Git.

* **Working Directory**: holds the actual files.
* **Index**: acts as a staging area.
* **HEAD**: points to the last commit you've made.



What about commands?[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#What-about-commands?)

There are also a variety of commands you can use in Git. You can take a look at a list of the available commands by running:

$ git help -a

Even though there are lots of commands, in the course, we will really only need about 10.

Let's Git into it!: Code-Along (15 min)[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Let's-Git-into-it!:-Code-Along-(15-min))

First, create a directory on your Desktop.

$ cd ~/Desktop

$ mkdir hello-world

You can place this directory under Git revision control using the following command:

$ cd hello-world *# dont forget to CD into the folder.*

$ git init

Git will reply:

Initialized empty Git repository in <location>

You've now initialized the working directory.

The .git folder[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#The-.git-folder)

We can look at the contents of this empty folder using this command:

ls -A

We should see that there is now a hidden folder called .git. This is where all of the information about your repository is stored. There is no need for you to make any changes to this folder. You can control all of the Git flow using git commands.

Add a file[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Add-a-file)

Let's create a new file.

$ touch a.txt

If we run git status, we should get:

On branch master

Initial commit

Untracked files:

(use "git add <file>..." to include in what will be committed)

a.txt

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

This means that there is a new, **untracked** file. Next, tell Git to take a snapshot of the contents of all files under the current directory. (Note the .).

$ git add .

This snapshot is now stored in a temporary staging area, which Git calls the "index."

Commit[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Commit)

To permanently store the contents of the index in the repository, (i.e., commit these changes to the "HEAD"), you need to run the following command:

$ git commit -m "Please remember this file at this time"

You should now get:

[master (root-commit) b4faebd] Please remember this file at this time

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 a.txt

Checking the log[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Checking-the-log)

If we want to view the commit history, we can run:

git log

As a result, you should see:

\* b4faebd (HEAD, master) Please remember this file at this time

'Git' on GitHub and GitHub Enterprise:[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#'Git'-on-GitHub-and-GitHub-Enterprise:)

For those of you that have not already set up your GitHub and GitHub Enterprise accounts, lets take a minute to do that now.

Account creation for both is simple. Create a *Username*, and provide an *email* & *password*.

*Keep in mind while you* ***can*** *use the same email, username and password for both accounts, they are* ***wholly separate****.*

[**GitHub**](https://github.com/) *This is yours*

[**GitHub Enterprise for General Assembly**](https://git.generalassemb.ly/join?source=header) *This is ours*

You can use a GitHub account to create a GitHub Enterprise account, but that will be **your** enterprise; in that, you will not be able to use it to access the General Assembly Enterprise GitHub.

If in the futur,e you join another GitHub Enterprise you will also need to create another account for said enterprise.

Making and Cloning Repositories: Code-Along (15 min)[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Making-and-Cloning-Repositories:-Code-Along-(15-min))

When using Git, GitHub and GHE it is common to have your repositories in several locations. Typically when we use GitHub and GHE we will have two repository locations, **Remote** and **Local**.

* **Remote:** - Repositories that are not stored in our current location/machine. Usually where we store the repo.
* **Local:** - Repositories that are stored on our current machine. Usually where we work on the repo.

**Let's do this together:**

1. Go to your GitHub account.
2. On the right-hand side, hit the + button for New repository.
3. Name your repository hello-world.
4. **Initialize this repository with a README**. (Now we can git pull).
5. Click the big, green Create Repository button.

We now need to connect our local Git repository with our newly created remote repository on GitHub. We have to add a "remote" repository, an address where we can send our local files to be stored.

On the right-hand side of your GitHub there should be a green 'Clone or download' button. This button should reveal a tiny window with a URL. Copy the provided URL, which is the path to this remote repo.

*Make sure you changed directories into hello-world prior to running this*

git remote add origin git@github.com:github-name/hello-world.git

*At this point you* ***may*** *be prompted for a password, especially when using GHE*

Pushing to GitHub[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Pushing-to-GitHub)

In order to send files from our local machine to our remote repository on GitHub, we need to use the command git push. However, you also need to add the name of the remote repo — in this case, we called it origin — and the name of the branch, in this case master.

git push origin master

This should fail because of new files in the remote repo.

Pulling from GitHub[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Pulling-from-GitHub)

Just as we added the README.md in our repo, we need to first pull that file to our local repository to check that we don't have a "conflict."

git pull origin master

git status

git add .

git commit -m "README.md"

Once we've done this, you should see the README file on your computer as it has been added to your local HEAD.

ls

Open up the README and type some kind of modification and/or addition then save it.

git add .

git commit -m"Updated README"

Now you can push your changes to your remote on GitHub.

git push origin master

Refresh your GitHub web page, and the files should appear.

Using the git remote add method is useful for taking an existing local repository and creating a new remote one (putting it on GitHub).

Independent Practice (10 min)[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Independent-Practice-(10-min))

Repeat the steps above, starting from the creation of the remote repo, to create a *local* and *remote* repository with your GitHub Enterprise account.

Cloning your first repository[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Cloning-your-first-repository)

Alternatively, you can clone this repo from your GitHub link to have it automatically configured in a new directory.

git clone https://github.com/github-username/hello-world.git

Now that everyone has a repository on GitHub, let's clone one!

Cloning allows you to make a local copy of a remote repository.

Navigate back to your Desktop, and **delete your hello-world repository**.

cd ~/Desktop

rm -rf hello-world

Now, ask the person sitting next to you for his or her GitHub name and navigate to that repository on GitHub.

https://www.github.com/<github-username>/hello-world

On the right-hand side, you will see the green button 'Clone or download'.

Clone that repo![¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Clone-that-repo!)

To retrieve the contents of the repo, all you need to do is:

$ git clone git@github.com/<partners-github-username>/hello-world.git

Git should reply:

Cloning into 'hello-world'...

remote: Counting objects: 3, **done**.

remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0

Receiving objects: 100% (3/3), **done**.

Checking connectivity... **done**.

You've now cloned your first repository! The process will be the same with GitHub Enterprise.

Cloning is must useful when there is a repo that exists remotely that we want on our local machines.

Create a Pull Request on GitHub: Code-Along (10 min)[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Create-a-Pull-Request-on-GitHub:-Code-Along-(10-min))

What's a "Pull Request"?[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#What's-a-"Pull-Request"?)

Often times when we are working on a project with other collaborators we all can't work on the same repo at the same time so we have these things called *branches*. With branches we can create a branch off of the main repo or "Master Branch", perform our work whether that be additions or alterations and then merge the changes we made on *our* branch back into the master.

You can probably think of a few ways that multiple people trying to merge their work can get messy. Fortunately, we have Pull Requests as means of queing and validating merges. Rather than having your branch merge automatically, your request will go through an administrator who can review your changes and additions and approve or deny your request.

*Even though we are trying to* ***push*** *our information into the master, it is called a pull request because we are making a request to the administrator to* ***pull*** *or branch into the master.*

Creating a Pull Request[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Creating-a-Pull-Request)

Before you can open a pull request, you must create a branch in your local repository, commit to it, and push that branch to a repository or fork on GitHub.

1. Visit the repository you've pushed to.
2. Click "Branch:master" on the left-hand side and create a branch called 'branch-edits' by typing in the box and hitting enter. This should put you on your new branches page.
3. Make sure you're on your new branch and in GitHub create a new '.md' or '.txt' file using the 'Create new file' button. Call it what you like and place some text in it.
4. Click the "Compare & pull request" button in the repository pr.
5. You'll land right onto the compare page. Next, you can click "Edit" at the top to pick a new branch to merge to using the "Head Branch" dropdown menu.
6. Select the target branch that your branch should be merged to using the "Base Branch" dropdown menu.
7. Review your proposed change.
8. Hit "Click to create a pull request" for this comparison.
9. Enter a title and description for your pull request.
10. Click "Send pull request."

As you are working on your own repo in your own GitHub you will be able to merge the branch at your leisure.

GitHub Example Project[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#GitHub-Example-Project)

A company is building onto their website and adding a few new features. They are assigning a team of engineers to complete the task.

Engineer 1 is responsible for redoing the home page. Engineer 2 is responsible for reworking content pages 1 & 2 Engineer 3 is responsible for reworking content pages 3 & 4

In this situation the master branch would probably be a copy of the files and scripts for the original website. As this is the master copy it is best not to make any edits to it until those changes have been vetted.

The engineer team starts by making branch off of the master called 'project\_1'.

From 'project\_1' the team of engineers would probably create their own branches to perform their tasks in.

* Engineer 1 : 'preoject\_1\_homepage'
* Engineer 2 : 'preoject\_1\_content1\_2'
* Engineer 3 : 'preoject\_1\_content3\_4'

Once each engineer completed their task they would merge the changes from their branch into 'project\_1'.

Now that all of the individual work has been compiled they can test the website and make sure that there are no bugs or issues then merge 'project\_1' back into the master copy.

By now you have have noticed that you are being prompted quite a bit for your password when interacting with GitHub or GitHub Enterprise via Git. There are two ways to work around this.

[**Catching your password in Git**](https://help.github.com/articles/caching-your-github-password-in-git/)

or

**Using SSH and SSH Agent** (*recommended*)  
You can use the guide in the [SSH Setup Guide](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/SSH-setup.md) or those available on github

* [Working with SSH key passphrases](https://help.github.com/articles/working-with-ssh-key-passphrases/)
* [Generating a new SSH key and adding it to the ssh-agent](https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/)
* [Adding a new SSH key to your GitHub account](https://help.github.com/articles/adding-a-new-ssh-key-to-your-github-account/)

Secure Shell (SSH)[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Secure-Shell-(SSH))

SSH, or Secure Shell, is a common means of adding an additional layer of security. Simply put, the SSH key is used to establish authenticity between a client and a server so that a secure connection or "tunnel" can be formed. This can be useful for secure file sharing or remote application access.

How SSH works[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#How-SSH-works)

The SSH process at a high level is relatively simple.

1. Client makes a request to the server.
2. Server responds asking for authentication.
3. Client provides authentication.
4. If authentication is correct, a connection is established.

**Note**: You will need to complete which ever process you choose twice. Once for your GitHub account and once for your GitHub Enterprise account.

**Note:** Setting up the SSH Agent is critical if you don't want to be prompted for a password everytime you push or pull via SSH.

Additional SSH Troubleshooting:[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Additional-SSH-Troubleshooting:)

**Access Denied**  
If you receive a message about "access denied," please notify an instructor or you can read [these instructions for diagnosing the issue.](https://help.github.com/articles/error-permission-denied-publickey/)

**HTTPS to SSH switch**  
If you're switching from HTTPS to SSH, you'll now need to update your remote repository URLs. For more information, see [Changing a remote's URL.](https://help.github.com/articles/changing-a-remote-s-url/)

Assess: Independent Practice (10 min)[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Assess:-Independent-Practice-(10-min))

* Show a partner how to use the following commands: init, add, commit, push, pull, and clone.

Conclusion[¶](http://localhost:8888/nbconvert/html/Desktop/SCHOOL_WORK/Week%201/your-development-environment-master/extra-materials/Git-%26-GitHub/Git-and-GitHub.ipynb?download=false#Conclusion)

Feel comfortable with Git, GitHub and GitHub Enterprise? As we'll be using them in alot a lot of our coursework, let's make sure you are!