

Github Orientation, Nanoenergy Lab

Written by Eric Rong

Introduction to Github and Git (including vocabulary)

This is the introduction to Github for the Johns Hopkins Nanoenergy Lab. Github is a remote, cloud-based platform for the development and management of what is primarily code, but other files and data may also be stored here. All content is stored in a **repository**, which is essentially a folder which contains all the data and files in one project. Each repository may be public or private (for instance, this repository, github.com/jhu-nanoenergy/documentation is public).

Github is essentially a storage utility which is designed around the **Git** Version Control System (VCS), a tool which allows you to track incremental progress. Git constructs a time-varying state of your code progress through keeping track of a sequence of **commits**, each of which represents an update from the previous commit (whether you're adding lines, changing lines, or removing lines of code). This way, you may track modifications to your code by walking through the commits, each of which can clearly tell you what exactly changed between the past commit and the current commit.

In practice, you'll be making several commits as you code, each containing some changes you make, and the frequency of committing changes is up to you to decide. However, it is advised to commit your changes often, as you can revert to a previous commit if you break your code, or otherwise want to undo different changes.

After a certain amount of commits, or whenever you wish to publish your code to the **remote** repository, you will **push** all the commits to the Github server, and those changes then be accessible by all who have access to it. Anybody can see the code in a public repository, but only collaborators may see the code in a private repository. However, regardless of visibility permissions, only collaborators of any repository may have push access (because even public repositories do not want random people changing their code.).

However, Git will not automatically refresh the **local** copy of the remote repository (which you **cloned** to your computer previously) when other people commit new changes to the remote repository. Therefore, you must manually **pull** changes from the remote repository to keep your version as up to date as possible.

In essence, Git is like a backup system for incremental development of your code, Github stores all of the data from Git in their servers, and your computer keeps a copy of all the files from Github and updates them upon request.

Using Github

Github CLI vs Github Desktop

The Git VCS was originally designed as a command line tool to use in the terminal, the Command-Line Interface shell (the Terminal app on macOS, Windows Powershell on Windows, or the Linux terminal on Linux). For people who do not write software often, it may be more intuitive to rely on the [Github Desktop app](#) (available for macOS and Windows) & the Github website, unless they have experience with Unix commands and the Command Line Interface. This guide will be updated to include information about that, but until then, it will focus on the Graphical User Interface of Github Desktop.

Manage Your Github Account

First, decide whether you wish to use the group account, github.com/jhu-nanoenergy, or to [create your own](#). It is generally preferable to use your own Github account so you can keep track of who has done what for easy accountability, and to use Github easily with your non-lab-related projects (it is a pain to sign off and on again). Just make sure to add your own account as a collaborator on the repositories you wish to work on and contribute to. To do so, click on the repository on the Github website, select the **Settings** tab as follows, select **Manage access** in the menu on the left, and select **Invite a collaborator** to add your personal account. Don't forget to accept the invitation in your email.

jhu-nanoenergy / documentation

Unwatch1Star0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Secrets

Actions

Moderation settings

Interaction limits

Who has access

PUBLIC REPOSITORY

This repository is public and visible to anyone.

Manage

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

You haven't invited any collaborators yet

Invite a collaborator

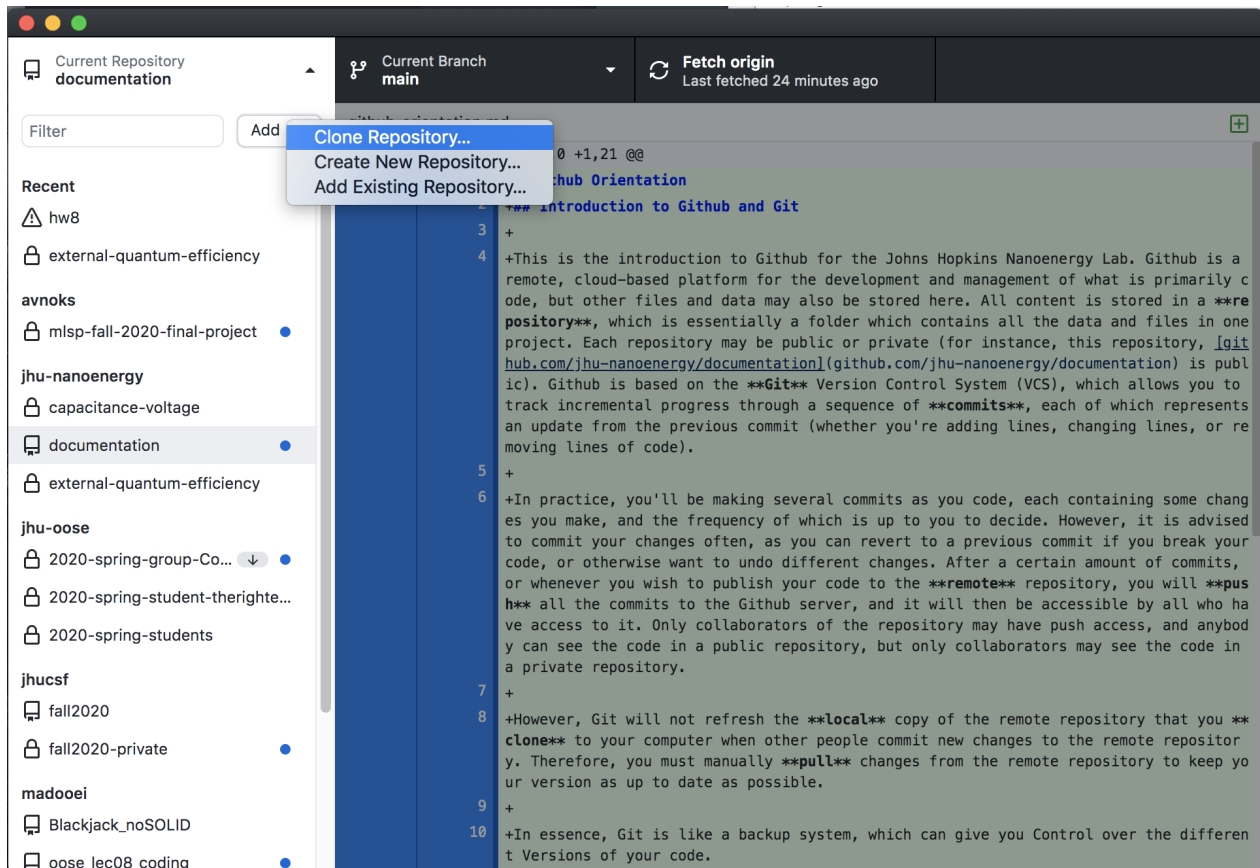
Using Github Desktop

Downloading Github Desktop

Next, download the [Github Desktop app](#) and sign into your account. You will see something like the following, except you likely will not have repositories showing on the left-hand tab.

Clone (Download) Existing Repository

To download a copy of an existing remote repository to your local computer, select **Clone Repository**.



Next, select which repository you wish to clone, and choose a local path on your computer where you want the repository to be located.

Clone a Repository

GitHub.com

GitHub Enterprise Server

URL

Filter your repositories

Your Repositories

avnoks/mlsp-fall-2020-final-project

jhu-nanoenergy/capacitance-voltage

jhu-nanoenergy/current-voltage

jhu-nanoenergy/documentation

jhu-nanoenergy/external-quantum-efficiency

Local Path


/Users/ericrong/Documents/Personal/Eric Rong/Education/Johns

Choose...

Cancel

Clone

After cloning, you will see that the repository (documentation) is selected, and any changes to the contents therein will be reflected (it should be a blank list immediately after, as you have made no changes yet).

 **Current Repository**
documentation ▼

Changes


History

☒ 0 changed files

Changing Repositories

If you have multiple projects in multiple repositories, you will find that you need to change the current repository in the Github Desktop app when you want to switch between projects. To change the repository, select the **Current Repository** dropdown and select the other one. All the repositories that show are ones which Git is tracking, so you must have either cloned them in the past, or created them locally.

Only when you choose the correct repository can you pull/push from it. Git does not automatically update your local files with any changes on the remote repository, so you must manually pull changes, as below.


Current Repository
documentation


▲

Filter


Add ▼

Recent

current-voltage


hw8


avnoks


mlsp-fall-2020-final-project

●


jhu-nanoenergy

capacitance-voltage

current-voltage

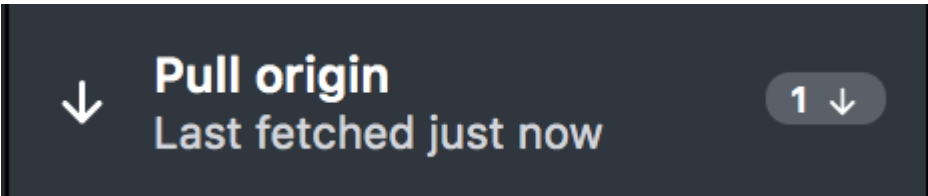
documentation

↑ ●

external-quantum-efficiency

Pull (Update Local Files) Your Cloned Repository

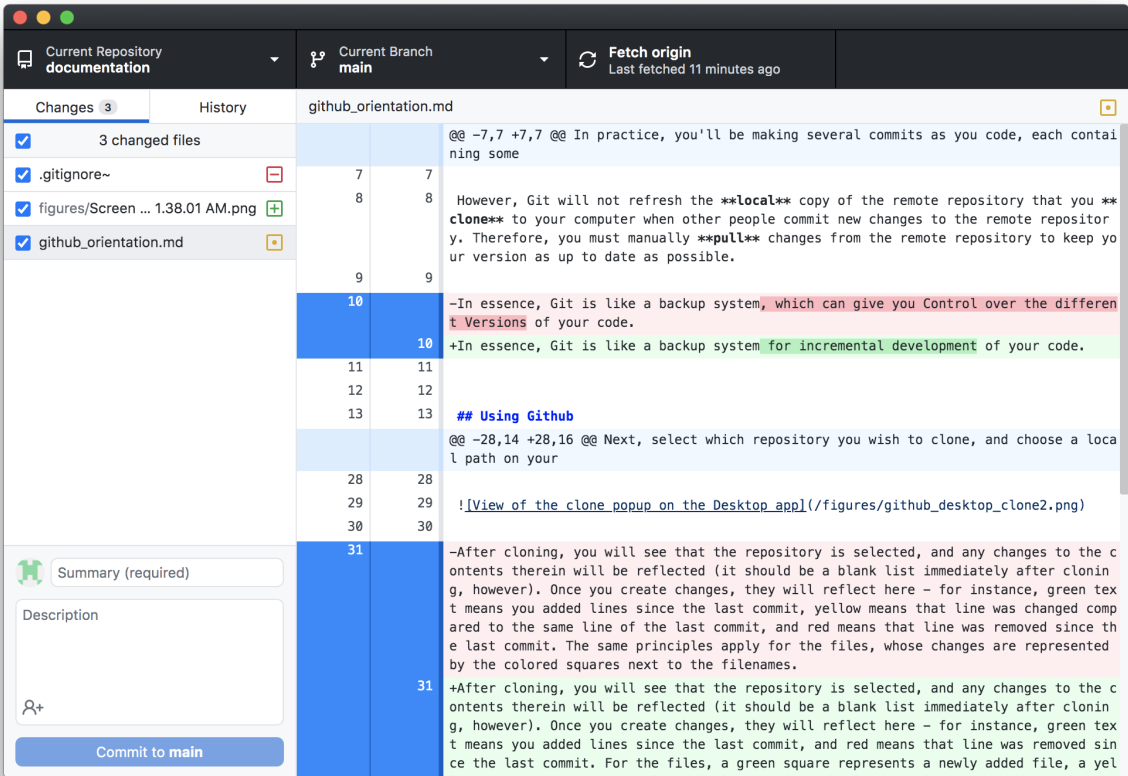
Also note the **Fetch origin** button in the top bar, which checks if the remote server has any changes that your local computer doesn't reflect. You should click this before you start making any changes so you don't run into Merge Conflicts (see below). If there are changes which you do not have locally, it will show you the option to **Pull origin**, to pull the changes from the remote repository. It also shows the number of new commits available (in this case 1).



Commit (Formalize Local Changes) Your Local Changes

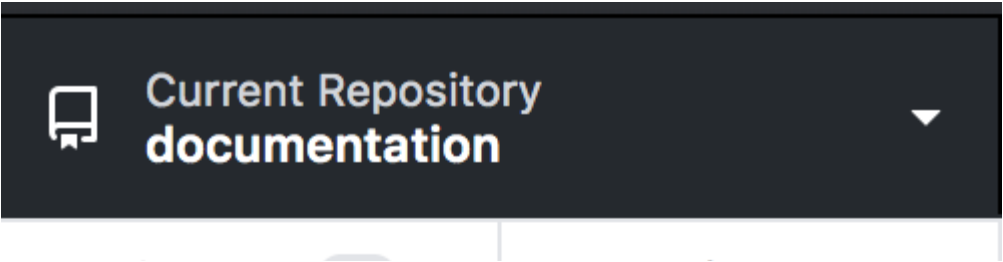
Once you create changes, they will reflect here under the names of files. For the files, a green square represents a newly added file, a yellow square represents a file with changes since the last commit, and a red square represents a file which was removed.

In the actual text within the files, the changes will also show up - green text means you added lines since the last commit, and red means that line was removed since the last commit.



When you are satisfied with the changes, you can see which files have been changed as below. You can review these changes and deselect anything you do not wish to publish to the remote repository (like .DS_Store, an autogenerated macOS file, in this case).

Write a commit message ("add intro to git/hub, begin using..." in this case), an *optional* description, and press "Commit to main."



Changes 11

History



11 changed files



.DS_Store



figur.../github_desktop_clone.png



fig.../github_desktop_clone2.png



figures/github_...rge_conflict.png



figures/github_...pull_origin.png



figur.../github_desktop_stash.png



fi.../github_desktop_stashed.png



figures/github_desktop_view.png



figures/repo_view.png



github_orientation.md

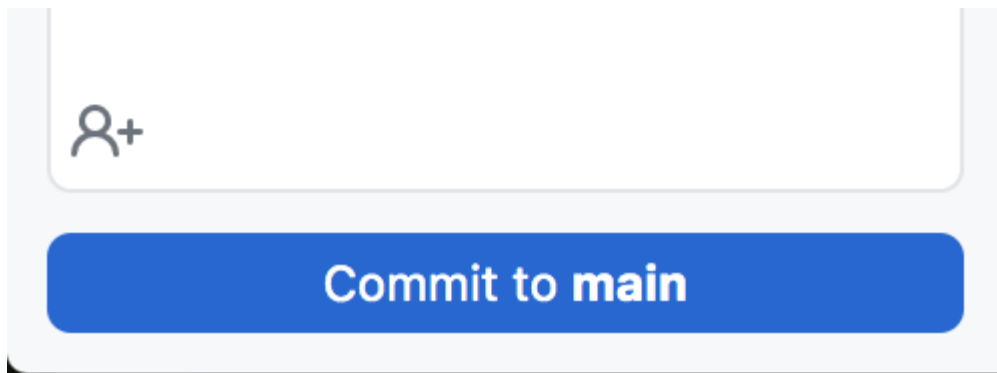


README.md



add intro to git/hub, begin using (

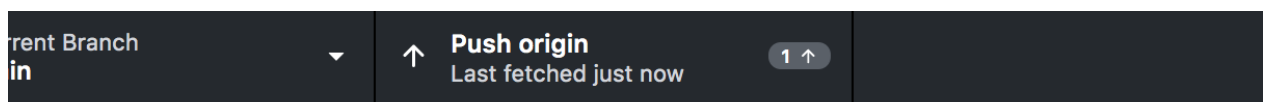
Description



Push (Update Remote Files) Your Local Commits

However, committing does not automatically publish to the remote repository. It merely **stages** your local commits before you decide to push your commits. Usually, people will commit every few minutes or so, and push every few commits - as stated before, committing often can be very useful if you ever need to revert to a recent, previous commit to get rid of any errors you introduced.

Before you push, make sure to fetch changes from remote and pull those changes, otherwise you run the risk of merge conflicts (see below). Then, when you're ready to push your staged commits, select either "Push origin" button and it will publish. If you fetch changes immediately before pushing, it is unlikely you will run into issues.





No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.



Push commits to the origin remote

You have 1 local commit waiting to be pushed to GitHub.

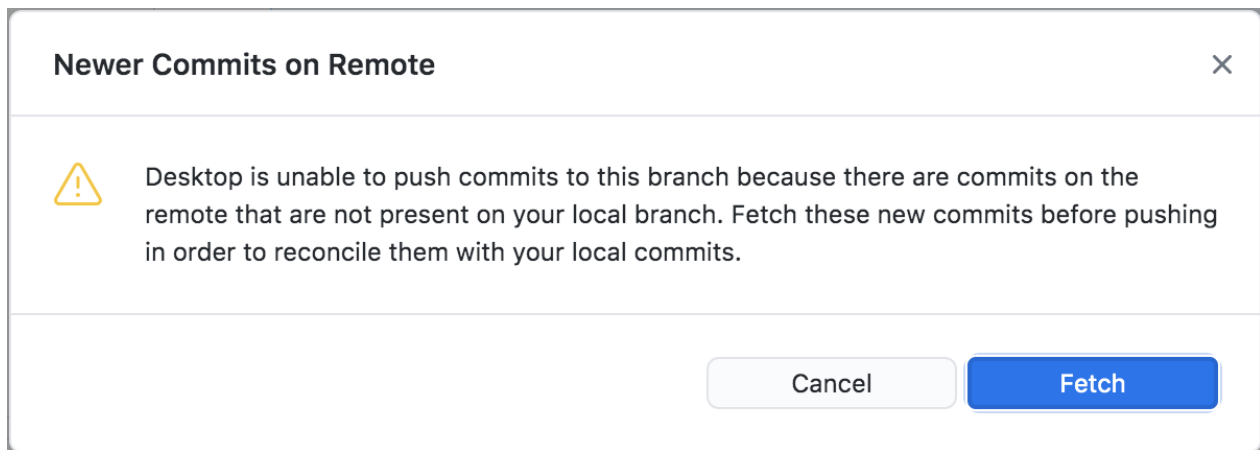
Always available in the toolbar when there are local commits waiting to be pushed or  

Push origin

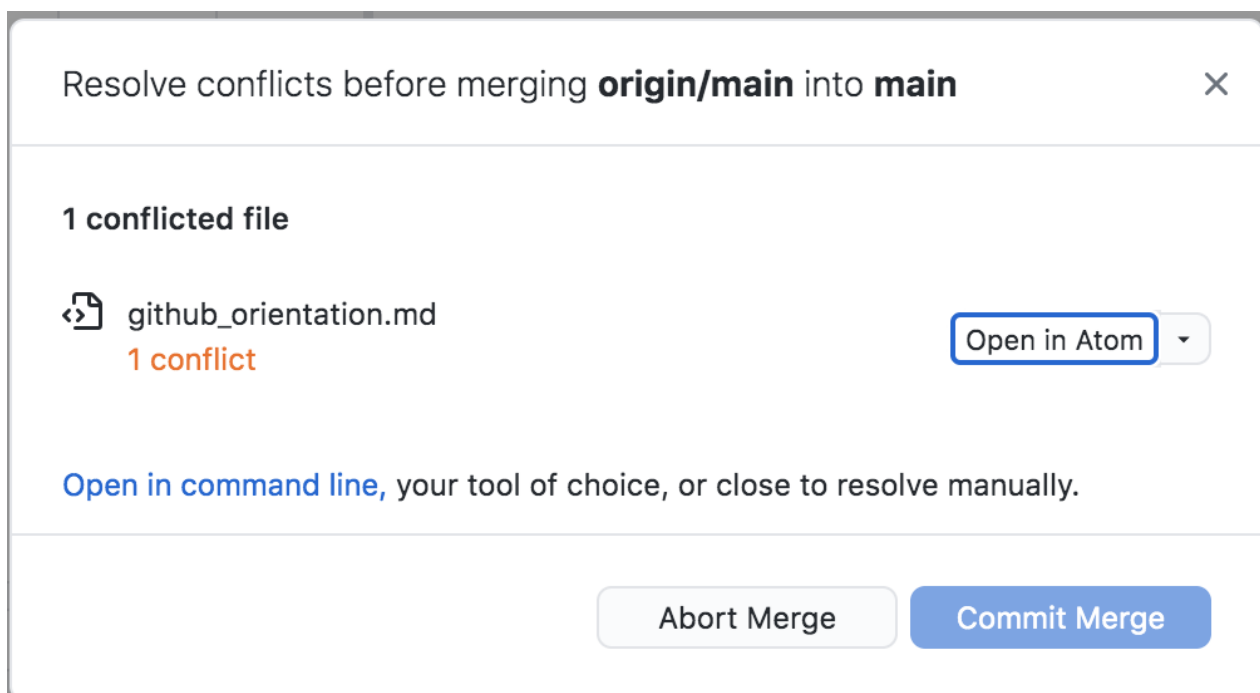
Merge Conflicts

Merge Conflicts When Pushing To Remote

You may have issues when you push code with your changes. For example, if someone else had committed code to the remote repository which you did not pull, you will get an error message. In this case, fetch the changes from the remote repository.



You may see a popup containing files where there are conflicting lines of code.



Open the conflicting file in your text editor of choice (mine is [Atom](#)). You will see some arrows with comparisons of what is on the line locally, and what the remote commit (with commit number [4c35e8e34a504893a420a73ffaa075cdc23892ec](#)) has, both of which are different. Reconcile these differences (by removing the arrows and deciding what the correct line(s) should be), and you can commit a conflict-free file.

```

55 <<<<<< HEAD
56 You may also have issues when you push code with your changes,
   • but someone else had committed code to the remote repository
   • which you did not pull.
57
58 ![View of merge conflict when pushing](/figures/
   • github_desktop_merge_conflict_push.png)
59
60 In this case, fetch the changes from the remote repository,
61
62 =====
63 asdf
64 >>>>>> 4c35e8e34a504893a420a73ffaa075cdc23892ec


```

This may be avoided by coordinating changes with your collaborators, and trying to not change the same lines of code between collaborators.

Merge Conflicts When Pulling From Remote

Sometimes, there may also be issues when pulling from the remote repository, if other changes conflict with changes you made. For example, if the remote commit changed line 1, but you also changed line 1 locally without having committed it, you will run into a merge conflict. You will get a popup that instructs you to stash your local changes and conform to the remote repository.

Error



Unable to pull when changes are present on your branch. The following files would be overwritten:

README.md

You can stash your changes now and recover them afterwards.

Close

Stash Changes and Continue

Try to avoid this by coordinating with collaborators on what you will each be working on, and in particular which lines you might each be working on. But if it happens, you have one of two options: either delete the file(s) which you changed locally, or if you wish to keep your changes, you can stash your local changes, and proceed to view those stashed changes.

View your stashed changes

You have 1 change in progress that you have not yet committed.

When a stash exists, access it at the bottom of the Changes tab to the left.

View stash

Restore your stashed changes to your local repository.

Changes

History

☒ 0 changed files

Stashed changes

Discard

Restore

Restore will move your stashed files to the Changes list.

github_orientation.md			
	40	40	@@ -40,6 +40,6 @@ Sometimes, there may be issues when pulling from the remote repository, if other
	41	41	![[View of the popup to stash your changes]](/figures/github_desktop_stash.png)
	42	42	
	43		-Try to avoid this by coordinating with collaborators on what you will each be working on, and in particular which lines you may be Stash your changes, but
		43	+Try to avoid this by coordinating with collaborators on what you will each be working on, and in particular which lines you might each be working on. But if it happens, stash your local changes, and
	44	44	
	45	45	To change the repository, select the Current Repository dropdown and select the other one. Only then can you pull/push.

Stashed Changes

Summary (required)

Description

See that it shows the differences between the conflicting commit and your local work. From this, decide which version to keep, remove the arrows, make sure the line(s) read exactly how you want them to, commit these changes, and push the commit with the conflict-free file to the remote repository.

☒ github_orientation.md

40	40	
41	41	![[View of the popup to stash your changes]](/figures/github_desktop_stash.png)
42	42	
	43	+<<<<<< Updated upstream
43	44	Try to avoid this by coordinating with collaborators on what you will each be working on, and in particular which lines you may be Stash your changes,
	45	+=====
	46	+Try to avoid this by coordinating with collaborators on what you will each be working on, and in particular which lines you might each be working on. But if it happens, stash your local changes, and
	47	+>>>>>> Stashed changes
44	48	