Group Project

Git and GitHub Activities

Step 0: Install git account

* Follow the instructions in (<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>) to install git (if it's not already installed).

**Working Locally**

Step 1: Create a local git repository

* When creating a new project on your local machine using git, you'll first create a new repository.
* To begin, open up a terminal and move to where you want to place the project on your local machine using the cd (change directory) command. For example, if you have a 'projects' folder on your desktop, you'd do something like:

cd ~/Desktop

mkdir myproject

cd myproject/

* To initialize a git repository in the root of the folder, run the **git init** command



Step 2: Add a new file to the repo

* Go ahead and add a new files to the project
* Once you've added or modified files in a folder containing a git repo, git will notice that changes have been made inside the repo. But, git won't officially keep track of the file (that is, put it in a commit) unless you explicitly tell it to
* After creating the new file, you can use the **[git status](http://git-scm.com/docs/git-status" \t "_blank)**command to see which files git knows exist

Step 3: Add a file to the staging environment

* If you rerun the git status command, you'll see that git has seen the file but has not been added to the staging area.



* Add a file to the staging environment using the **git add**command.
* For example run **: git add "Group Project Assignment 16-17.pdf"**

Step 4: Create a commit

* Run the command **git commit -m "Your message about the commit"**
* The message at the end of the commit should be something related to what the commit contains - maybe it's a new feature, maybe it's a bug fix, maybe it's just fixing a typo.

Step 5: Create a new branch

* Branches allow you to move back and forth between 'states' of a project. For instance, if you want to add a new page to your website you can create a new branch just for that page without affecting the main part of the project. Once you're done with the page, you can **[merge](http://git-scm.com/docs/git-merge" \t "_blank)** your changes from your branch into the master branch. When you create a new branch, Git keeps track of which commit your branch 'branched' off of, so it knows the history behind all the files
* To create a new branch to develop your web page. Here's what you'll do: Run **[git checkout -b <my branch name>](http://git-scm.com/docs/git-checkout" \t "_blank).** This command will automatically create a new branch and then 'check you out' on it, meaning git will move you to that branch, off of the master branch.



* After running the above command, you can use the **[git branch](http://git-scm.com/docs/git-branch" \t "_blank)** command to confirm that your branch was created



(This guide has been sourced from <https://guides.github.com/introduction/flow/>)

**Working With GitHub**

If you only want to keep track of your code locally, you don't need to use GitHub. But if you want to work with a team, you use GitHub to collaboratively modify the project's code

**Step 0**: Create GitHub account and download github application

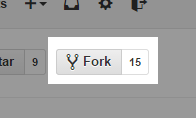
* Create a GitHub account (<https://github.com/>). Accounts are free for public repositories, but there's a charge for private repositories.
* Install github desktop https://desktop.github.com/

### Step 1: Create a new repository on GitHub

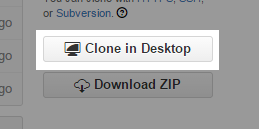
* To create a new repo on GitHub, log in and go to the GitHub home page. You should see a green '+ New repository' button



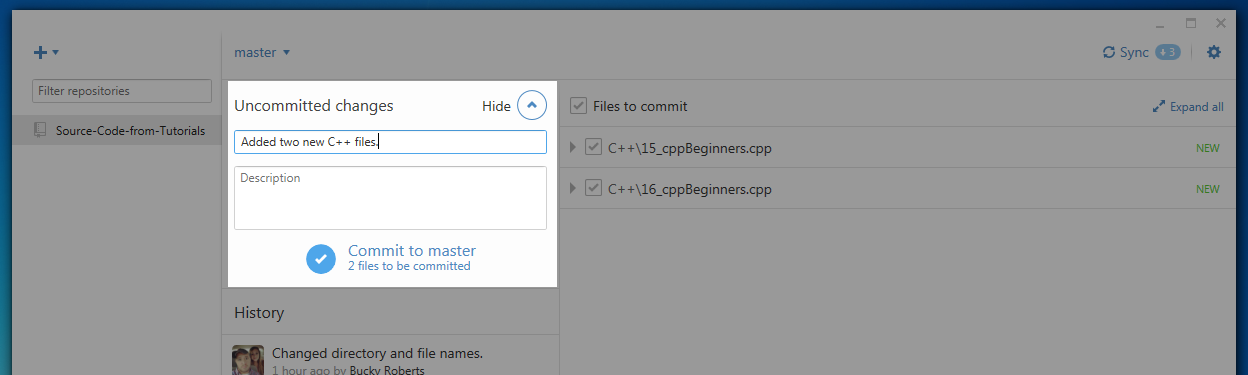
* Once you repository is created, you can upload your files to the newly created repository.
* On the top right of this page, click "Fork". This will create a copy for you.

[](https://camo.githubusercontent.com/78544ab6e644e34db5ed865b34d9061653680ecb/687474703a2f2f692e696d6775722e636f6d2f5554777a4f67432e706e67)

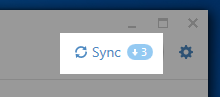
* Once your copy has been created, click “Clone in Desktop” to download and save it on your computer.

[](https://camo.githubusercontent.com/e911f48324d4a45db57d27becc1e27271ff032ed/687474703a2f2f692e696d6775722e636f6d2f754e79356948672e706e67)

* From here you can make changes, add folders, add source code files, etc...
* When you are finished with your changes, open GitHub app for Windows or Mac
* Add a commit message and click "Commit to master"

[](https://camo.githubusercontent.com/7c2e0c3191ccb68732772304a6a2aa916db34775/687474703a2f2f692e696d6775722e636f6d2f6a48636a5876462e706e67)

* Click "Sync" on the top right to save the changes to your GitHub account.

[](https://camo.githubusercontent.com/270a687138af9467f3412bc8595b5795006a1a45/687474703a2f2f692e696d6775722e636f6d2f686b396d495a532e706e67)

**Step 2:  Understanding GitHub Flow**

### 

### Step 2.1: Create a branch

When you're working on a project, you're going to have a bunch of different features or ideas in progress at any given time – some of which are ready to go, and others which are not. Branching exists to help you manage this workflow.

When you create a branch in your project, you're creating an environment where you can try out new ideas. Changes you make on a branch don't affect the master branch, so you're free to experiment and commit changes, safe in the knowledge that your branch won't be merged until it's ready to be reviewed by someone you're collaborating with.

### Step 2.2: Add commits

Once your branch has been created, it's time to start making changes. Whenever you add, edit, or delete a file, you're making a commit, and adding them to your branch. This process of adding commits keeps track of your progress as you work on a feature branch.

Commits also create a transparent history of your work that others can follow to understand what you've done and why. Each commit has an associated commit message, which is a description explaining why a particular change was made. Furthermore, each commit is considered a separate unit of change. This lets you roll back changes if a bug is found, or if you decide to head in a different direction

### Step 2.3: Create a pull request

Pull Requests initiate discussion about your commits. Because they're tightly integrated with the underlying Git repository, anyone can see exactly what changes would be merged if they accept your request.

You can open a Pull Request at any point during the development process: when you have little or no code but want to share some screenshots or general ideas, when you're stuck and need help or advice, or when you're ready for someone to review your work.

### Step 2.4: Discuss and review your code

Once a Pull Request has been opened, the person or team reviewing your changes may have questions or comments. Perhaps the coding style doesn't match project guidelines, the change is missing unit tests, or maybe everything looks great and props are in order. Pull Requests are designed to encourage and capture this type of conversation.

You can also continue to push to your branch in light of discussion and feedback about your commits. If someone comments that you forgot to do something or if there is a bug in the code, you can fix it in your branch and push up the change. GitHub will show your new commits and any additional feedback you may receive in the unified Pull Request view.