GitHub

GIT + GITHUB

Abstract

A condensed manual on what Git and GitHub are and how to use these powerful tools

Acknowledgements & Disclaimers

Any and all images, screenshots, source code snippets, and additional documentation included in this manual are either the sole property of Yuri Khechoyan or third-party entities.

Computer programs are literary works under the definition of the Copyright Act, 17 U.S.C. § 101

Any Git commands shown, will be italicized in blue

These Git commands must be entered as shown, if not - commands will not be executed

In order to execute all Git commands, the Enter key must be pressed

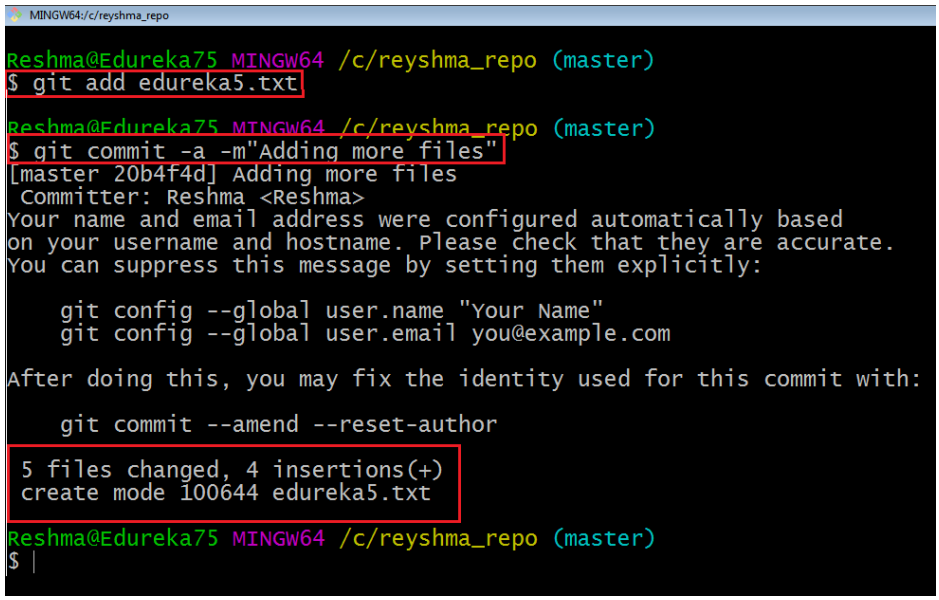
Due to this book being a condensed version, it will not cover absolutely everything. It will mainly cover the basics.

This book is written in the style of an instruction manual. You may skip around section(s) if you have these procedures completed prior to reading this book.

Table Of Contents

1. What is Git?

Git is a powerful Version Control based system that has a User Interface similar to Command Prompt on Windows or Terminal on Mac. Git is used in order to create version-controlled workspaces called *Repositories or Repos*.



Git uses specialized key words in order to execute a task. The specialized keys may or may not be the same key words used inside of Command Prompt (Windows) or Terminal (Mac). These key words give you the ability to create, modify, delete your directories however you see fit.

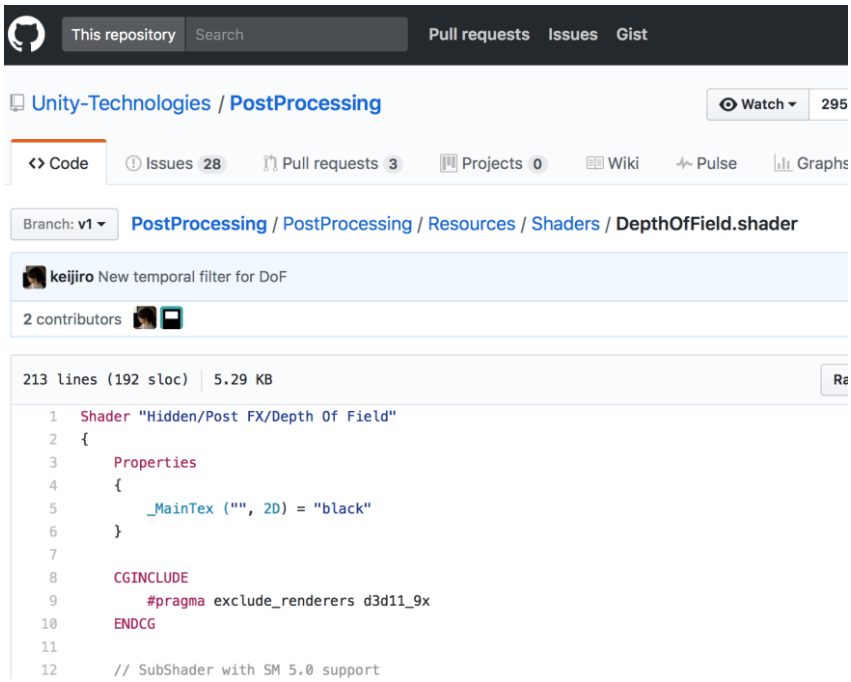
- Basic Git commands will be included on a later page in this manual

2. What is GitHub?

GitHub is an online platform that enables developers to put their source code somewhere for storage and code management purposes.

- **Public Repositories** are open source projects. Anyone can view, use, and modify your code to their heart's content. But only authorized developers can contribute to a specific project. Typically, Public Repository are free.

- **Private Repositories** are projects that are closed off and are only visible to those working on it. (example: the source code that makes Microsoft Word work is only seen by Microsoft Developers/Software Engineers). Private Repositories can come at a cost.



Uses of GitHub:

- Identifying where software bugs occur
- Identify who is responsible for what source code changes
- Identify what code was added, modified, and deleted between versions.
- Keep archive of all of versions made to project (commits)

3. Getting Started with Git

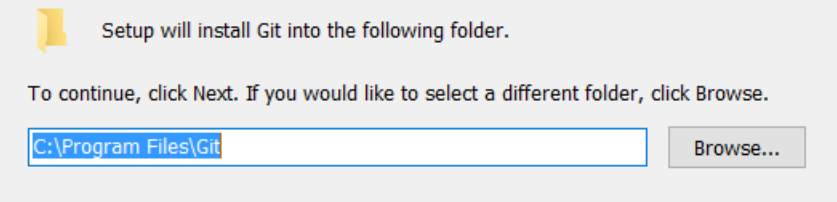
3.1 Download & Install Git

First things first… Let's download Git.

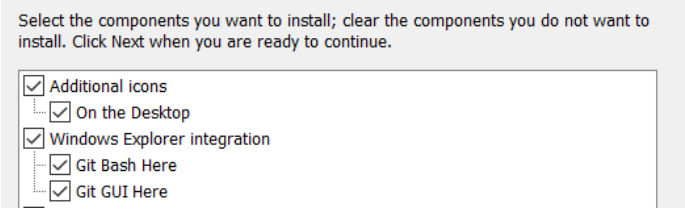
In order to download Git, visit https://git-scm.com/. Once there, you will need to download the correct version of the program. This means that you need to download the Windows or Mac files (whichever system you plan to run Git on).

3.2 Installation Process

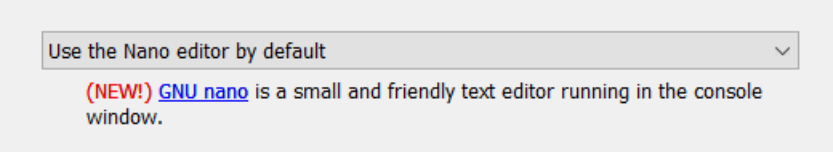
1. Open Installation Wizard
2. Select Installation Location for Git.



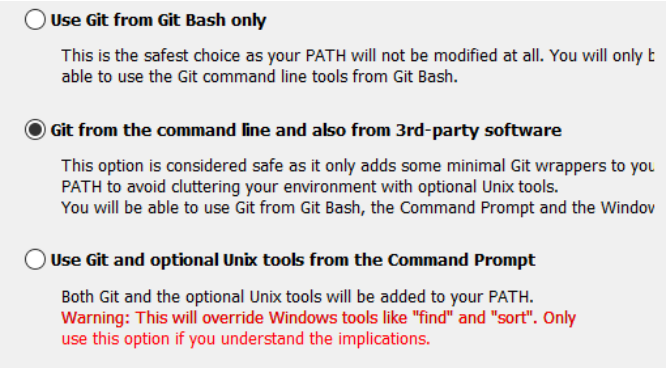
3. **Enable Additional Icons** on the Desktop.



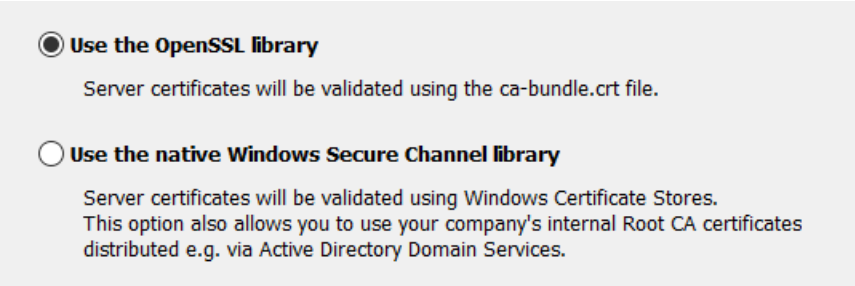
4. When choosing your default text editor, select the 2nd option: ***Use the Nano editor by editor***



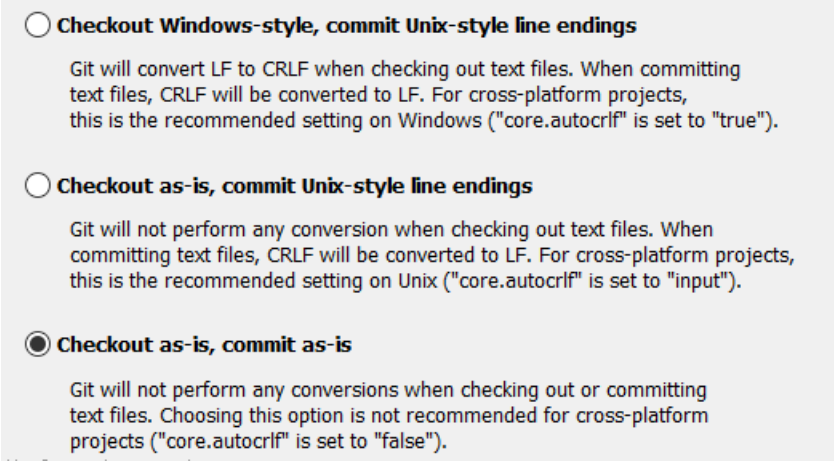
5. When choosing your PATH environment, select the 2nd option: ***Git from the Windows Command line and also from 3rd-Party software***



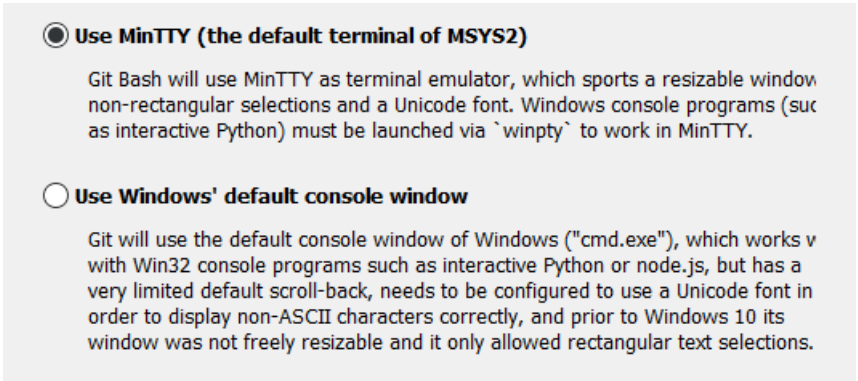
6. When choosing the HTTPS transport backend, select ***Use OpenSSL library***



7. When choosing to configure line ending conversions, select the last option: ***Checkout as-is, commit as-is***



8. When choosing terminal emulator, select the 1st option: ***Use MinTTY (the default terminal of MSYS2)***



9. Continue clicking ***Next*** until installation concludes

3.3 Initial Git Configuration

NOTE: Sections 3.3 and 3.4 are only done once during initial setup. You will not need to do this again unless you install Git on another system. Quotation marks are needed.

1. After installation is complete, launch Git

   {Type in the git command: ***git version***}

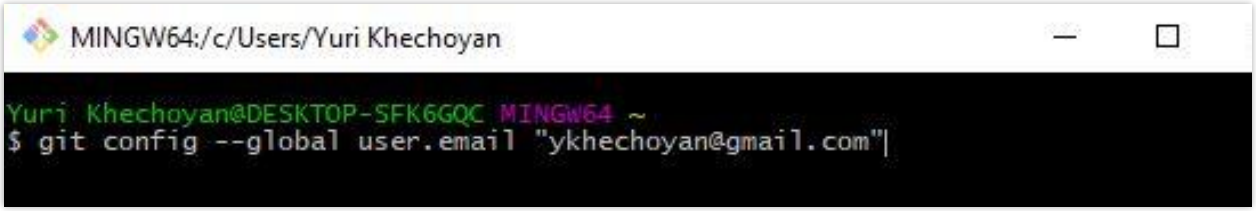
   This will have Git print out onto the console which version of Git is installed. This will not only tell you that Git has been successfully installed, but also a verification measurement to see if the version of Git installed matches the latest version available. If they match, move to Step 2.

2. Next, we will need to have Git authenticated. This is done so that when you start making commits to GitHub, your computer can sign off on those changes and let GitHub know that it is you that is pushing those changes. In place of *Username*, type in your name.

{Type in the command: ***git config --global user.name "Username"***}

3. Then do the same thing for email. In place of email, type in your email (ex. name@domain.com)

{Type in the command: ***git config --global user.email*** name@domain.com}



3.4 Initial Git Configuration: Verification

4. To verify that the correct credentials have been entered:

{Type in the command: ***git config --global --list***}

NOTE: If the Username has been entered incorrectly, repeat Step 2. Repeat Step 4 to verify again.

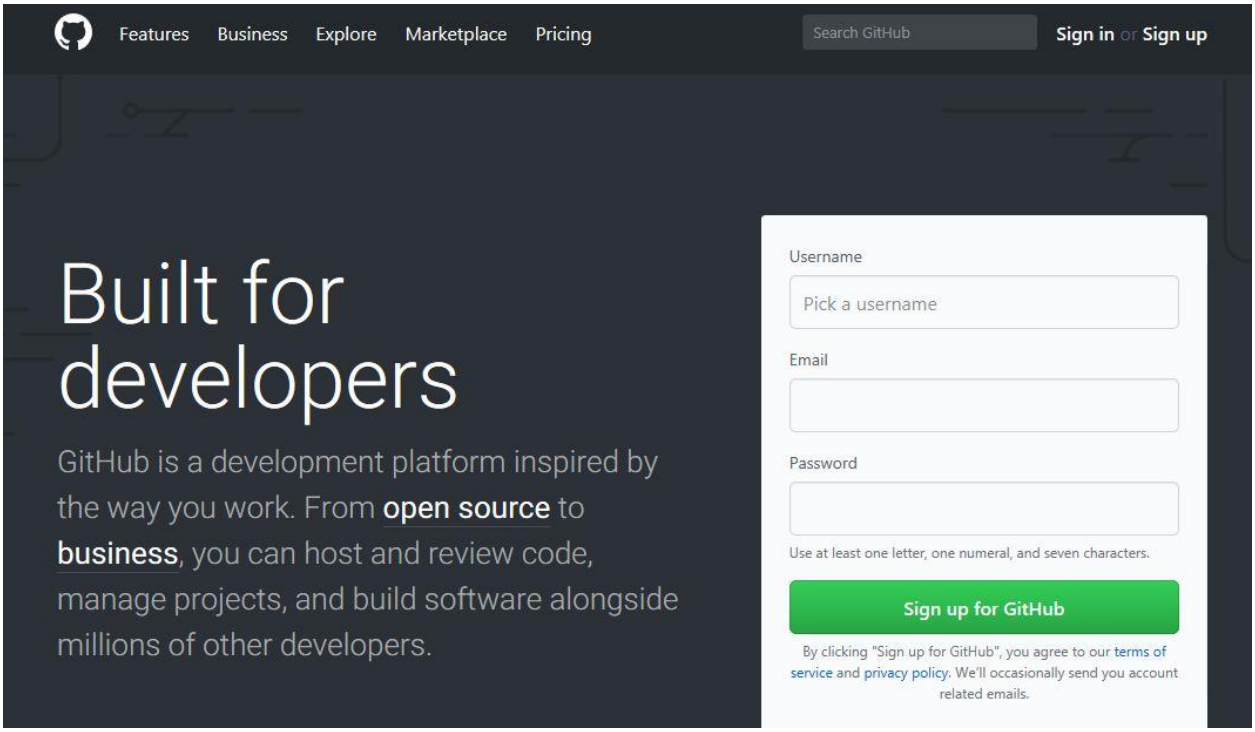
NOTE: If the email has been entered incorrectly, repeat Step 3. Repeat Step 4 to verify again.

4. Getting Started with GitHub

4.1 Creating a GitHub Account

Congratulations! You have successfully downloaded and installed Git! Now onto the Cloud Based Repository.

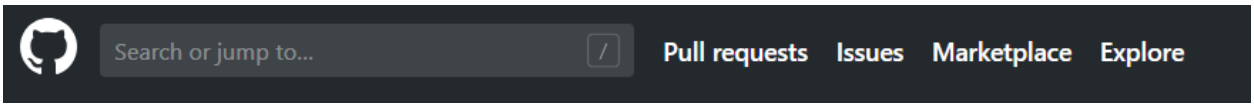
1. Go to https://github.com/
2. Provide a Username, Password, and an Email
3. Click the green Sign Up Button
4. Verification email will be sent to email address you provided
5. Log In

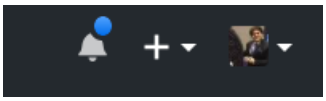


4.2 GitHub UI Overview

Once logged into GitHub it may look overwhelming. But do not worry. It is easy to figure out what each part is and does.

At the top of the website is your navigation bar. This will be used to peruse through GitHub's site.





On the top left, is GitHub's Mascot appropriately named: ***Octocat!*** To the right of that is the search bar. The search bar can be used to search the entire GitHub database for:

- Repositories by name (if you do not have the direct hyperlink to them)
- Users

To the right of the search bar you will find:

Pull Requests: If you are the administrator of a repository, this is where you can view, review, and approve changes.

Issues: This is where you can view the list of identified issues and bugs across all your repositories/projects.

Marketplace: This is where you integrate different 3rd party extensions to connect into your GitHub projects.

Explore: This is where you can read up on what's new with GitHub, see trending repositories, see any upcoming GitHub events, and other resources.

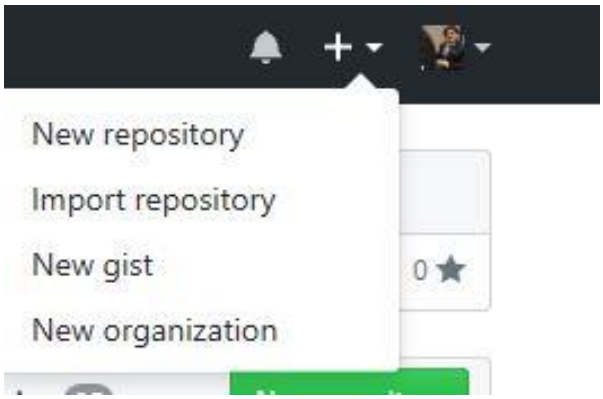
Notifications: This is self-explanatory

+: This is the create button

Your Account: This is where you can administer your account

4.3 Far Right Side of the Nav. Bar

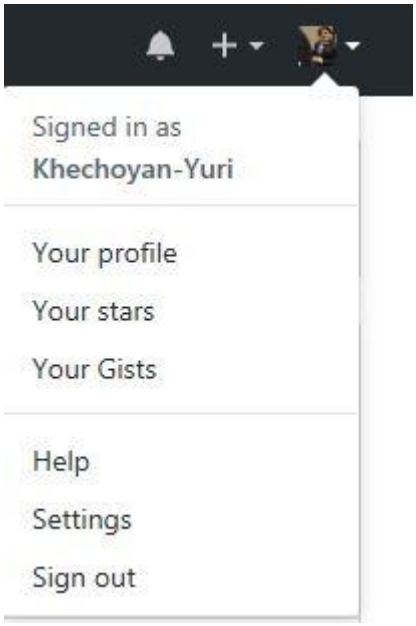
The right-hand side of the Navigation Bar is what you will use to navigate through GitHub the most.



To the right of your Notifications is your create button. You will use this for several things:

- Create a new Repository (Private or Public)
- Import an existing Repository (your projects from GitHub's database)
- Create a new Gist (a text document; that doesn't really have purpose)
- New Organization

When you click on your Profile picture, it opens up a different sub-menu.



- Your Profile
- Stars (Favorite Repositories)
- Gists
- Help
- Settings
- Sign Out

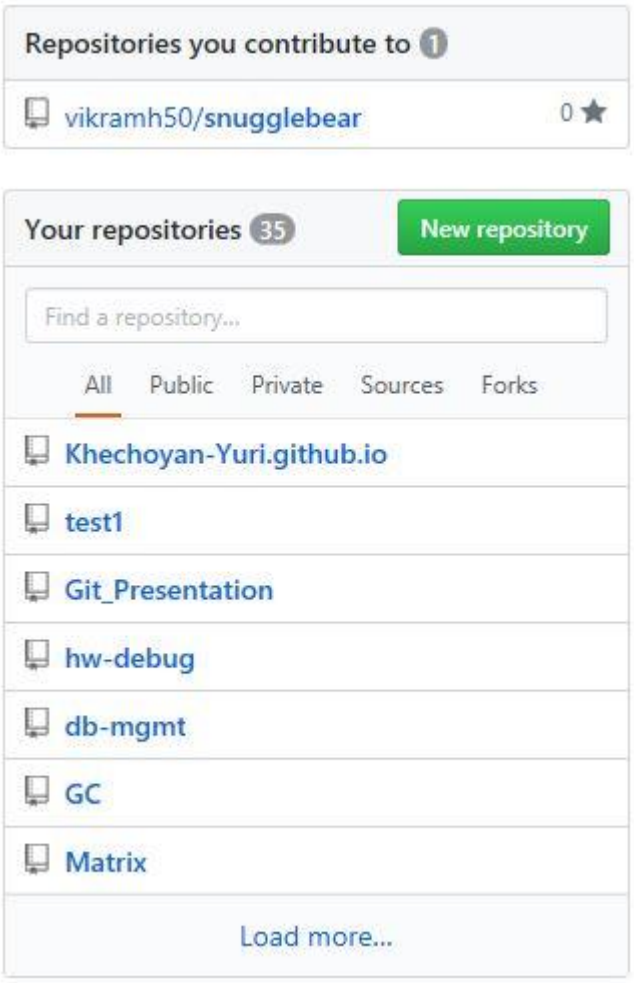
Settings:

Settings is where you will go to modify your personal/account information, add a payment method (if you plan to create Private Repositories), and Delete existing repositories that you have created.

4.4 Below Navigation Bar

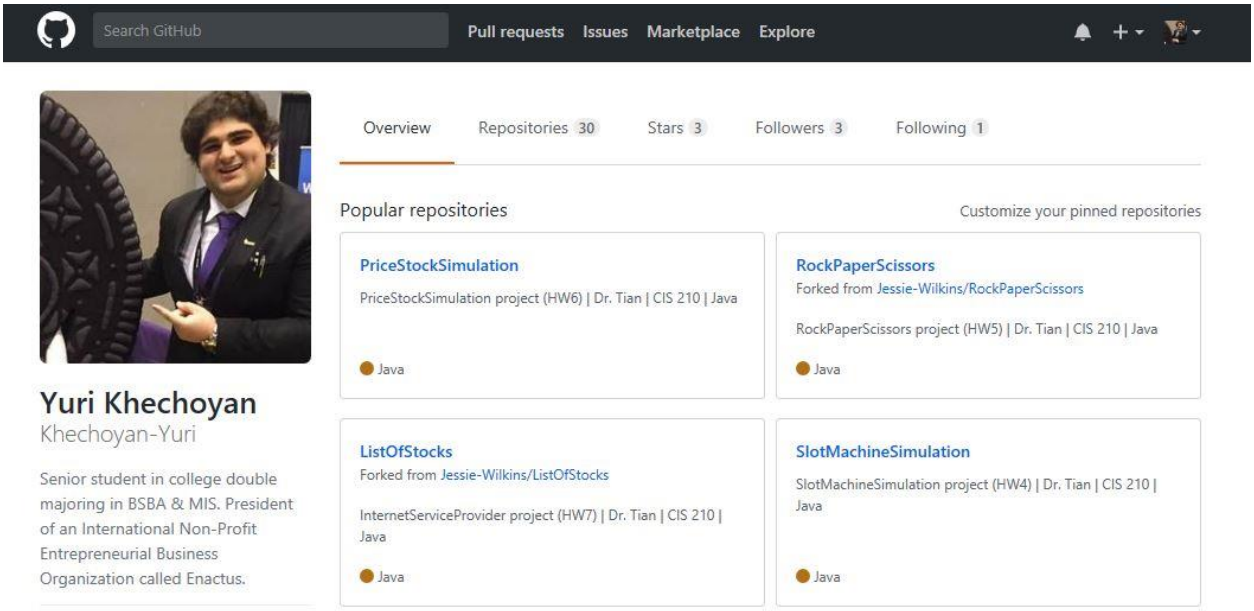
Below the Navigation Bar, you will find quick links to Repositories that you have contributed to and your personal repositories that you have created. Your repositories are categorized as follows:

- All
- Public
- Private (if you have a paid account)
- Sources (sources of your repositories)
- Forks (carbon copies of other repositories that you have copied over to your own account)



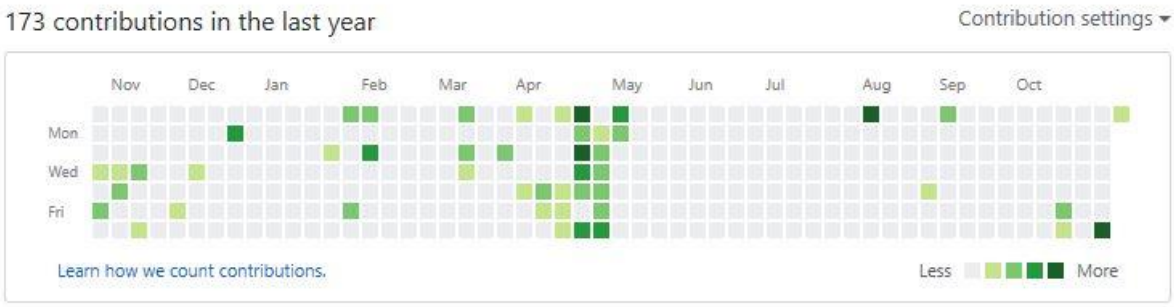
4.5 Your Profile

Your profile consists of some very useful information.



- Your profile picture
- An “About Me” blurb
- Your most popular repositories
- **Repositories**: All of the repositories you have created
- Stars: Repositories that you have favorited/liked
- Followers: People that follow your work
- Following: People that you are following

If you scroll further down, you will find some more useful information...



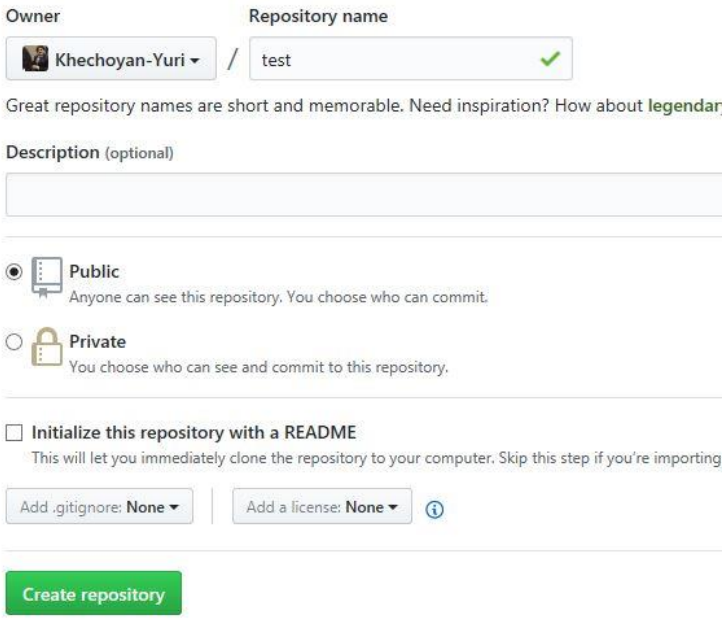
There is a timeline of all the commits that you have made in the past year. The darker the green the squares are, the more commits you have made during that day. Each day of the year is denoted by a single square.

5. Creating Your First Repository on GitHub

5.1 New Repository

When creating a new repository, there are a few things that require your input:

- Repository Name
- Description: this can be added later
- Public or Private Classification
- Initializing with a README.md
- .gitignore restrictions
- Licensing restrictions



- When choosing a name for your repository, GitHub will automatically search through its database and see if there is a pre-existing repository within your own account that has the same name. If so, GitHub will tell you. Otherwise if the repository name has not been taken, you will see a green check mark next to the name.

NOTE: Think of this as your own dedicated folder. It is acceptable for 2 people to have 1 project with the same name if and only if it is in their respective folders. But it is not

acceptable for one to have 2 projects in one's account with the same name.

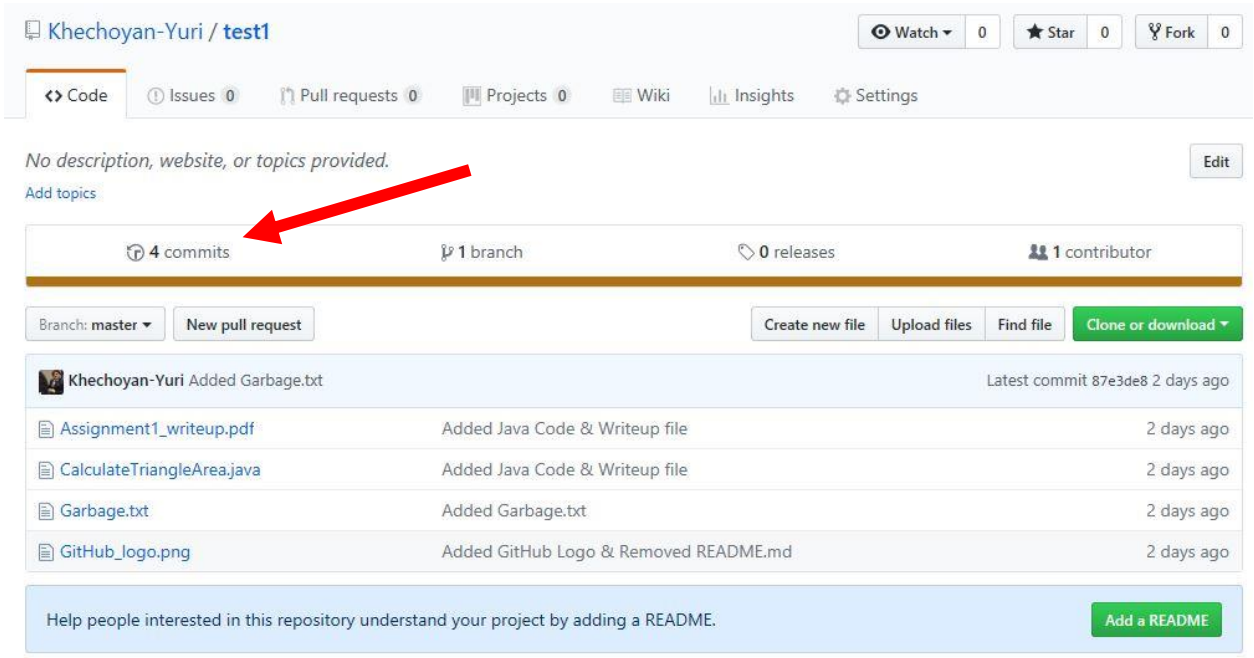
- If one wants to create a private repository has a pre-existing method of payment method of payment on file, choose Private. Otherwise, leave it as Public.
- Initializing with a README.md – check the box for now. This file can be deleted later.
- You can also customize the type of file you want git to ignore and incorporating the type of license you want your repository/project to have.

For now, leave both *.gitingnore* and *License* at their default values: none.

5.2 Your (Semi) First Commit

Congratulations! You have created your new repository! You did not see it, but when you clicked on the green Create Repository button, you actually made the first commit to that project!

Once inside the main directory of your repository on GitHub, it will look something like this:



YOU: But Yuri! My directory looks nothing like this! WHY?!

ME: I'm glad you asked – you beautiful/handsome stranger. The reason is because I added a few files and instead of having made 1 commit, I have made 4 in total. But in general, the UI is the same. When you click on the number of commits, it will take you into an archive containing all of the commits that were made by you and your authorized contributors.

Moving along to the right side of the number of commits made for that specific project you will find:

- **Branches**: it is exactly what they sound like. If you want to experiment with your code without affecting the final product, you create a branch! The default branch that is always used is called the *master* branch.
- **Releases**: are also pretty self-explanatory. These are Milestones in the code that you and your team create (e.g. v1.0, v2.36, v192.168.2.1, and so on….)

- **Contributors:** These are the people that are working on that given repository/project – whether that be you or other people authorized by you.

5.3 The Stars are Watching the Forks

At the top of your repository directory you will find some useful information as well.

- **Code**: This is where all your files will live
- **Issues**: Where any known and recorded issues within that repo/project live
- **Pull Requests:** Where the project/repo administrator can view, review, and approve changes that are submitted by other contributors.
- **Wiki**: Basically, a Wikipedia page for that project. This becomes helpful if your project is large and has many contributors. This will help understand more about the project.
- **Insights**: This is where the Analytics for that specific project live.
- **Settings**: Used for Renaming the Repo, Adding Contributors, Deleting the Repo, and more.

Then above that, you will find:

- **Watch:** This means the number of users that are watching the project outside of the contributors list.
- **Star:** This is the number of users that favorited/liked your project (similar to Facebook likes)
- **Fork**: This is the number of users that have made a carbon copy of your entire repository on their accounts so that they can have their own version to play around with, improve on, etc.

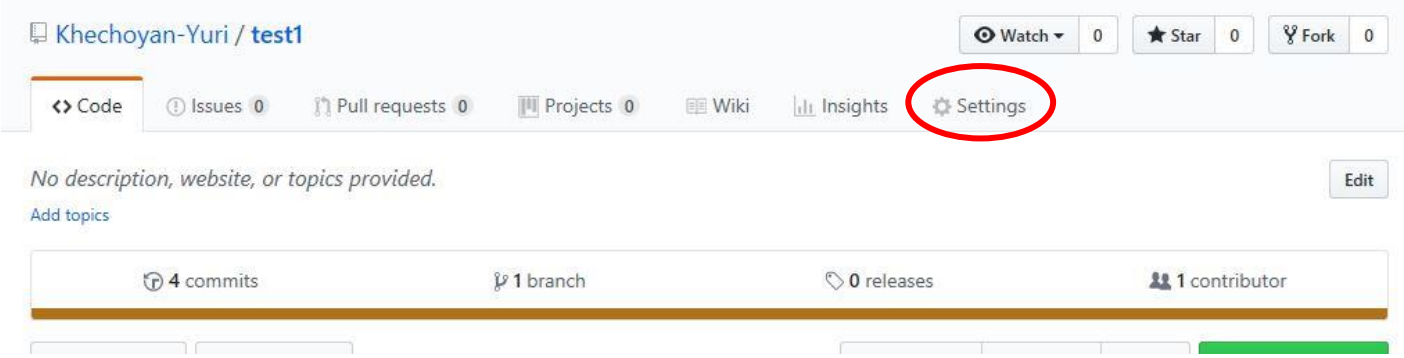
NOTE: A Fork ***ONLY*** carbon copies a given project at that given moment in time. If changes are made to the project, the carbon copy version of the project WILL NOT receive the same updates.

5.4 Ocean's 1011 (Creating a team)

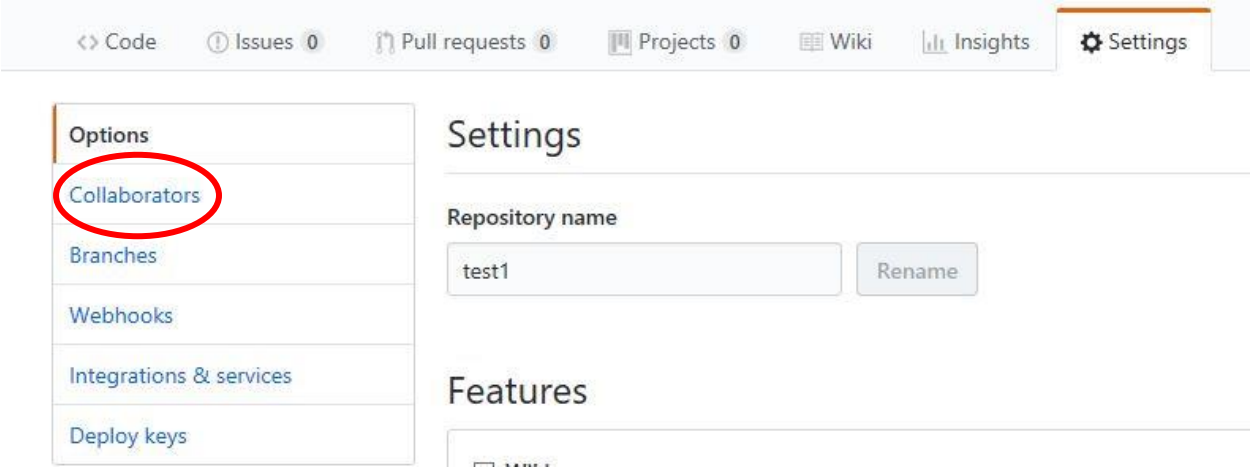
When creating a team (adding contributors) to your newly created repository/project, there are 2 things that need to occur:

- You must authorize them
- They must accept your authorization invite

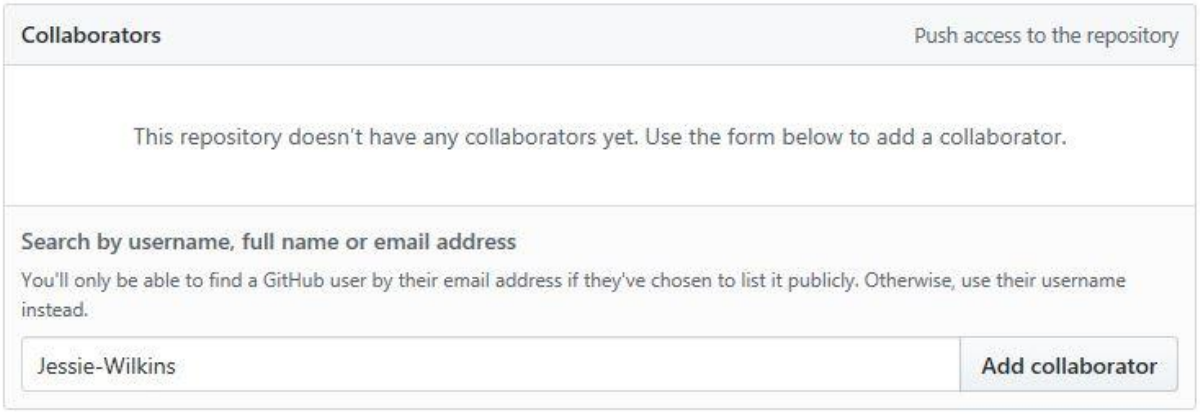
Since we are still in the main directory of your repo on GitHub, click the on **Settings** tab.



Next, on the left-hand side you will see more sections that you could go into about that project. We will click on **Collaborators**.



Finally, you will need to search for your collaborator by their GitHub Username. If you do not know their GitHub username, no worries! GitHub gives you the ability to find them by email as well. The contributor will be added to the project when they accept your authorized invitation.



5.5 Local Repository Creation with Git

> You’re still here? Marvelous! You have made my day! Let’s continue. Now that you have your GitHub repo created, it’s time to go back to Git and create a local repository. When completed, we will then link the two repositories together so that we could establish a connection between the two.

Since we have already completed the steps to configure your Git (back in section 3.3 & 3.4), we will not repeat these steps.

6. Back to Git

6.1 Commander in Chief (Git Commands)

Now it's time to get into the reason for this guide's existence.

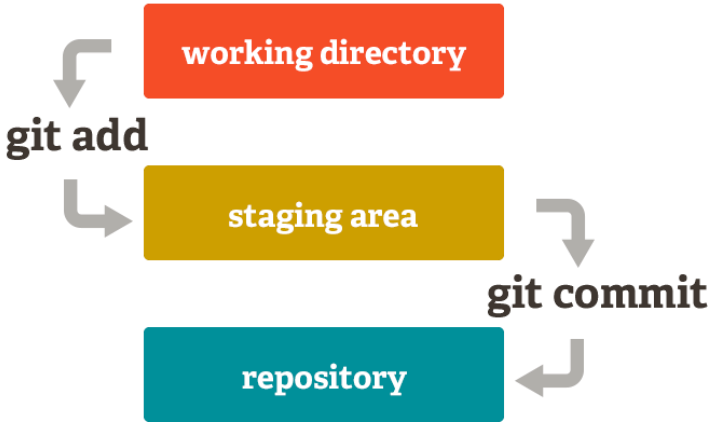
NOTE: These commands are very powerful. Follow them in the order that they appear. If these commands are not typed in accurately, the action associated with each command will not be executed. On the contrary, it is possible to completely destroy your file system if you are not careful with these commands.

IMPORTANT NOTE: Below is a diagram to illustrate how you control source code and other files within Git in order to upload from your computer (working directory) to GitHub (repository):

1. A ***Working Directory*** is at the current when you are editing your source code

2. Once you are satisfied with your changes, you must add your updated code to what is known as the ***"Staging Area".*** This basically means that your code is ready to be pushed to a repository

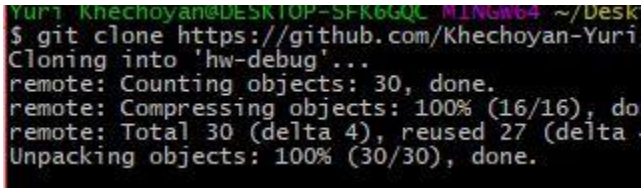
3. Once completed, you are then able to commit your new source code to the ***GitHub Repository*** that is hosted in the cloud



Git Commands (blue) & their meanings

mkdir - *This* stands for *"Make Directory"*. This performs the same action as right clicking & creating a new folder.

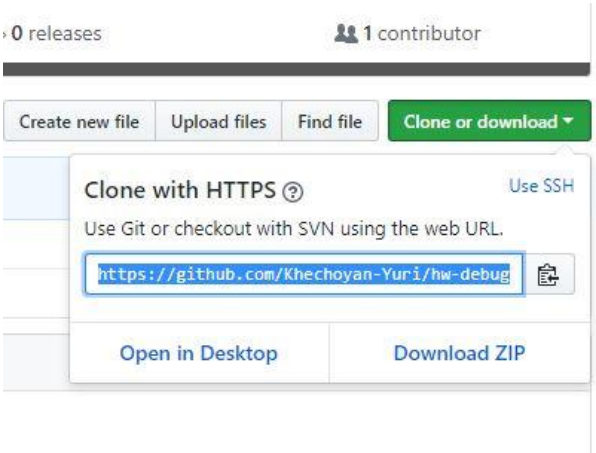
git init – This initializes the directory/folder that you are currently in to become a local Repository on your computer



git clone repository.git URL – This is used for the initial clone to have your local Repository match the one in GitHub. Hyperlink **MUST** have the .git extension. Once Clone command is executed, you will not use it until you create another local repo and match it to a different GitHub repo.

Obtaining the Repository Link

To obtain the repo link, on the main page of your project, there will be a green "Clone or Download" button. Click that and GitHub will show you which link you should use to clone, pull from, and push to.



pwd – Stands for *"Print Working Directory"*. This tells you where on your computer Git is currently looking, (which Directory)

ls – (Lowercase L) - This is used to "list" all contents in directory that Git is currently inside of..

(example: c/users/Username/Desktop)

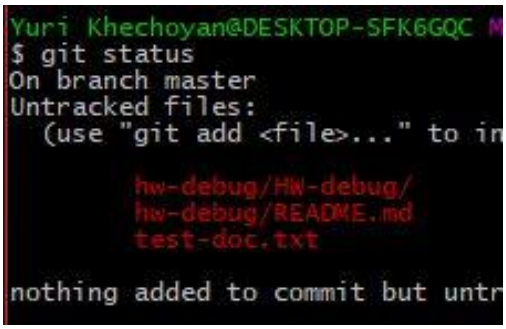
cd – Stands for *"Change Directory"*. This is used to enter into and out of folders (like you would with mouse clicks)

cd ..– (there is 1 space between the *d* and the first '.'): The double period after the cd allows you to move 1 directory backwards (example: if inside of FirstName/Documents, it will move you back into the FirstName/ directory).

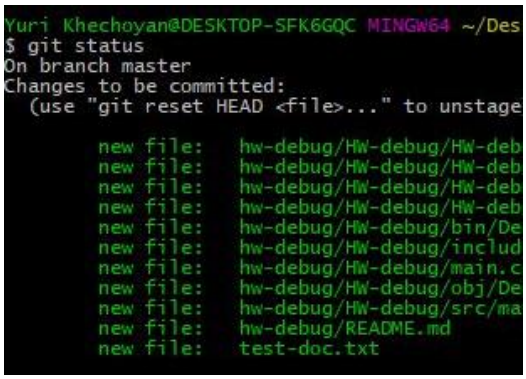
~ *– The tilde symbol let you to move directly into your profile folder on your computer*

It moves you to:
C:\Users\Username directory on your computer (it's a shortcut)

git – any major action will start with ‘git’ in front of it



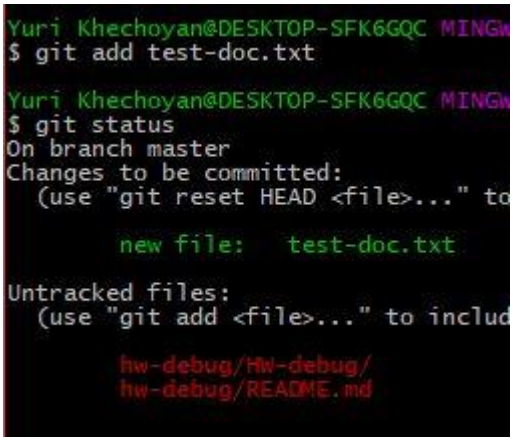
git status – gives you a status of your project. If files have been modified **BUT NOT** *add*ed to the staging area by **YOU** (not your team), it will let you know by showing your contents in red. Otherwise if files have been modified **AND** *add*ed to staging area by **YOU** (not your team), contents will show up in green. When added, they are added to the staging area.



exit – This does the same as the ‘X’ button in any program. It closes them. This is just the keyboard version for Git.

The Next few commands need to be done IN THIS ORDER TO UPLOAD (add, commit, push)

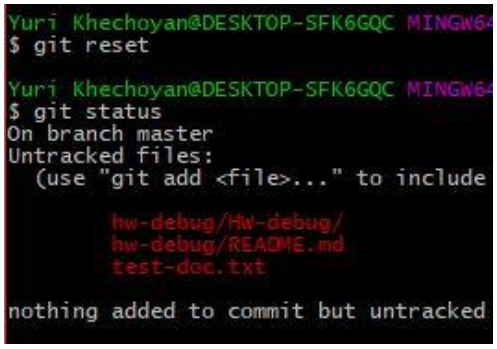
git add insert file name – This adds only specific files to the “staging area”. Make sure to include full name of file and extension.



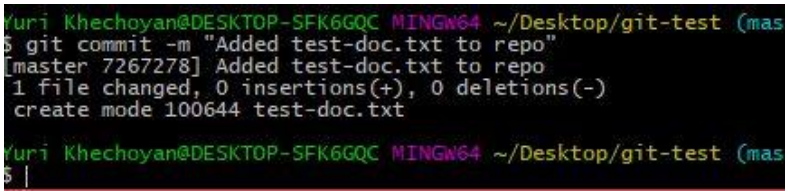
----------OR YOU CAN USE----------

git add .– the '.' that is placed 1 space after "add" means *"ALL FILES"*. This is much easier method of adding new/modified files to the staging area!

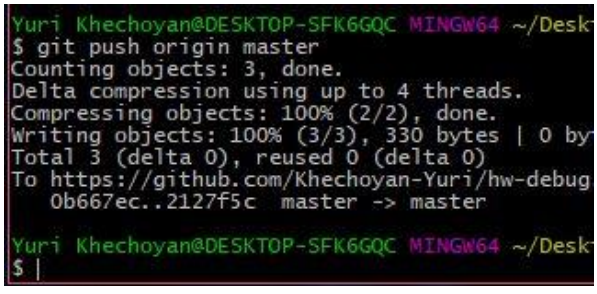
git reset – In the case where files were wrongfully added to the staging area, use this command to undo the *git add* command from earlier



git commit -m "Add your commit message here" – This lets Git know that at this time, this is the version you want to push to GitHub. There is a space between each major part (git, commit, -m, and the quotations). The *-m* is telling Git that there is a message that you want accompanied with your commit. Inside the quotations is where you would want to tell the people you are working with or even your future self what you did: i.e. added something, revised something, deleted something, fixed bugs, etc. **And keep the message short! No more than 1 sentence!!! If multiple things were done, separate what was done by a: ";" or "|"**



git push origin master – this is the final step needed in order to complete the upload. This command is actually needed in order to fully upload to GitHub. What this means is: I am pushing new or modified files/code to the origin of the master branch within my GitHub repository.



When other people make changes and you want an updated version

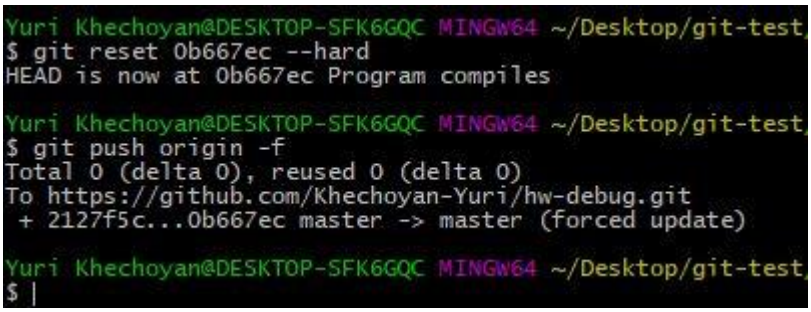
git pull repository.git link - This will be used to pull any new updates from the GitHub repository to your local Repo. Once you type in pull, right click with the mouse and paste the link (found below) then hit the ENTER key. Hyperlink **MUST** have the .git extension.

Instructions on how to delete the latest commit and revert back to the older version

git reset <commit # reverting back to> --hard

git push origin -f

Even though you can see all of the commits made separately, this command is powerful and dangerous all at the same time. Make sure that if you decide to use this, you use it responsibly. This code will delete any and ALL commits from the repo up to a certain commit. That is why you have to provide the commit # (7-character id). The latest version of the files is what is known as the "head". Think of it as the first person in line. When first person in line leaves, there is a new line leader. Every time you make a new commit, that



specific commit becomes the head. And every time you execute this command, the previous commit behind it becomes the new head.

NOTE: So, if let's say for an example - I made a mistake when uploading the latest commit on Nov. 7th ***(commit: 2127f5c)*** because I later found out something went terribly wrong in the code, what I would do is execute: *git reset 0b667ec --hard*. Followed by the execution: *git push origin -f*. *IF* the mistake is not that drastic, you can override it by updating the code and making another commit.

So, what I am telling Git here is: I made a mistake when making that commit, please completely delete it and go back to the version *0b667ec* of the code. Then I would like you to force push/override (*git push origin -f*) that change. Keep in mind, if you have Git go back further with your commits (for an example): go back to a commit made a year ago, **any and all progress made with commits after that point will be deleted and you will not be able to retrieve them!**

As you can see (above), the commit made on Nov. 7th is longer in existence. It has been successfully removed.

6.2 General Git Procedure

1. Type in: pwd so you know where Git is currently looking in
    a. Use the cd into ~ to quickly move into your profile folder on your computer
2. *cd* into wherever you would like to store the repo folder
3. Use *mkdir <Folder name>* if you haven't already created a folder to store your repo.
4. *cd* into the Repo folder that you have created in step 2.
5. Type in *git init* to initialize it.
6. Then use the *git clone* command (clone generally would be used at the very beginning. Then pull would be used every other time)
7. Once you have made any changes to the project, use the codes in this order (see above for exact code list):
    a. *git add (file name or .)*
    b. *git commit -m "message"*
    c. *git push origin master*

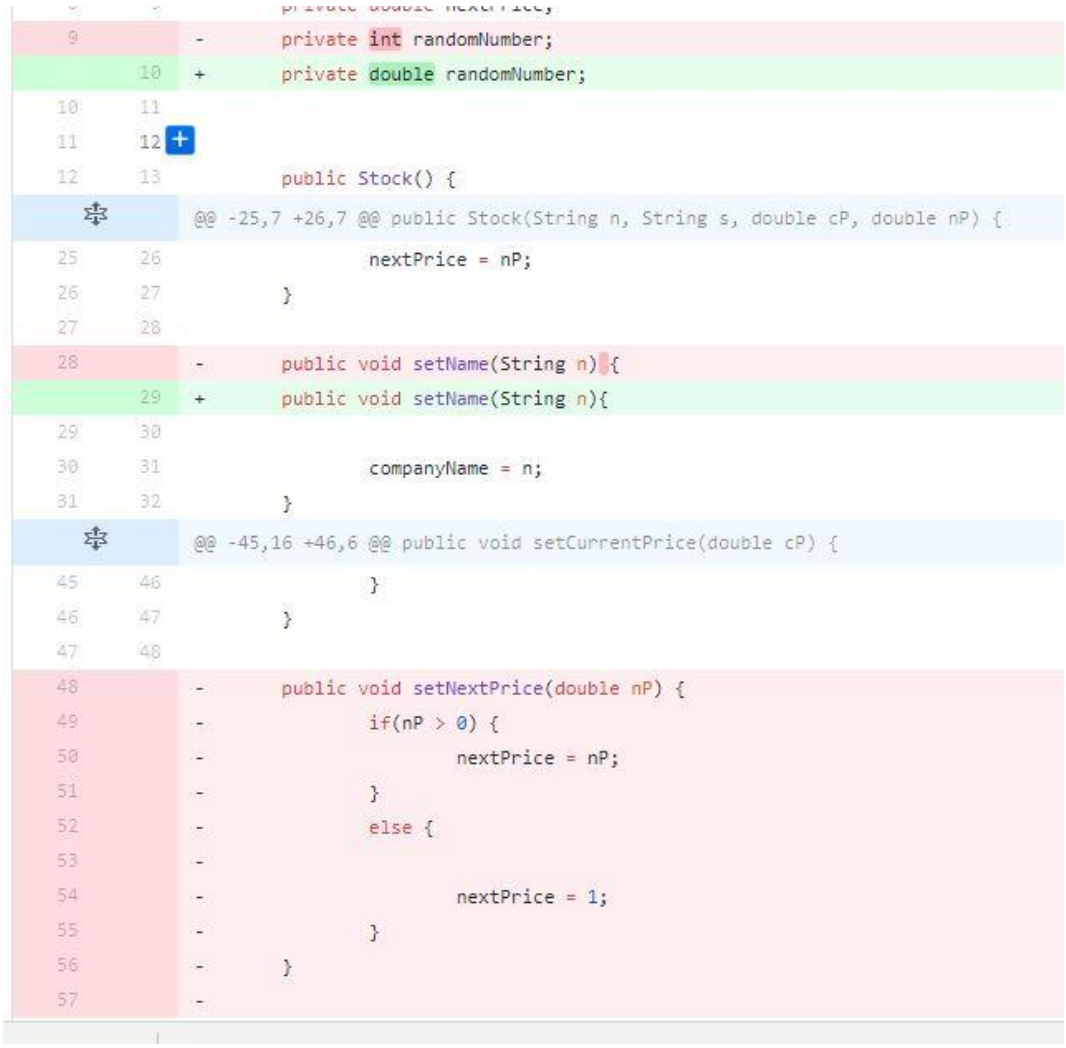
ONLY REPEAT THE PULL REQUEST WHEN ANOTHER CONTRIBUTOR HAS MADE A PUSH REQUEST

    d. *git pull* (when new changes have been made by other team members)

7. Analytics

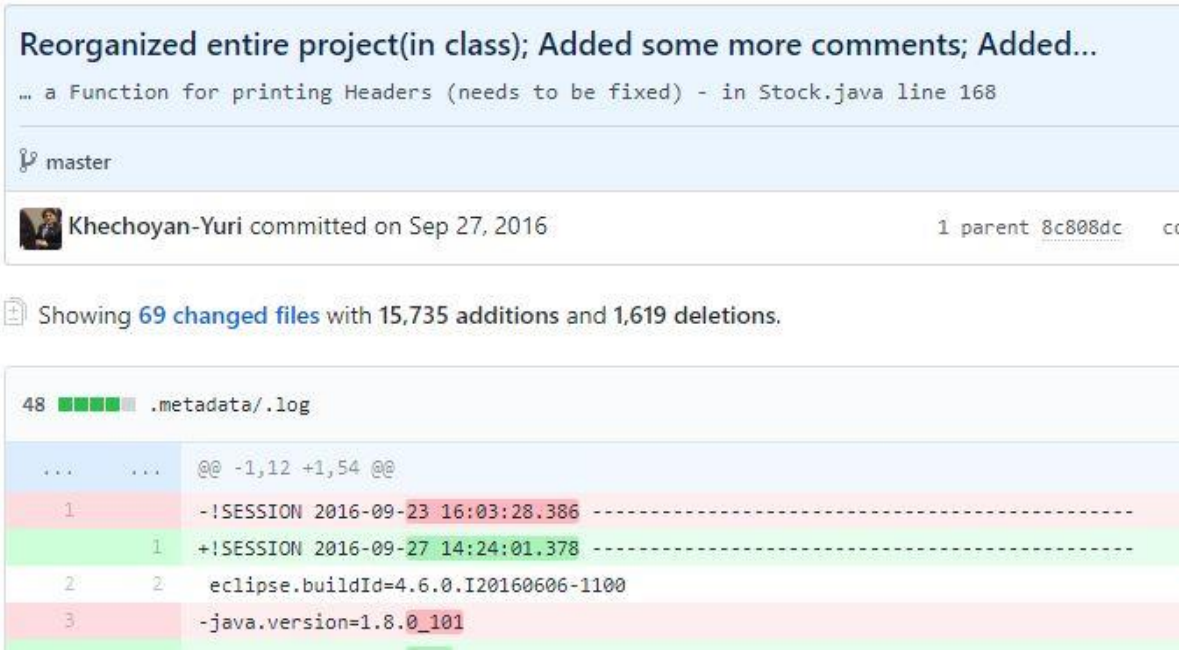
7.1 Crunching Numbers and Crushing Bugs (Code Analytics)

When working with and managing code, it is imperative to understand what has been done to the code in previous iterations (commits). When looking at the code after new commits have been pushed, everything that has been highlighted in green is all of the new code that has been added as part of that commit at that point in time. And subsequently, any code highlighted in red is all of the code that has been deleted for that specific commit at that given time. You can ignore the lines highlighted in blue.



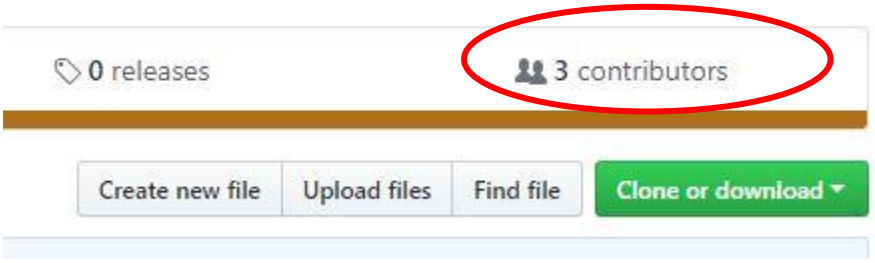
7.2 Git Commit Summary

Git will even summarize the events that occurred when you jump into the exact commit number. As you can see from the image, for a certain commit that I have made on Sept. 27, 2016 – I have made 15,735 additions and 1,619 deletions. The reason why I have modified so much is because the commit message reads: “Reorganized entire project.” That is the main thing that I did in this commit. If your commit message is too long, it will continue the message below the header [example: “… a Function for printing Headers (needs to be fixed)”]



7.3 Contributions Summary

When you want to see who has had the most amount of activity within a project, click on the **Contributors** link from the main directory of your repo.



Once accessed, you can see who has had the most activity within the project/repo.

7.4 What's Your Programming Poison? (language colors)

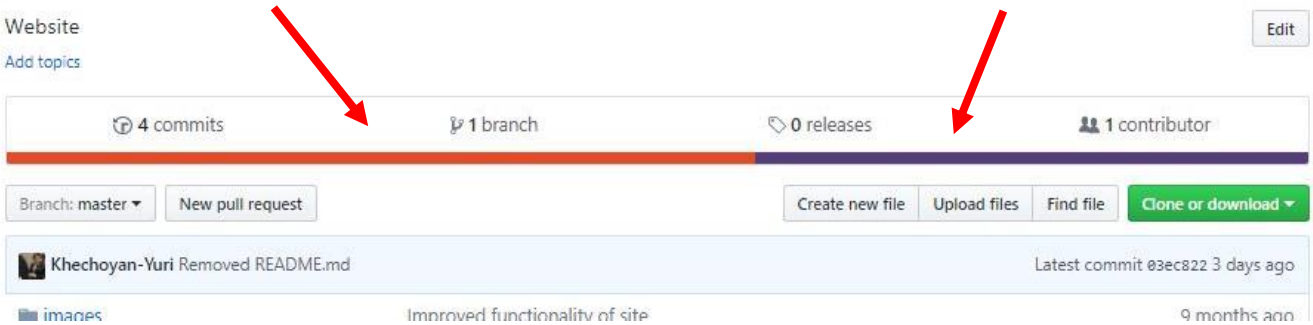
No matter what language you program in, GitHub has a way of identifying your source code language by the extension of those files. On GitHub, each language has a unique color identifier associated with it.

To see the full list of colors: Click Here

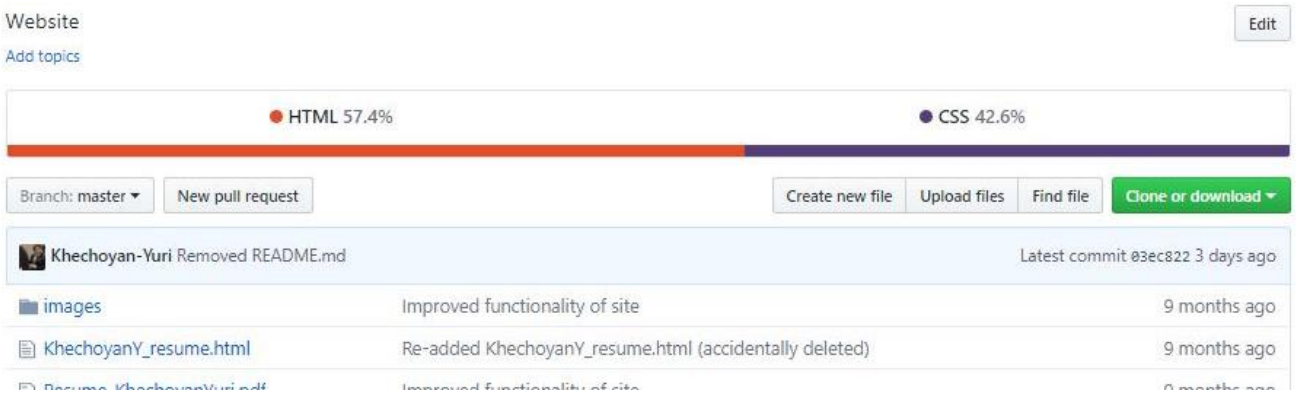
There are several ways for GitHub to let you know that it has identified your coding language(s) within projects.

Option 1: is in your profile under Repositories.

Option 2: If you are inside the main directory of your project, the bar going across; underneath your commits, branches, releases, and contributors is the color associated with the languages used within your project.



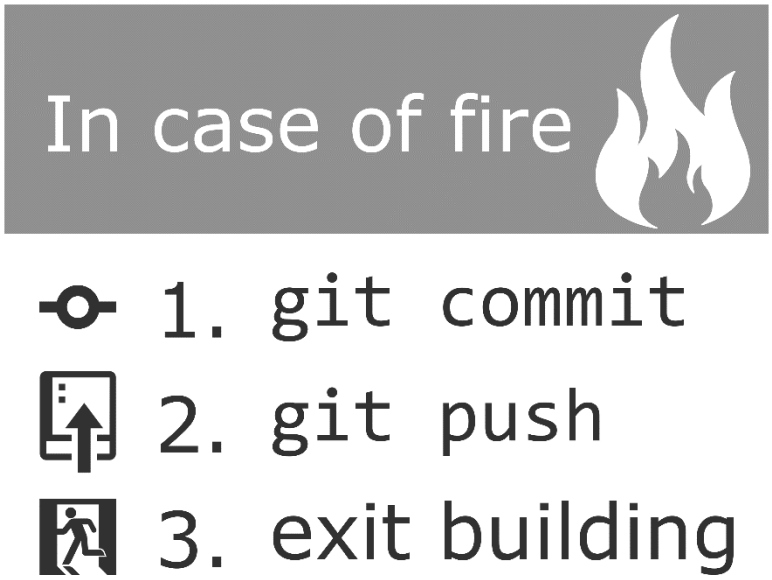
If you click anywhere on that bar, the bar will execute an animation, revealing what languages were used in the project. And as an added bonus, it will even show you what percentage of the project has been completed in that language!



8. End Notes

8.1 Fire Hazard Tips

When working with code, if there is ever a fire, remember...



8.2 Copyright

Copyright Act, 17 U.S.C. § 101 Definitions:

Except as otherwise provided in this title, as used in this title, the following terms and their variant forms mean the following:

A “computer program” is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.
“Copies” are material objects, other than phonorecords, in which a work is fixed by any method now known or later developed, and from which the work can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device. The term “copies” includes the material object, other than a phonorecord, in which the work is first fixed.
“Copyright owner”, with respect to any one of the exclusive rights comprised in a copyright, refers to the owner of that particular right.

A “device”, “machine”, or “process” is one now known or later developed.

A work is “fixed” in a tangible medium of expression when its embodiment in a copy or phonorecord, by or under the authority of the author, is sufficiently permanent or stable to permit it to be perceived, reproduced, or otherwise communicated for a period of more than transitory duration. A work consisting of sounds, images, or both, that are being transmitted, is “fixed” for purposes of this title if a fixation of the work is being made simultaneously with its transmission.

An “international agreement” is:
(1) the Universal Copyright Convention;
(2) the Geneva Phonograms Convention;
(3) the Berne Convention;
(4) the WTO Agreement;
(5) the WIPO Copyright Treaty;
(6) the WIPO Performances and Phonograms Treaty; and
(7) any other copyright treaty to which the United States is a party.

A “joint work” is a work prepared by two or more authors with the intention that their contributions be merged into inseparable or interdependent parts of a unitary whole.

For purposes of section 513, a “proprietor” is an individual, corporation, partnership, or other entity, as the case may be, that owns an establishment or a food service or drinking establishment, except that no owner or operator of a radio or television station licensed by the Federal Communications Commission, cable system or satellite carrier, cable or satellite carrier service or programmer, provider of online services or network access or the operator of facilities therefor, telecommunications company, or any other such audio or audiovisual service or programmer now known or as may be developed in the future, commercial subscription music service, or owner or operator of any other transmission service, shall under any circumstances be deemed to be a proprietor.

“Publication” is the distribution of copies or phonorecords of a work to the public by sale or other transfer of ownership, or by rental, lease, or lending. The offering to distribute copies or phonorecords to a group of persons for purposes of further distribution, public performance, or public display, constitutes publication. A public performance or display of a work does not of itself constitute publication.

To perform or display a work “publicly” means:
(1) to perform or display it at a place open to the public or at any place where a substantial number of persons outside of a normal circle of a family and its social acquaintances is gathered; or
(2) to transmit or otherwise communicate a performance or display of the work to a place specified by clause (1) or to the public, by means of any device or process, whether the members of the public capable of receiving the performance or display receive it in the same place or in separate places and at the same time or at different times.

“Registration”, for purposes of sections 205(c)(2), 405, 406, 410(d), 411, 412, and 506(e), means a registration of a claim in the original or the renewed and extended term of copyright.
“Sound recordings” are works that result from the fixation of a series of musical, spoken, or other sounds, but not including the sounds accompanying a motion picture or other audiovisual work, regardless of the nature of the material objects, such as disks, tapes, or other phonorecords, in which they are embodied.
“State” includes the District of Columbia and the Commonwealth of Puerto Rico, and any territories to which this title is made applicable by an Act of Congress.

A “transfer of copyright ownership” is an assignment, mortgage, exclusive license, or any other conveyance, alienation, or hypothecation of a copyright or of any of the exclusive rights comprised in a copyright, whether or not it is limited in time or place of effect, but not including a nonexclusive license.

A “transmission program” is a body of material that, as an aggregate, has been produced for the sole purpose of transmission to the public in sequence and as a unit.

To “transmit” a performance or display is to communicate it by any device or process whereby images or sounds are received beyond the place from which they are sent.

For purposes of section 411, a work is a “United States work” only if:

(1) in the case of a published work, the work is first published—
(A) in the United States;
(B) simultaneously in the United States and another treaty party or parties, whose law grants a term of copyright protection that is the same as or longer than the term provided in the United States;
(C) simultaneously in the United States and a foreign nation that is not a treaty party; or

(D) in a foreign nation that is not a treaty party, and all of the authors of the work are nationals, domiciliaries, or habitual residents of, or in the case of an audiovisual work legal entities with headquarters in, the United States;

(2) in the case of an unpublished work, all the authors of the work are nationals, domiciliaries, or habitual residents of the United States, or, in the case of an unpublished audiovisual work, all the authors are legal entities with headquarters in the United States; or

(3) in the case of a pictorial, graphic, or sculptural work incorporated in a building or structure, the building or structure is located in the United States.

A "useful article" is an article having an intrinsic utilitarian function that is not merely to portray the appearance of the article or to convey information. An article that is normally a part of a useful article is considered a "useful article".

The author's "widow" or "widower" is the author's surviving spouse under the law of the author's domicile at the time of his or her death, whether or not the spouse has later remarried.

The "WIPO Copyright Treaty" is the WIPO Copyright Treaty concluded at Geneva, Switzerland, on December 20, 1996.

The "WIPO Performances and Phonograms Treaty" is the WIPO Performances and Phonograms Treaty concluded at Geneva, Switzerland, on December 20, 1996.

Intellectual Property and Communications Omnibus Reform Act of 1999, as enacted by section 1000(a)(9) of Public Law 106-113, nor the deletion of the words added by that amendment:

(A) shall be considered or otherwise given any legal significance, or

(B) shall be interpreted to indicate congressional approval or disapproval of, or acquiescence in, any judicial determination, by the courts or the Copyright Office. Paragraph (2) shall be interpreted as if both section 2(a)(1) of the Work Made for Hire and Copyright Corrections Act of 2000 and section 1011(d) of the Intellectual Property and Communications Omnibus Reform Act of 1999, as enacted by section 1000(a)(9) of Public Law 106-113, were never enacted, and without regard to any inaction or awareness by the Congress at any time of any judicial determinations.