GitHub | What is Source Control?

Source control is a tool to manage source code of a project where there are multiple developers. Github is a tool to use specifically for source control.

Version Control is terminology that is thrown around a lot which is similar to source control, except it doesn't manage your source code, but your digital assets of your project. These 2 terms can be used together and are exchanged at times.

Recommended Tutorials:
https://youtu.be/fQLK8Ib_SKk - This tutorial series explains git well.
https://education.github.com/git-cheat-sheet-education.pdf - good cheat sheet by GitHub Education

Getting Started

For first time users, it's easier to use a GUI, graphical user interface, to understand the commands before graduating into the terminal world. If you are already comfortable in bash, linux, or terminal operations, using the command line may be easier than a GUI.

Recommended GUIs:

- GitHub Desktop
- GitKraken*

FlyUC Github Organization:
https://github.com/fly-uc

I recommend GitKraken as you can see the commit history nicely for when the projects get more collaborative.
If you go the command line route, just install Git Bash (Mac users, google the bash install) onto your computer. Ensure that all PATH variables and setup for your PC is done correctly. (LFS Support for larger projects, etc.)

Basic Source Control Terminology

Repository - Repository is the project itself. There are 2 types, remote and local. The remote is what can be accessed by all members, while the local version is the one you can only access.

Commit - Is a "revision", change to an individual file that you make

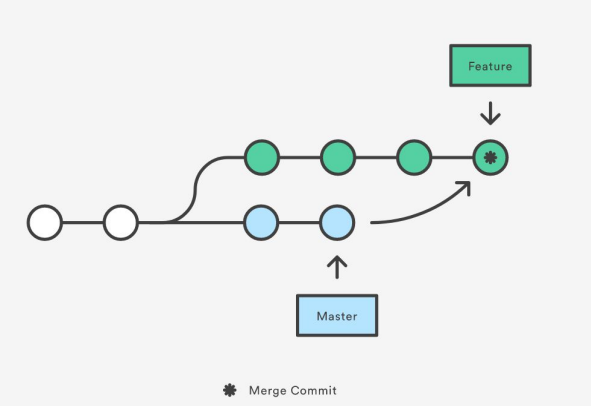
Branch - Acts like a pointer to a specific version of the project and commits made

Fetch - Similar to a refresh, it checks if the remote repository has any updates in comparison to your local status and notifies you

Pull - It downloads the commits from the remote onto your local branch and updates your project

Push - “Pushes” your commits to the remote repository (pushes to the same location of your local commit.) Ex: you made changes to your local branch called feature/fix-battery-calc and when you push it, the remote branch feature/fix-battery-calc updates with the commits you made. Now everyone can access the changes you made on that branch.

Merge - lets you take the independent lines of development created by git branch and integrate them into a single branch



Rebase - integrates changes from one branch into another. It is an alternative to the "merge" command. Rebase differs from merge by rewriting the commit history in order to produce a straight, linear succession of commits. **Usually use merge.**

Times to use Rebase - when a commit earlier in the master branch (not the most recent) needs to be changed and you want to **rewrite that commit history.** It’s important that you don’t use it often and on public branches easily. Try to use the merge command more than this as rebasing often is more complicated, a difficult concept to grasp.