**What is GitHub and version control?**

GitHub is a cloud-based website for the storing of files, accessible through git, which is version control software hosted through GitHub. Version control is the management of files in such a way that tracks changes and allows for easy rollbacks to earlier versions of files if necessary.

**What is a repo?**

GitHub organizes its content in repositories, or repos for short. Each repo contains files related to a given project, and the repo format allows for version control across all the files inside. The repo is stored remotely but can be cloned onto a local machine. Changes made on the local machine will not reflect on the repo unless those changes unless committed and pushed, allowing for an earlier version of files to be easily accessed, the essence of version control.

**What is the difference between local, remote, and origin?**

The remote branch is what can be found on GitHub; it is the repo, and all of its contents, being stored on the cloud. Usually, it will be the most up-to-date branch, but it can be kept behind the local branch for version control.

Rather than a branch, the origin is shorthand for the name of the git server a remote repo and its clones originate from. It is used in lieu of the repo URL, making referencing easier. The origin is defined from this URL upon initial cloning of the repo.

The local repo branch is stored on your computer, linked to the remote branch by the defined origin. Changes can be made here, but they will not reflect on the remote branch unless committed and pushed. Similarly, the local branch will not update on its own, so any changes must be pulled from the remote branch.

**The major actions and their meanings.**

Clone – Clone is the first action to do when accessing a GitHub repo. Cloning a repo puts that repo, and all its contents, onto your local machine, unaltered from the remote repo. Generally, using clone is only necessary the first time a repo is accessed and put onto a local machine.

Fork – Another means of accessing and editing a remote repo, one which creates a copy of a repo downstream from the original, allowing changes to be made without affecting the parent repo. This repo is created on your own GitHub account, with you as the author and owner. The forked repo can then be cloned/pulled locally, with any changes committed and pushed locally instead going to the downstream fork, leaving the original repo unaltered. At the same time, forked repos can be updated from the main branch and can push their changes onto the main branch, although the process is more complicated than pushing an pulling locally.

Stage – The first step after saving changes locally if you wish to update the remote repo, staging changes alerts GitHub that changes have been made, but these changes are not yet added to the repository. The add command is used to stage changes in the command line, while the reset command undoes changes should you wish not to commit them. Staging can be done selectively, on a file-by-file basis, allowing for finished files to be pushed remotely while keeping files that still need work as they are in the remote repository.

Commit – The commit action is used after saving and staging changes within a repo locally and is an intermediate step to transferring them to the remote repo. Specifically, the commit command identifies and records changes to the files on the local branch, in addition to when the changes were made and the user authoring the changes. Additionally, the author is required to include a brief message describing the changes. Every time you want to update the repo with changes made locally, the changes must be staged and committed before being pushed.

Push – Once local changes are committed to the main branch, they can then be pushed to the remote repository, updating it to the current committed state on a local machine. If there is a direct conflict between the local and remote file versions, the user will be alerted and asked which version they want in the final updated repo.

Pull – The opposite of push, this updates the local version of a repo based on the current version of the main remote branch. It is similar to clone, but it requires a local version of the repo to exist, and it is better to use once the local repo is created. Unlike the push command, no staging and commitment of changes is necessary in the remote repo prior to the pull.

Fetch – Fetching allows you to view changes made to the remote repo and compare the version with what is stored locally without overwriting the local files, which pull does. In essence, fetch only checks to see if an updated version of the repo is available remotely, while pull, after doing the same check, automatically updates the local repo and its files if updates are available.

Stage/commit

Push

Stage/commit

Staging area - intermediate

Fetch

Fork

Fetch

Remote – Local compare versions

Clone/pull

Push

Staging area - intermediate

Clone/pull

Local Branch – downstream copy

Local Branch – main origin

Remote Branch – downstream copy

Remote Branch – main origin