**Git Cheat Sheet**

**Git**

* An open-source program for tracking changes in text files. It was written by the author of the Linux operating system and is the core technology that GitHub, the social and user interface, is built upon.

**Commit**

* An individual change to a file (or set of files).
* It's similar to when you save a file, except that, with Git, every time you save, it creates a unique ID (a.k.a., the "SHA" or "hash") that allows you to keep a record of what changes were made when and by whom.
* Commits usually contain a commit message — a brief description of what changes were made.
* *Synonym*: a revision

**Diff**

* A diff is the difference in changes between two commits (saved changes).
* The diff will visually describe what was added or removed from a file since its last commit.

**Remote**

* The version of something that is hosted on a server, most likely GitHub.com. It can be connected to local clones so that changes can be synced.

**Repository**

* The most basic element of Git.
* A repository is a project's folder that contains all of the project files (including documentation) and stores each file's revision history. Repositories can have multiple collaborators and can be either public or private.

**Fork**

* A personal copy of another user's repository that lives on your account.
* Forks allow you to freely make changes to a project without affecting the original.
* Forks remain attached to the original, allowing you to submit a pull request to the original's author to update it with your changes.

**Clone**

* A copy of a repository that lives on your computer instead of on a website's server, or the act of making that copy.
* With your clone, you can edit the files in your preferred editor and use Git to keep track of your changes without having to be online.
* A clone, however, is connected to the remote version so that changes can be synced between the two.
* You can push your local changes to the remote version to keep them synced when you're online.

**Push**

* Pushing refers to sending your committed changes to a remote repository such as GitHub.com.

| **Command** | **Description** |
| --- | --- |
| git init | Tells Git to start monitoring the current folder I'm in. In other words, for my working directory, create a new "timeline" where I can manage my source code. |
| git status | Gets the current status of Git. Files that are "staged" (about to be committed) and files that are unstaged (files that have changed since the last commit but are not about to be committed) will both show up here. |
| git add path/to/directory/or/file | Adds a file to the "stage." The stage can be thought of as an in-between state between the last commit and what is ready to be committed. Once we run the command git commit, all staged files will be committed to the timeline. |
| git commit -m "Commit message" | Commits all staged files to the timeline. If -m "Commit message" is committed, Git will open your default text editor (typically Vim) to enter a longer message. If for any reason Vim opens, you can close it by typing :q. |
| git log | Visualizes the timeline. You can scroll with the arrow keys (or jand k) and exit the view by typing q. |
| git diff path/to/directory/or/file | Shows the tracked but unstaged changes of the given file or directory. |
| git clone http://path/to/repo | Creates a new local git repo copied from a remote one. |
| git push origin master | Sends local changes to the tracked remote repository. |

* For instance, if you change something locally, you'd want to then push those changes so that others may access them.