From the first Git tutorial video alone, I was able to pick out and discover more features about command-line Git than I have ever used. Considering that my only ever use for Git was pushing, pulling, and committing to a repository, I think the content that I took away from the video is substantial.

Starting out, I found out that Git is not just some database that holds many repositories for later use. It's structure is called the "distrubuted version control system." This means that a Git server holds a repository and its versions, which it then distributes to appropriate users. Users use the "git clone" command to do this (assuming that the repository is available to those users). Every user that pulls from that repository effectively has their own version that they can edit before pushing those changes back into the repository in the server. If there are any discrepencies between the versions users push into the repository, Git has commands that aid in finding and resolving those discrepencies through the command line. This, of course, obviously does not account for users communicating amongst themselves about the changes they have made. Nonetheless, Git heavily assists in resolving any repository push errors.

I was also intrigued by the concept that using Git involves three main areas: the working directory (our computers), the staging area (Git), and the Git repository (the Git servers). Command-line Git, from what I understand, works in the "staging area." We make changes to a file, commit the changes to a staging area, then finally pushing those changes into the repository server. I have found that with these three areas in play, Git allows users to accurately spot discrepencies and errors between file versions just like I mentioned above.

The rest of the content I have discovered I could simply have in one giant list. The majority of them being strings of commands that apply a variety of functions for the user. The concept of merging files, for example, had allows eluded me, especially with me trying to understand the order of the Git commands. From what I understand, the best way to apply a merge is to create a new branch, do whatever changes I need in that branch, then merge it with the "main" branch of my repository. All of this must be done before I commit said changes to the server repository.

Git seems to be an exceptionally streamlined way to execute version changes. I can dare say that once I get accustomed to using Git, I won't have to be so dependent on the GitHub Desktop application!