Hello world, and welcome to my short report on a multi-version collaboration system called Git. If you’re an average joe, at this point you may be wondering “what the crap is Git?”; if you’re one of my professors you already know what the crap Git is, and are waiting patiently for me to explain it to you. In this report, I will cover the basics of Git, staging, committing, and of course pushing and pulling. Again, I know those phrases might not make sense now, but hopefully they do by the end of this.

First, we need to understand a bit more about Git before we dive deeper into the complexities of staging, committing, pushing, and pulling. Basically, Git is a multi-version production system devised to help users to collaborate on digital projects (stories, code, movies, etc.). When I say *multi-version production system* I refer to the ability that Git users have to document and save multiple versions of the files they work on in an organized way.

Git is organized into multiple branches. There is a master branch which demonstrates the current state of any given project being developed. This branch often contains different file types. Once a collaborator on that project sees a revision that can be made, they create a new branch. This branch is basically a clone of the master branch with the difference being that any changes made to their clone branch won’t affect the state of the master branch. That is, until they stage, commit, and push their changes.

Staging and committing is the process of saving and documenting any changes a collaborator has made to their cloned branch. Here’s how it works. A collaborator makes a change to a file from the cloned master branch. Git recognizes this change and presents it as an un-staged file or a work-in-progress file. Once the file is no longer a WIP the collaborator then stages the file, or in other words prepares it to be committed to the branch. This is done by tagging the file with a brief description as to what has been changed. In order to save this change to the master branch the collaborator must send these changes to the master branch by pushing them.

Pushing and pulling commits is an essential part of Git and all of its parts. Pushing and pulling also has a lot to do with two things called Git hub and Git Kraken. Git Hub is a server where the master branch, and all of its subsequent branches are stored. Git Kraken is a program that demonstrates these branches and their files graphically on a collaborator’s local machine. Git Kraken provides users the ability make new branches and edit files easily. However, the changes they make in Git Kraken are only made locally. They have to send them or push these changes to the Git Hub server so that the other collaborators and pull them from the server to their local machines. Once pulled others can continue to work on them. When they’re finished with their changes they push them to the server and the work continues.

There you have it a somewhat brief explanation of Git, Git Hub, and Git Kraken. Hopefully I was able to explain it all in a clear and understandable way. If not google is a great resource for answering any questions you may have.