**Staging Area:**A place where we can group files together before we "commit" them to Git.

**Commit** A "commit" is a snapshot of our repository. This way if we ever need to look back at the changes we've made (or if someone else does), we will see a nice timeline of all changes.

**Wildcards:**We need quotes so that Git will receive the wildcard before our shell can interfere with it. Without quotes our shell will only execute the wildcard search within the current directory. Git will receive the list of files the shell found instead of the wildcard and it will not be able to add the files inside of the octofamily directory.

**git remote:**Git doesn't care what you name your remotes, but it's typical to name your main one origin.

It's also a good idea for your main repository to be on a remote server like [GitHub](http://github.com/) in case your machine is lost at sea during a transatlantic boat cruise or crushed by three monkey statues during an earthquake.

**git stash:**Sometimes when you go to pull you may have changes you don't want to commit just yet. One option you have, other than commiting, is to stash the changes.

Use the command 'git stash' to stash your changes, and 'git stash apply' to re-apply your changes after your pull.

**HEAD**The HEAD is a pointer that holds your position within all your different commits. By default HEAD points to your most recent commit, so it can be used as a quick way to reference that commit without having to look up the SHA.

**Commit Etiquette:**You want to try to keep related changes together in separate commits. Using 'git diff' gives you a good overview of changes you have made and lets you add files or directories one at a time and commit them separately.

1.17 Undo

git reset did a great job of unstaging octodog.txt, but you'll notice that he's still there. He's just not staged anymore. It would be great if we could go back to how things were before octodog came around and ruined the party.

Files can be changed back to how they were at the last commit by using the command: git checkout -- <target>. Go ahead and get rid of all the changes since the last commit for octocat.txt

git checkout -- octocat.txt

**The '--'**So you may be wondering, why do I have to use this '--' thing? git checkout seems to work fine without it. It's simply promising the command line that there are no more options after the '--'. This way if you happen to have a branch named octocat.txt, it will still revert the file, instead of switching to the branch of the same name.