***Git:***

git init

git status

git add <filename>

git add '\*.tx­t' % You also can use wildcards if you want to add many files of the same type.

git add . %add all: You can also type git add .. The dot represents the current directory, so everything in it, and everything beneath it gets added.

git reset %You can use git reset <filename> to remove a file or files from the staging area.

git commit –a -m ’added new benchmarks’ %makes Git automatically stage every file that is already tracked before doing the commit, letting you skip the git add part

git commi­t -m "Add cute octoc­at story­"

git log --summary % see more information for each commit. You can see where new files were added for the first time or where files were deleted. It's a good overview of what's going on in the project.

git log p -2 %options is -p, which shows the diff introduced in each

commit. You can also use -2, which limits the output to only the last two entries

gitk %You can see the commit history in the top half of the window along with a nice ancestry graph. The diff viewer in the bottom half of the window shows you the changes introduced at any commit you click.

git remote add origin <https://github.com/happy-face/try_git.git>

git push -u origin master

git remote –v %It lists the shortnames of each remote handle you’ve specified and the URL that Git has stored for the shortname to be expanded to.

git remote show origin % It lists the URL for the remote repository as well as the tracking branch information. The command helpfully tells you that if you’re on the master branch and you run git pull, it will automatically merge in the master branch on the remote after it fetches all the remote references. It also lists all the remote references it has pulled down.

git remote rename pb paul

git remote rm paul

git pull origi­n maste­r

git stash

git diff HEAD %diff of our most recent commit, which we can refer to using the HEAD pointer.

git diff --sta­ged % This command compares your staged

changes to your last commit

git diff %That command compares what is in your working directory with what is in your staging area.

git reset­ octof­amily/octo­dog.txt

git check­out -- octoc­at.txt %Files can be changed back to how they were at the last commit

%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.

git branc­h clean­\_up % create branch "clean\_up"

git checkout <branch> %switch between branches

git checkout -b new\_branch %to checkout and create a branch at the same time. This is the same thing as doing:git branch new\_branch,git checkout new\_branch

git rm '\*.tx­t' %not only remove the actual files from disk, but will also stage the removal of the files for us.

git rm -r folder\_of\_cats %This will recursively remove all folders and files from the given directory.

git rm --cached readme.txt %keep the file in your working tree

but remove it from your staging area

git mv file\_from file\_to %rename a file in Git

git commit -m "Delete stuff"

git commit -am "Delete stuff" %If you happen to delete a file without using 'git rm' you'll find that you still have to 'git rm' the deleted files from the working tree. You can save this step by using the '-a' option on 'git commit', which auto removes deleted files with the commit.

git merge­ clean­\_up

git branch -d <branch name> % delete a branch which is merged

git branch -D <branch name> % force delete a branch which is not merged

git clone git://github.com/schacon/grit.git mygrit

$ git help <verb>

$ git <verb> --help

$ man git-<verb>

$ git config --list

**Changing Your Last Commit**

$ git commit -m 'initial commit'

$ git add forgotten\_file

$ git commit --amend

**Create a new repository on the command line:**

touch README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin <https://github.com/happy-face/try_git.git>

git push -u origin master

**Push an existing repository from the command line**

git remote add origin <https://github.com/happy-face/try_git.git>

git push -u origin master

**Force delete**

What if you have been working on a feature branch and you decide you really don't want this feature anymore? You might decide to delete the branch since you're scrapping the idea. You'll notice that git branch -d bad\_feature doesn't work. This is because -d won't let you delete something that hasn't been merged.

You can either add the --force (-f) option or use -D which combines -d -f together into one command.

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.

test