GoldbergGregGitTutorial-05-31-2016

**Part 3**

**What is GitHub:**

GitHub is a web-based Git repository hosting service. It offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. Unlike Git, which is strictly a command-line tool, GitHub provides a Web-based graphical interface and desktop as well as mobile integration.

**When was it created:**

February 8th, 2008

**Why:**To provide a graphical user interface for Git. This allows for greater ease of use especially for newer developers. In addition to a lowering barriers for use, Github provides several collaboration features such as bug tracking, feature requests, task management, and wikis for every project. Making it a perfect tool for collaborating development teams.

**By who:**

Tom Preston-Werner

Chris Wanstrath

PJ Hyett

**What similar platforms exist:**

GitLab, BitBucket, Stash, Beanstalk, SourceForge, GoogleCode, CodePlex

**Why would you use such a platform:**

Github allows for incredibly easy collaboration, which is paramount for open source programming. It’s easy to offer suggested changes to others’ code through Github. Version control systems keep these revisions straight, and store the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute. This increases the ability to produce better code and thus a better product.

**Part 4**

Press enter to submit commands

> git init

Initialized empty Git repository in /.git/

Success!

$ git status

# On branch master  
#  
# Initial commit  
#  
nothing to commit (create/copy files and use "git add" to track)

Success!

$ git status

# On branch master  
#  
# Initial commit  
#  
# Untracked files:  
# (use "git add <file>..." to include in what will be committed)  
#  
# octocat.txt  
nothing added to commit but untracked files present (use "git add" to track)

Success!

$ git add octocat.txt

Nice job, you've added octocat.txt to the Staging Area

$ git status

# On branch master  
#  
# Initial commit  
#  
# Changes to be committed:  
# (use "git rm --cached <file>..." to unstage)  
#  
# new file: octocat.txt  
#

Success!

$ git commit -m "Add cute octocat story"

[master (root-commit) 20b5ccd] Add cute octocat story  
1 file changed, 1 insertion(+)  
create mode 100644 octocat.txt

Success!

$ git add '\*.txt'

Success!

$ git commit -m 'Add all the octocat txt files'

[master 3852b4d] Add all the octocat txt files  
4 files changed, 4 insertions(+)  
create mode 100644 blue\_octocat.txt  
create mode 100644 octofamily/baby\_octocat.txt  
create mode 100644 octofamily/momma\_octocat.txt  
create mode 100644 red\_octocat.txt

Success!

$ git log

commit 3852b4db1634463d0bb4d267edb7b3f9cd02ace1  
Author: Try Git <try\_git@github.com>  
Date: Sat Oct 10 08:30:00 2020 -0500  
  
Add all the octocat txt files  
  
commit b652edfd888cd3d5e7fcb857d0dabc5a0fcb5e28  
Author: Try Git <try\_git@github.com>  
Date: Sat Oct 10 08:30:00 2020 -0500  
  
Added cute octocat story

Success!

$ git remote add origin https://github.com/try-git/try\_git.git

Success!

$ git push -u origin master

Branch master set up to track remote branch master from origin.

Success!

$ git pull origin master

Updating 3852b4d..3e70b0f  
Fast-forward  
yellow\_octocat.txt | 1 +  
1 file changed, 1 insertion(+)  
create mode 100644 yellow\_octocat.txt

Success!

$ git diff HEAD

diff --git a/octocat.txt b/octocat.txt  
index 7d8d808..e725ef6 100644  
--- a/octocat.txt  
+++ b/octocat.txt  
@@ -1 +1 @@  
-A Tale of Two Octocats  
+[mA Tale of Two Octocats and an Octodog

Success!

$ git add octofamily/octodog.txt

Success!

$ git diff --staged

diff --git a/octofamily/octodog.txt b/octofamily/octodog.txt  
new file mode 100644  
index 0000000..cfbc74a  
--- /dev/null  
+++ b/octofamily/octodog.txt  
@@ -0,0 +1 @@  
+[mwoof

Success!

$ git reset octofamily/octodog.txt

Success!

$ git checkout -- octocat.txt

Success!

$ git branch clean\_up

Success!

$ git checkout clean\_up

Switched to branch 'clean\_up'

Success!

$ git rm '\*.txt'

rm 'blue\_octocat.txt'  
rm 'octocat.txt'  
rm 'octofamily/baby\_octocat.txt'  
rm 'octofamily/momma\_octocat.txt'  
rm 'red\_octocat.txt'

Success!

$ git status

# On branch clean\_up  
# Changes to be committed:  
# (use "git reset HEAD <file>..." to unstage)  
#  
# deleted: blue\_octocat.txt  
# deleted: octocat.txt  
# deleted: octofamily/baby\_octocat.txt  
# deleted: octofamily/momma\_octocat.txt  
# deleted: red\_octocat.txt  
#  
# Untracked files:  
# (use "git add <file>..." to include in what will be committed)  
#  
# octofamily/

Did not use git commit

$ git commit -m "Remove all the cats"

[clean\_up 63540fe] Remove all the cats  
5 files changed, 5 deletions(-)  
delete mode 100644 blue\_octocat.txt  
delete mode 100644 octocat.txt  
delete mode 100644 octofamily/baby\_octocat.txt  
delete mode 100644 octofamily/momma\_octocat.txt  
delete mode 100644 red\_octocat.txt

Success!

$ git checkout master

Switched to branch 'master'

Success!

$ git merge clean\_up

Updating 3852b4d..ec6888b  
Fast-forward  
blue\_octocat.txt | 1 -  
octocat.txt | 1 -  
octofamily/baby\_octocat.txt | 1 -  
octofamily/momma\_octocat.txt | 1 -  
red\_octocat.txt | 1 -  
5 files changed, 5 deletions(-)  
delete mode 100644 blue\_octocat.txt  
delete mode 100644 octocat.txt  
delete mode 100644 octofamily/baby\_octocat.txt  
delete mode 100644 octofamily/momma\_octocat.txt  
delete mode 100644 red\_octocat.txt

Success!

$ git branch -d clean\_up

Deleted branch clean\_up (was ec6888b).

Success!

$ git push

To https://github.com/try-git/try\_git.git  
3e70b0f..4c19eb6 master -> master

Success!

>

**Part 5**

Define the following terms in the context of Git (2 lines maximum):

* Repository - contains all of the project files (including documentation), and stores each file's revision history.
* Commit - is an individual change to a file (or set of files). It's like when you save a file, except with Git, every time you save it creates a unique ID that allows you to keep record of what changes were made when and by who.
* Push - refers to sending your committed changes to a remote repository such as GitHub.com. For instance, if you change something locally, you'd want to then push those changes so that others may access them.
* Branch - a parallel version of a repository. It is contained within the repository, but does not affect the primary or master branch allowing you to work freely without disrupting the "live" version.
* 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.
* Merge - take the changes from one branch (in the same repository or from a fork), and applies them into another. This often happens as a Pull Request (which can be thought of as a request to merge), or via the command line
* Clone - a copy of a repository that lives on your computer instead of on a website's server somewhere, 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.
* Pull - refers to when you are fetching in changes and merging them. For instance, if someone has edited the remote file you're both working on, you'll want to pull in those changes to your local copy so that it's up to date.

**Part 7**

* Forked <https://github.com/paceuniversity/courses>
* In my repo, clicked pencil icon to edit the readme.md
* Added my name and date to readme file
* Added commit comment with my name and date
* Chose Create a new branch for this commit and start pull request\*
* Added comment with my name and date
* Merged pull request to master branch\*
* Went to <https://github.com/paceuniversity/courses>
* New pull request
* Compare Across Forks
* Selected my version of master from my repo
* Added comment and created pull request

\*Probably should of chose commit directly to master branch