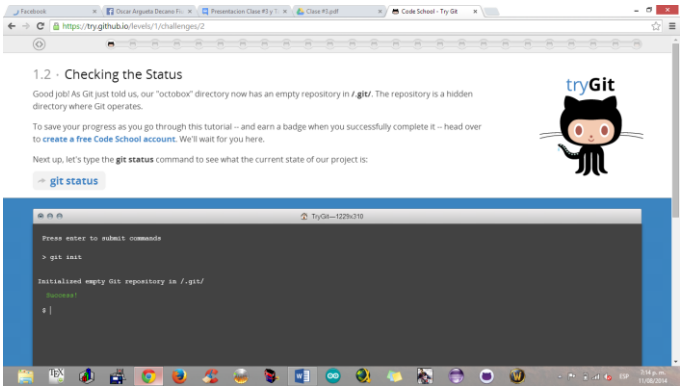
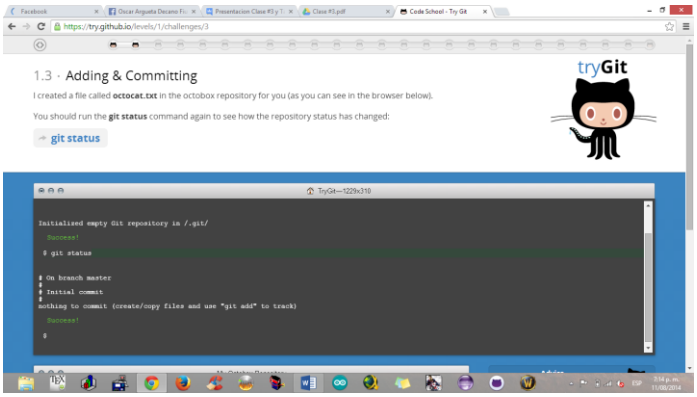
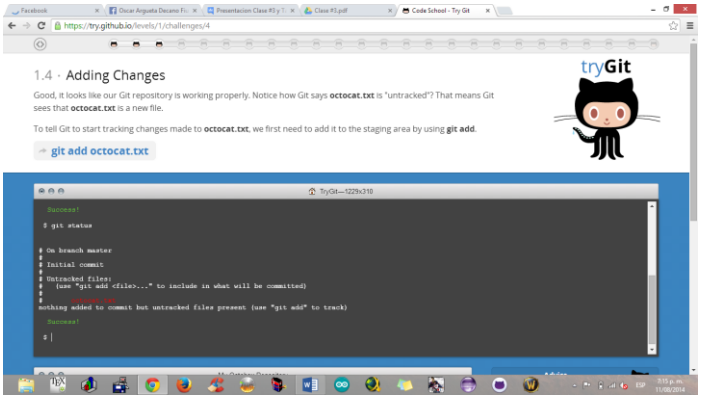
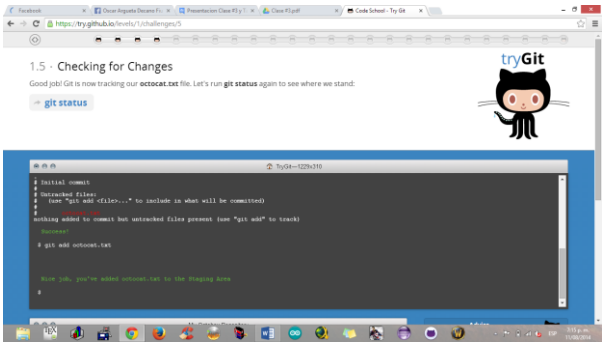
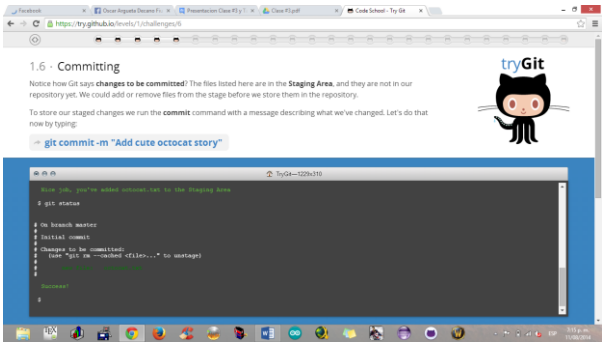
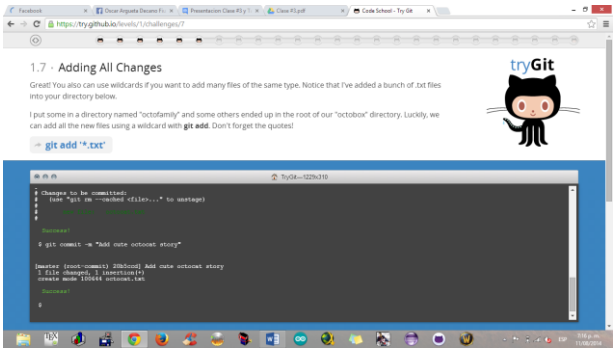

https://try.github.io/levels/1/challenges/2
1.2 · Checking the Status
Good job! As Git just told us, our "octobox" directory now has an empty repository in /.git/. The repository is a hidden directory where Git operates.
To save your progress as you go through this tutorial -- and earn a badge when you successfully complete it -- head over to create a free Code School account. We'll wait for you here.
Next up, let's type the git status command to see what the current state of our project is:
git status
tryGit
Press ctrl+c to submit commands
> git init
Initialized empty Git repository in /.git/
Success!
$

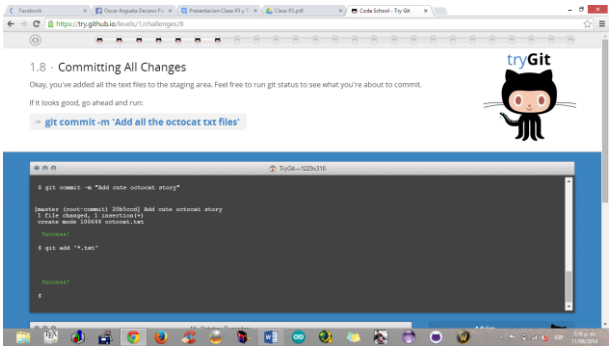

https://try.github.io/levels/1/challenges/3
1.3 · Adding & Committing
I created a file called octocat.txt in the octobox repository for you (as you can see in the browser below).
You should run the git status command again to see how the repository status has changed:
git status
tryGit
Initialized empty 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!
$

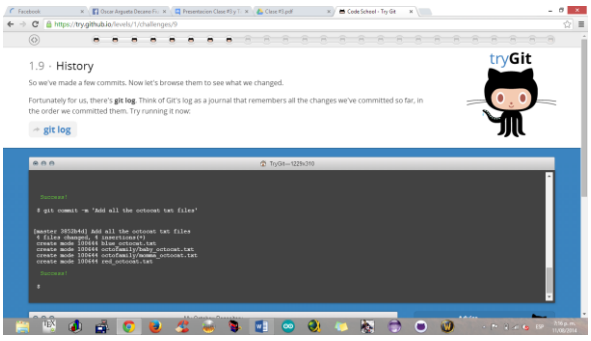

https://try.github.io/levels/1/challenges/4
1.4 · Adding Changes
Good, it looks like our Git repository is working properly. Notice how Git says octocat.txt is "untracked"? That means Git sees that octocat.txt is a new file.
To tell Git to start tracking changes made to octocat.txt, we first need to add it to the staging area by using git add.
git add octocat.txt
tryGit
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!
$

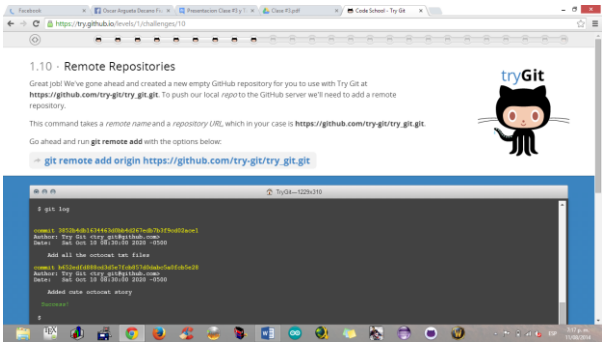

https://try.github.io/levels/1/challenges/5
1.5 · Checking for Changes
Good job! Git is now tracking our octocat.txt file. Let's run git status again to see where we stand:
git status
tryGit
Initial commit
#
Untracked files:
(use "git add <file>..." to include in what will be committed)
#
#	octocat.txt
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
$

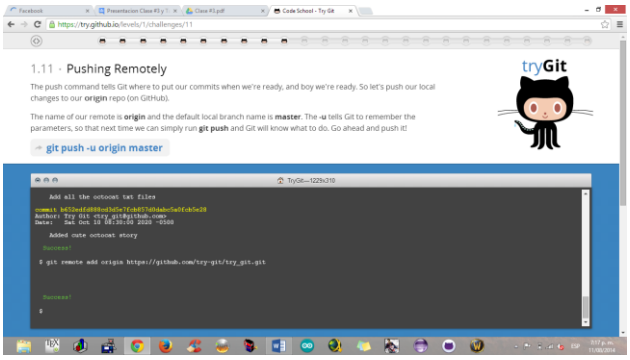

1.6 · Committing
tryGit
Notice how Git says changes to be committed? The files listed here are in the Staging Area, and they are not in our repository yet. We could add or remove files from the stage before we store them in the repository.
To store our staged changes we run the commit command with a message describing what we've changed. Let's do that now by typing:
git commit -m "Add cute octocat story"
$ git status
# On branch master
# Initial commit
# Changes to be committed:
# (use "git rm --cached <file>..." to unstage)
Success!

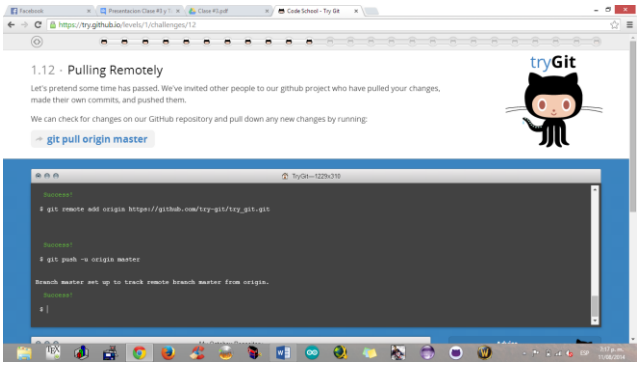

1.7 · Adding All Changes
tryGit
Great! You also can use wildcards if you want to add many files of the same type. Notice that I've added a bunch of .txt files into your directory below.
I put some in a directory named "octofamily" and some others ended up in the root of our "octobox" directory. Luckily, we can add all the new files using a wildcard with git add. Don't forget the quotes!
git add '*.txt'
Changes to be committed:
(use "git rm --cached <file>..." to unstage)
Success!
$ git commit -m "Add cute octocat story"
[master (root-commit) 28b5ced] Add cute octocat story
1 file changed, 1 insertion(+)
create mode 100644 octocat.txt
Success!

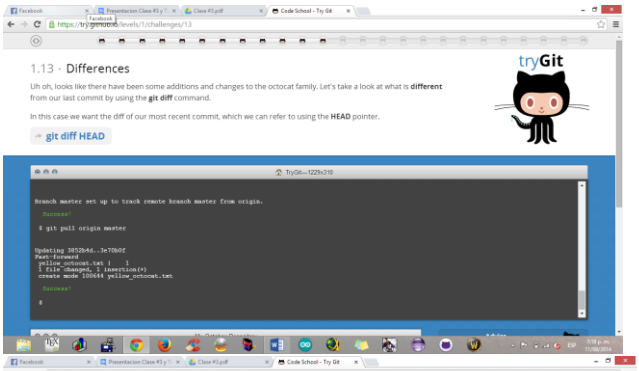

1.8 · Committing All Changes
tryGit
Okay, you've added all the text files to the staging area. Feel free to run git status to see what you're about to commit.
If it looks good, go ahead and run:
git commit -m 'Add all the octocat txt files'
$ git commit -m "Add cute octocat story"
[master (root-commit) 28b5ced] Add cute octocat story
1 file changed, 1 insertion(+)
create mode 100644 octocat.txt
Success!
$ git add '*.txt'
Success!

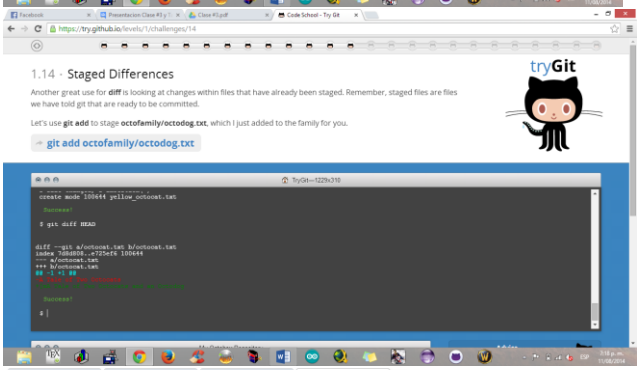

1.9 · History
tryGit
So we've made a few commits. Now let's browse them to see what we changed.
Fortunately for us, there's git log. Think of Git's log as a journal that remembers all the changes we've committed so far, in the order we committed them. Try running it now.
git log
Success!
$ git commit -m 'Add all the octocat txt files'
[master 3052b4d] 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!

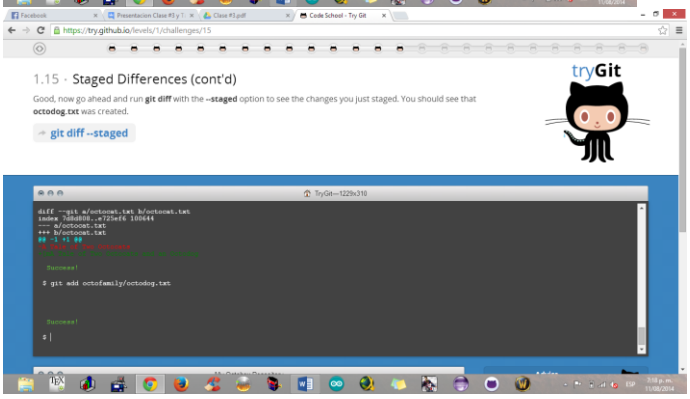

1.10 · Remote Repositories
Great job! We've gone ahead and created a new empty GitHub repository for you to use with Try Git at https://github.com/try-git/try_git.git. To push our local repo to the GitHub server we'll need to add a remote repository.
This command takes a remote name and a repository URL which in your case is https://github.com/try-git/try_git.git.
Go ahead and run git remote add with the options below:
git remote add origin https://github.com/try-git/try_git.git
tryGit
$ git log
Add all the octocat tat files
Added cute octocat story
Success!

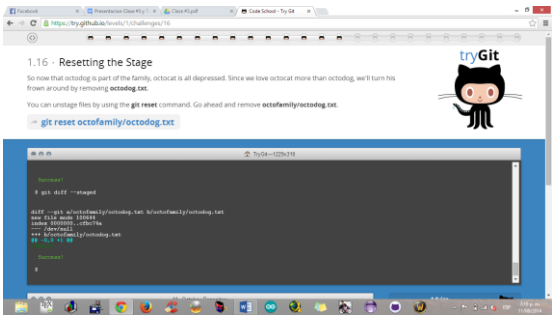

1.11 · Pushing Remotely
The push command tells Git where to put our commits when we're ready, and boy we're ready. So let's push our local changes to our origin repo (on GitHub).
The name of our remote is origin and the default local branch name is master. The -u tells Git to remember the parameters, so that next time we can simply run git push and Git will know what to do. Go ahead and push it!
git push -u origin master
tryGit
Add all the octocat tat files
Added cute octocat story
Success!
$ git remote add origin https://github.com/try-git/try_git.git
Success!

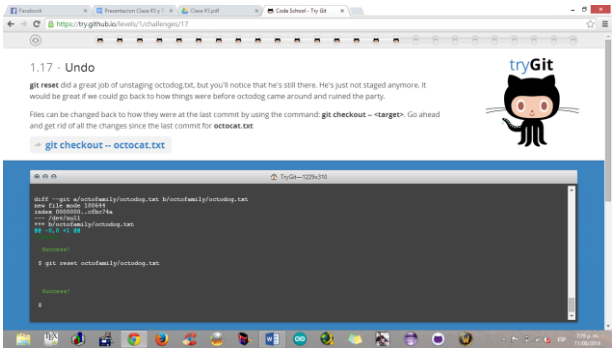

1.12 · Pulling Remotely
Let's pretend some time has passed. We've invited other people to our github project who have pulled your changes, made their own commits, and pushed them.
We can check for changes on our GitHub repository and pull down any new changes by running:
git pull origin master
tryGit
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!

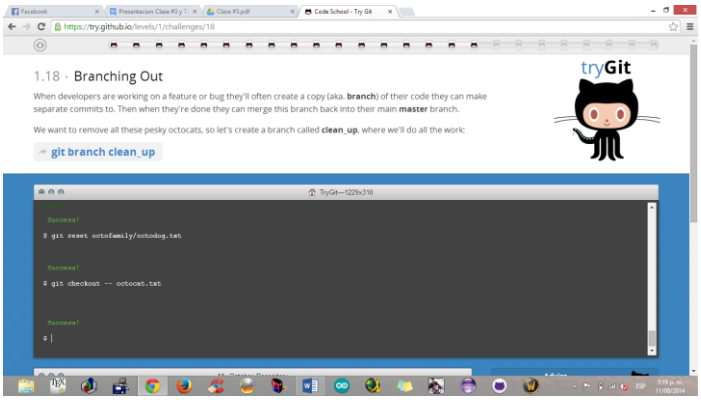

1.13 · Differences
Uh oh, looks like there have been some additions and changes to the octocat family. Let's take a look at what is different from our last commit by using the git diff command.
In this case we want the diff of our most recent commit, which we can refer to using the HEAD pointer.
git diff HEAD
tryGit
Branch master set up to track remote branch master from origin.
Success!
$ git pull origin master
Updating 3052b4d..3e70b0f
Fast-forward
yellow_octocat.txt | 1 +
1 file changed, 1 insertion(+)
create mode 100644 yellow_octocat.txt
Success!
$

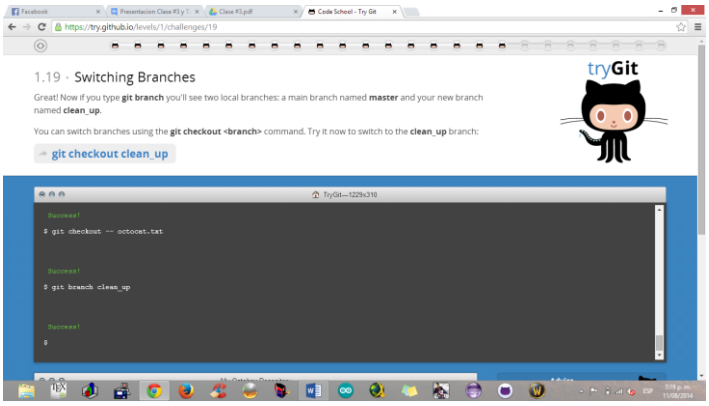

1.14 · Staged Differences
Another great use for diff is looking at changes within files that have already been staged. Remember, staged files are files we have told git that are ready to be committed.
Let's use git add to stage octofamily/octodog.txt, which I just added to the family for you.
git add octofamily/octodog.txt
tryGit
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
Success!
$

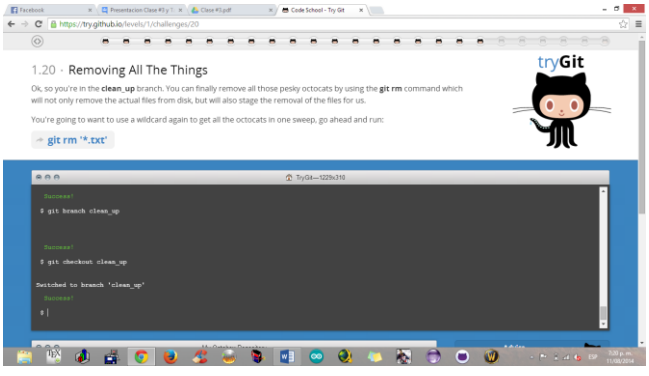

1.15 · Staged Differences (cont'd)
Good, now go ahead and run git diff with the --staged option to see the changes you just staged. You should see that octodog.txt was created.
git diff --staged
tryGit
diff --git a/octocat.txt b/octocat.txt
index 7d8d808..e725ef6 100644
--- a/octocat.txt
+++ b/octocat.txt
Success!
$ git add octofamily/octodog.txt
Success!
$

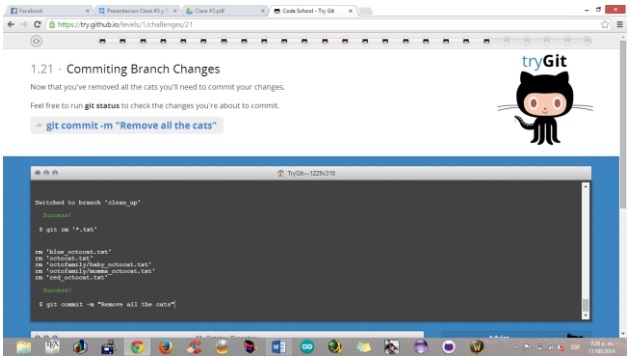

1.16 · Resetting the Stage
So now that octodog is part of the family, octocat is all depressed. Since we love octocat more than octodog, we'll turn his frown around by removing octodog.txt.
You can unstage files by using the git reset command. Go ahead and remove octofamily/octodog.txt.
git reset octofamily/octodog.txt
tryGit
Success!
$ git diff --staged
diff --git a/octofamily/octodog.txt b/octofamily/octodog.txt
new file mode 100644
index 0000000..c8bc76a
--- /dev/null
+++ b/octofamily/octodog.txt
Success!
$

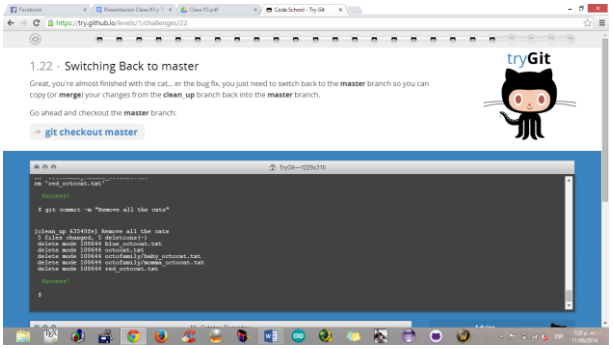

https://try.github.io/levels/1/challenges/17
1.17 · Undo
tryGit
git reset did a great job of unstaging octodog.txt, but you'll notice that he's still there. He's just not staged anymore. It would be great if we could go back to how things were before octodog came around and ruined the party.
Files can 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
$ git reset octofamily/octodog.txt

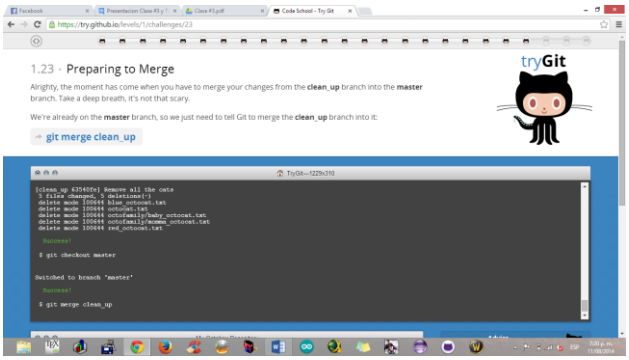

https://try.github.io/levels/1/challenges/18
1.18 · Branching Out
tryGit
When developers are working on a feature or bug they'll often create a copy (aka. branch) of their code they can make separate commits to. Then when they're done they can merge this branch back into their main master branch.
We want to remove all these pesky octocats, so let's create a branch called clean_up, where we'll do all the work:
git branch clean_up
$ git reset octofamily/octodog.txt
$ git checkout -- octocat.txt

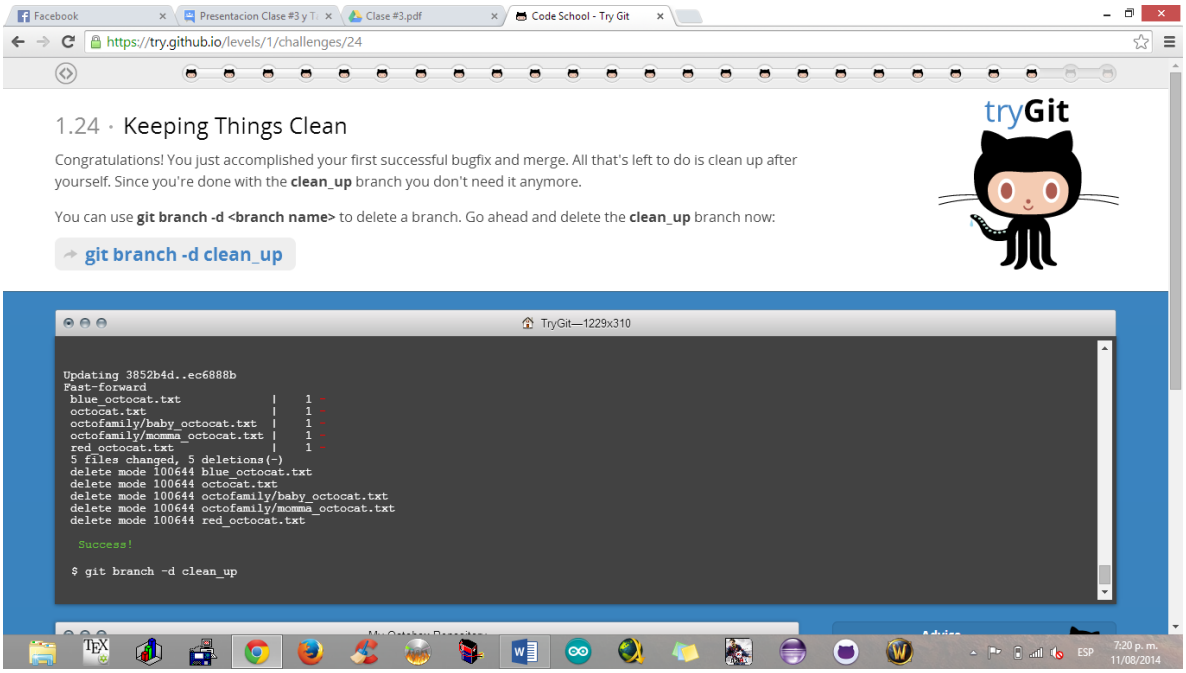

https://try.github.io/levels/1/challenges/19
1.19 · Switching Branches
tryGit
Great! Now if you type git branch you'll see two local branches: a main branch named master and your new branch named clean_up.
You can switch branches using the git checkout <branch> command. Try it now to switch to the clean_up branch:
git checkout clean_up
$ git checkout -- octocat.txt
$ git branch clean_up

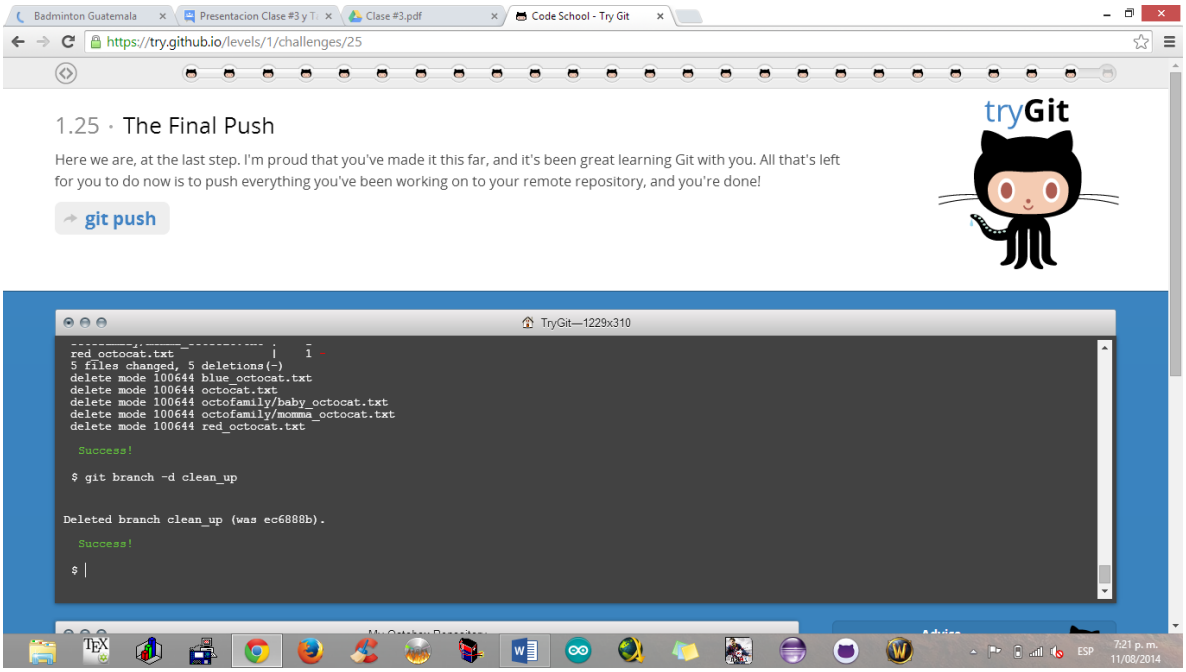

https://try.github.io/levels/1/challenges/20
1.20 · Removing All The Things
tryGit
Ok, so you're in the clean_up branch. You can finally remove all those pesky octocats by using the git rm command which will not only remove the actual files from disk, but will also stage the removal of the files for us.
You're going to want to use a wildcard again to get all the octocats in one sweep, go ahead and run:
git rm '*.txt'
$ git branch clean_up
$ git checkout clean_up
Switched to branch 'clean_up'


1.21 · Commiting Branch Changes
Now that you've removed all the cats you'll need to commit your changes.
Feel free to run git status to check the changes you're about to commit.
git commit -m "Remove all the cats"
tryGit
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 commit -m "Remove all the cats"


1.22 · Switching Back to master
Great, you're almost finished with the cat... er the bug fix, you just need to switch back to the master branch so you can copy (or merge) your changes from the clean_up branch back into the master branch.
Go ahead and checkout the master branch:
git checkout master
tryGit
rm 'red_octocat.txt'
Success!
$ git commit -m "Remove all the cats"
[clean_up 635405e] 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!
$


1.23 · Preparing to Merge
Alrighty, the moment has come when you have to merge your changes from the clean_up branch into the master branch. Take a deep breath, it's not that scary.
We're already on the master branch, so we just need to tell Git to merge the clean_up branch into it:
git merge clean_up
tryGit
[clean_up 635405e] 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


Facebook
Presentacion Clase #3 y T
Clase #3.pdf
Code School - Try Git
https://try.github.io/levels/1/challenges/24
1.24 · Keeping Things Clean
tryGit
Congratulations! You just accomplished your first successful bugfix and merge. All that's left to do is clean up after yourself. Since you're done with the clean_up branch you don't need it anymore.
You can use git branch -d <branch name> to delete a branch. Go ahead and delete the clean_up branch now:
git branch -d clean_up
TryGit—1229x310
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
ESP
7:20 p. m.
11/08/2014


Badminton Guatemala
Presentacion Clase #3 y T
Clase #3.pdf
Code School - Try Git
https://try.github.io/levels/1/challenges/25
1.25 · The Final Push
tryGit
Here we are, at the last step. I'm proud that you've made it this far, and it's been great learning Git with you. All that's left for you to do now is to push everything you've been working on to your remote repository, and you're done!
git push
TryGit—1229x310
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!
$
ESP
7:21 p. m.
11/08/2014