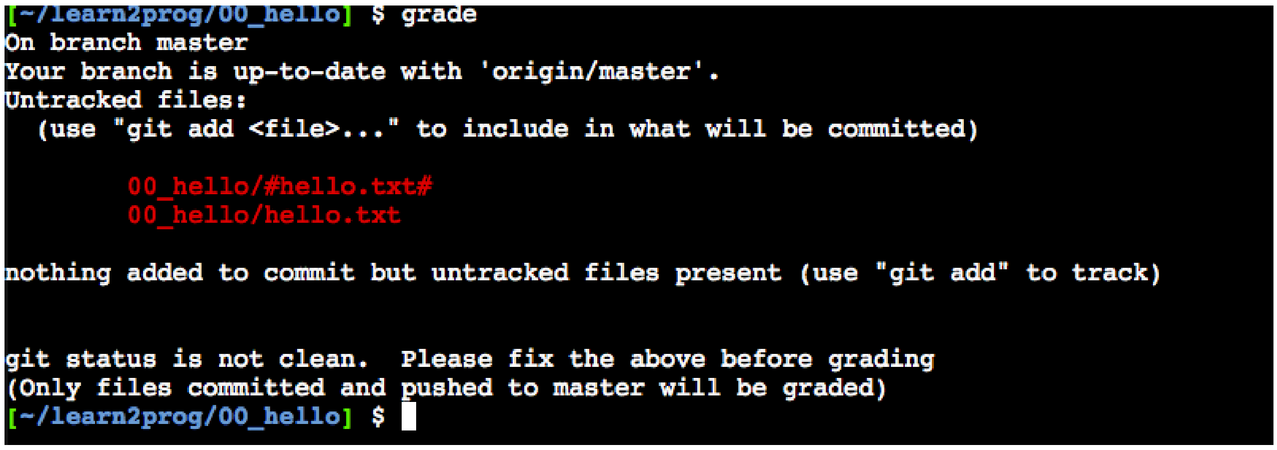
[Optional Reference] Help me fix...

coursera.org/learn/writing-running-fixing-code/supplement/ZwawT/optional-reference-help-me-fix

This reading is a reference for resolving a variety of common problems with **git** and **grade**. While you are certainly welcome to read this now, we recommend it primarily to come back to if you encounter one of these problems. Please do not feel like you need to absorb all the information in this document to continue. Some of these (especially the ones at the end) are somewhat unusual and strange problems that other learners have encountered---hopefully you will never run into most of these problems!

Git status is not clean (untracked files)



```
[~/learn2prog/00_hello] $ grade
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        00_hello/#hello.txt#
        00_hello/hello.txt

nothing added to commit but untracked files present (use "git add" to track)


git status is not clean.  Please fix the above before grading
(Only files committed and pushed to master will be graded)
[~/learn2prog/00_hello] $ 
```

When you get this error message, it means that you have files which are not in git. The grade command does not want to grade your work when this happens because it looks like maybe you have not submitted all of your work (and would be annoyed/confused to get a grade based on only some of your files). How you fix this depends on what you want to do to each file:

> If you want to submit the file, then you need to add it to git. For example, to submit hello.txt you would do

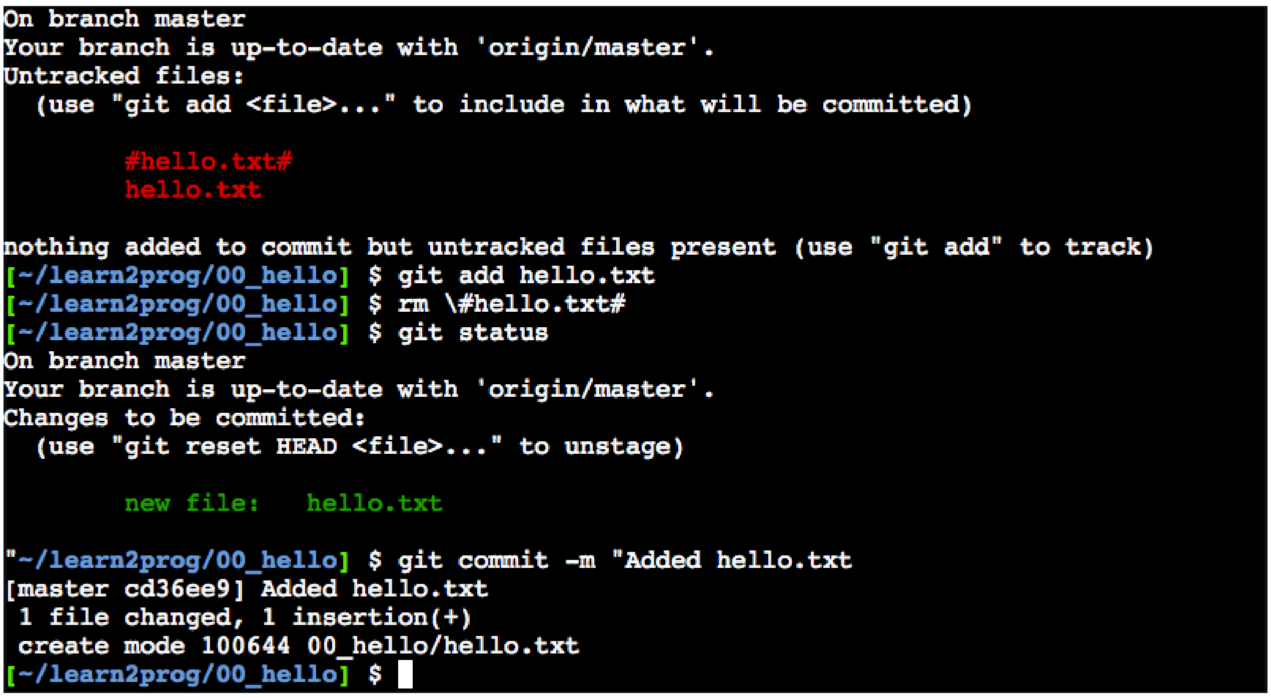
git add hello.txt

> If you want to delete the file, then you would remove it with the rm command. Note that if you remove the file it is gone forever and cannot be recovered, since it is not in git. For example, if you want to remove #hello.txt# you would do

rm \#hello.txt#

Note that because this filename starts with a # sign (which the command shell interprets specially) you need to write \# instead of #. Also note that this command is not “git rm” as the file is not in git. You are just removing the file.

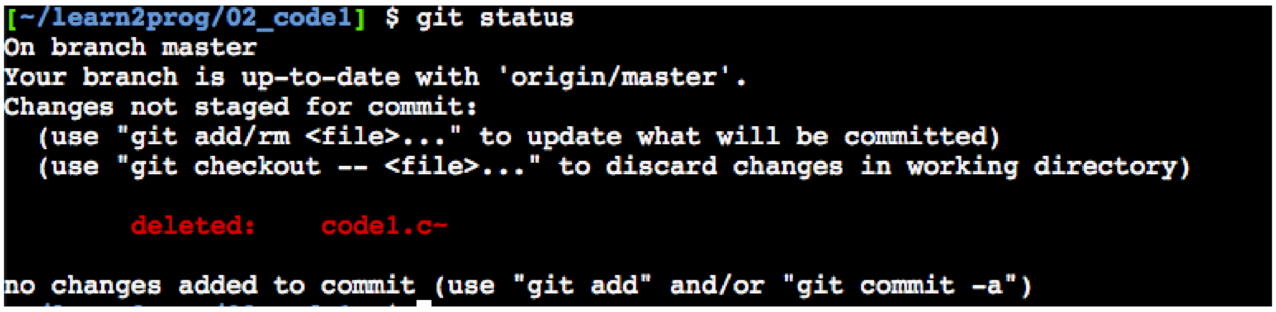
After doing these, you would need to do git commit to commit your changes, then git push to send the commit to where the grader can find it. After that you can do grade.



```
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        #hello.txt#
        hello.txt

nothing added to commit but untracked files present (use "git add" to track)
[~/learn2prog/00_hello] $ git add hello.txt
[~/learn2prog/00_hello] $ rm \#hello.txt#
[~/learn2prog/00_hello] $ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   hello.txt

"~/learn2prog/00_hello] $ git commit -m "Added hello.txt
[master cd36ee9] Added hello.txt
 1 file changed, 1 insertion(+)
 create mode 100644 00_hello/hello.txt
[~/learn2prog/00_hello] $
```

Git status is not clean (deleted files)

If you have deleted a file that git was tracking with “rm” (rather than “git rm”) then git will list the file as deleted but not staged for commit. For example:



```
[~/learn2prog/02_code1] $ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    code1.c~

no changes added to commit (use "git add" and/or "git commit -a")
```

Again, what you do here depends on what outcome you want:

> **If you meant to delete the file and do not want it anymore**: then you just need to tell git that you want to git rid of it with “git rm filename” (and then commit). For example, if the file were code1.c~ you would do

git rm code1.c~

git commit -m “removed backup file”

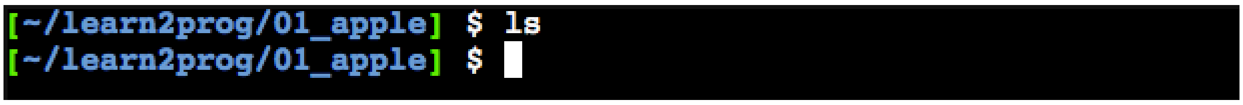
> If you did not meant to delete that file, and want it back: then you can recover the most recently committed version with “git checkout filename”. For example, if the file were code1.c you would do

git checkout code1.c

If you need to restore a version other than the most recent, see the next section.

I deleted or messed up a file I need

Suppose you accidentally delete a README file. For example, here I have accidentally removed the README for 01_apple. When I do "ls" I see no files.



```
[~/learn2prog/01_apple] $ ls
[~/learn2prog/01_apple] $ 
```

If you have just deleted a file that was committed into git, then you can recover it with

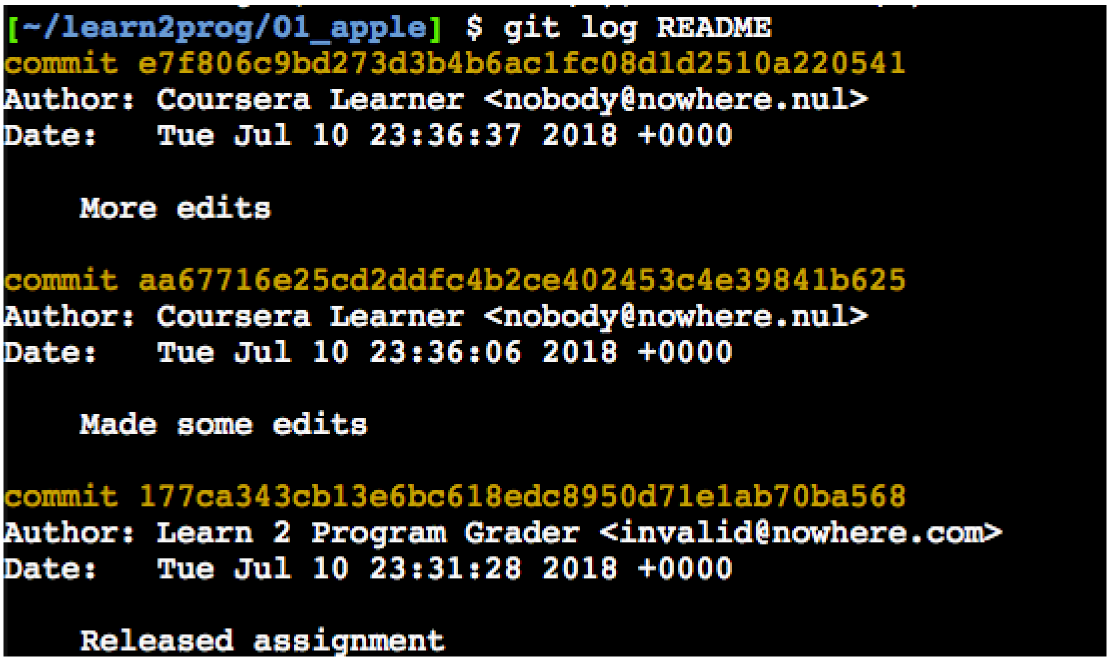
git checkout README

This will restore the most recently committed version from git.

If you have committed a messed up version into git, then you need to do a bit more work to recover a previous commit. The first step is to find the correct past version that you want. You can accomplish this with the git log command. For example, if we are trying to restore our README to a past version,

git log README

will show us the commit history for the README. Note that it will display the log messages that you put when you commit (either with -m "message" or in emacs). Writing good messages can help you find which version to recover!



```
[~/learn2prog/01_apple] $ git log README
commit e7f806c9bd273d3b4b6ac1fc08d1d2510a220541
Author: Coursera Learner <nobody@nowhere.nul>
Date:   Tue Jul 10 23:36:37 2018 +0000

    More edits

commit aa67716e25cd2ddfc4b2ce402453c4e39841b625
Author: Coursera Learner <nobody@nowhere.nul>
Date:   Tue Jul 10 23:36:06 2018 +0000

    Made some edits

commit 177ca343cb13e6bc618edc8950d71e1ab70ba568
Author: Learn 2 Program Grader <invalid@nowhere.com>
Date:   Tue Jul 10 23:31:28 2018 +0000

    Released assignment
```

In this example, we have 3 possible past version to recover. The top is the most recent commit, and is what we would restore with just "git checkout README". However, let us suppose that we want to recover the version released by the grading system. This is the oldest version, which is at the bottom (it also says the Author is Learn 2 Program Grader, meaning it came from the grading system rather than Coursera Learner, meaning it came from you). You need to get the commit id which is the long hex number in yellow, in this case 177ca343cb13e6bc618edc8950d71e1ab70ba568 (just copy and paste it! Do not try to retype it yourself). Then you can do

git checkout 177ca343cb13e6bc618edc8950d71e1ab70ba568 README

to restore that version of the file.

I ran grade, but do not see the next assignment!

If you graded an assignment but do not see the next one, you should do the following:

- **git pull** : make sure everything is up to date. If you run into problems with this, then fix them before proceeding (see other entries in this help document for advice, or post on the forums if it is not covered and you cannot figure it out).
- Read your **grade**.txt for the current assignment: Run 'cat grade.txt' in the directory of the assignment that you just did. What grade does it say you got? If you did not pass, then the system will not release the next assignment yet. Try to understand and fix your mistakes on this assignment.
- **cd ~/learn2prog** then **ls** : do you see the next assignment now? If so, great

If not, then you have an unusual problem, and need to post on the forums. Drew will try to sort it out as soon as he can.

Nothing to commit, working directory is clean

This error happens when you try to do git commit, but everything is the same as your previous commit. That is, you haven't changed anything since you last committed, so there is no need to commit again. You should make sure you have saved everything. Otherwise, you can just git push (if you have not already) and grade.

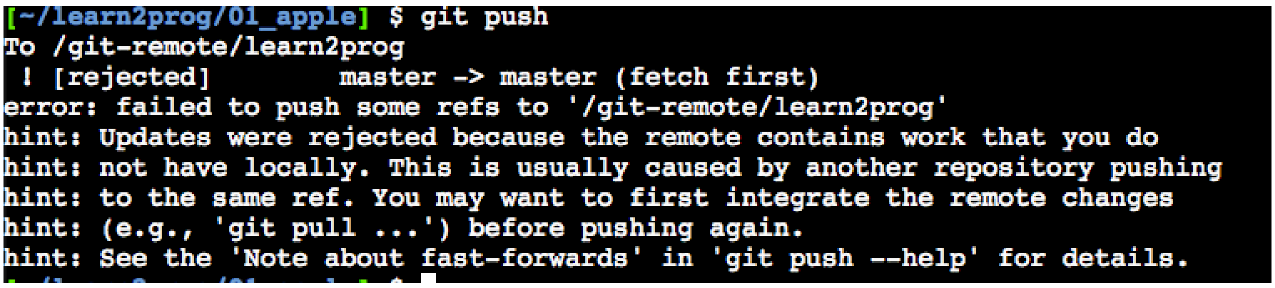
No changes add to to commit

This error happens when there are changes, but you have not staged them for commit. That is, you have changed files, but not done "git add" to tell git that you want it to put those changes into the commit. You can either use "git add" to stage any files you want (tell git to put them in the next commit), or give git commit the "-a" option to commit all changed files:

*git commit -**a**m "Did assignment 46"*

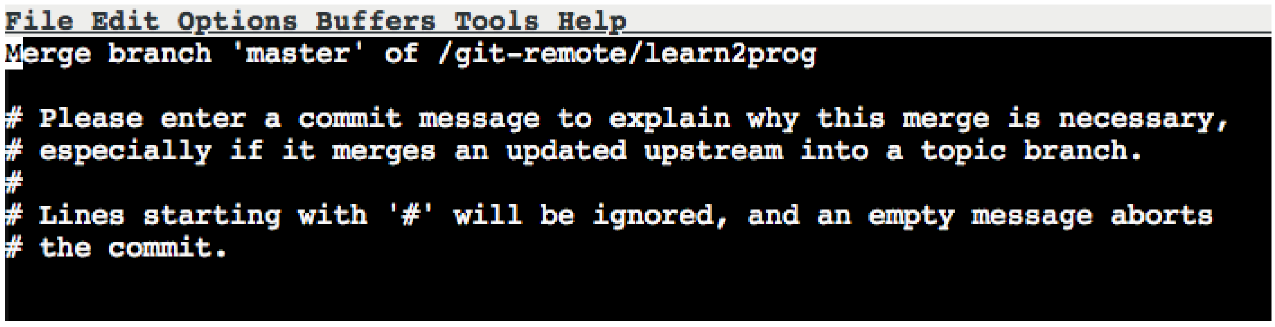
Cannot git push: need to git pull first

If the grading system does something for you (produces a grade file, or releases an assignment), and you do not git pull to receive it before you try to git push more changes, you will receive an error like this:



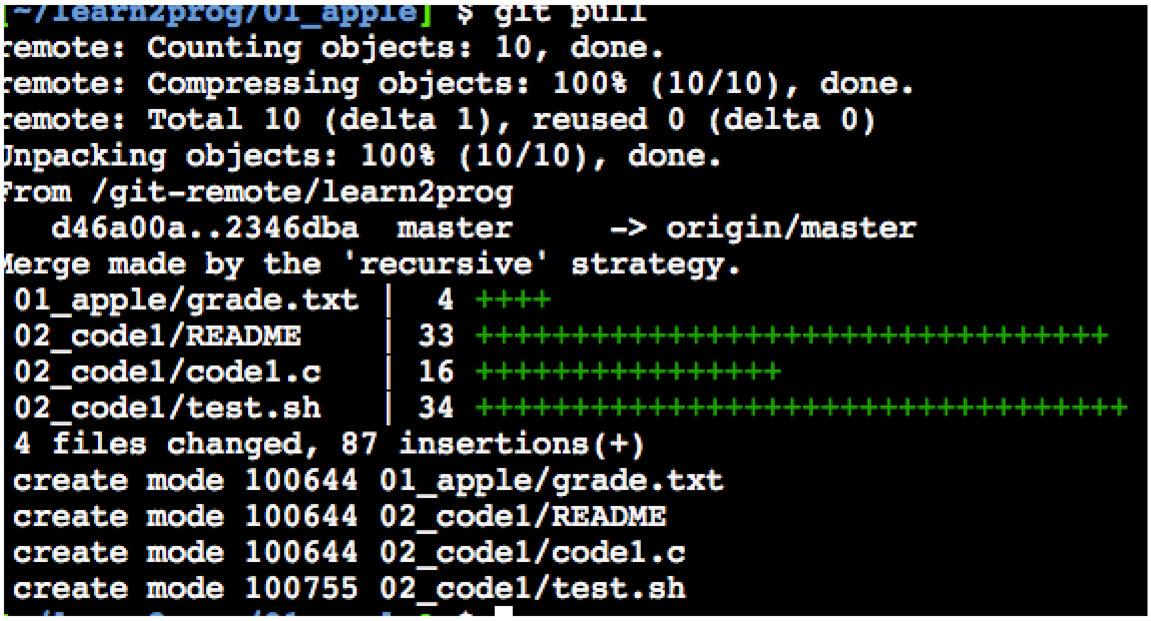
```
[~/learn2prog/01_apple] $ git push
To /git-remote/learn2prog
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to '/git-remote/learn2prog'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

This just means that git cannot push changes until it has an up to date copy of all the things that the grader has done. If you run git pull, then git will open emacs for you to edit the commit message (it makes a commit when it merges the changes together) that looks like this



```
Merge branch 'master' of /git-remote/learn2prog

# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
```

You can just save (control-x control-s) and quit (control-x control-c) to accept the default message. In almost all cases, you will see output that looks something like this:



```
[~/learn2prog/01_apple] $ git pull
remote: Counting objects: 10, done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 10 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (10/10), done.
From /git-remote/learn2prog
   d46a00a..2346dba  master     -> origin/master
Merge made by the 'recursive' strategy.
 01_apple/grade.txt |  4 ++++
 02_code1/README    | 33 +++++++++++++++++++++++++++++++++
 02_code1/code1.c   | 16 ++++++++++++++++
 02_code1/test.sh   | 34 ++++++++++++++++++++++++++++++++++
 4 files changed, 87 insertions(+)
 create mode 100644 01_apple/grade.txt
 create mode 100644 02_code1/README
 create mode 100644 02_code1/code1.c
 create mode 100755 02_code1/test.sh
```

If the above happened, git has merged the two versions together by itself. If this is what happened, you can git push and grade.

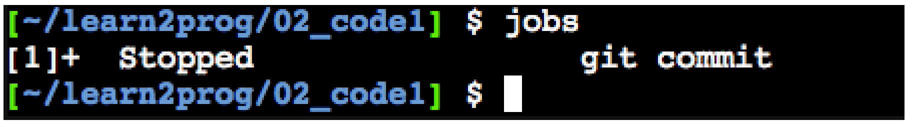
The other outcome, which should only happen if you and the grader both modified the same files (which should NOT happen---do not edit your grade.txt files!) is that git could not merge things and wants you to merge them yourself. Since this should be incredibly rare, we address it at the end in the section “Merge Conflicts”

<u>The below problems should be much rarer</u> (but are listed, as learners have encountered them)

Unable to create '/home/student/learn2prog/.git/index.lock': File exists.

This error happens when you either already have "git" running (possibly suspended) or git crashed in some way. This error really should not happen, but most likely arises if git opened emacs for a commit message, and you suspended emacs (leaving git running, waiting for a commit message).

> First run "jobs" to see what programs you have suspended. If you have suspended git, then us "fg" to bring it back to the foreground and finish it



```
[~/learn2prog/02_code1] $ jobs
[1]+  Stopped                 git commit
[~/learn2prog/02_code1] $ 
```

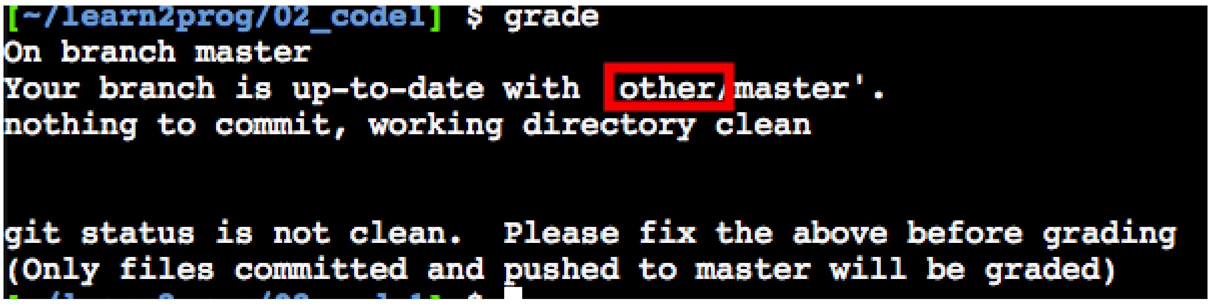
> Otherwise, just do

rm /home/student/learn2prog/.git/index.lock

and then try again on whatever git command you were doing.

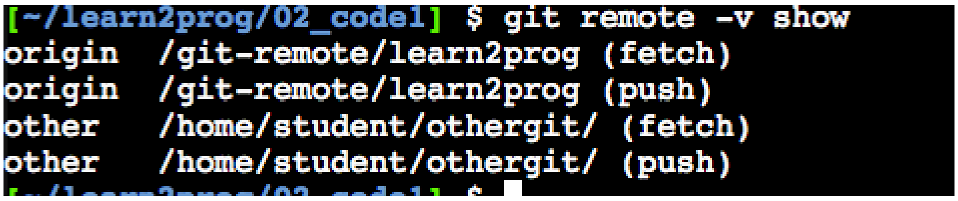
Git status is not clean, with seemingly no information

If you run grade and get an error message about git status not being clean, but there seems to be no other information, look carefully at the second line, which should say "Your branch is up-to-date with 'origin/master'. If it says something else, like this:



```
[~/learn2prog/02_code1] $ grade
On branch master
Your branch is up-to-date with 'other/master'.
nothing to commit, working directory clean


git status is not clean.  Please fix the above before grading
(Only files committed and pushed to master will be graded)
```

then you have changed your upstream remote (what git considers to be the default place to push to/pull from). The grader is unhappy here, since you are not longer pushing to/pulling from it by default. We realize this error message could be better, but never even thought this problem would come up. The first thing you should do is check that you have the correct setup for the remote called "origin" with the command "git remote show -v" :

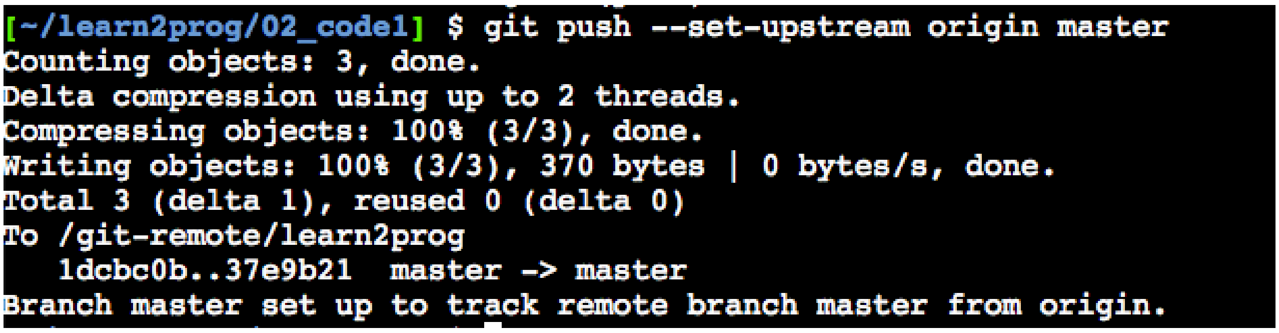


```
[~/learn2prog/02_code1] $ git remote -v show
origin  /git-remote/learn2prog (fetch)
origin  /git-remote/learn2prog (push)
other   /home/student/othergit/ (fetch)
other   /home/student/othergit/ (push)
```

Here, you are looking at the two lines that say “origin” and want to make sure that they say /git-remote/learn2prog (and one says fetch and the other says push). It is fine to have any other remotes you want (such as your github account, etc), just make sure that origin is setup as above.

Now run

git push --set-upstream origin master

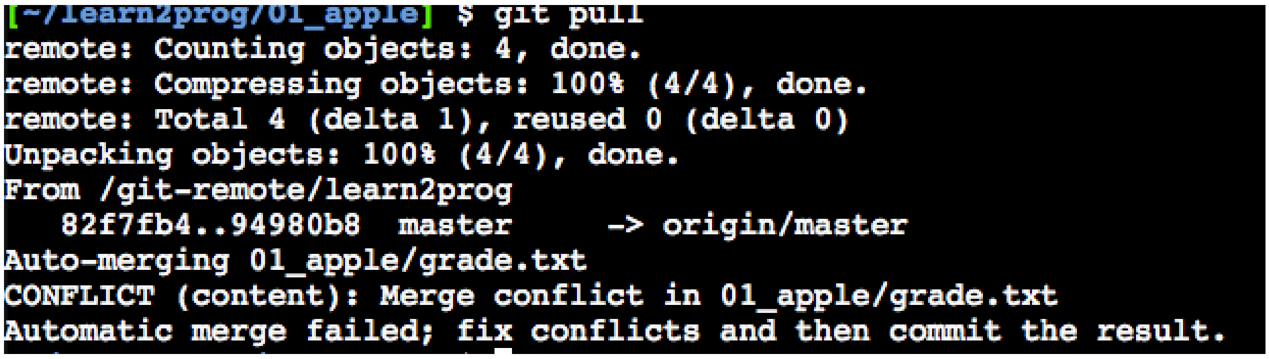


```
[~/learn2prog/02_code1] $ git push --set-upstream origin master
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 370 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To /git-remote/learn2prog
   1dcbc0b..37e9b21  master -> master
Branch master set up to track remote branch master from origin.
```

This command set your “upstream” (default push/pull remote) back to the grader.

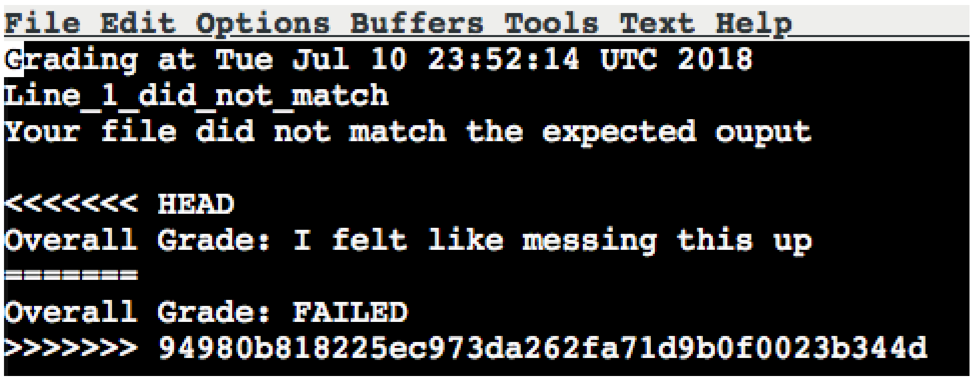
Merge Conflicts

This problem really should not happen: it requires you and the grader to modify the same file. If it does, this is what will happen:



```
[~/learn2prog/01_apple] $ git pull
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
From /git-remote/learn2prog
   82f7fb4..94980b8  master     -> origin/master
Auto-merging 01_apple/grade.txt
CONFLICT (content): Merge conflict in 01_apple/grade.txt
Automatic merge failed; fix conflicts and then commit the result.
```

If you do this, then to fix it you have to open the conflicting file in emacs, and you will see something like this:



```
Grading at Tue Jul 10 23:52:14 UTC 2018
Line_1_did_not_match
Your file did not match the expected ouput

<<<<<<< HEAD
Overall Grade: I felt like messing this up
=======
Overall Grade: FAILED
>>>>>>> 94980b818225ec973da262fa71d9b0f0023b344d
```

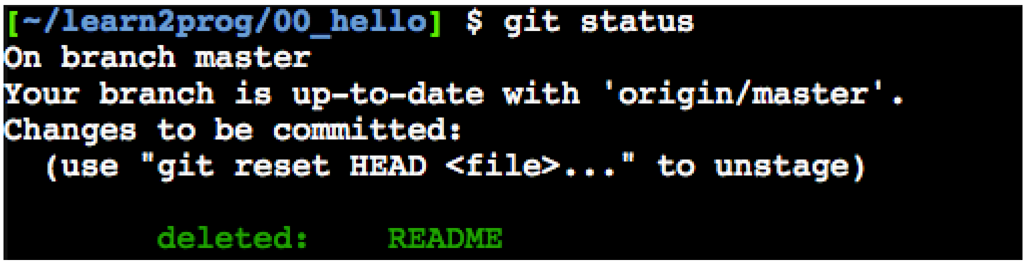
Here, git shows what was in the two different versions of the file (the graders and yours), with <<<< ==== and >>>>. The text between the <<<< and the ==== is what was in your version, and the txt between the ==== and the >>>> is what the grader had. Edit the file to be whatever it is supposed to be, save, and quit. Then do *git add grade.txt* (or whatever file the problem was with), and *git commit -m "Fixed merge conflict"*.

Question: I 'git rm'ed a file, how can I recover?

If you "git rm"ed a file (in this example, README), you can't just do "git checkout". Instead, you should first do

git status

to see if you committed that removal or not



```
[~/learn2prog/00_hello] $ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    README
```

If the output looks like the above image, you have staged, but not committed the removal. As the message there says you can undo this with

git reset HEAD README

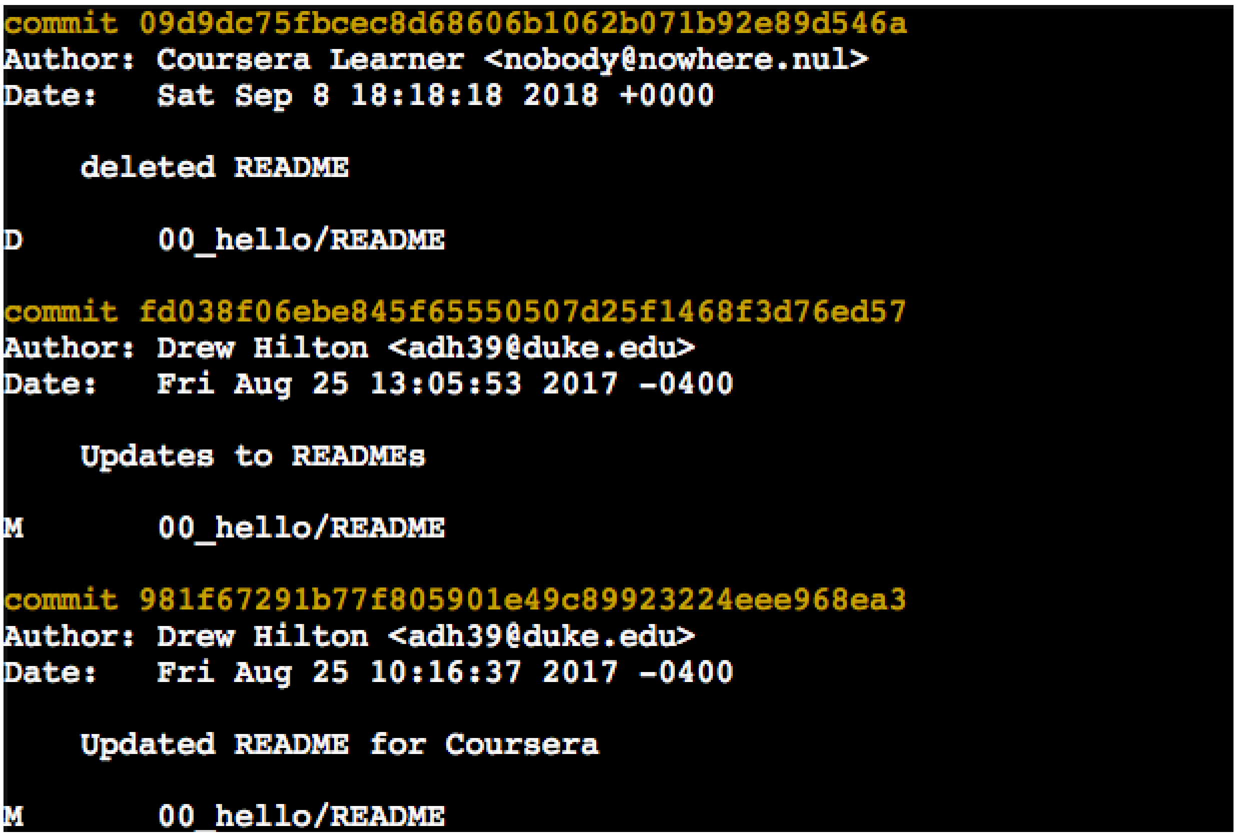
git checkout -- README

If you already committed the deletion (does not show up as above in git status) then do

git log -- README

git log --name-status -- README

You should see something like this:



```
commit 09d9dc75fbcec8d68606b1062b071b92e89d546a
Author: Coursera Learner <nobody@nowhere.nul>
Date:   Sat Sep 8 18:18:18 2018 +0000

    deleted README

D       00_hello/README

commit fd038f06ebe845f65550507d25f1468f3d76ed57
Author: Drew Hilton <adh39@duke.edu>
Date:   Fri Aug 25 13:05:53 2017 -0400

    Updates to READMEs

M       00_hello/README

commit 981f67291b77f805901e49c89923224eee968ea3
Author: Drew Hilton <adh39@duke.edu>
Date:   Fri Aug 25 10:16:37 2017 -0400

    Updated README for Coursera

M       00_hello/README
```

At the end of each commit, it lists what was changed (thats what --name-status does) and D means deleted. Find the one with the D, and then look at the one BEFORE (underneath it) in this case, it is fd038f06ebe845f65550507d25f1468f3d76ed57

so you need to do

git checkout fd038f06ebe845f65550507d25f1468f3d76ed57 README

You will then want to "git add README" to put it back in git.