**SIMPLE TUT**

**right click the folder > git bash here** but it’s easier to work in CMD than BASH for commits!

git init (for first time)

git add . (never do “git add .” TWICE ON ACCIDENT it sucks!)

(Also type git reset to unstage any files after git add .)

git status

git commit -m 'Project' Copy this example. 50 character message max

**git remote add origin your\_url\_name** (for first time) NO QUOTES (use insert not paste!)

IF PASTING FROM YOUR VIRTUAL MACHINE YOU MUST USE SSH:  
**git remote add origin git@github.com:Elegant-Geek/odin-project-notes.git**

git remote add origin https://github.com/Elegant-Geek/Ruby\_Revisited\_Crowdfund.git

troubleshooting this ^^^^

git remote -v to reveal any pasting errors (“fatal: protocol **''https'** is not supported”)

**git remote remove origin** (removes the erroneous git remote add origin line)

git remote -v reveals no crap plugging up your remote link thing (good)

or just set the correct url like this: **git remote set-url origin git://new.url.here** <https://stackoverflow.com/questions/16330404/how-to-remove-remote-origin-from-a-git-repository>

**TIP: try not to have 2 git repositories that share the same main local folder u are in!**

git push -u origin ~~master~~ main (you only need “git push” after first push -u origin master)

TLDR: always use soft not hard

Don’t forget the GIT INIT initial step or you’ll get **“Error: remote origin already exists”**

**Diagram

Description automatically generated**

* Graphical user interface, text, application

  Description automatically generated

**HOW TO CLONE A GITHUB REPOSITORY**

**2) select the DOWNLOAD ZIP option. Done.**

**TROUBLESHOOTING:**

**CHANGE MESSAGE**

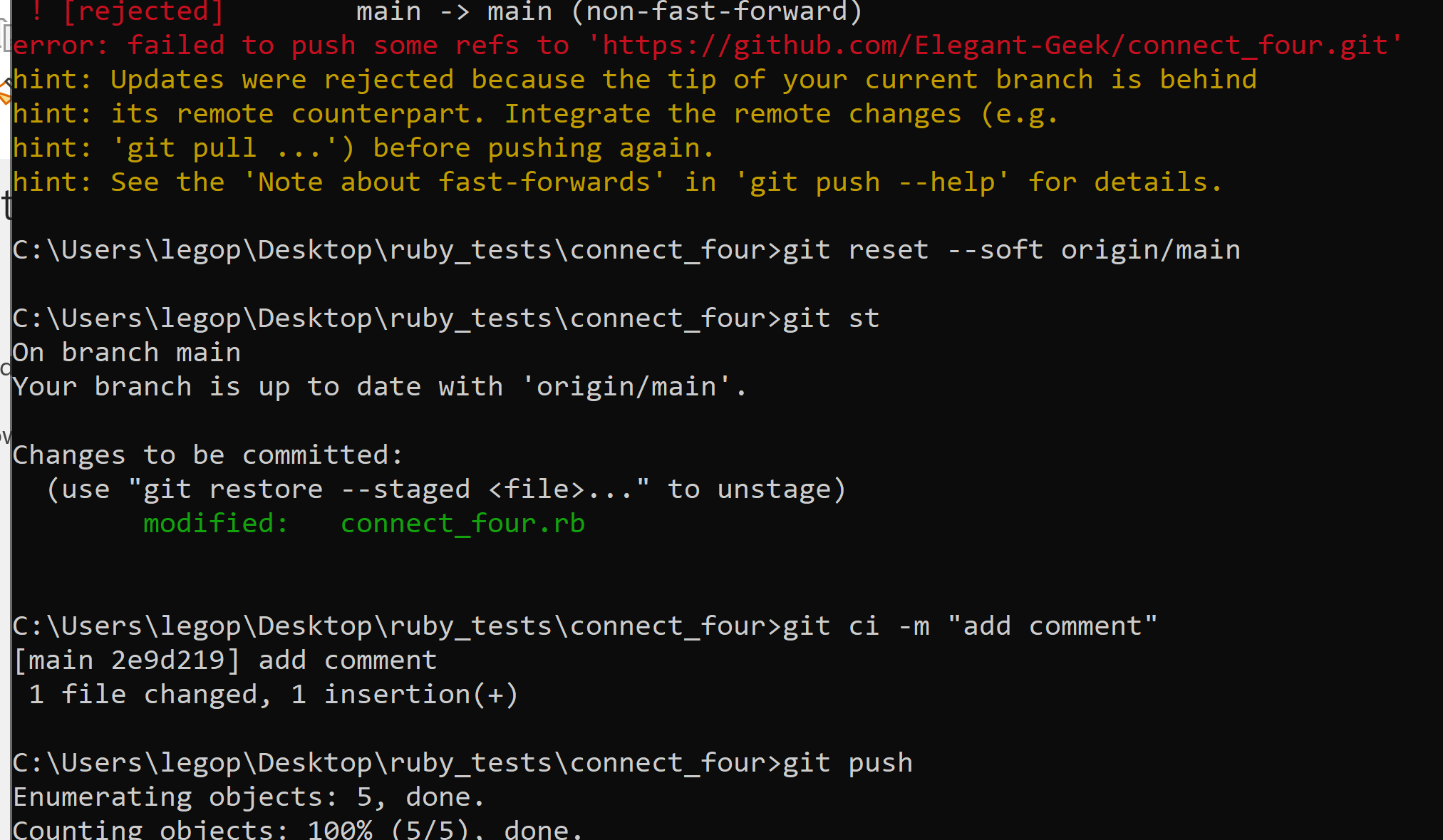
* **git commit --amend -m "New message"** For new message for your most recent commit. (only works if SUBLIME is installed, but it ROCKS!
* THEN IMMEDIATELY AFTER, use the **git push --force** command to force push over the old commit (updates the message)

**UNDO TANGLE**

Text

Description automatically generated

Maybe try **git reset --soft origin/main** first before the red text. Then all you gotta do is git add. Git ci -m “insert message about whatever your most recent update was about” then git push with no issues! (6.14.23)



^ successful untangling using a oneliner: **git reset --soft origin/main**

**git reset --hard origin/master** << this will revert back to last thing you uploaded to github and resolves any other issues. “**throw away all my staged and unstaged changes,** forget everything on my current local branch and make it exactly the same as origin/master.” (reverts back to last local commit in the git log I think? NO. It will be up to date with what you push to REMOTE aka online. It will clear all local changes that have not been pushed to remote yet. Only good for when you are undoing a tangle! 12.8.22)

(After running this type “git st” to see “Your branch is up to date with 'origin/master'.” GREAT!)

SOLUTION: <https://stackoverflow.com/questions/2452226/master-branch-and-origin-master-have-diverged-how-to-undiverge-branches>

DESRIPTION: <https://stackoverflow.com/questions/15432052/what-is-the-meaning-of-git-reset-hard-origin-master>

**“Error: remote origin already exists”** [**FIX IS HERE**](https://www.cloudbees.com/blog/remote-origin-already-exists-error)

**NOTE: ^^^^^^ this fix (link) is only if you want to commit future changes to a new place on github (aka detach old remote and attach a new remote to a previous local git log!)**

**HOW TO CLONE A GITHUB REPOSITORY**

1. Create new repository in github ready for launch (it should be on the standby page)
2. Graphical user interface, text, application

   Description automatically generated **get to this screen in original (not new) repository.**
3. **select the DOWNLOAD ZIP option. Done.**

**NOTE: YOU MUST copy from github as shown above. DO NOT just copying “ctrl c ctrl v-ing” your literal local files into a new directory otherwise when you try to push to new repository you get the “remote already exists” error shit.**

1. **^^ just download the zip and extract all within the downloads folder.**
2. **then cut and paste those extracted project files into a new folder u created locally (ex. “inverted-calculator” folder)**
3. **then cd into that newly created local folder on cmd line, type git add . to add the copied files then do:**

**git st**

**git ci -m “inverted calc keyboard”**

**git remote add origin** [**https://github.com/Elegant-Geek/inverted-keyboard-calculator.git**](https://github.com/Elegant-Geek/inverted-keyboard-calculator.git)

**git push -u origin main**

**UNDO LAST COMMIT: SAFELY**

**git reset --soft HEAD~1 << this is safe soft undo commit that still preserves work**

**^^ then immediately git pull before making modifications to a file. Otherwise you have merge conflicts if you have already pushed the commit to github I think**

ALWAYS PULL FROM GITHUB BEFORE YOU MODIFY STUFF AND PUSH BACK UP TO GIT! IF YOU NEED TO EDIT A FILE, YOU HAVE TO PULL CURRENT (LAST COMMIT) FROM GITHUB MODIFY IT, THEN PUSH / COMMIT.

^^ If you do **git reset --soft HEAD~1** without pulling the most current copy from github afterwards, you get a tangled mess of unable to merge files! Always pull current github thing before (making modifications) and then trying to push!

[**https://devconnected.com/how-to-undo-last-git-commit/#:~:text=The%20easiest%20way%20to%20undo,removed%20from%20your%20Git%20history**](https://devconnected.com/how-to-undo-last-git-commit/#:~:text=The%20easiest%20way%20to%20undo,removed%20from%20your%20Git%20history)**.**

**UNDO LAST COMMIT: NOT SAFE (deletes work!)**

**removing previous commits from git first then github (order matters)**

STEP 1: **git stash --include-untracked** STASH ALL LOCAL SHIT OR MAKE A BACKUP OF UNTRACKED (never added/staged/committed) FILES OR ELSE THEY GO BYE BYE IN STEP 2

STEP 2: **git reset --hard HEAD~1** << to undo last commit in GIT REPOSITORY (again this **clears all your work** in the working directory as well) use **HEAD~1** so you safely delete one at a time. Use “git log” in cmd to view and confirm what got deleted before moving on to step 2. **THIS CLEARS ALL UNTRACKED FILES IN THE DIRECTORY YOU ARE IN. BAD.** For future reference, prefer git stash to git reset --hard. git stash creates a temporary "stash" commit with your changes. These can be easily retrieved with git stash apply or git stash pop. When you nuke your working directory with git reset --hard, it is much more difficult to undo, sometimes even impossible.

STEP 2: **git push -f origin HEAD^:master** << to update (rollback) in github repository so that your github commits are now freshly synced up with your LOCAL repository (git)!

^^ NOTE: this STEP 2 bit should never be used in a repository where multiple people are collaborating with you! This is strictly for individual / independent projects.

There's one very important caveat with this.. and thats the "--hard" part. --**hard blows away your local uncommitted changes (aka non committed files).** And you cant get them back like this (as they've not been committed anywhere). I believe there is nothing you can do about that

**AND NEVER EVER DO THIS COMMAND BELOW:**

Context of why this is in here: I only did it to wipe a github repo that only had one commit to it (aka I luckily never had a long running git log), but this will wipe all GIT LOG files too! BAD.

**rm -rf .git deletes the git repository completely as if the repository (AND local git log) was never initialized**. NEVER EVER USE THIS! It won’t delete the files in your working directory (like whatever current version you have) but it wipes the git repository you created for it on your computer so you would lose all previous versions of the project and **only have your current working version.**

NO UNDO’s. >>> <https://stackoverflow.com/questions/70343660/how-to-undo-rm-rf-git>

<https://stackoverflow.com/questions/5834014/lf-will-be-replaced-by-crlf-in-git-what-is-that-and-is-it-important> (NOTES about LF CRLF stuff in git pushes ^^^^^^^)

[**UPLOAD/PUSH TO GIT**](https://www.datacamp.com/community/tutorials/git-push-pull)

1. Find the folder directory that hosts all your files > right click the folder > git bash here (this takes u right to it in git bash)

A picture containing diagram

Description automatically generated

1. Initialize the git repository. Use **git init** to initialize the repository.

Text

Description automatically generated

NOTE how I am in the studio\_game bit, NOT the /c/ bit.

1. Add the file to the new local repository.

* Use **git add .** in your bash to add all the files to the given folder.
* Use **git status** in your bash to view all the files which are going to be staged to the first commit.

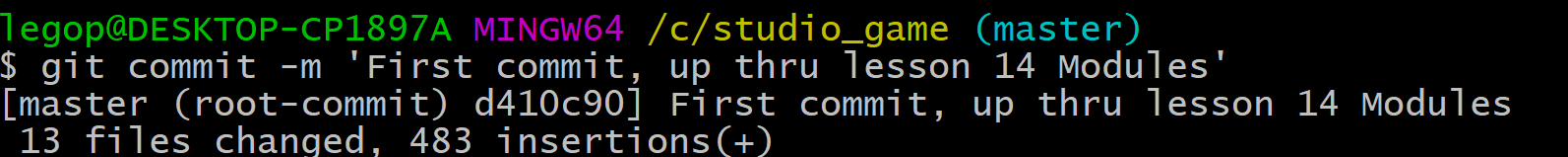
Graphical user interface, text, application

Description automatically generated

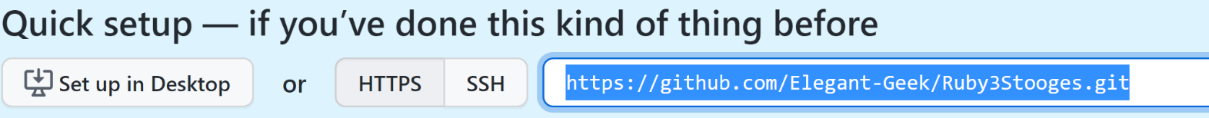
^^^^^^^^ Git **add .** and then **git status** commands. ^^^^^^^

1. Commit the files staged in your local repository by writing a commit message.

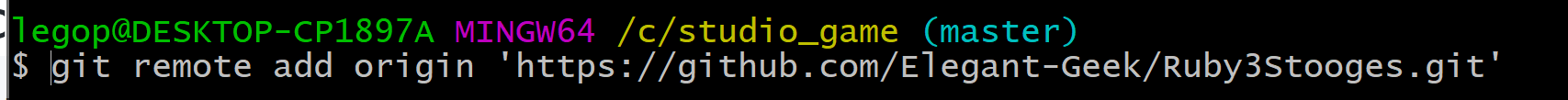
* You can create a commit message by **git commit -m 'your message',** which adds the change to the local repository. **git commit -m 'Final Project'**
* git commit uses '-m' as a flag for a message to set the commits with the content where the full description is included, and a message defining "what was changed"// "why was the change made". git commit -m 'your message'



1. Copy your remote repository's URL from GitHub.



1. Add project to remote repository with **git remote add origin 'your\_url\_name'**

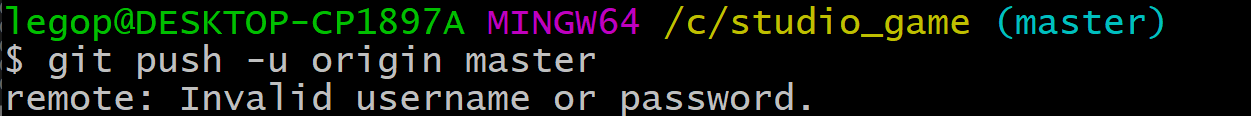


* TIP is to **PRESS INSERT** not ctrl + v to past URL. Use INSERT

1. Push the code in your local repository to GitHub

**git push -u origin master** is used for pushing local content to GitHub.

In the code, the origin is your default remote repository name and '-u' flag is upstream, which is equivalent to '-set-upstream.' and the master is the branch, name.upstream is the repository that we have cloned the project.



1. Login with GITHUB password etc
   * Username: Elegant-Geek
   * Password: pSS9j~(8d4qz
2. View your files in your repository hosted on GitHub. You can finally see the file hosted on GitHub. (AKA just by refreshing your quick setup page!)
3. **rm -rf .git** to retrack back to step 2 if the **git push -u origin master** step failed (video files too large etc)

NOTE: After initial push to git, all u need to do is repeat a few steps:get into the correct directory, then do **git add .** and then **git status** and then finally **git commit -m 'your message'**

Right click main folder holding your documents, select Git Bash here

Type **git add .** then type **git status**

**git commit -m “First commit, up thru lesson 14 Modules”**

**git push -u origin master DONE!**

then you can view your commits to the project!A picture containing logo

Description automatically generated

[**TO INSTANTLY MODIFY MESSAGE IF YOU MESSED UP**](https://www.educative.io/edpresso/how-to-change-a-git-commit-message-after-a-push)

1. **git commit --amend -m "New message"** < that’s when u type in the new message
2. **git push --force https://github.com/Elegant-Geek/Ruby3Stooges master**

* the format for the above line is……. git push --force repository-name branch-name (master)

Text

Description automatically generated

Tip from last session: use **git remote -v** to see where newly moved directories are getting their files pushed. Idk how to proceed from here now that I know that I clearly don’t want my exoticar stuff getting pushed to the wrong repo…

**git remote -v** is great for confirming that your stuff is getting pushed to the right place

How to add the remote repo back to your cmd line/git project so that new changes push to the right repo <https://articles.assembla.com/en/articles/1136998-how-to-add-a-new-remote-to-your-git-repo>

**NOTE: It is not possible to rename a file in Git and remember the history. And it is a pain.**

[**https://stackoverflow.com/questions/2314652/is-it-possible-to-move-rename-files-in-git-and-maintain-their-history**](https://stackoverflow.com/questions/2314652/is-it-possible-to-move-rename-files-in-git-and-maintain-their-history)

You lose all the LOCAL history and should proceed either from pulling from github before making any new changes anywhere then work from there. If you were to continue on without pulling the project from github first, then you would have to push all new local changes to a new or different repository. Github only recognizes when you push from the familiar local that is linked to the remote. If you remove/lose your local git repository info, then you can no longer push new changes directly to github from that directory via cmd, bash, etc. IT SUCKS. You’ll know this has happened when either

1. git remote -v isn’t showing that changes will push to the correct repo
2. the .git hidden folder (should be faded to indicate that it is hidden) is no longer in the project folder!

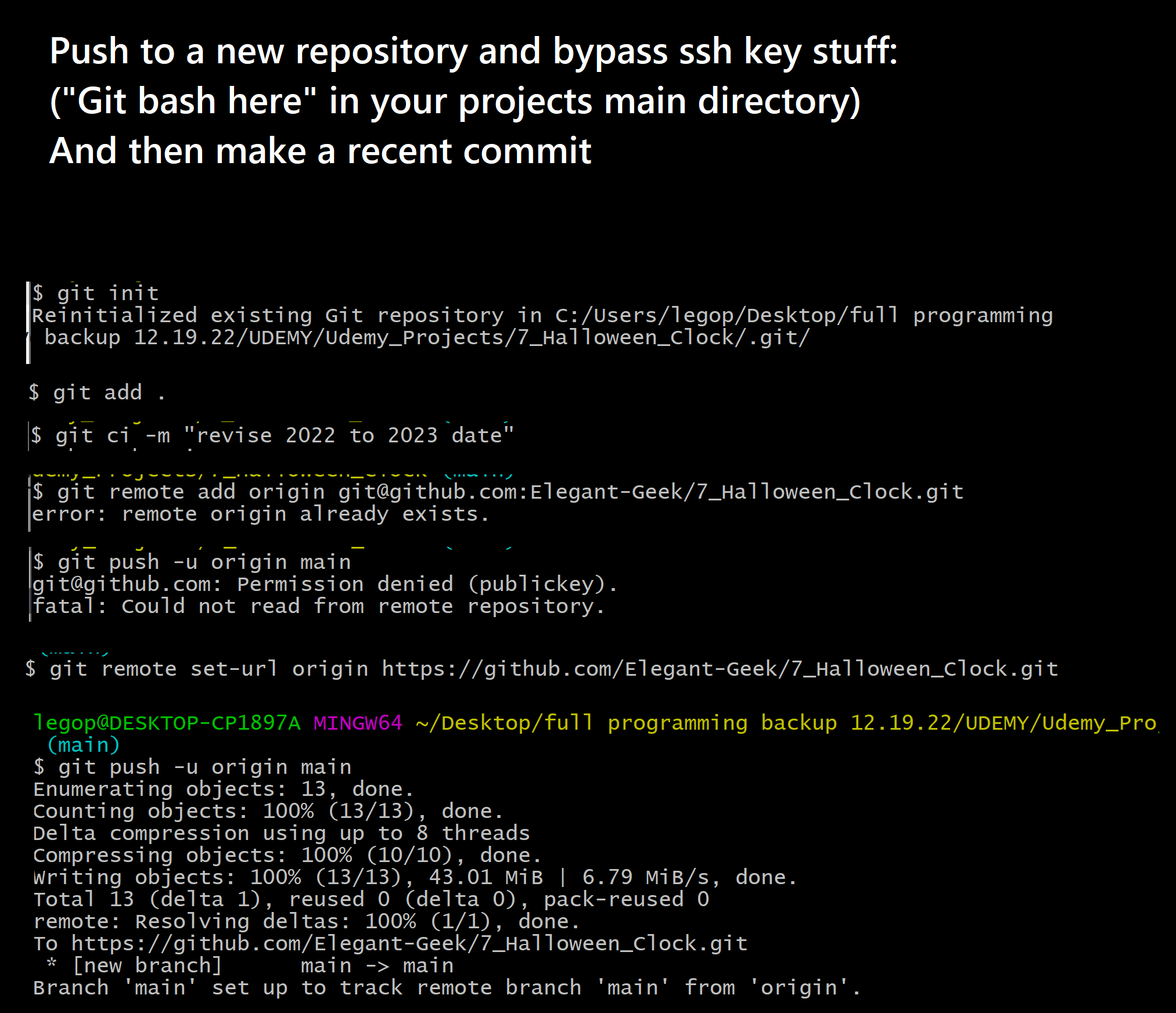
* **YOU HAVE TO EITHER**

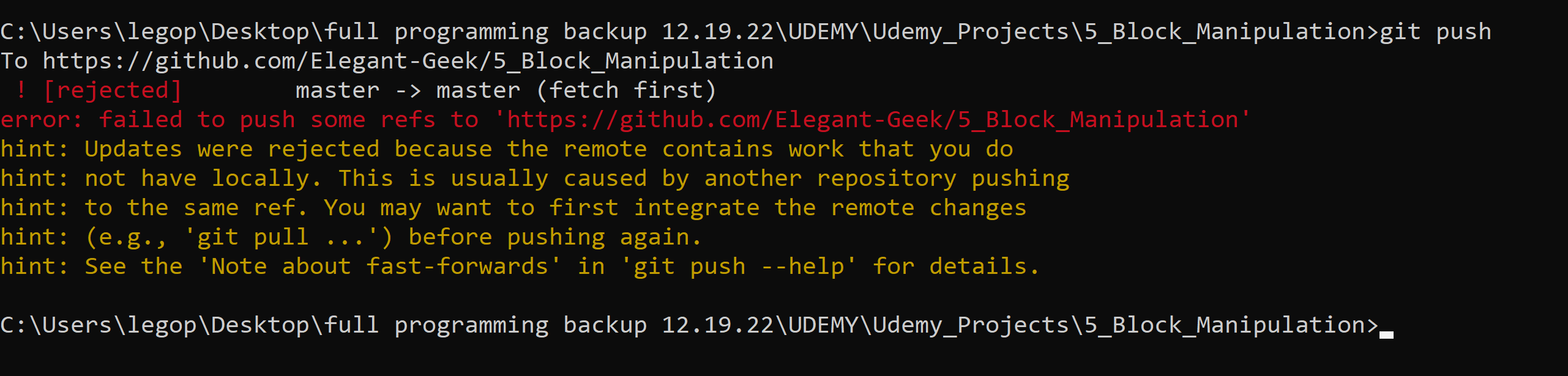
1. GIT PUSH TO NEW REPOSITORY LONG AFTER YOU’VE MOVED FILES/REORGANIZED OR
2. PULL PROJECT FROM GITHUB AND THEN GO FORWARD FROM THERE….

* **Biggest takeaway is that you must make a new repo and push changes in there. Don’t delete old one, just start new one I guess**
* <https://stackoverflow.com/questions/30068298/git-fatal-could-not-read-from-remote-repository-please-make-sure-you-have-th>

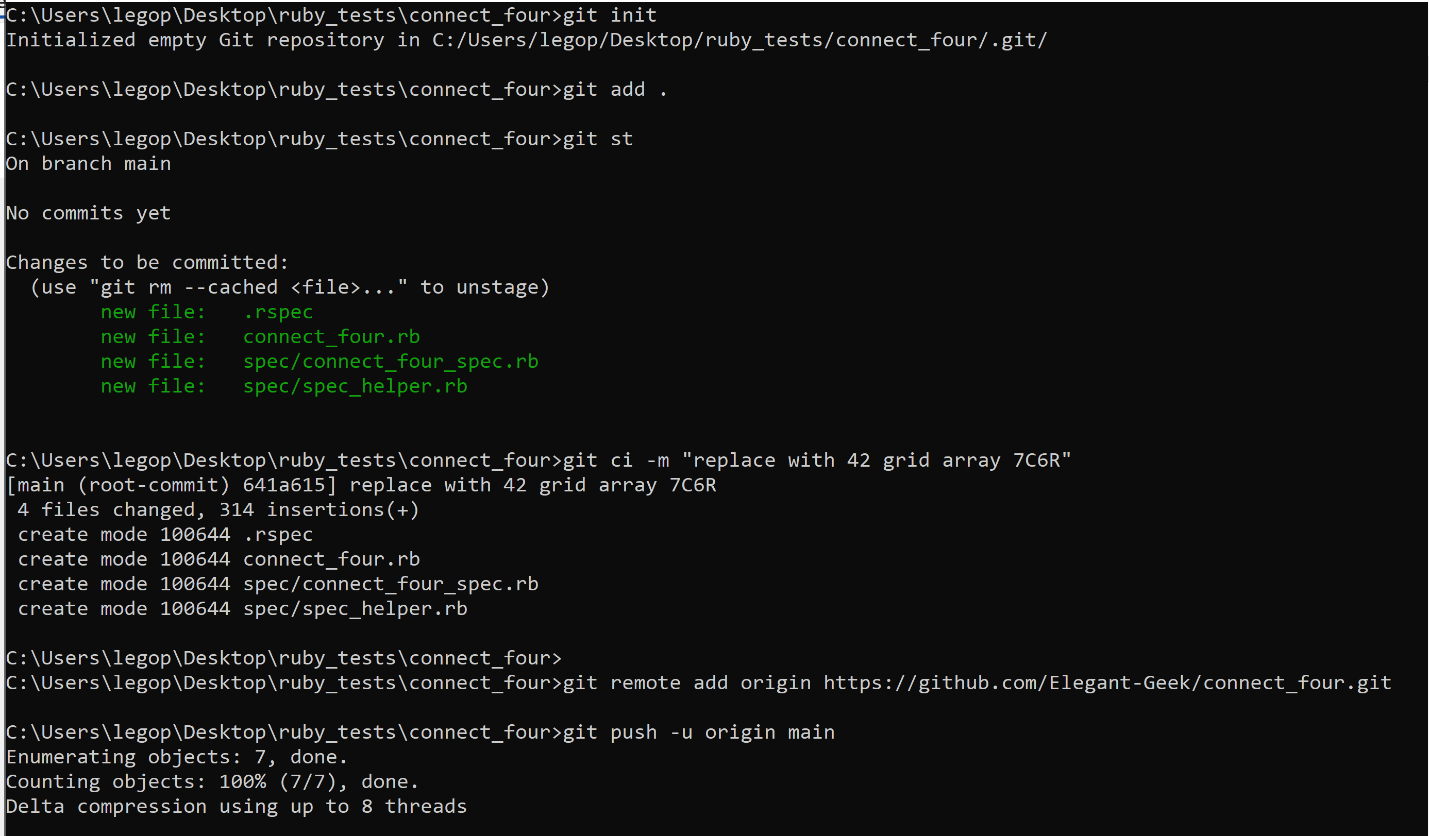
NOTE: you can still rename and relocate files WITHIN your project folder, but once you move the entire project folder around and/or rename it, all the local data (.git file in the project) is lost.

**TLDR: do not move or rename your main project file or else all local data (and your portal to push changes to your github repo) will be lost!**



****

**NOTE:** once you screw up where your files are stored, (aka if git remote -v isn’t showing that changes will push to the correct repo), then you have to do a git pull and then work from there. As shown above, it looks like you can’t tell it where to be remotely or set the remote relative to where you are (you are in project directory) Idk where the original git and git log info for the original project is stored but it seems like once you pull the project from github then make changes, you’ll be ok? GIT SUCKS! The hint says that there is another repo (my original which is hidden somewhere on my computer I guess) that was once used to push the changes before…



A SUCCESSFUL START TO FINISH INIT AND PUSHED NEW REPOSISTORY 😊