**https://github.com/EliArad/gitlearn.git**

Create in git a new project – also known as a new repository

**…or create a new repository on the command line**

echo "# gitlearn" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/EliArad/gitlearn.git

git push -u origin master

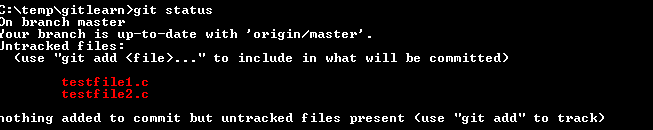
**…or push an existing repository from the command line**

git remote add origin https://github.com/EliArad/gitlearn.git

git push -u origin master

Adding two files , before add and commit using joe editor for windows

git status to show the current status of the local branch

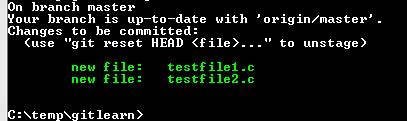


Do git add each file or git add –u for all files that are changed

For the first time, you can select only git add for each file

git add testfile1.c testfile2.c

After git add , it will show in git status the change to be commited:



Before pushing to the github server , we need to do commit and add comment about the commit

We can do:

Git commit –m “ message”

Or just git commit and an default editor will be open:

Pushing to remote:

We can do git remote –v

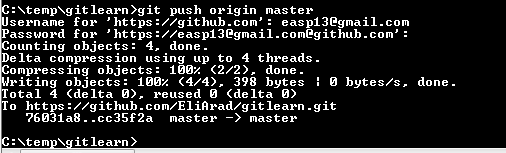


We will get the remote because we created the branch using the first step above

git remote add origin https://github.com/EliArad/gitlearn.git

If we did not have any remote yet , we can add using the git remote command

Now we can push to remote the local commit



Until here it was the easy part.

When working with git always with the same local branch , without using two cloning copies , it is always easy to sync up.

We do git pull and git push.

This document will try to show how to work with two local branches , how to update and pull individual change and more.

So let’s clone another copy to another directory:

Create a directory gitlearn1 and clone to it:

git clone <https://github.com/EliArad/gitlearn.git>

we have our last version:



Now , lets edit in one branch the file testfile1.c

void main ()

{

printf("version 2.0");

}

We want to return to base , we want to ignore that change.

We can do

git checkout \*

for all files

Or just the modified one.

Now that we do git status:

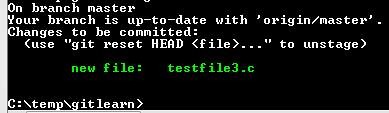
We get:

Your branch is up-to-date with 'origin/master'.

Now , lets change the file again

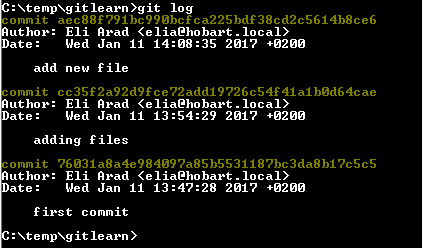
And go to a different branch to add new file

Testfile3.c



And let’s commit that change:

Git log , will show us the history of our commits:



Note to the commit SHA name

commit **aec88f791bc990bcfca225bdf38cd2c5614b8ce6**

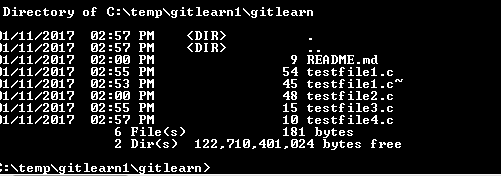
we will talk about it later.

Now , lets return to the second branch , we want to get only the change that we commit:

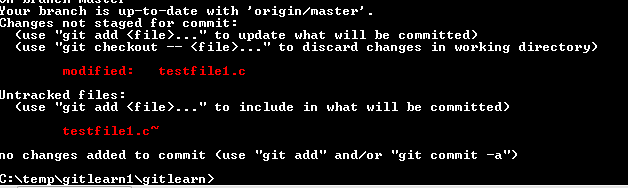


We need to get our testfile3.c

Git pull , will bring the file that is exists on remote origin master without touching the local files



If we do again git status, we see that testfile1.c is modified but was not changed when we did pull



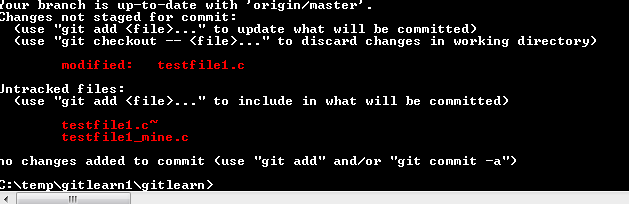
The way I am working is not to let git or any other version control to do the merge for me,

To avoid it , I use another branch , and merge the changes using compare tool.

So I did small change in a c file in one branch and commit it and push to server.

Two options

1. Save the file in a different name to be prepare for checkout
2. Create another local clone, get latest version , do merge and commit



Now let’s do checkout to discard our local change, but we did saved the testfile1\_mine.c

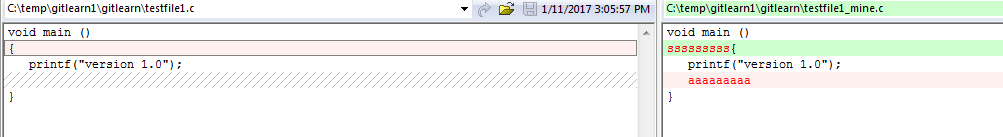
Note , git pull will not do the pull in that case and will prompt that we need to do merge

We cannot do just checkout , we need to force checkout!!

git fetch

git checkout origin/master <filepath>

Using beyond compare in windows or meld tool in linux we can do the manual merge:



Copy from right to left:

Save and do commit and push.

Now we can sync it up.