

Git & GitHub (not the same thing).

What is Git ? (Git is local on the laptop, or on computer)

- Helps to provide code versioning.

You are writing code, and you have written code for feature1.

You are writing code, for feature2, and there is a realization, that things are Breaking, so we want to go to feature1 back.

So in version control systems, versions are maintained, and we can go back to the previous version.

For big projects:

- multiple people work on the project. So how do people collaborate?

- base code is taken by three developers. Say each developer works on different features

Base code

- developer-1 (feature 1)
- developer-2 (feature 2)
- developer-3 (feature 3)

If there is no collaboration, there will be sequential process,
 But this can't be happening. So, we need to take the individual feature code,
 And start working on that.

```
#
=====
=====
```

```
#
=====
=====
```

To check whether git is installed, just do:

Git --version..

1st scenario:

Start by creating a repository in GitHub,
 and then the entire procedure is done.

```
((base) luv@Luvs-MBP ApacheSparkProject % git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

```
((base) luv@Luvs-MBP ApacheSparkProject % git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        main.py

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

If the existing file is modified git status will say modified.

If a new file (say main.py) is added in the above case, then it will say Untracked files.

How to commit these changes? Because whenever we make changes, By default the changes are unstaged. We have to stage these changes.

Making changes/modifying files (Unstaged) –
> stage → commit

1. Make changes/ add new files.
2. We need to stage the changes.
(So that they re to be committed)
3. We can commit all the staged changes.

So how to stage the changes? Or new files?

Git add . -> any changes which are unstated, please stage them.

now if we check, then these changes are ready to be committed, as
they are staged.

```
(base) luv@Luvs-MBP ApacheSparkProject % git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
        new file:   main.py

(base) luv@Luvs-MBP ApacheSparkProject %
```

#

```
=====
=====
```

After the changes are staged, we need to put commit them.
So how to commit these changes ?

```
(base) luv@Luvs-MBP ApacheSparkProject % git commit -m "making changes to readme and adding a new main file"
[main bd31031] making changes to readme and adding a new main file
 2 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 main.py
(base) luv@Luvs-MBP ApacheSparkProject %
```

git commit -m "making changes to readme and

adding a new main file"

```
[(base) luv@Luvs-MBP ApacheSparkProject % git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
(base) luv@Luvs-MBP ApacheSparkProject %
```

now if you see there is nothing to commit further.

#

```
=====
=====
```

So in short,

1. Modify, add, delete the files whatever changes are necessary according to the work.

2. `git add .`

3. `git commit -m "meaningful message"`

#

```
=====
=====
```

Now it says that local (laptop) is ahead of

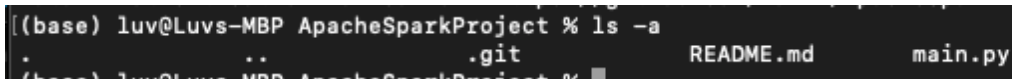
remote (GitHub) by 1 commit.
After doing `git commit -m`, still on the
GitHub, the push hasn't yet happened.
So we have to push the changes.. to
wherever we want to push.

4. Get a token (personal access token) and
put the following:

```
git remote set-url origin https://  
{personal_access_token}@github.com/luv91/  
ApacheSparkProject
```

`personal_access_token` need to be generated
via Github.

5. `git push origin main`



```
((base) luv@Luvs-MBP ApacheSparkProject % ls -a  
.  
..  
.git  
README.md  
main.py  
((base) luv@Luvs-MBP ApacheSparkProject %
```

`.git` folder → would be there for any
folder which is tracked by git.

Now if you do `git status` and check, there
is nothing to commit:

```
012070711b031001 main -> main
[(base) luv@Luvs-MBP ApacheSparkProject % git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
(base) luv@Luvs-MBP ApacheSparkProject % █
```

#

=====

=====

Following are the steps:

1. We created a repository on Github.
2. We cloned that repository on our local.
3. We made a few changes and added new files.
4. You will commit the changes to git (2 step process).
 - a. git add . (To stage the changes)
 - b. git commit -m "meaningful message" (to commit it)
5. To push to main branch (to push the changes to GitHub), again a two step process:

first do authentication if asked:

a. `git remote set-url origin https://{personal_access_token}@github.com/luv91/`
`ApacheSparkProject`

b. `git push origin main`

#

=====

=====

```
((base) luv@Luvs-MBP lending_club_project % git init
Initialized empty Git repository in /Users/luv/Documents/BigDataTrendyTech/ApacheSparkProject/files_for_github/lending_club_project/.git/
((base) luv@Luvs-MBP lending_club_project % ls -a
.
..
.DS_Store
.git          config       lib          logger.py    main.py
((base) luv@Luvs-MBP lending_club_project %
```

Scenario 2. Currently, we are somewhere in some folder,

Which is not a GitHub repository, and we have to tell GitHub to start managing it.

`git init`. Doing that will add a `.git` folder into it.


```

((base) luv@Luvs-MBP lending_club_project % git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .DS_Store
        config/
        lib/
        logger.py
        main.py

nothing added to commit but untracked files present (use "git add" to track)
(base) luv@Luvs-MBP lending_club_project %

```

Now bringing it to the staging area.

```

((base) luv@Luvs-MBP lending_club_project % git add .
((base) luv@Luvs-MBP lending_club_project % git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .DS_Store
        new file:   config/project.conf
        new file:   config/spark.conf
        new file:   lib/transformations.py
        new file:   lib/utils.py
        new file:   logger.py
        new file:   main.py

(base) luv@Luvs-MBP lending_club_project %

```

```

((base) luv@Luvs-MBP lending_club_project % git commit -m "adding intial set of files"
[main (root-commit) abda69e] adding intial set of files
 7 files changed, 7 insertions(+)
 create mode 100644 .DS_Store
 create mode 100644 config/project.conf
 create mode 100644 config/spark.conf
 create mode 100644 lib/transformations.py
 create mode 100644 lib/utils.py
 create mode 100644 logger.py
 create mode 100644 main.py
(base) luv@Luvs-MBP lending_club_project %

```

Currently they are all on local git. They are not on remote.

```
create mode 100644 main.py  
[(base) luv@Luvs-MBP lending_club_project % git status  
On branch main  
nothing to commit, working tree clean  
(base) luv@Luvs-MBP lending_club_project %
```

Now we have to push it to the remote..

Can I say here.. git push origin main.
(Here origin means the place, from where the cloning has happened),
But when it is from the local, we are not cloning it from anywhere.

Thus there is no origin currently for this project.

When we clone a remote project the origin is referring to that.

But when we start developing on local. Then we need to set the origin.

So first create an empty repository and get its url: <https://github.com/luv91/lendingclubproj.git>

So command is: git remote add origin <https://github.com/luv91/>

`lendingclubproj.git`

The above command means, our origin is set now..

To verify the origin: `git remote -v`

```
(base) luv@Luvs-MBP lending_club_project % git remote -v
origin  https://github.com/luv91/lendingclubproj.git (fetch)
origin  https://github.com/luv91/lendingclubproj.git (push)
(base) luv@Luvs-MBP lending_club_project %
```

#

=====

=====

On which branch we are: `git branch`

```
(base) luv@Luvs-MBP lending_club_project % git branch
* main
(base) luv@Luvs-MBP lending_club_project %
```

Let's say instead of main, it was a master branch, then we would have to Change it to main branch.

To change from master to main branch.

`git branch -M main` ; where -M means Modify.

```
[(base) luv@Luvs-MBP lending_club_project % git branch -M main
[(base) luv@Luvs-MBP lending_club_project % git branch
* main
(base) luv@Luvs-MBP lending_club_project %
```

#

```
=====
=====
```

Since the origin is set now.. we can do the following:

First, on terminal
git remote set-url origin https://
{personal_access_token}@github.com/luv91/
lendingclubproj

Then: Git push origin main

#

```
=====
=====
```

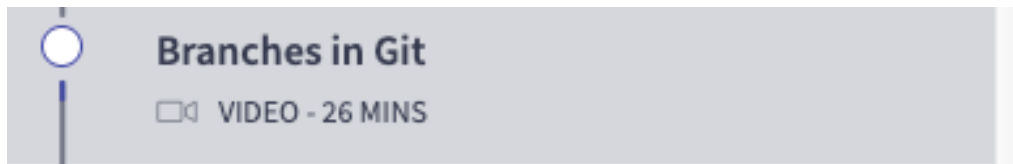
If you say the following: git push -u
origin main
Then, for the next command, you just have
to say: git push; and not add the origin
main.

#

=====

=====

Branches in Git:



There is something called as a main branch..

How to check on which branch you are? ->
git branch

How to change a branch name -> git branch
-M main (if we need to change the name to
main).

Question: How do you create a new branch?

-> create a branch : git branch feature1
(feature1 is the name of the branch
currently).

```
[(base) luv@Luvs-MBP lending_club_project % git branch feature1
[(base) luv@Luvs-MBP lending_club_project % git branch
feature1
* main
(base) luv@Luvs-MBP lending_club_project %
```

As can be seen now there are two branches:

feature1, and main.

Currently in the main branch.

How to change branch, that is how to go to feature1 branch? : git checkout feature1

```
[(base) luv@Luvs-MBP lending_club_project % git checkout feature1  
Switched to branch 'feature1'  
[(base) luv@Luvs-MBP lending_club_project % git branch  
* feature1  
  main  
(base) luv@Luvs-MBP lending_club_project % ]
```

So now we are at feature1 branch..
We should not touch the main branch..
different people have their own different
Branches.

Say if we have to create the feature2
branch.. what code will come in it?

Currently we are. At feature1 branch, and
if we do git branch feature2

Then the code will be taken from the
feature1 branch, since we are at feature1
currently.

If we do the command: git checkout -b
feature2 (it will do two things at once,

first it will create feature2 branch, and next it would checkout to that branch).

```
[(base) luv@Luvs-MBP lending_club_project % git checkout -b feature2
Switched to a new branch 'feature2'
[(base) luv@Luvs-MBP lending_club_project % git branch
feature1
* feature2
main
(base) luv@Luvs-MBP lending_club_project %
```

Created and switched to feature2.

so: `git checkout -b feature2` (to create and navigate to a new branch).

#

=====

If I do : `git checkout -b feature3` (this will currently take the code from feature2).

What if I want it to take the code from main?

- `git checkout main`
- `git checkout -b feature3`

But let's say I am on main and I am creating a branch feature3, and want it to

have the
Base code from feature2.

- git checkout -b feature3 feature2
(feature2 here signifies where to take the
code from).

#

=====

What if the branch needs to be deleted.

To delete a particular branch, you should
be on some other branch,,

Say want to delete feature3, can't if you
are on feature3

Command to delete a branch.. : git branch
-d feature3

```
((base) luv@Luvs-MBP lending_club_project % git branch -d feature3  
error: Cannot delete branch 'feature3' checked out at '/Users/luv/Documents/BigDataTrendyTech/SparkProject/files_for_github/lending_club_project'
```

```
((base) luv@Luvs-MBP lending_club_project % git checkout feature2  
Switched to branch 'feature2'  
((base) luv@Luvs-MBP lending_club_project % git branch -d feature3  
Deleted branch feature3 (was abda69e).  
((base) luv@Luvs-MBP lending_club_project % git branch  
feature1  
* feature2  
main  
((base) luv@Luvs-MBP lending_club_project %
```


We moved to feature2 branch, and then deleted feature3, and that is possible..

say we modified a file.. (say we modified main.py)

```
(base) luv@Luvs-MBP lending_club_project % git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   main.py

no changes added to commit (use "git add" and/or "git commit -a")
(base) luv@Luvs-MBP lending_club_project % git add .
(base) luv@Luvs-MBP lending_club_project % git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   main.py

(base) luv@Luvs-MBP lending_club_project % git commit -m "adding feature1"
[main ed211a0] adding feature1
 1 file changed, 3 insertions(+), 1 deletion(-)
(base) luv@Luvs-MBP lending_club_project %
```

Now since we made the changes in the main branch..

If we switch from main to feature1 branch, we can see, that there will be on changes.

So, if we do any change.. on any branch, it says in that branch..

—>

Let's push the changes to the main..

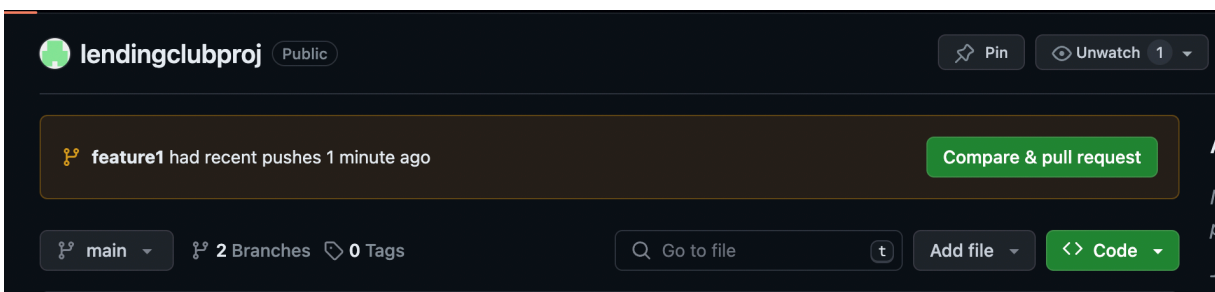
```
[(base) luv@Luvs-MBP lending_club_project % git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 10 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 296 bytes | 296.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/luv91/lendingclubproj
    abda69e..ed211a0  main -> main
(base) luv@Luvs-MBP lending_club_project %
```

So the changes are pushed..

#

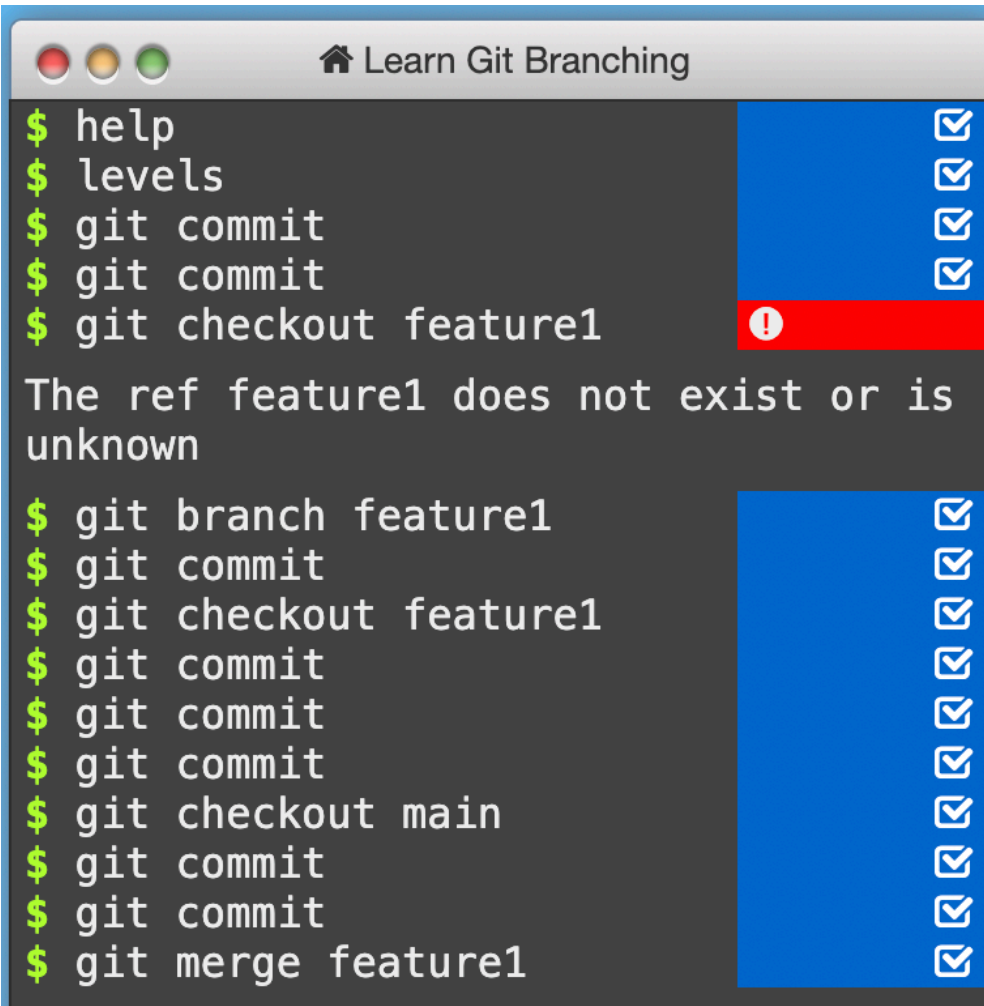
```
=====
=====
```

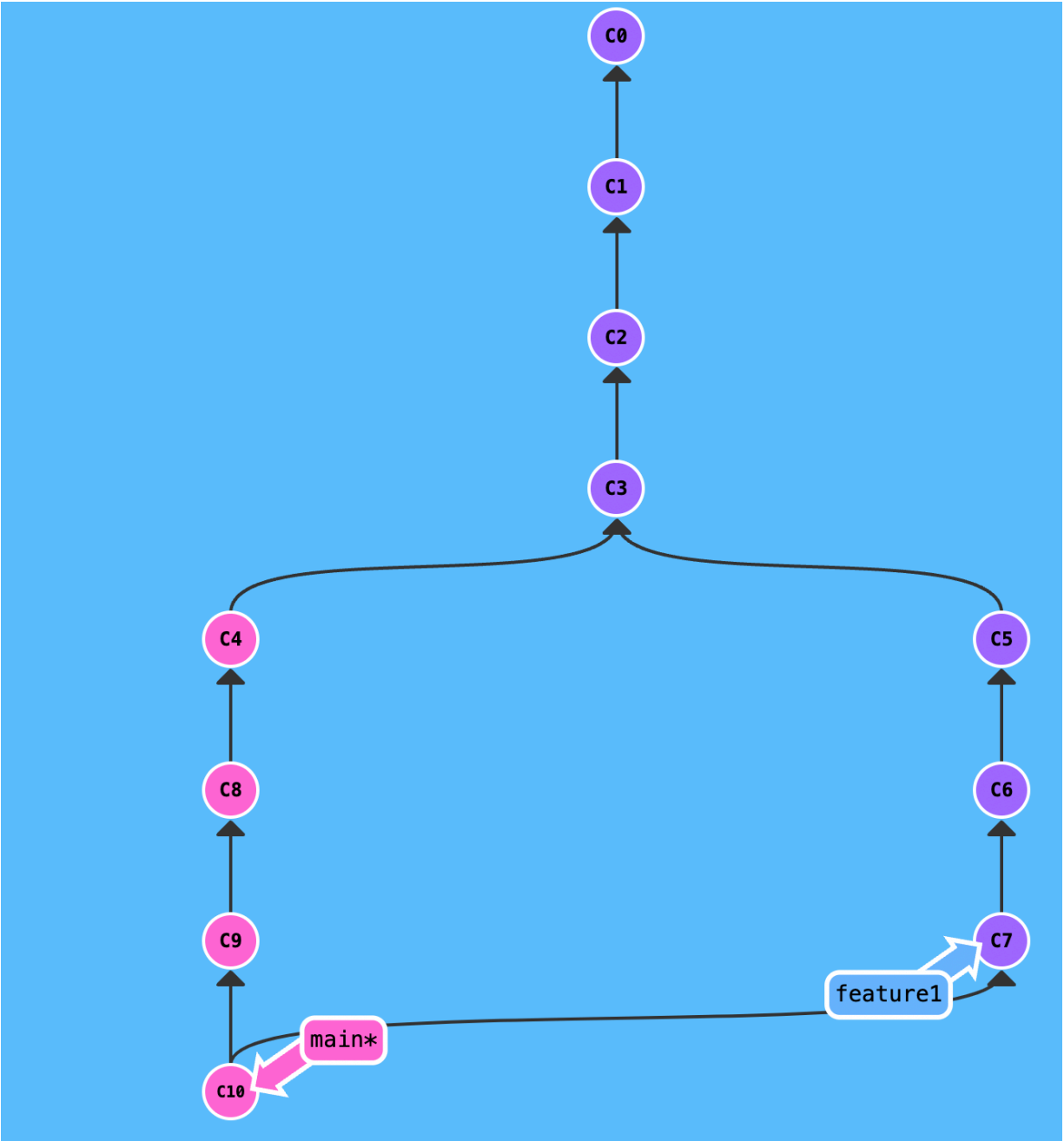
Now there are two branches, and Github shows something like:



So feature1 has changes which are not there in the main branch.. so those need to be fixed.

learngitbranching.js.org





#

=====

=====

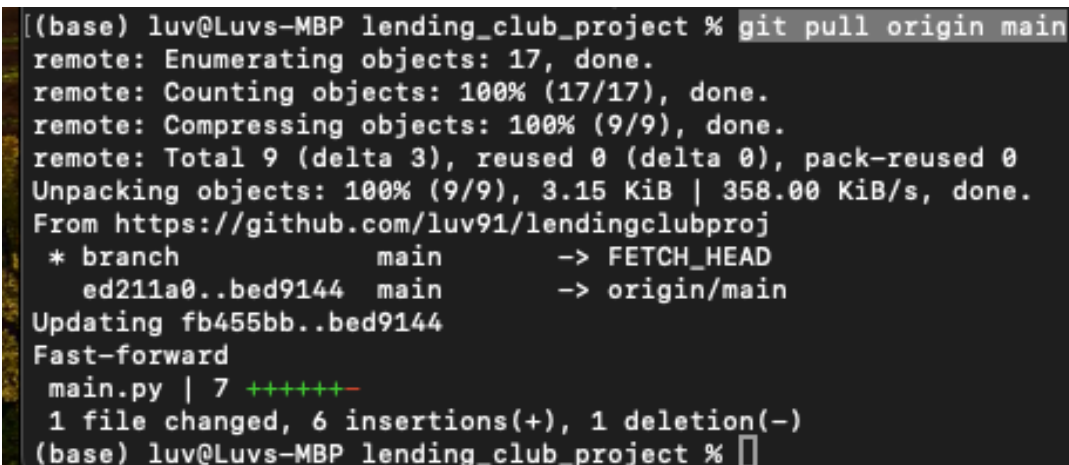
We learnt branching in git already..

If you want to merge your changes to main branch, then create a pull request.

Once the all request is approved and the code is merged, then we will see Both main and feature1 have the same code.

To make sure, local main branch is in sync with Github main branch.

```
git pull origin main
```



```
[(base) luv@Luvs-MBP lending_club_project % git pull origin main
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 9 (delta 3), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), 3.15 KiB | 358.00 KiB/s, done.
From https://github.com/luv91/lendingclubproj
 * branch          main          -> FETCH_HEAD
    ed211a0..bed9144  main        -> origin/main
Updating fb455bb..bed9144
Fast-forward
 main.py | 7 ++++++
 1 file changed, 6 insertions(+), 1 deletion(-)
(base) luv@Luvs-MBP lending_club_project %
```

When you are making changes and you feel you did something wrong in the codebase and you

Want to go back to the previous code base then how to achieve that?

How to make changes, so that we can go back to the previous changes..

#

=====

=====

There are 3 scenarios:

1. Scenario 1: you made changes which are not even staged.
2. Scenario 2: you made changes and have staged those changes.
3. Scenario 3: you made changes and have committed those changes

Talking about Scenario 1:

you made changes which are not even staged.

```
nothing to commit, working tree clean
(base) luv@Luvs-MBP lending_club_project % git status
On branch feature1
Your branch is up to date with 'origin/feature1'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   config/project.conf
        modified:   config/spark.conf

no changes added to commit (use "git add" and/or "git commit -a")
(base) luv@Luvs-MBP lending_club_project %
```

I can just do: `git restore config/`

project.conf

This will restore the changes in project.conf in config file.

#

=====

Scenario 2: I made the changes, and I will stage these changes.. but will not commit them..

How do I stage -> git add .

```
[(base) luv@Luvs-MBP lending_club_project % git add .
[(base) luv@Luvs-MBP lending_club_project % git status
On branch feature1
Your branch is up to date with 'origin/feature1'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   config/project.conf
        modified:   config/spark.conf
```

So these are stages.. ready to be committed.

What if I want to get rid of these changes?

Git restore --staged config/project.conf

If we do git status now.. we see that one

of the change, is brought to the Previous level.. and that's why project.conf is in red..

```
on branch feature1
Your branch is up to date with 'origin/feature1'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   config/spark.conf

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   config/project.conf

(base) luv@Luvs-MBP lending_club_project %
```

Again we will do, git restore config/project.conf

This will then bring it to the original code base..

So this one is a 2-step process.

So if it is staged, first you onstage them and then restore the files.

#

```
=====
=====
```

Senario 3: you made changes and have committed those changes

Git log → shows all the commits. (With most recent commit at the top).

- git reset --hard hashvalue (there will be some hash value)..

- git reset --hard
84a3810cd43970c41cb5f20ecbf6d90eab1c2c6e

#

=====

=====

Git Stash

Say what if I have made changes, and they are not committed yet.. I want to park the changes,
Somewhere like back side.. and later get them wherever required..

```
(base) luv@Luvs-MBP lending_club_project % git status
On branch feature1
Your branch is up to date with 'origin/feature1'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   config/project.conf

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   config/spark.conf
        modified:   lib/transformations.py

(base) luv@Luvs-MBP lending_club_project %
```

Keep the changes at a place, so that later I can retrieve it..

- git stash..

```
[(base) luv@Luvs-MBP lending_club_project % git stash  
Saved working directory and index state WIP on feature1: 7f85a0a committing changes to spark.conf  
(base) luv@Luvs-MBP lending_club_project % █
```

```
[(base) luv@Luvs-MBP lending_club_project % git status  
On branch feature1  
Your branch is up to date with 'origin/feature1'.  
  
nothing to commit, working tree clean  
(base) luv@Luvs-MBP lending_club_project % █
```

The changes will now not be visible in the files.. they will be gone from the files.. But where are they gone?

To get back the Changes -: git stash pop

```
[(base) luv@Luvs-MBP lending_club_project % git stash pop  
On branch feature1  
Your branch is up to date with 'origin/feature1'.  
  
Changes not staged for commit:  
  (use "git add <file>..." to update what will be committed)  
  (use "git restore <file>..." to discard changes in working directory)  
    modified:   config/project.conf  
    modified:   config/spark.conf  
    modified:   lib/transformations.py  
  
no changes added to commit (use "git add" and/or "git commit -a")  
Dropped refs/stash@{0} (0b1dd293c8fe6b146c0459d28099af64ce649cd7)  
(base) luv@Luvs-MBP lending_club_project % █
```

After git stash pop,, everything is unstaged now..

#

=====

Git stash (to park the changes somewhere)

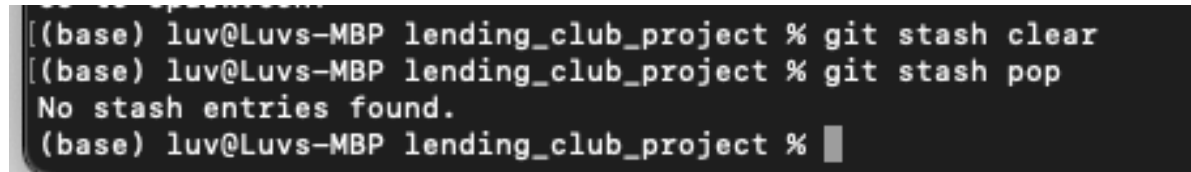
Git stash pop (to get the changes back).

#

=====

What if I want to get rid of these changes altogether?

– git stash clear

A terminal window with a dark background and light-colored text. It shows the following commands and output:

```
[(base) luv@Luvs-MBP lending_club_project % git stash clear
[(base) luv@Luvs-MBP lending_club_project % git stash pop
No stash entries found.
(base) luv@Luvs-MBP lending_club_project %
```

When we tried to retrieve the changes again, they are gone.. can't come back.

#

=====

=====

Let's say two set of things are edited now..

```
[(base) luv@Luvs-MBP lending_club_project % git stash  
Saved working directory and index state WIP on feature1: 7f85a0a committing changes to spark.conf  
[(base) luv@Luvs-MBP lending_club_project % git stash list  
stash@{0}: WIP on feature1: 7f85a0a committing changes to spark.conf  
stash@{1}: WIP on feature1: 7f85a0a committing changes to spark.conf  
(base) luv@Luvs-MBP lending_club_project %
```

- git stash pop -> it will pop up the top most thing..

- git stash save "changes in conf files"

These messages will appear now when we do git stash list.

```
1100  
[(base) luv@Luvs-MBP lending_club_project % git stash list  
stash@{0}: On feature1: chagnes in transformation files  
stash@{1}: On feature1: change in conf files  
stash@{2}: WIP on feature1: 7f85a0a committing changes to spark.conf  
(base) luv@Luvs-MBP lending_club_project %
```

- after we do git stash pop, and then do git stash list, we will see the bottom of the stack thing ..

```

((base) luv@Luvs-MBP lending_club_project % git stash pop
On branch feature1
Your branch is up to date with 'origin/feature1'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   lib/transformations.py

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (67bc385e3555515164ea9069e0580d316c03cf18)
((base) luv@Luvs-MBP lending_club_project % git stash list
stash@{0}: On feature1: change in conf files
stash@{1}: WIP on feature1: 7f85a0a committing changes to spark.conf
((base) luv@Luvs-MBP lending_club_project %

```

#

```

=====
=====

```

If we are creating a new file.. and trying to stash it it doesn't get stashed by default..

So to stash them, (so to stash even the untracked file)..

- git stash -u (-u means untracked, with this even the untracked file will be stashed.

#

```

=====
=====

```

