



# git

## LEARNING GIT

Git Fundamentals

### ABSTRACT

This is where we learn to use git and github, to make development of our group project easier and accessible from anywhere

T G CHIPOYERA

Learning Git

## Introduction to git

Wadup dawgs, this is an intro to git and how to use the basics of git to finish our pracs.

Definition from wiki - Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

Basically, we can use git to manage our files.

With git, it can check who changed a file, added a file, deleted a file and everything a person does to the repo(A software repository, or “repo” for short, is a storage location for software packages)

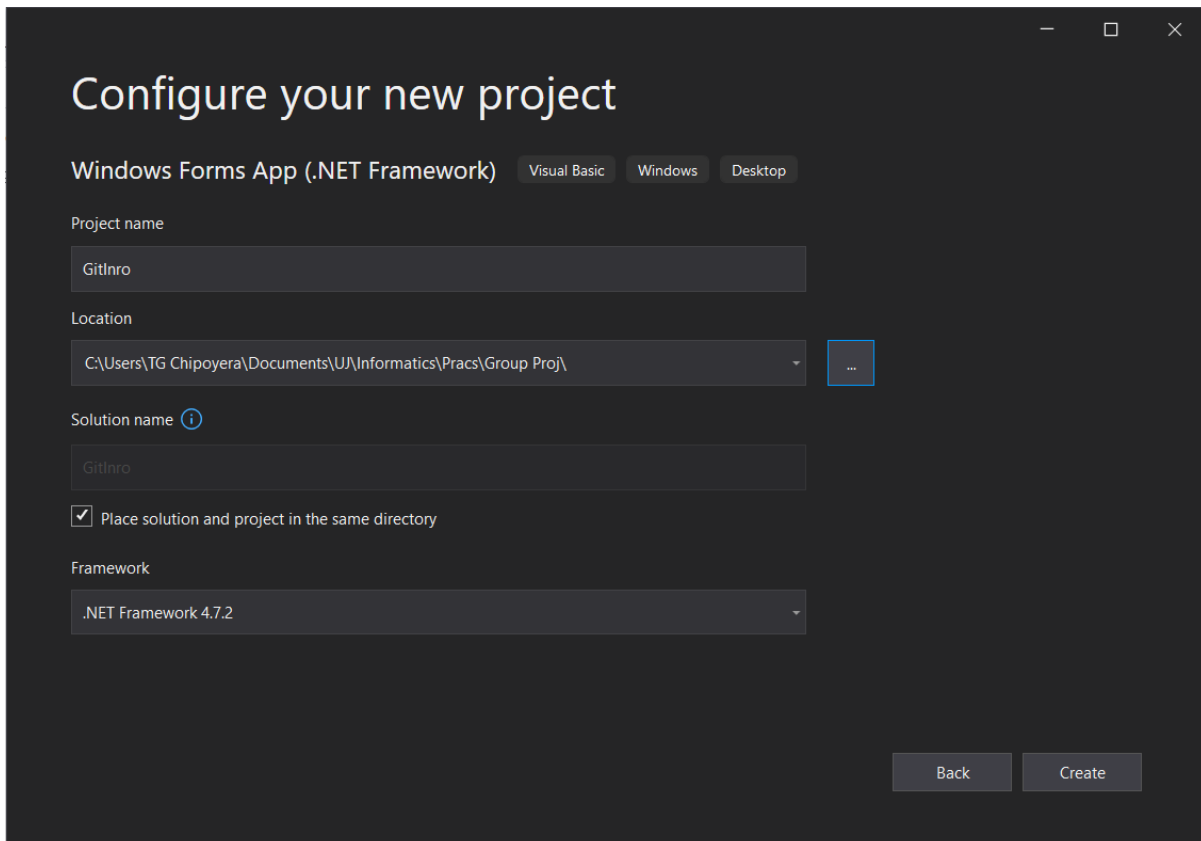
# **Part 1**

## Demonstration of git

To check if you have git installed, go to any command line and type this, if you get any version number, it's installed and if not, you have to install from <https://git-scm.com/downloads>

```
TG Chipoyera@LAPTOP-FD2AHV34 MINGW64 ~  
$ git --version  
git version 2.32.0.windows.2  
  
TG Chipoyera@LAPTOP-FD2AHV34 MINGW64 ~  
$ |
```

A new project was made to demonstrate.



Configure your new project

Windows Forms App (.NET Framework) Visual Basic Windows Desktop

Project name  
GitIntro

Location  
C:\Users\TG Chipoyera\Documents\UJ\Informatics\Pracs\Group Proj\ ...

Solution name ⓘ  
GitIntro

☒ Place solution and project in the same directory

Framework  
.NET Framework 4.7.2

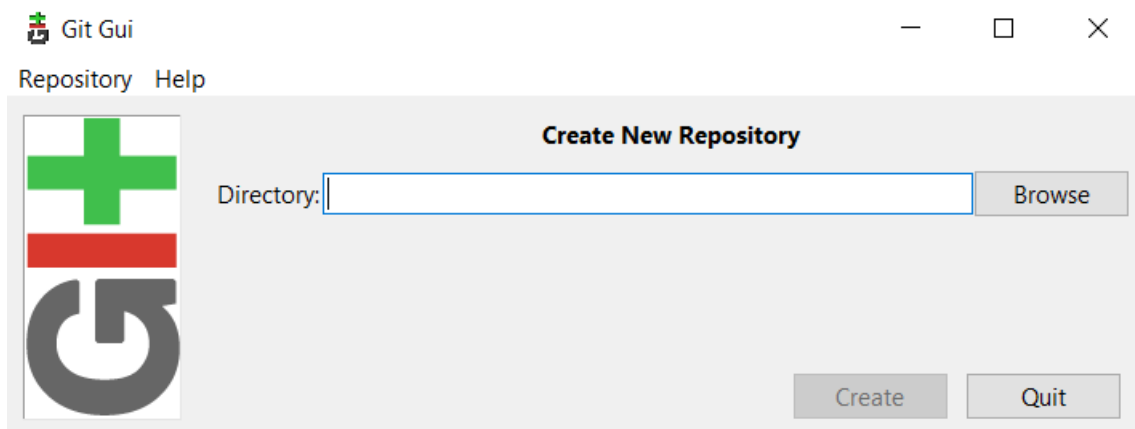
Back Create

Now when a new project has been made, in order to start using git, It needs to be initialized.



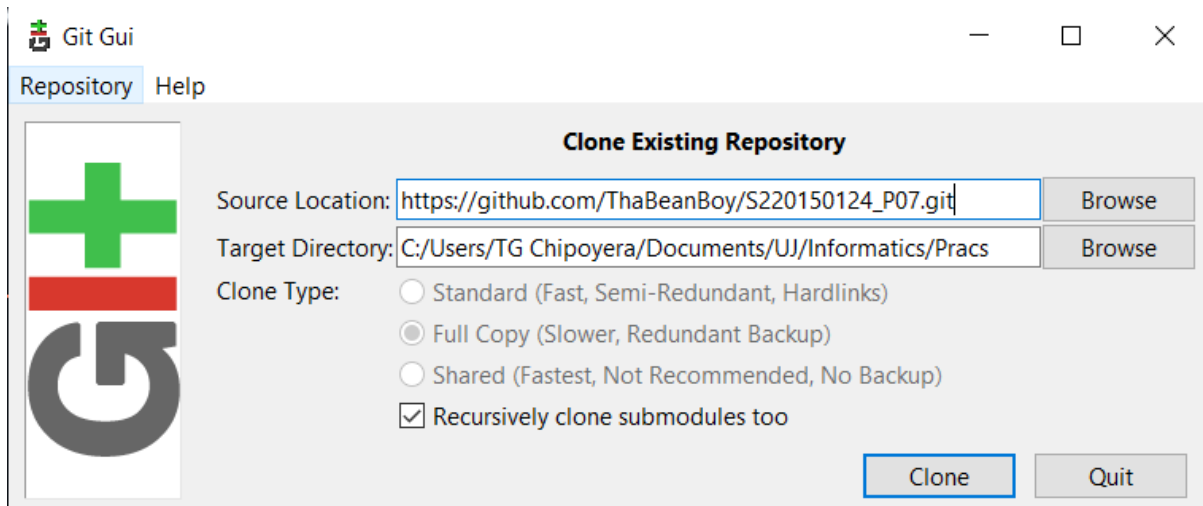
1. To initialize git, click the 'Create New Repository'. **Please be aware, you don't need to create a new repo if**

- a) You already initialized git/created the repo
- b) If you cloned a repo from an online repo like Github



I recommend clicking browse, it makes it easier to select the directory

2. Cloning a repo, when you clone a repo, there's no need to create repo or initialize git, cause git is already working in the cloned repository



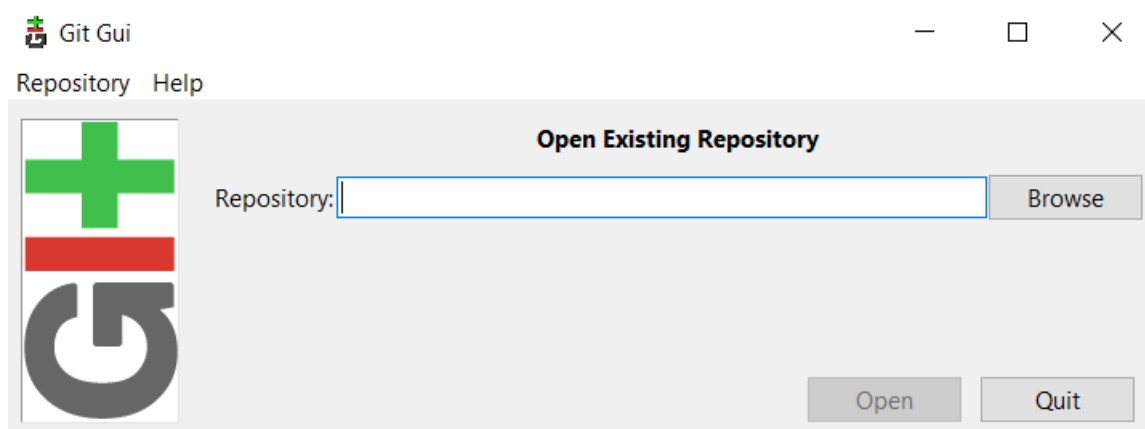
In this example, I'm cloning from an online repo, but you can also clone a repo in you machine.

The details of the repo you are cloning should go into 'Source Location'.

In the 'Target Directory', select the directory in which you want the repo to be, **If you errors saying that 'xyz already exist blah blah blah,' at the end of the text, add '/DirectoryName' . If it doesn't work, hit me up on discord real quick**

3. To opening a repo, you can click one of you recent repositories or click 'Open Existing Repository'

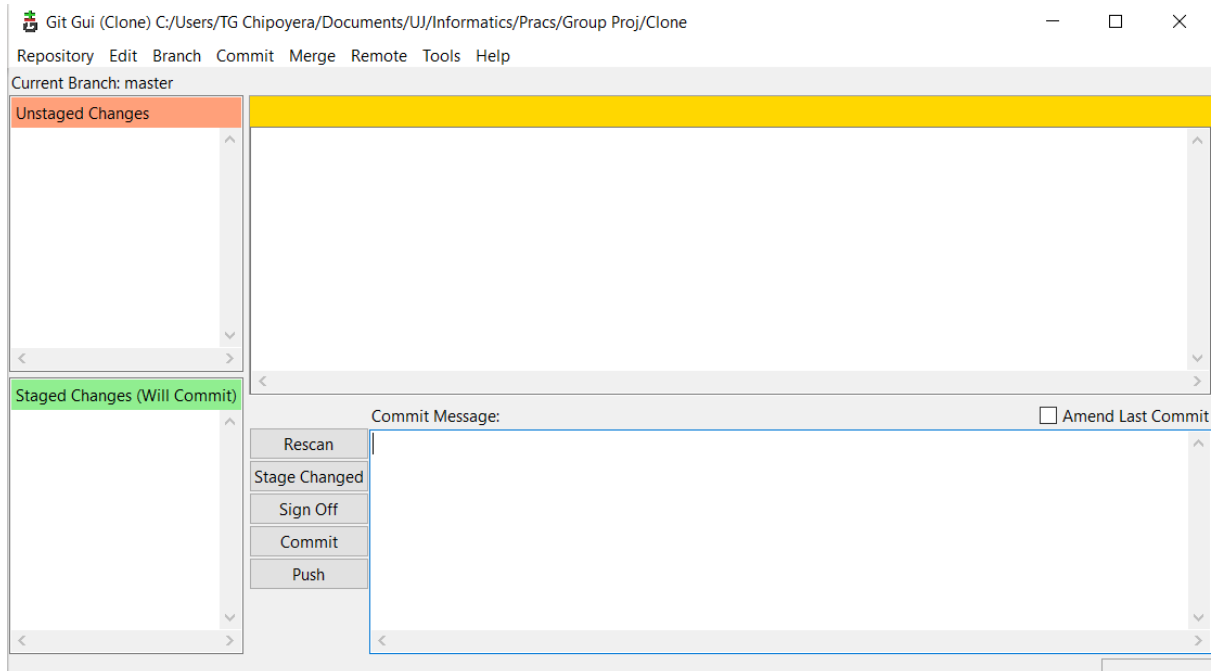
If you click 'Open Existing Repository', this will be opened



Select the repo, then you'er good to go to 2<sup>nd</sup> part

# **Part 2**

Alright, if you did anything right in part 1, it should lead you to a page that looks similar to the image below



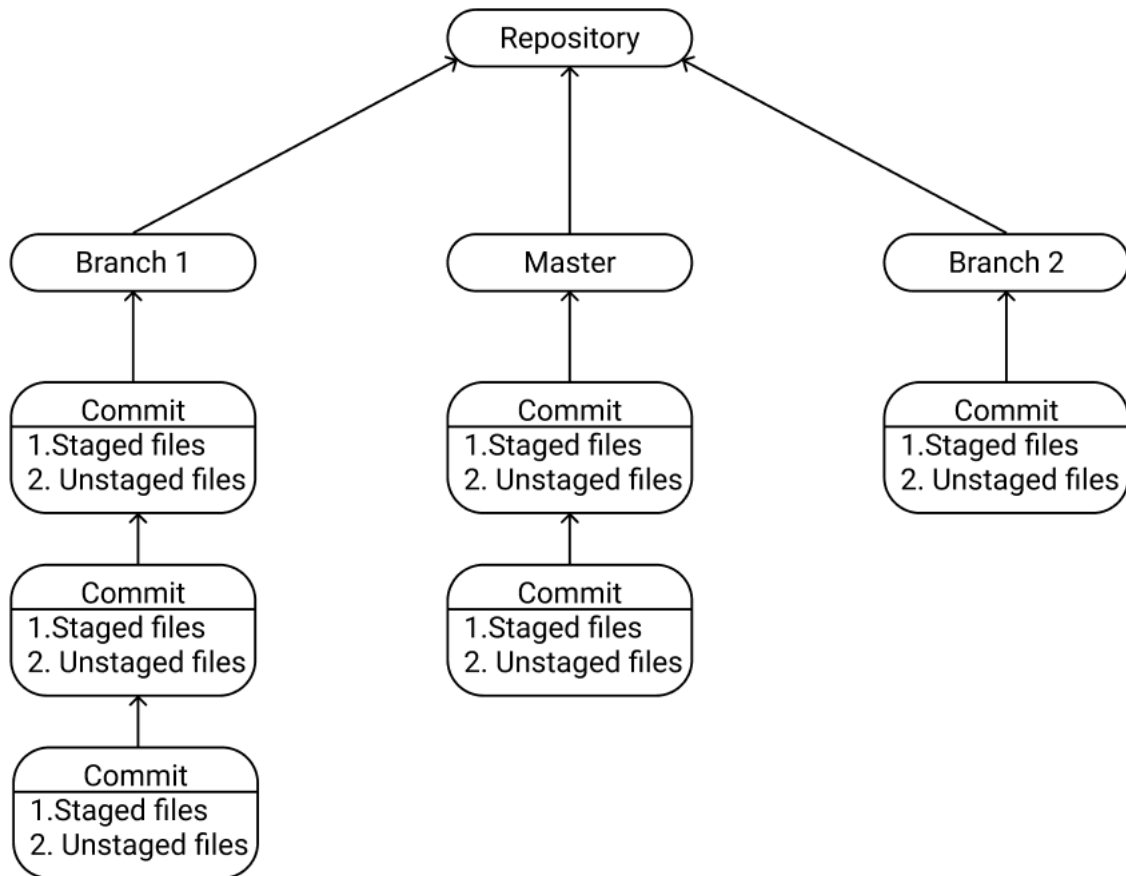
This is where we start using git in it's glory,

I'd like to break down a few things down hereCommits

1. Unstaged files
2. Staged files

Let's break things down





We'll start from the very bottom, with the files:

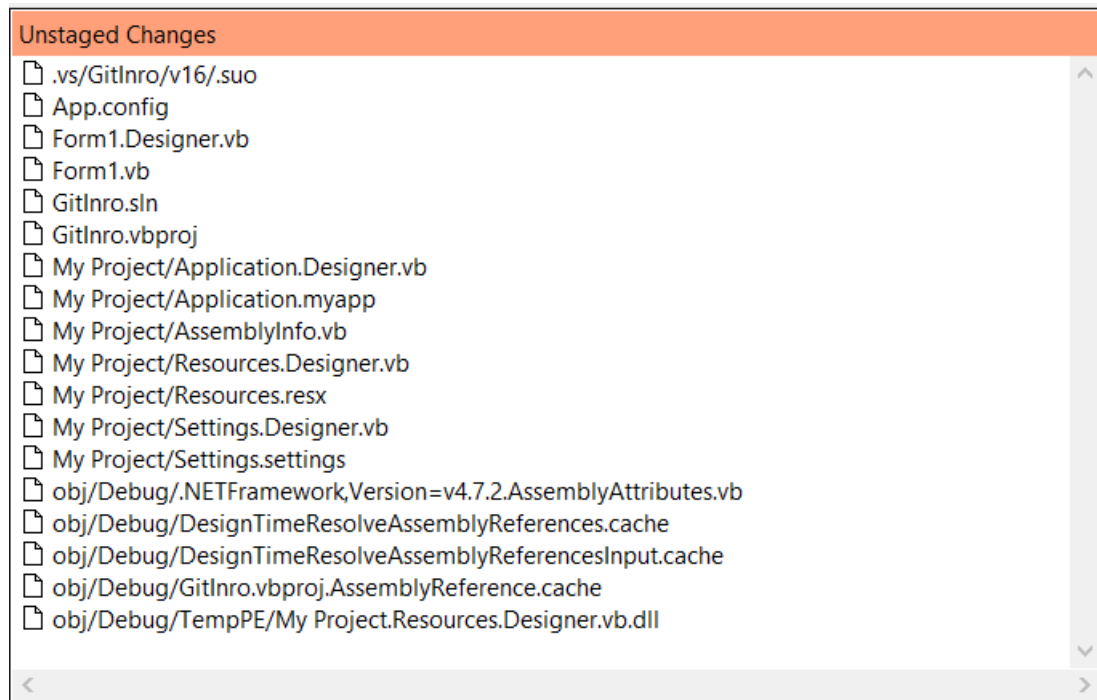
Our programme is made out of 1 or more files, these files are constantly changing because we may be adding features, debugging or anything extra.

Let's talk about staged and unstaged changes:

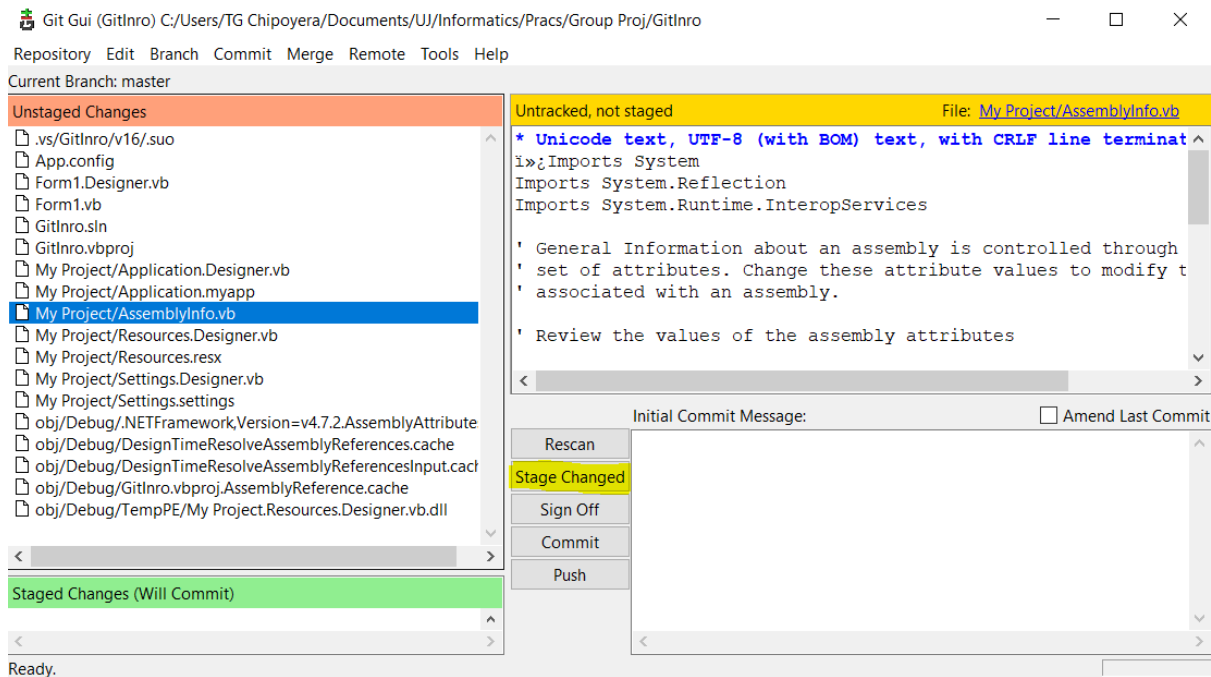
When you change and save files as you code, those files will be put in the unstaged changes, When you feel that the features you have added are working or the programme is working after debugging, The changes are moved to the Staged files,

It is important to move things to the staged part, because when you commit, the changes in unstaged changes won't be committed. Only staged changes are committed

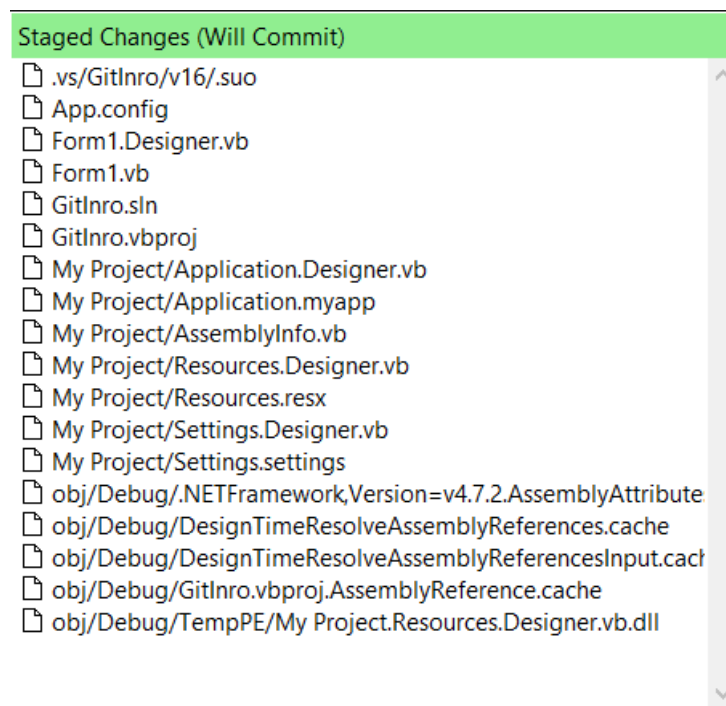
The thing about commits is, let's say your programme keeps on changes, the bugs are overwhelming you and a lot of things are happening, You can easily go back to the last commit – It's like messing up a level in games, if you messed up bad you can go back to the last savepoint.



Since the repo has been created, git will tell you that these files and directories are new to it, we'll 1<sup>st</sup> stage these then commit them

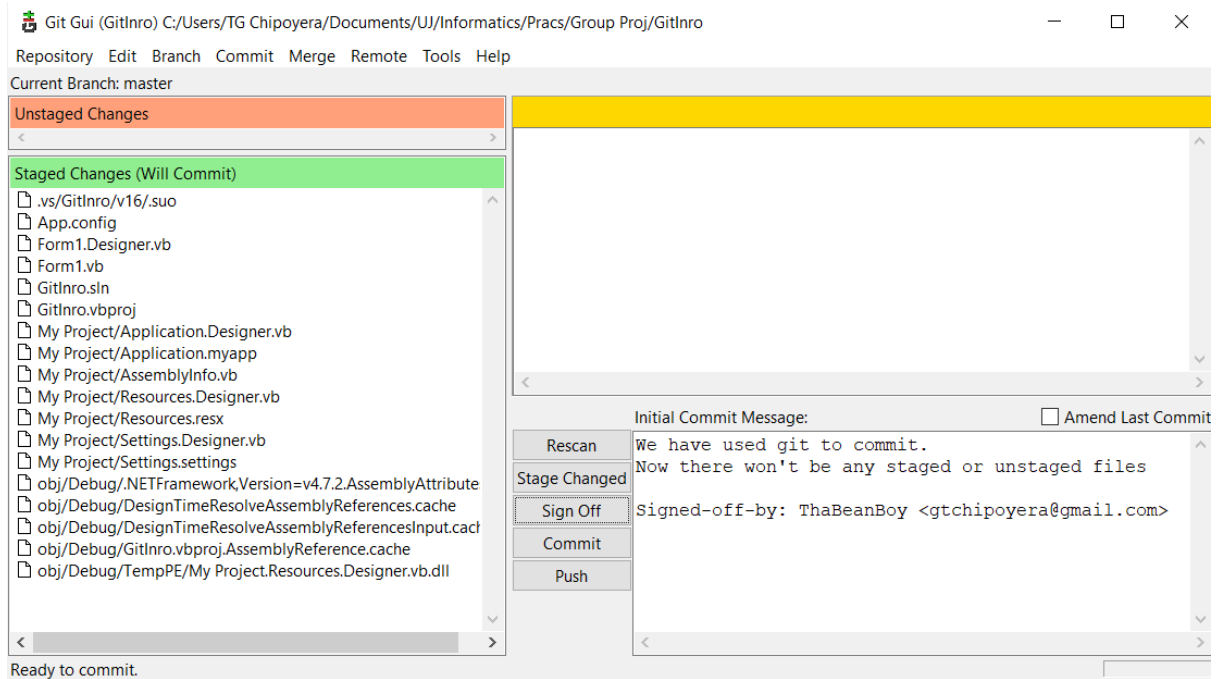


Once you click and confirm to stage all changed files, all the unstaged changes will be transferred to staged changes

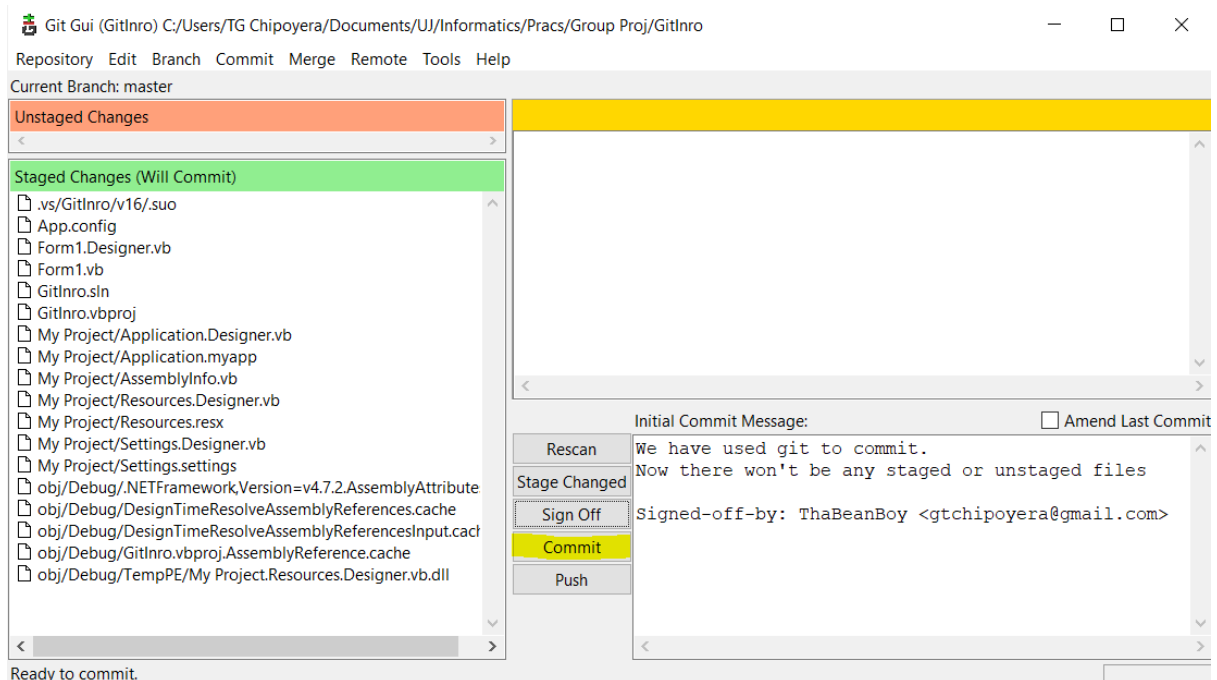


If you have any staged changes, You need to write down your commit message. Your commit message should explain what you have done,

Here's a tip, look at the staged changes and just summarise what you did in those files.



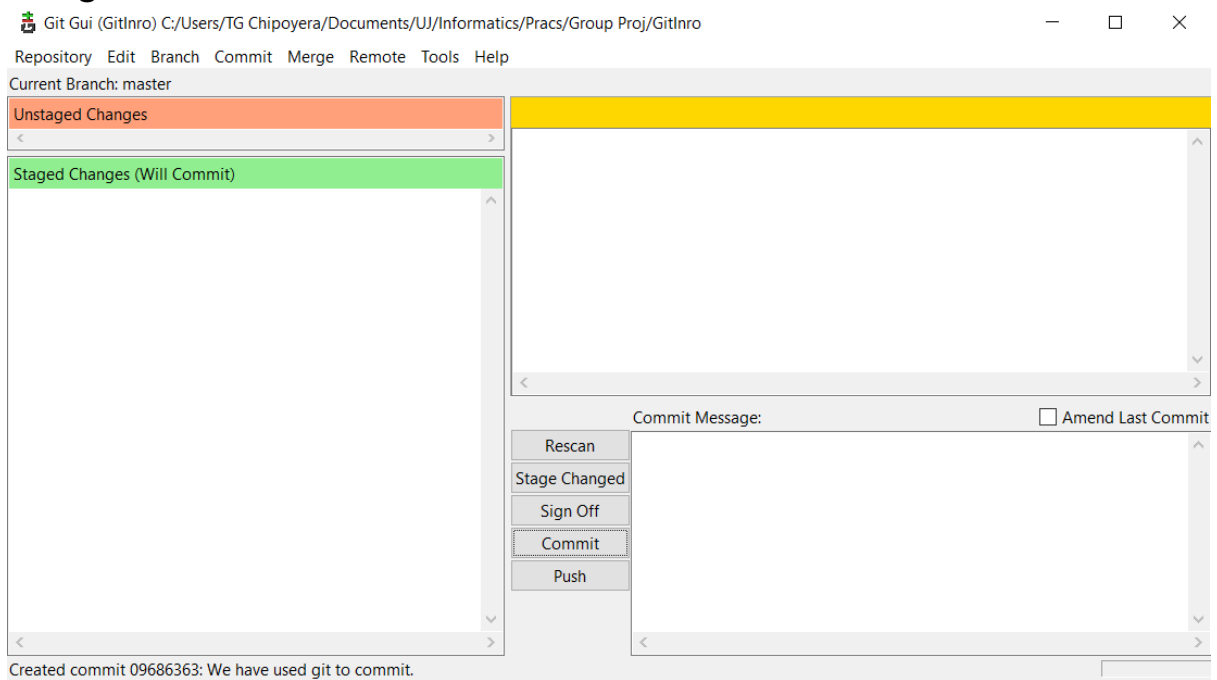
Another tip, if you press the Sign Off button, at the bottom of your commit message, git will automatically put your username and email address there



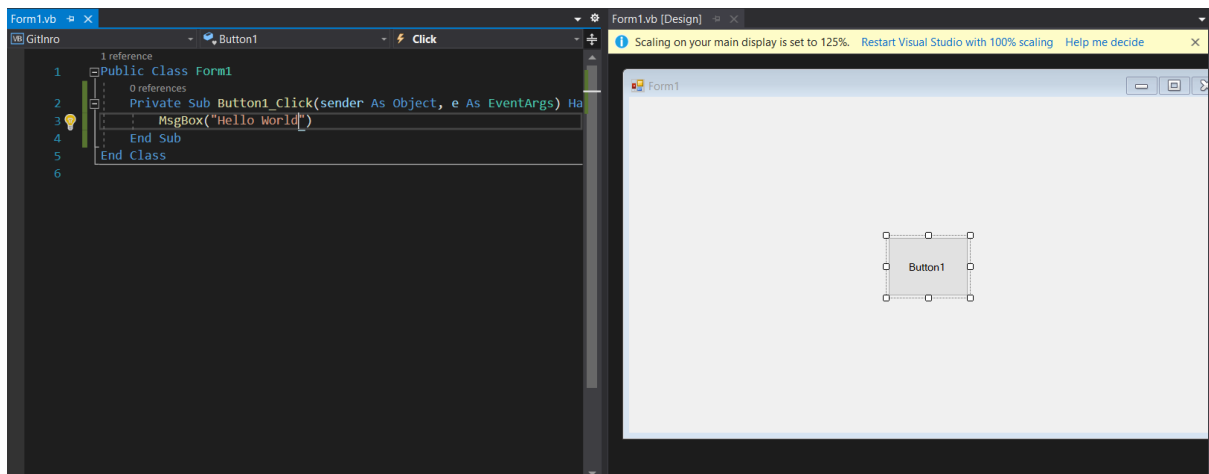
Now press commit

After committing the changes, you won't have any staged changes, because all the changes are committed, look at the

image below

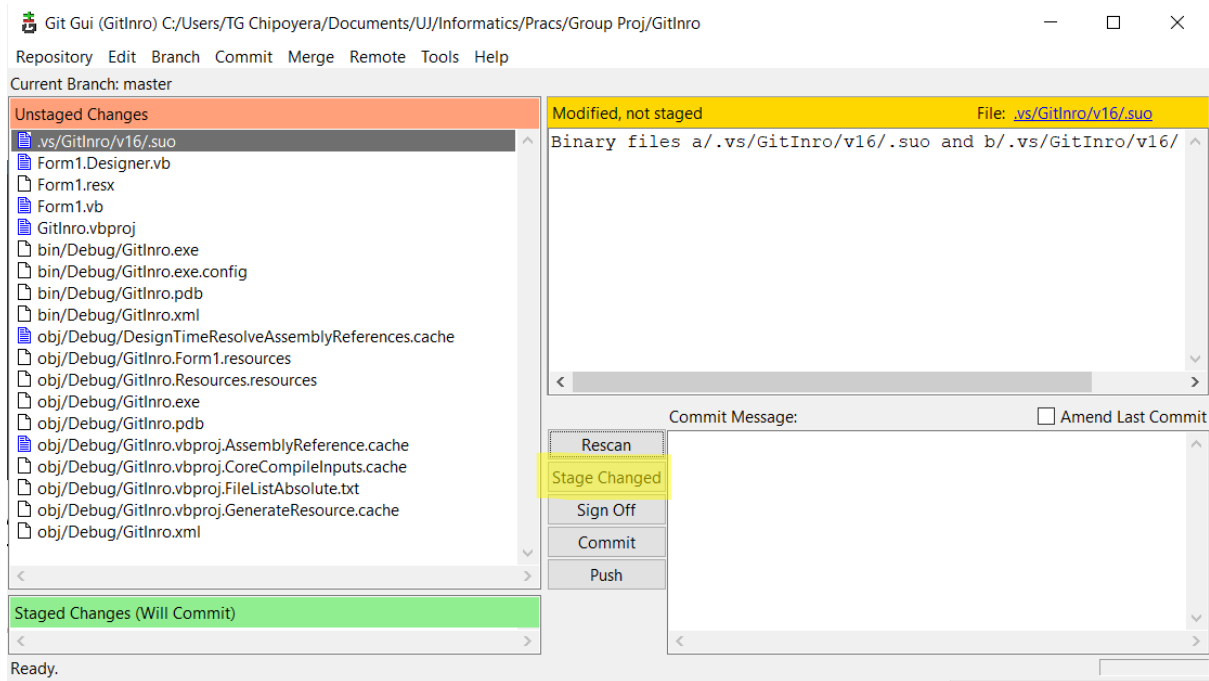


Let's repeat the same process

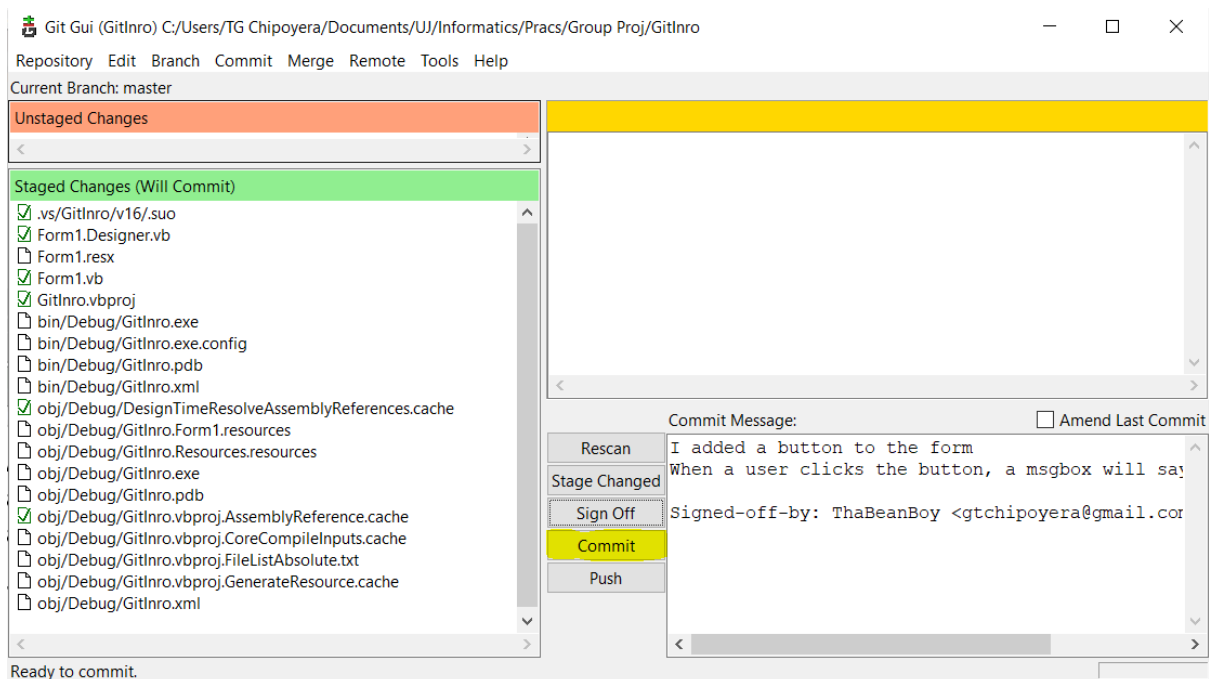


As you can see, I added some code to Forms1.vb and a button to the designer,

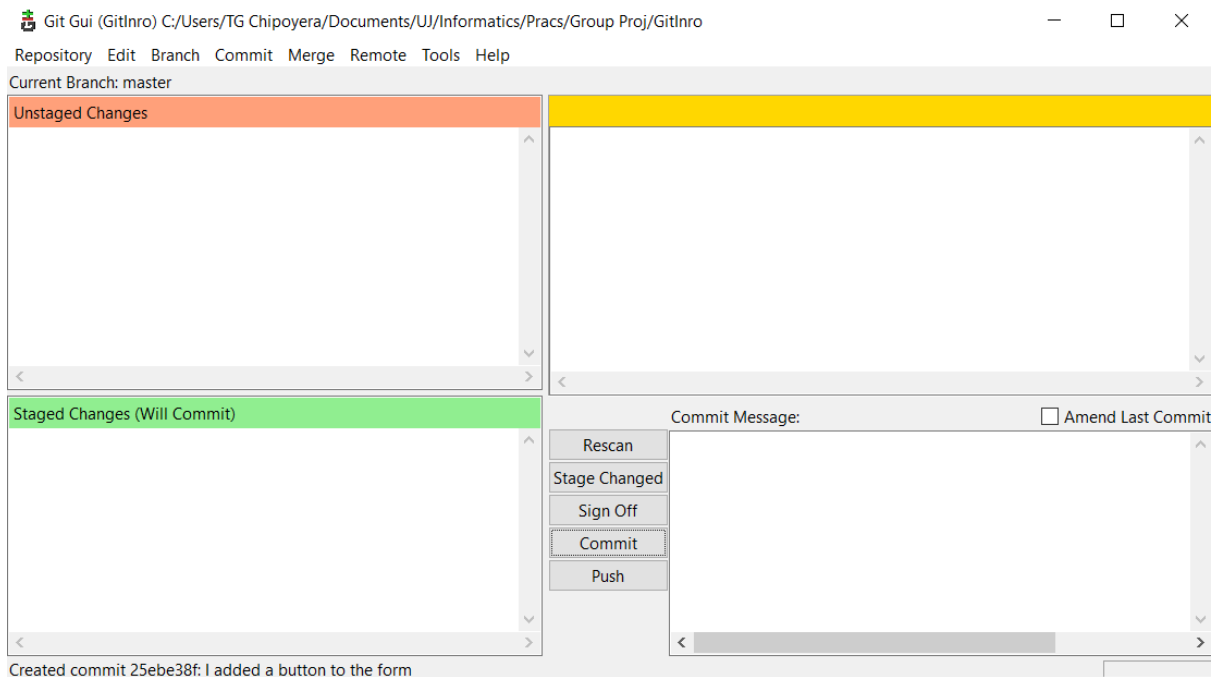
Now let's look at git gui, **Tip, everytime you open git gui, press 'Rescan so you can see new changes'**



As you can see, git will show you the changes you made, Rest assured that the app does make a message box when you click a button,

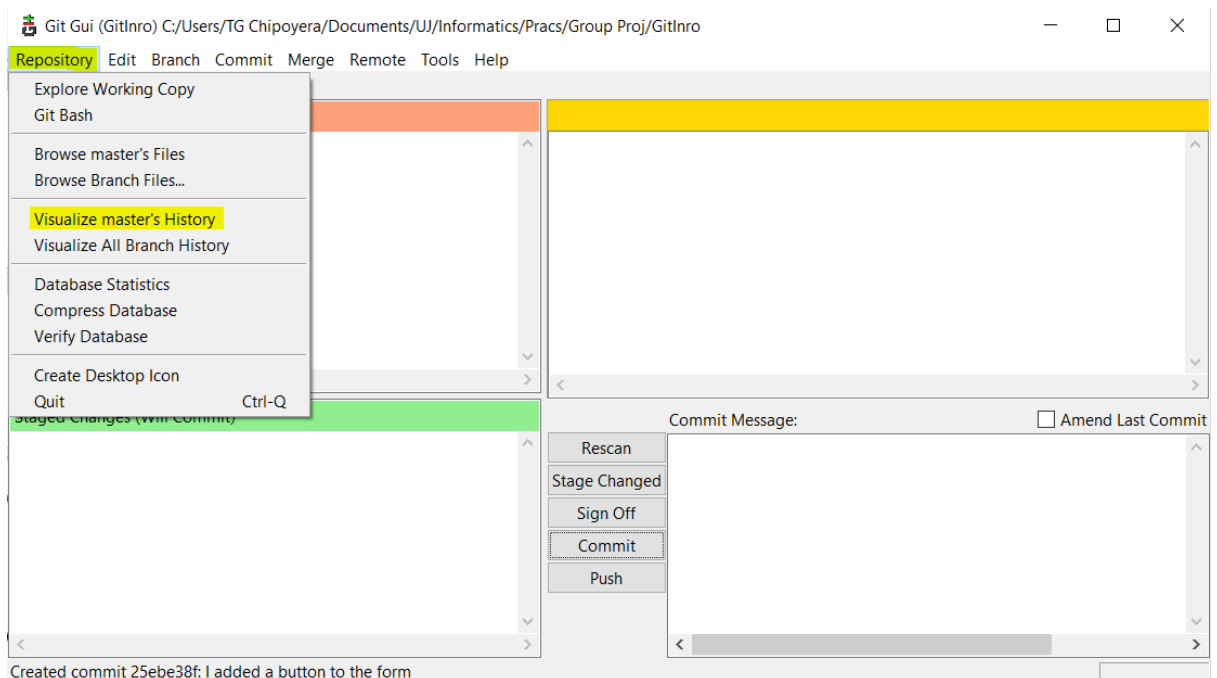


As you can see, the changes have been staged and there's a message in the commit message box, now you can commit the changes

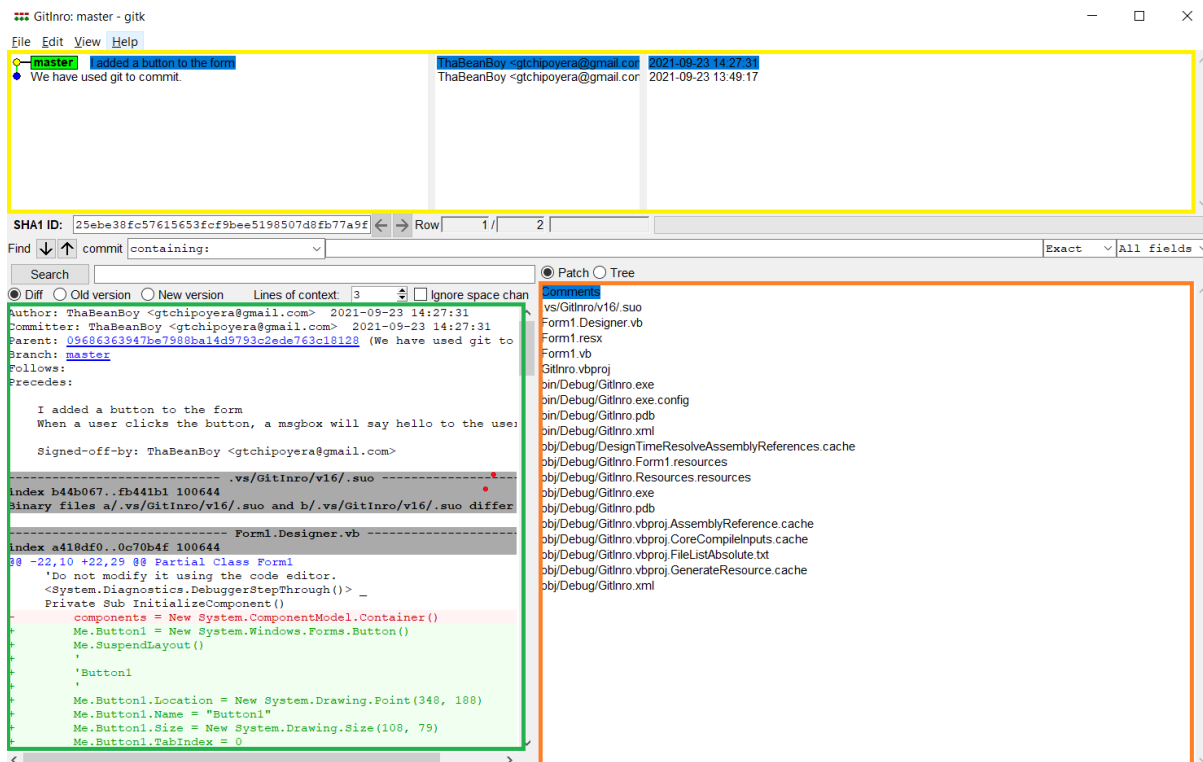


Once again, every change has been committed,

If you want to see the commit history – (A history of all commits) check this out



Click 'Repository' at the top, then click 'Visualize master's History', this window below will open up



The yellow box shows you the commit messages, who made the commit message and date and time of the commit.

The green box shows the changes of each files (the lines of text/code), In Diff mode you will see red and green lines, the red lines show you the lines that were removed and the green lines show you the lines that were added.

The orange box just shows you the files that have been changed

The cool thing is, branches can also can be visualized, it's not only the master branch. Branches is discussed in the next part.



# **Part 3**

# Git Branches

Branches allow us to isolate code, when we discussed how files make up the program, we also discussed debugging and adding new features etc...

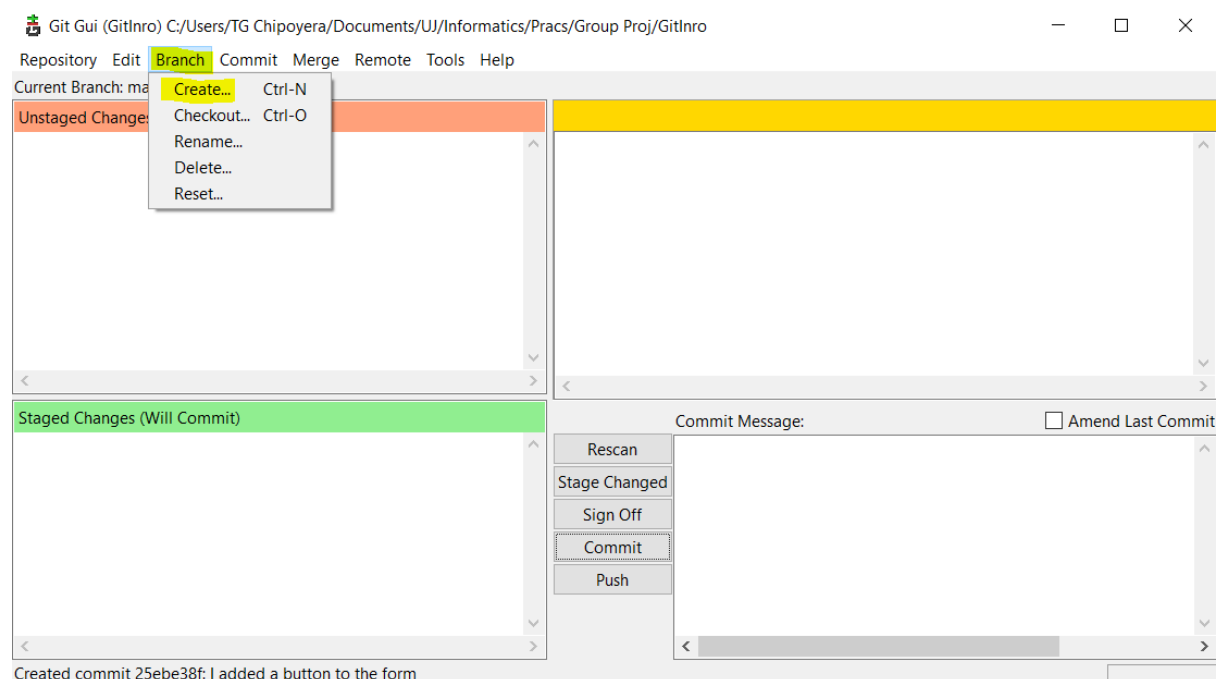
For example, Let's say I want to add a sub routine to the sub routine, but I don't want the master to have the same thing

Generally, we don't work on the master branch, this is because the master branch should only have code that works, merges will be discussed, but rule of thumb, work on a different branch either than master, - If the code in your branch breaks the app, it won't affect the master branch.

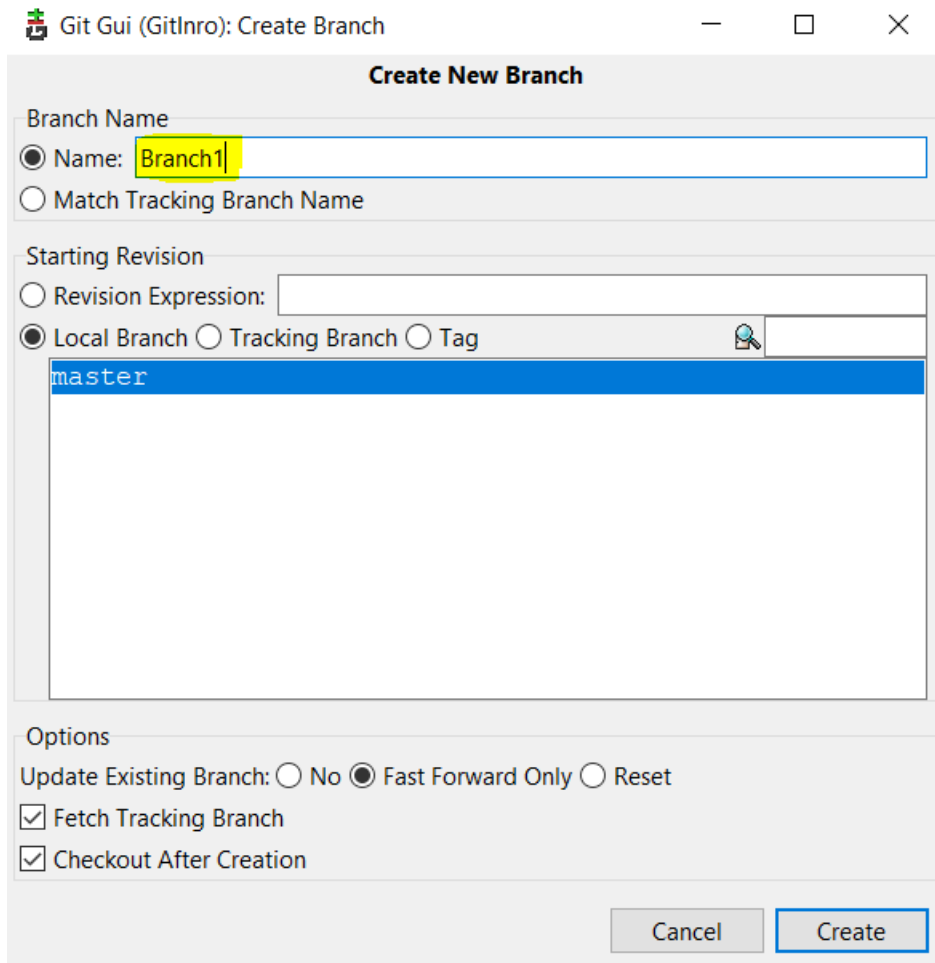
This is an example of branches in action.

First we make the branch:

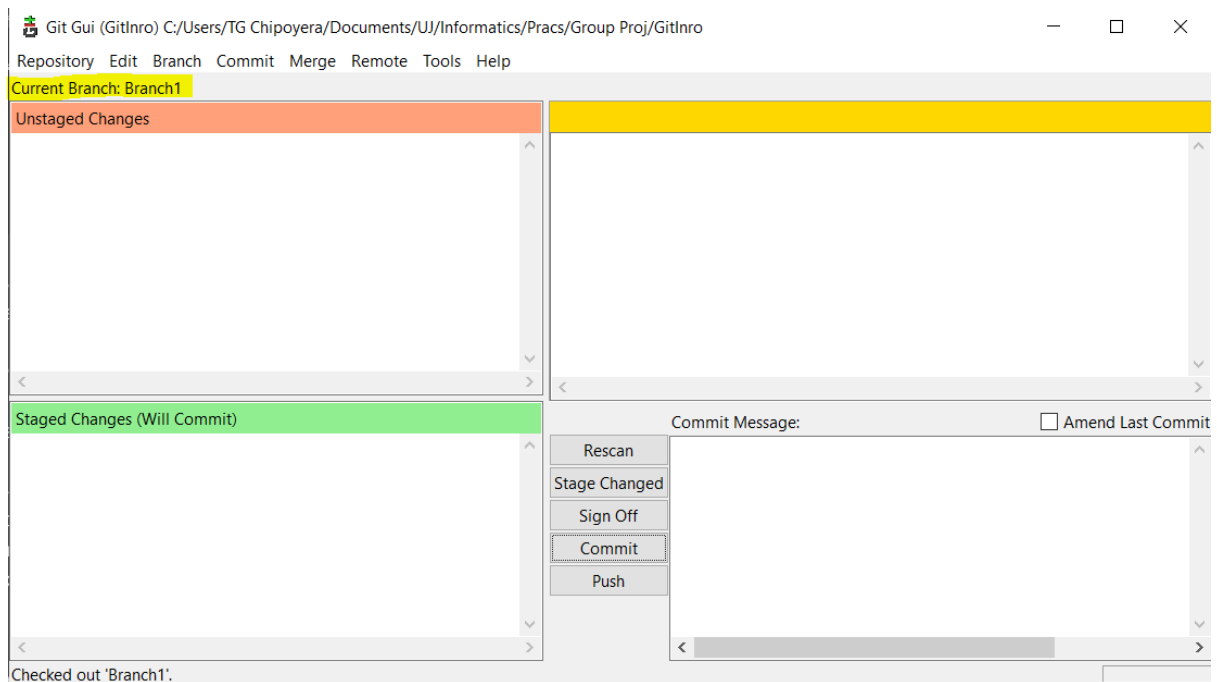
Click branch then create



You can name the branch anything you want



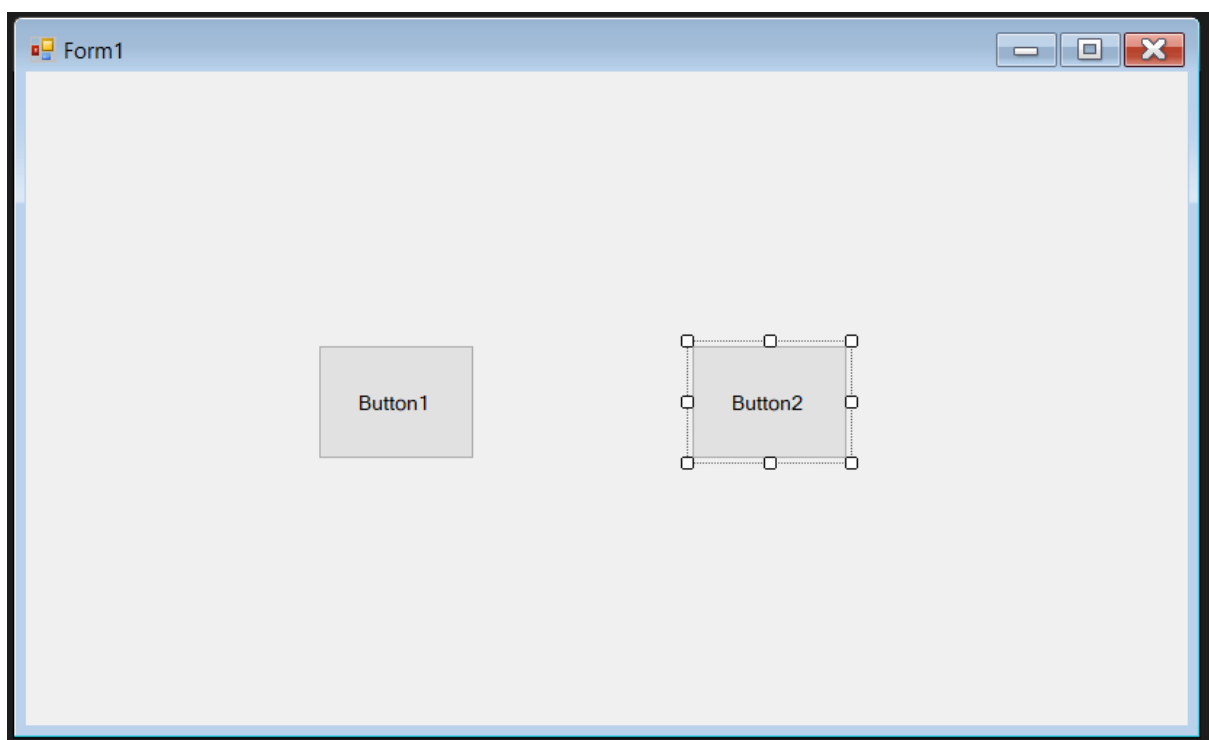
At the top (In the image below), you'll then see that you are in the Branch1 branch (Look at the yellow highlight)



Now changes have been introduced in the Branch1 branch

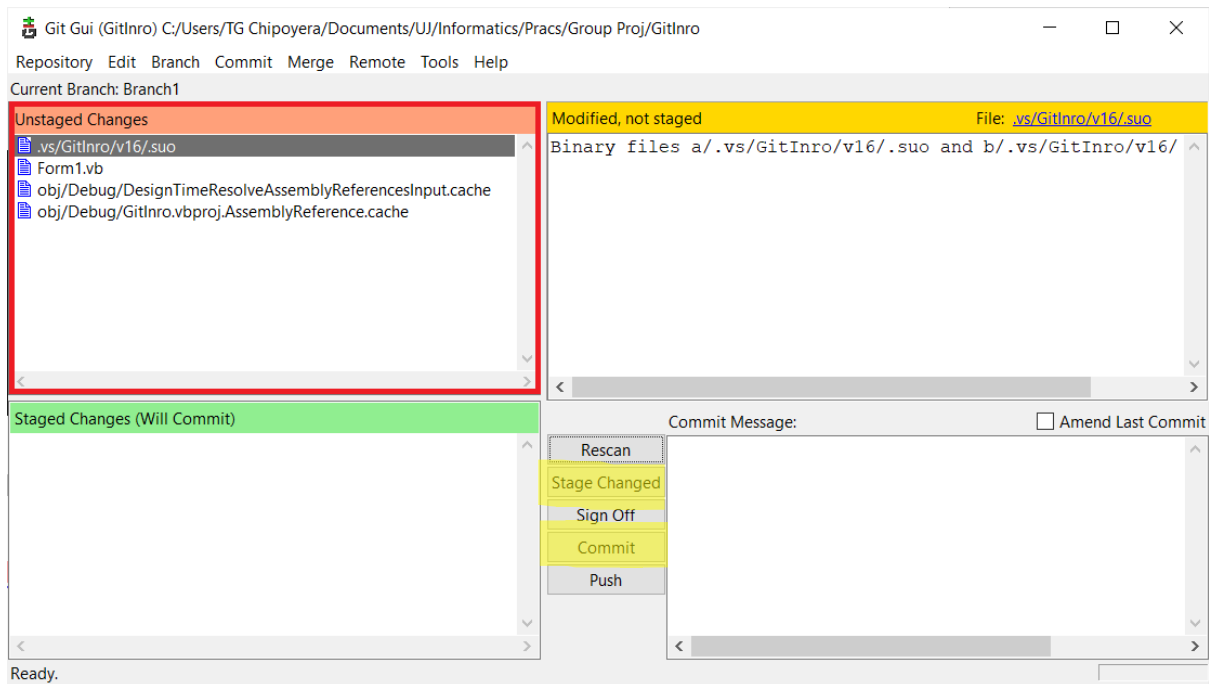
The form has 2 buttons instead of 1 now

```
1  Public Class Form1
2      Private Sub Message(Txt As String)
3          MsgBox(Txt)
4      End Sub
5
6      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
7          MsgBox("Hello World")
8          Message("I'm a coder")
9      End Sub
10 End Class
11
```



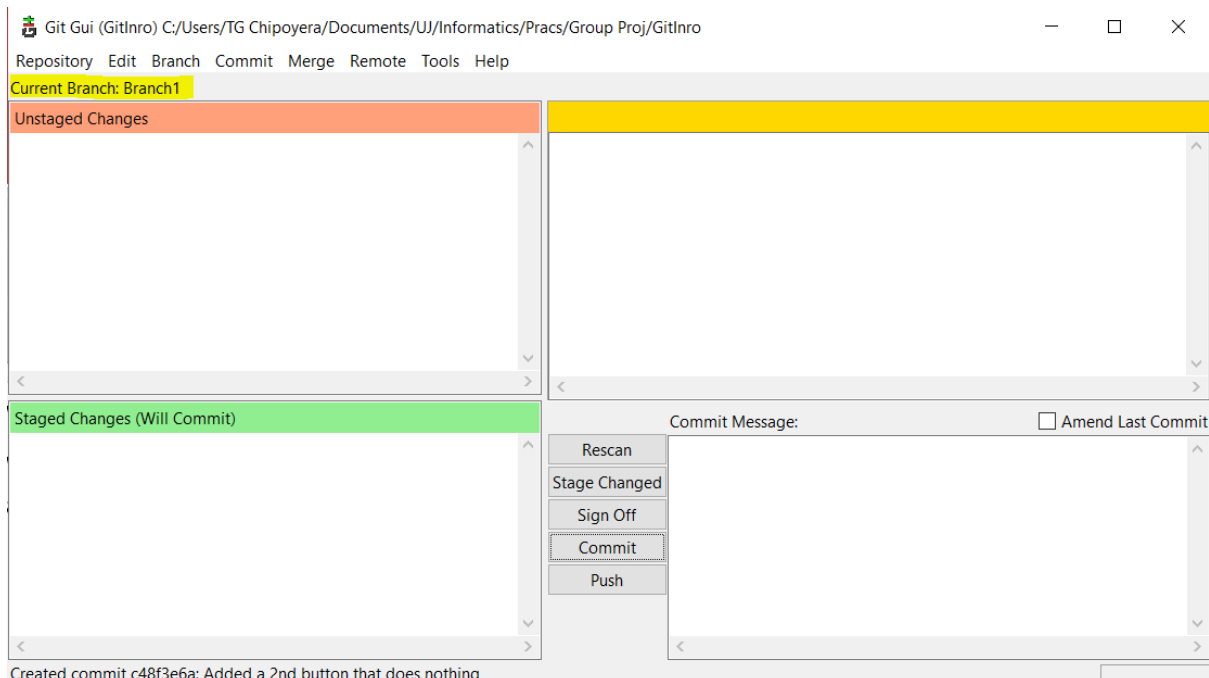
Remember that we are still in the Branch1 branch, the green lines on the left sideshow you what changed, if the color is yellow, you haven't saved the file...

Now let's take a look at the unstaged files. **Remember you have to rescan to see the latest changes.** Once again, we have unstaged.

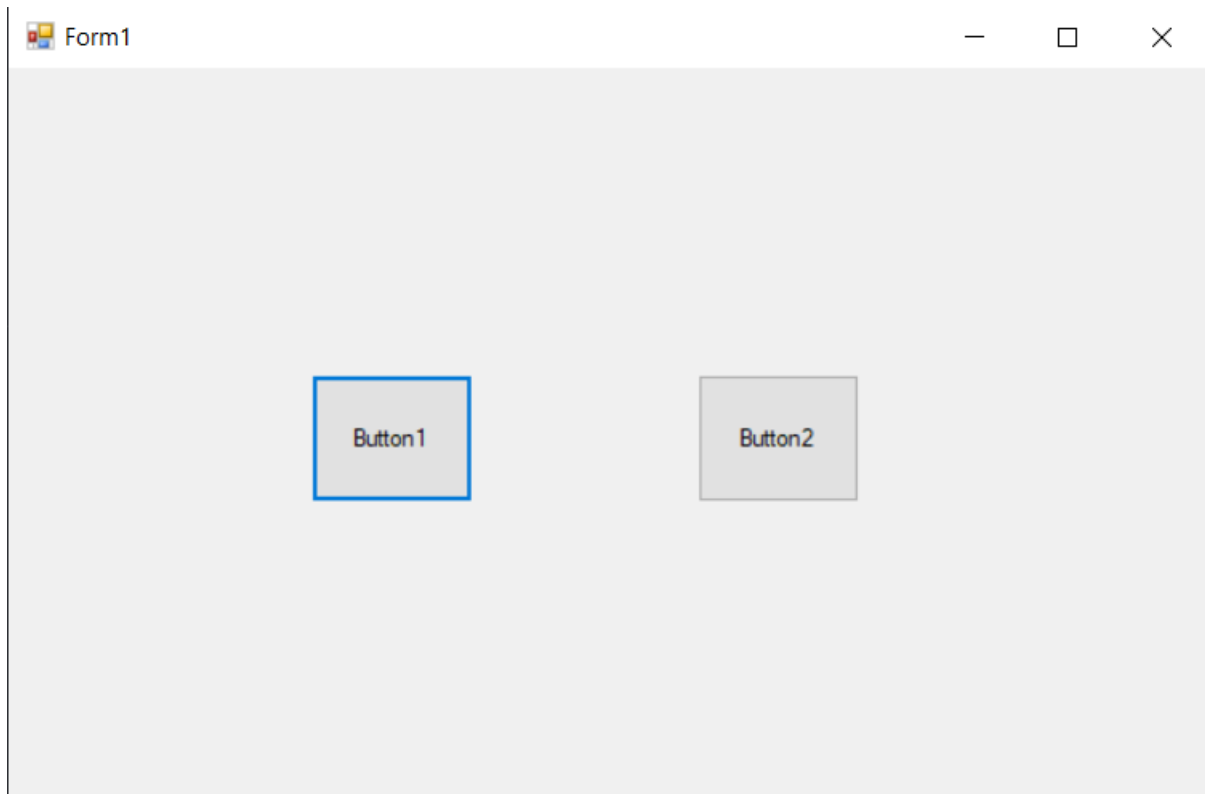


When we are sure that we want to commit,

We once again press staged changed, write a commit message and then commit. You'll see that there are no staged nor unstaged changes like before

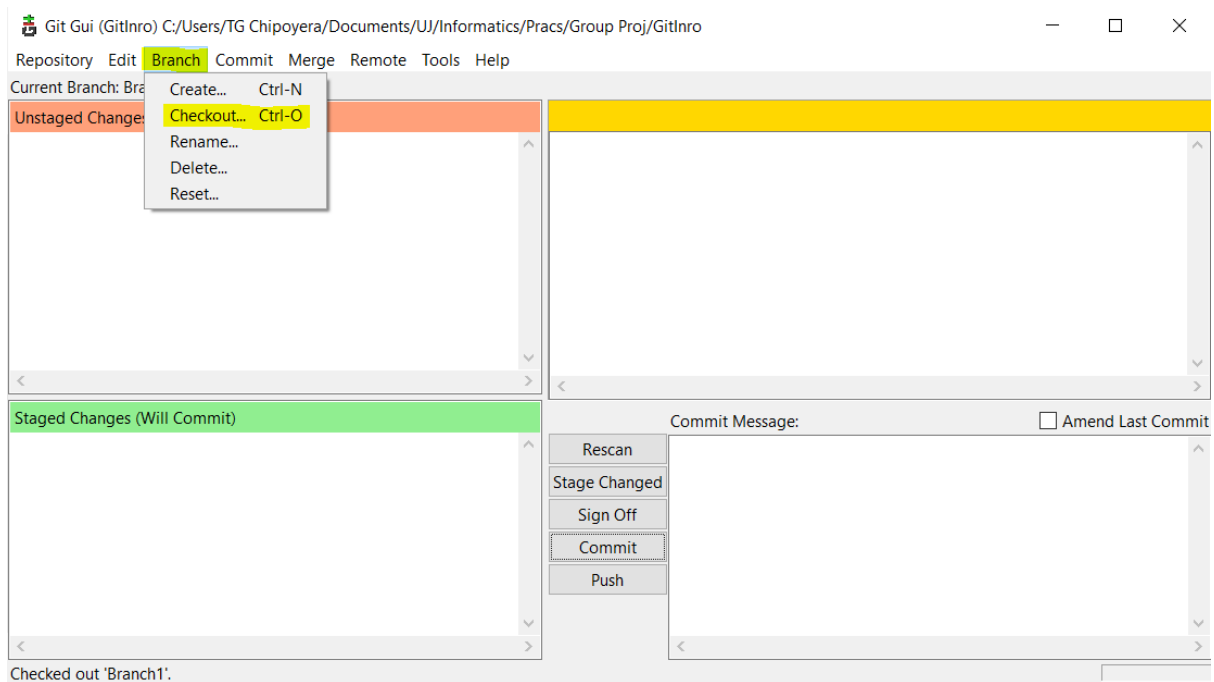


When I press start in Visual Studio, the result will be the image below, REMEMBER We are on the Branch1 branch, this is the result



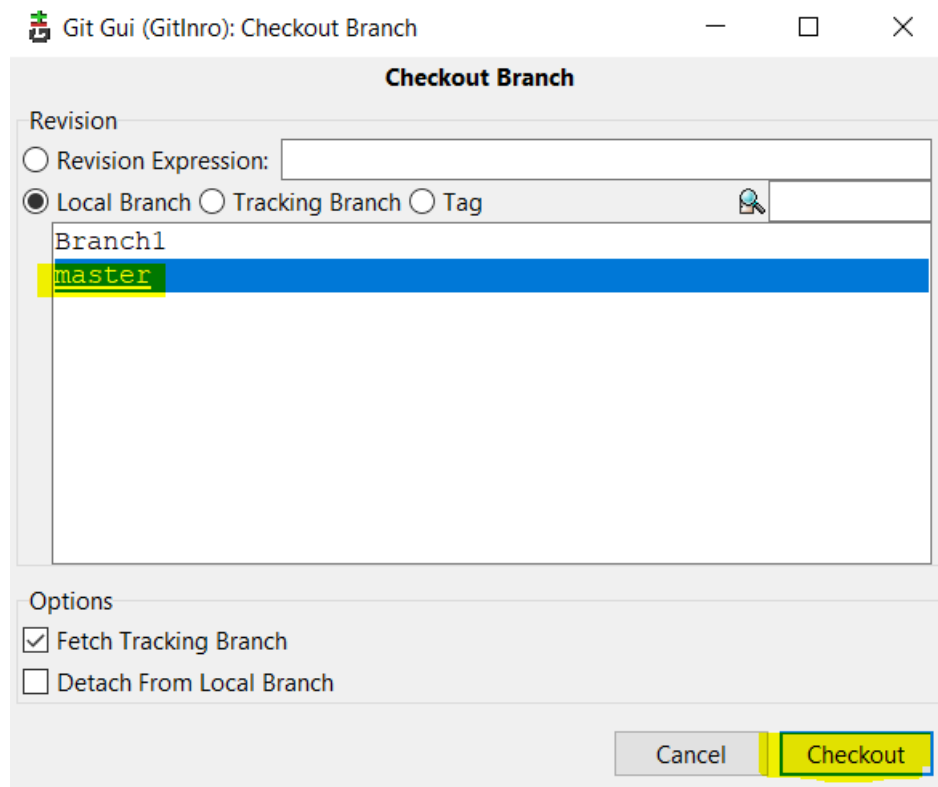
But let's go see the result of the master branch

To go see the master branch, click branch at the top, then click checkout

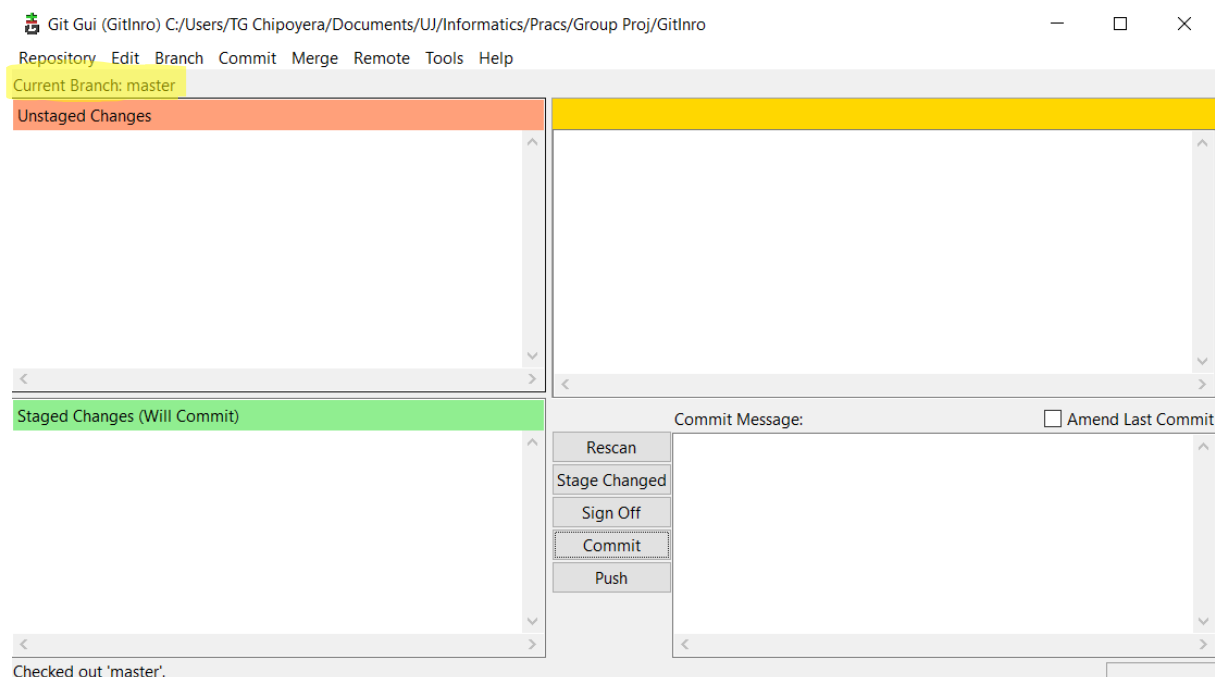


This window will then be opened, click the branch you want to go to. **NB: When trying to see other branches, otherwise known**

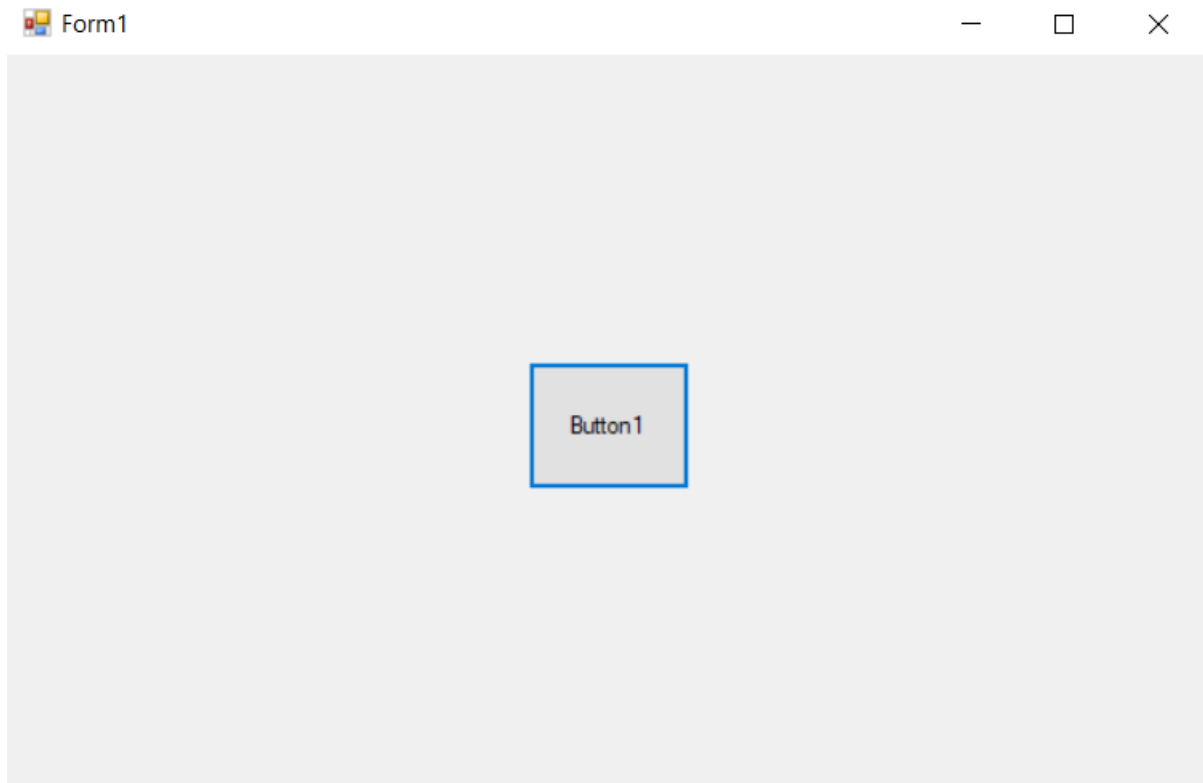
as 'checking out a branch', you are not allowed to see other branches changes are committed. Hence If an error pops up saying you should commit blah blah blah, go back and rescan, if you see unstaged changes, stage and commit them



Now you'll see that you are in the master branch

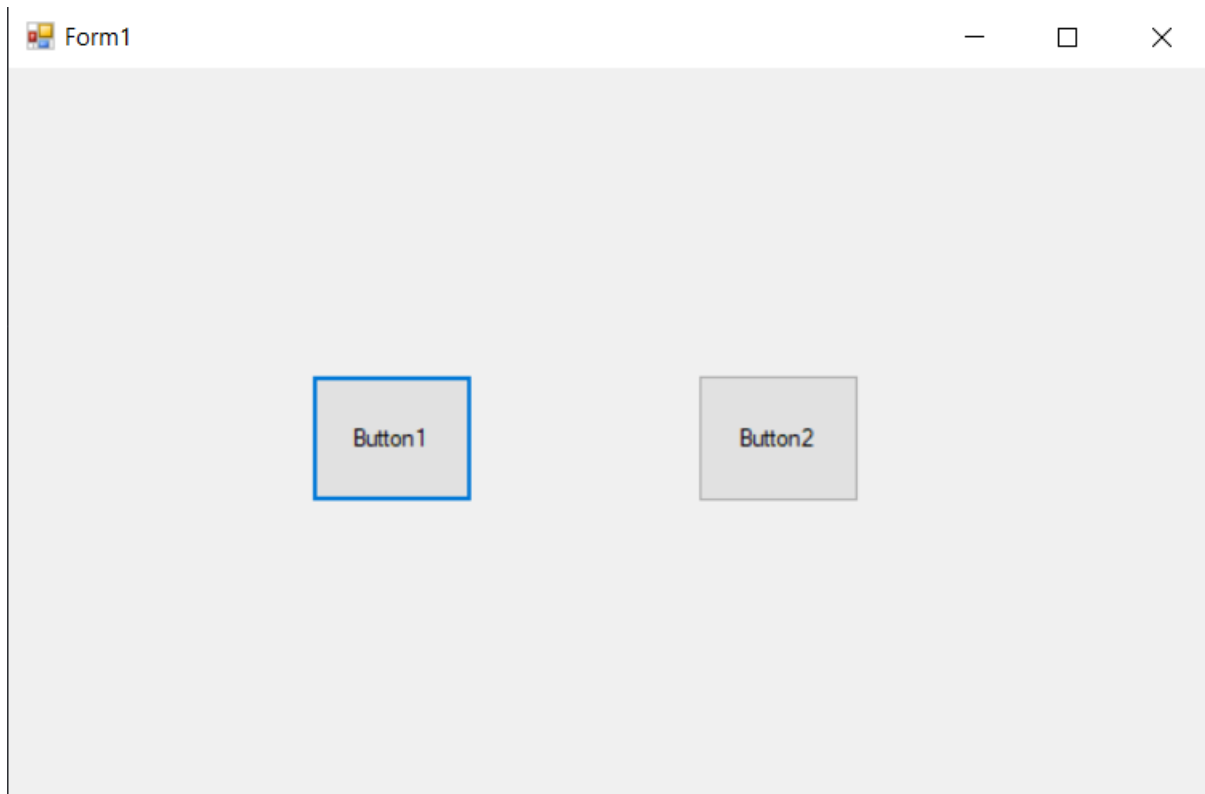


Back in Visual Studio, I pressed start again, and guess what the result is



But, bro when I pressed start while I was in the branch1 Branch, the result was





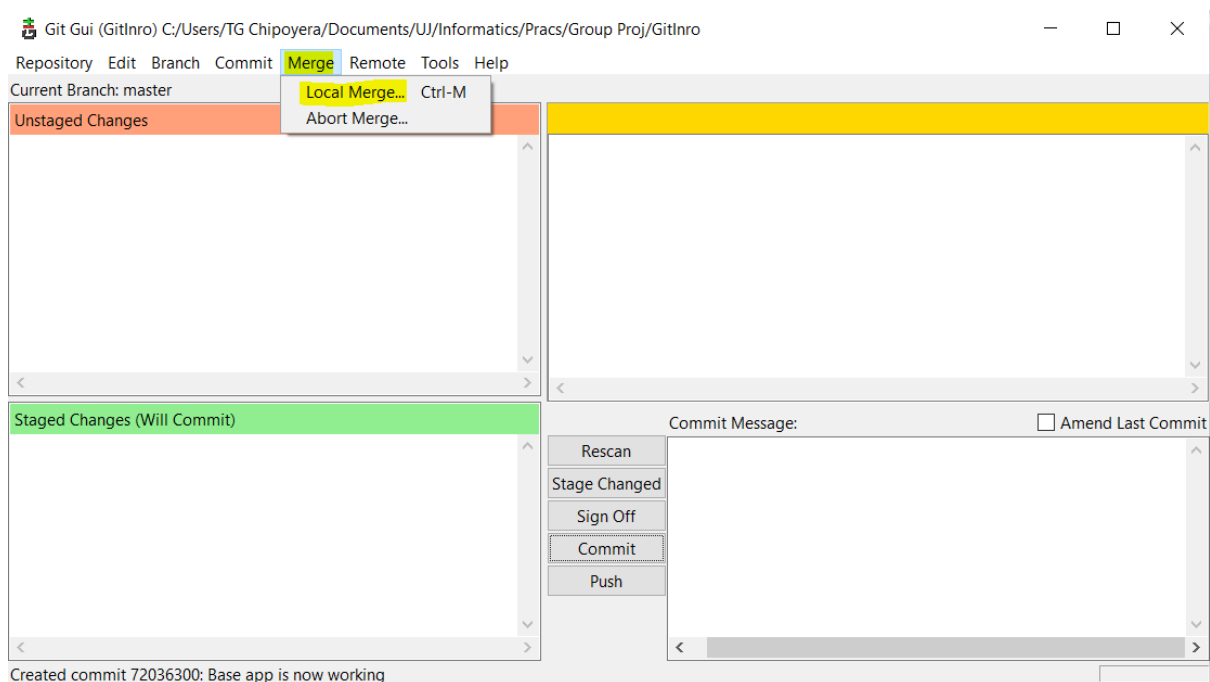
This is because, every change you make 1 branch, is not transferred to the other, hence you get to play around and edit files

Now that you have seen branches and how you can make changes in an isolated environment, code that does not the overall app, you can basically see If Member 1 makes changes to files, and Member 2 makes his own changes, It won't result in weird commits where we have confusion.

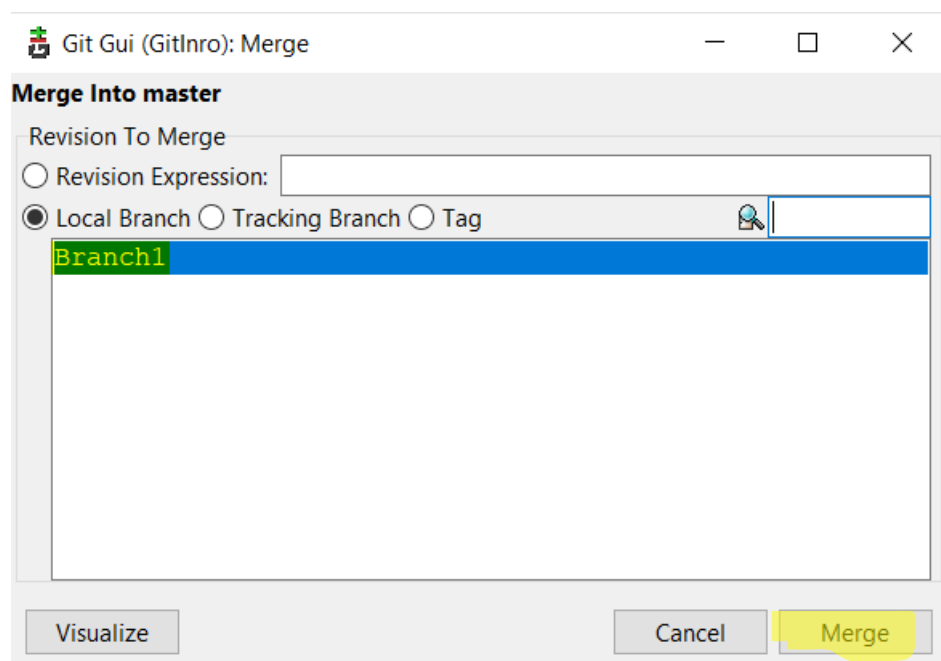
Now we should remember that we may have added, debugged or something else, we might want to carry the things we did in 1 branch to another branch. This is called MERGING.

For instance, lets say we were making a programme, I made a branch called Coffees, if I'm satisfied with the code I wrote in the Coffees branch, I might want to use the Coffees code in the Café branch,

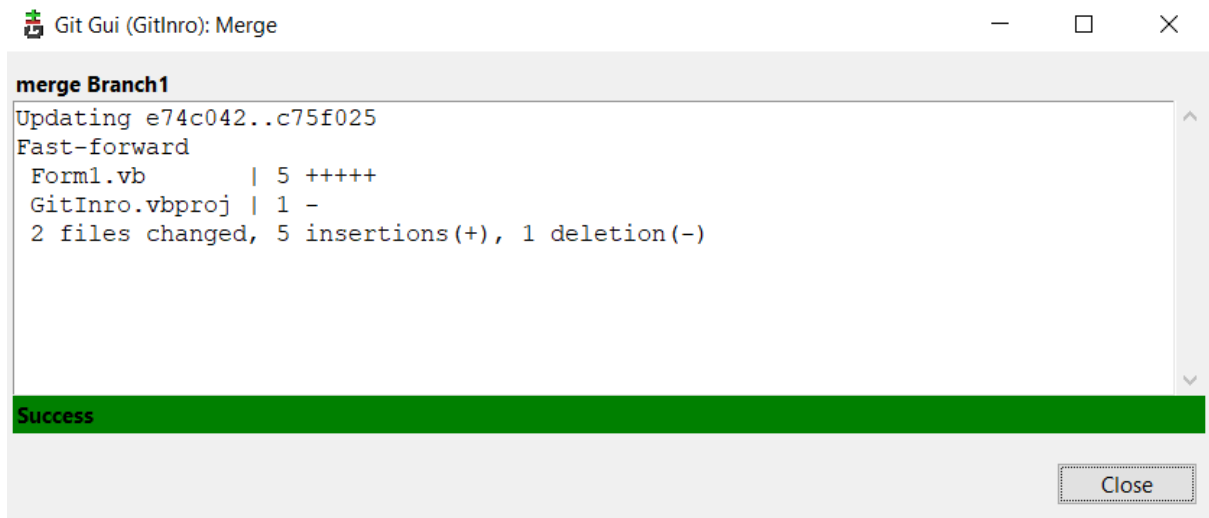
In this example, I'll merge the code I wrote in the Branch1 branch and try implementing it in the master branch, check this out...



Now click the branch that has the code you want

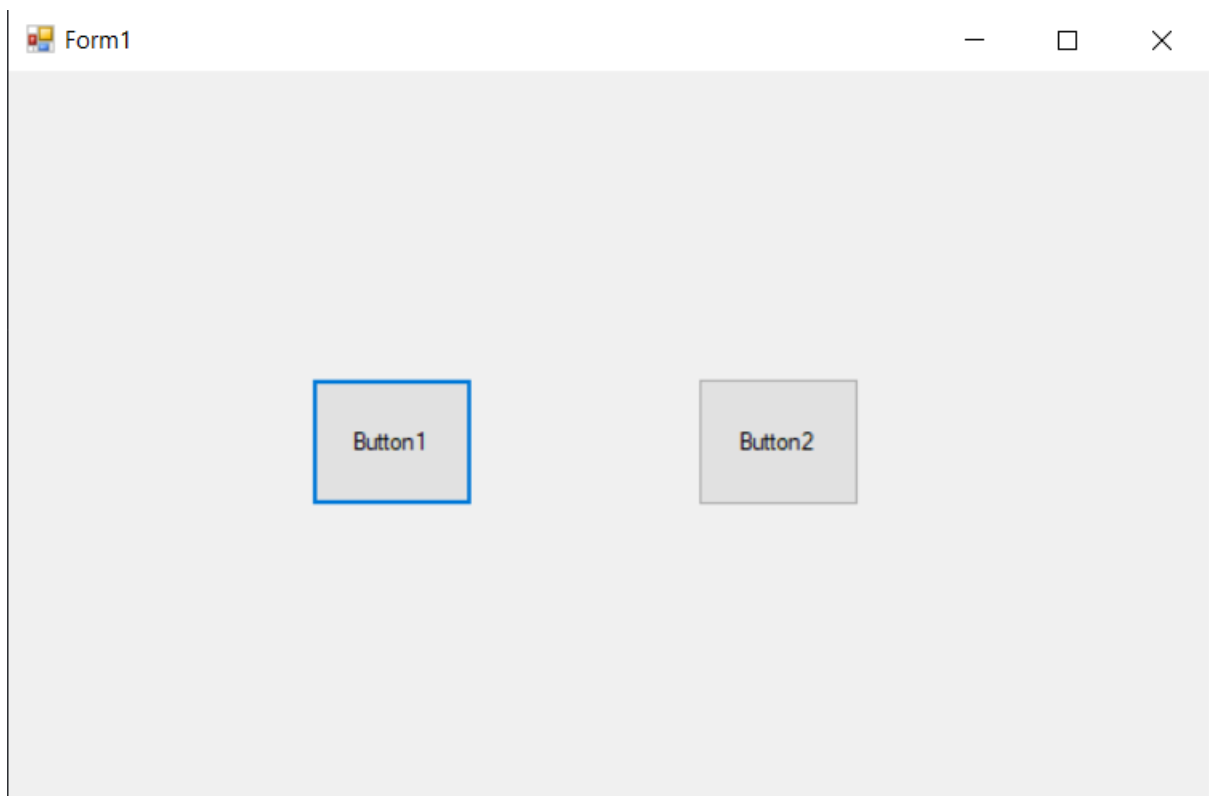


You should get a popup like



Now let's checkout the master branch and run the code in the master code

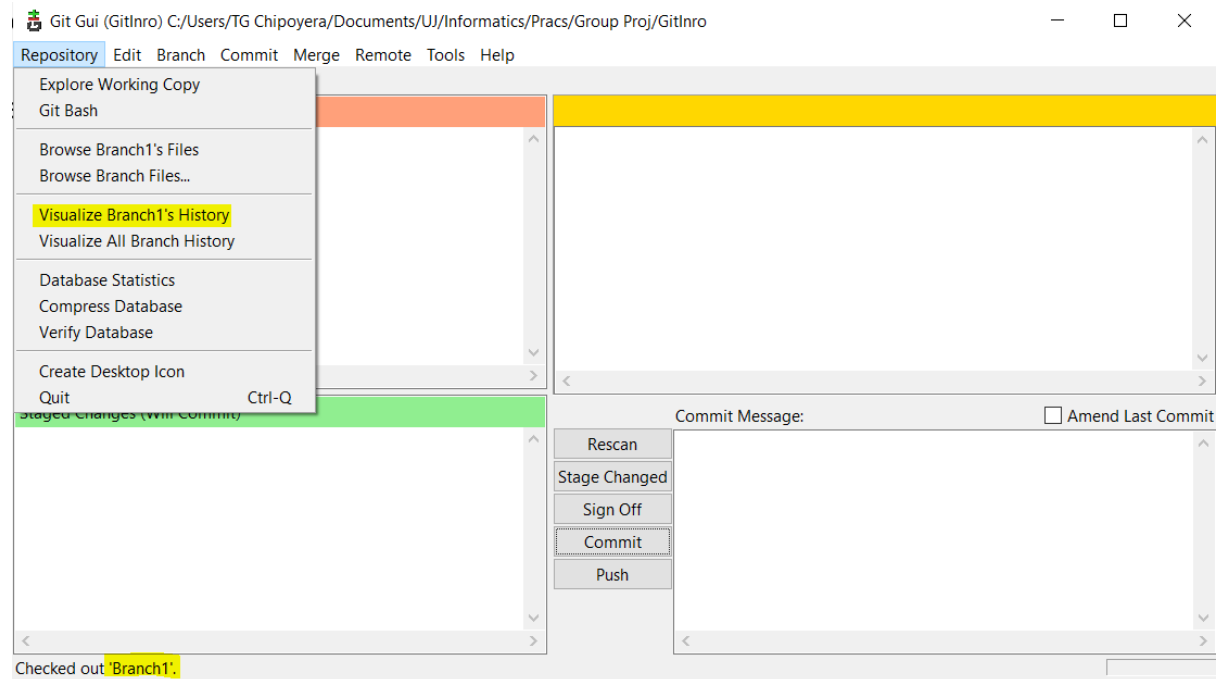
You will get



Now everything we did in the Branch1 branch, can be used in the master branch

Quick note: Remember how we visualized the master branch, (Seeing all previous commits that were made by the team and the commit messages), Other branches can do that aswell

Check this out



Now if you want, you can click visualize all branch history to see everything.

Finishing notes:

Git is a very widely used technology, because of the things we discussed and a few other things. It's good that you learnt git now, because once we are out in our future careers, developing apps in teams and all, git will always be there.

Git isn't the only version control system, but it is the most popular...

In the next document, We will rediscuss git clone, discuss git pull and git push...

Homies, rest easy now