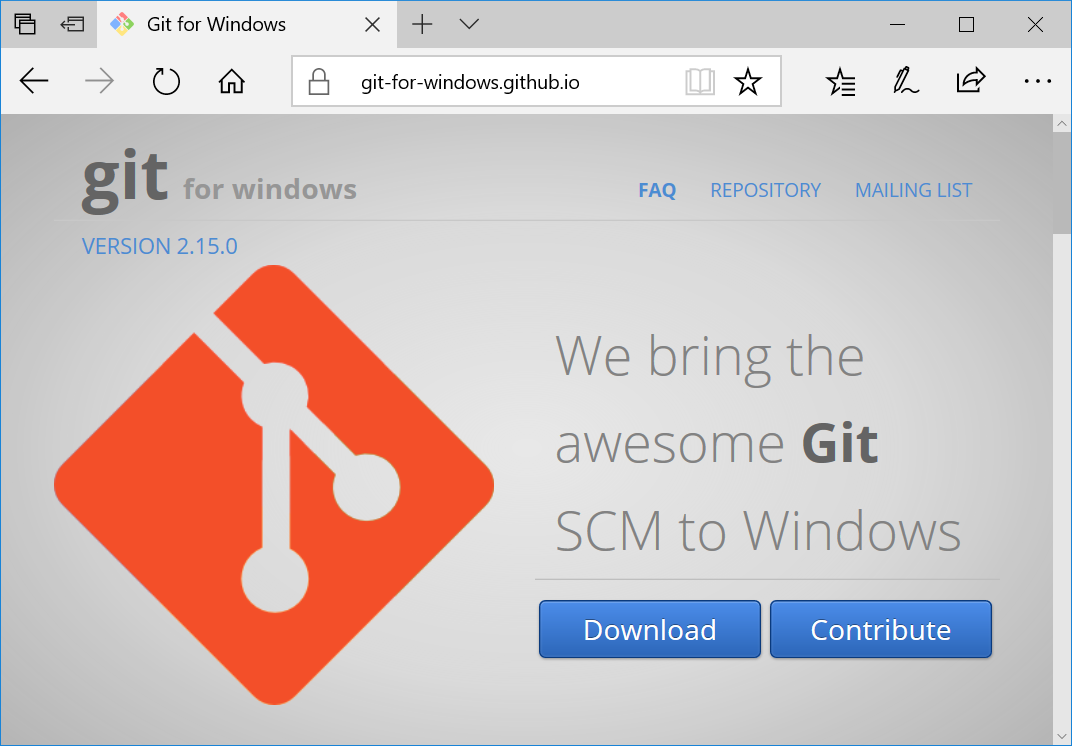
**Getting started with git gui on Windows**

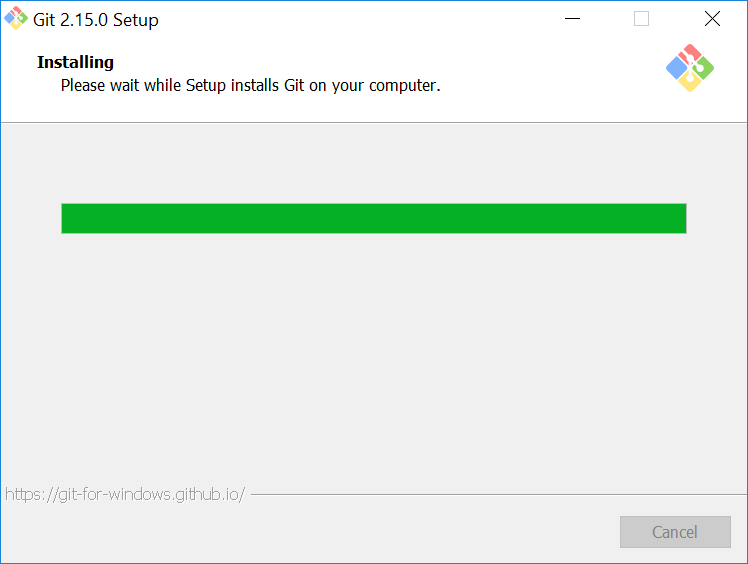
**Install git for Windows**

* Get a copy of the Git .exe installer:

Download git execute file from <https://git-for-windows.github.io/>



* Double click to install git on your machine

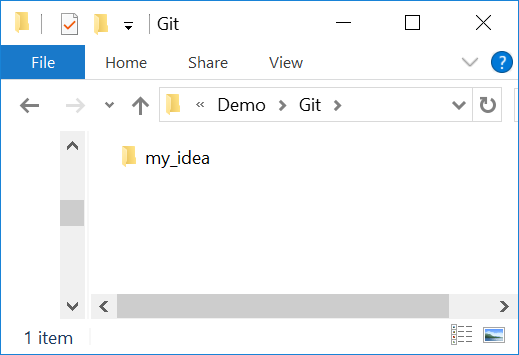


* Accept all the defaults

**A pretend project**

For simplicity, let us make a folder for our project in the C:\Demo\Git\.

* Create a new folder named my\_idea
* Create a new file named good\_idea.txt
* Add some text like “This is my good idea”

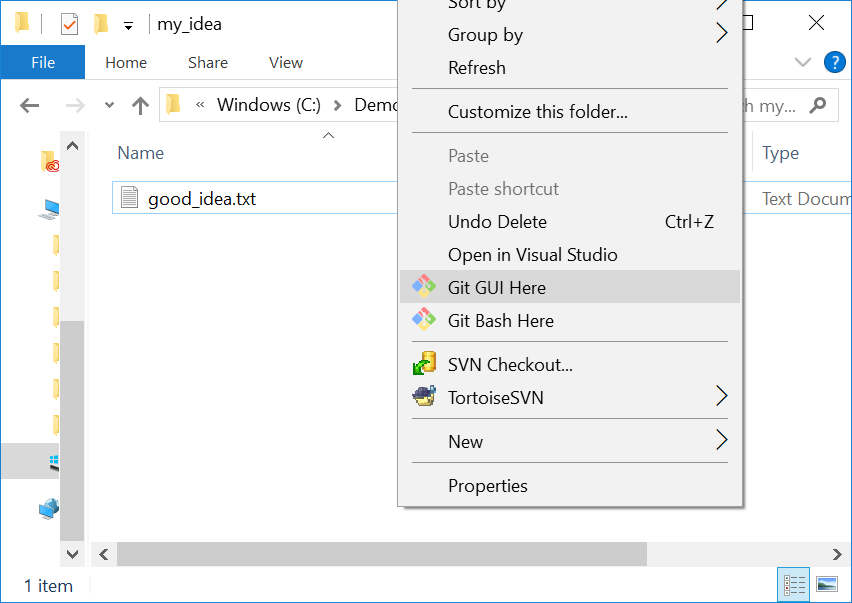


**Putting the project into version control**

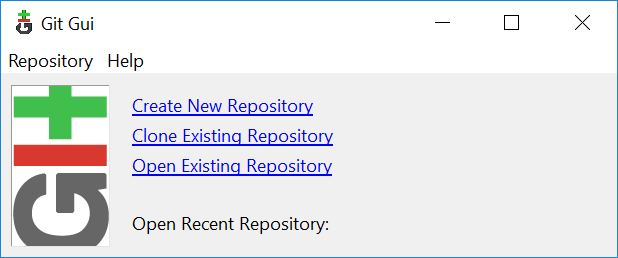
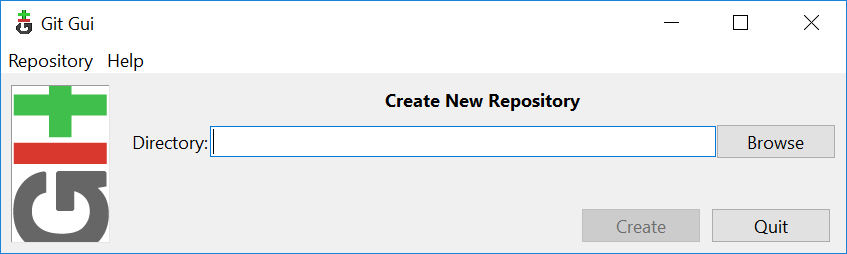
Now we have a very small project, just a folder with a single file in it. We’ll now put this folder into version control.

**Initializing the repository**

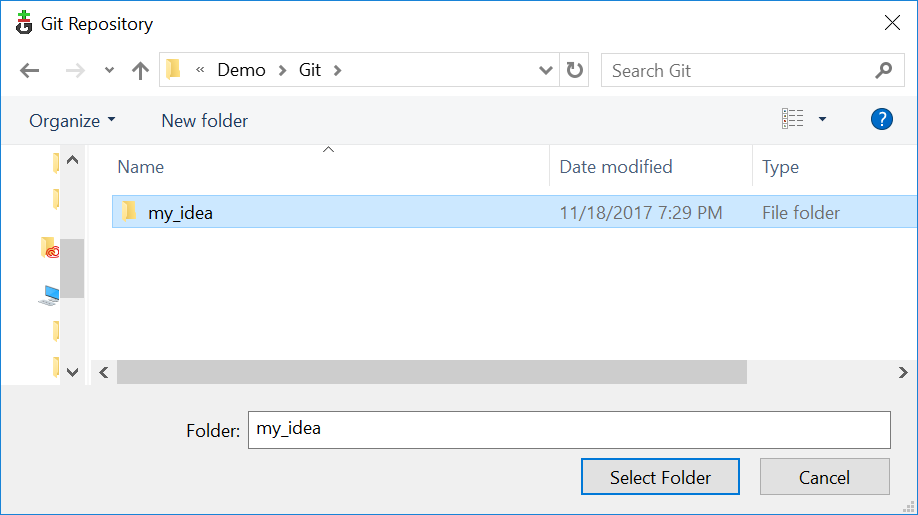
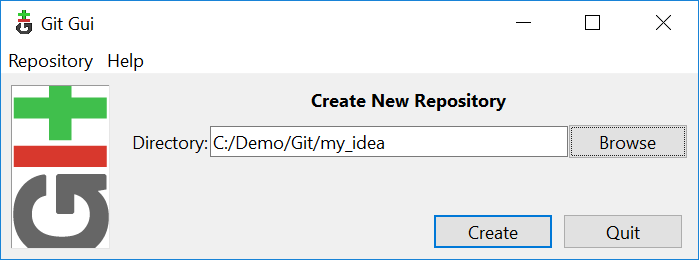
* Right click in the my\_idea folder, choose “Git GUI Here”



Click “Create New Repository”

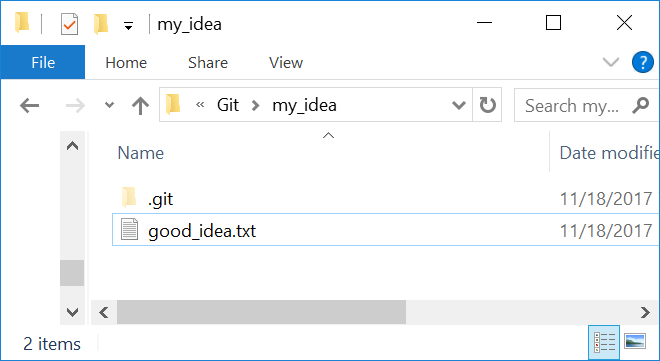
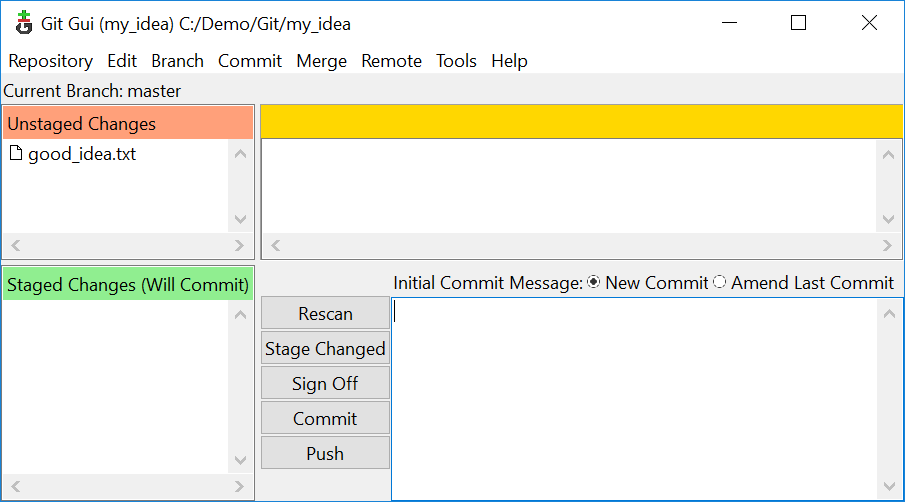
 

Click “Browse” and select my\_idea folder

Click “Create”. You my notice that there is now a hidden folder called .git in the my\_idea folder.

* You will get a dialog like this:

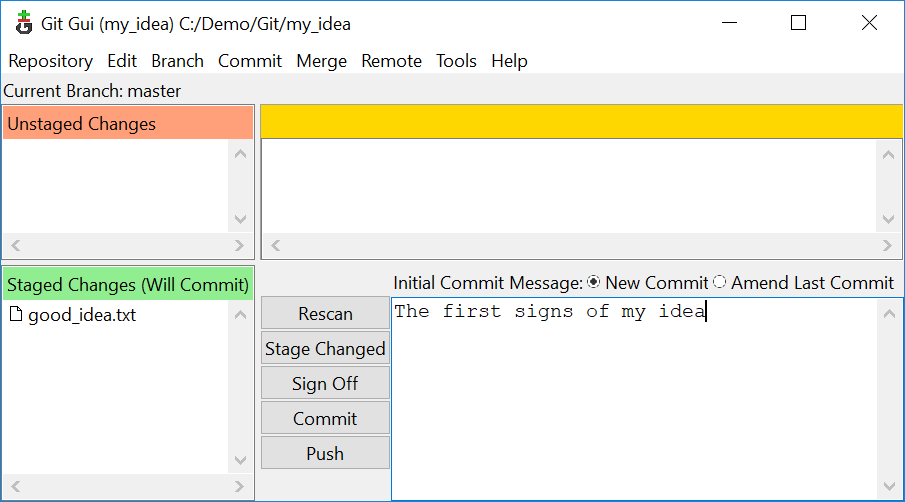
 

**Adding stuff to the repository**

Now we have a git database in our folder, we need to add our file.

Notice that our file is at the top left, and is “Unstaged”. That means that, at the moment, it is not going to be part of any “Commit” we do. A commit is where we record changes into the database. We do want to record our file, so

* Either click on the little icon to the left of the good\_idea.txt filename in the dialog, or go to the “Commit” menu, “Stage to commit” option. Notice the goog\_idea.txt file then goes to the “Staged changes(Will Commit)” section of the dialog.



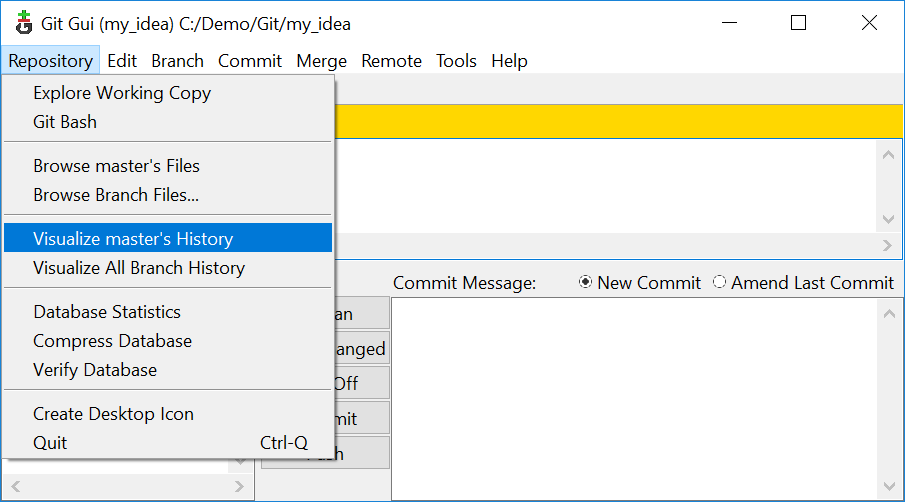
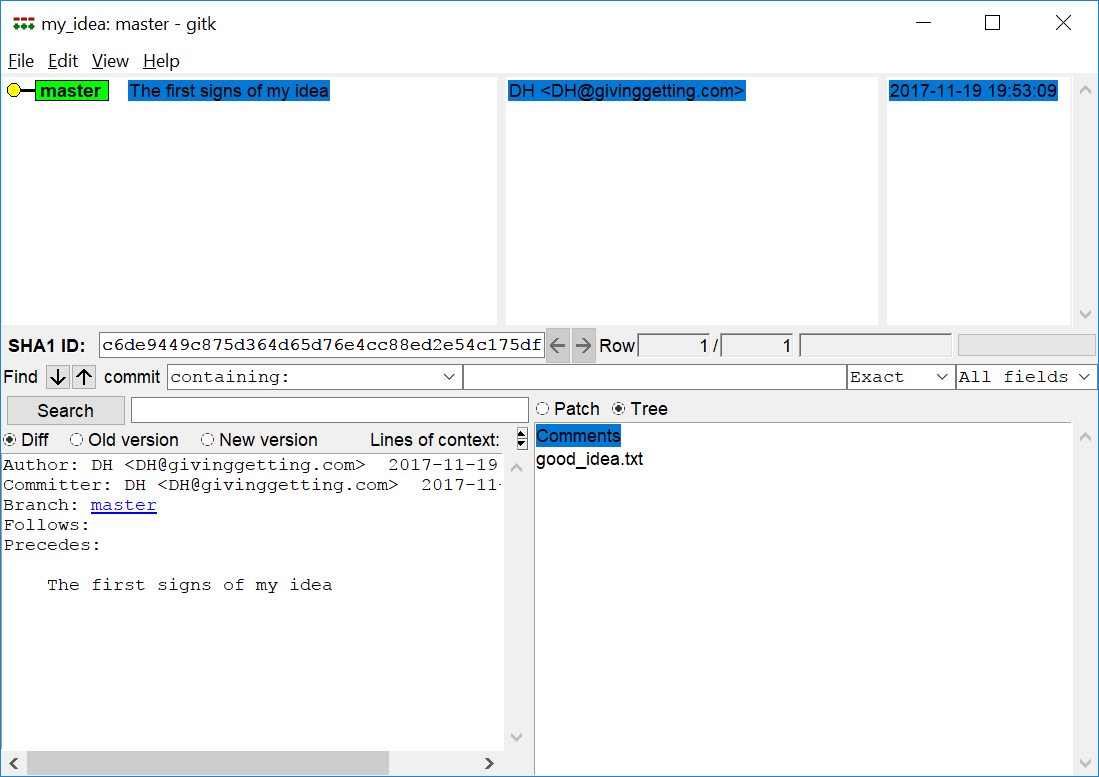
Add a message in the lower right box, like “The first signs of my idea.”

The commit message is a reminder what set of changes the commit has. It’s very useful to put meaningful messages here for quick reminders of what you intended with the changes. Now click “Commit”. Congratulations!

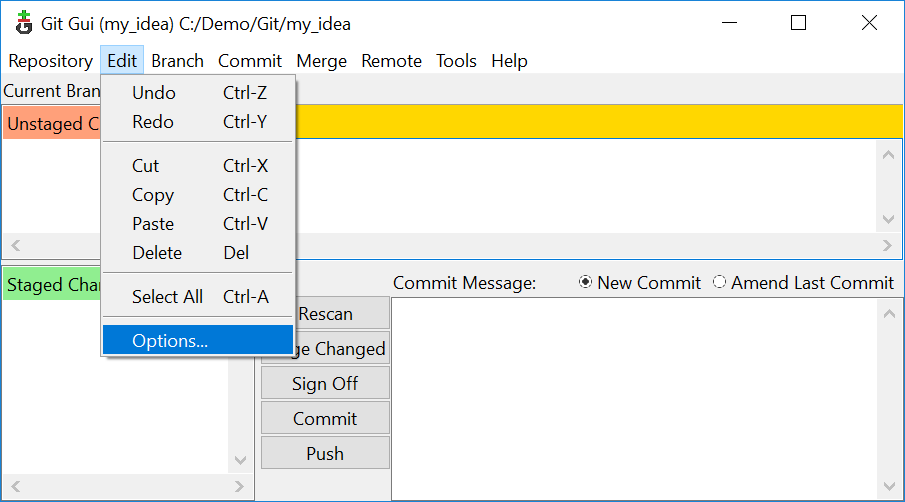
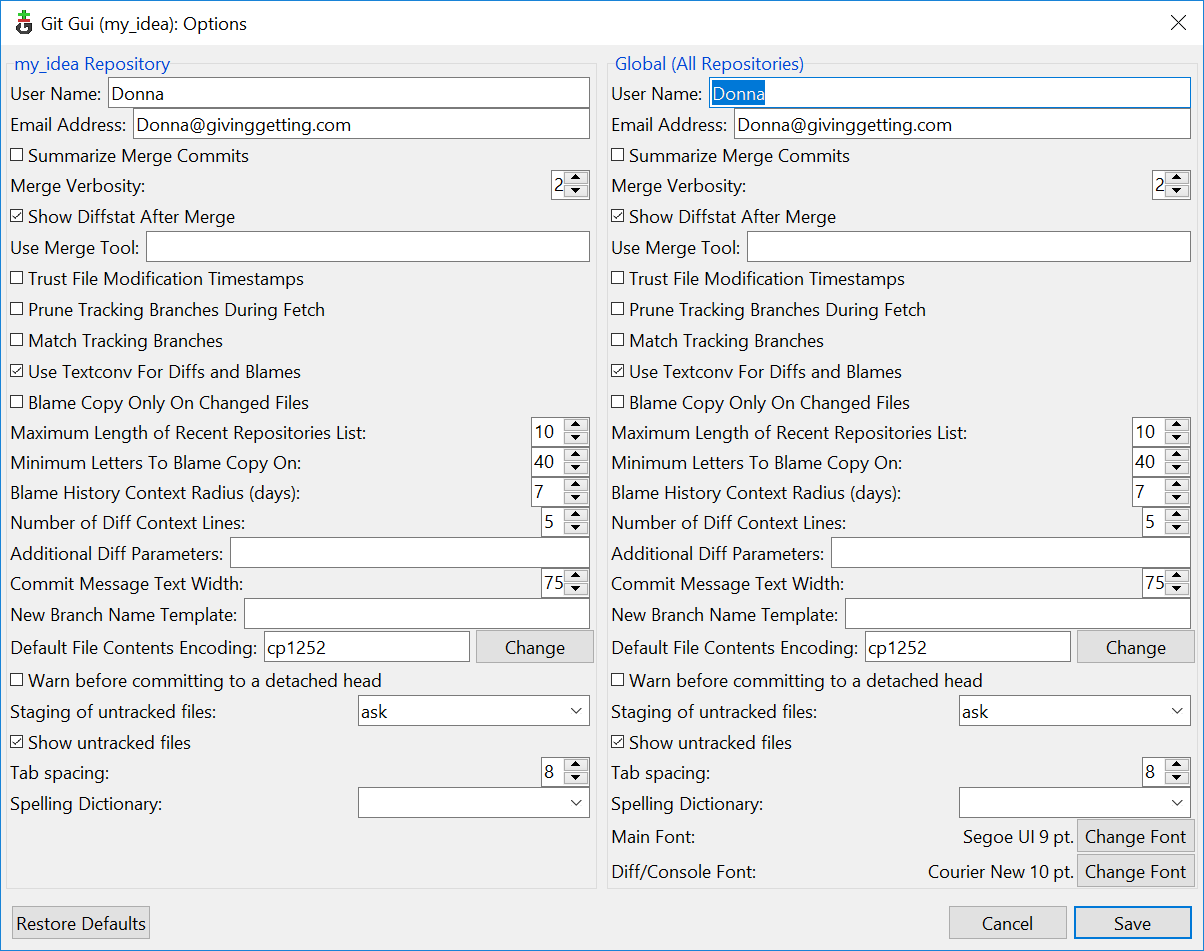
**Speak with your own voice**

Each commit has someone who wrote the contents of the commit – the “author”. So you can get all the blame, and all the credit, you can identify yourself to git. That way git can see who’s doing the work (or doing the damage).

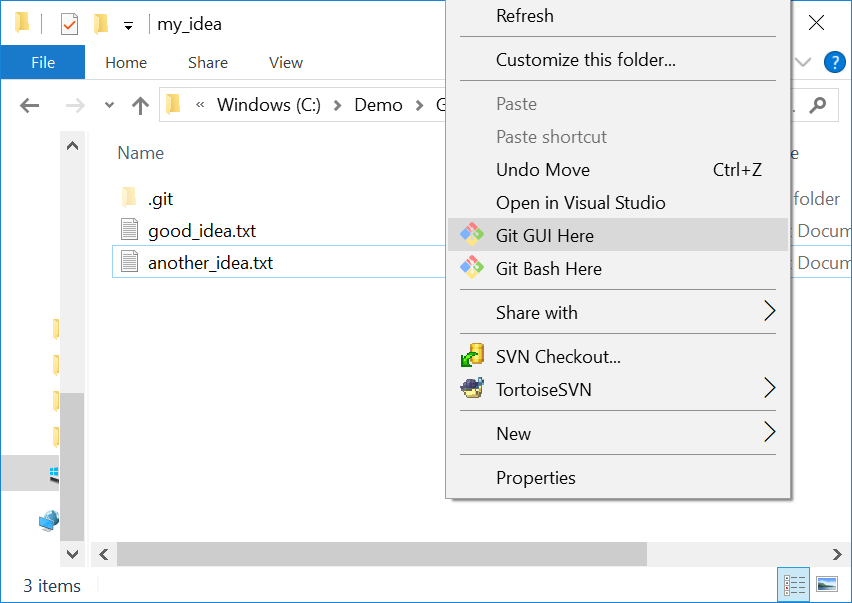
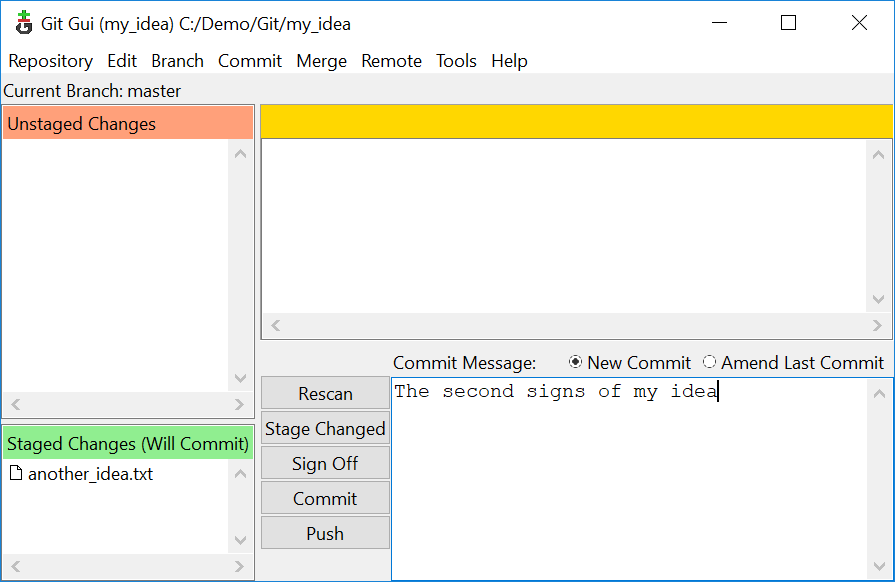
* Click on the “Repository” menu and select “Visualize master’s history”:

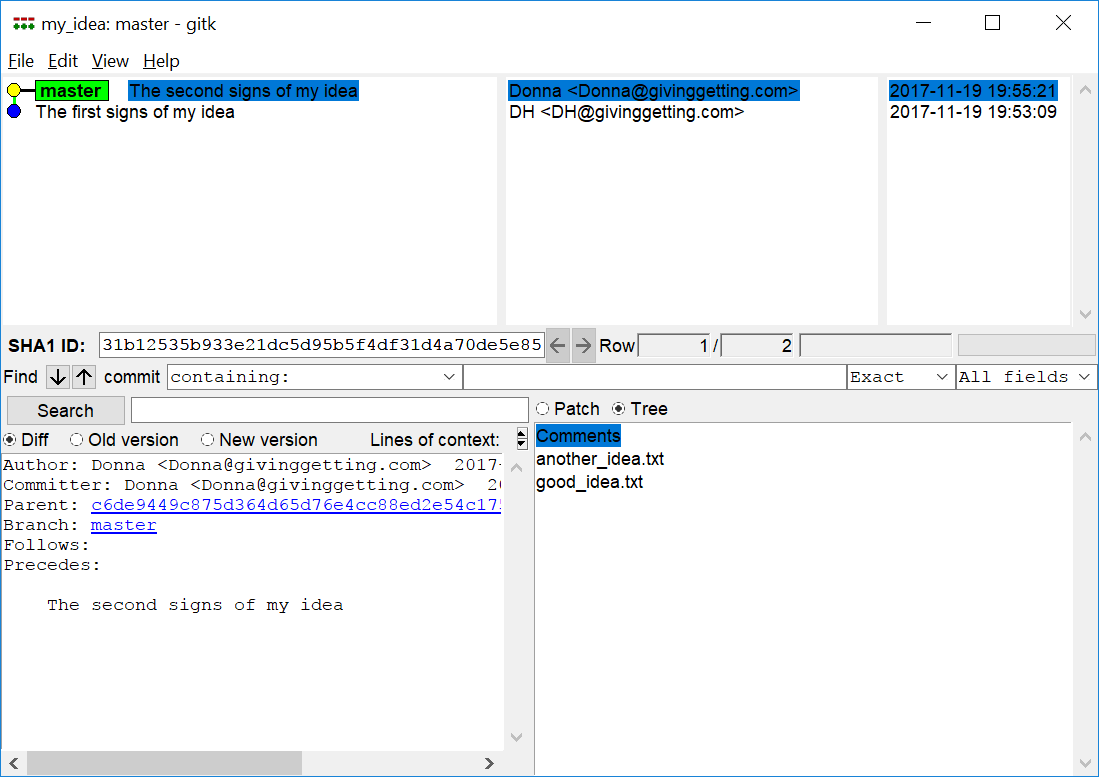
* Go to menu Edit – Options. Set your username and email address in the Right hand panel (global options).

* Let’s show ourselves that I have become myself. Make a new file another\_idea.txt. Start “Git GUI Here” with a right click, stage the file by clicking on the icon next to the filename, add a message and click commit.

* Open Git history from the Git Gui, choose the “Repository” menu, “Visualize master’s history”. Notice that you have two commits now, and the second one has your name on it.

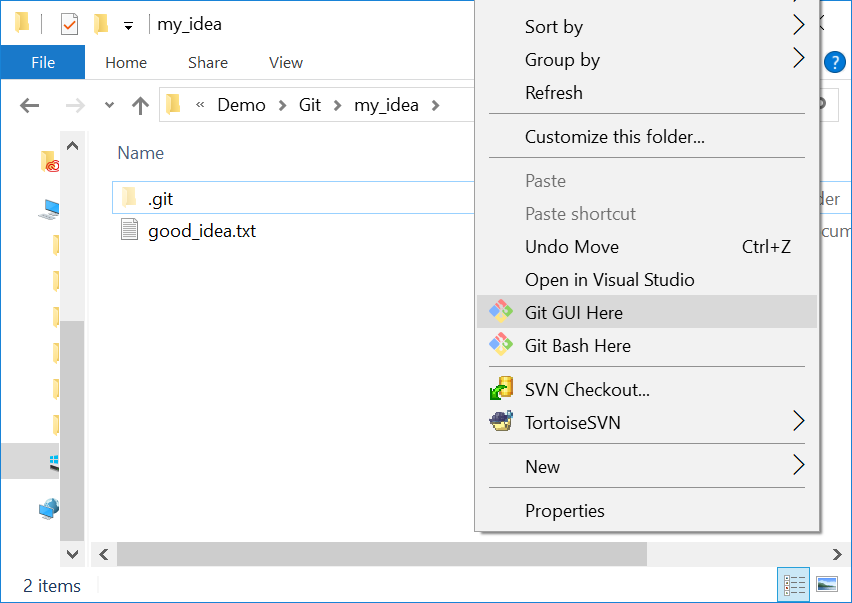


**Making a backup**

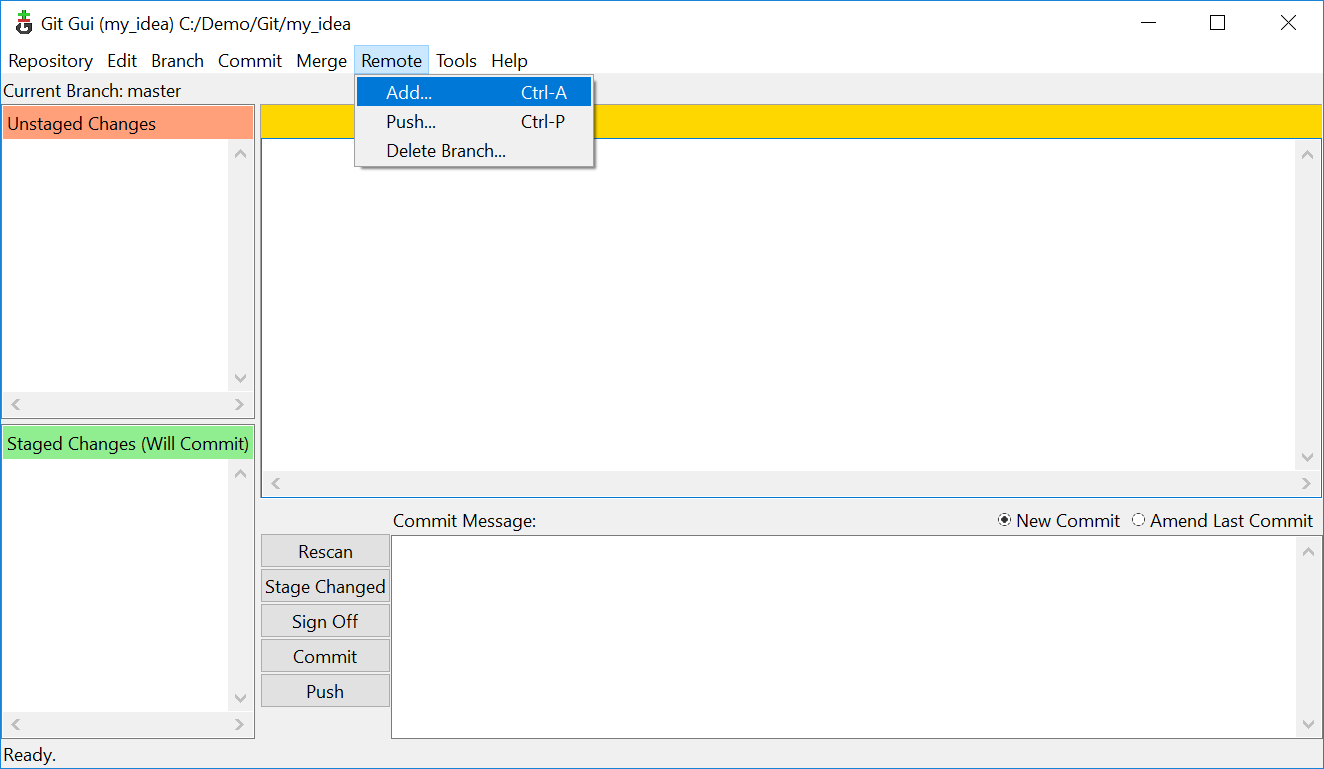
Now you have your changes backed up into your repository, but you will probably want to back up the repository somewhere.

Let’s pretend to back the repository up to a shared drive (but in fact we’ll back up to the C:\Demo\Git).

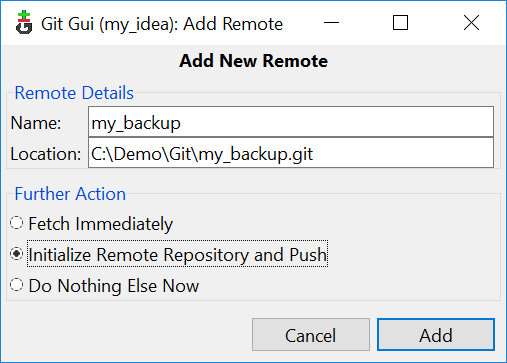
* Right click in the my\_idea forlder, chose “Git GUI Here”



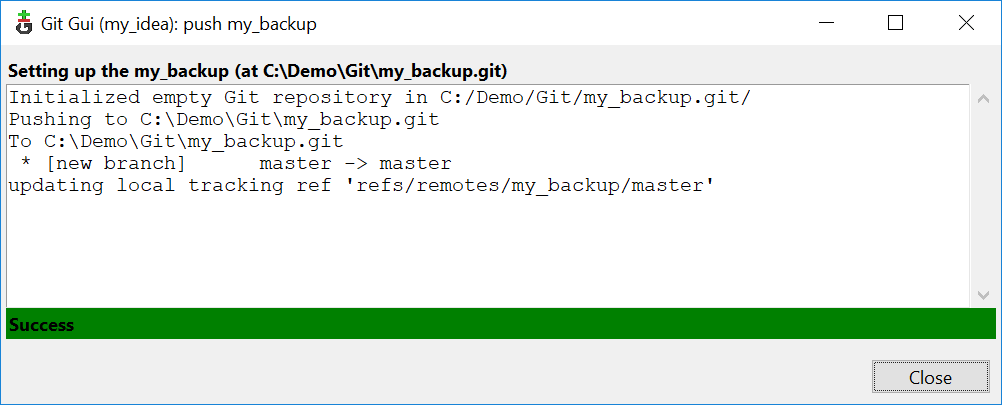
* Choose menu “Remote”, “Add”:



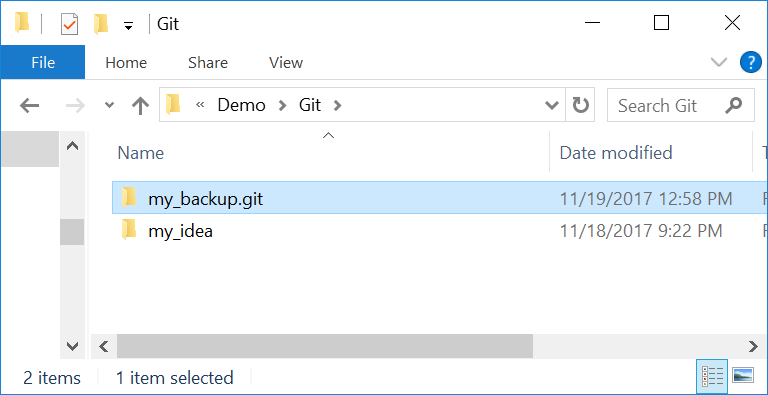
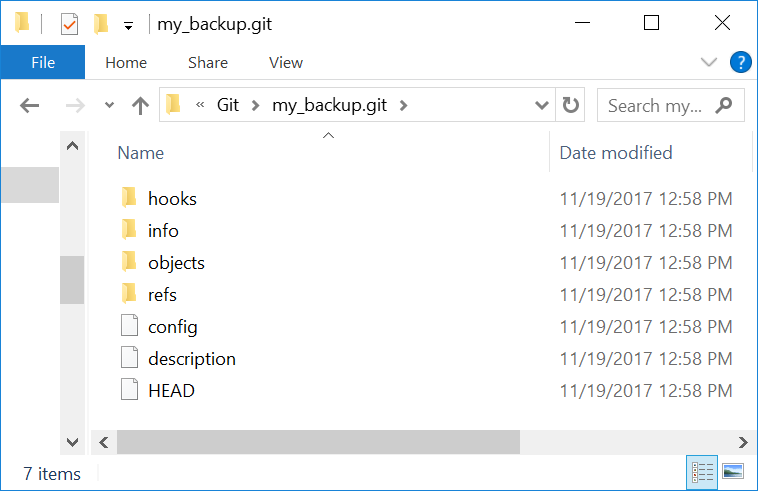
* Give a name for your backup (no spaces) and a path to write the backup to. I like to add .git to the folder name for the backup, because this will be a backup with only the repository (the .git subfolder, but not the .txt files):



* Select “Initialize repository and push”, then click “Add”;



* If you get “Success” then your files have been backed up to this other folder.

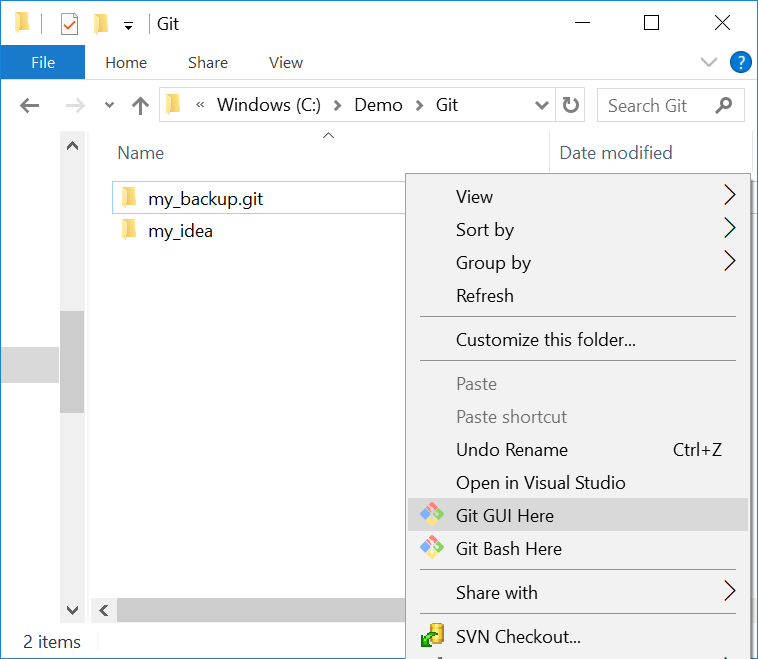
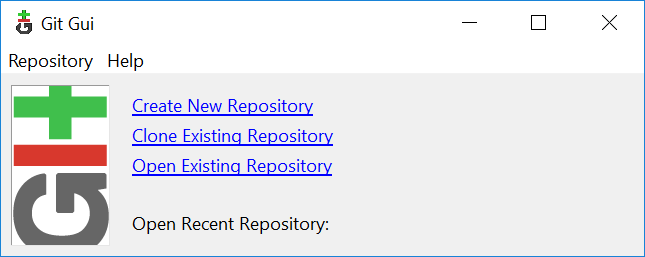
 

**Working from another machine**

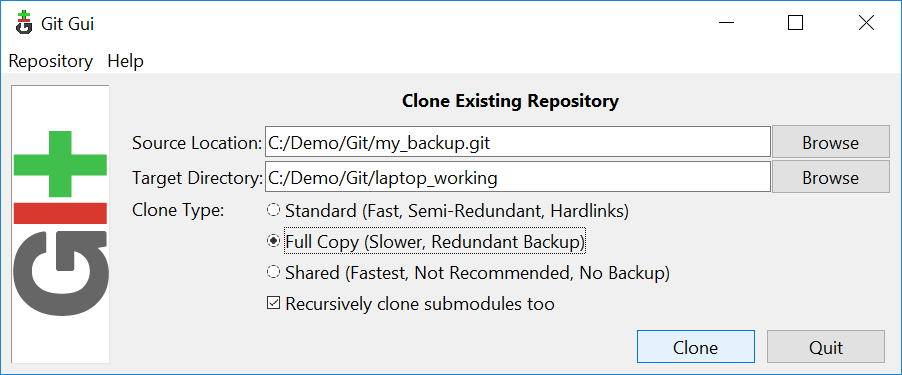
Now you can work from any computer and send changes back to the backup copy.

We can pretend to do this by making a new copy as if it was on (say) your laptop. But in fact we’ll create the folder in the C:\ drive again.

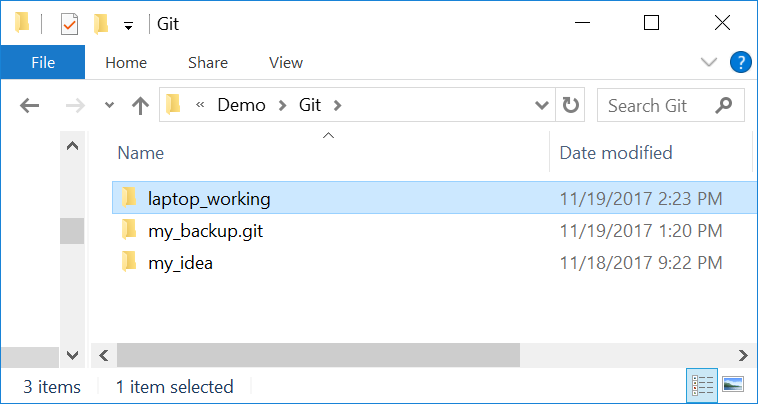
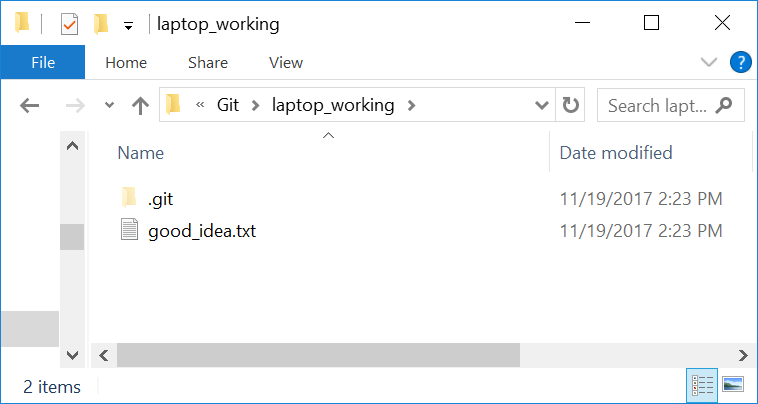
* Right click on the C:\Demo\Git, choose “Git GUI Here”:

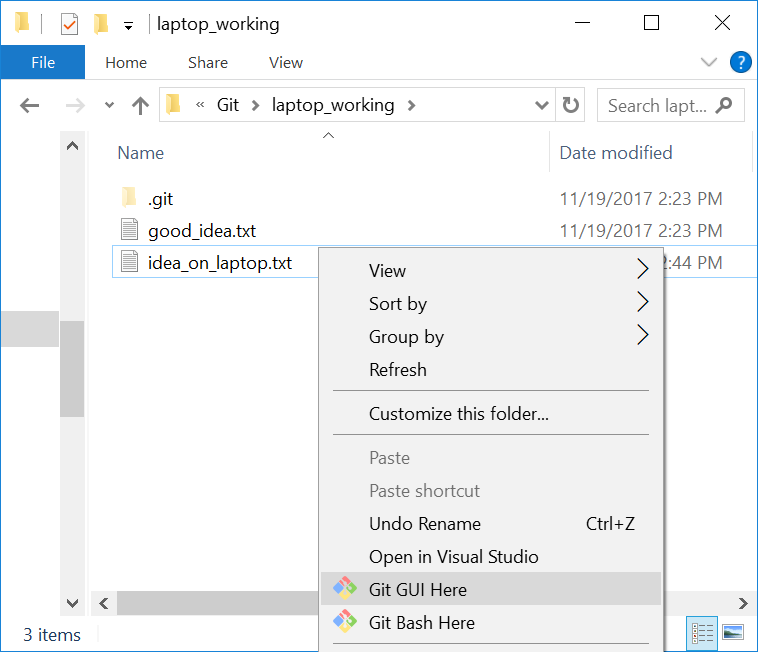
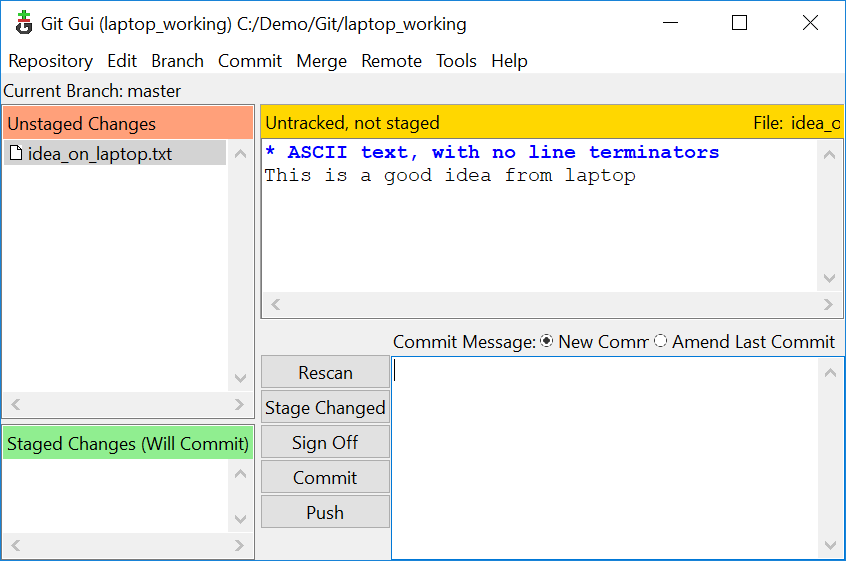
* Select “Clone existing repository”. For the “Source location”, browse to the backup that you just made. Type C:/Demo/Git/laptop\_working for the target directory. Choose “full copy” from the options (for safety). Click on “Clone”;



* You now have a C:/Demo/Git/laptop\_working folder that is a clone of the “my\_backup” repository, and that also has the same data as the C:/Demo/Git/my\_idea folder.

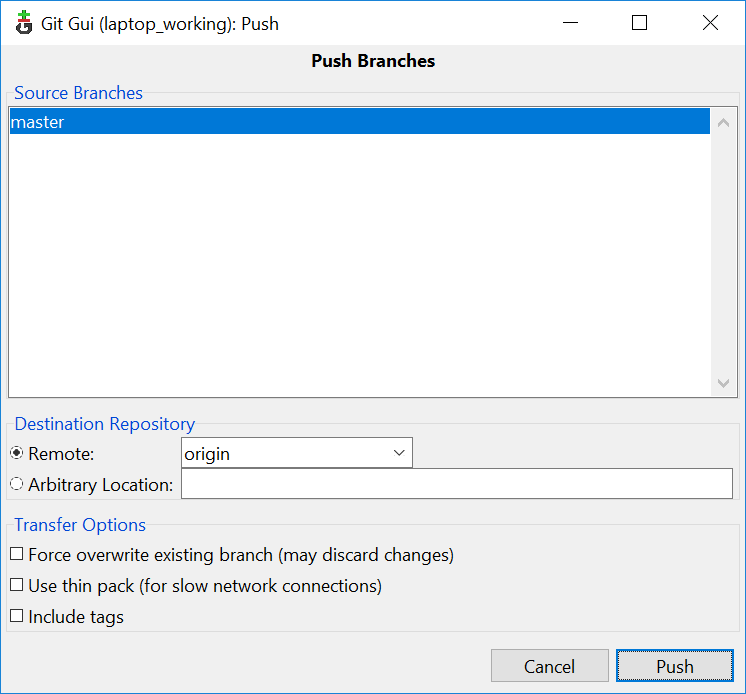
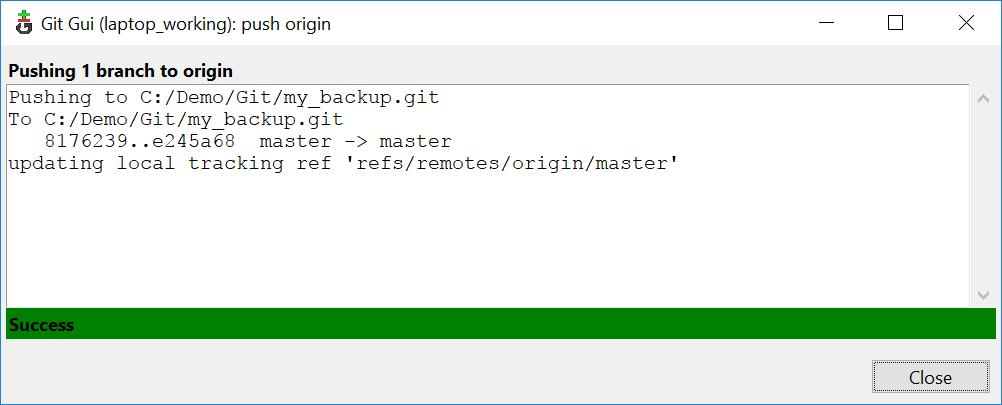
 

* Let’s make some changes on the laptop. Go to the laptop\_working folder. Make a new file idea\_on\_laptop.txt and type some text in it. Save.
* Right click, “Git GUI Here”, stage the idea\_on\_laptop.txt file, add a commit message and click “Commit”;

Now you have this contents in your laptop\_working folder.

You put the changes back into the backup by doing a “Push”

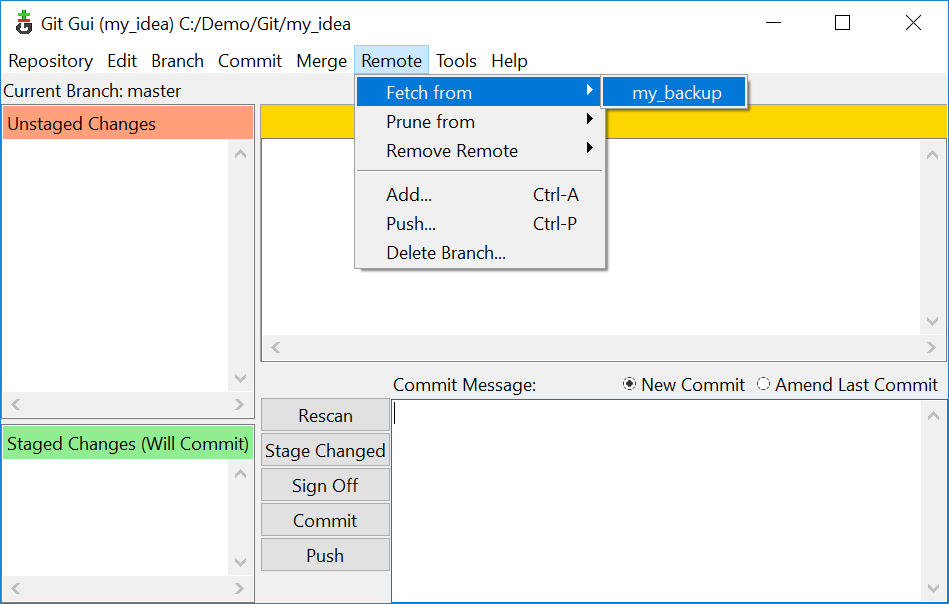
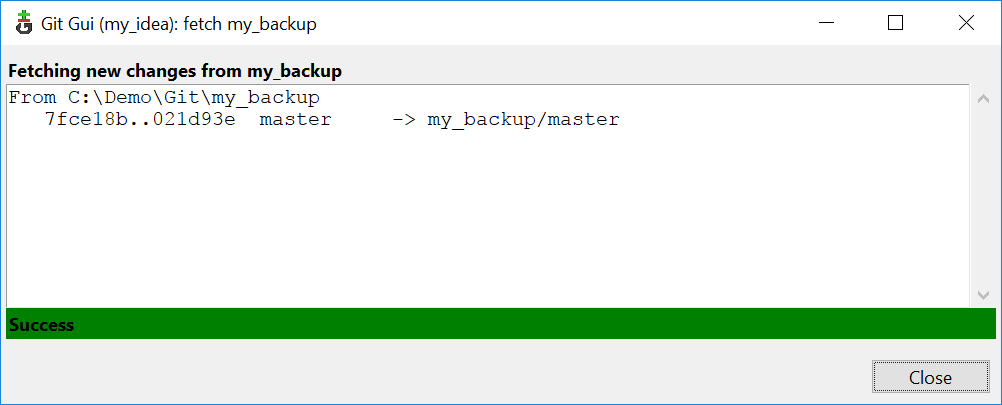
 

The changes go back to the my\_backup.git repository.

**Getting changes from a common backup**

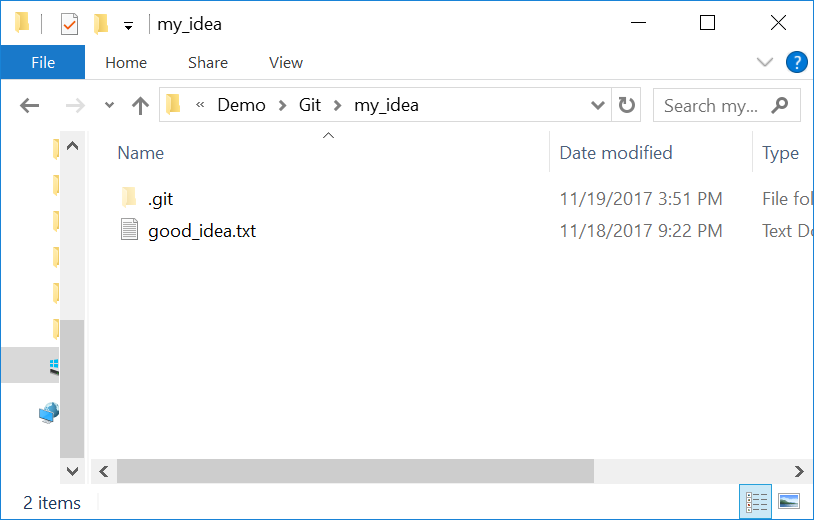
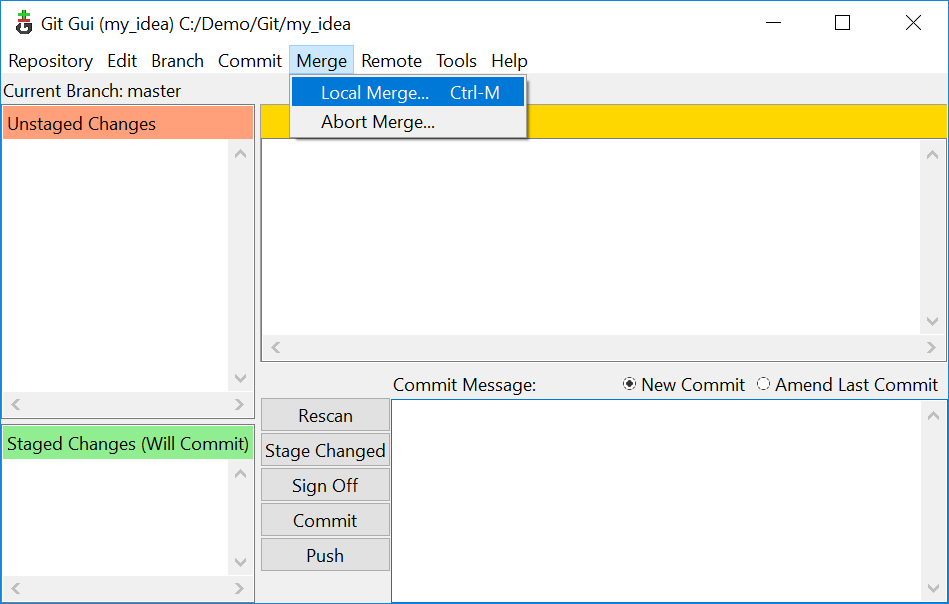
Let’s pretend that we’ve gone back to our original computer and we want the changes that we pushed from the laptop.

* Go back to the my\_idea folder. Notice you don’t have the idea\_on\_laptop file yet.
* Right click, “Gui GUI Here”, “Remote” menu, “Fetch from” from the my\_backup remote.

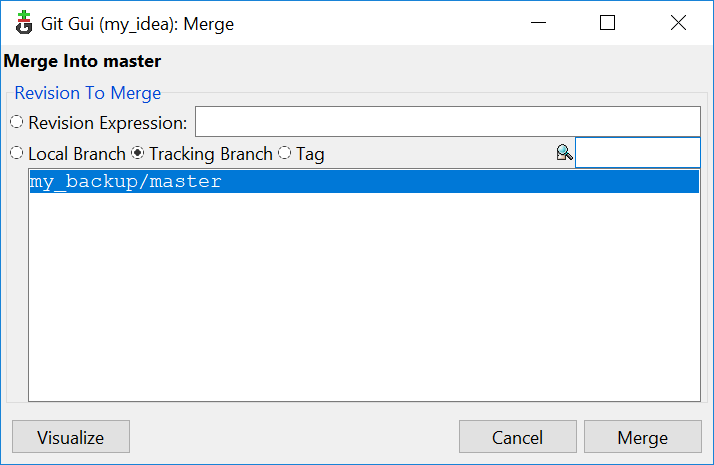
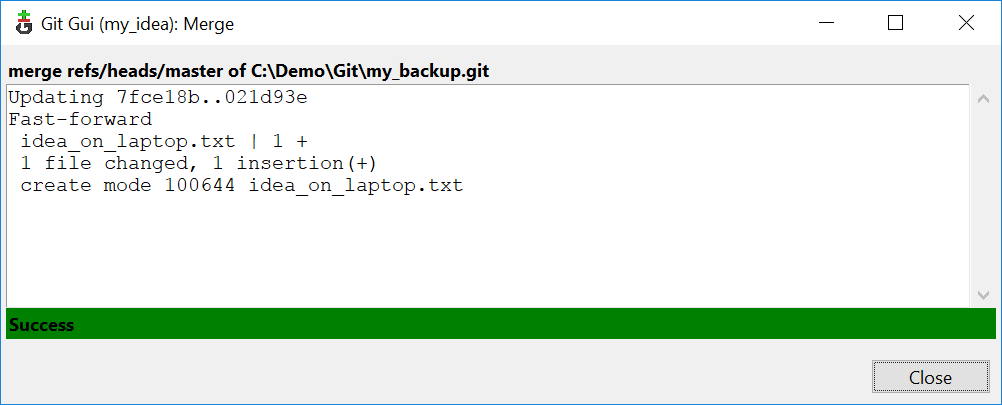
 

Now you have the changes in the repository (the .git subdirectory) – but not in the working tree.

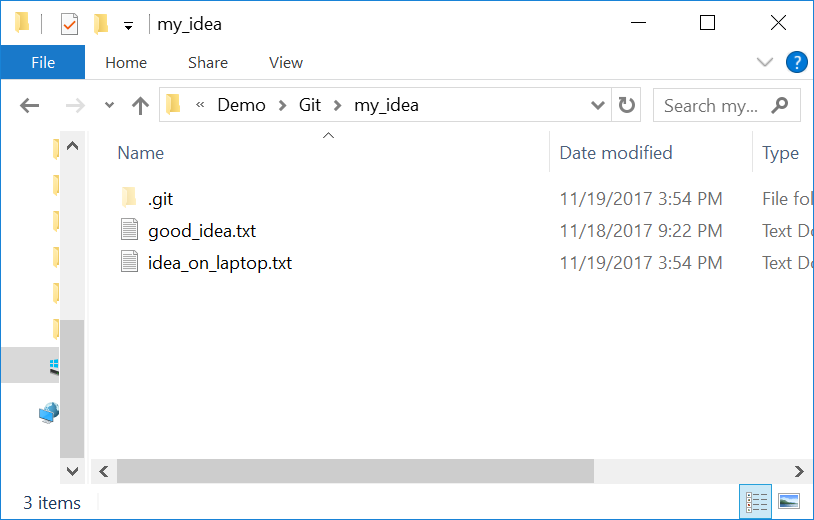
Specifically, we don’t have the idea\_on\_laptop.txt file in the folder yet:

In the dialog, accept the defaults and click “Merge”:

Now you have your file – and you are synchronized with the laptop.



**Review**

* Code states stored so you can
* See what you’ve done
* Go back to an earlier state if you make a mistake
* Send files to and from different computers
* If you are working with someone else they can add their changes
* You can see what they did, they can see what you did
* It’s easy to undo if one of you makes a mistake

This system is powerful, very useful and sometimes confusing. Consult an expert if you get confued. It is fairly difficult to lose data if you ask an expert after you have made a mistake or got confused.

**Reference**

<http://matthew-brett.github.io/pydagogue/git_gui_windows.html>

Git allows groups of people to work on the same documents (often code) at the same time, and without stepping on each other's toes. It's a distributed version control system.