# Session 2 - Open Source Project (SourceForge & GitHub) and TortoiseGit Exercises

### Activity 1.

Visit the http://sourceforge.net/ web site. Click on the OpenSource Software 🡪 All Software option. On the left side of the screen, under Category, click on Software Development. Read through some of the open source projects and summarise what kinds of projects are being hosted. Write down 10 different kinds (or categories) of projects below.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name five projects that you have used their products.

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
4. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Activity 2

In the http://sourceforge.net/ web site, look at one of the Top Download Projects (e.g. FileZilla@) of the week. This list is changing every day. Look at the project details (Summary, Reviews, Support, Forums etc.) and answer the following questions:

1. How many people are active developers? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. When the project was registered? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. How many public forum messages posted? (Browse at the Support and Project Forum messages to get a feel about their discussions.) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. When was the last message posted? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. What was the last message about? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Is the project still active? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Activity 3 (Search GitHub repositories)

1. Register a user account with GitHub. Make sure that you keep your login information. In the later sessions, we are going to use this account to access to your open source projects.
2. How do you search repositories in the GitHub? You may need to review the following article:

<https://help.github.com/articles/searching-repositories/#search-by-repository-topic>

Try to search the following:

Visit <http://github.com> 🡪 choose Explore or Explore GitHub.

1. In the Search box, e.g. type:

* FileZilla in:name

Matches repositories with FileZilla in their name.

* FileZilla in:name, description

Matches repositories with FileZilla in their name or description.

* FileZilla in:readme

Matches repositories mentioning FileZilla in their README file.

1. Search based on number of forks

* FileZilla forks:>5

Matches repositories that with at least 5 forks.

1. Search on the size of the repository

* size: <100

Matches repositories that are smaller than 100KB.

1. Search based on when a repository was created or last updated

* [FileZilla created:<2015-01-01](https://github.com/search?q=webos+created%3A%3C2011-01-01&type=Repositories)

Matches repositories with the word "FileZilla" that were created before 2015.

* [FileZilla pushed:<2016-02-01](https://github.com/search?q=css+pushed%3A%3C2013-02-01&type=Repositories)

Matches repositories with the word "FileZilla" that were pushed to before February 2016.

* [FileZilla pushed:>=2016-03-06 fork:only](https://github.com/search?q=case+pushed%3A%3E%3D2013-03-06+fork%3Aonly&type=Repositories)

Matches repositories with the word "FileZilla" that were pushed to on or after March 6th, 2016, and that are forks.

1. Search within a user’s repositories

* user:pld-linux

Matches repositories from @pld-linux.

1. Search by repository topic

* topic:modelling

Matches repositories that have been classified with the topic “modelling”.

1. Search based on the number of stars a repository has

* stars:>=100 fork:true language:java

Matches repositories with the at least 500 stars, including forked ones, that are written in PHP.

1. Search based on main language of a repository

* hotel language:java

Matches repositories with the word "hotel" that are written in Java.

### Activity 4 - TortoiseGit

<https://tortoisegit.org/docs/tortoisegit/>

Last session we have introduced the Git Bash commands. Git is strictly a command-line tool.

This session, we look at TortoiseGit which is a Windows based Git repository management client. This tool uses the Windows UI to quickly execute commands to your git repository. When you right click it pops up a context menu.

It is an open source software released under GNU General Public License. It is an alternative to GitHub for Windows, GitHub Desktop or TFS-Git (Team Foundation Version Control).

GitHub is a Git repository web-based hosting service which offers the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. However, GitHub provides a web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as wikis, task management, and bug tracking and feature requests for every project.

1. Install TortoiseGit

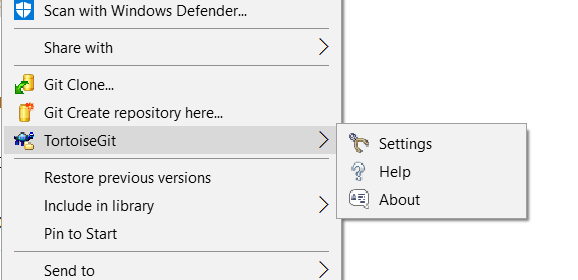
Check whether your computer has installed the TortoiseGit by pressing  🡪 Control Panel 🡪 Programs 🡪 Programs and Features, do you see TortoiseGit 2.7.x. installed? The latest version download is 2.7.1

*If not, you should be able to find the Git installation file from L:\its\files\...\git\TortoiseGit-2.7.0.0-64bit.msi and install it into your computer using the default options.*

You can download the TortoiseGit from the official site. <https://tortoisegit.org/>

1. Check the TortoiseGit user settings

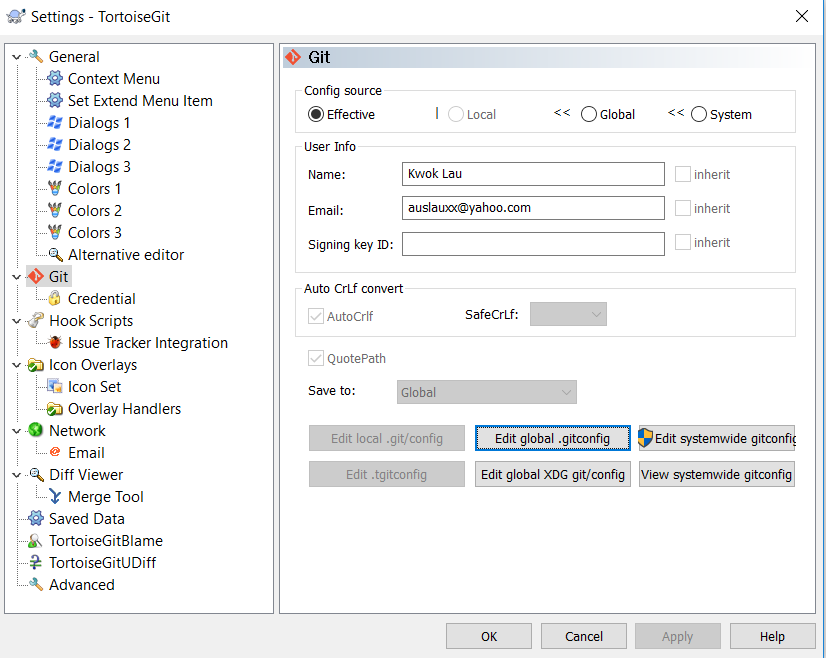
After the installation, use File Explorer. Right click on a folder, you should be able to see the intelligent Git menu as the following:

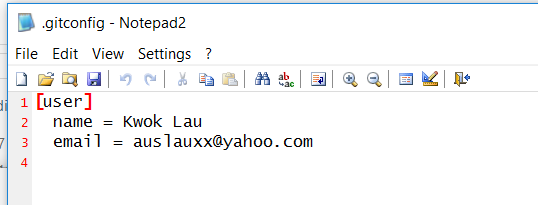


Choose the Settings 🡪 Git 🡪 Edit global .gitconfig

In Linux, file starting with the . is a hidden file. If you do not see it, in File Explorer, click View 🡪 check the Hidden Items. Also check the File name extensions option.

You then should be seeing your name and email.





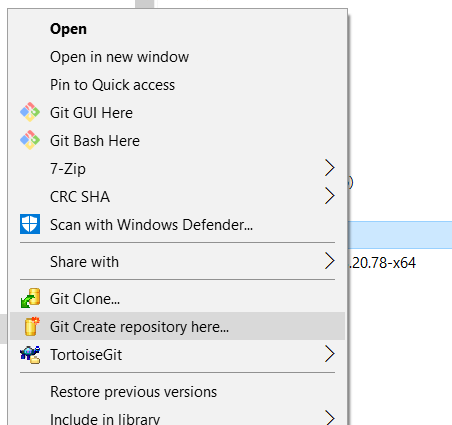
If it is not you, change it to your name and email. Remember: every time you use a new PC, you need to check this.

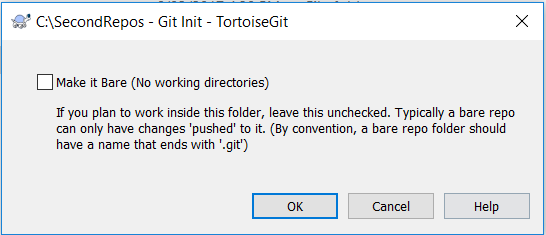
Change it, if it is not in your name & email.

1. Create a new local repository

In File Explorer, create a folder called SecondRepos in the c:\ directory.

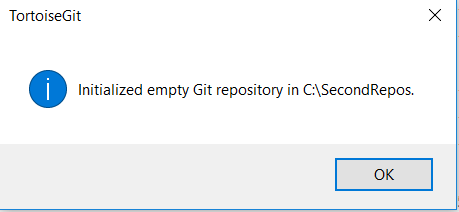
Created repository using TortoiseGit by right clicking on c:\SecondRepos folder and choose Git Create repository here option.



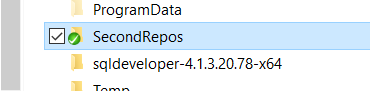


Do not check this.

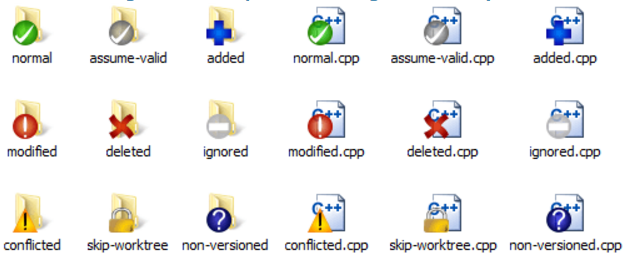
Click OK here. You are going to use this as a working directory. i.e. You will create an empty git folder with the working directory in it.



Click OK. Press F5 to refresh. You should be able to see the overlay icon with the green tick on it.



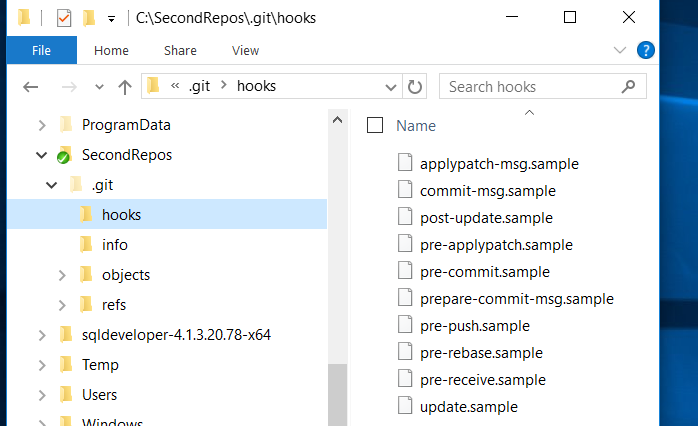
TortoiseGit has several **overlay icons**. They show the status of the files or folders. e.g.



The following table explains the meaning of the overlay icons.

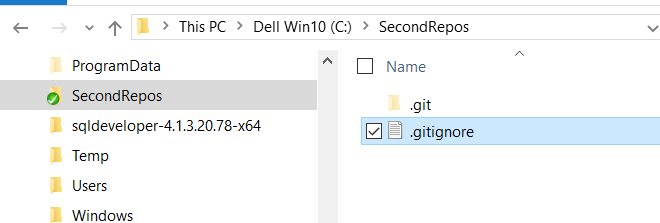
|  |  |
| --- | --- |
|  | A fresh checked out working tree has a green checkmark as overlay. That means the Git status is *normal*. |
|  | As soon as you start editing a file, the status changes to *modified* and the icon overlay then changes to a red exclamation mark. That way you can easily see which files were changed since you last updated your working tree and need to be committed. |
|  | If during an update a *conflict* occurs then the icon changes to a yellow exclamation mark. |
|  | If during an update a *conflict* occurs then the icon changes to a yellow exclamation mark. |
|  | This icon shows you that some files or folders inside the current folder have been scheduled to be *deleted* from version control or a file under version control is missing in a folder. |
|  | The plus sign tells you that a file or folder has been scheduled to be *added* to version control. |
|  | The bar sign tells you that a file or folder is *ignored* for version control purposes. This overlay is optional. |
|  | This icon shows files and folders which are not under version control, but have not been ignored. This overlay is optional. |

At the moment, the C:\**SecondRepos** folder has a **.git** folder in it. Make sure that your **File Explorer** has selected to display the hidden files. i.e. **File Explorer** 🡪 **View** 🡪 check **Hidden Items**.

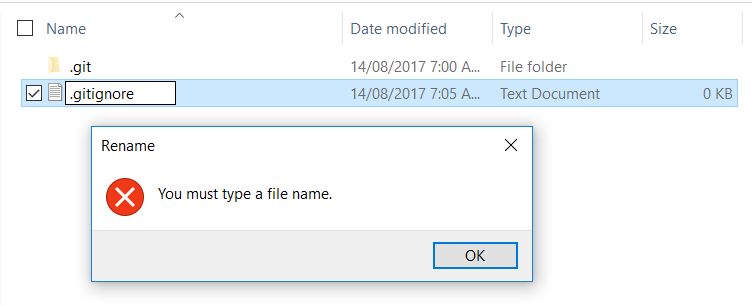


1. Version control the .gitignore file patterns for your target language

Download the .gitignore file from Moodle session materials. Copy the .gitignore file into the C:\SecondRepos folder. Git uses this file to determine which files and directories to ignore, before you make a commit.



*Note: If you're using Windows it will not let you create a file without a filename in Windows Explorer. It will give you the error "You must type a file name" if you try to rename a text file as .gitignore*

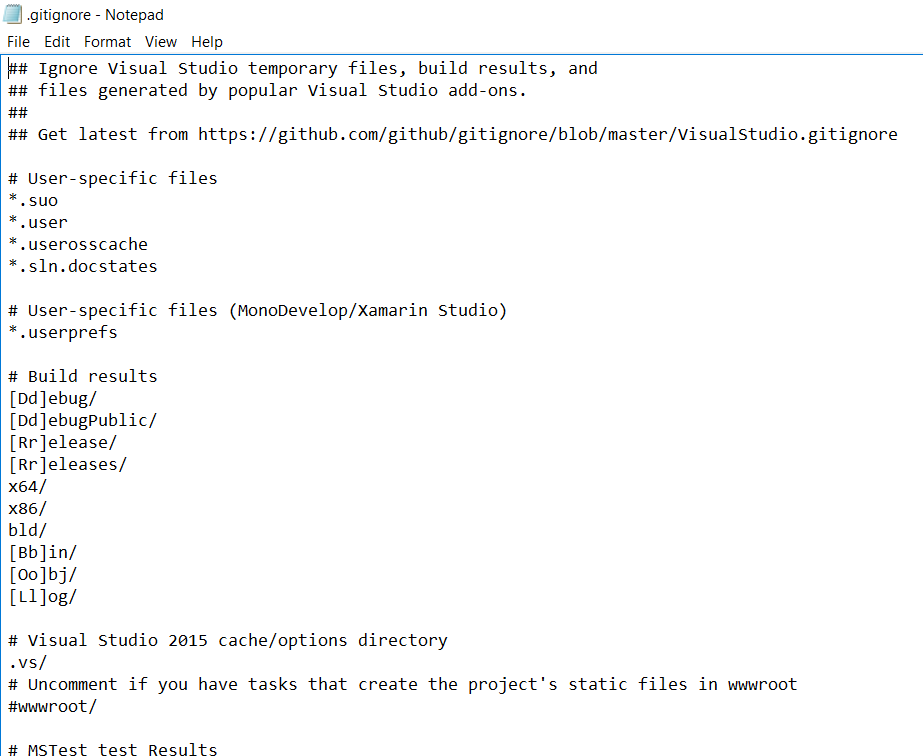


*To get around this I used the following steps*

1. *Create the text file gitignore.txt*
2. *Open it in a text editor and add your rules, then save and close*
3. *Hold SHIFT, right click the folder you're in, then select Open Powershell window here (i.e. Press the Windows key* *, type Powershell --> Windows PowerShell desktop app), navigate to the folder (i.e. using ls or cd command etc.)*
4. *Then rename the file in the command line, with*

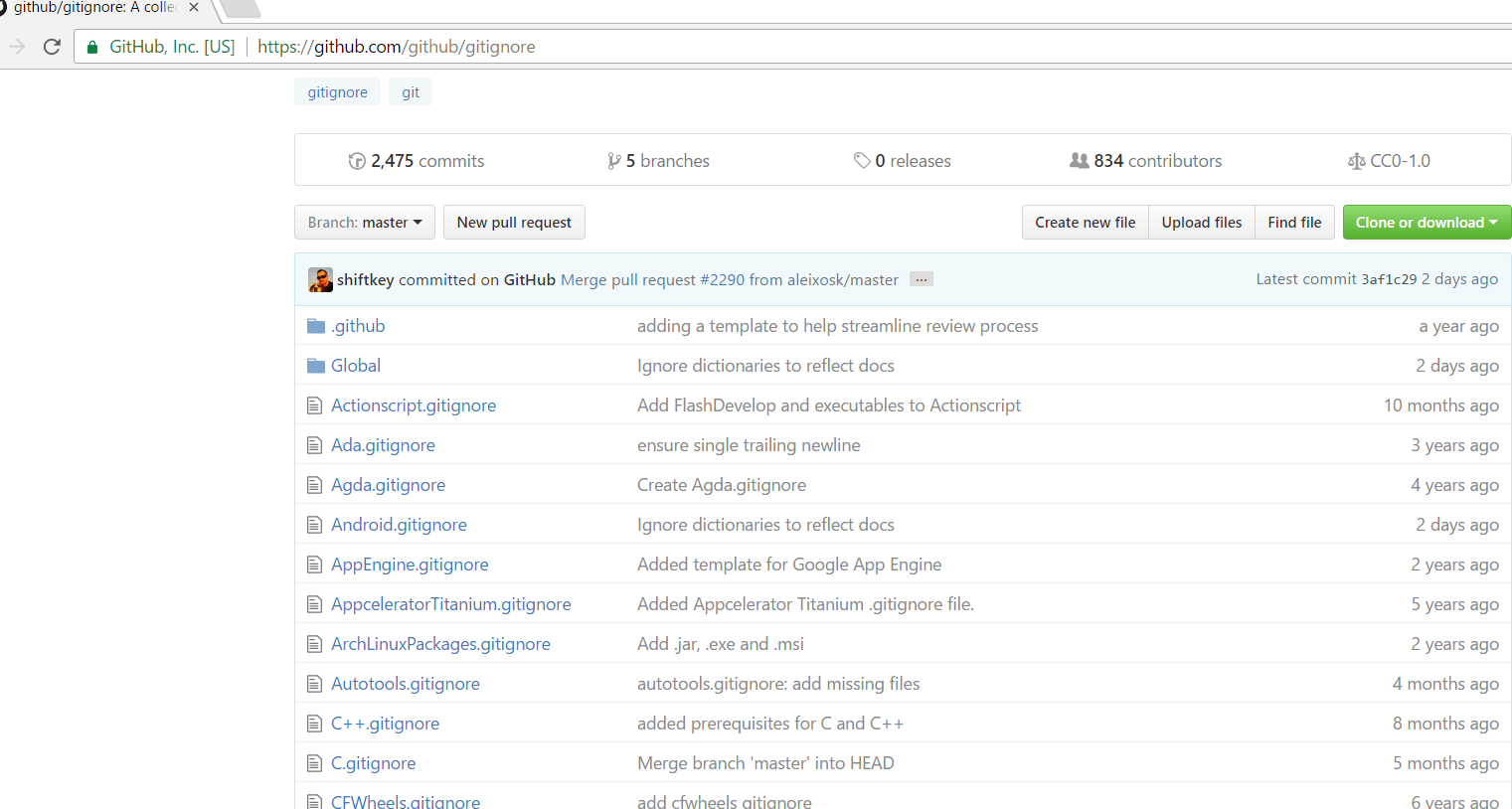
> ren gitignore.txt .gitignore

Use the Notepad to open the empty .gitignore file.



You should be able to see some file with extension like debug release bin obj log etc. and many more.

GitHub maintains an official list of recommended  .gitignore  files for many popular operating systems, environments, and languages  <https://github.com/github/gitignore>



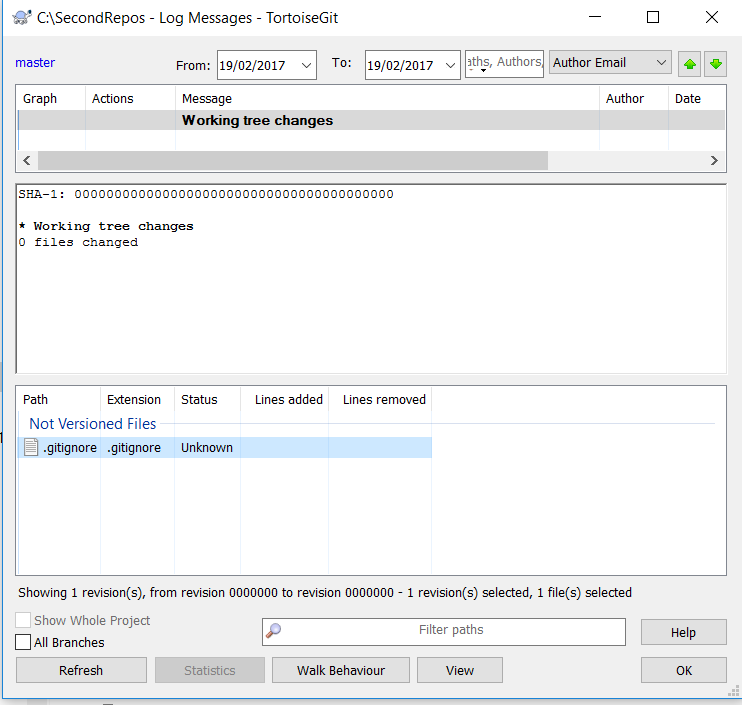
Do you able to find the Java .gitignore file? Now download the file from the GitHub and unzip it. One of the file sure be for Java. In here we use C#.

\*\* Note: for consistency here, download the .gitignore file from Moodle Session 2 and copy and paste the content into the *empty* .gitignore file in the C:\SecondRepos folder.

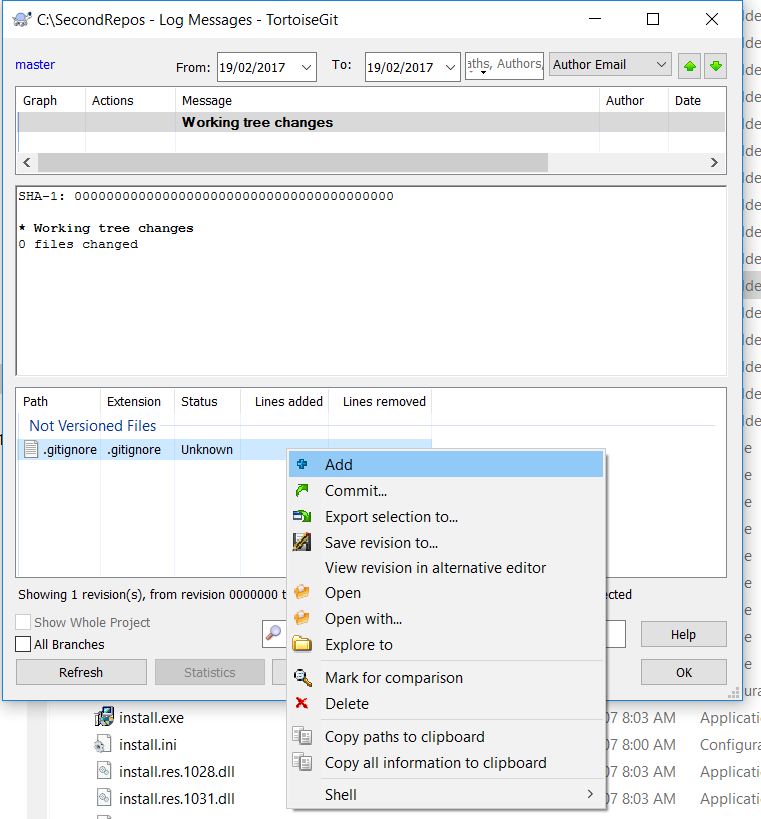
A .gitignore file (with appropriate file and directory names to be ignored) should be committed into your repository, in order to share the ignore rules with any other users that clone the repository.

The following step shows you how to source control the .gitignore file into the repository.

Right click on C:\SecondRepos 🡪 TortoiseGit 🡪 Show log.

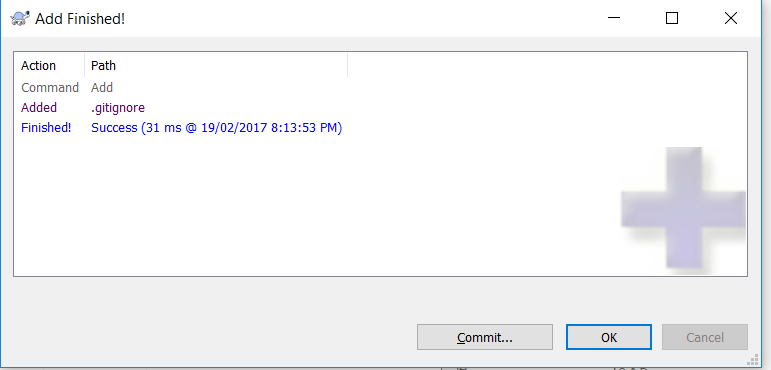


You can see that the .gitignore file is not version control yet. Select the file .gitignore.

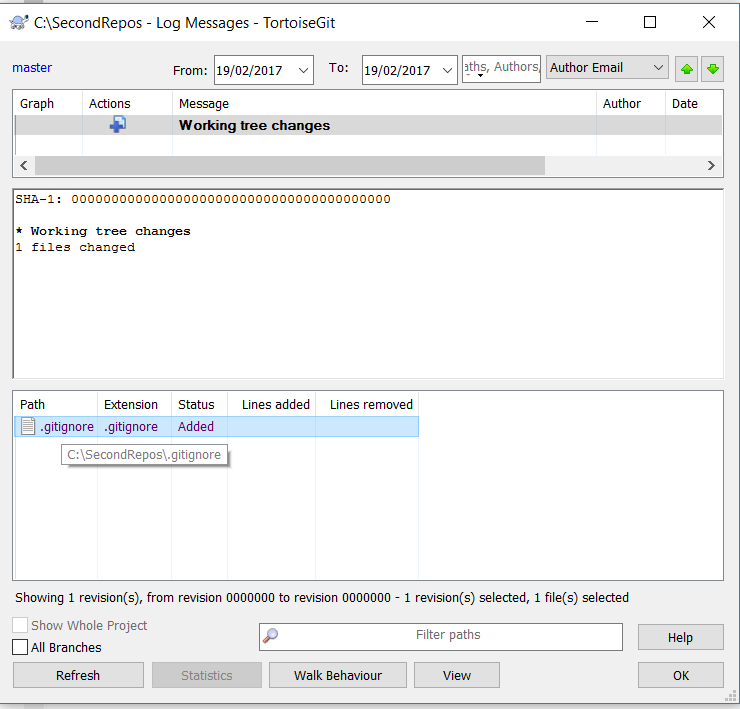


You may use the Add option to stage the files/directories for update.

The Commit option will do the staging as well as committing the changes to the repository.

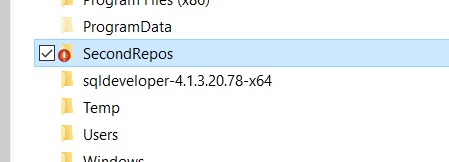


Choose OK in the meantime. Right click on C:\SecondRepos 🡪 TortoiseGit 🡪 Show log again.



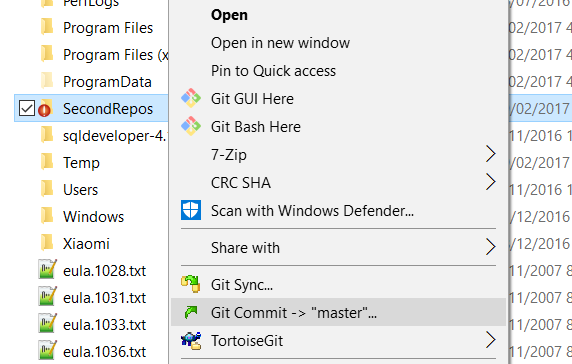
The status is Added.

Pay attention that the working directory has the overlay icon  now.

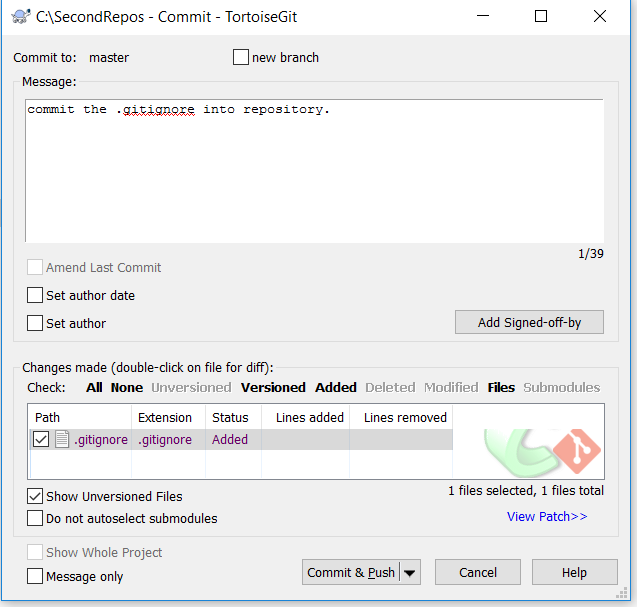


The working tree has changes and needs to be committed.

Now, right click on C:\SecondRepos 🡪 choose Git Commit -> “Master”.



Type a message such as “Commit the .gitignore into repository”.

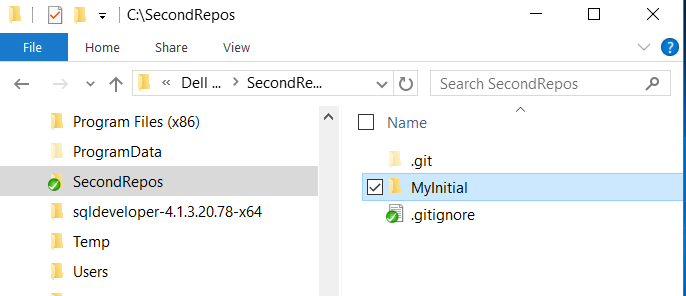


Enter the comment as “Commit the .gitignore into repository”.

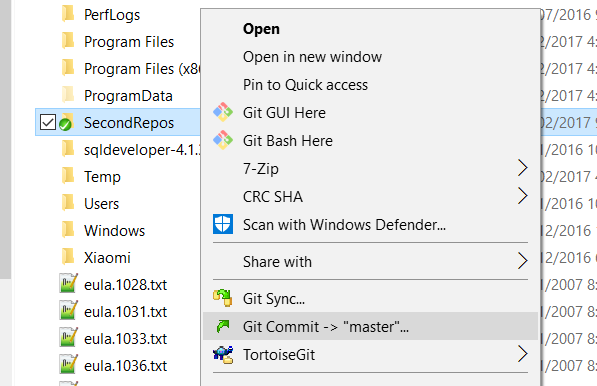
Commit to the master branch and click OK.

1. Version control the MyInitial C# program.

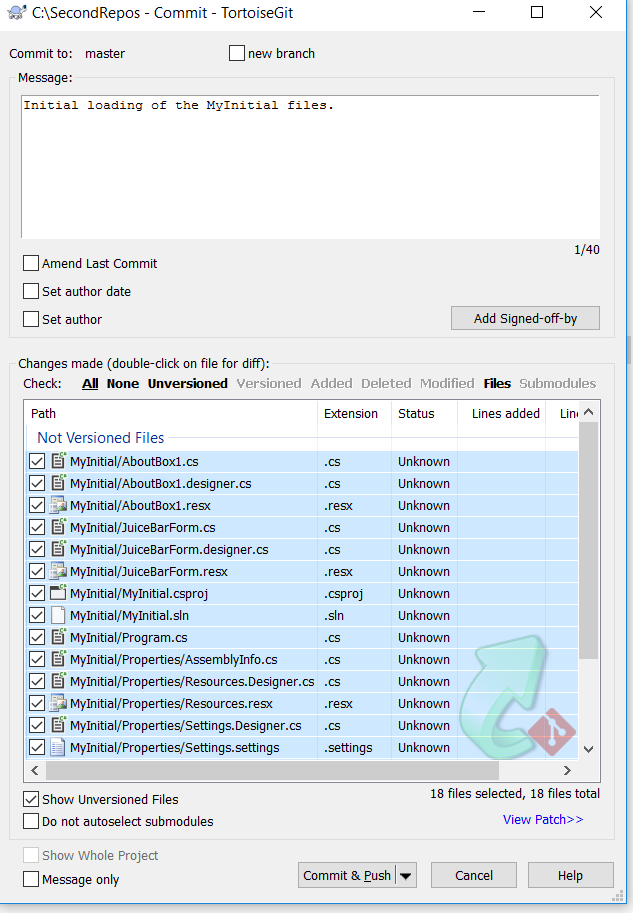
Copy the student files MyInitial C# program (i.e. sources files) into the C:\SecondRepos git folder.



Right click on the SecondRepos directory and choose Git Commit -> “master”.



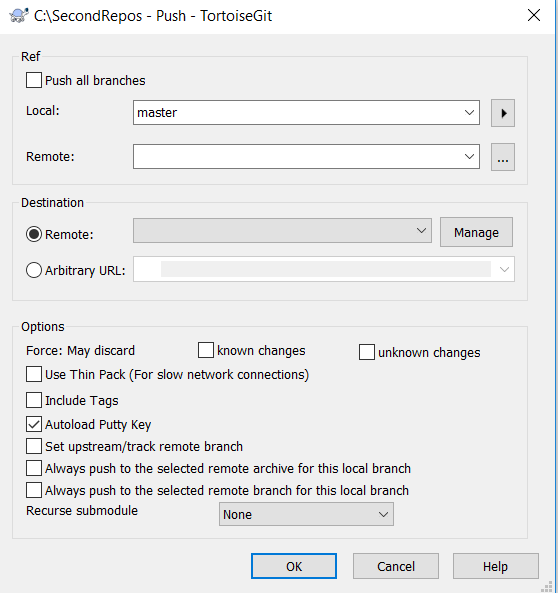
You should be able to see the following. With the .gitignore, it filters the files and only the source files are included. Choose All to select all files.



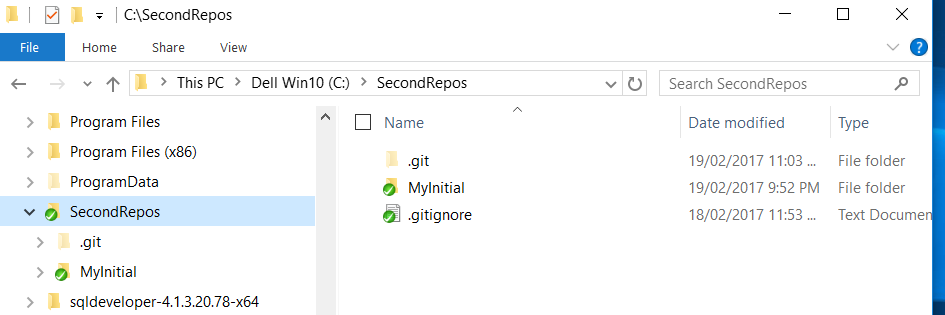
Add a comment: “Initial loading of the MyInitial files.”

Click Commit. Follow the instructions to commit files to local repository.

* choose Commit.



* choose Cancel.



1. Start the development

Now the working directory has been version controlled the files/directories in the repository (i.e. in the master branch) for further development. You may use the Visual Studio 2015 C# to modify the project files in the MyInitial folder.

e.g. Modify the codes on the form with different prices on drinks. e.g.

|  |  |  |
| --- | --- | --- |
|  | From: | To: |
| 12 Ounce | 3.00 | 4.00 |
| 16 Ounce | 3.50 | 5.00 |
| 20 Ounce | 4.00 | 6.00 |

Other than the labels, make sure that you modify the codes as well. Rebuild the codes and execute the program if succeeded. Save all files. (Note: do not commit the changes yet.)

Use File Explorer, see the changes of the overlay icons. Write down which are the files being changed.

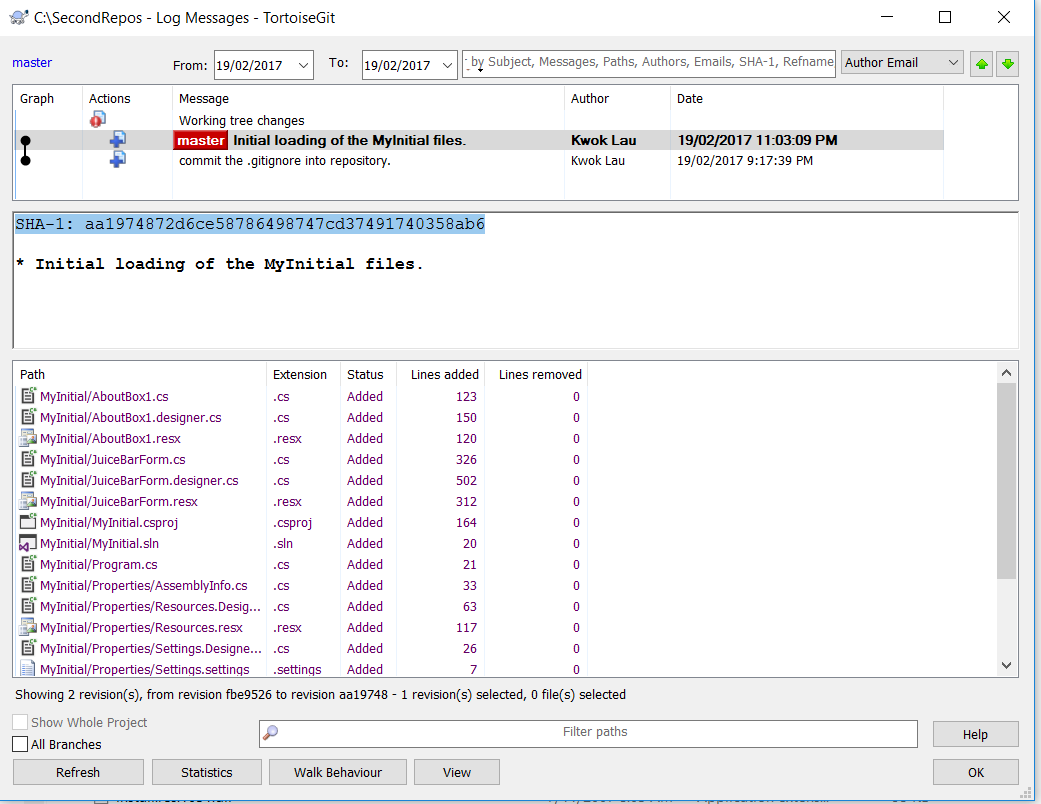
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Right click on the git repository 🡪 TortoiseGit 🡪 Show Log.



Write down the hash keys in the chronological order :

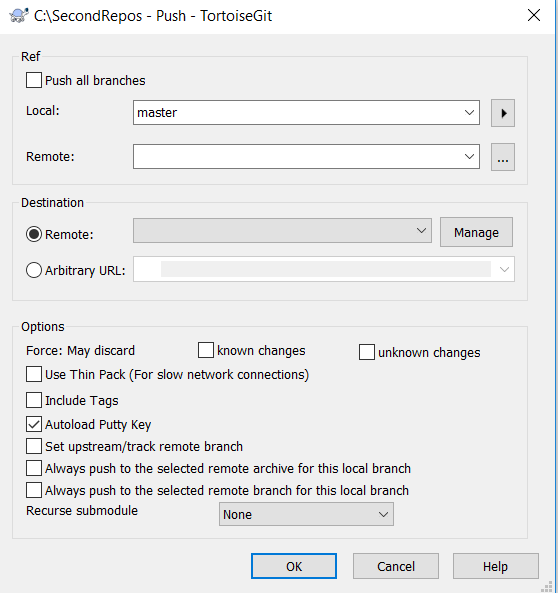
|  |  |
| --- | --- |
| Commit message | Hash keys |
|  |  |
|  |  |

You may commit your changes back to the repository. Right click on the git repository 🡪 TortoiseGit 🡪 Git Commit -> “master”.

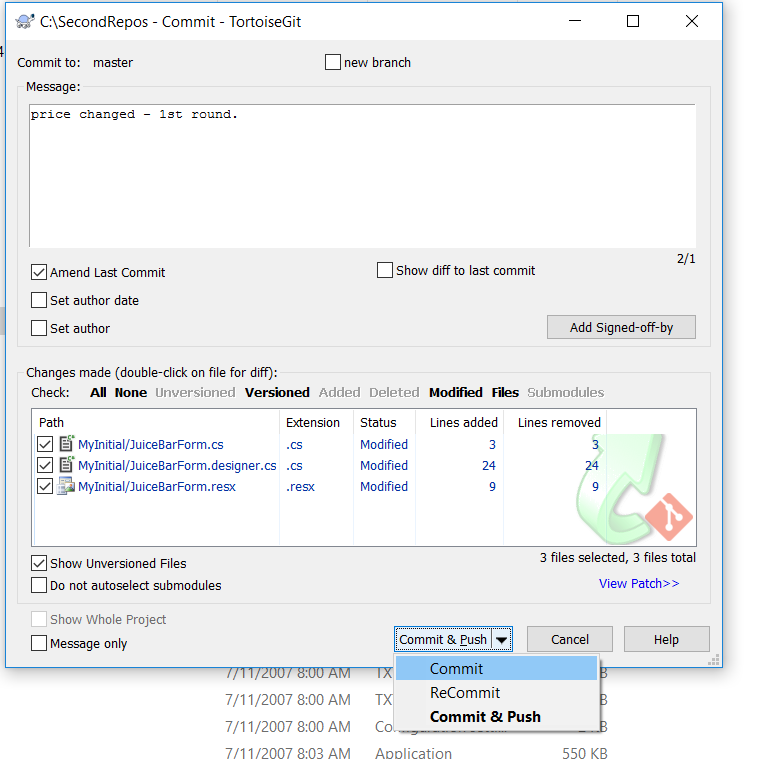
Add a comment: “price changed – 1st round”

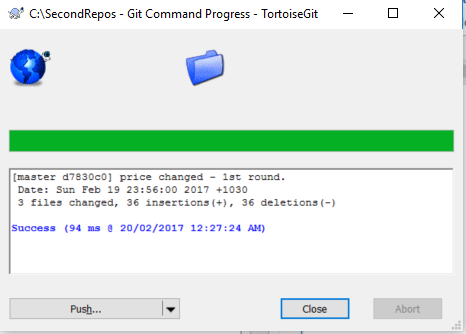
* choose Commit & Push.

Choose Commit.



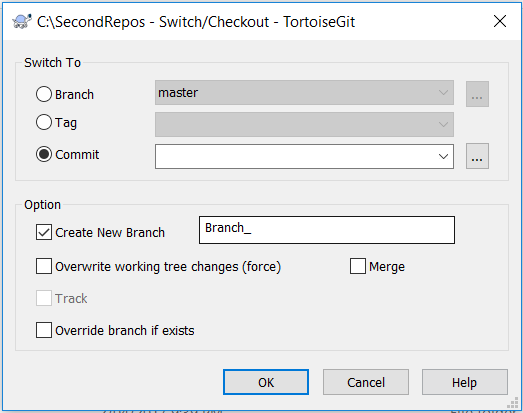
* choose Cancel.



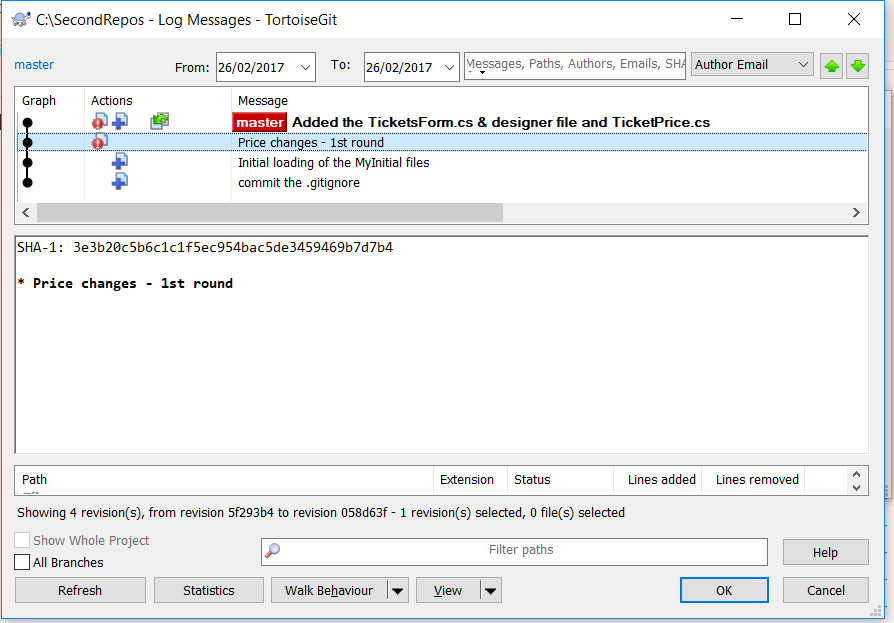


1. Review the show log and Revision Graph.
2. Try with more modification and commit. See the Show log and note the hash index.
3. Check out with an earlier hash index. See what happen to the files. Do they roll back to the earlier changes? \_\_\_\_\_\_\_\_\_\_\_\_\_\_

i.e. right click on the repository 🡪 TortoiseGit 🡪 Switch/CheckOut 🡪



Click on the ….



Choose an earlier version.

See the files in Visual Studio. Does the source code go back to the earlier version?

Similarly, as the previous steps, what about checkout a latest version now. Use Visual Studio to see whether the source code is the latest version.

Additional Resources:

<https://tortoisegit.org/docs/tortoisegit/>