Math 4610 Fundamentals of Computational Mathematics - Topic 8.

Using git to Work Locally on Your Computer

You can chose to work on projects on Github by logging onto the Github web site with your username and password. However, if your internet connection is not as good as you might like, you can use "git" to synchronize the work on your project. The git environment is a Version Control System (VCS) which allows different groups to work on the same project without stepping on each other's toes. In addition, the software keeps track of versions of the software that you can revert to if something goes wrong. In this topic we will learn a bit about git and start using git to synchronize your work on Github. Your work will be downloaded from Github for grading. So, you will need to learn how to push and pull versions of your work as it changes.

Using git to Work Locally on Your Computer

The first thing to do is to make sure that git is available within your terminal or terminal emulator. After opening a terminal, use the following command:

```
koebbe% which git
```

If git is available, you will see output like that in the following figure. In this case, the executable for git is in the folder /usr/bin.
If git is not available, you can install the software on most computers. If you need some help getting the software installed, talk to your instructor.

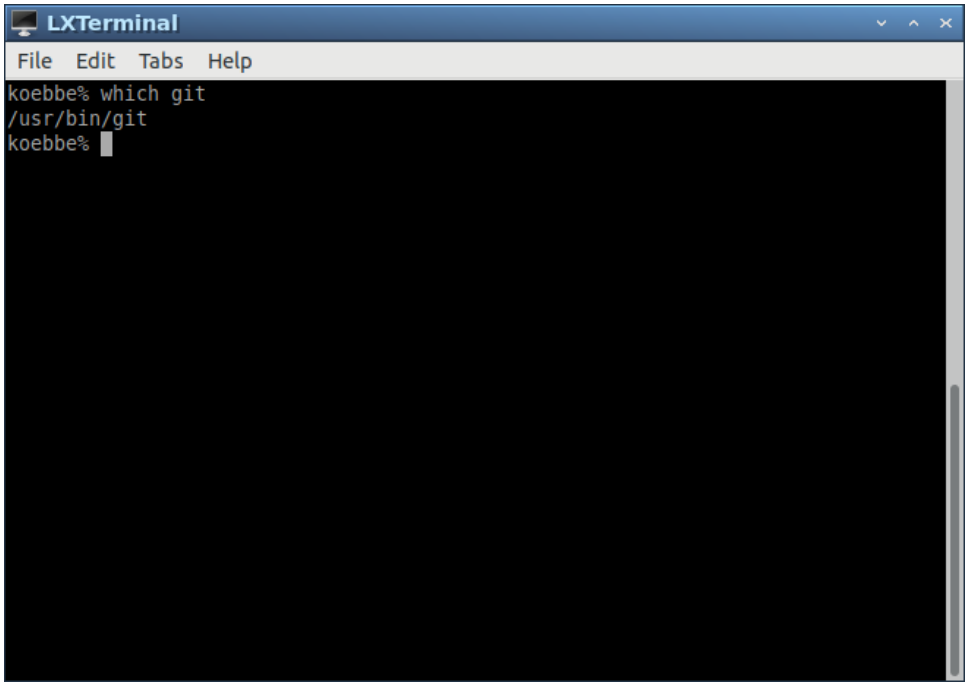

Figure 1: Checking for git using the which command.

Commands in the git VCS

This section of notes will not cover all possible ways to use and modify git. Instead, we will look at some basic commands that git uses to share the data in a repository. The general form for a command using git is the following:

```
% git command [options]
```

To start, we can type in the command

```
koebbe% git --help
```

This produces a couple of screens of output. Another way to display the same output, one screen at a time, is to pipe the output from the command above into another Unix command, more. The result is

```
koebbe% git --help | more
```

The concept of pipes in Unix is to take the output from one command and use this as input to another command. So, the output from

```
koebbe% git --help
```

is piped into the more command. The more command will display output one screen at a time. You should try this on any file in your folder some time to see how is works. Something like

```
koebbe% more myfile.txt
```

will display the contents of the file named, myfile.txt. If you do this on a binary file, the output will not be readable.
What you should notice is all of the options for running commands within git. The end result is that you can read the first screen and then hit a space bar to read the next screen. The rest of this section of notes will go over a bare minimum of git so that you can work on your own computer at home.

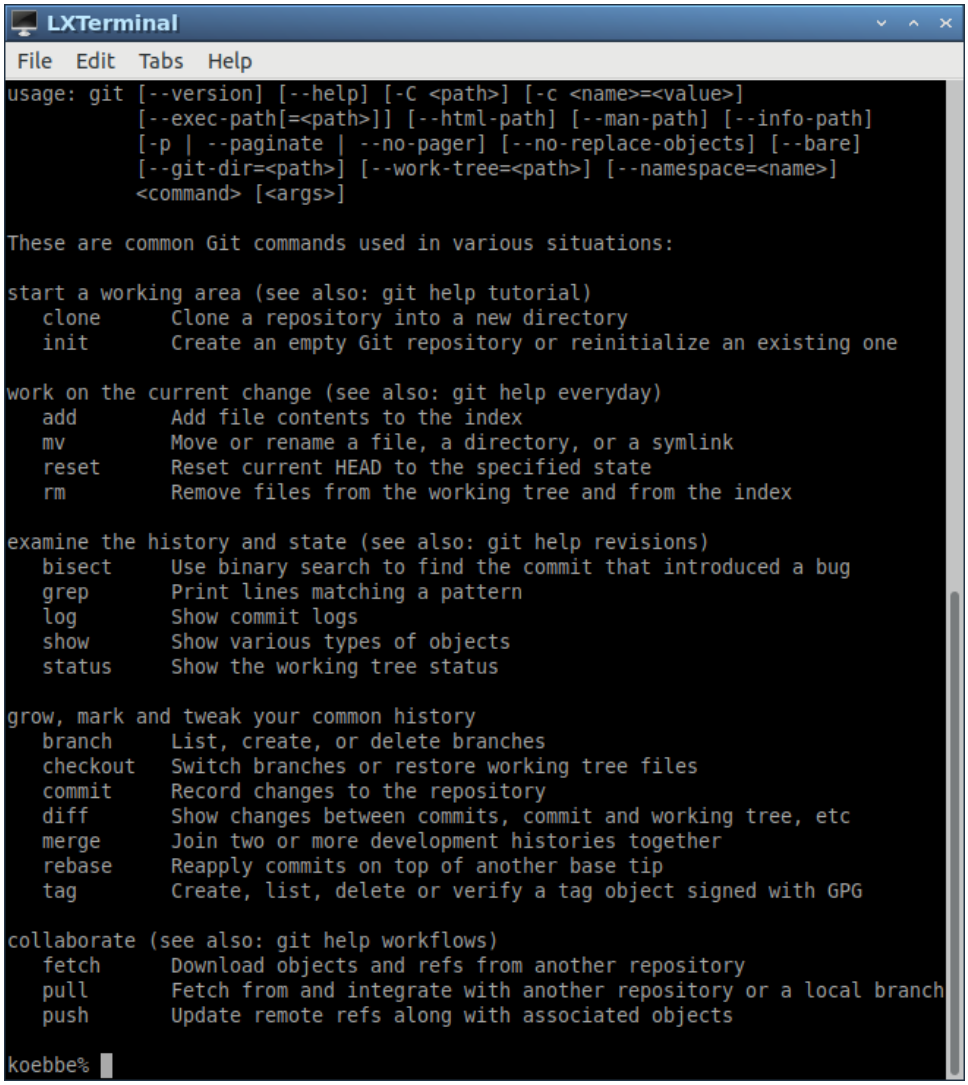

Figure 2: The first part of the output gives about half of the possible options we can use with git.

Initializing a Folder Using the git init

To start, there are two ways to initialize a git folder. The first is to create a new folder and initialize the folder using the following three commands.

```
koebbe% mkdir tempdir
koebbe% cd tempdir
koebbe% git init
```

As the output states, the result is an empty Git repository. The other method will work well when making a copy of the Github repository that already exists. You can clone a repository from another site - Github. To do this, change into the folder where the repository folder will end up.

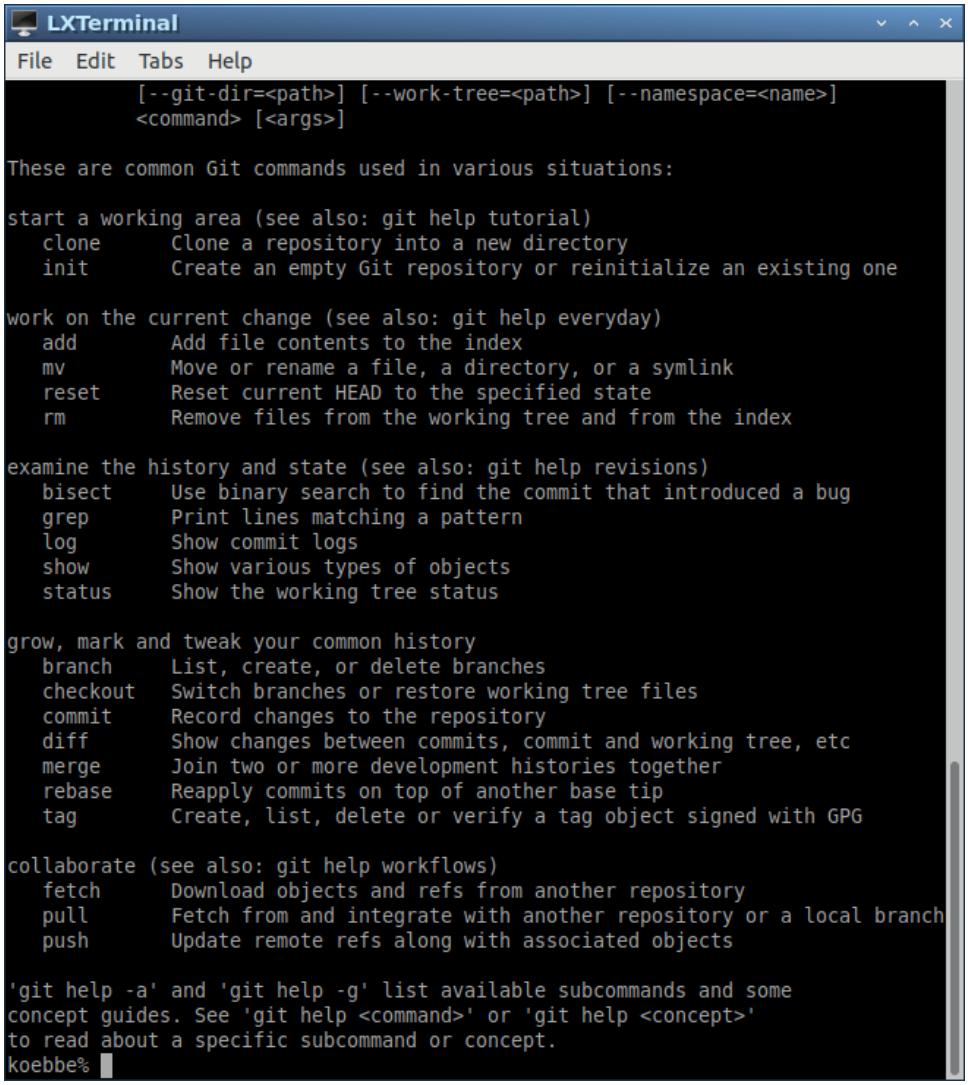

Figure 3: The second part of the output from piping into the more command is shown in this figure.

Initializing a Folder Using the git clone

We can use the clone command in git to make a copy of a repository on Github. To do this, start by changing folders to the location where you want a copy of the repository to be cloned. This can be done as follows.

```
koebbe% cd foldername
koebbe% git clone https://www.github.com/username/repositoryname
```

So, if Fred has chosen fred as a username and the repository, math4610, on Github, the pair of commands would be

```
koebbe% cd repository_location
koebbe% git clone https://www.github.com/fred/math4610
```

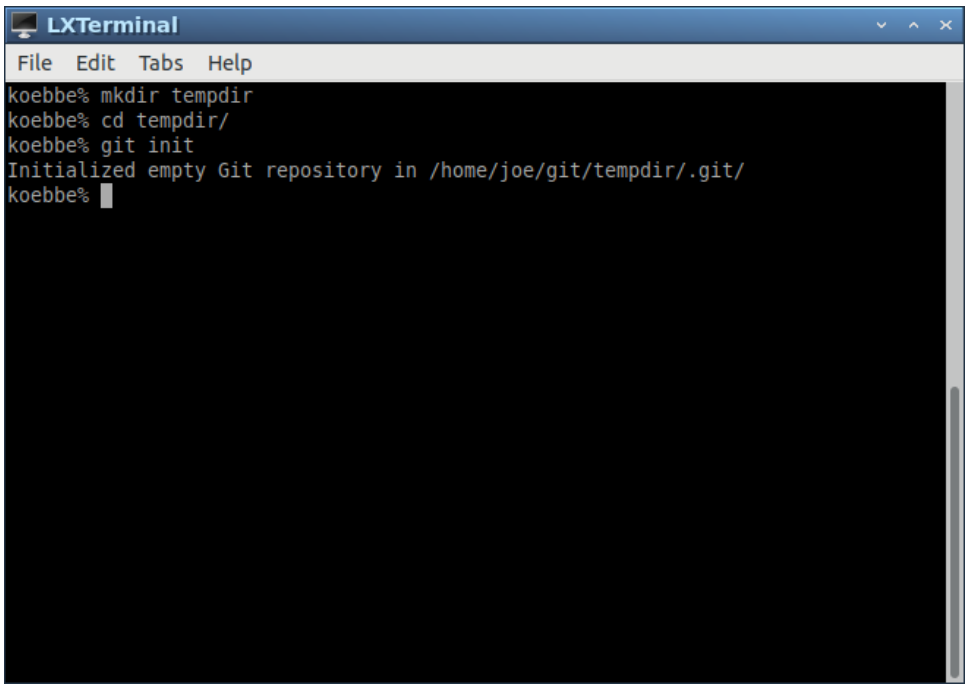

Figure 4: Creation of a local repository using the git init command. This requires a folder to initialize.

This will put a copy of everything in math4610 in a directory on your computer. Note that in the figure below a different repository was used. The math4610 repository is too large and takes a bit of time to clone.

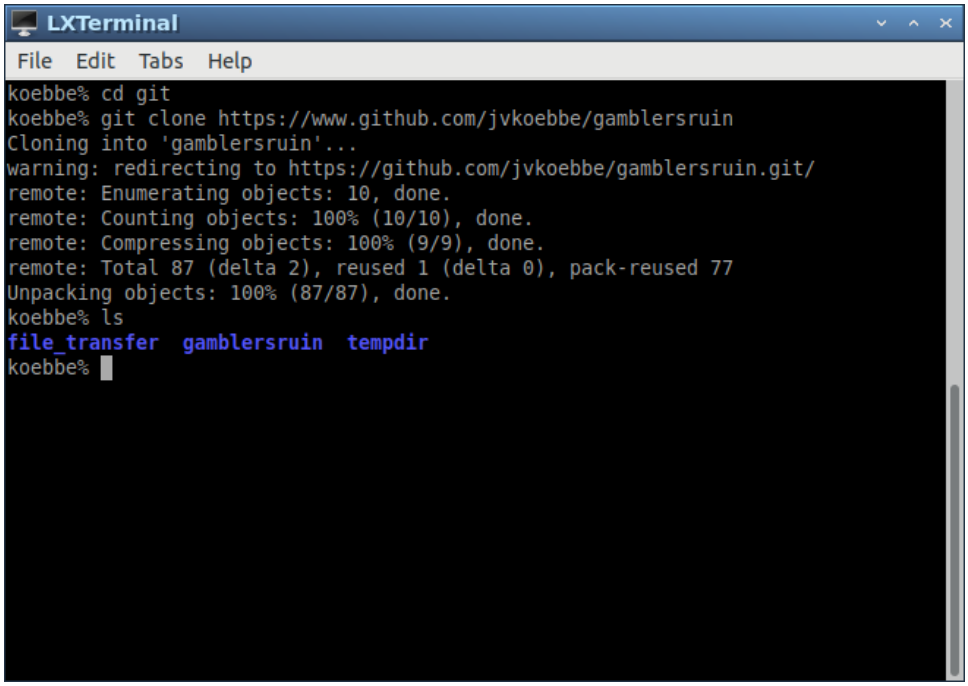

Figure 5: How to create a local repository using the git clone command. The output of the command indicates that a folder was created. In this case the name is gambelersruin.

Using the git status Command

One of the most used commands in git is the status command. This will tell you about any changes that have been made by you between sessions. For example,

```
koebbe% git status
```

You will use this command over and over if you are being efficient. An example of the output is shown in the next figure when working in repository. Note that the git status command is embedded in the work shown in the figure below.

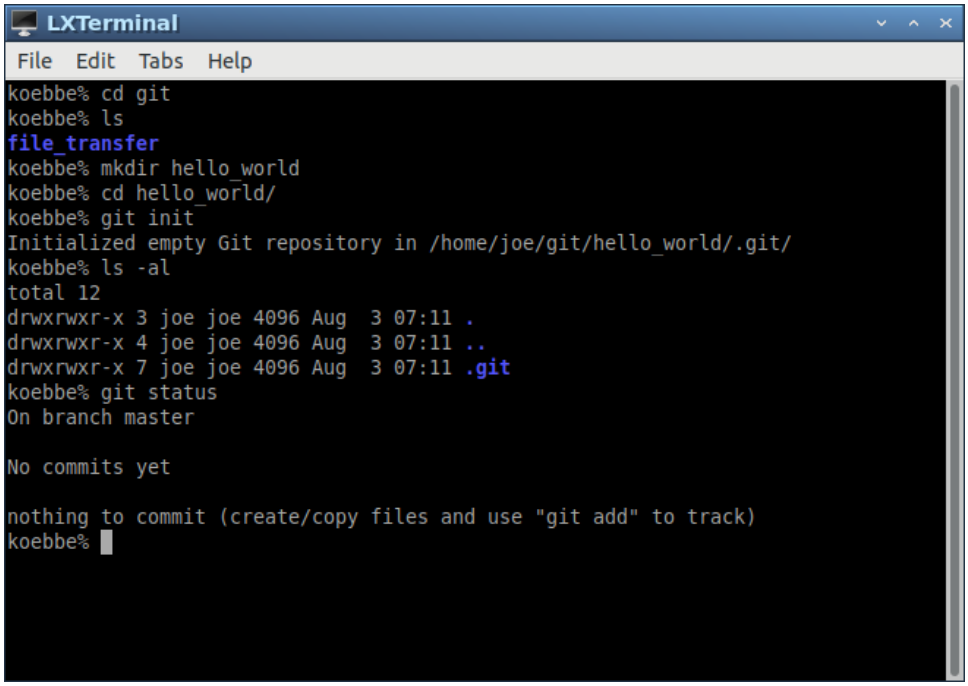

Figure 6: Using the git status command to List Changes.

The git commit Command

Once the status command has been invoked, we will want to commit the changes so that they can be copied back to our main working project. If files have been added or removed, the output from the status command will produce a message to that effect. So, we will need to add/remove the files to this branch of the repository. The following figure shows. this. Note that the last command will commit the changes. We will see the results of the commit command below.

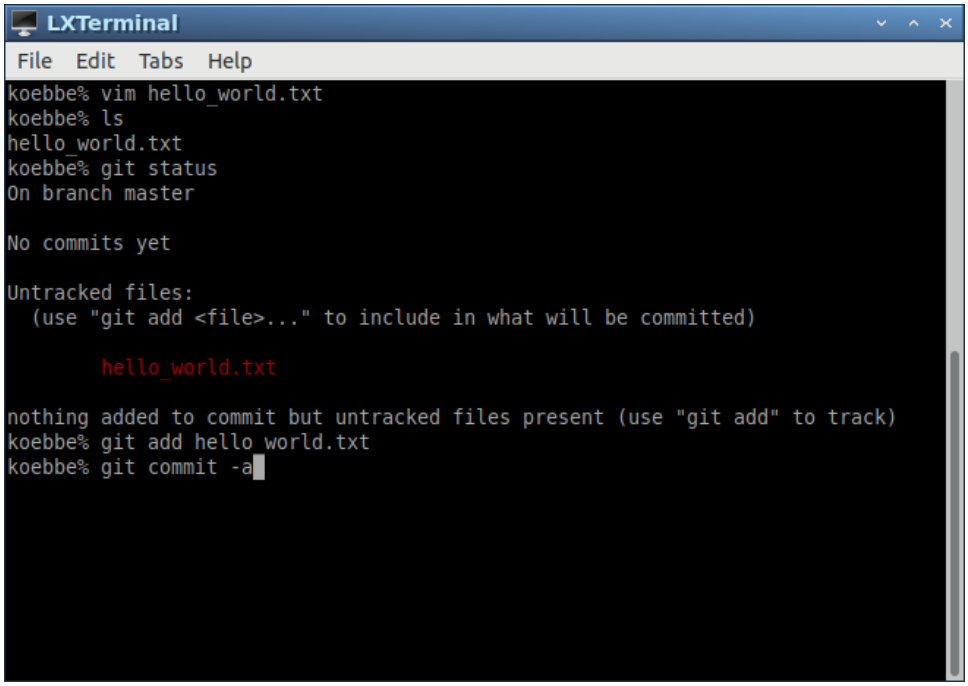

Figure 7: Using git status to identify changes and then add a file named hello_world.txt to the repository. Output from the git add command is shown in the following factor.

The git push Command

In the last figure, the git commit command has been typed in to complete the status change. The command is

```
koebbe% git commit -a
```

The output from the command will be inside an editor. You will need to enter a comment to document the changes that have been made. If no comment is added the commit command will be aborted. All you need is to add a single line to the file that pops up for the commit to take place.

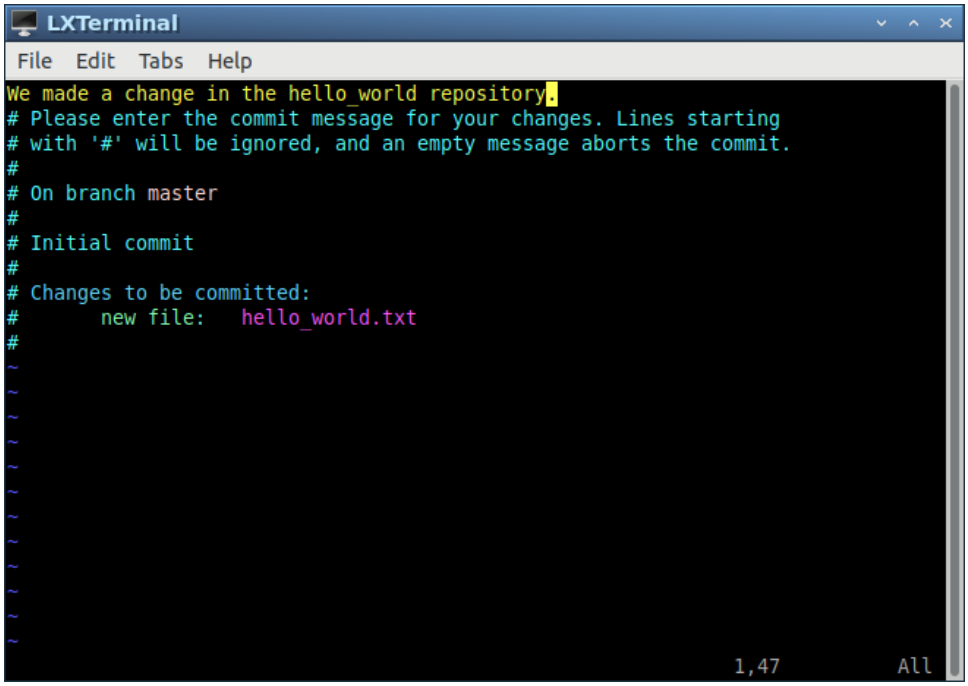

Figure 8: In this figure an editor window is displayed where a comment about the changes must be included. Usually, the comment should indicate what was changed and how. If no change is made to the file, the commit will be aborted

---

Running the Git Status Command Again

Using the git status again will show that everything is up to date. So, the message we see indicates the Master Branch has nothing to commit. We can see this in the next figure.

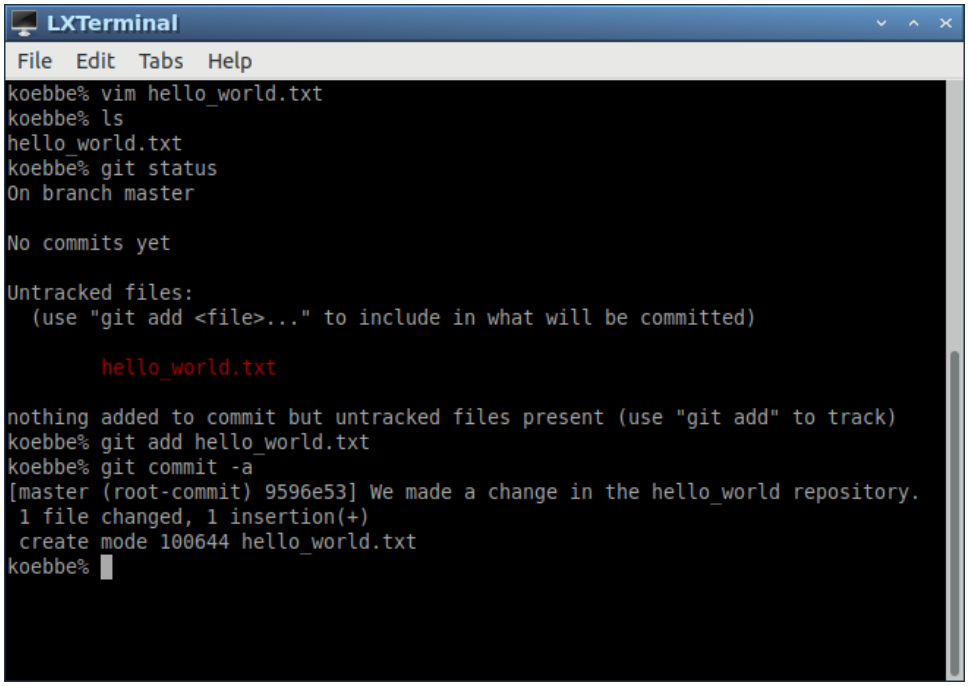

Figure 9: Once done editing the comment file, control is returned to the terminal.

Using The README.md File As A Starting Point

When you access your repository from the outside world, a browser will actually find the README.md file. So, when a repository is created, it is a good idea to create and modify this file with a basic description for the repository. An example is given in the next two figures that show how to change into the main folder for a repository on your computer and then edit the README.md file.

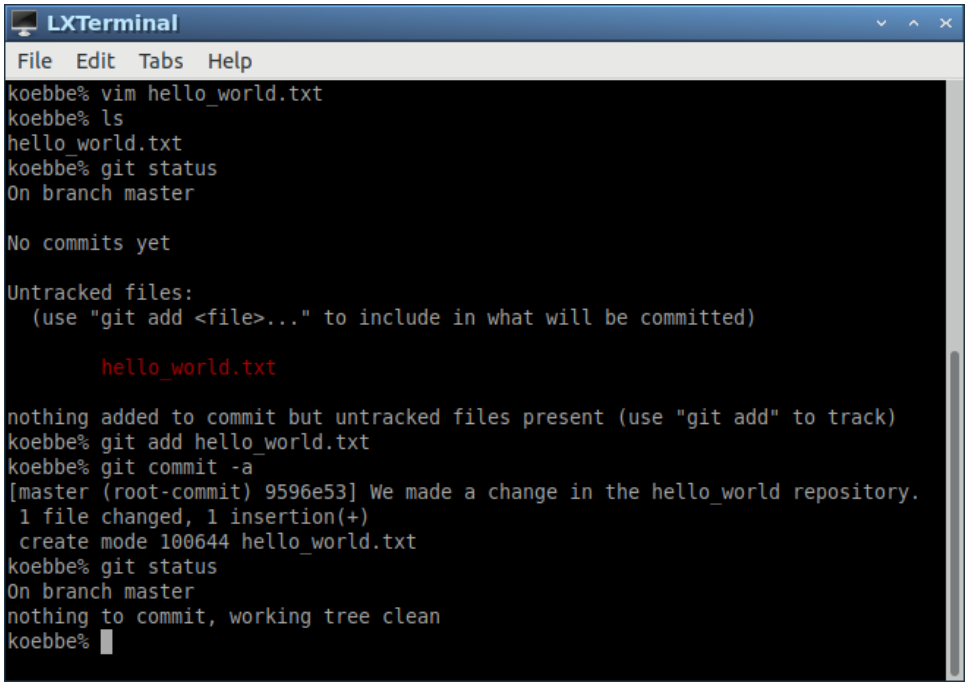

Figure 10: Using the git status command to see that no more commits need to be done.

Modification of the README file

In this and the next section of this topic, we will see how to include the modifications on a repository on Github. The next two figures show how to check the status of the repository, include modifications to the repository, and then push those back to the main repository on Github.

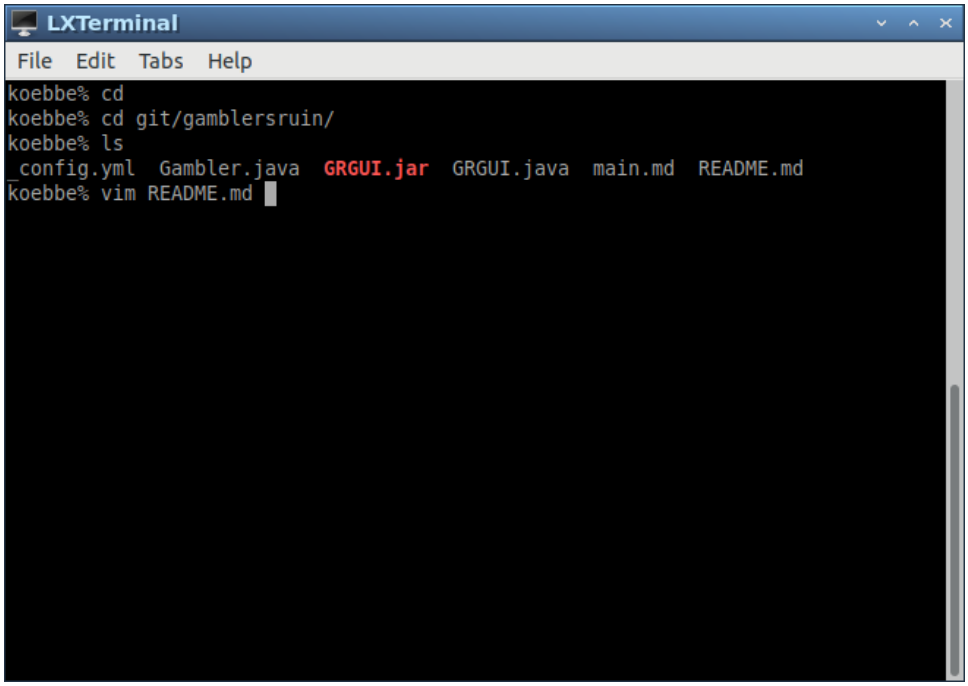

Figure 11: Changing into a main folder for a repository and then....

Using Git Push to Merge Changes

The next step is to merge any changes locally with the repository on Github. The command central to doing this is is

```
koebbe% git push
```

You will need to enter your Github user id and password to be able to complete the command. The next figures use the following git commands to do the work.

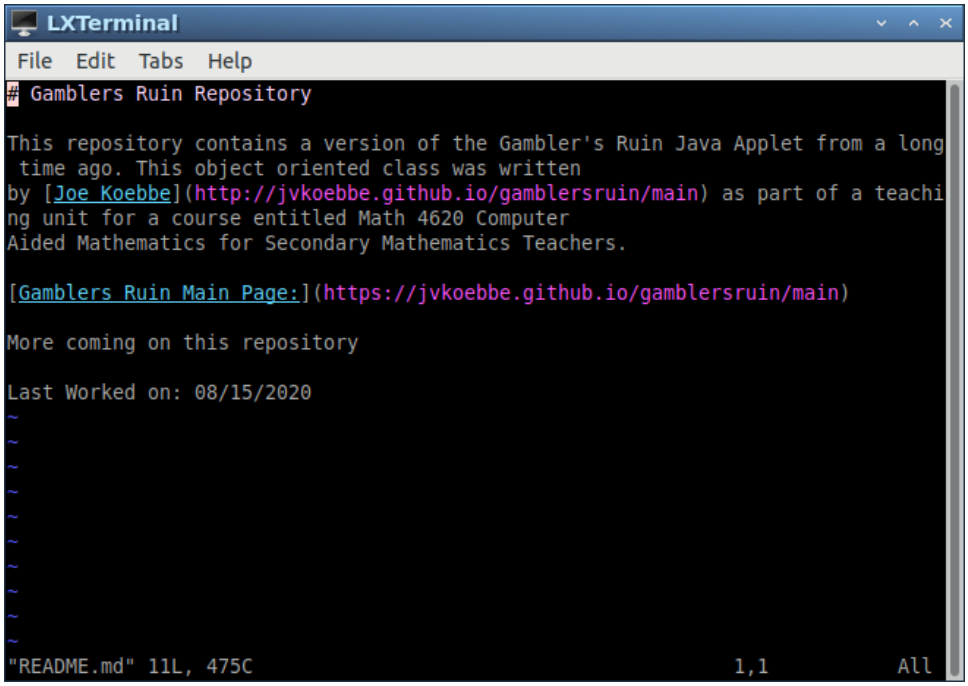

Figure 12: ...editing the README.md file to include basic details about the contents of the repository.

...And One Last Check of Status

Finally, we check the status of the repository which shows everything is clean.

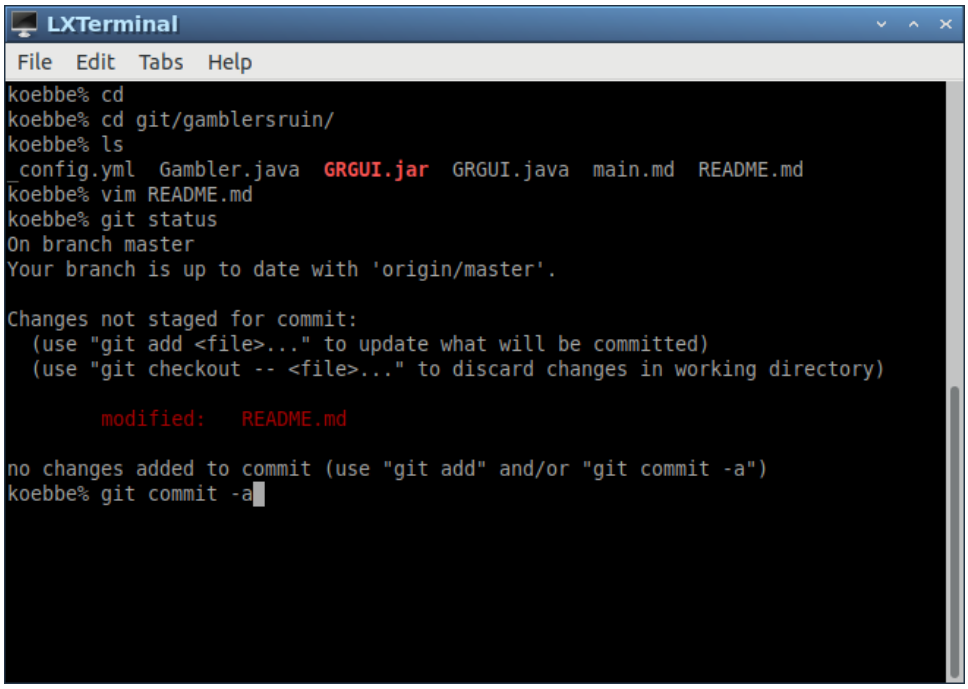

Figure 13: Changes to the README.md file that git status sees and....

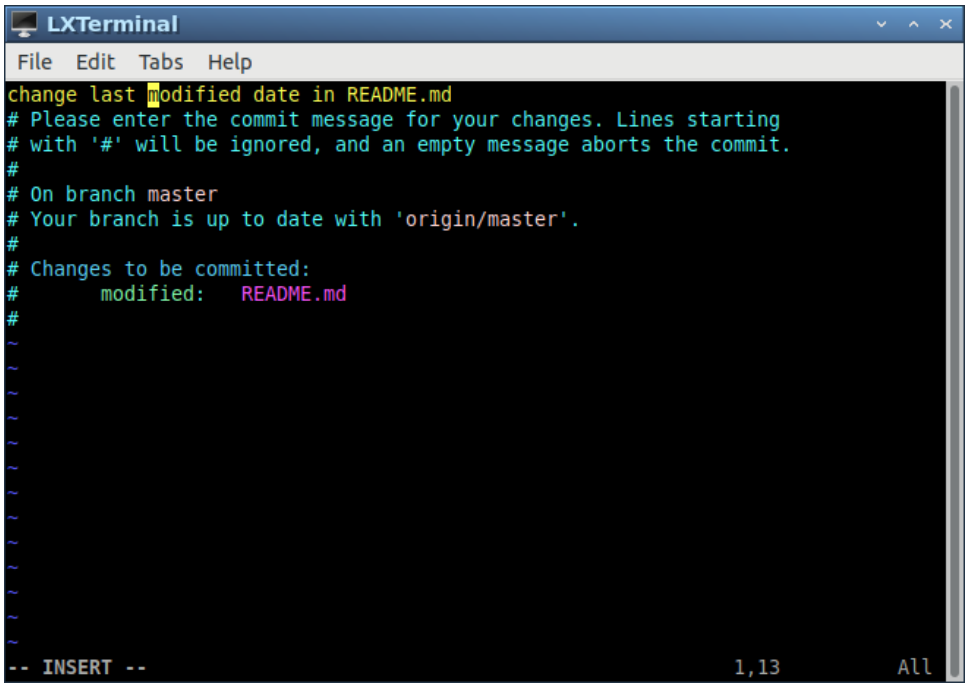

Figure 14: ... followed by a comment entered into the documentation file.

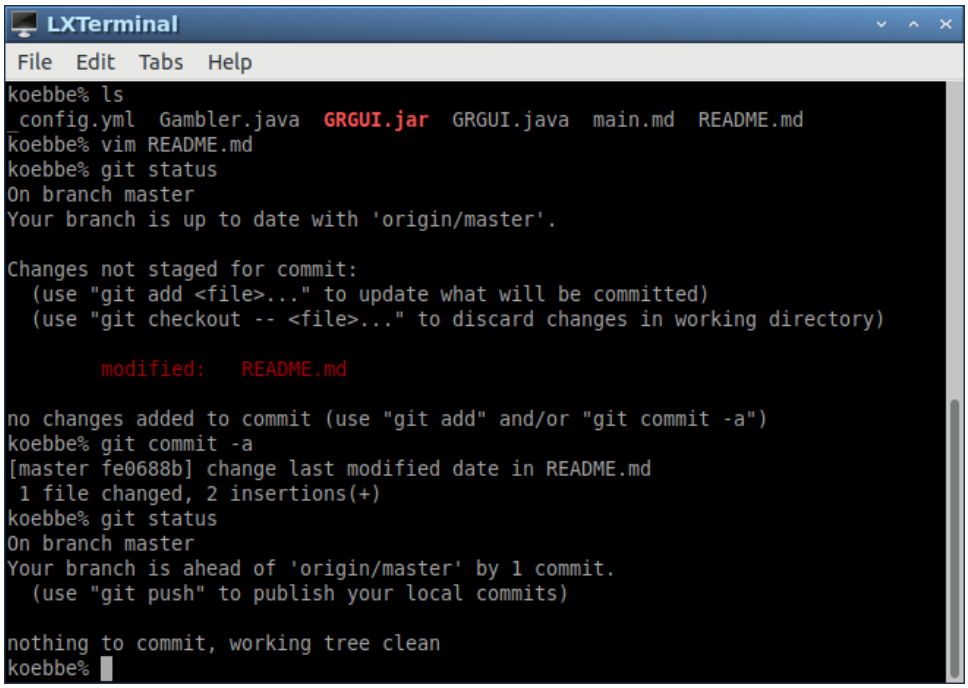

Figure 15: Using git status to identify the file that has been modified and to commit the change locally followed by a message to indicate the next step....

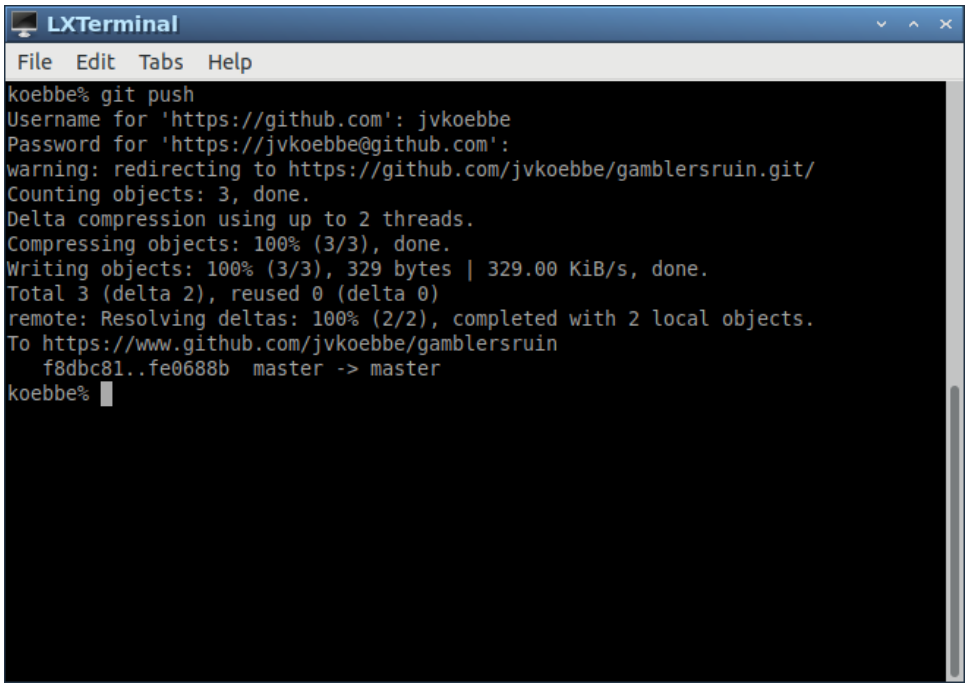

Figure 16: ... which is to use the git push command.

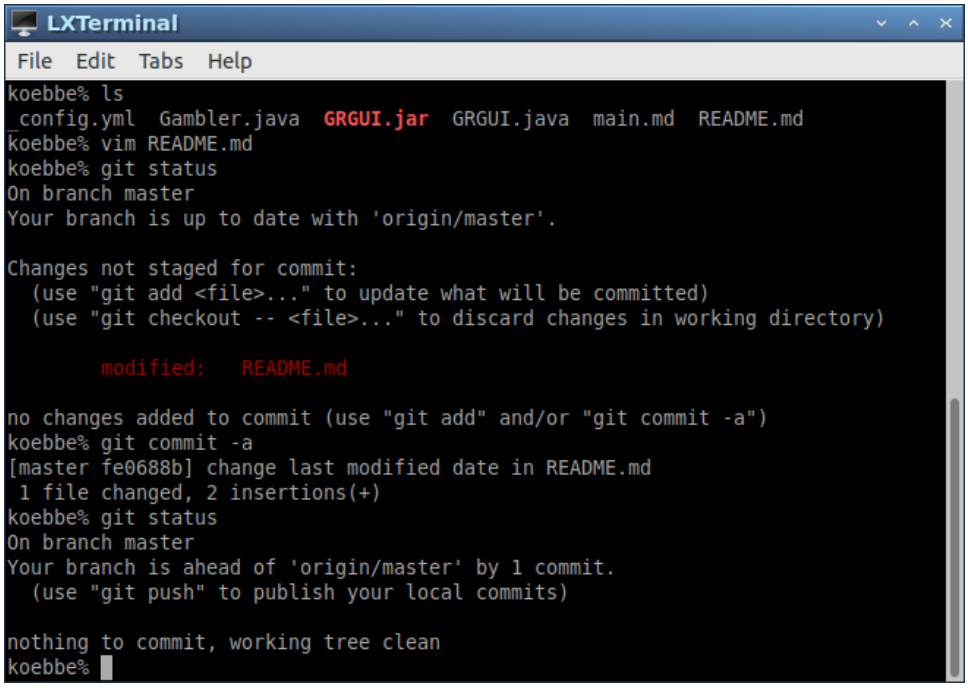

Figure 17: ... one last status check to complete the whole process.