
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.

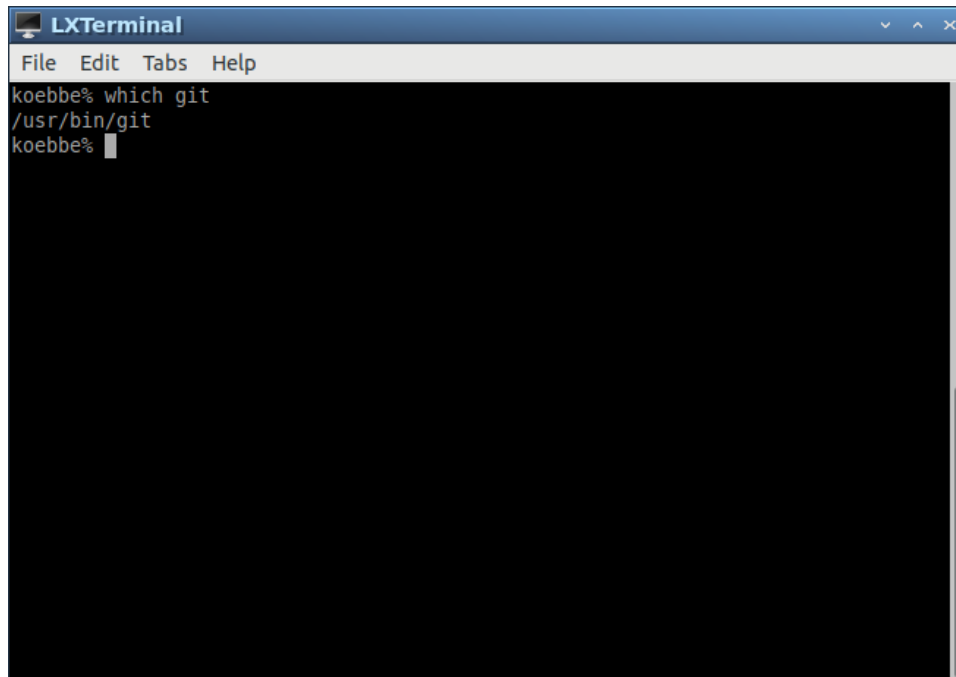
Using git to Work Locally on Your Computer

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 named, /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.

A screenshot of an LXTerminal window. The window has a title bar with 'LXTerminal' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal area shows the command 'koebbe% which git' being entered, followed by the output '/usr/bin/git' on the next line. The prompt 'koebbe%' is visible again on the third line, indicating the command has finished executing.

```
LXTerminal
File Edit Tabs Help
koebbe% which git
/usr/bin/git
koebbe%
```

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

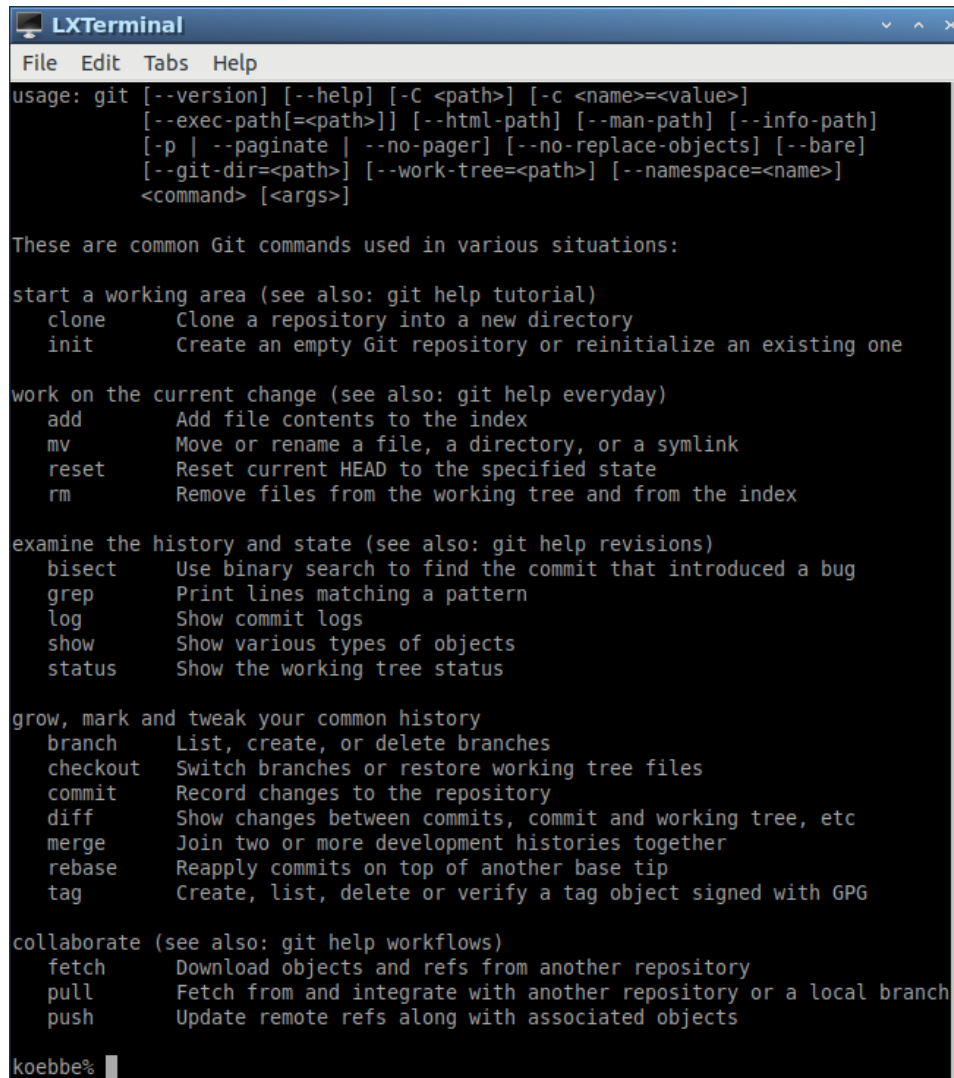
```
koebbe% git --help
```

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 it 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.



```
LXTerminal
File Edit Tabs Help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one


work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset      Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index


examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status


grow, mark and tweak your common history
  branch     List, create, or delete branches
  checkout   Switch branches or restore working tree files
  commit     Record changes to the repository
  diff       Show changes between commits, commit and working tree, etc
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  tag        Create, list, delete or verify a tag object signed with GPG


collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

koebbe%
```

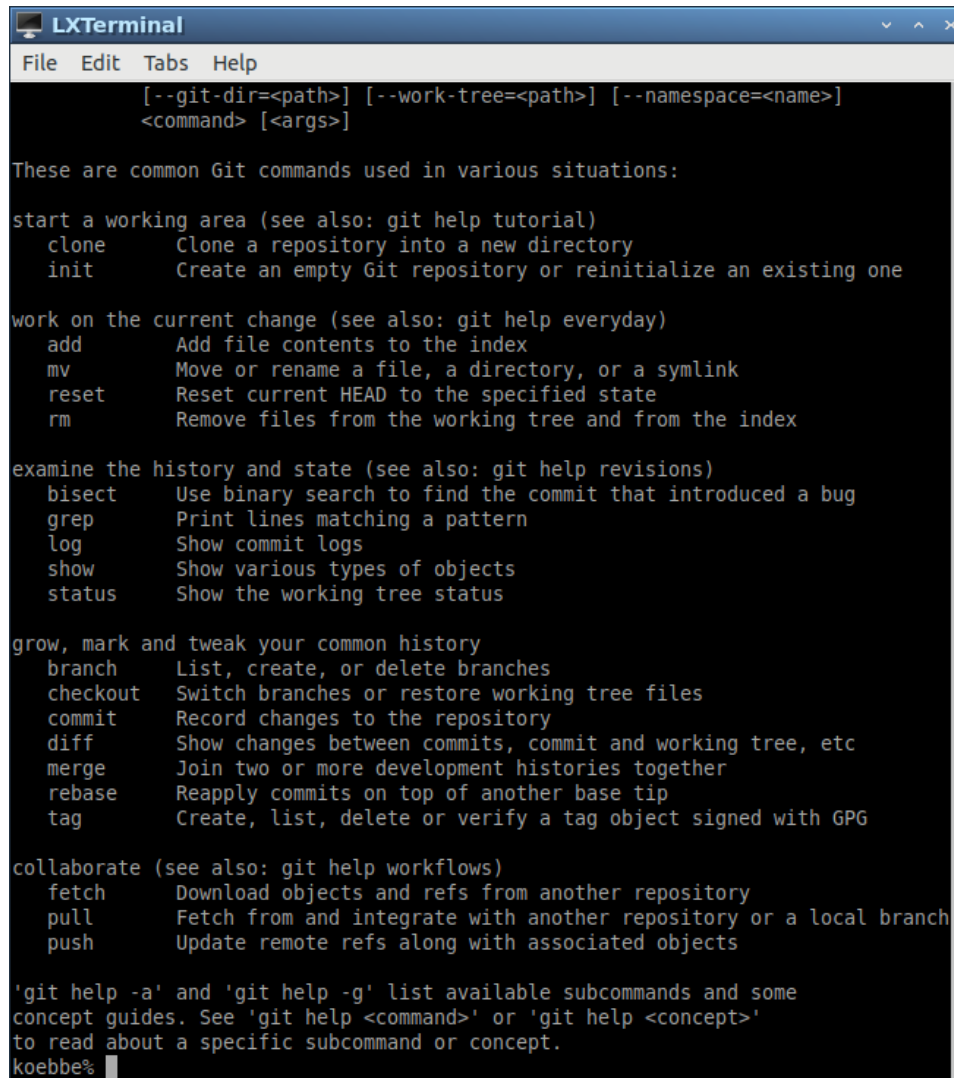
Figure 2: The first part of the output gives about half of the possible options we can use with git.

Initializing a Folder Using the git init

The git command platform is used to create folders that mirror repositories like those created on Github. 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
```

The first command makes a directory which will be initialized to a git folder. The next command changes the working directory to be the new directory, and the last does the initialization. As the output shows, the result is an empty git repository. The other method will work well when making a copy of a Github repository that already exists. You can clone a repository from another site, like Github. To do this, change into the folder



```
LXTerminal
File Edit Tabs Help

[ -git-dir=<path>] [ -work-tree=<path>] [ --namespace=<name>]
<command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset      Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

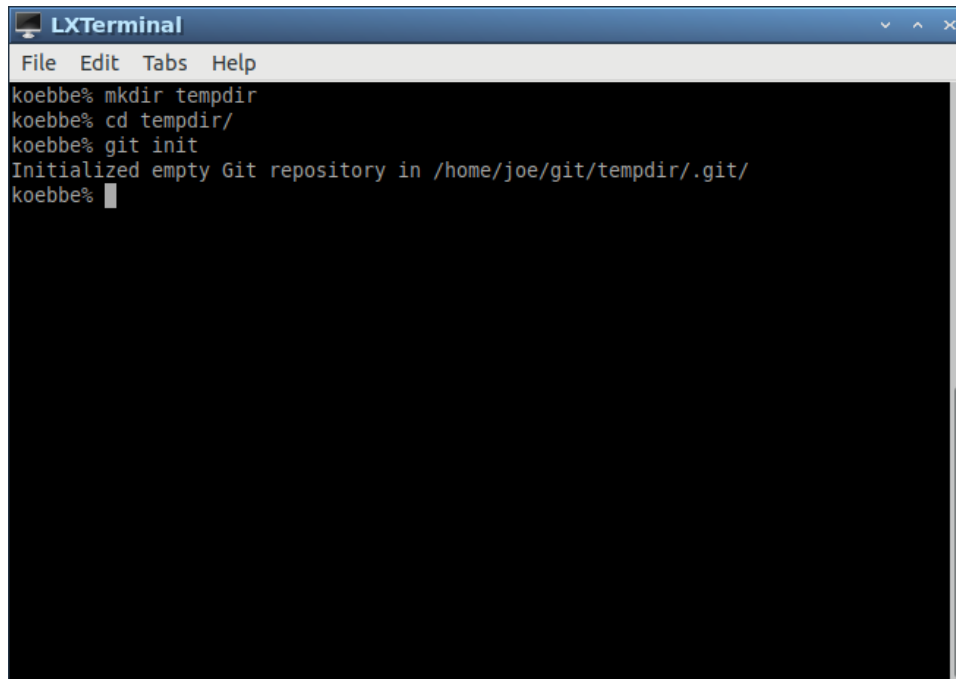
grow, mark and tweak your common history
  branch     List, create, or delete branches
  checkout   Switch branches or restore working tree files
  commit     Record changes to the repository
  diff       Show changes between commits, commit and working tree, etc
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
koebbe%
```

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

where the repository folder will end up.

A screenshot of an LXTerminal window. The window has a title bar with 'LXTerminal' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal area shows a series of commands and their output: 'koebbe% mkdir tempdir', 'koebbe% cd tempdir/', 'koebbe% git init', and 'Initialized empty Git repository in /home/joe/git/tempdir/.git/'. The prompt 'koebbe%' is followed by a cursor.

```
LXTerminal
File Edit Tabs Help
koebbe% mkdir tempdir
koebbe% cd tempdir/
koebbe% git init
Initialized empty Git repository in /home/joe/git/tempdir/.git/
koebbe%
```

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

Initializing a Folder Using the git clone

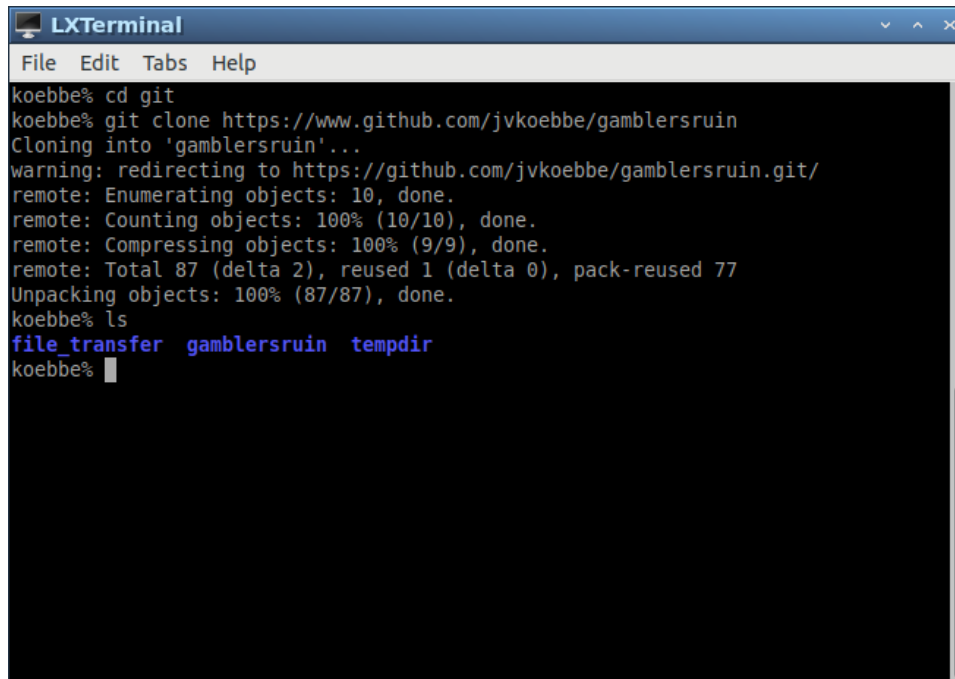
We can use the clone command in git to make a copy of a repository on Github. To do this, start by changing folders to the location where you want a copy of the repository to be cloned. This can be done as follows.

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

for an existing folder. 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
```

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

A screenshot of an LXTerminal window. The window has a title bar with 'LXTerminal' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal area shows a series of commands and their outputs. The user enters 'cd git', then 'git clone https://www.github.com/jvkoebbe/gamblersruin'. The output shows the cloning process, including a warning about redirecting to the correct repository URL, enumerating and counting objects, and compressing them. After the clone is complete, the user enters 'ls', and the output shows 'file transfer gamblersruin tempdir'. The prompt 'koebbe%' is visible at the end of each line.

```
koebbe% cd git
koebbe% git clone https://www.github.com/jvkoebbe/gamblersruin
Cloning into 'gamblersruin'...
warning: redirecting to https://github.com/jvkoebbe/gamblersruin.git/
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 87 (delta 2), reused 1 (delta 0), pack-reused 77
Unpacking objects: 100% (87/87), done.
koebbe% ls
file transfer gamblersruin tempdir
koebbe%
```

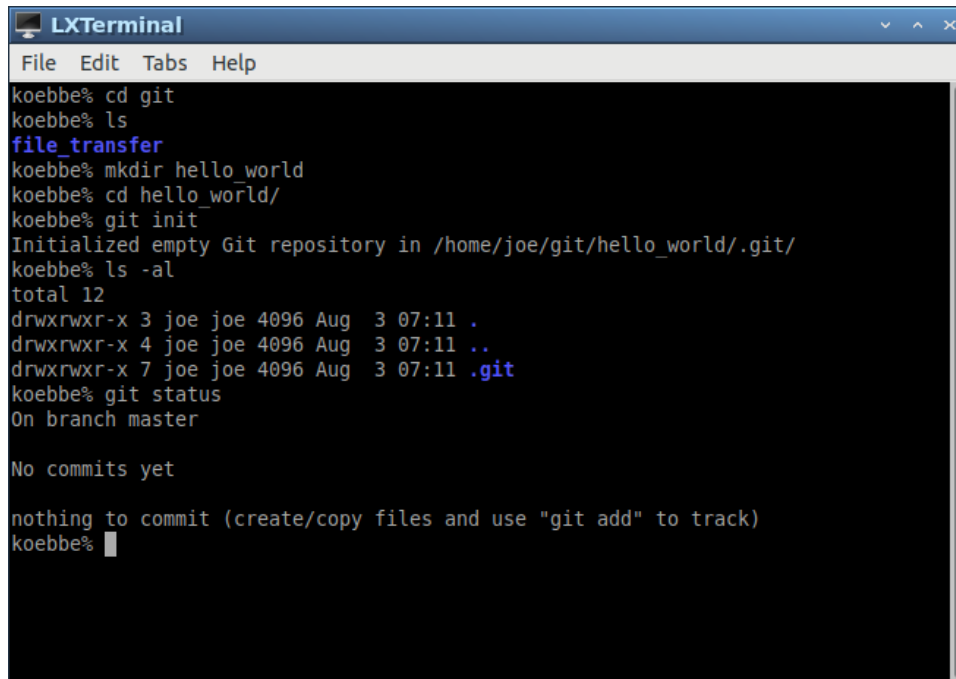
Figure 5: How to create a local repository using the git clone command. The output of the command indicates that a folder was created. In this case the name is gambelersruin.

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
```

will produce a list of files and folders to which changes have been made. Note that the git commands can be run from the main folder or any subfolder within the git folder tree structure. 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.

A screenshot of an LXTerminal window. The window has a title bar with 'LXTerminal' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows a series of commands and their outputs. The user 'koebbe' navigates to a 'git' directory, lists files, creates a 'hello_world' directory, and enters it. They then initialize a Git repository. The 'ls -al' command shows the directory structure, including the newly created '.git' directory. Finally, the 'git status' command is run, showing the current branch as 'master' and indicating that there are no commits yet and nothing to commit.

```
LXTerminal
File Edit Tabs Help
koebbe% cd git
koebbe% ls
file_transfer
koebbe% mkdir hello_world
koebbe% cd hello_world/
koebbe% git init
Initialized empty Git repository in /home/joe/git/hello_world/.git/
koebbe% ls -al
total 12
drwxrwxr-x 3 joe joe 4096 Aug  3 07:11 .
drwxrwxr-x 4 joe joe 4096 Aug  3 07:11 ..
drwxrwxr-x 7 joe joe 4096 Aug  3 07:11 .git
koebbe% git status
On branch master

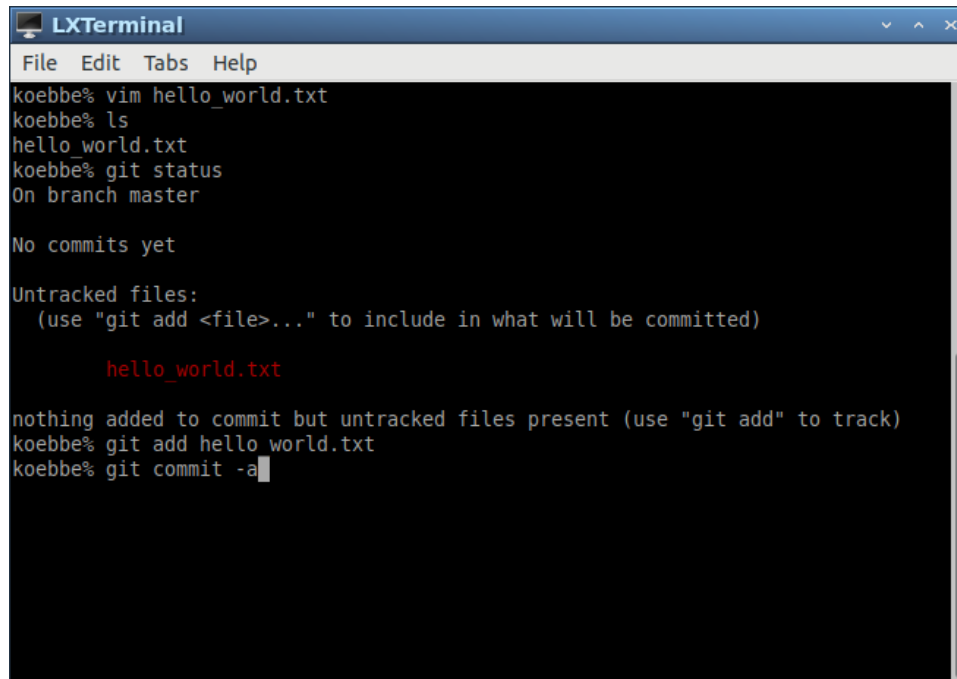
No commits yet

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

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.

A screenshot of an LXTerminal window. The window has a title bar with 'LXTerminal' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows a series of commands and their outputs. The user starts by creating a file 'hello_world.txt' with 'vim', then lists files with 'ls', and checks the Git status with 'git status'. The status output shows 'On branch master' and 'No commits yet'. It lists 'Untracked files:' as 'hello_world.txt' in red text. The user then adds the file to the repository with 'git add hello_world.txt' and commits it with 'git commit -a'.

```
LXTerminal
File Edit Tabs Help
koebbe% vim hello_world.txt
koebbe% ls
hello_world.txt
koebbe% git status
On branch master

No commits yet

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

        hello_world.txt

nothing added to commit but untracked files present (use "git add" to track)
koebbe% git add hello_world.txt
koebbe% git commit -a
```

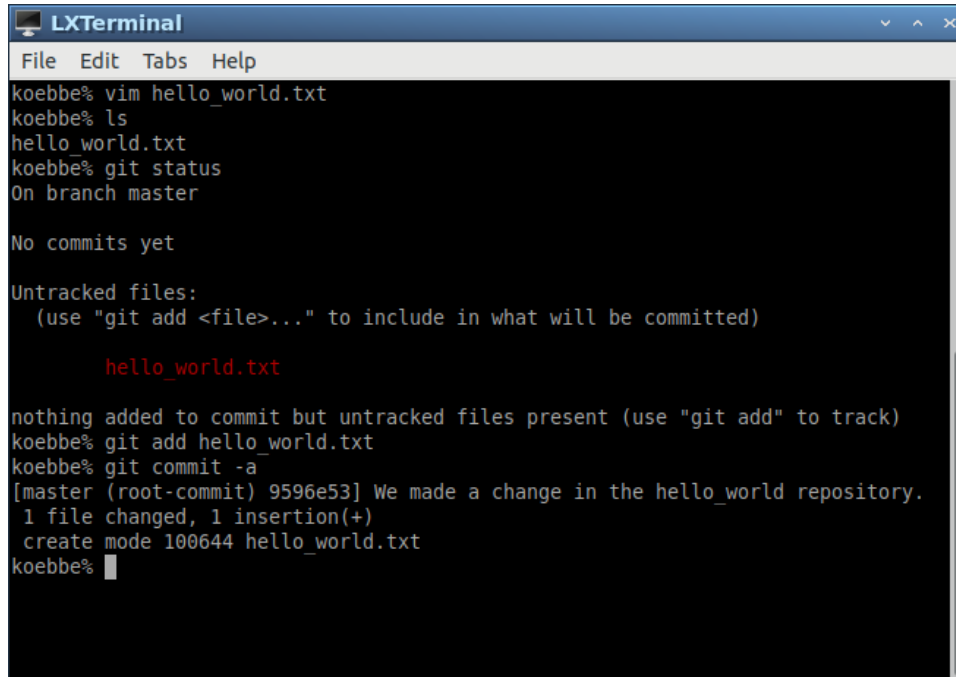
Figure 7: Using git status to identify changes and then add a file named hello_world.txt to the repository. Output from the git add command is shown in the following factor.

The git push Command

In the last figure, the git commit command has been typed in to complete the changes identified by the git status command. 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 (or just one character) to the file that pops up for the commit to take place.



```
LXTerminal
File Edit Tabs Help
koebbe% vim hello_world.txt
koebbe% ls
hello_world.txt
koebbe% git status
On branch master

No commits yet

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

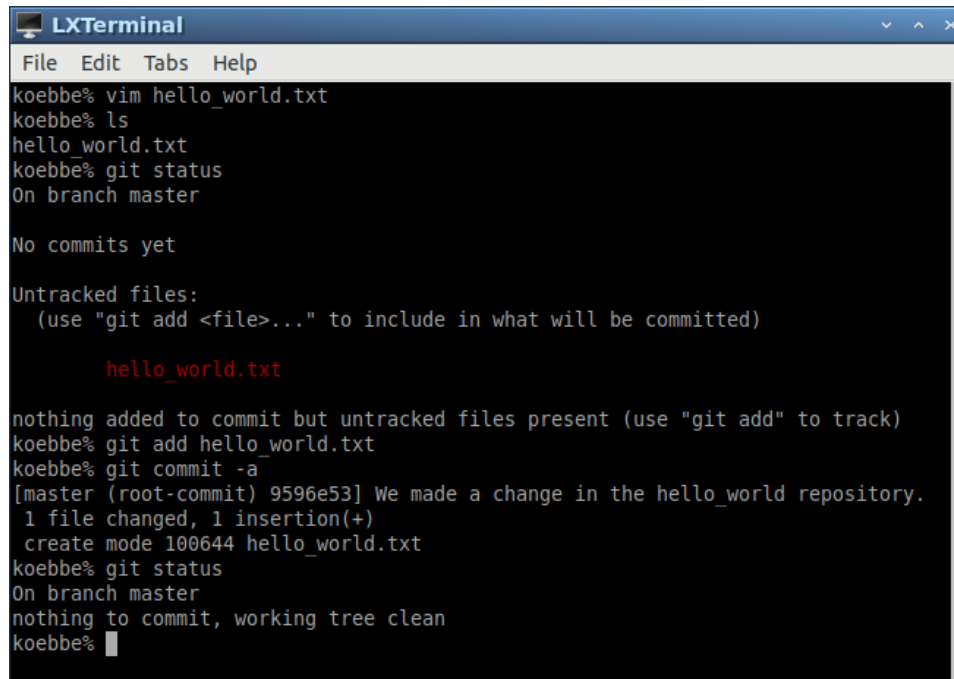
        hello_world.txt

nothing added to commit but untracked files present (use "git add" to track)
koebbe% git add hello_world.txt
koebbe% git commit -a
[master (root-commit) 9596e53] We made a change in the hello_world repository.
 1 file changed, 1 insertion(+)
 create mode 100644 hello_world.txt
koebbe%
```

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.

A screenshot of an LXTerminal window. The window has a title bar with 'LXTerminal' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows a series of commands and their outputs. The user starts by running 'vim hello_world.txt', then 'ls', which shows 'hello_world.txt'. Then 'git status' is run, showing 'On branch master' and 'No commits yet'. It lists 'Untracked files: hello_world.txt' with a hint to use 'git add'. The user then runs 'git add hello_world.txt' and 'git commit -a', which creates a new commit. Finally, 'git status' is run again, showing 'On branch master' and 'nothing to commit, working tree clean'.

```
LXTerminal
File Edit Tabs Help
koebbe% vim hello_world.txt
koebbe% ls
hello_world.txt
koebbe% git status
On branch master

No commits yet

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

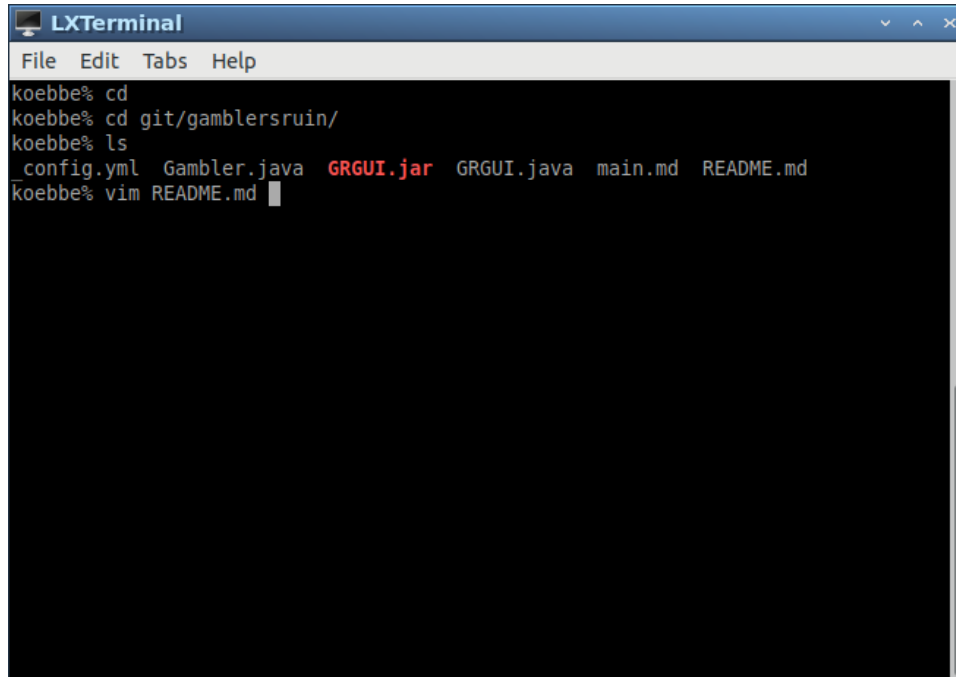
        hello_world.txt

nothing added to commit but untracked files present (use "git add" to track)
koebbe% git add hello_world.txt
koebbe% git commit -a
[master (root-commit) 9596e53] We made a change in the hello_world repository.
 1 file changed, 1 insertion(+)
 create mode 100644 hello_world.txt
koebbe% git status
On branch master
nothing to commit, working tree clean
koebbe% █
```

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

Modification of the README file

In the next sections 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.

A screenshot of an LXTerminal window. The title bar says "LXTerminal". The menu bar has "File", "Edit", "Tabs", and "Help". The terminal content shows a user named "koebbe" performing several commands: "cd", "cd git/gamblersruin/", "ls", and "vim README.md". The "ls" command output lists files: "_config.yml", "Gambler.java", "GRGUI.jar" (in red), "GRGUI.java", "main.md", and "README.md".

```
LXTerminal
File Edit Tabs Help
koebbe% cd
koebbe% cd git/gamblersruin/
koebbe% ls
_config.yml  Gambler.java  GRGUI.jar  GRGUI.java  main.md  README.md
koebbe% vim README.md
```

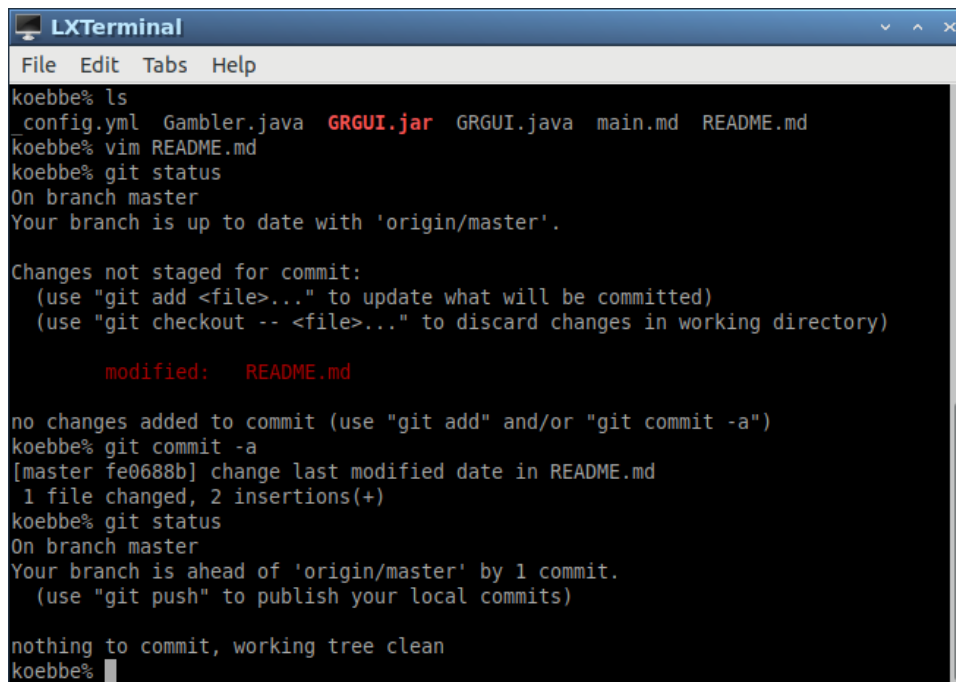
Figure 11: Changing into a main folder for a repository and then....

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

```
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.



```
LXTerminal
File Edit Tabs Help
koebbe% ls
config.yml Gambler.java GRGUI.jar GRGUI.java main.md README.md
koebbe% vim README.md
koebbe% git status
On branch master
Your branch is up to date with 'origin/master'.

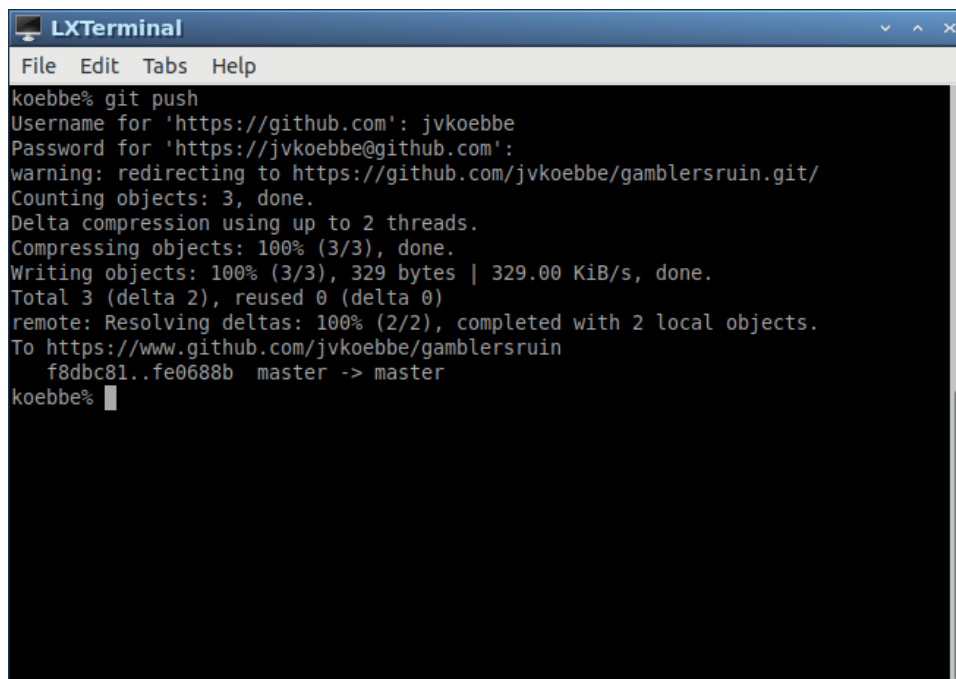
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
koebbe% git commit -a
[master fe0688b] change last modified date in README.md
1 file changed, 2 insertions(+)
koebbe% git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

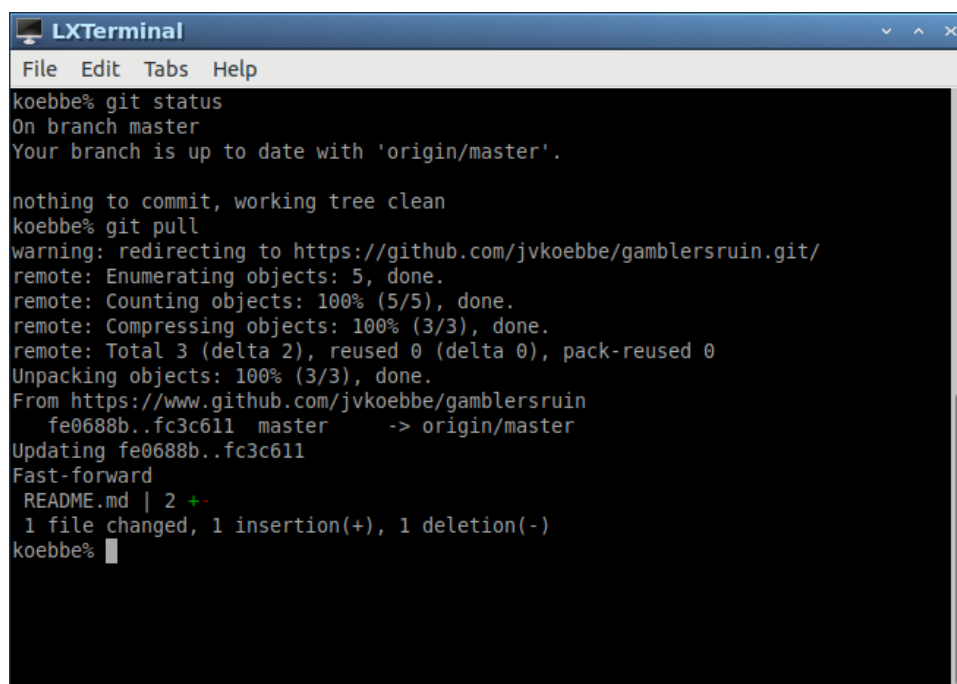
nothing to commit, working tree clean
koebbe%
```

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....



```
LXTerminal
File Edit Tabs Help
koebbe% git push
Username for 'https://github.com': jvkoebbe
Password for 'https://jvkoebbe@github.com':
warning: redirecting to https://github.com/jvkoebbe/gamblersruin.git/
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 329 bytes | 329.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://www.github.com/jvkoebbe/gamblersruin
   f8dbc81..fe0688b  master -> master
koebbe%
```

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



```
LXTerminal
File Edit Tabs Help
koebbe% git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
koebbe% git pull
warning: redirecting to https://github.com/jvkoebbe/gamblersruin.git/
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://www.github.com/jvkoebbe/gamblersruin
   fe0688b..fc3c611  master    -> origin/master
Updating fe0688b..fc3c611
Fast-forward
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
koebbe%
```

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