

## 1. Version Control

Version control makes collaborating with other developers infinitely easier. It is a system that tracks all the changes a user has made to their files. It provides a complete log of all user changes so that a user can hunt for bugs easier, revert changes and if a user revisits a project a few months down the line, it'll be easier to pick up where a user left off because a user documented their own thought process.

Let's say a user has a file that they want to make a change to, but they do not want to overwrite the original in case it does not work out. Chances are, a user will make a copy of that file and name it something like `project-backup.html` or `project-original.html`. It gets out of hand very quickly if a user wishes to preserve multiple versions of that file. Version control eliminates the need for all that. A user has one set of files and all their changes are tracked separately.

One of the best version control software online is GitHub. GitHub is a web-based hosting service for all things Git. It is a repository for users to upload their projects to. It is a great place for developers to store all of their projects and access other users' projects. Its applications include web hosting service and social media network.

When multiple people work on a project at the same time, they can branch off the main project with their own versions full of changes they themselves have made. After they are done, it is time to merge that branch back with the master, the main directory of the project.

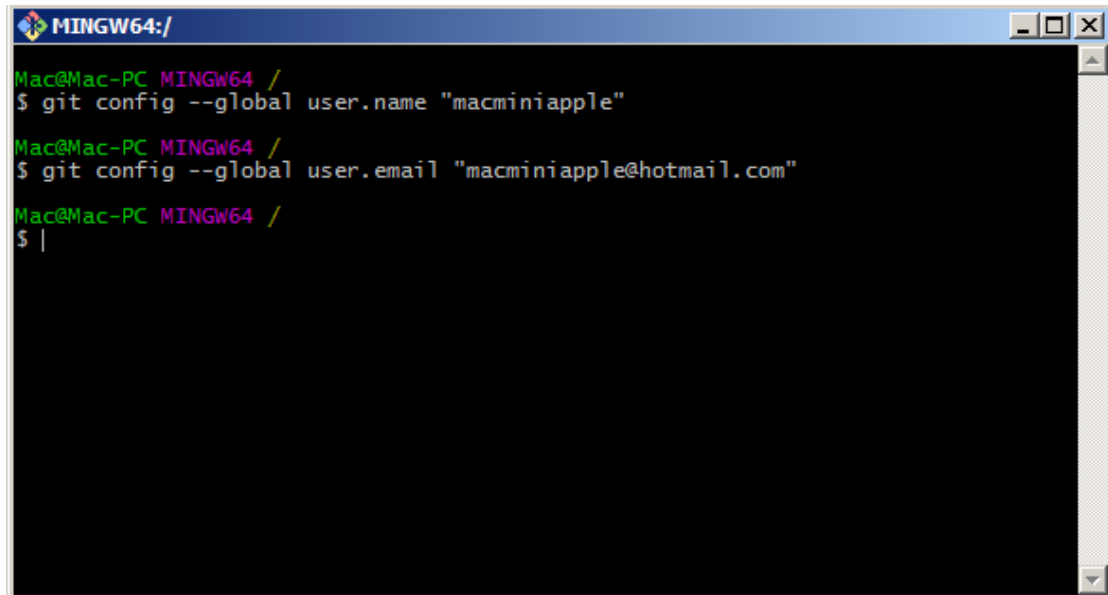
## 2. Basics of Git-Specific Commands

1. `git checkout`: Literally allows user to “check out” a repository that user are not currently inside. This is a navigational command that lets user move to the repository user want to check. User can use this command as `git checkout master` to look at the master branch, or `git checkout cats` to look at another branch.
2. `git branch`: This command will let user build a new branch, or timeline of commits, of changes and file additions that are completely user own. User title goes after the command. Example `git branch cats`, cats is the title.
3. `git merge`: When user are done working on a branch, user can merge their changes back to the master branch, which is visible to all collaborators. `git merge cats` would take all the changes user made to the “cats” branch and add them to the master.
4. `git commit`: Git most important command. After user make any sort of change, user input this in order to take a “snapshot” of the repository. Usually it goes `git commit -m “Message here.”` The `-m` indicates that following section of the command should be read as a message.
5. `git push`: If user is working on their local computer, and want their commits to be visible online on GitHub as well, their “push” the changes up to GitHub with this command.
6. `git pull`: If user is working on their local computer and want the most up-to-date version of their repository to work with, user “pull” the changes down from GitHub with this command.

7. `git init`: Initializes a new Git repository. User run this command inside a repository or directory, it is just a regular folder. Only after user input this does it accept further Git commands.
8. `git config`: Short for “configure” , this is most usefull when user is setting up Git for the first time.
9. `git help`: Type this into the command line to bring up the most command git commands. User can also be more specific and type “git help init” or another term to figure out how to use and configure a specific git command.
10. `git status`: check the status of user repository. See which files are inside it, which changes still need to be committed, and which branch of the repository user is currently working on.
11. `git add`: This does not add new files to user repository. Instead, it bring new files to Git attention. After user add files, they included in Git “snapshots” of the repository.

### 3. Get Start with Github Tutorial

First open Git Shell or Git Bash, in this tutorial will using Git Bash. Open in, then need to setup a global user name and email like image 3.1 below.



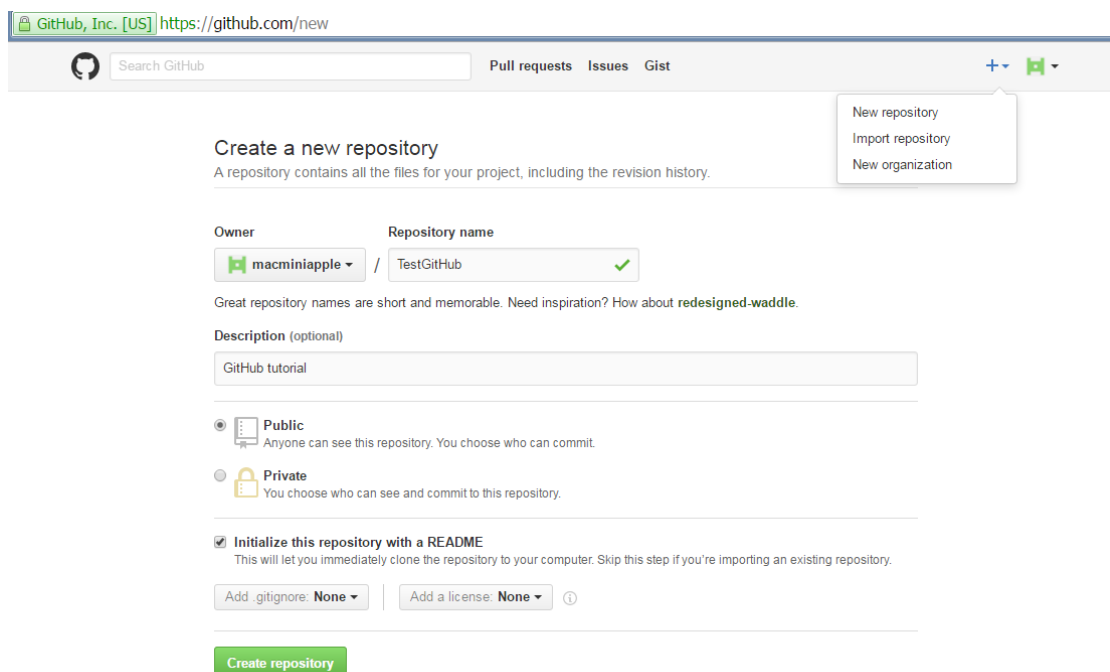
```
Mac@Mac-PC MINGW64 /
$ git config --global user.name "macminiapple"

Mac@Mac-PC MINGW64 /
$ git config --global user.email "macminiapple@hotmail.com"

Mac@Mac-PC MINGW64 /
$ |
```

Image 3.1

Next go to the github.com and login the username and password. After that at the top-right corner click on “+” and “New repository”. The form will come out ten fill everything.



GitHub, Inc. [US] <https://github.com/new>

Search GitHub Pull requests Issues Gist + -

**Create a new repository**  
A repository contains all the files for your project, including the revision history.

Owner: macminiapple / Repository name: TestGitHub ✓

Great repository names are short and memorable. Need inspiration? How about [redesigned-waddle](#).

Description (optional): GitHub tutorial

☒ Public  
Anyone can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

☒ Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None Add a license: None ⓘ

**Create repository**

Image 3.2

As image 3.3 make sure to copy the HTTPS link.

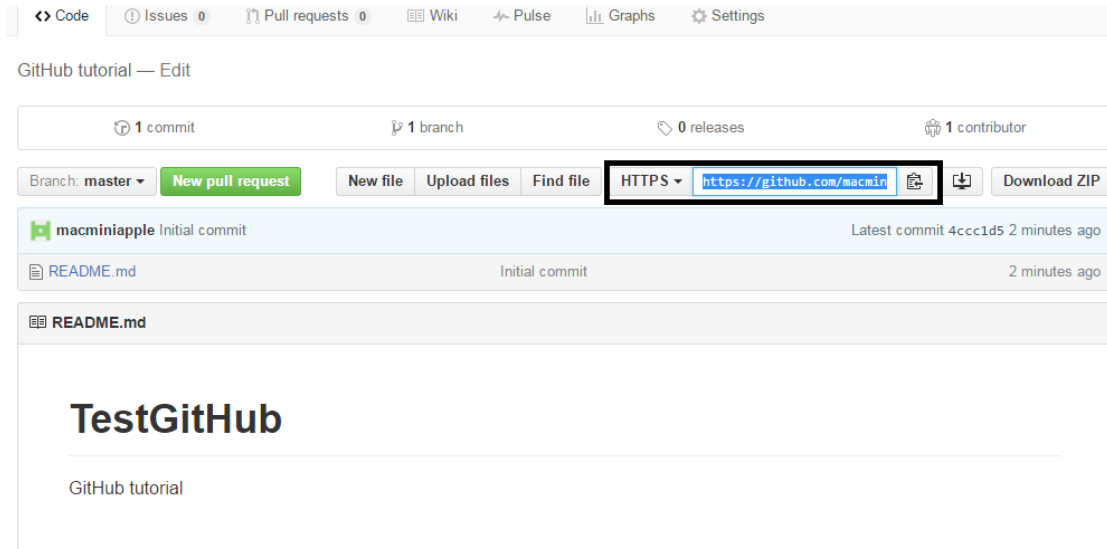


Image 3.3

Image 3.4 type in git clone and paste the HTTPS link. After that cd to TestGitHub.



Image 3.4

Let say if want to open the readme file and edit the contain say what is the program about for other users to view later on. Now can type “ls” to see what is in TestGitHub folder, then type vim or other like nano to access that file and edit like image 3.5.



The screenshot shows a Windows command prompt window with a title bar that reads "MINGW64:/D/TestGitHub". The window has a black background with white text. The prompt is "Mac@Mac-PC MINGW64 /D/TestGitHub (master)". The first command entered is "\$ ls", which outputs "README.md". The second command entered is "\$ vim README.md", which starts the Vim text editor. The Vim interface shows the filename "README.md" at the top, followed by a line of text "Hello World!". The cursor is positioned at the end of the line. The bottom status bar of the Vim window shows "1" and "1" (line and column numbers).

```
Mac@Mac-PC MINGW64 /D/TestGitHub (master)
$ ls
README.md
Mac@Mac-PC MINGW64 /D/TestGitHub (master)
$ vim README.md
```

Image 3.5

Image 3.6 show that interface was changed, now press “I” for insert mode. Now can typing any, after that press “ESC” to exit all mode and type “:x” for exit and save or “:q” for exit without save. But in this tutorial will use “:x” for exit and save.

[illegible]

Image 3.6

Now open the README.md file, it changed.

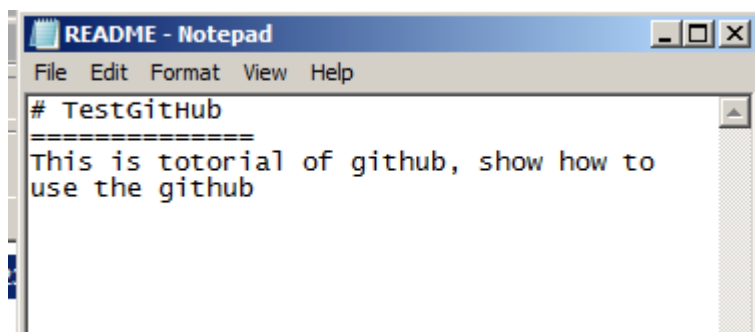


Image 3.7

Now type “git status” it will show what is modified.

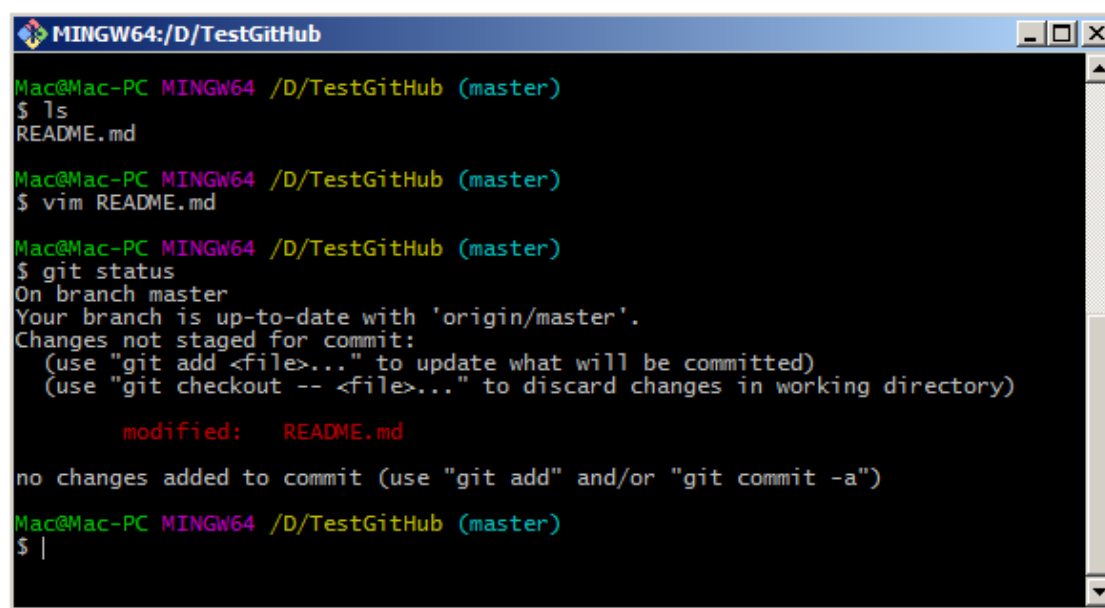


Image 3.8

Now README.md was change, it time to upload and share the file to other users. “git add README.md” to add that file ready to upload online.

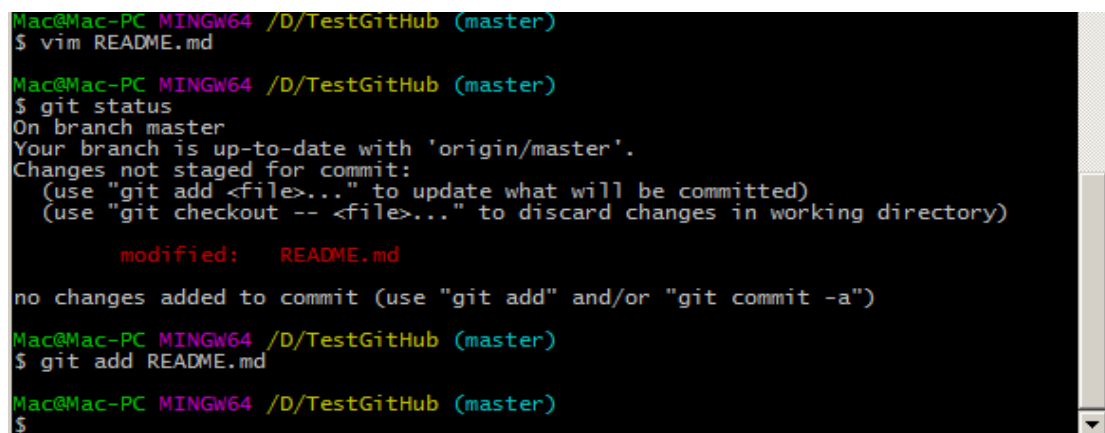
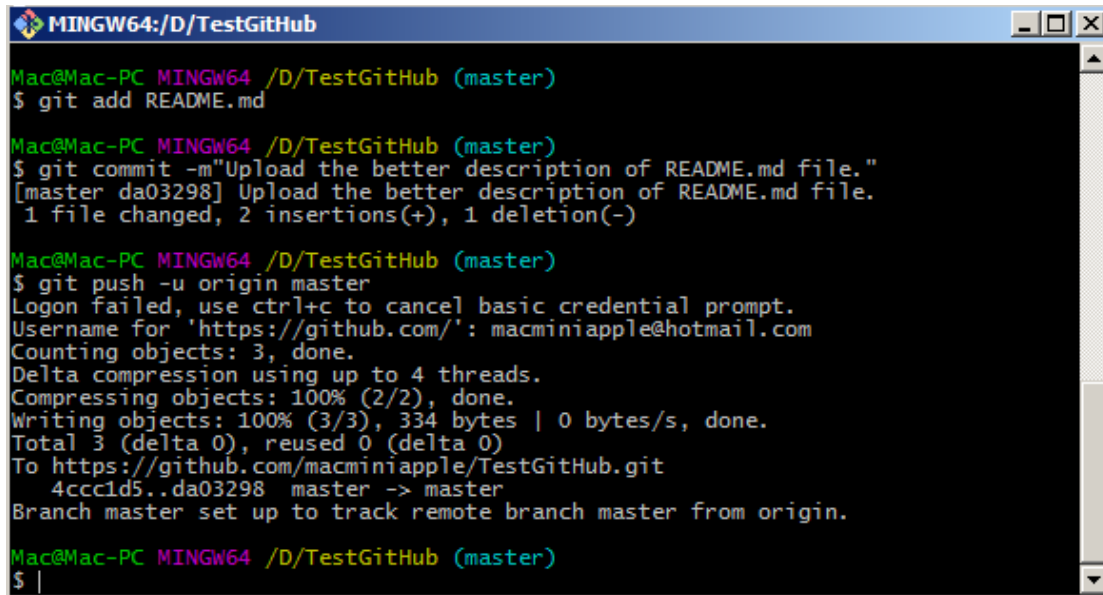


Image 3.9

It time to commit and upload type “git commit -m “Upload the better description of README.md file.”” and “git push -u origin master”. After that type in username and password.



```
MINGW64:/D/TestGitHub
Mac@Mac-PC MINGW64 /D/TestGitHub (master)
$ git add README.md

Mac@Mac-PC MINGW64 /D/TestGitHub (master)
$ git commit -m"Upload the better description of README.md file."
[master da03298] Upload the better description of README.md file.
1 file changed, 2 insertions(+), 1 deletion(-)

Mac@Mac-PC MINGW64 /D/TestGitHub (master)
$ git push -u origin master
Logon failed, use ctrl+c to cancel basic credential prompt.
Username for 'https://github.com/': macminiapple@hotmail.com
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 334 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/macminiapple/TestGitHub.git
4ccc1d5..da03298 master -> master
Branch master set up to track remote branch master from origin.

Mac@Mac-PC MINGW64 /D/TestGitHub (master)
$ |
```

Image 3.10

Now go github and check. Macminiapple just upload 2 minutes ago.

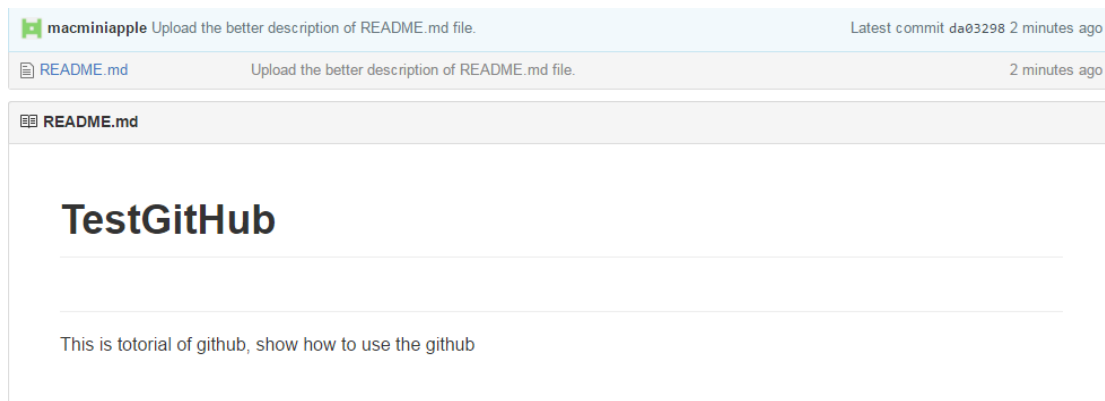


Image 3.11