5.1 Version Control System (Git)

Version control is a system that is based on the concept of tracking changes made to a file or a directory over time so that you are able to recall specific versions later on. One of the following version control system is Git whereby it is also one of the most commonly used system for software development and any version control tasks. Git is type of version control system where it emphasis on data integrity, speed and support for distributed, non-linear workflows. Similarly to other version control systems, every Git working directory is a repository that comes with a complete history and version-tracking capabilities.

The advantages of using a version control system such as Git are that the file names and directory structures that are consistent for all team members now, you can now make any changes you want at ease and retrieve or revert them back when required and it also provide a throughout understanding on who made a change, where and when it happened to avoid confusion among fellow team members. You would not have to be worry about accidentally overwriting some files and could not retrieve them back when that has been done.

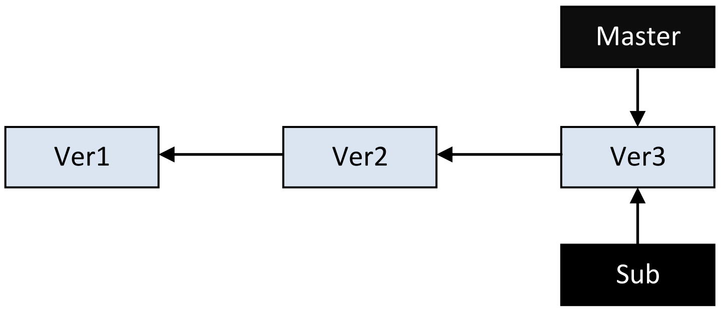
Pull, Add, Commit and Push

A detailed explanation on the basics steps while using Git are shown below:

1. When a user creates a repository, the user will be also known as the project leader/developer for the recently created repository.
2. Once a collaborator is recruited by the project leader into the project/repository, the collaborators will have granted access on the existing file in the repository. If the collaborators wish to edit the current files, they will need to first "**Clone**" the repository into their local computer whereby it creates a local repository for them to edit.
3. When the project leader/collaborators are working on the file and are done editing it, they first have to save it to their local computer. Then if they wish to share the file with other collaborators or update the file, they are required to "**Add**" the file to the staging index. However, this does not update or upload the file just yet.
4. To confirm the changes they wish to make in the local repository, they must first "**Commit**" the files which will only then update or upload the files into the local repository.
5. Even after committing the files, other collaborators are still not able to use the updated file just yet, the user has to "**Push**" the file. This files will them be send to the remote "master" repository where other users/collaborators are able to use the newly updated or uploaded file.
6. When other collaborators wish to gain access to the newly updated or uploaded file, they have to connect to the remote "master" repository and "**Pull**" the file. after pulling the file, they are now able to see what changes has been made and also a commit statement made by the user who edited it.

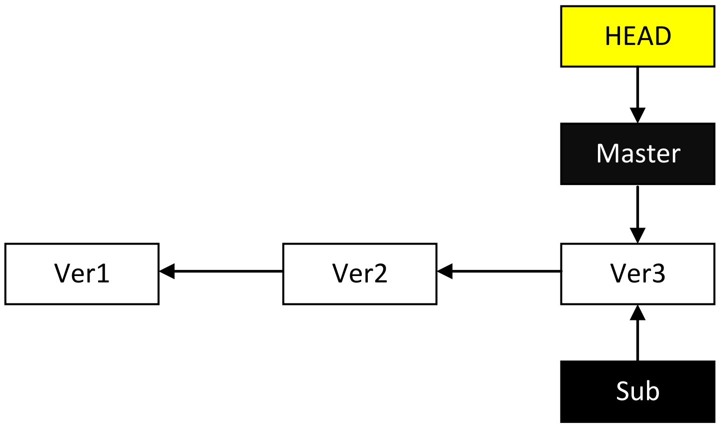
Branching

A branch in Git is simply a lightweight movable pointer to one of the following commits the user made. The default branch name in Git is master. As you initially make commits, you're given a master branch that points to the last commit you made. Every time you commit, it moves forward automatically. Therefore, branching is a basically where the user are able to diverge from the remote "master" repository and continue doing their work without effect the remove "master" repository.

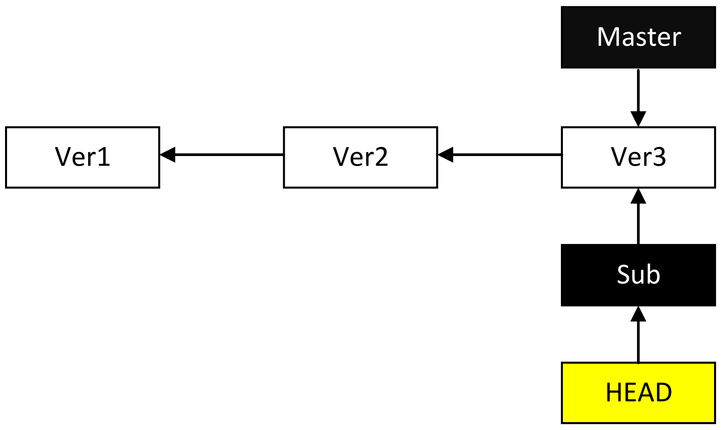


For example, the current master file is pointing to Ver3 and if the user wish to continue working on the Ver3 file without affecting the master file. The user may create a new branch, using the given Git command:

```
$ git branch Sub
```



To determine which branch you are currently on, Git have a special pointer called HEAD which will by default points to the original branch that will be the master branch.

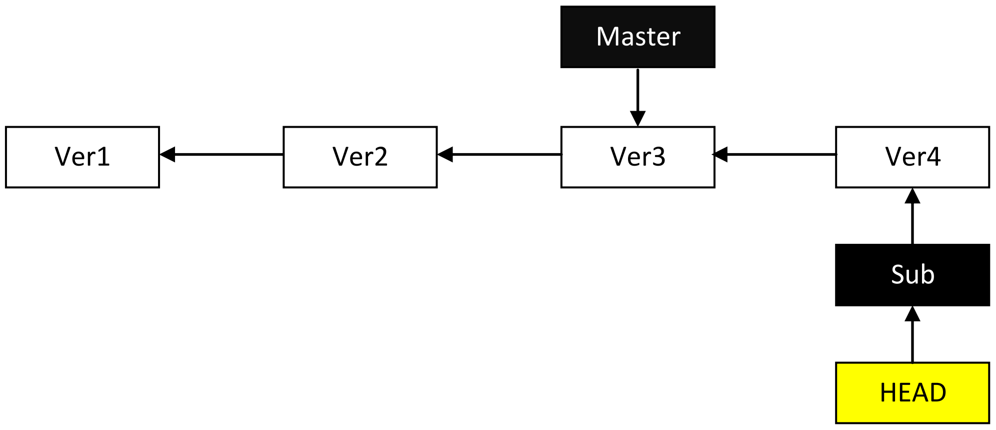


Once the new branch is created, user are able to switch to the newly generated branch using the Git command below:

```
$ git checkout Sub
```

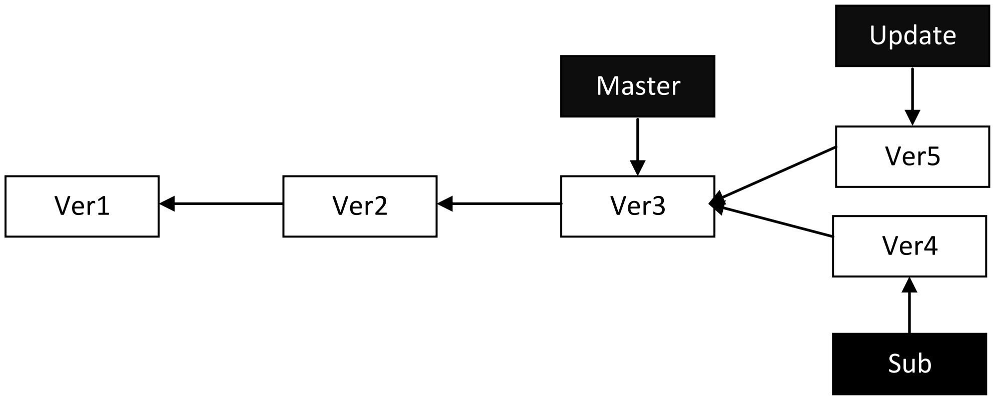
So what this new branch does, well you will see after we do another commit.

```
$ git commit -a -m 'Changes made'
```



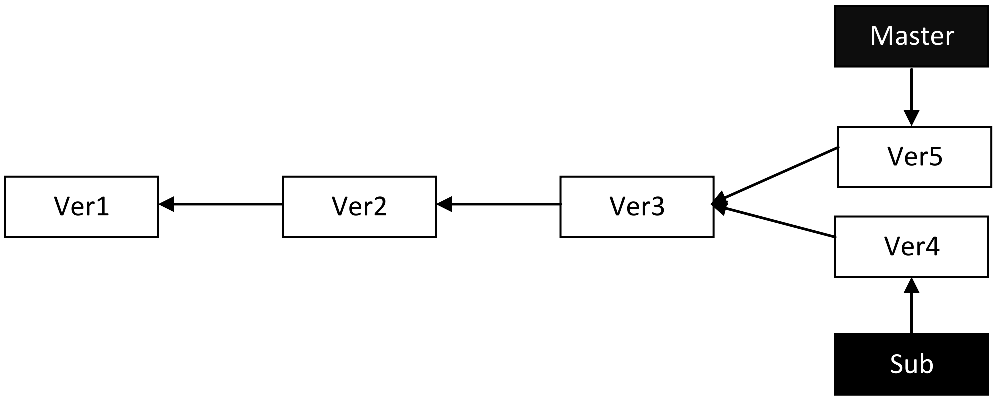
As the commit has been completed, the sub branch will now move forward to a new version whereas for the master branch will remains pointing to where the sub branch was created.

Merging

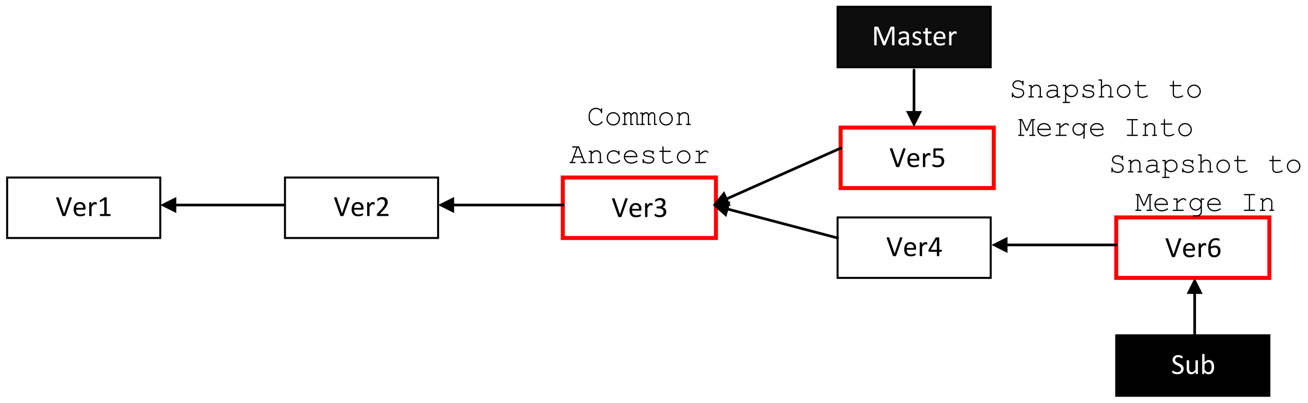


There are many type of merge used in a Git , one of the most easiest and common one is the "Fast-forward" merge. When user attempt to merge one commit with a commit that can be reached by following the first commit's history, Git simplifies things by moving the pointer forward because there is no divergent work to merge together. For this case, Git move forward to the Master branch and Update branch is merge into the Master branch using the following Git command:

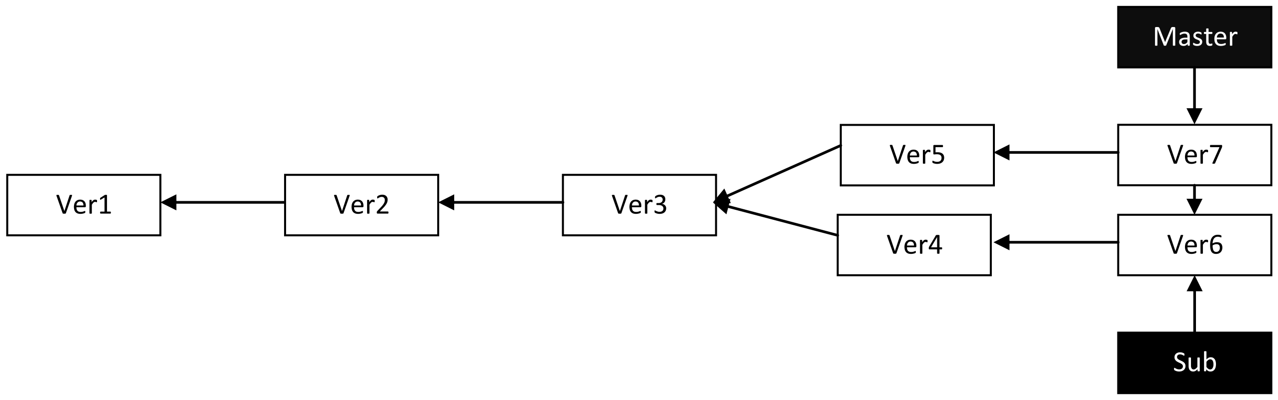
```
$ git checkout master

$ git merge Update
```



By doing so, we first switch be the HEAD pointer to the master branch and then merge the file from the Update branch into the Master branch, whereby the older version file in the Master branch is being updated by the new file merge from the Update branch.



In another merging situation, when latest commit on the branch you are pointing to is not a direct ancestor of the branch you are trying to merge in, Git will has to do some work to determine to best possible merge strategy. Git will then determine that the recursive strategy is best method and perform a "Three-way" merge. The three-way merge uses two snapshots pointed to by the branch tips and the commons ancestor of the two branch that will be Ver3 in this case.



Rather than just moving the branch pointer forward this time, Git creates a new snopshot from the results of three-way merge and automatically creates a new commit that point to Ver7. This commit is referred as the merge commit and it is special in a sense where it will contain more than one parent.

Why is Merging Necessary

Merging is necessary in Git in a sense that when users are working on a separate branch other than the master branch. Any changes made are treat as nothing or ignored if the users did not merge the separate branch into the master branch and push the file so that other users/collaborators may see the changes made.

Merging Conflicts

There are times when the merge process does not operate as planned while changing a part of the same file in separate branch when trying to merge them together. As the merge would not be successful Git will notify you regarding the conflict faced. In order to resolve this conflict, the users are required to first enter the giver Git command shown below:

```
$ git status
```

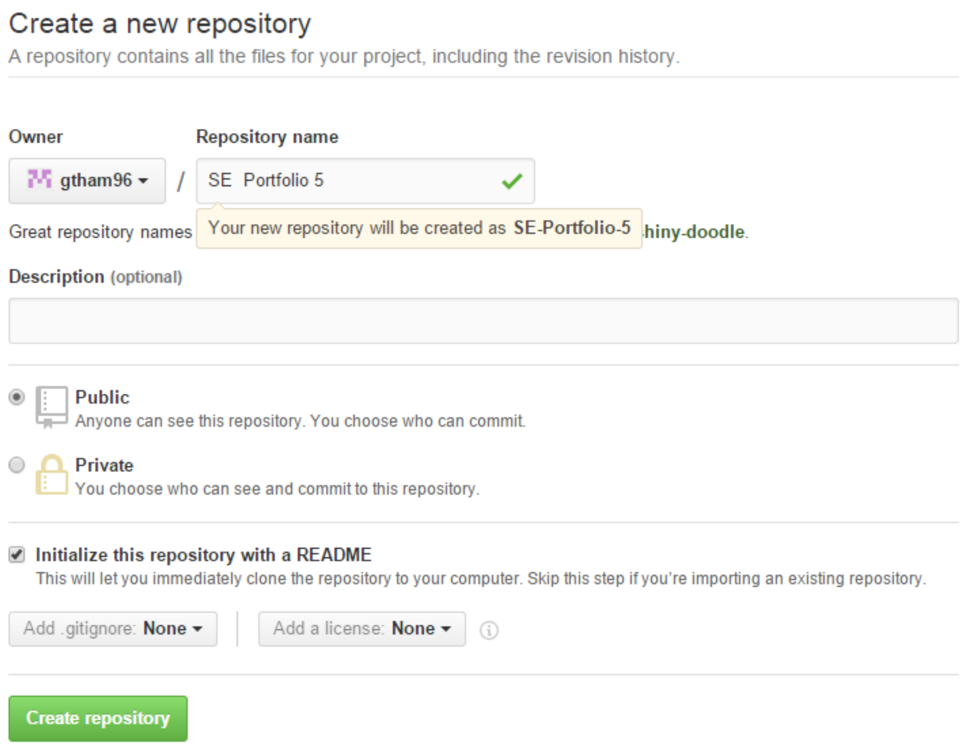
By doing so, we will now be able to see what/which file is causing the conflict that will be shown as the unmerged path. After knowing what is causing the issue, the user can use the following Git command to resolve this conflict completely:

```
$ git mergetool
```

This will open up a GUI that allows you to steps you through each conflict and decide on how you want to merge it. The mergetool also allows user to be able to select which file to hold on and to dispose off and detect the differences from both file which is causing the conflict.

5.2 Demonstration

A short demonstration on the basics command of Git will be shown below, GitHub was chosen my platform where I will demonstrate it.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: gtham96 / Repository name: SE Portfolio 5

Great repository names ... Your new repository will be created as SE-Portfolio-5 ...hiny-doodle.

Description (optional)

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

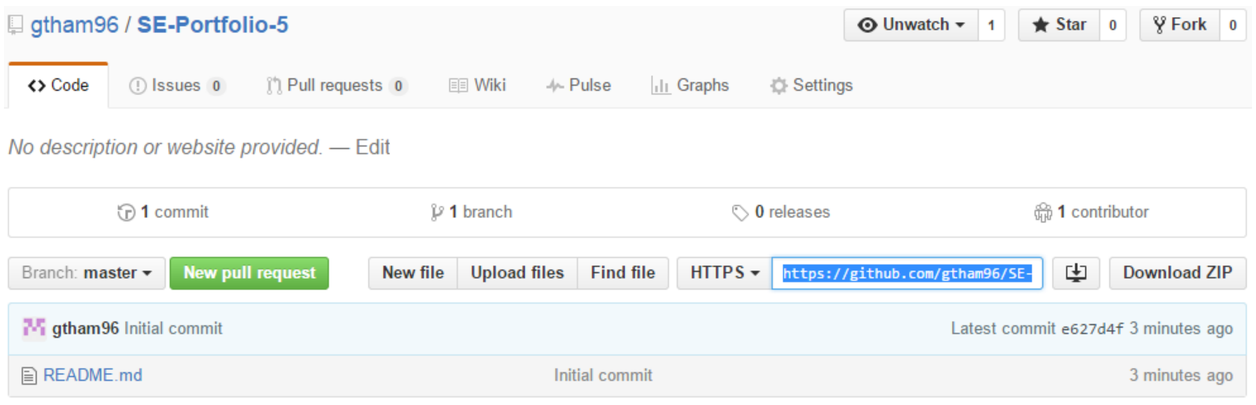
Private — You choose who can see and commit to this repository.

[✓] 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

After creating a repository, user are required to copy the newly created repository URL and clone it to create a local repository of the created repository as shown below:



gtham96 / SE-Portfolio-5

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Wiki | Pulse | Graphs | Settings

No description or website provided. — Edit

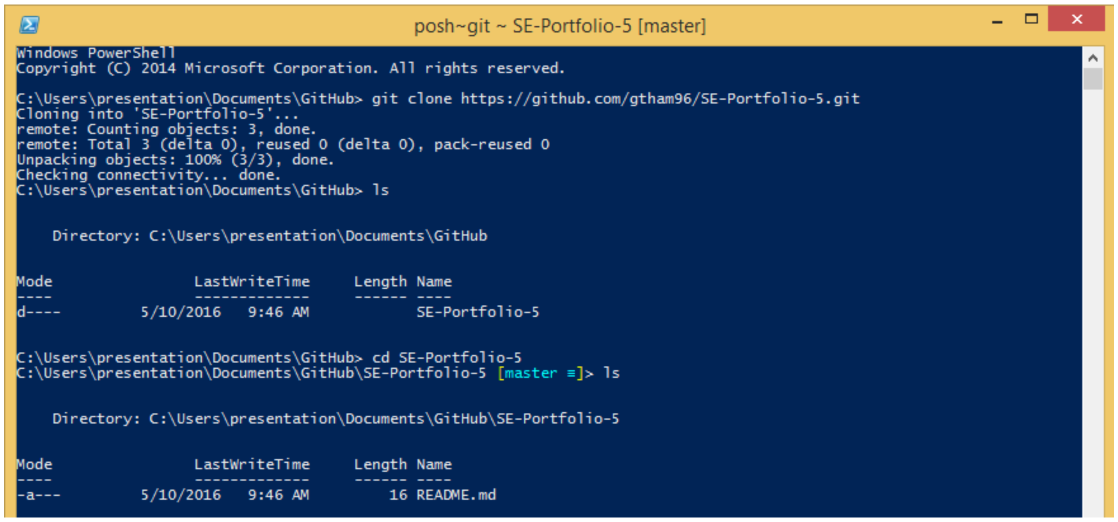
1 commit | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | New file | Upload files | Find file | HTTPS | https://github.com/gtham96/SE- | Download ZIP

gtham96 Initial commit — Latest commit e627d4f 3 minutes ago

README.md — Initial commit — 3 minutes ago

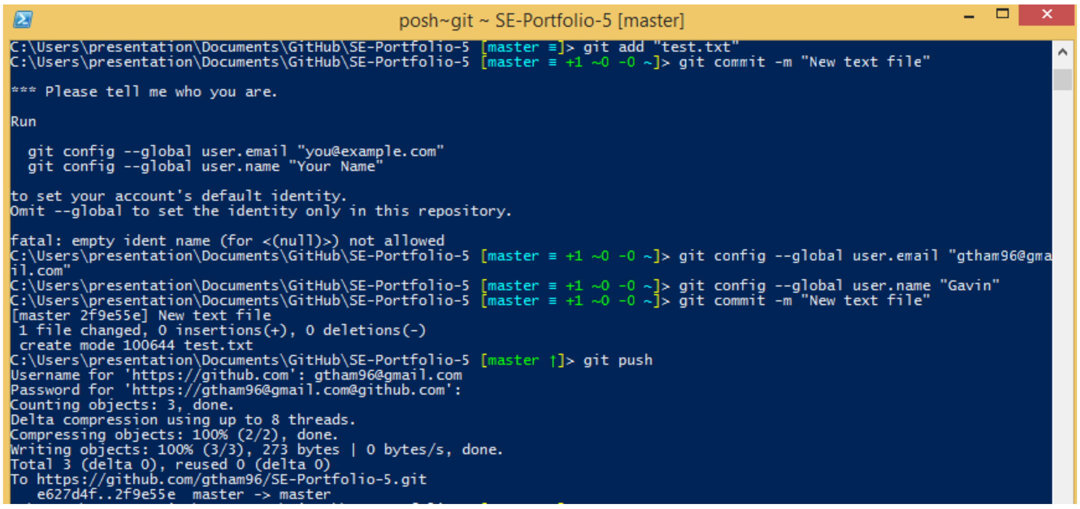
Before we could start cloning the newly created repository as our local repository, the user have to launch GitShell and input the Git clone command with the repository URL. To start working on the local repository, the user has to first change directory into the local repository this steps will all be shown below:



posh~git ~ SE-Portfolio-5 [master]

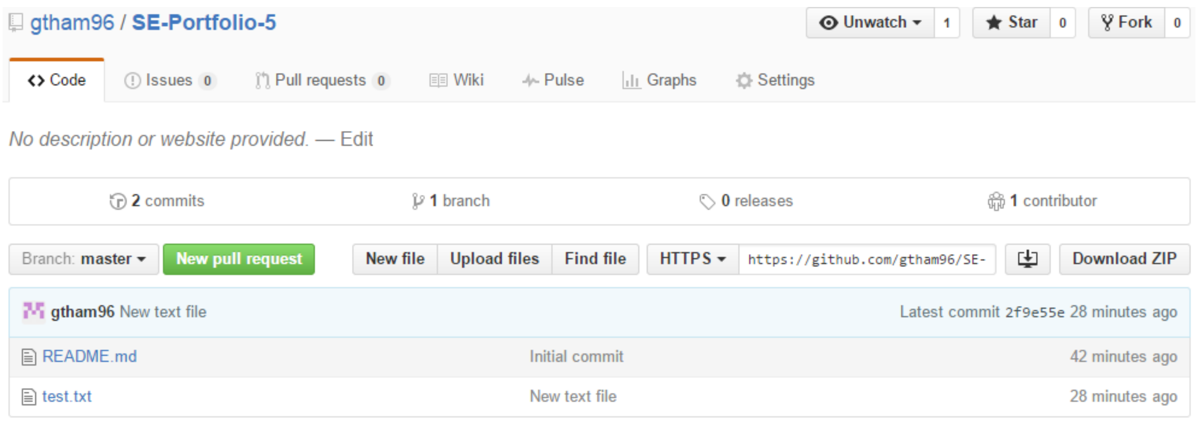
```
Windows PowerShell
Copyright (C) 2014 Microsoft Corporation. All rights reserved.

C:\Users\presentation\Documents\GitHub> git clone https://github.com/gtham96/SE-Portfolio-5.git
Cloning into 'SE-Portfolio-5'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
Checking connectivity... done.
C:\Users\presentation\Documents\GitHub> ls


    Directory: C:\Users\presentation\Documents\GitHub


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----         5/10/2016   9:46 AM            SE-Portfolio-5


C:\Users\presentation\Documents\GitHub> cd SE-Portfolio-5
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]> ls


    Directory: C:\Users\presentation\Documents\GitHub\SE-Portfolio-5


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---         5/10/2016   9:46 AM         16 README.md
```

After cloning the created repository as a local repository, if the user wants to add or update a file into remote repository the user have to first add the file into the staging index and then proceed with committing the changes, the –m tag is used when the user wish to leave a commit message. The file will not be added or updated into the remote repository just yet. All the previous step mentioned is just to place the edited file into staging index with a commit message. In order to add or update the file into the remote repository the user have to use the Git push command and login to their own GitHub account.



posh~git ~ SE-Portfolio-5 [master]

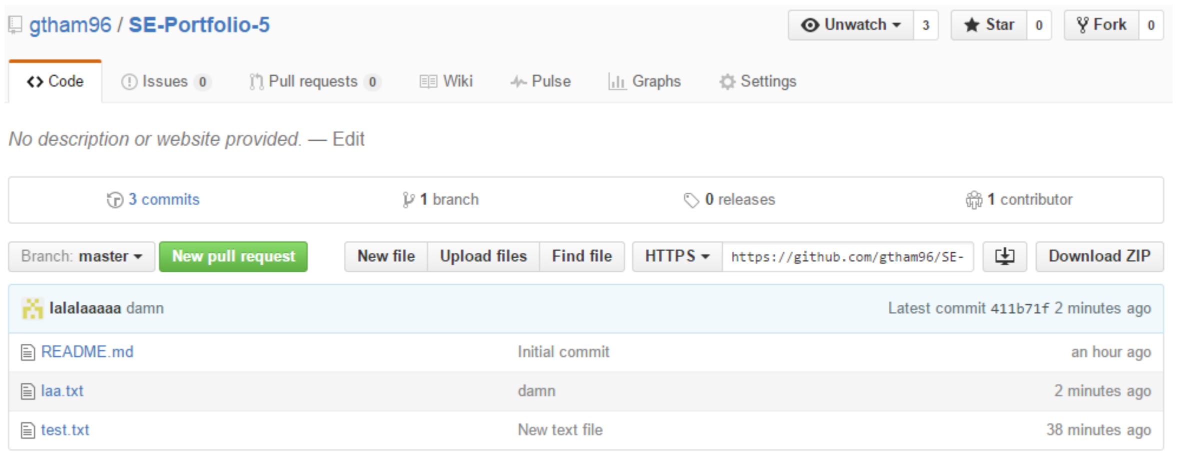
```
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]> git add "test.txt"
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡ +1 ~0 -0 ~]> git commit -m "New text file"

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: empty ident name (for <(null)>) not allowed
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡ +1 ~0 -0 ~]> git config --global user.email "gtham96@gma
il.com"
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡ +1 ~0 -0 ~]> git config --global user.name "Gavin"
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡ +1 ~0 -0 ~]> git commit -m "New text file"
[master 2f9e55e] New text file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.txt
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ↑]> git push
Username for 'https://github.com': gtham96@gmail.com
Password for 'https://gtham96@gmail.com@github.com':
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 273 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/gtham96/SE-Portfolio-5.git
   e627d4f..2f9e55e  master -> master
```



gtham96 New text file — Latest commit 2f9e55e 28 minutes ago

README.md	Initial commit	42 minutes ago
test.txt	New text file	28 minutes ago

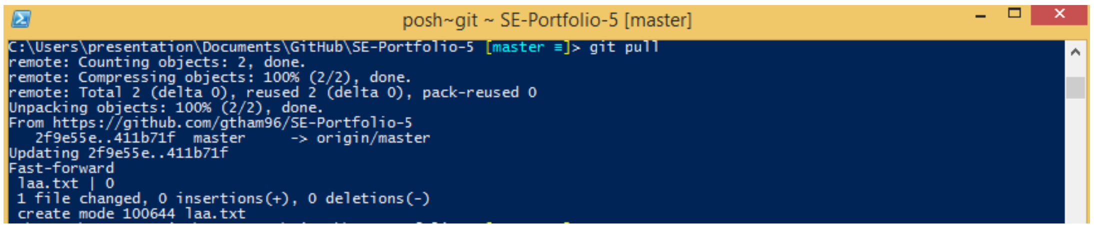
Above is the remote repository with the test.txt file that the user has just use the Git push command to add or update the file into the remote repository.



lalalaaaaa damn — Latest commit 411b71f 2 minutes ago

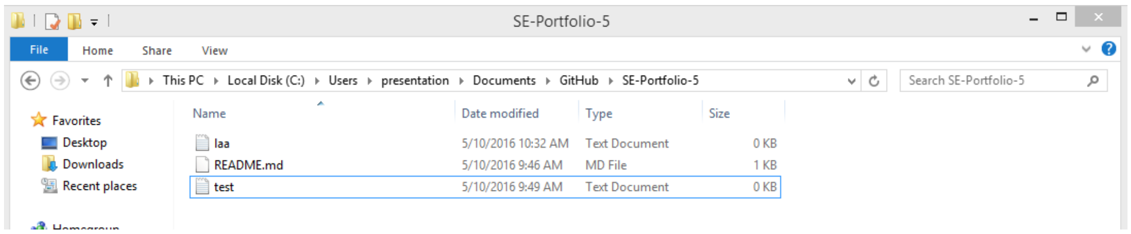
README.md	Initial commit	an hour ago
laa.txt	damn	2 minutes ago
test.txt	New text file	38 minutes ago

In another situation where the user have recruited some collaborators and wish to get the newly added or updated file from the remote repository into their local repository. The user must first enter the Git pull command to retrieve this newly added or updated file into their local repository.



```
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]> git pull
remote: Counting objects: 2, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (2/2), done.
From https://github.com/gtham96/SE-Portfolio-5
   2f9e55e..411b71f  master     -> origin/master
Updating 2f9e55e..411b71f
Fast-forward
 laa.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 laa.txt
```

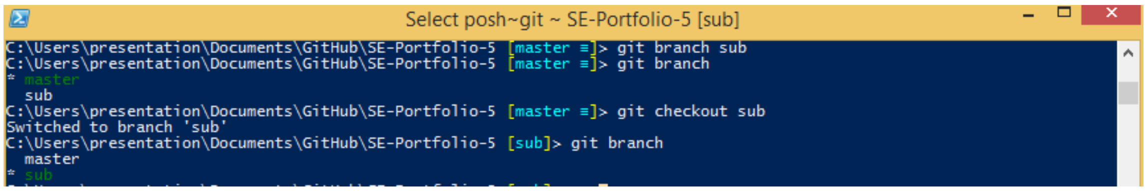
After the Git pull command, the newly added or updated file from the remote repository will now add or update those file into the local repository.



Name	Date modified	Type	Size
laa	5/10/2016 10:32 AM	Text Document	0 KB
README.md	5/10/2016 9:46 AM	MD File	1 KB
test	5/10/2016 9:49 AM	Text Document	0 KB

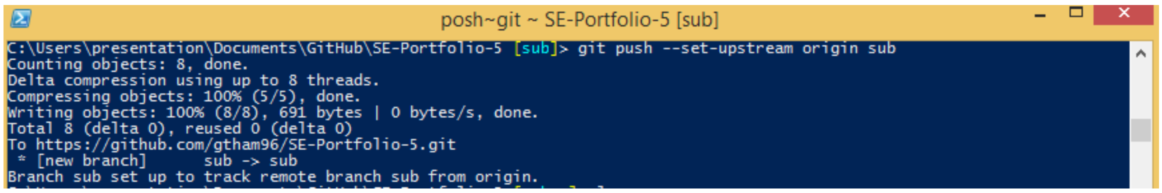
Branching

In some case when the user prefer to edit the files without affect the master file. For this case the user may use the following Git Branch and Git checkout to create and switch to the newly created branch.



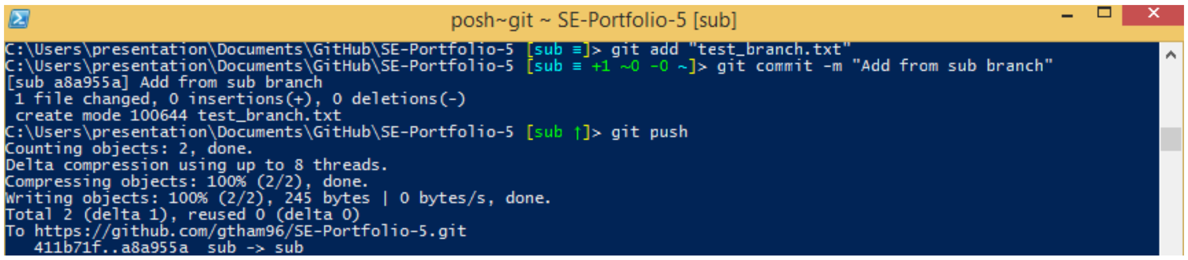
```
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]> git branch sub
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]> git branch
* master
  sub
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]> git checkout sub
Switched to branch 'sub'
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [sub]> git branch
  master
* sub
```

To track all the changes made in the newly created sub branch, the user is required to enter the following Git push –set-upstream origin “branch_name”

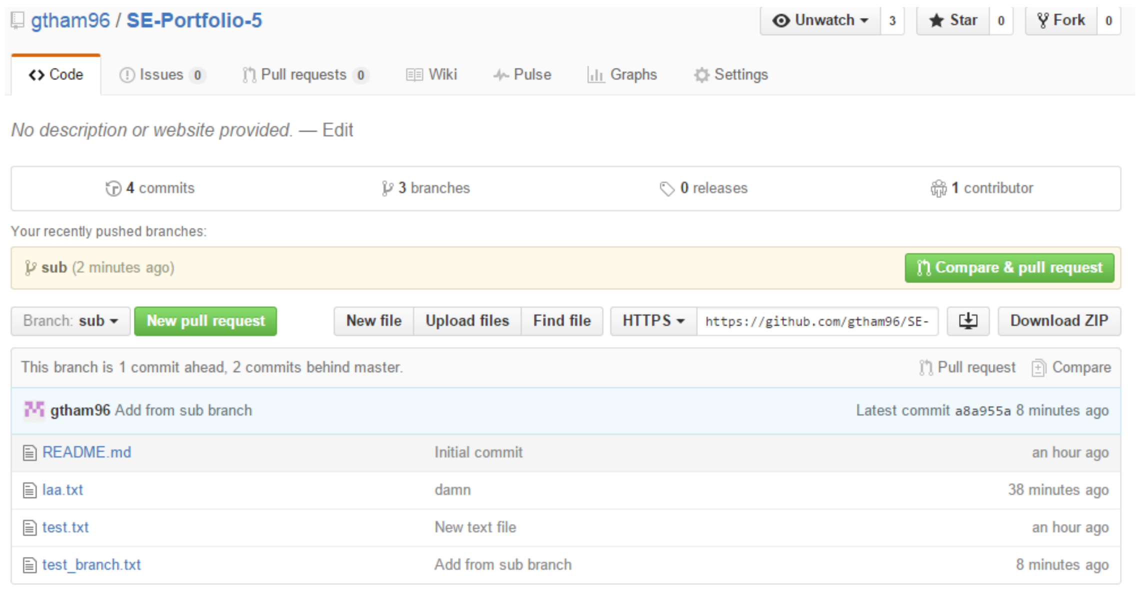


```
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [sub]> git push --set-upstream origin sub
Counting objects: 8, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (8/8), 691 bytes | 0 bytes/s, done.
Total 8 (delta 0), reused 0 (delta 0)
To https://github.com/gtham96/SE-Portfolio-5.git
 * [new branch]      sub -> sub
Branch sub set up to track remote branch sub from origin.
```

To make changes to the remote repository with any edited or newly created file, the user is required to enter the Git push command again.



```
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [sub ≡]> git add "test_branch.txt"
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [sub ≡ +1 ~0 -0 ~]> git commit -m "Add from sub branch"
[sub a8a955a] Add from sub branch
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test_branch.txt
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [sub ↑]> git push
Counting objects: 2, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 245 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
To https://github.com/gtham96/SE-Portfolio-5.git
   411b71f..a8a955a  sub -> sub
```


gtham96 / SE-Portfolio-5

No description or website provided. — Edit

4 commits | 3 branches | 0 releases | 1 contributor

Your recently pushed branches:

sub (2 minutes ago)

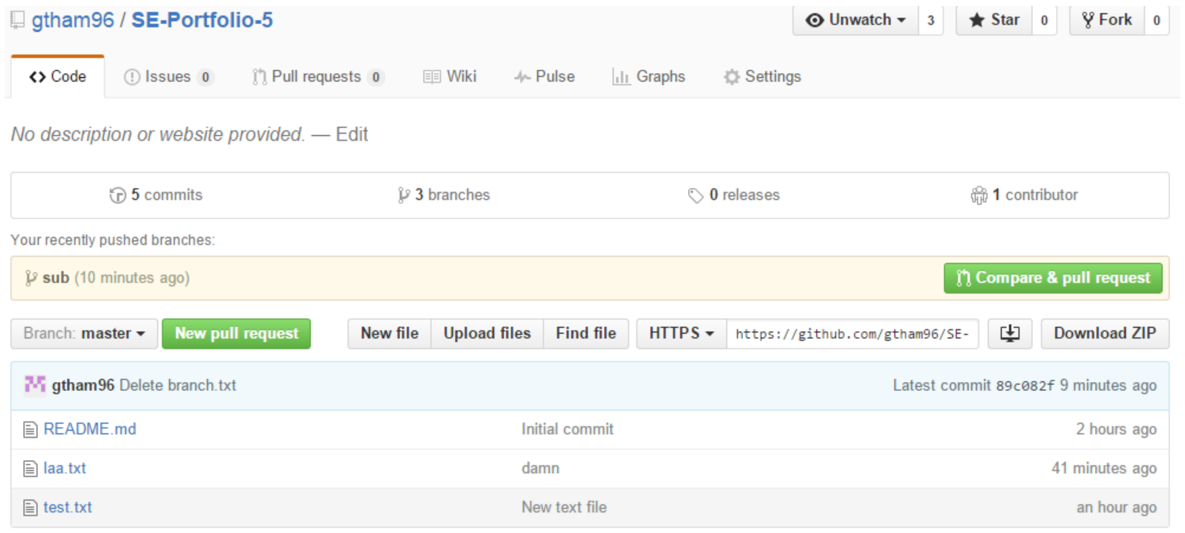
Branch: sub

This branch is 1 commit ahead, 2 commits behind master.

gtham96 Add from sub branch — Latest commit a8a955a 8 minutes ago

README.md	Initial commit	an hour ago
laa.txt	damn	38 minutes ago
test.txt	New text file	an hour ago
test_branch.txt	Add from sub branch	8 minutes ago

The Git push will add now add the test_branch.txt into the remote repository inside of the sub branch and the master branch will go unaffected.


gtham96 / SE-Portfolio-5

No description or website provided. — Edit

5 commits | 3 branches | 0 releases | 1 contributor

Your recently pushed branches:

sub (10 minutes ago)

Branch: master

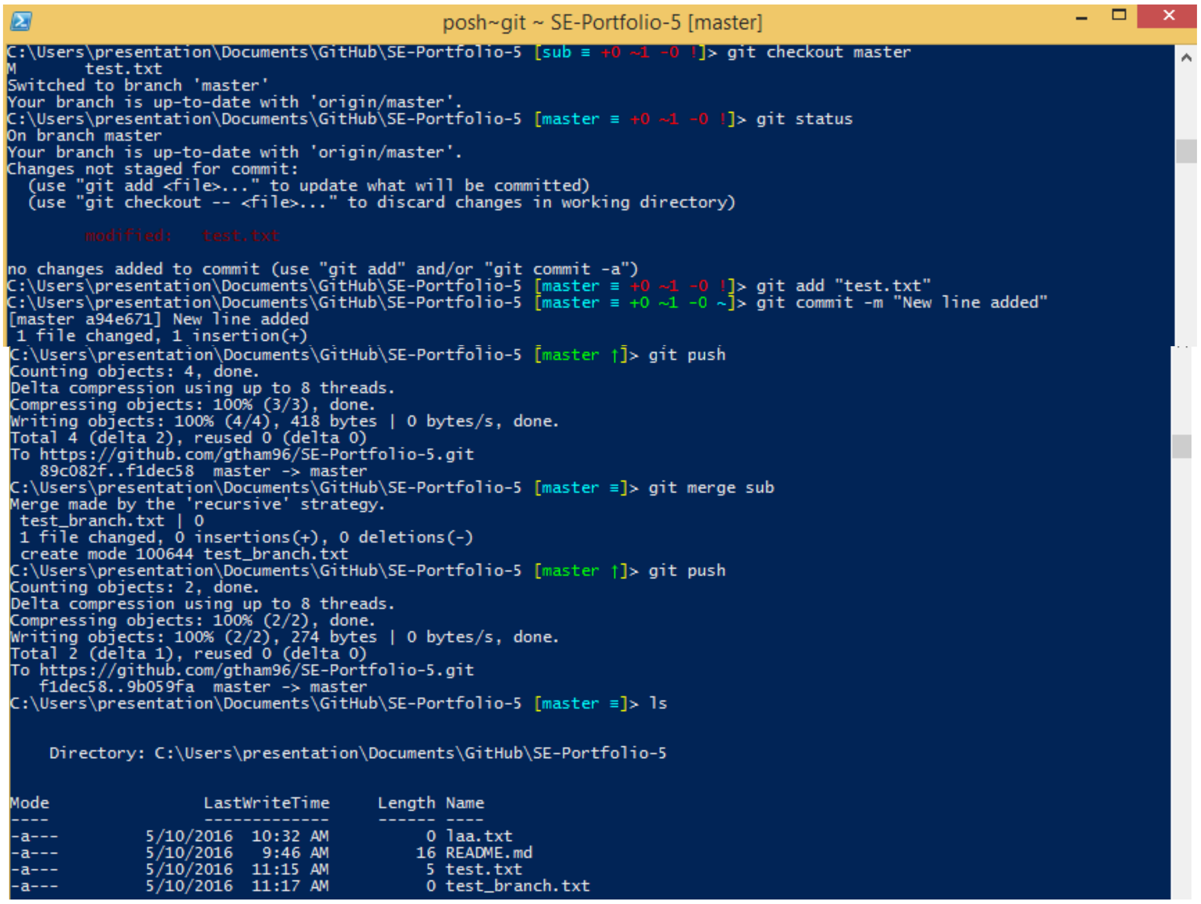
gtham96 Delete branch.txt — Latest commit 89c082f 9 minutes ago

README.md	Initial commit	2 hours ago
laa.txt	damn	41 minutes ago
test.txt	New text file	an hour ago

Merging

Recursive merge occurs when there are commits ahead on the master branch where another branch was created. Git automatically decides the most suitable common ancestors to merge into, while creating snapshot of the current branch commits. For this case, our sub branch is now ahead of the master branch.

Let us return to the master branch and edit a file in the local repository. This edit would not be applies to the sub branch. The edited file is then being added and committed to a stage index and push to the remote repository. Now if the user decide the merge the two branches, where currently the master branch contains a updated version of “test.txt” and the sub branch contains a new file “test_branch.txt”. Git will automatically determine the two branches most common ancestor and apply the “recursive” strategy for both branches to merge together. After merging together, the sub branch files will now merge into the master branch as shown below.



```
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [sub ≡ +0 ~1 -0 !]> git checkout master
M       test.txt
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡ +0 ~1 -0 !]> 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:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡ +0 ~1 -0 !]> git add "test.txt"
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡ +0 ~1 -0 ~]> git commit -m "New line added"
[master a94e671] New line added
 1 file changed, 1 insertion(+)
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ↑]> git push
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 418 bytes | 0 bytes/s, done.
Total 4 (delta 2), reused 0 (delta 0)
To https://github.com/gtham96/SE-Portfolio-5.git
   89c082f..f1dec58  master -> master
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]> git merge sub
Merge made by the 'recursive' strategy.
 test_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test_branch.txt
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ↑]> git push
Counting objects: 2, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 274 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
To https://github.com/gtham96/SE-Portfolio-5.git
   f1dec58..9b059fa  master -> master
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]> ls


    Directory: C:\Users\presentation\Documents\GitHub\SE-Portfolio-5


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---         5/10/2016  10:32 AM          0 laa.txt
-a---         5/10/2016   9:46 AM         16 README.md
-a---         5/10/2016  11:15 AM          5 test.txt
-a---         5/10/2016  11:17 AM          0 test_branch.txt
```