SOFT20111

SOFTWARE ENGINEERING

PORTFOLIO 5

MERGING: THEORY & PRACTICE

PREPARED BY

BRANDON LAU KOK QUIN

A3195

5.1 Git (Theory)

Git is a version control system that is commonly used when developing software and version control tasks. It emphasize on data integrity, speed and also support for distributed, non-linear workflows. Every Git working directory is a repository with a completely detailed history and full version- tracking capabilities. This provides the project developers and also contributors to select specific version, which may be older than the current working version, to develop or revise from it.

Git allows a project team to working on the same file, while avoiding confusion when multiple contributors are editing the file. It works in a way that each individual has their own local version of the file when editing, once they're done, they may send the edited file to the git repository to amend the master file for the project.

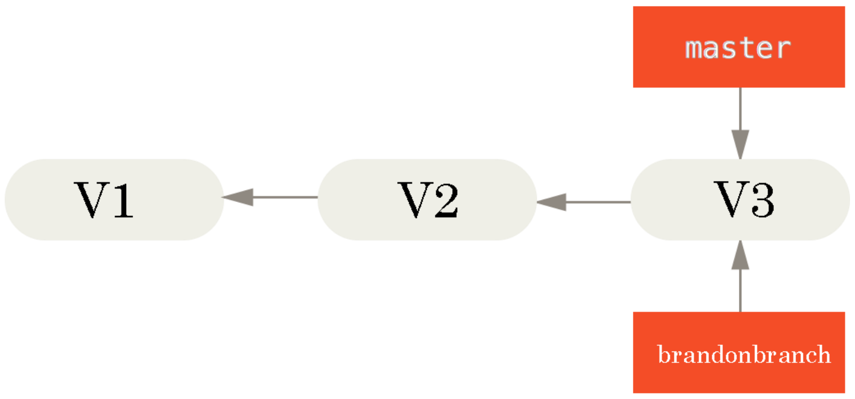
Pull, Add, Commit and Push

Below are the detailed explanation on the **basic** steps when using Git:

1. When a new repository is created, files are added to it by the project leader/developers.
2. Once a new contributor is recruited into the project, they may get the existing file from the repository. This is usually known as "**Clone**" where they download the repository into their local machine, consequently creating a local repo for them to edit.
3. User then edits the file, and save it to their local machine when working on the file. When they're done editing and feels that the file should be uploaded to the repository for the other contributors to use, they will "**Add**" the local file to the staging index. However, this does not change anything in the repository yet.
4. To make changes in the local repository, the user has to "**Commit**" the files. Which will then replace or create new files inside the local repository.
5. Once committing, the user has to "**Push**" the file, sending it to the remote "master" repository. This will allow other users to use the file.
6. When another user wish to edit the newest version of the file, they shall connect to the remote repository and "**Pull**" the file. After they pull the file, they can see what are the changes made and also commit comments left by the previous contributor.

Branching

Branching is a very useful way of doing continuous changes to the file, while not committing the changes and pushing the file to the remote repository. If a user wish to create a new version of the file, they can create a branch for it and commit the file there instead of constantly changing the master file.



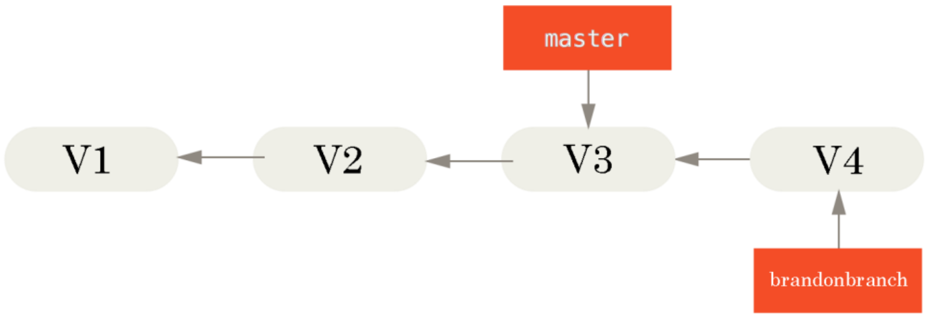
For example, the current master file is V3, while I decided to do some changes to the file while maintaining my own working branch. I will then create a new branch, using the git command:

```
$ git checkout –b brandonbranch
```

This will create a new branch, based on the master version V3. When I wish to commit changes to my own branch, I can use the command:

```
$ git commit –a –m 'made changes in [brandonbranch]'
```

Once the commit is done, a newer version will be created. However, only my branch will be pointing at the new version. While the master version of the file is still V3.

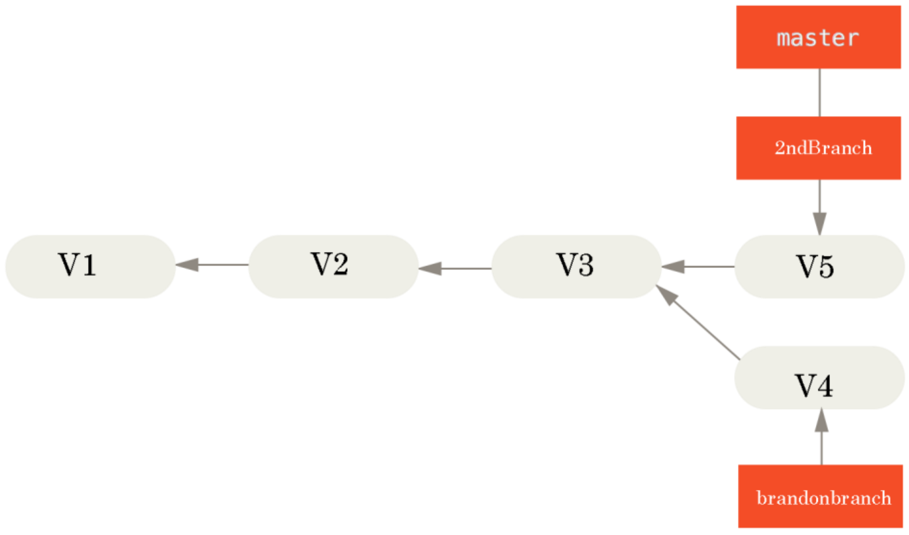


Merging

Merging is necessary in git because when there are users working on a branch, they may merge their file into the master branch to finalize their work. It would mean nothing if the user edits the file but don't merge it into the master branch. Once they determine the file is up-to-date, they may push it to the remote repository for others to pull or fetch.

Following the previous example shown above in the explanation for branching, when a user decides that both the branch should combine together, they shall merge the file back into one branch.

There are many ways of merging a file, one of the most common one is the **fast-forward merge**. When the user tries to merge one commit with another commit than can be reached by following the first commit's history, Git moves the pointer forward because there's no divergent work to merge with.

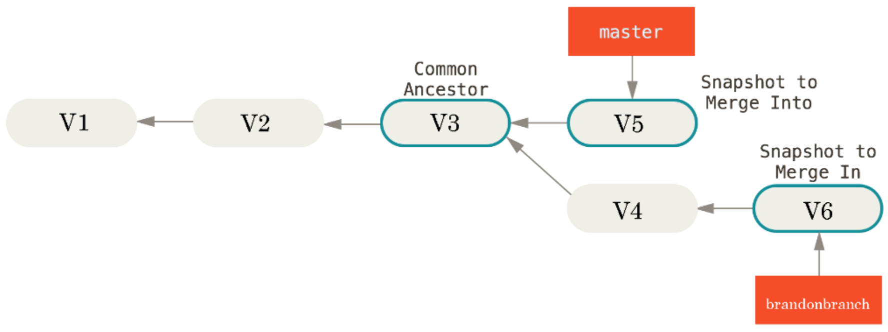


In the example above, I've created a 2ndBranch. When I want to implement the fix in the 2nd branch, I merge the 2nd branch with the master branch. The following command is used:

```
$git checkout master
$git merge 2ndBranch
```

By doing this, I first switch back to the master branch and then merge the file from 2ndBranch into the master. This will replace the older version of files with the new updated ones. This is also known as fast-forward merge.

In another situation where both your master and branch has more than 1 commit ahead, git will determine the merging strategy to merge the files. This is known as the recursive method of merging, below is a picture to show the example:



As shown in the diagram, the branch that was created from V3, has two new commits which is V4 and V6. While working on V4, the master branch has a new commit which is V5. Git creates a snapshot to merge the files in a three-way merge. It determines which the best common ancestor that both branch shares is.

Merge Conflicts

Occasionally, some merge would not go as planned when the same part of the same file and edited. The merge won't be successful and git will inform you regarding the conflict. To solve the conflict, the following command is used:

```
$git status
```

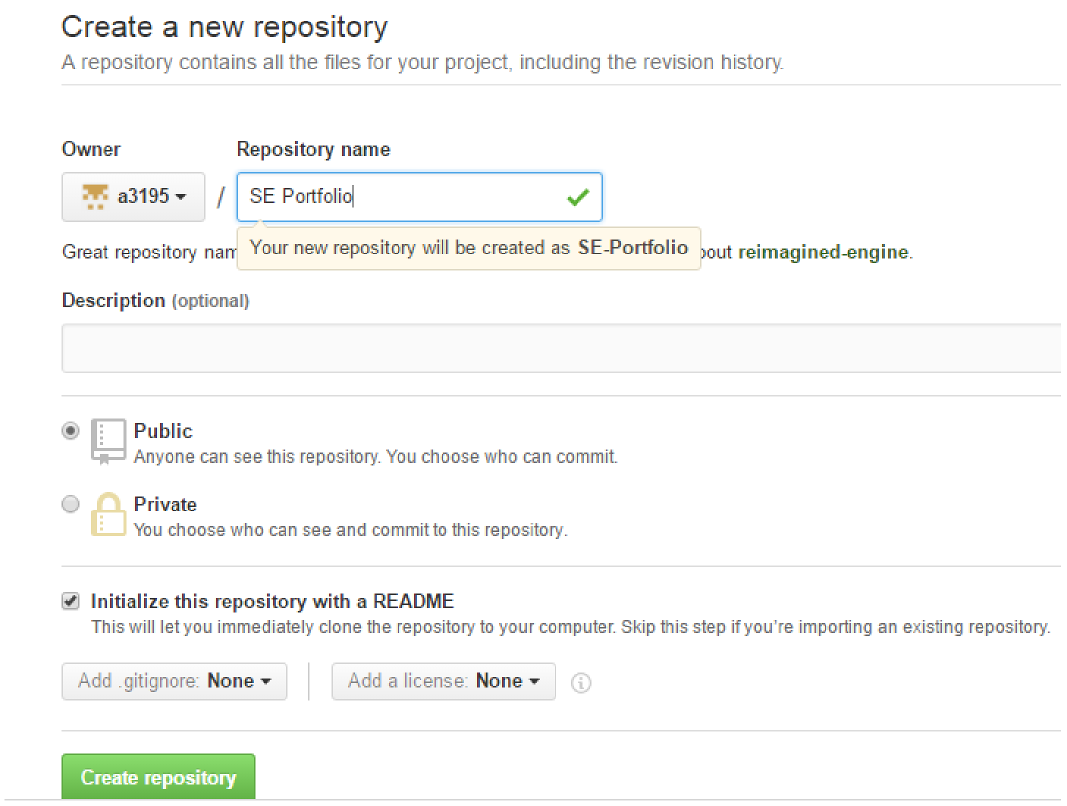
From here, we will be able to see which file is causing the conflict. It will be listed as unmerged file. To solve the conflict, we can use the mergetool as designed by git to handle merging conflicts. The user is able to select which file to keep and which file to dispose while using the mergetool. They are also able to see the differences from both file.

5.2 Demonstration

Below is the demonstration on the basics of Git. GitHub was selected as my platform for practicing.

Starting up

To start using GitHub, first the user must be registered. Then, user has to create a new repository from the user control panel page. Below is the repository creation page.



Create a new repository

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

Owner: a3195 / Repository name: SE Portfolio

Great repository nam Your new repository will be created as **SE-Portfolio**)out **reimagined-engine**.

Description (optional)

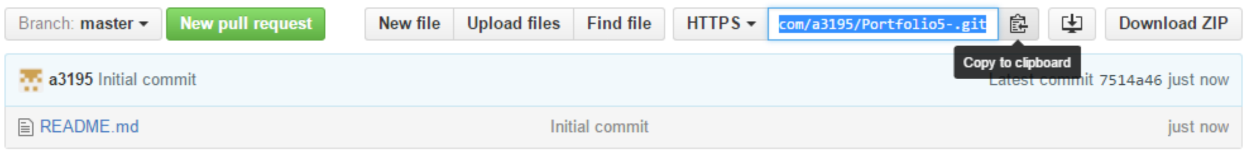
[✓ ●] **Public** Anyone can see this repository. You choose who can commit.

○ **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**

Once a repository is created, user have to copy the repository URL as shown below:

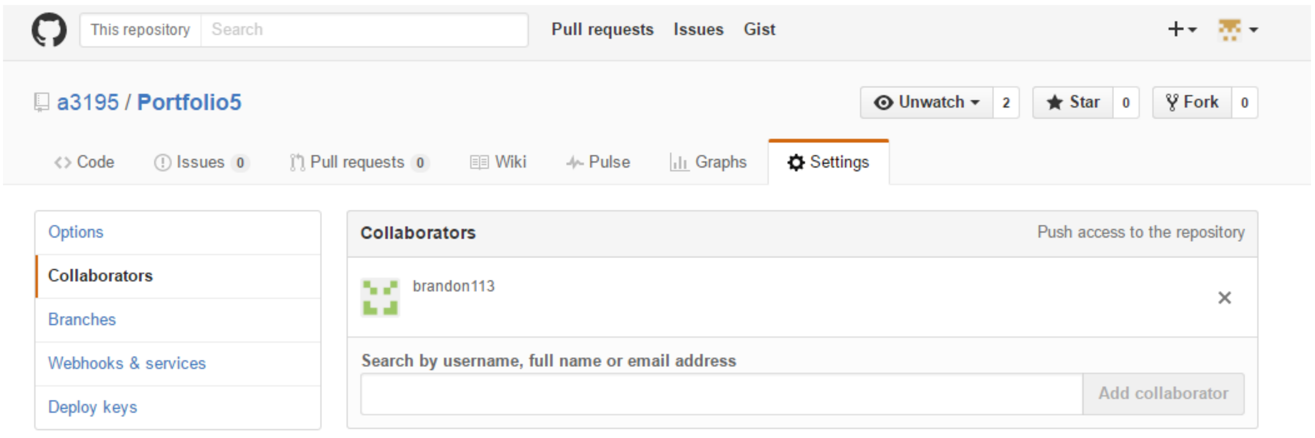


com/a3195/Portfolio5-.git

a3195 Initial commit — Latest commit 7514a46 just now

README.md — Initial commit — just now

Adding contributors/collaborators



Collaborators — Push access to the repository

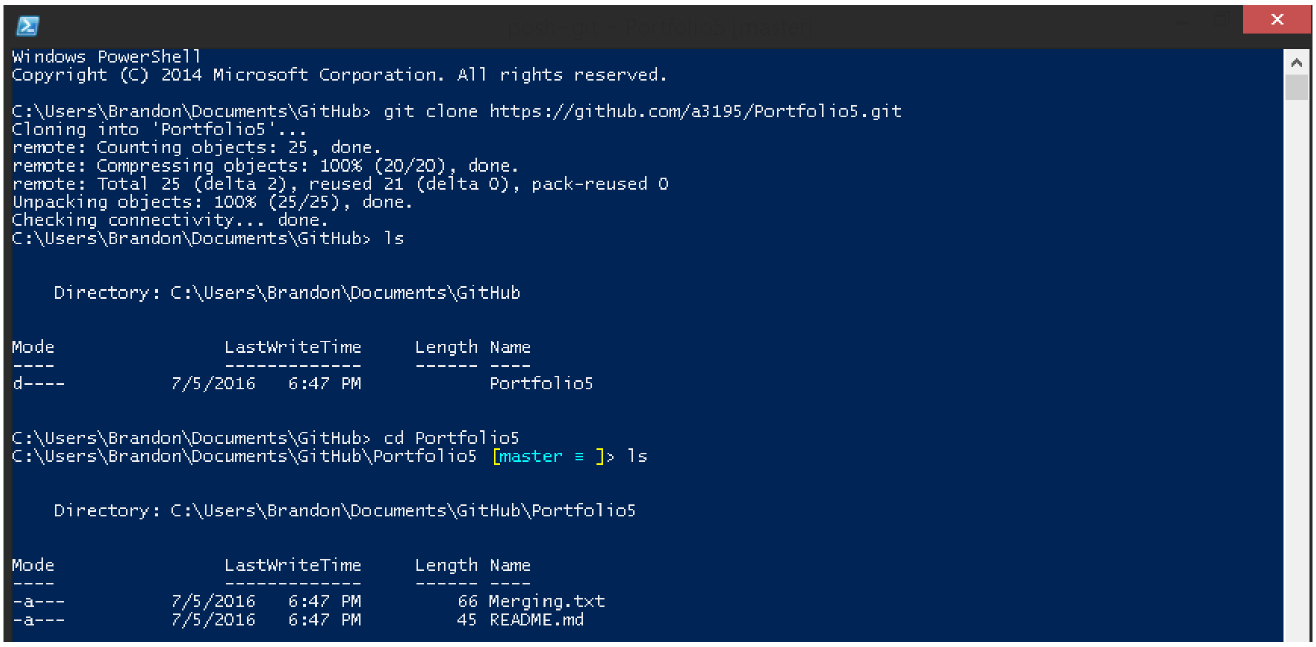
brandon113

Search by username, full name or email address

To add other users to work on the repository, we can add them by going to the repository setting as shown above and add collaborators by searching for their username.

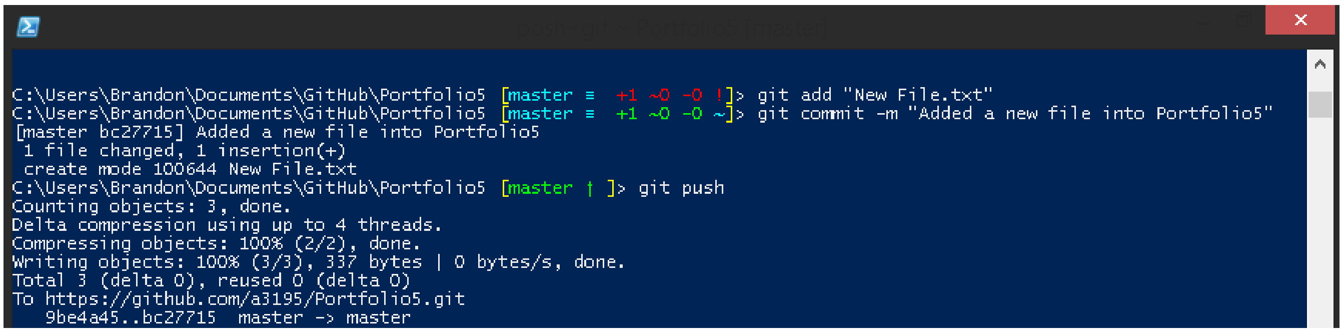
Clone, Add, Commit & Push

Once the user start up the Git Shell, they have to clone the remote repository into the local repository. Then, the user has to change the directory into the local repository to start working on the repo.



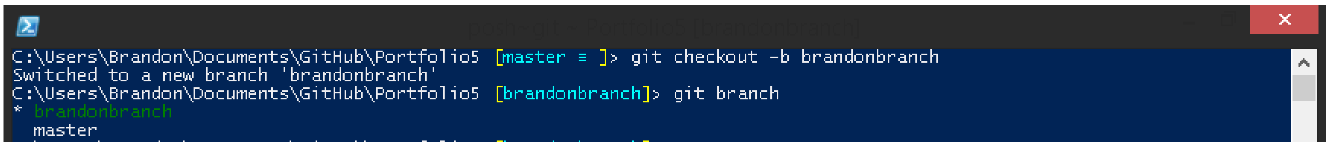
```
Windows PowerShell
Copyright (C) 2014 Microsoft Corporation. All rights reserved.

C:\Users\Brandon\Documents\GitHub> git clone https://github.com/a3195/Portfolio5.git
Cloning into 'Portfolio5'...
remote: Counting objects: 25, done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 25 (delta 2), reused 21 (delta 0), pack-reused 0
Unpacking objects: 100% (25/25), done.
Checking connectivity... done.
C:\Users\Brandon\Documents\GitHub> ls


    Directory: C:\Users\Brandon\Documents\GitHub


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----          7/5/2016   6:47 PM            Portfolio5


C:\Users\Brandon\Documents\GitHub> cd Portfolio5
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡ ]> ls


    Directory: C:\Users\Brandon\Documents\GitHub\Portfolio5


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---          7/5/2016   6:47 PM         66 Merging.txt
-a---          7/5/2016   6:47 PM         45 README.md
```

When the user wish to add or update a file in the local repository, they have to use the Add function to add the file into the staging index. Once the file is added, they have to commit the changes. –m is used to leave a commit message for future references. The file will not be added to the remote repository unless the user enters [git push] to push the file and update the remote repository.



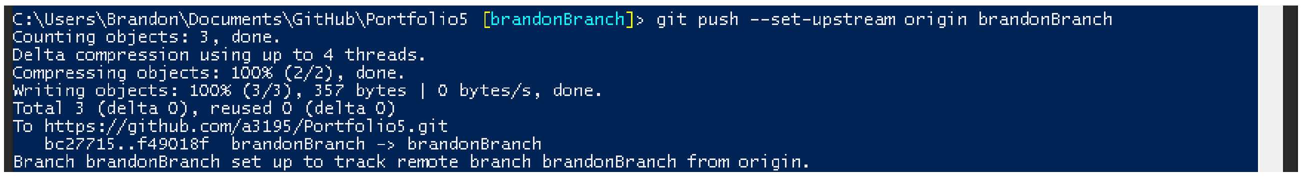
```
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡  +1 ~0 -0 !]> git add "New File.txt"
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡  +1 ~0 -0 ~]> git commit -m "Added a new file into Portfolio5"
[master bc27715] Added a new file into Portfolio5
 1 file changed, 1 insertion(+)
 create mode 100644 New File.txt
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ↑ ]> git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 337 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/a3195/Portfolio5.git
   9be4a45..bc27715  master -> master
```

When a user decides to create a branch to do alter files while not affecting the master file, they can use the following codes to create a branch and also add files and merge it back to the master branch once they're done.

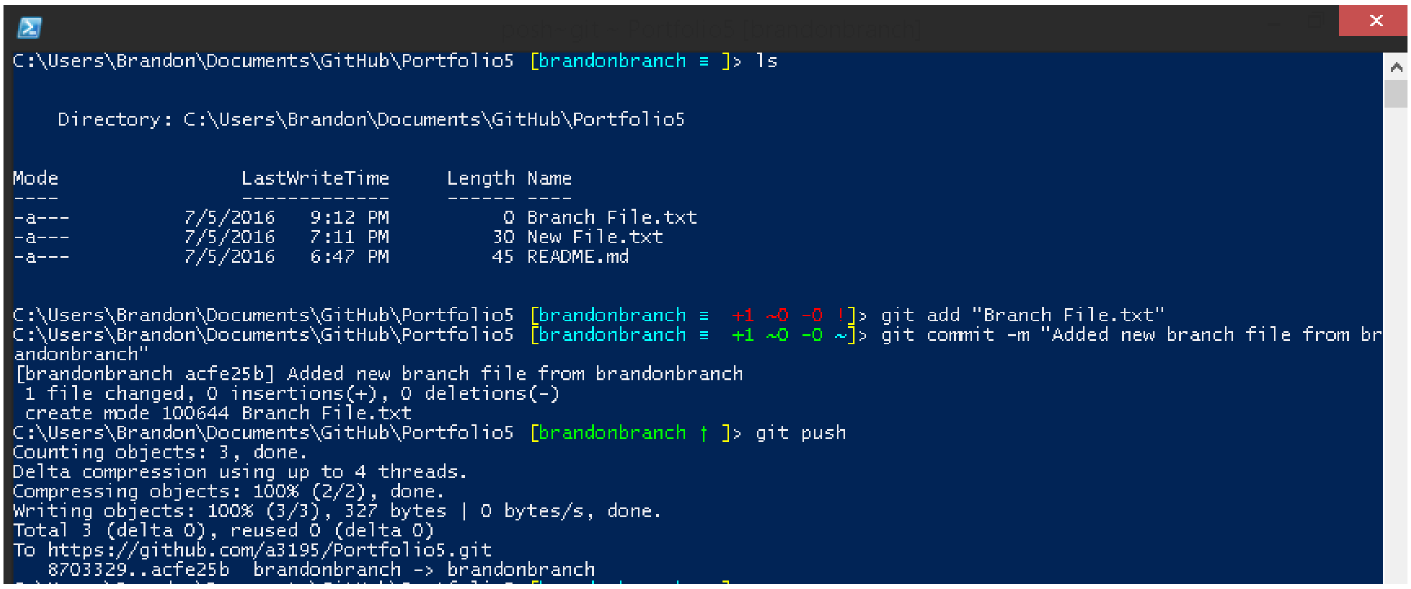


```
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡ ]> git checkout -b brandonbranch
Switched to a new branch 'brandonbranch'
C:\Users\Brandon\Documents\GitHub\Portfolio5 [brandonbranch]> git branch
* brandonbranch
  master
```

User has to set the upstream for the new branch in order to link to the master branch

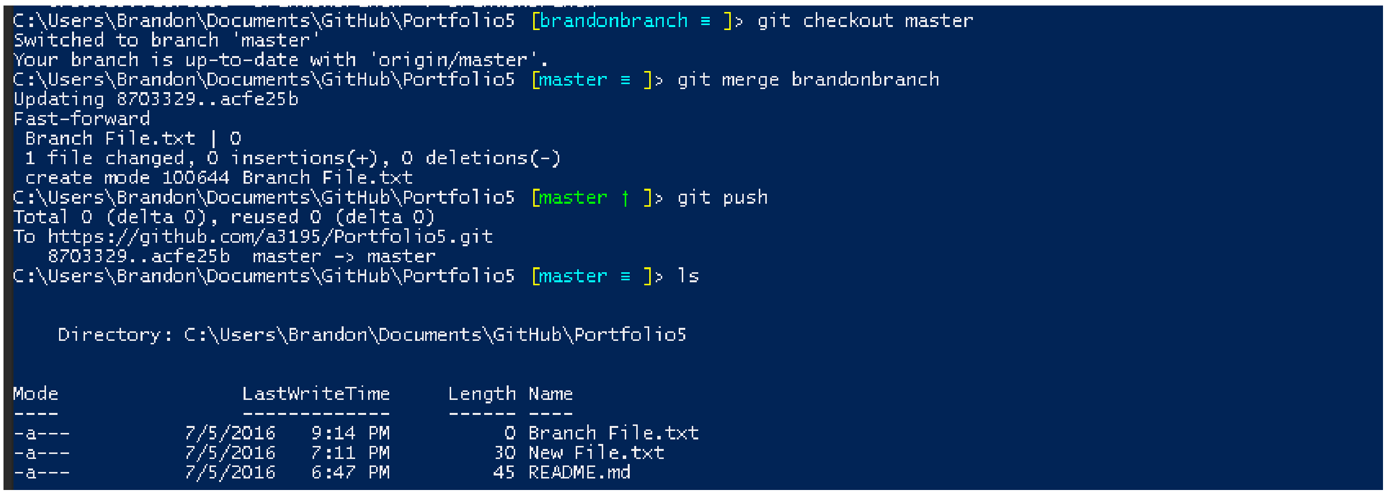


```
C:\Users\Brandon\Documents\GitHub\Portfolio5 [brandonBranch]> git push --set-upstream origin brandonBranch
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 357 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/a3195/Portfolio5.git
   bc27715..f49018f  brandonBranch -> brandonBranch
Branch brandonBranch set up to track remote branch brandonBranch from origin.
```



```
C:\Users\Brandon\Documents\GitHub\Portfolio5 [brandonbranch ≡ ]> ls


    Directory: C:\Users\Brandon\Documents\GitHub\Portfolio5


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---          7/5/2016   9:12 PM          0 Branch File.txt
-a---          7/5/2016   7:11 PM         30 New File.txt
-a---          7/5/2016   6:47 PM         45 README.md


C:\Users\Brandon\Documents\GitHub\Portfolio5 [brandonbranch ≡  +1 ~0 -0 !]> git add "Branch File.txt"
C:\Users\Brandon\Documents\GitHub\Portfolio5 [brandonbranch ≡  +1 ~0 -0 ~]> git commit -m "Added new branch file from br
andonbranch"
[brandonbranch acfe25b] Added new branch file from brandonbranch
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Branch File.txt
C:\Users\Brandon\Documents\GitHub\Portfolio5 [brandonbranch ↑ ]> git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 327 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/a3195/Portfolio5.git
   8703329..acfe25b  brandonbranch -> brandonbranch
```

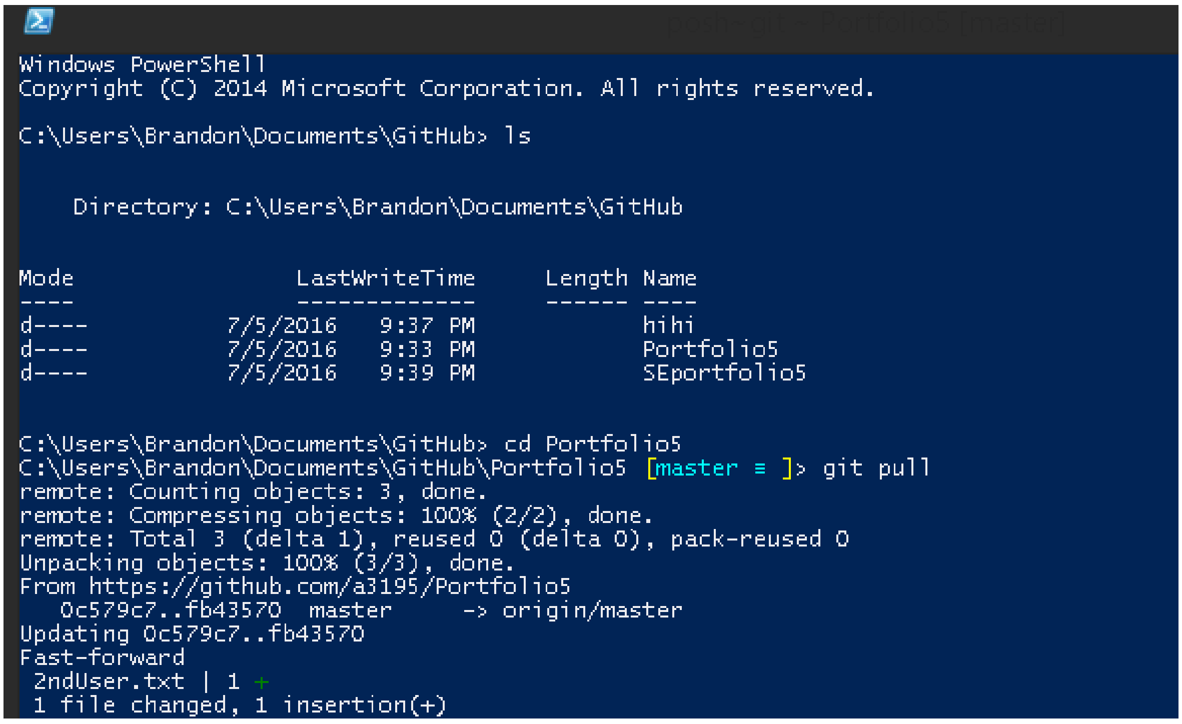
Once the user is done inside the new branch, they may checkout back to the master branch and merge the file using git merge [newbranchname]. When the branches merge without conflicts, they have to push the master branch file to the remote repository using git push. This merge technique is also called the fast-forward merge when the master branch has no commits ahead of the new branch.



```
C:\Users\Brandon\Documents\GitHub\Portfolio5 [brandonbranch ≡ ]> git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡ ]> git merge brandonbranch
Updating 8703329..acfe25b
Fast-forward
 Branch File.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Branch File.txt
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ↑ ]> git push
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/a3195/Portfolio5.git
   8703329..acfe25b  master -> master
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡ ]> ls


    Directory: C:\Users\Brandon\Documents\GitHub\Portfolio5


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---          7/5/2016   9:14 PM          0 Branch File.txt
-a---          7/5/2016   7:11 PM         30 New File.txt
-a---          7/5/2016   6:47 PM         45 README.md
```

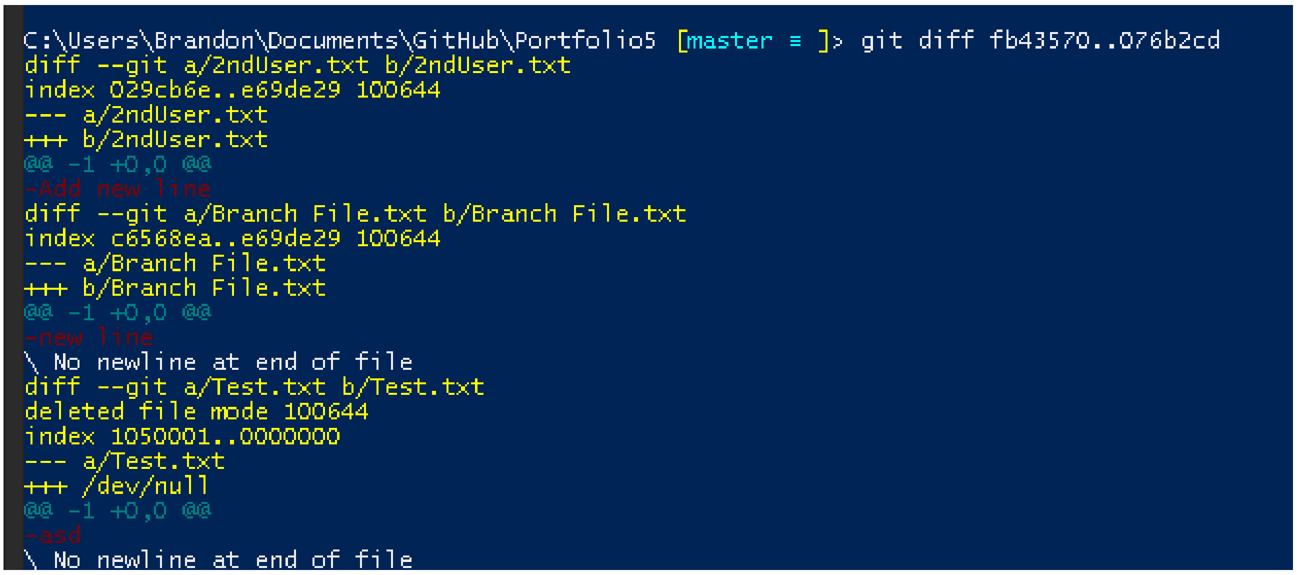
Pull

When other users pushes to the remote repository, we can update our local repository by using the git pull command. This will download the latest files from the origin to our local repository.



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

C:\Users\Brandon\Documents\GitHub> ls


    Directory: C:\Users\Brandon\Documents\GitHub


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----          7/5/2016   9:37 PM            hihi
d----          7/5/2016   9:33 PM            Portfolio5
d----          7/5/2016   9:39 PM            SEportfolio5


C:\Users\Brandon\Documents\GitHub> cd Portfolio5
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡ ]> git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/a3195/Portfolio5
   0c579c7..fb43570  master     -> origin/master
Updating 0c579c7..fb43570
Fast-forward
 2ndUser.txt | 1 +
 1 file changed, 1 insertion(+)
```

To see the differences between the old and new file, we can use the git diff [new SHA] .. [old SHA] which is the hash key for every commit. For example, I would like to see the differences made since the last commit for the file 2ndUser.txt:

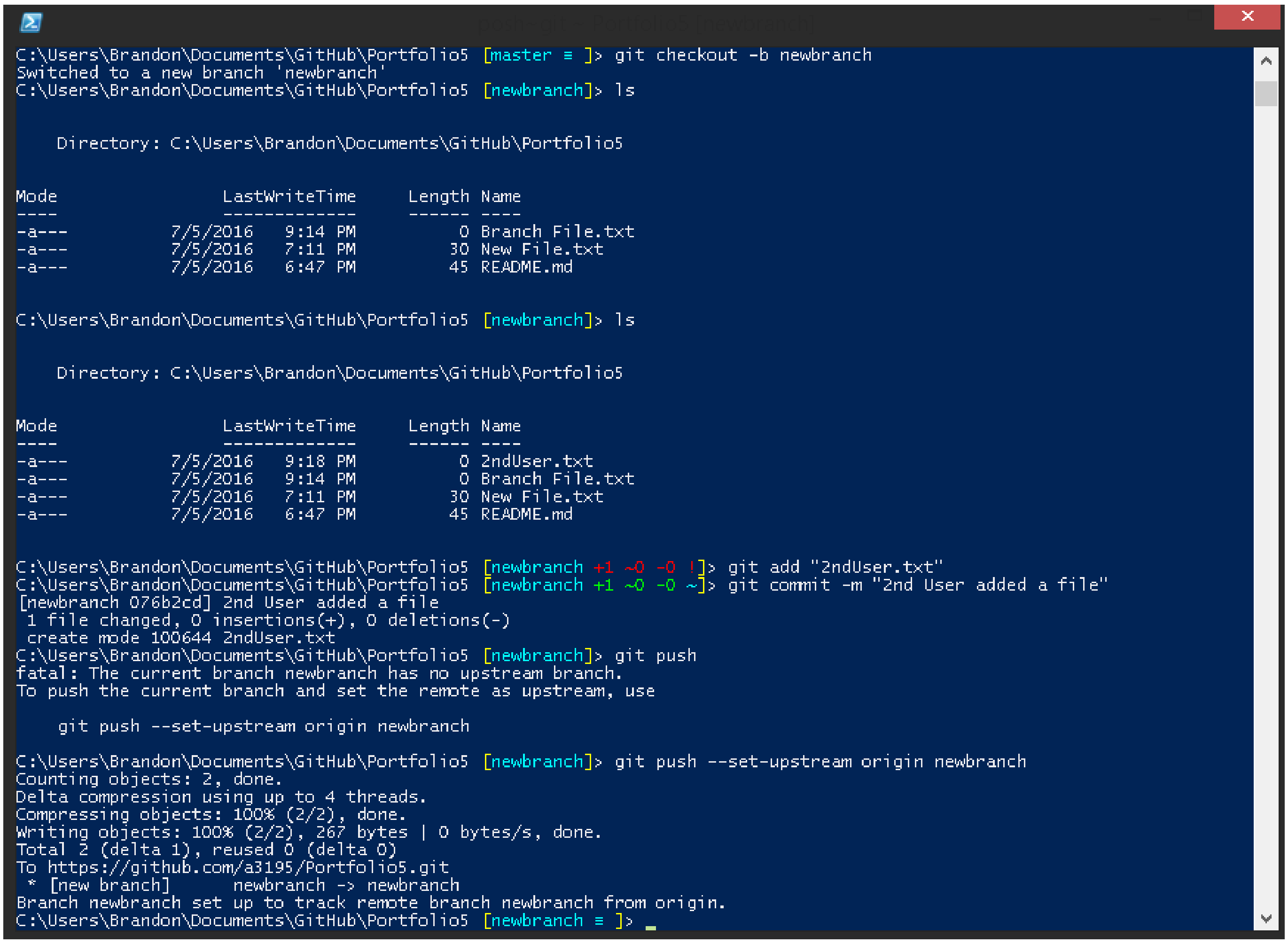


```
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡ ]> git diff fb43570..076b2cd
diff --git a/2ndUser.txt b/2ndUser.txt
index 029cb6e..e69de29 100644
--- a/2ndUser.txt
+++ b/2ndUser.txt
@@ -1 +0,0 @@
-Add new line
diff --git a/Branch File.txt b/Branch File.txt
index c6568ea..e69de29 100644
--- a/Branch File.txt
+++ b/Branch File.txt
@@ -1 +0,0 @@
-new line
\ No newline at end of file
diff --git a/Test.txt b/Test.txt
deleted file mode 100644
index 1050001..0000000
--- a/Test.txt
+++ /dev/null
@@ -1 +0,0 @@
-asd
\ No newline at end of file
```

Recursive Merge

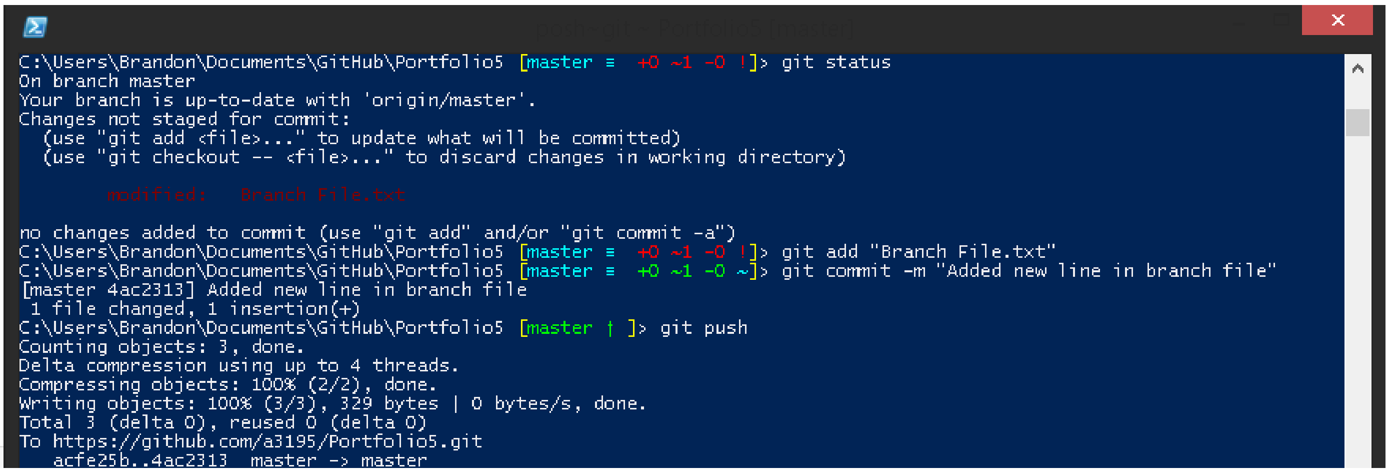
Recursive merge occurs when there are commits ahead on the master branch where another branch was created. Git automatically decides the most suitable common ancestors to merge into, while creating snapshot of the current branch commits.

In this example, a new branch named newbranch is created. A file named "2ndUser.txt" is added to the branch. While this file was committed and pushed to the branch only, the master branch will not have the file "2ndUser.txt"



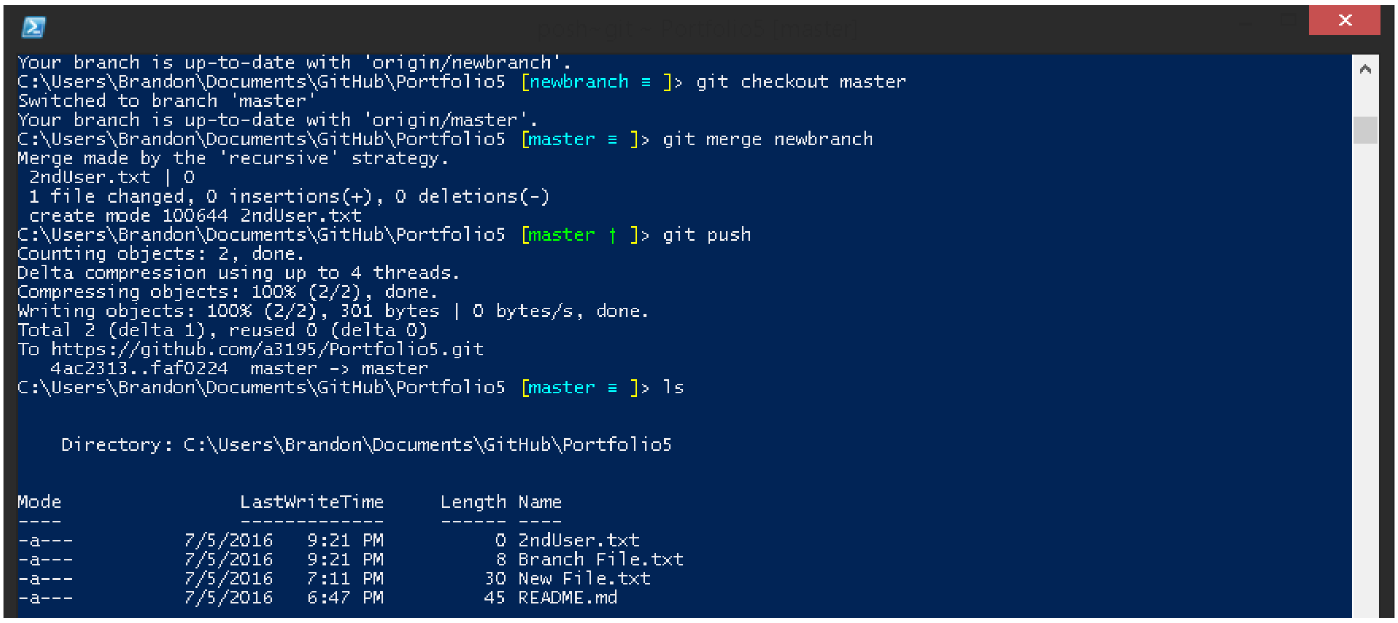
```
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡ ]> git checkout -b newbranch
Switched to a new branch 'newbranch'
C:\Users\Brandon\Documents\GitHub\Portfolio5 [newbranch]> ls


    Directory: C:\Users\Brandon\Documents\GitHub\Portfolio5


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---          7/5/2016   9:14 PM          0 Branch File.txt
-a---          7/5/2016   7:11 PM         30 New File.txt
-a---          7/5/2016   6:47 PM         45 README.md


C:\Users\Brandon\Documents\GitHub\Portfolio5 [newbranch]> ls


    Directory: C:\Users\Brandon\Documents\GitHub\Portfolio5


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---          7/5/2016   9:18 PM          0 2ndUser.txt
-a---          7/5/2016   9:14 PM          0 Branch File.txt
-a---          7/5/2016   7:11 PM         30 New File.txt
-a---          7/5/2016   6:47 PM         45 README.md


C:\Users\Brandon\Documents\GitHub\Portfolio5 [newbranch +1 ~0 -0 !]> git add "2ndUser.txt"
C:\Users\Brandon\Documents\GitHub\Portfolio5 [newbranch +1 ~0 -0 ~]> git commit -m "2nd User added a file"
[newbranch 076b2cd] 2nd User added a file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2ndUser.txt
C:\Users\Brandon\Documents\GitHub\Portfolio5 [newbranch]> git push
fatal: The current branch newbranch has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin newbranch

C:\Users\Brandon\Documents\GitHub\Portfolio5 [newbranch]> git push --set-upstream origin newbranch
Counting objects: 2, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 267 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
To https://github.com/a3195/Portfolio5.git
 * [new branch]      newbranch -> newbranch
Branch newbranch set up to track remote branch newbranch from origin.
C:\Users\Brandon\Documents\GitHub\Portfolio5 [newbranch ≡ ]> _
```

Returning to the master branch, there is a file which was edited. This edit does not exist in the newbranch however. The updated file is added, committed and pushed to the master branch.



```
C:\Users\Brandon\Documents\GitHub\Portfolio5 [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:   Branch File.txt

no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡  +0 ~1 -0 !]> git add "Branch File.txt"
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡  +0 ~1 -0 ~]> git commit -m "Added new line in branch file"
[master 4ac2313] Added new line in branch file
 1 file changed, 1 insertion(+)
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ↑ ]> git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 329 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/a3195/Portfolio5.git
   acfe25b..4ac2313  master -> master
```

Once the user decides to merge the branch, where master branch contains a newer version of “Branch File.txt”, while newbranch contains a “2ndUser.txt” file, Git determines its’ common ancestors and uses the “recursive” strategy to merge.



```
Your branch is up-to-date with 'origin/newbranch'.
C:\Users\Brandon\Documents\GitHub\Portfolio5 [newbranch ≡ ]> git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡ ]> git merge newbranch
Merge made by the 'recursive' strategy.
 2ndUser.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2ndUser.txt
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ↑ ]> git push
Counting objects: 2, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 301 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
To https://github.com/a3195/Portfolio5.git
   4ac2313..faf0224  master -> master
C:\Users\Brandon\Documents\GitHub\Portfolio5 [master ≡ ]> ls


    Directory: C:\Users\Brandon\Documents\GitHub\Portfolio5


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---          7/5/2016   9:21 PM          0 2ndUser.txt
-a---          7/5/2016   9:21 PM          8 Branch File.txt
-a---          7/5/2016   7:11 PM         30 New File.txt
-a---          7/5/2016   6:47 PM         45 README.md
```