

Git – 5 commands to success

Table of Contents

Command 1 (and 1.1)	2
init & cd	2
Command 2	4
Pull	4
Command 3 (and 3.1)	5
Add	5
Command 4	7
Commit	7
Command 5	8
Push	8

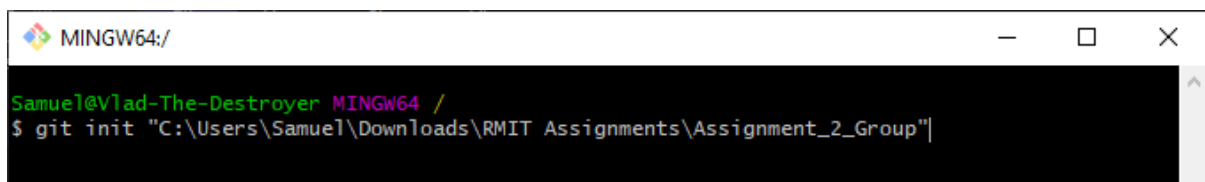
Command 1 (and 1.1)

init & cd

There are two things to do when first starting with git.

1. The command `git init "<your directory path to the local repository>"` (note the double quotes)

Example: `git init "C:\Users\Samuel\Downloads\RMIT Assignments\Assignment_2_Group"`



```
MINGW64:/
Samuel@Vlad-The-Destroyer MINGW64 /
$ git init "C:\Users\Samuel\Downloads\RMIT Assignments\Assignment_2_Group"
```

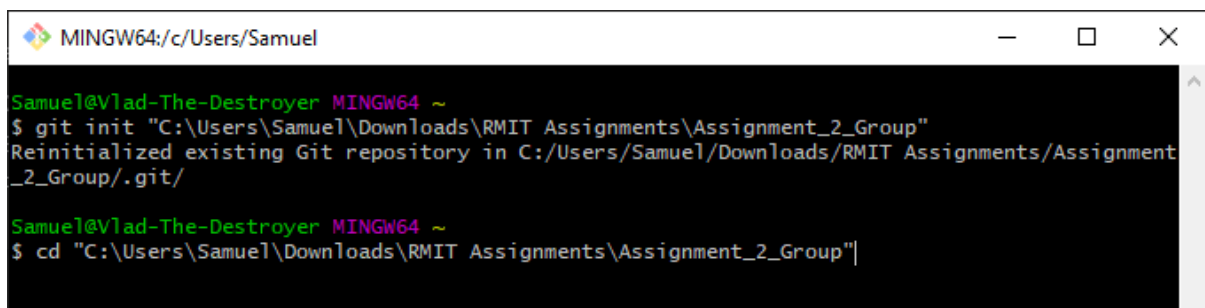
This initialises the local repository (repo) – basically, it creates the relevant files in a hidden folder at the directory location you specified, so git will work.

You only need to do this the very first time you start a local repo. The next time/s you access this local repo you will simply need to ensure your Git Bash console is set to the correct directory (covered next!).

Note:

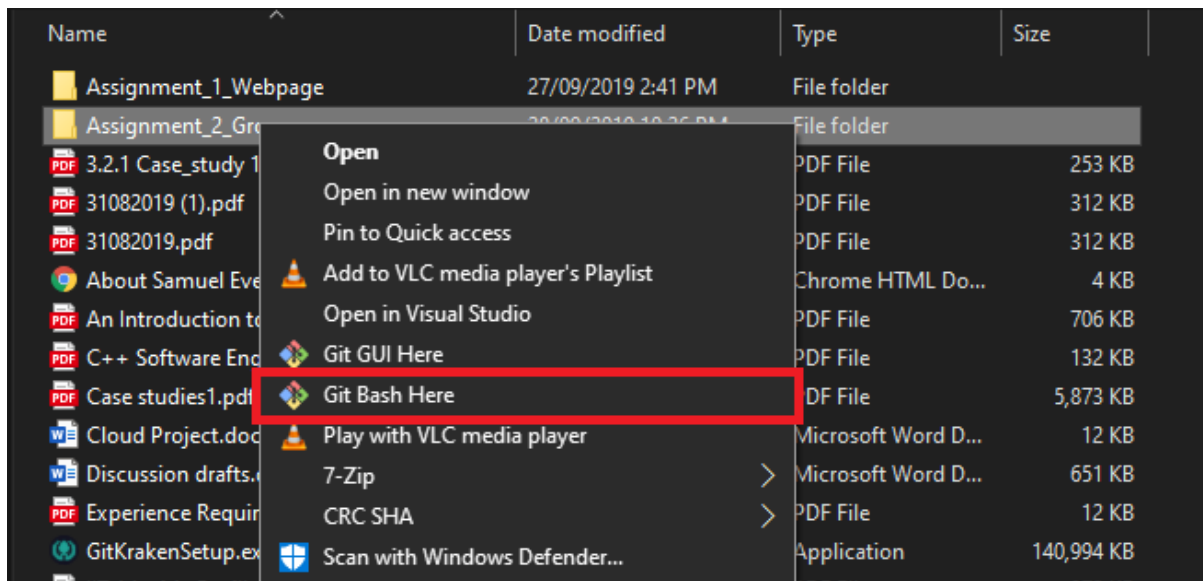
You can choose anywhere you want on your computer to contain your project and initialise git to, just make it somewhere you remember!

2. Set the directory path to your local repo in the git bash console – There are two ways to do this:
 - a. Type `cd "<your local repo pathway here>"` (note the double quotes)
 - i. **Example:** `cd "C:\Users\Samuel\Downloads\RMIT Assignments\Assignment_2_Group"`



```
MINGW64:/c/Users/Samuel
Samuel@Vlad-The-Destroyer MINGW64 ~
$ git init "C:\Users\Samuel\Downloads\RMIT Assignments\Assignment_2_Group"
Reinitialized existing Git repository in C:/Users/Samuel/Downloads/RMIT Assignments/Assignment_2_Group/.git/
Samuel@Vlad-The-Destroyer MINGW64 ~
$ cd "C:\Users\Samuel\Downloads\RMIT Assignments\Assignment_2_Group"
```

- b. Locate the folder in windows explorer or mac finder, right click the folder and click 'Git Bash Here'.



Now your local repository location is selected. Whatever we do next, this is the location that will be affected. Any files in this location will now be managed by git so any changes made, additional files created, or files removed etc. will be tracked.

Command 2

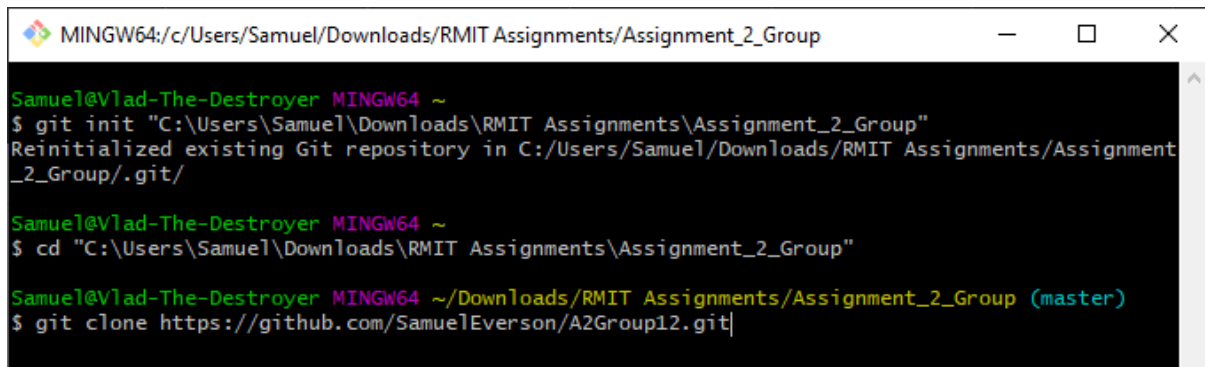
Pull

Now that we have initialised our local git repo and set the path in git bash, we can pull a repo from the web, such as a GitHub repo.

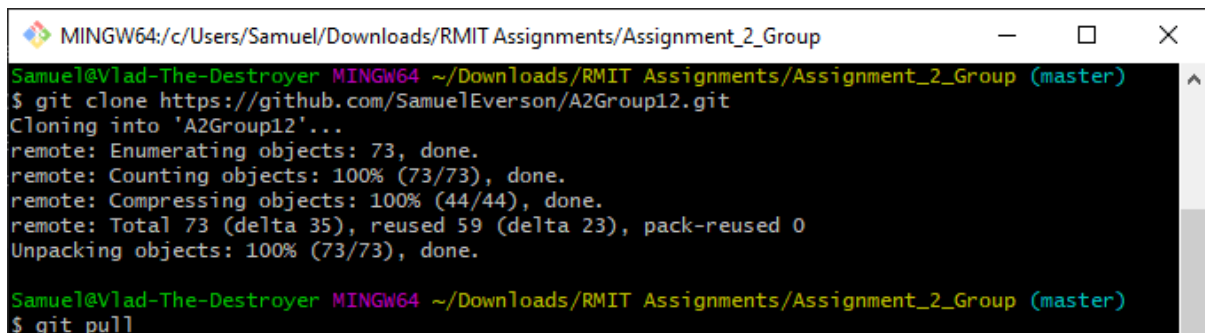
We first need to set up our repository (repo) location to push and pull from.

To do this you can get the URL from the remote repo and use it in the command `git clone`, e.g.;

`git clone https://github.com/SamuelEverson/A2Group12.git`

A terminal window titled 'MINGW64:/c/Users/Samuel/Downloads/RMIT Assignments/Assignment_2_Group'. The prompt is 'Samuel@Vlad-The-Destroyer MINGW64 ~'. The user enters '\$ git init "C:\Users\Samuel\Downloads\RMIT Assignments\Assignment_2_Group"', which outputs 'Reinitialized existing Git repository in C:/Users/Samuel/Downloads/RMIT Assignments/Assignment_2_Group/.git/'. The user then enters '\$ cd "C:\Users\Samuel\Downloads\RMIT Assignments\Assignment_2_Group"', and the prompt changes to 'Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)'. Finally, the user enters '\$ git clone https://github.com/SamuelEverson/A2Group12.git'.

Once that is set up simply type 'git pull' and it will pull or 'download' the files stored in the GitHub repo.

A terminal window titled 'MINGW64:/c/Users/Samuel/Downloads/RMIT Assignments/Assignment_2_Group'. The prompt is 'Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)'. The user enters '\$ git clone https://github.com/SamuelEverson/A2Group12.git', which outputs 'Cloning into 'A2Group12'...', 'remote: Enumerating objects: 73, done.', 'remote: Counting objects: 100% (73/73), done.', 'remote: Compressing objects: 100% (44/44), done.', 'remote: Total 73 (delta 35), reused 59 (delta 23), pack-reused 0', and 'Unpacking objects: 100% (73/73), done.'. The user then enters '\$ git pull'.

Remember these files are as up to date as the latest push from each user (which we will cover shortly).

Now you've done a pull command you will also notice if you navigate to the folder in windows explorer or mac finder, any files updated with the pull will be available (hence why I sometime refer to a pull as a download).

Command 3 (and 3.1)

Add

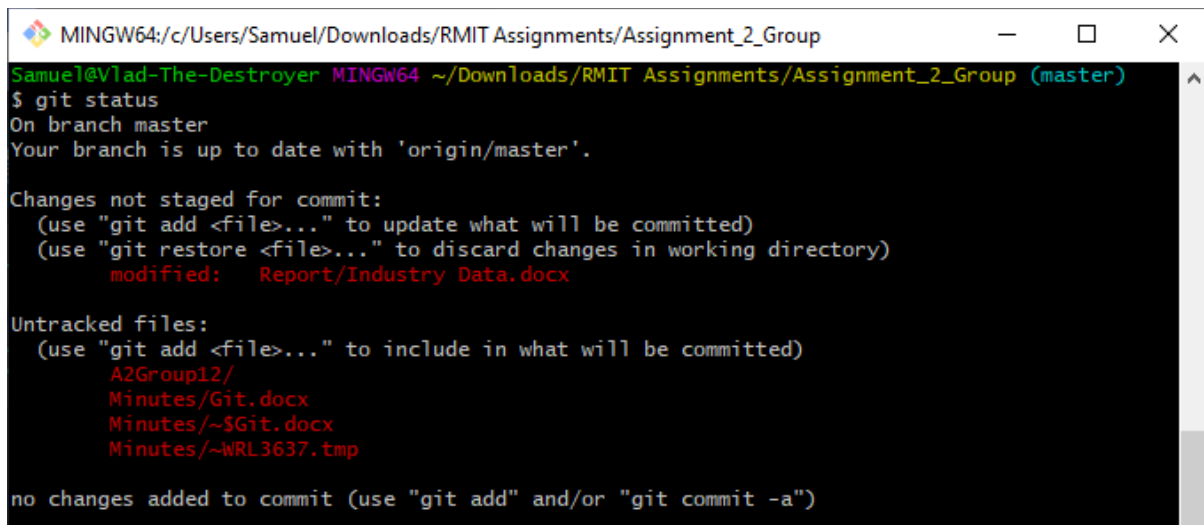
So, we've set the git bash to the correct directory and pulled the latest repo from GitHub. Now assume we've made some changes to a document and it's time add the changes to the 'staging' area – in other words we are going to make git recognize these changes are going to take effect.

Command: `git add <file name or wildcard>`

Note:

Command 3.1 would be `git status`. Git status shows you what files in your local repo either have changes and are ready to be added to the staging area OR which files have already been added.

First I like to check what's happening in the staging area with `git status`:

A terminal window titled 'MINGW64: c:/Users/Samuel/Downloads/RMIT Assignments/Assignment_2_Group' showing the output of the 'git status' command. The output indicates the user is on the 'master' branch and that the branch is up to date with 'origin/master'. It lists 'Changes not staged for commit' with instructions on how to use 'git add' and 'git restore'. A specific file, 'Report/Industry Data.docx', is listed as 'modified'. Under 'Untracked files', it lists 'A2Group12/Minutes/Git.docx', 'Minutes/~\$Git.docx', and 'Minutes/~WRL3637.tmp'. At the bottom, it states 'no changes added to commit (use "git add" and/or "git commit -a")'.

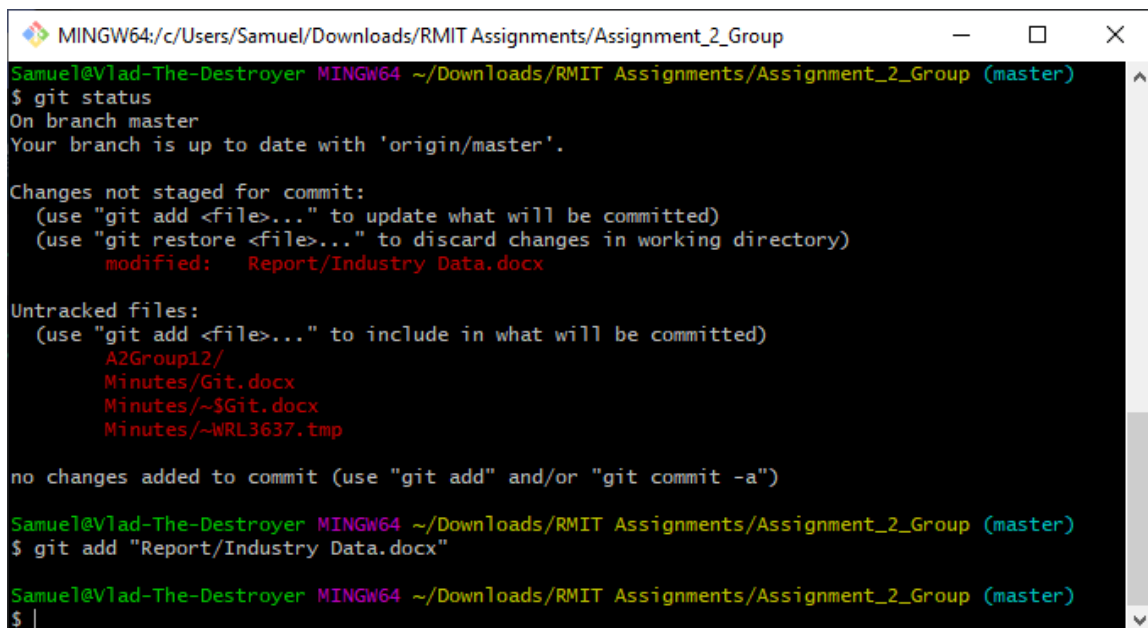
```
MINGW64: c:/Users/Samuel/Downloads/RMIT Assignments/Assignment_2_Group
Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Report/Industry Data.docx

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        A2Group12/
        Minutes/Git.docx
        Minutes/~$Git.docx
        Minutes/~WRL3637.tmp

no changes added to commit (use "git add" and/or "git commit -a")
```

You can either type the specific file name/s you want to add to the staging area or you can use wildcards, for example, `git add *.docx` will add all files with the .docx file extension (regular Microsoft Word document) that have changes (including new and removed files). Another example is simply `git add .` which will add all files with changes in the directory regardless of file type.

A terminal window showing the same 'git status' output as the previous screenshot, followed by the execution of 'git add "Report/Industry Data.docx"' and a new prompt line. The window title and initial status output are identical to the previous screenshot.

```
MINGW64: c:/Users/Samuel/Downloads/RMIT Assignments/Assignment_2_Group
Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)
$ 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 restore <file>..." to discard changes in working directory)
        modified:   Report/Industry Data.docx

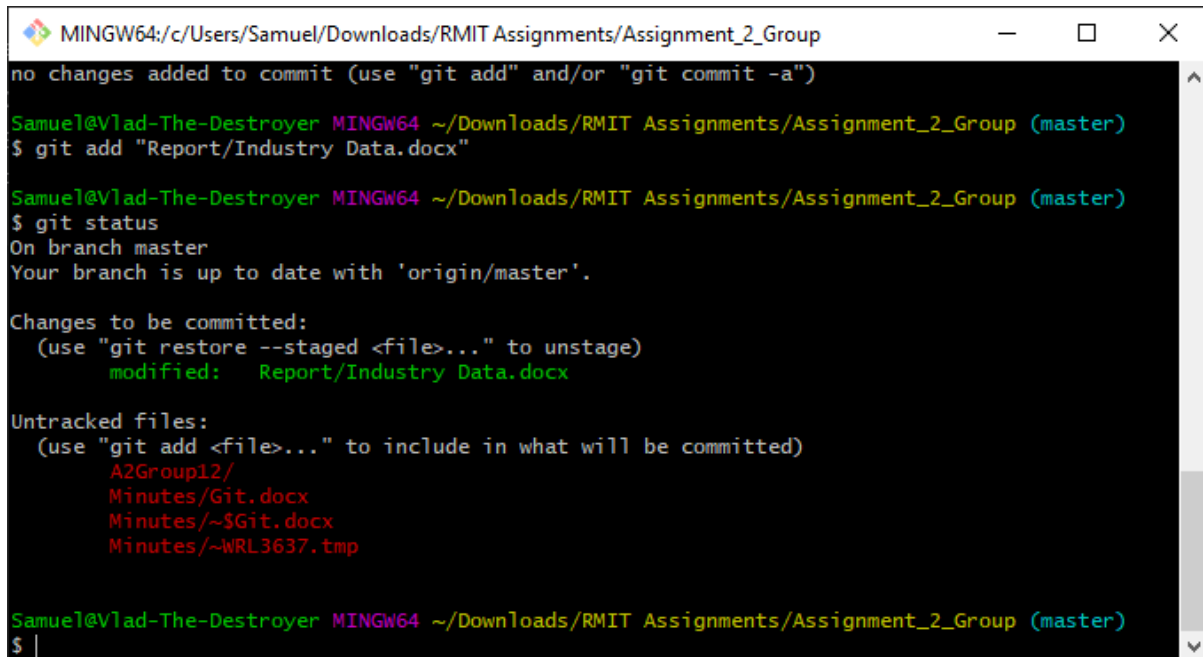
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        A2Group12/
        Minutes/Git.docx
        Minutes/~$Git.docx
        Minutes/~WRL3637.tmp

no changes added to commit (use "git add" and/or "git commit -a")

Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)
$ git add "Report/Industry Data.docx"

Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)
$ |
```

Note there is no confirmation that the file is added but we can input another status command to see the changes:

A screenshot of a terminal window titled 'MINGW64: c:/Users/Samuel/Downloads/RMIT Assignments/Assignment_2_Group'. The terminal shows the following sequence of commands and output:
1. Initial state: 'no changes added to commit (use "git add" and/or "git commit -a")'.
2. Command: '\$ git add "Report/Industry Data.docx"'.
3. Command: '\$ git status'.
4. Output: 'On branch master', 'Your branch is up to date with 'origin/master''.
5. Section: 'Changes to be committed:' followed by '(use "git restore --staged <file>..." to unstage)' and 'modified: Report/Industry Data.docx' in green text.
6. Section: 'Untracked files:' followed by '(use "git add <file>..." to include in what will be committed)' and a list of untracked files in red text: 'A2Group12/', 'Minutes/Git.docx', 'Minutes/~\$Git.docx', and 'Minutes/~WRL3637.tmp'.
7. The prompt '\$ |' is visible at the bottom.

```
MINGW64: c:/Users/Samuel/Downloads/RMIT Assignments/Assignment_2_Group
no changes added to commit (use "git add" and/or "git commit -a")

Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)
$ git add "Report/Industry Data.docx"

Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   Report/Industry Data.docx

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    A2Group12/
    Minutes/Git.docx
    Minutes/~$Git.docx
    Minutes/~WRL3637.tmp

Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)
$ |
```

Notice how there is now a green text for 'changes to be committed' which is the file we just added!

Command 4

Commit

Git commit simply commits to the changes made and updates the files in git, which will then allow us to push the changes to the remote repo (GitHub).

If you just type 'git commit', a notepad window will open which allows you to leave notes regarding the changes made – think of it as being able to add a change log entry each time you commit to a change in one or more files.

To make things a bit easier (unless you have a big note to leave) you can add the option **-m <insert note here>** to add your note in the command which prevents the notepad window opening – the note is still added all the same!

```
MINGW64/c/Users/Samuel/Downloads/RMIT Assignments/Assignment_2_Group
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Report/Industry Data.docx

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        A2Group12/
        Minutes/Git.docx
        Minutes/~$Git.docx
        Minutes/~WRL3637.tmp

Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)
$ git commit -m 'Removed repeated information from the 2nd page of Industry Data.docx'
[master b17c1da] Removed repeated information from the 2nd page of Industry Data.docx
1 file changed, 0 insertions(+), 0 deletions(-)
rewrite Report/Industry Data.docx (75%)

Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)
$ |
```

If we look at git status again now, we will see our local repo is ahead of the remote repository (GitHub) and thus is ready to be pushed (uploaded) or published to the remote repo.

```
MINGW64/c/Users/Samuel/Downloads/RMIT Assignments/Assignment_2_Group

Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)
$ git commit -m 'Removed repeated information from the 2nd page of Industry Data.docx'
[master b17c1da] Removed repeated information from the 2nd page of Industry Data.docx
1 file changed, 0 insertions(+), 0 deletions(-)
rewrite Report/Industry Data.docx (75%)

Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        A2Group12/
        Minutes/Git.docx
        Minutes/~$Git.docx
        Minutes/~WRL3637.tmp

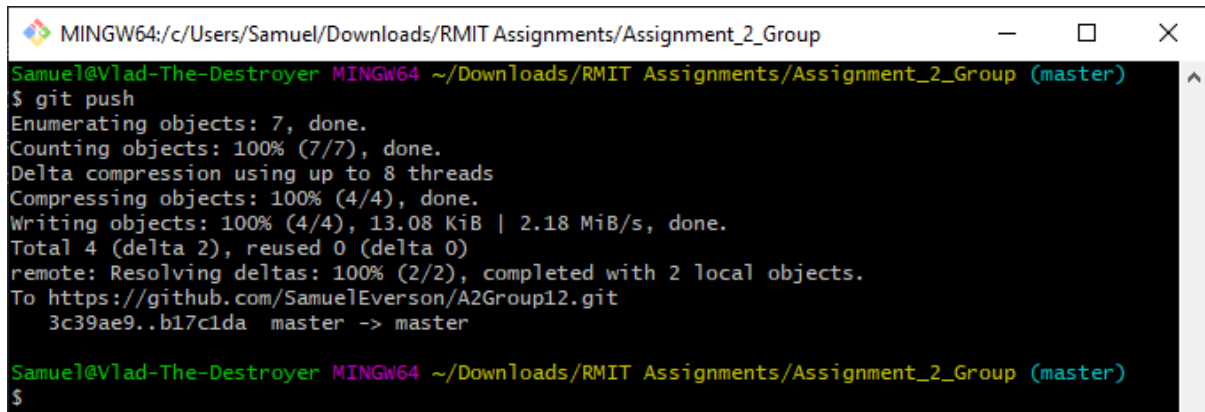
nothing added to commit but untracked files present (use "git add" to track)

Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)
$ |
```

Command 5

Push

Finally, as mentioned earlier, `git push` will publish or upload your changes to the remote repo. This then makes your changes visible to other members of the remote repo (or anyone that accesses the files). For other users to see your changes, they would need to pull the repo again, after you have pushed your changes that is.

A screenshot of a terminal window titled 'MINGW64:/c/Users/Samuel/Downloads/RMIT Assignments/Assignment_2_Group'. The prompt is 'Samuel@Vlad-The-Destroyer MINGW64 ~/Downloads/RMIT Assignments/Assignment_2_Group (master)'. The user enters '\$ git push'. The output shows the process of enumerating, counting, compressing, and writing objects to the remote repository. It indicates that 4 objects (delta 2) were pushed, reusing 0 objects. The commit is resolved and pushed to the master branch on the remote repository at 'https://github.com/SamuelEverson/A2Group12.git' with the commit hash '3c39ae9..b17c1da'. The prompt returns to '\$'.

And that's it!

5 (and a few other) commands to master git!