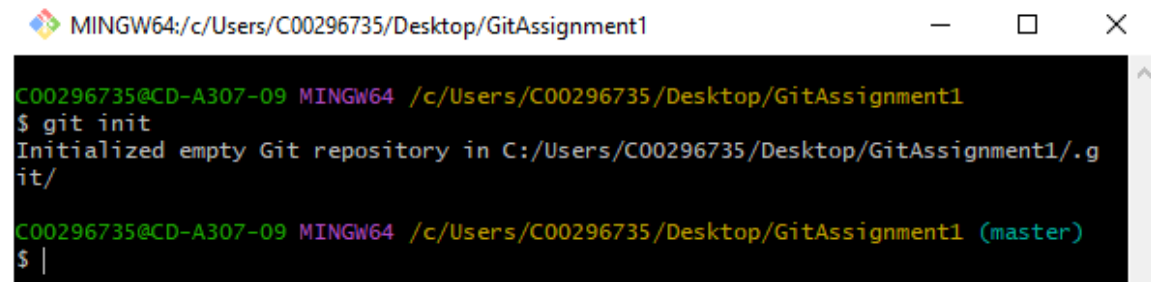


Git Init

Creates a new repository within a folder (appears as hidden .git file when view hidden files is enabled). This allows you to continue with other commands.

Syntax:

git init (Creates a new local repository within the current folder)



```
MINGW64:/c:/Users/C00296735/Desktop/GitAssignment1

C00296735@CD-A307-09 MINGW64 /c:/Users/C00296735/Desktop/GitAssignment1
$ git init
Initialized empty Git repository in C:/Users/C00296735/Desktop/GitAssignment1/.git/

C00296735@CD-A307-09 MINGW64 /c:/Users/C00296735/Desktop/GitAssignment1 (master)
$ |
```

Git Status

Displays the status of files within the folder, such as new files, modified files and files that are added/staged. New/modified files are displayed as red and added/staged files are shown as green.

Syntax:

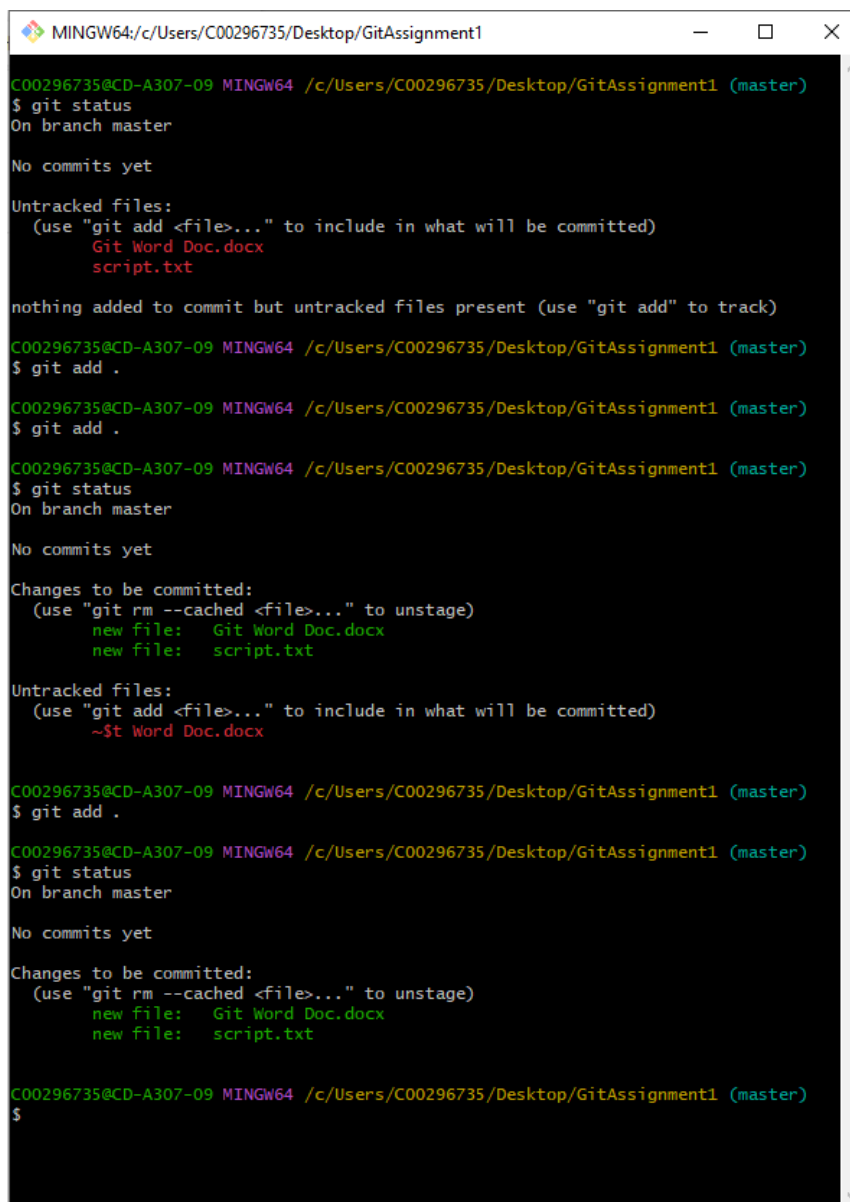
git status (Lists all modified, added and current files)

Git Add

Adds new/modified files to the index and stages them to be committed using the git add command. It is advisable to do another git status command to confirm that the git add was successful. MUST BE DONE BEFORE COMMIT.

Syntax:

git add <file name/.> (Adds the selected file or all files to be staged for commit)

A screenshot of a terminal window titled 'MINGW64: c:/Users/C00296735/Desktop/GitAssignment1'. The terminal shows a series of Git commands and their outputs. The first command is 'git status', which shows 'On branch master', 'No commits yet', and 'Untracked files: Git Word Doc.docx, script.txt'. The next two commands are 'git add .' and 'git add .', which stage the files. The third 'git status' command shows 'Changes to be committed: new file: Git Word Doc.docx, new file: script.txt' and 'Untracked files: ~\$t Word Doc.docx'. The final two 'git add .' commands stage the remaining file. The last 'git status' command shows the same committed files and untracked file. The terminal text is as follows:

```
C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Git Word Doc.docx
        script.txt

nothing added to commit but untracked files present (use "git add" to track)
C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (master)
$ git add .
C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (master)
$ git add .
C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Git Word Doc.docx
        new file:   script.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ~$t Word Doc.docx

C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (master)
$ git add .
C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Git Word Doc.docx
        new file:   script.txt

C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (master)
$
```

Git Commit

Creates a new commit using the staged files within the index and a log message documenting the changes. This commit will be able to be seen on GitHub when git remote is used later.

Syntax:

git commit (Enters editor mode/multiline comment)

Git Log

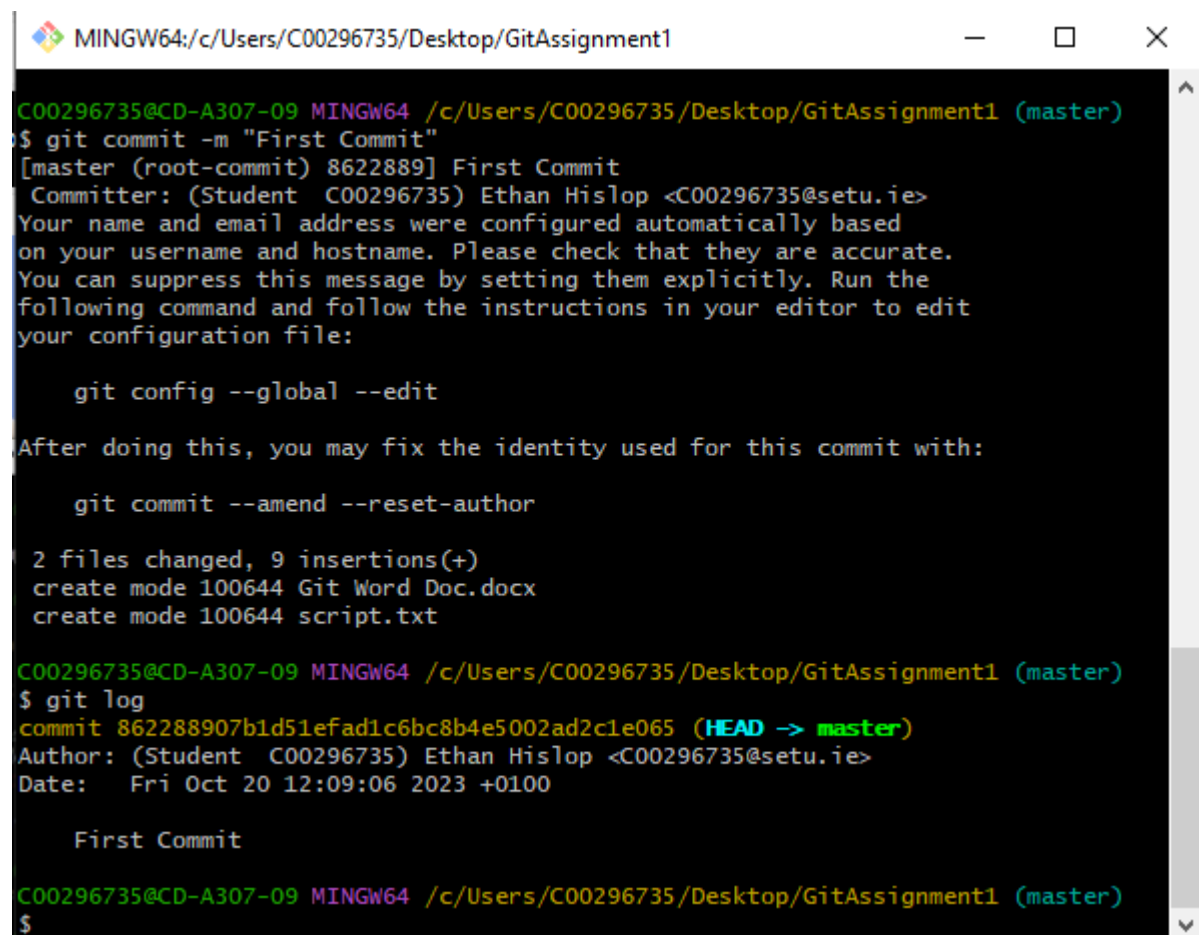
Shows the logs within a commit, such as message logs included with a commit command (as shown below).

Syntax:

git log (Display the entire commit history) [<space> for more and <q> to exit]

git log -stat (Shows the altered files and line changed)

git log -graph (Draws up branch paths)



```
MINGW64:/c:/Users/C00296735/Desktop/GitAssignment1
C00296735@CD-A307-09 MINGW64 /c:/Users/C00296735/Desktop/GitAssignment1 (master)
$ git commit -m "First Commit"
[master (root-commit) 8622889] First Commit
  Committer: (Student C00296735) Ethan Hislop <C00296735@setu.ie>
  Your name and email address were configured automatically based
  on your username and hostname. Please check that they are accurate.
  You can suppress this message by setting them explicitly. Run the
  following command and follow the instructions in your editor to edit
  your configuration file:

    git config --global --edit

  After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

  2 files changed, 9 insertions(+)
  create mode 100644 Git Word Doc.docx
  create mode 100644 script.txt

C00296735@CD-A307-09 MINGW64 /c:/Users/C00296735/Desktop/GitAssignment1 (master)
$ git log
commit 862288907b1d51efad1c6bc8b4e5002ad2c1e065 (HEAD -> master)
Author: (Student C00296735) Ethan Hislop <C00296735@setu.ie>
Date:   Fri Oct 20 12:09:06 2023 +0100

    First Commit

C00296735@CD-A307-09 MINGW64 /c:/Users/C00296735/Desktop/GitAssignment1 (master)
$
```

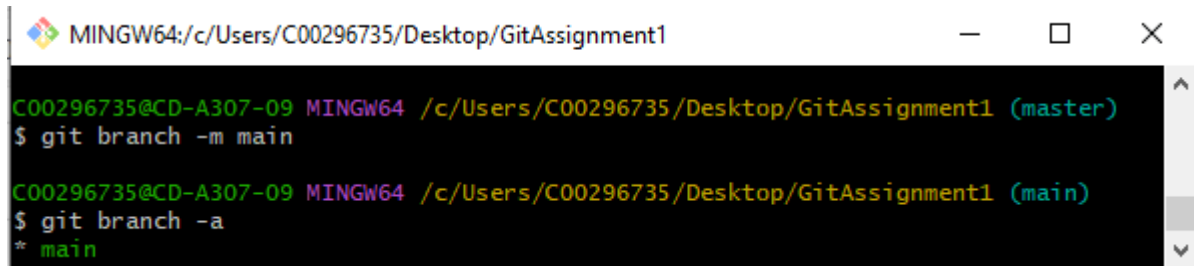
Git Branch

Creates a new branch within the current local repository. You can also use a command to list all the current branches within the current repository. The default branch is called either master or main, below it is remained from master to main.

Syntax:

`git branch <name>` (Creates a new branch in the current local repository)

`git branch -a/-l/-r` (Lists all branches within the repository)

A terminal window titled 'MINGW64:/c/Users/C00296735/Desktop/GitAssignment1' showing the execution of git branch commands. The prompt is 'C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (master)'. The first command is '\$ git branch -m main', which changes the current branch to 'main'. The second command is '\$ git branch -a', which lists the current branch as '* main'.

```
MINGW64:/c/Users/C00296735/Desktop/GitAssignment1
C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (master)
$ git branch -m main
C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (main)
$ git branch -a
* main
```

Git Remote

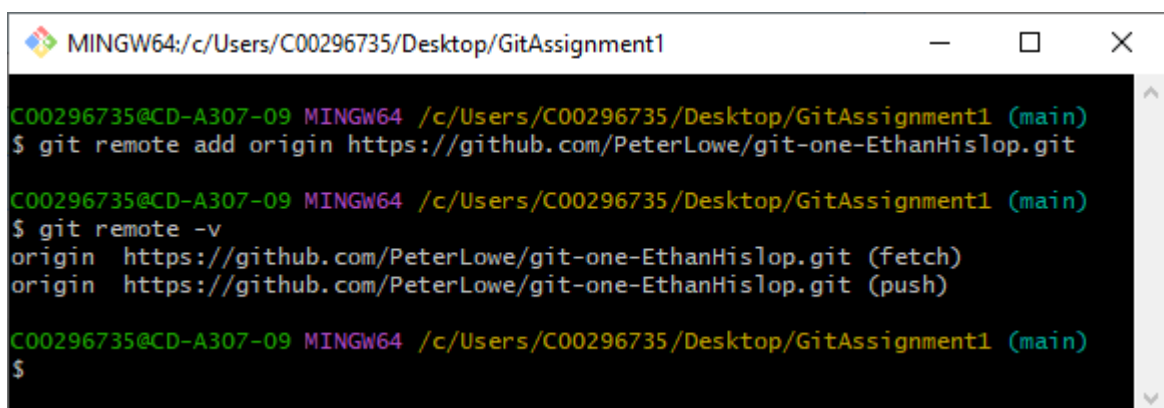
Allows the creation of a remote repository within the current folder. Also allows you to list and delete these remote repositories if necessary. This will allow you to view your work and commits up on GitHub when they are pushed with the git push command (see next command).

Syntax:

`git remote add <name> <URL>` (Creates a new remote repository)

`git remote -v` (Lists all repositories on remote server)

`git remote rm <name>` (Delete an existing remote repository)

A terminal window titled 'MINGW64:/c/Users/C00296735/Desktop/GitAssignment1' showing the execution of git remote commands. The prompt is 'C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (main)'. The first command is '\$ git remote add origin https://github.com/PeterLowe/git-one-EthanHislop.git'. The second command is '\$ git remote -v', which lists the remote repository 'origin' with its URL and the operations 'fetch' and 'push'.

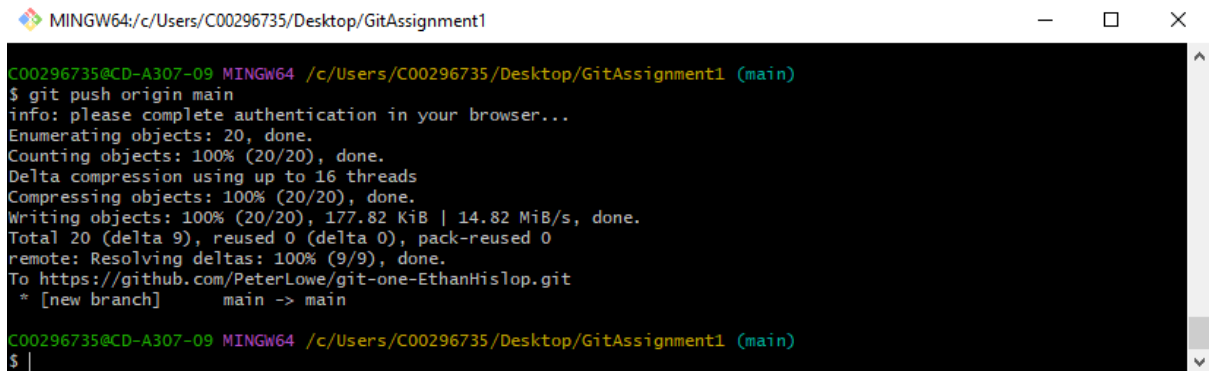
```
MINGW64:/c/Users/C00296735/Desktop/GitAssignment1
C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (main)
$ git remote add origin https://github.com/PeterLowe/git-one-EthanHislop.git
C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (main)
$ git remote -v
origin https://github.com/PeterLowe/git-one-EthanHislop.git (fetch)
origin https://github.com/PeterLowe/git-one-EthanHislop.git (push)
C00296735@CD-A307-09 MINGW64 /c/Users/C00296735/Desktop/GitAssignment1 (main)
$
```

Git Push

Uploads all local commits within the repository/branch to the matching remote repository/branch. These can then be viewed through GitHub if the command was successful.

Syntax:

`git push <remote> <branch>` (Pushes the given remote and branch commit up to GitHub)

A terminal window titled 'MINGW64:/c:/Users/C00296735/Desktop/GitAssignment1' showing the execution of 'git push origin main'. The output shows the process of enumerating, counting, compressing, and writing objects, followed by resolving deltas and pushing to the remote repository 'https://github.com/PeterLowe/git-one-EthanHislop.git'. The push is successful, creating a new branch 'main' on the remote.

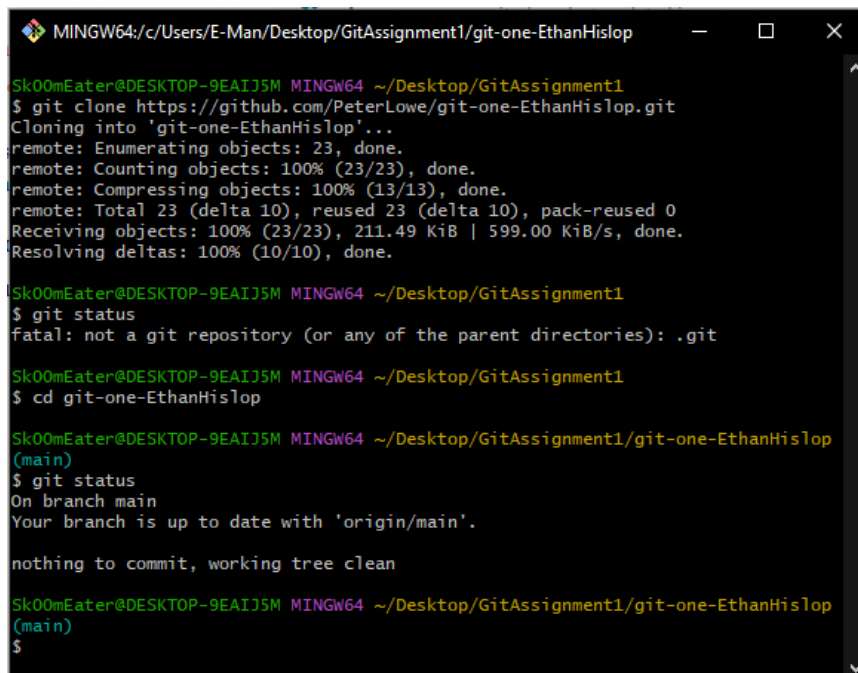
```
MINGW64:/c:/Users/C00296735/Desktop/GitAssignment1
C00296735@CD-A307-09 MINGW64 /c:/Users/C00296735/Desktop/GitAssignment1 (main)
$ git push origin main
info: please complete authentication in your browser...
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 16 threads
Compressing objects: 100% (20/20), done.
Writing objects: 100% (20/20), 177.82 KiB | 14.82 MiB/s, done.
Total 20 (delta 9), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (9/9), done.
To https://github.com/PeterLowe/git-one-EthanHislop.git
 * [new branch]      main -> main
C00296735@CD-A307-09 MINGW64 /c:/Users/C00296735/Desktop/GitAssignment1 (main)
$
```

Git Clone

Clones an existing repository (usually a remote one) into a new directory (perhaps on another file system or computer).

Syntax:

`git clone <URL>` (Clones the remote repository within the indicated URL)

A terminal window titled 'MINGW64:/c:/Users/E-Man/Desktop/GitAssignment1/git-one-EthanHislop' showing the execution of 'git clone https://github.com/PeterLowe/git-one-EthanHislop.git'. The output shows the cloning process, including enumerating, counting, and compressing objects. After cloning, the user runs 'git status' in the parent directory, which shows a fatal error because it's not a git repository. Then, the user runs 'cd git-one-EthanHislop' and 'git status' again, which shows they are on the 'main' branch and the working tree is clean.

```
MINGW64:/c:/Users/E-Man/Desktop/GitAssignment1/git-one-EthanHislop
Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1
$ git clone https://github.com/PeterLowe/git-one-EthanHislop.git
Cloning into 'git-one-EthanHislop'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 23 (delta 10), reused 23 (delta 10), pack-reused 0
Receiving objects: 100% (23/23), 211.49 KiB | 599.00 KiB/s, done.
Resolving deltas: 100% (10/10), done.

Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1
$ git status
fatal: not a git repository (or any of the parent directories): .git

Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1
$ cd git-one-EthanHislop

Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop
(main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

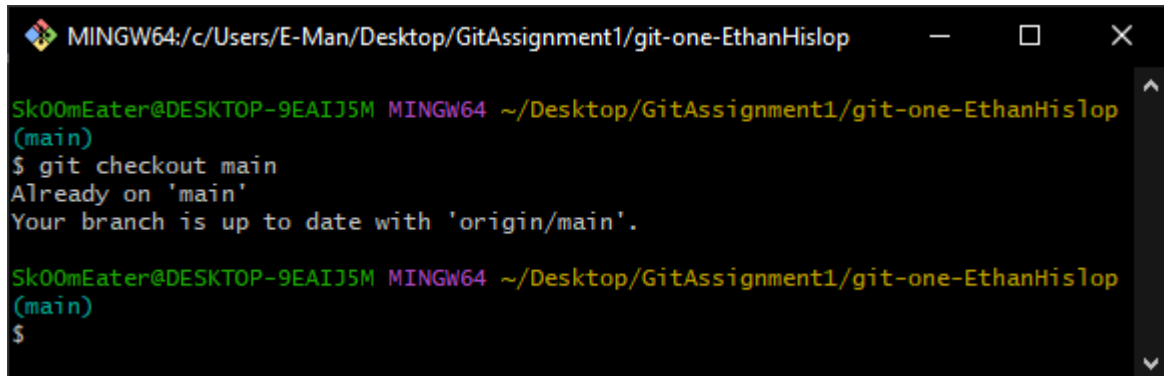
Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop
(main)
$
```

Git Checkout

Allows the switching between different branches within a repository or, more accurately, makes the branch/node the current head. As shown in the screenshot below the head is assigned to main so no change is needed to be made (there are no other branches within this repository).

Syntax:

git checkout <branch_name> (Makes this node the current head)



```
MINGW64:/c/Users/E-Man/Desktop/GitAssignment1/git-one-EthanHislop
Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop
(main)
$ git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.

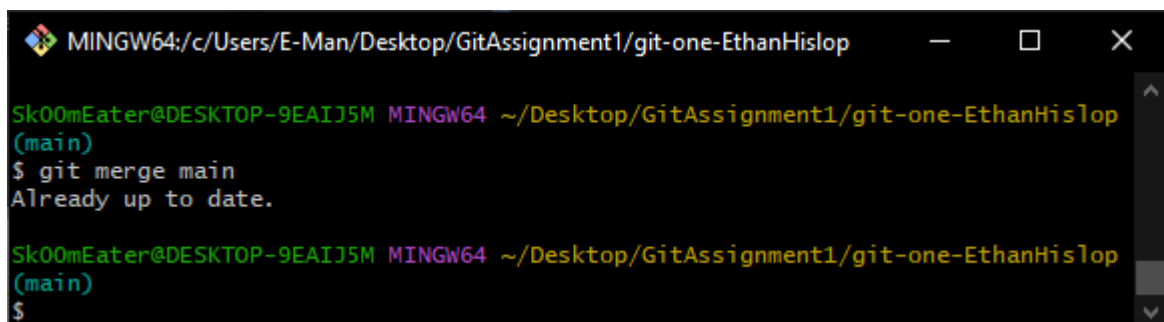
Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop
(main)
$
```

Git Merge

Allows the current assigned head/branch of the repository to be merged with the main/master branch. Useful for combining two different branches, which is what this command is most used for. The below screenshot shows that everything is up to date (as there is only one branch in this repository, which is main/master).

Syntax:

git merge master/main (Merge the current head with main/master branch, git auto applies changes)



```
MINGW64:/c/Users/E-Man/Desktop/GitAssignment1/git-one-EthanHislop
Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop
(main)
$ git merge main
Already up to date.

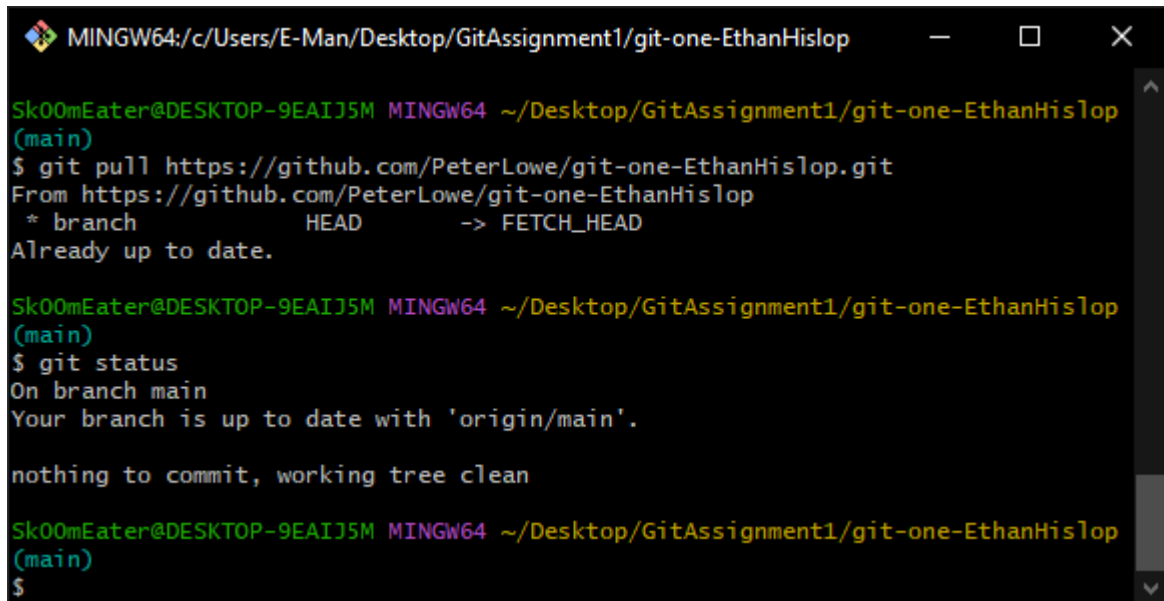
Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop
(main)
$
```

Git Pull

This command performs a fetch and merge command at the same time. It will first fetch files from the remote repository then it will merge them with the current head/branch. As shown below, no changes were made in this case as the local repository and the remote repository are both the same and up to date.

Syntax:

`git pull <remote>` (Perform a fetch and a merge)

A screenshot of a Windows Command Prompt window titled "MINGW64:/c/Users/E-Man/Desktop/GitAssignment1/git-one-EthanHislop". The prompt shows a user named "Sk00mEater@DESKTOP-9EAIJ5M" in the "MINGW64" environment. The user is in the directory "~/Desktop/GitAssignment1/git-one-EthanHislop" on the "main" branch. They run the command `$ git pull https://github.com/PeterLowe/git-one-EthanHislop.git`. The output shows the remote repository being fetched from `https://github.com/PeterLowe/git-one-EthanHislop`, with a mapping of `* branch HEAD -> FETCH_HEAD`. The result is "Already up to date." The user then runs `$ git status`, which reports "On branch main" and "Your branch is up to date with 'origin/main'." Finally, it states "nothing to commit, working tree clean". The prompt ends with a dollar sign `$`.

```
Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop
(main)
$ git pull https://github.com/PeterLowe/git-one-EthanHislop.git
From https://github.com/PeterLowe/git-one-EthanHislop
 * branch      HEAD       -> FETCH_HEAD
Already up to date.

Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop
(main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop
(main)
$
```

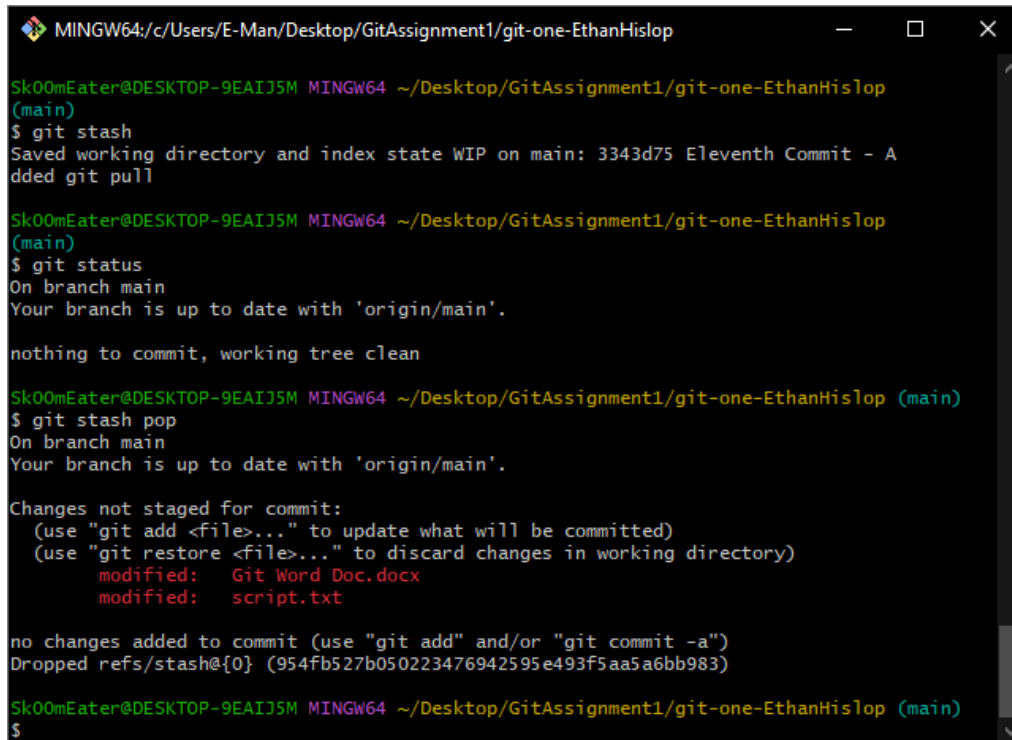
Git Stash

Shelves changes made to the current working directory so another thing can be worked on, allowing you to come back and reapply these changes later. CAN ONLY BE DONE BEFORE COMMIT NOT AFTER

Syntax:

git stash (Stashes away an un-staged or staged, but not committed, work to be reapplied later)

git stash pop (Removes the stash changes and applies them to your working directory)

A screenshot of a Windows terminal window titled "MINGW64:/c/Users/E-Man/Desktop/GitAssignment1/git-one-EthanHislop". The terminal shows a user named "Sk00mEater@DESKTOP-9EAIJ5M" performing several Git commands. First, they run "git stash", which saves the working directory and index state to a new stash on the 'main' branch. Then, they run "git status", which shows that the branch is up to date and the working tree is clean. Finally, they run "git stash pop", which removes the stash and applies the changes back to the working directory. The terminal output shows that the changes are not staged for commit and that the refs/stash@{0} has been dropped.

```
Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop
(main)
$ git stash
Saved working directory and index state WIP on main: 3343d75 Eleventh Commit - Added git pull

Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop
(main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop (main)
$ git stash pop
On branch main
Your branch is up to date with 'origin/main'.

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:   Git Word Doc.docx
        modified:   script.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (954fb527b050223476942595e493f5aa5a6bb983)

Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop (main)
$
```

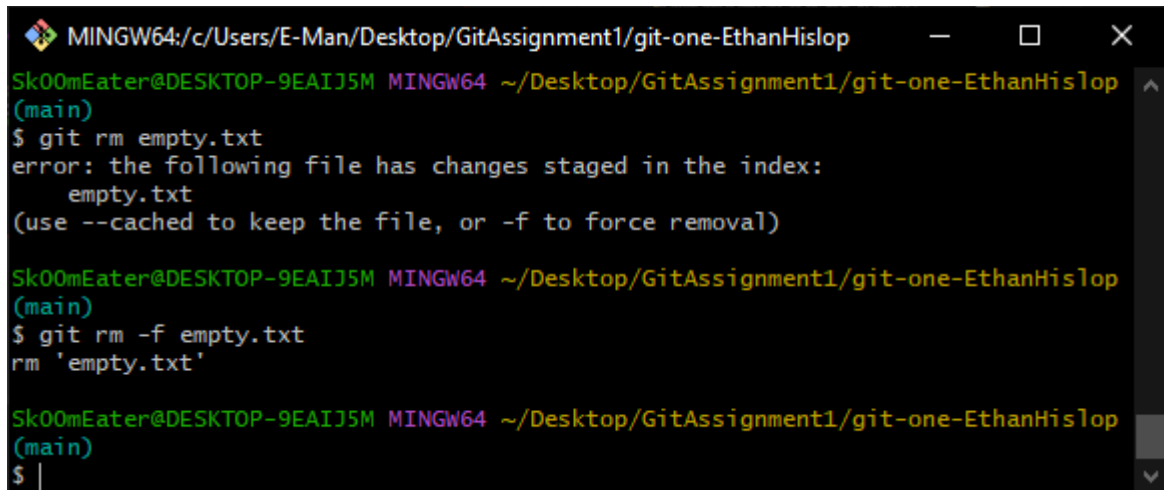

Git RM

This can be used to remove one or multiple files from a directory, mainly tracked ones from the index, staging index and the local working repository. This applies to all of these and can't be used for only one.

Syntax:

`git rm <file name>` (Removes file from repository)

`git rm -f <file name>` (Removes file from repository by force, disregarding errors)

A screenshot of a Windows terminal window with a black background and white text. The window title is 'MINGW64:/c/Users/E-Man/Desktop/GitAssignment1/git-one-EthanHislop'. The prompt is 'Sk00mEater@DESKTOP-9EAIJ5M MINGW64 ~/Desktop/GitAssignment1/git-one-EthanHislop (main)'. The user enters '\$ git rm empty.txt'. The terminal shows an error: 'error: the following file has changes staged in the index: empty.txt (use --cached to keep the file, or -f to force removal)'. The user then enters '\$ git rm -f empty.txt' and the terminal shows 'rm 'empty.txt''. The prompt returns to '\$ |'.