

## Init

Git init is used to initialise a git repository

```
C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One
$ git init
Initialized empty Git repository in C:/Users/C00295678/Desktop/Git Lab One/.git/
```

It creates the repository in the folder directory GitBash was opened in

It also creates a new folder as a hidden item

NAME	DATE MODIFIED	TYPE	SIZE
.git	20/10/2023 11:36	File folder	
git report	20/10/2023 11:40	Microsoft Word D...	0 KB
README	20/10/2023 11:26	Markdown Source...	1 KB
script.txt	20/10/2023 11:40	Text Document	0 KB

This .git folder indicates that this is the folder where the repository was created.

Syntax:

Git init

## Add

Git add sends files to the “staging area”. In this area, they are prepared to be committed. Add should be used anytime a file has been updated by the user and hasn’t been updated in the repository. This includes any new files introduced, and any previous files that have been updated. The user can specify any file within their folder by typing the filename after “git add”, and they can add all files in the folder by using a “.”

Syntax:

Git add <filename> / .

```
C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One (main)
$ git add script.txt.txt

C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One (main)
$ git add .
```

```
C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One (main)
$ git status
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 report.docx
        modified:   script.txt.txt
```

## Status

Git status is used to display the current status of files in the source folder and the staging area. It will display what files are yet to be added and which files are yet to be committed. A red file is one that hasn't been added to the staging area yet, while green files are files that exist in the staging area, but are yet to be committed. Git status should be used following any add or commit done by the user to verify the process has been completed correctly.

Syntax:

Git status

```
C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   git report.docx
        modified:   script.txt.txt

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 report.docx
        modified:   script.txt.txt
```

## Commit

Git commit is used to fully introduce a file from the staging area into the repository. As stated previously, a git status will show uncommitted files as green. Committed files won't appear when a git status is called, a signal that the commit was successful. Unlike an add, however, a commit must come with a message. The line must read "git commit -m "<commit title>". This means that when the commit is viewed in the repository, there is a title included. If this is not done, git will open up VI, a program that is much harder to use than a "-m" function.

Syntax:

Git commit <filename>

Git commit .

Git commit <filename> -m "comment"

Example below:

```

C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   git report.docx
        modified:   script.txt.txt

C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One (main)
$ git commit -m "commit"
[main db739e0] commit
Committer: (Student C00295678) Stephen Arthur <C00295678@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, 2 insertions(+), 4 deletions(-)

C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

```

## Remote

Git remote creates a remote version of a repository. This version of a repository is common among multiple coders, and will be held on an online host like GitHub, or a more private host for a large corporation. While the users will have a local repository to work on, they must push their changes up to the remote, even if those changes are committed to the local repository. A remote is added with the command "git remote add origin remote\_directory\_url" This version of the repository is updated by multiple users separately, and as such is used for working in a group.

Syntax:

Git remote add origin <repositorydirectory>

```

C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One (main)
$ git remote add origin https://github.com/PeterLowe/git-one-stephenarthur64.git

```

## Push

Git push is used to push up the local changes to a repository to the associated remote. The -u function must be used in order to associate the push request with the remote the changes are being pushed to. A push must always be done when git is being closed for the day, so as to protect the files being used. Pushes should also be done often so that, in the event of something like a crash, the minimum amount of work is lost.

Syntax:

Git push -u (associates) origin <branchname>

```
C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One (main)
$ git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 13.04 KiB | 13.04 MiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/PeterLowe/git-one-stephenarthur64.git
   c09a707..b28b8c6  main -> main
branch 'main' set up to track 'origin/main'.
```

## Branch

Git branch creates a new branch in the local repository where files can be added. A branch is a string of nodes that acts parallel to the main or master branch. A new branch can be made with the “git branch <branchname>” command. This new branch can be swapped to using the checkout command, and can now add and commit files to this branch. When this branch is pushed to the remote, it will be prompted to merge with the main branch.

Syntax:

Git branch <branchname>

```
C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One (main)
$ git branch test
```

## Checkout

Git checkout is used to swap between nodes and branches. It can be used to change which node or branch is being updated with files. The command “git checkout <node or branch>” is used to swap the section being used. It can also be used to create new branches with “git checkout -b <branch name>”.

Syntax:

Git checkout <branchname>

Git checkout -b <branch name> (makes new branch)

```
C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One (main)
$ git checkout test
Switched to branch 'test'
Your branch is up to date with 'origin/test'.
```

```
C00295678@CD-A307-20 MINGW64 /c/Users/C00295678/Desktop/Git Lab One (main)
$ git checkout -b test2
Switched to a new branch 'test2'
```

## Merge

The merge function in git is used to merge 2 branches together. This function is useful for re-incorporating parallel developments to a project but must be done with caution. If there is a conflict in the merge, like two identical variable names, the merge is stopped. The merge is done by git applying the commits done to the parallel branch to the branch being merged into. If conflict occurs, --abort walks back the merge, while --continue keep it going. This function is used often to ensure a common project. It must also be done regularly for small changes, to prevent a massive merge that results in far too many conflicts than can reasonably fixed.

Syntax:

Git merge <branch1> <branch2>

```
stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/OneDrive_2023-10-21/Git Lab 1/Git Lab One (main)
$ git merge test main
Merge made by the 'ort' strategy.
 READIT.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 READIT.md
```

## Log

The log function is used to view the history of commits in the repository. It displays every commit that has been made. This display only shows a limited number of commits, so it is navigate by using the space key, and the user can quit using the q key. Variations like --stat show the individual changes made in each commit, while --graph displays the information in a more graphed format, and -n shows the last n commits. This function is useful for tracking previous commits, and is also one of the reasons the user must always name commits, so as to make the log clearer.

Git log

Git log --stat

Git log --graph

```
stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git Lab 1/Git Lab One (main)
$ git log
commit 12eca08cc6ce9d1272794887eec281916cbd1ad9 (HEAD -> main, origin/main)
Author: Lucy Arthur <C00295678@setu.ie>
Date: Sat Oct 21 15:30:18 2023 +0100

    merge

commit aeaeb1a442ced7cc424b443c04b78741a18f95f9
Merge: 12ce56c e51f985
Author: Lucy Arthur <C00295678@setu.ie>
Date: Sat Oct 21 15:22:10 2023 +0100

    Merge branch 'test'

commit 12ce56c9de5a3821a63c09077f2f59cbde08412b
Author: (Student C00295678) Stephen Arthur <C00295678@setu.ie>
Date: Fri Oct 20 12:57:36 2023 +0100

    checkout

commit 6b3d37cd5f1f112690fa95920c6bafba84a26443 (test2)
Author: (Student C00295678) Stephen Arthur <C00295678@setu.ie>
Date: Fri Oct 20 12:51:53 2023 +0100

    branch
```

```

stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git Lab 1/Git Lab One (main)
$ git log --stat
commit 12eca08cc6ce9d1272794887eec281916cbd1ad9 (HEAD -> main, origin/main)
Author: Lucy Arthur <C00295678@setu.ie>
Date: Sat Oct 21 15:30:18 2023 +0100

    merge

git report.docx | Bin 155436 -> 172963 bytes
script.txt.txt | 4 ++++
~$t report.docx | Bin 162 -> 0 bytes
~WRL0005.tmp | 0
"/357\200\272" | 6 ++++++
5 files changed, 10 insertions(+)

commit aeaeb1a442ced7cc424b443c04b78741a18f95f9
Merge: 12ce56c e51f985
Author: Lucy Arthur <C00295678@setu.ie>
Date: Sat Oct 21 15:22:10 2023 +0100

    Merge branch 'test'

commit 12ce56c9de5a3821a63c09077f2f59cbde08412b
Author: (Student C00295678) Stephen Arthur <C00295678@setu.ie>
Date: Fri Oct 20 12:57:36 2023 +0100

    checkout

git report.docx | Bin 141348 -> 155436 bytes
script.txt.txt | 5 +++++
2 files changed, 4 insertions(+), 1 deletion(-)

```

```

stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git Lab 1/Git Lab One (main)
$ git log --graph
* commit 12eca08cc6ce9d1272794887eec281916cbd1ad9 (HEAD -> main, origin/main)
  | Author: Lucy Arthur <C00295678@setu.ie>
  | Date: Sat Oct 21 15:30:18 2023 +0100
  |
  | merge
  |
  | * commit aeaeb1a442ced7cc424b443c04b78741a18f95f9
  |   | Merge: 12ce56c e51f985
  |   | Author: Lucy Arthur <C00295678@setu.ie>
  |   | Date: Sat Oct 21 15:22:10 2023 +0100
  |   |
  |   | Merge branch 'test'
  |   |
  |   | * commit e51f985b01fb9b5936d977d45d887951bc999ecd (origin/test, test)
  |   |   | Author: (Student C00295678) Stephen Arthur <C00295678@setu.ie>
  |   |   | Date: Fri Oct 20 12:44:40 2023 +0100
  |   |   |
  |   |   | readit
  |   |   |
  |   |   | * commit 12ce56c9de5a3821a63c09077f2f59cbde08412b
  |   |   |   | Author: (Student C00295678) Stephen Arthur <C00295678@setu.ie>
  |   |   |   | Date: Fri Oct 20 12:57:36 2023 +0100
  |   |   |   |
  |   |   |   | checkout
  |   |   |   |
  |   |   |   | * commit 6b3d37cd5f1f112690fa95920c6bafba84a26443 (test2)
  |   |   |   |   | Author: (Student C00295678) Stephen Arthur <C00295678@setu.ie>
  |   |   |   |   | Date: Fri Oct 20 12:51:53 2023 +0100
  |   |   |   |   |
  |   |   |   |   | branch
  |   |   |   |   |
  |   |   |   |   |

```

## Clone

Git clone creates a complete clone of a repository that can be independently updated and changed. It operates as its own repository. The clone can then be merged back into the main repository with a simple pull command. The clone function can be useful in the even that the user wants to make a series of changes to a project without interrupting the main, or if the user wants to work remotely on a machine apart from the codebase.

Syntax:



















Git clone <repository>

Example below:

```

stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git/Git Lab One (main)
$ git clone https://github.com/PeterLowe/git-one-stephenarthur64
Cloning into 'git-one-stephenarthur64'...
remote: Enumerating objects: 60, done.
remote: Counting objects: 100% (60/60), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 60 (delta 32), reused 45 (delta 17), pack-reused 0
Receiving objects: 100% (60/60), 425.74 KiB | 198.00 KiB/s, done.
Resolving deltas: 100% (32/32), done.

```

NAME	STATUS	DATE MODIFIED	TYPE
 .git		22/10/2023 18:40	File folder
 git-one-stephenarthur64		22/10/2023 18:46	File folder
 ;		21/10/2023 15:18	File
 git report		22/10/2023 18:46	Microsoft Word Document
 READIT.md		21/10/2023 15:22	MD File
 README.md		21/10/2023 15:18	MD File
 rmtest		22/10/2023 18:34	Text Document
 script.txt		22/10/2023 18:41	Text Document
 .		21/10/2023 15:22	File

## Pull

Git pull is used to update a local repository with a remote repository. It pulls the commits made to the remote repository and applies them to the local. Options are then used to decide if these changes will rebase the local or merge with it. This command is used when the remote is ahead of the local, which may be useful if changes have been made to it and your local repository is behind. If the local is already caught up with the remote, nothing happens

Syntax:

Git pull <options> <repository>

```

stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git/Git Lab One (main)
$ git pull
Already up to date.

```

## Stash

Git stash creates a stash of a dirty node, storing it away while the user can return to a clean version of the repository. The stash can be recovered at any time with the “apply” function. This could be useful if the user wants to temporarily move back to a clean repository without losing all of their work on the node. Stashes are also named as “stash@{n}”, n being ordered with 0 as the most recent, 1 as the next most recent, etc.

Syntax:

Git stash

Git stash apply

```
stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git/Git Lab One (main)
$ git stash
Saved working directory and index state WIP on main: 7e25811 pull

stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git/Git Lab One (main)
$ git stash apply
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 report.docx

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

## Rm

Git rm removes a file from either the working directory or staging area, depending on the syntax. This command should be used with caution, as there is no warning, it will simply remove the files. The user should be sure that they use this command carefully and only when they are sure they know what the file is and what they are removing it from.

Syntax:

Git rm

Git rm <filename> --cached -> removes from staging area

Rm <filename> -> removes from working directory

```
stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git/Git Lab One (main)
$ ls
: ';'  READIT.md  README.md  'git report.docx'  rmtest.txt  script.txt.txt

stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git/Git Lab One (main)
$ git add rmtest.txt

stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git/Git Lab One (main)
$ git ls-files
;
READIT.md
README.md
git report.docx
rmtest.txt
script.txt.txt
"\357\200\272"

stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git/Git Lab One (main)
$ git rm rmtest.txt
error: the following file has changes staged in the index:
    rmtest.txt
(use --cached to keep the file, or -f to force removal)

stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git/Git Lab One (main)
$ git rm rmtest.txt --cached
rm 'rmtest.txt'

stevi@BigLittleGuy MINGW64 ~/OneDrive/Desktop/Git/Git Lab One (main)
$ git ls-files
;
READIT.md
README.md
git report.docx
script.txt.txt
"\357\200\272"
```



