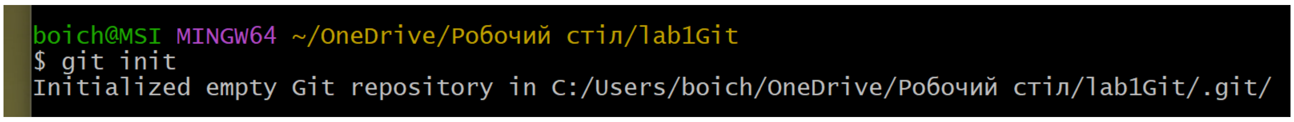
Ostap Boychuk

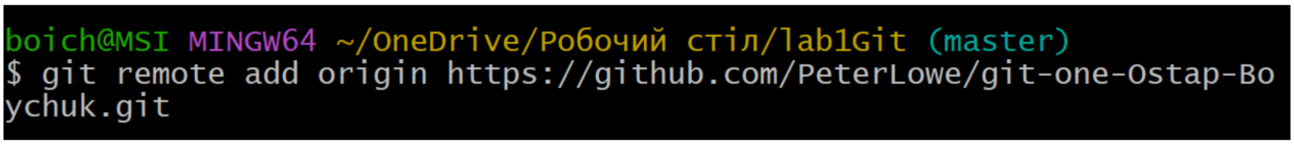
Lab 1 Git

"git init" is a command used in the Git version control system to initialise a new repository. When I run "git init" in a directory, it creates a new Git repository that allows me to track and manage changes in project files. This command sets up the necessary directories and data structures. It also creates a hidden .git directory where Git stores project information such as commit history, branches, and configuration settings.



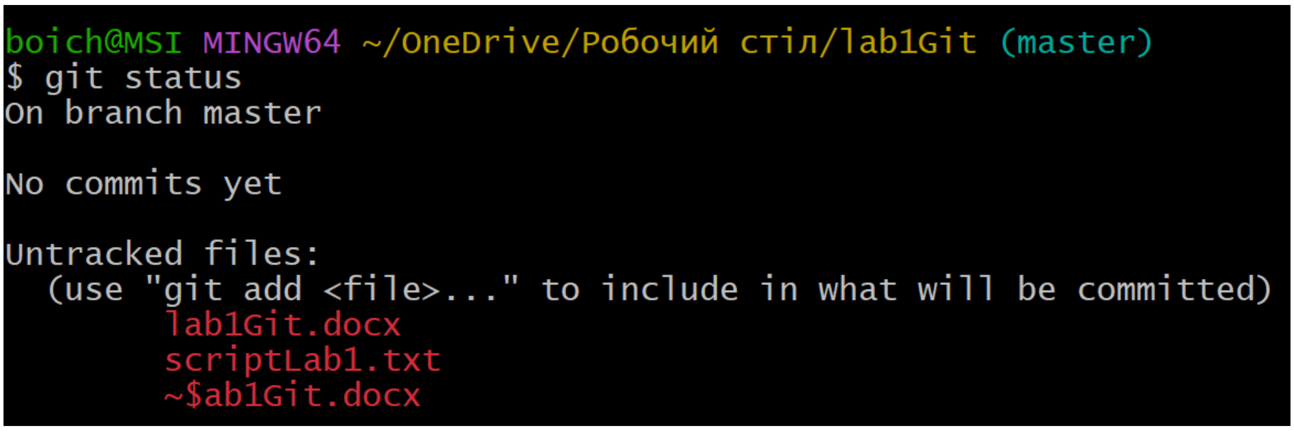
```
boich@MSI MINGW64 ~/OneDrive/Робочий стіл/lab1Git
$ git init
Initialized empty Git repository in C:/Users/boich/OneDrive/Робочий стіл/lab1Git/.git/
```

The **"git remote add origin"** command is used in Git to set up a connection between your local Git repository and a remote repository, typically hosted on a platform like GitHub or GitLab. The term "origin" is a conventional name for the remote repository, but it can be customized. When you run "git remote add origin <URL>", you're essentially telling Git where to find the remote repository.



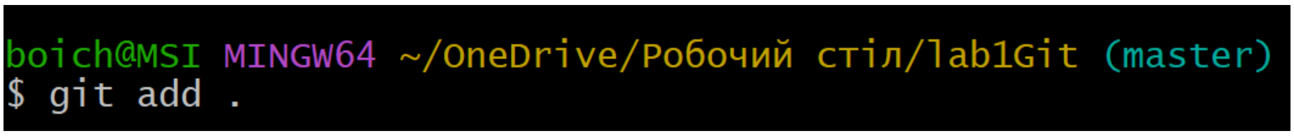
```
boich@MSI MINGW64 ~/OneDrive/Робочий стіл/lab1Git (master)
$ git remote add origin https://github.com/PeterLowe/git-one-Ostap-Bo
ychuk.git
```

"git status" is a Git command that provides a summary of the current state of a local Git repository. When you run "git status," Git will display information about which files are tracked or untracked, which changes have been made but not committed, and the current branch you are on. This command is helpful for understanding the status of the project you are doing.



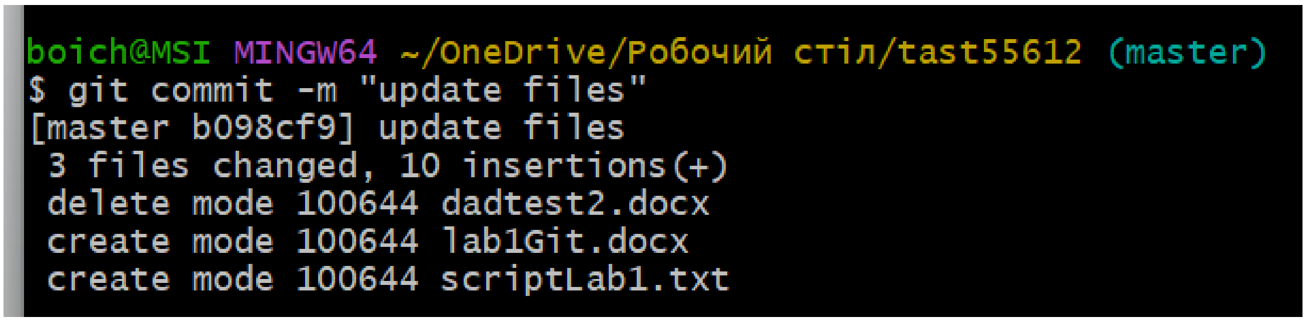
```
boich@MSI MINGW64 ~/OneDrive/Робочий стіл/lab1Git (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        lab1Git.docx
        scriptLab1.txt
        ~$ab1Git.docx
```

"git add ." - is a Git command that stages all changes in the current directory and its subdirectories for the next commit. When you run "git add .", Git will keep track of any new files and stage changes to existing files. It is often used when a developer wants to include all the changes they have made to your project before creating a new commit, saving time and ensuring that all their modifications are included in the future commit.



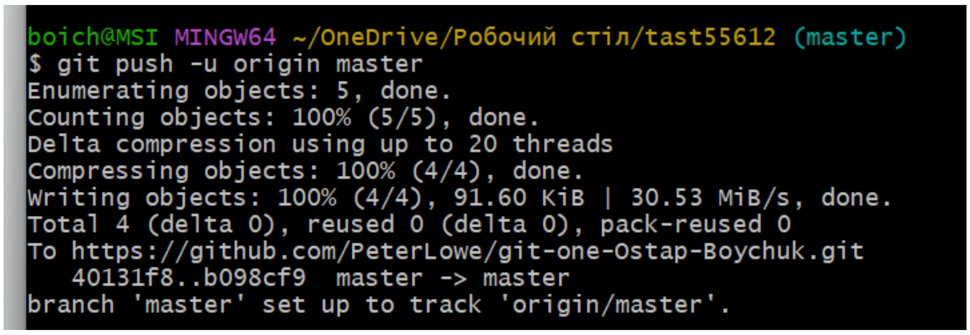
```
boich@MSI MINGW64 ~/OneDrive/Робочий стіл/lab1Git (master)
$ git add .
```

"git commit" is a Git command used to commit changes to your local Git repository. When you run "git commit", you create a snapshot of the changes you made (using "git add") and provide a descriptive entry that explains the purpose or context of the commit. Each commit is a point in the history of your project, making it easier to track changes, collaborate with others, and revert to previous versions if necessary.



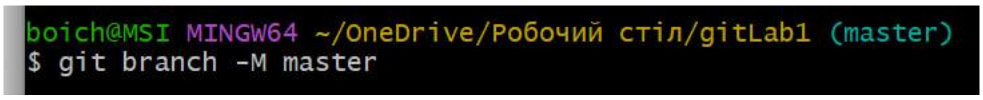
```
boich@MSI MINGW64 ~/OneDrive/Робочий стіл/tast55612 (master)
$ git commit -m "update files"
[master b098cf9] update files
 3 files changed, 10 insertions(+)
 delete mode 100644 dadtest2.docx
 create mode 100644 lab1Git.docx
 create mode 100644 scriptLab1.txt
```

"git push" is a Git command used to push locally committed changes to a remote Git repository. When you run a git push, Git transfers your code, along with its commit history and branches, to a remote server, making it available to others and updating the remote repository with your changes.



```
boich@MSI MINGW64 ~/OneDrive/Робочий стіл/tast55612 (master)
$ git push -u origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 20 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 91.60 KiB | 30.53 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/PeterLowe/git-one-Ostap-Boychuk.git
   40131f8..b098cf9  master -> master
branch 'master' set up to track 'origin/master'.
```

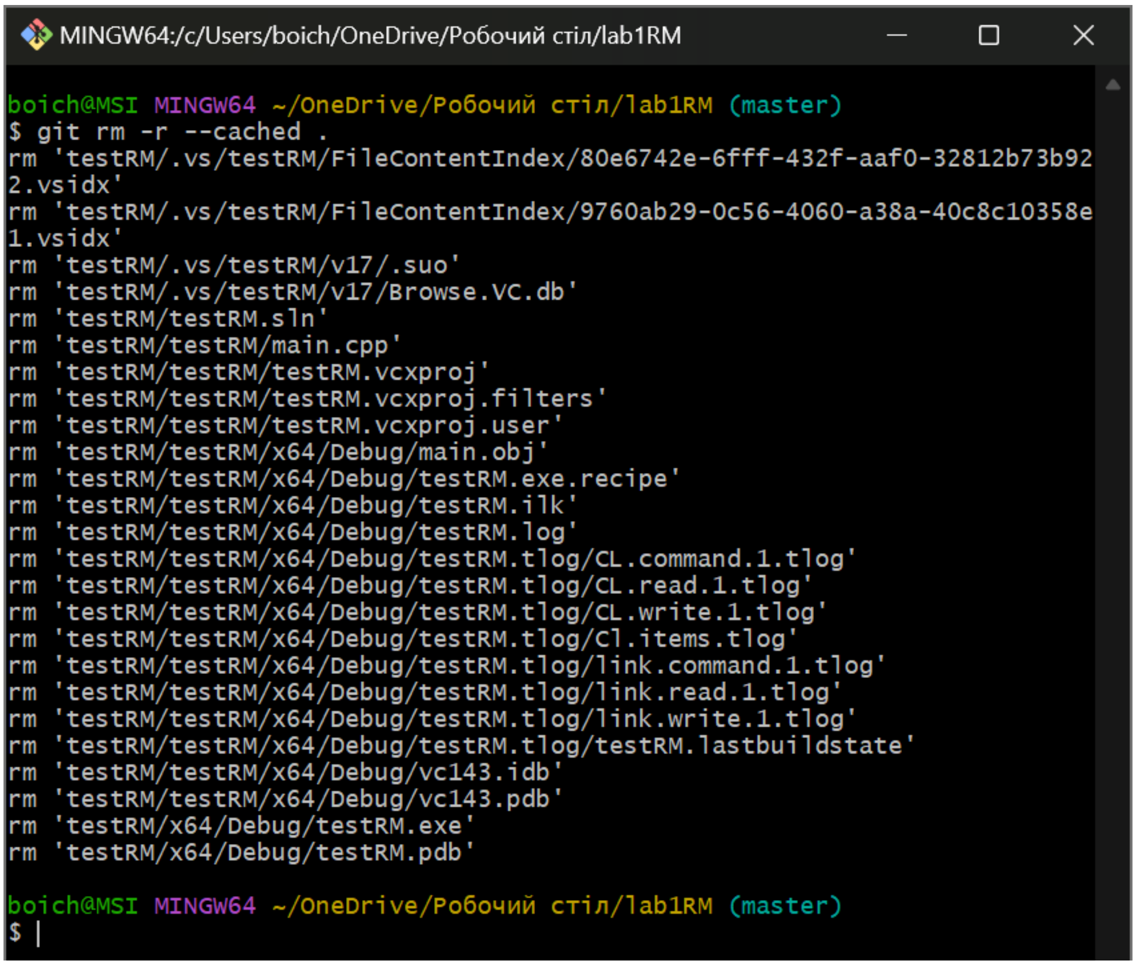
"git branch" is a Git command used to list, create, or delete branches in a Git repository. When you run "git branch" without any arguments, it displays a list of all existing branches, indicating which one is the current branch with an asterisk. You can also use "git branch" to create a new branch by specifying its name, and to delete branches using the "-d" or "-D" flag.



```
boich@MSI MINGW64 ~/OneDrive/Робочий стіл/gitLab1 (master)
$ git branch -M master
```

git rm --cached <file>: This command will prepare the file for removal from the repository, but it will remain in your working directory. This is useful when you want to stop tracking a file but keep it locally.

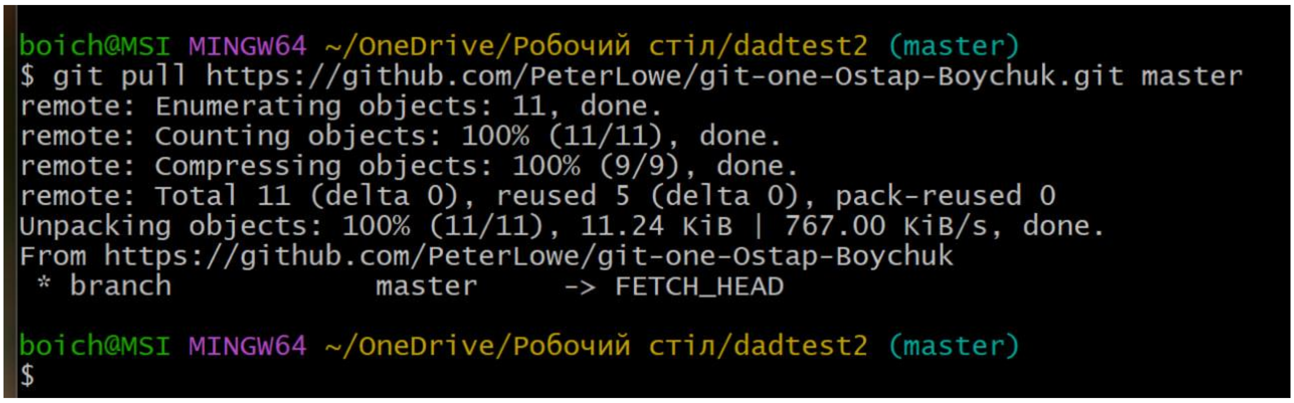
"git rm" helps you manage your project's file structure by removing unnecessary or outdated files from version control.



MINGW64:/c/Users/boich/OneDrive/Робочий стіл/lab1RM

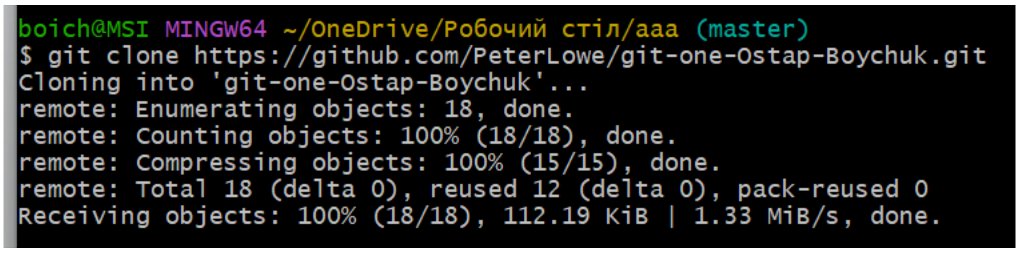
```
boich@MSI MINGW64 ~/OneDrive/Робочий стіл/lab1RM (master)
$ git rm -r --cached .
rm 'testRM/.vs/testRM/FileContentIndex/80e6742e-6fff-432f-aaf0-32812b73b92
2.vsidx'
rm 'testRM/.vs/testRM/FileContentIndex/9760ab29-0c56-4060-a38a-40c8c10358e
1.vsidx'
rm 'testRM/.vs/testRM/v17/.suo'
rm 'testRM/.vs/testRM/v17/Browse.VC.db'
rm 'testRM/testRM.sln'
rm 'testRM/testRM/main.cpp'
rm 'testRM/testRM/testRM.vcxproj'
rm 'testRM/testRM/testRM.vcxproj.filters'
rm 'testRM/testRM/testRM.vcxproj.user'
rm 'testRM/testRM/x64/Debug/main.obj'
rm 'testRM/testRM/x64/Debug/testRM.exe.recipe'
rm 'testRM/testRM/x64/Debug/testRM.ilk'
rm 'testRM/testRM/x64/Debug/testRM.log'
rm 'testRM/testRM/x64/Debug/testRM.tlog/CL.command.1.tlog'
rm 'testRM/testRM/x64/Debug/testRM.tlog/CL.read.1.tlog'
rm 'testRM/testRM/x64/Debug/testRM.tlog/CL.write.1.tlog'
rm 'testRM/testRM/x64/Debug/testRM.tlog/Cl.items.tlog'
rm 'testRM/testRM/x64/Debug/testRM.tlog/link.command.1.tlog'
rm 'testRM/testRM/x64/Debug/testRM.tlog/link.read.1.tlog'
rm 'testRM/testRM/x64/Debug/testRM.tlog/link.write.1.tlog'
rm 'testRM/testRM/x64/Debug/testRM.tlog/testRM.lastbuildstate'
rm 'testRM/testRM/x64/Debug/vc143.idb'
rm 'testRM/testRM/x64/Debug/vc143.pdb'
rm 'testRM/x64/Debug/testRM.exe'
rm 'testRM/x64/Debug/testRM.pdb'

boich@MSI MINGW64 ~/OneDrive/Робочий стіл/lab1RM (master)
$ |
```

"git pull" is a Git command that combines two actions in one step: it pulls changes from a remote repository and then automatically merges those changes into your current branch. This command is useful for updating your local code base in a remote repository, ensuring that you have the latest changes made by others in your project. However, in some cases, this can lead to merge conflicts that will need to be resolved manually.



```
boich@MSI MINGW64 ~/OneDrive/Робочий стіл/dadtest2 (master)
$ git pull https://github.com/PeterLowe/git-one-Ostap-Boychuk.git master
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 11 (delta 0), reused 5 (delta 0), pack-reused 0
Unpacking objects: 100% (11/11), 11.24 KiB | 767.00 KiB/s, done.
From https://github.com/PeterLowe/git-one-Ostap-Boychuk
 * branch            master     -> FETCH_HEAD

boich@MSI MINGW64 ~/OneDrive/Робочий стіл/dadtest2 (master)
$
```

"git clone" is a Git command used to create a copy of a remote Git repository on your local machine. When you run "git clone", it downloads all files, commit history and branches from the remote repository and creates a local copy, allowing you to work with the project's code and history.



```
boich@MSI MINGW64 ~/OneDrive/Робочий стіл/aaa (master)
$ git clone https://github.com/PeterLowe/git-one-Ostap-Boychuk.git
Cloning into 'git-one-Ostap-Boychuk'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 18 (delta 0), reused 12 (delta 0), pack-reused 0
Receiving objects: 100% (18/18), 112.19 KiB | 1.33 MiB/s, done.
```