TASK 5: CREATE A TUTORIAL FOR GIT AND GITHUB WITH THE COMMANDS.

What is Git?

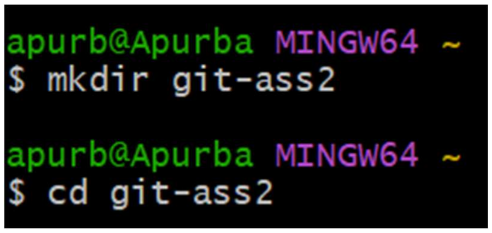
1. Git is a open source tool version control tool that helps developers to manage the code.
2. It is also called a source code Management tool.
3. Git is a version control tool(VCT).

What are Git Command Line?

Git command line tools are command prompt use to interact with Git .They help to create repositories, tract changes ,collaborate with others and manage project history.

Some git commands are-

mkdir
cd
git init
touch
git add
git clone
git status
git log
git commit



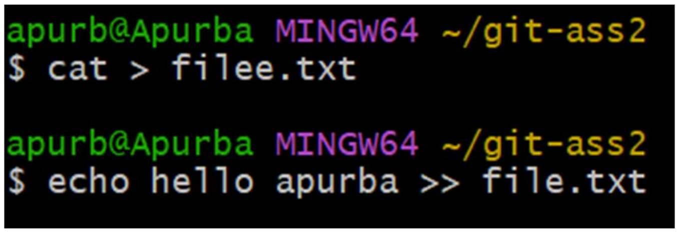
```
apurb@Apurba MINGW64 ~
$ mkdir git-ass2

apurb@Apurba MINGW64 ~
$ cd git-ass2
```

mkdir git-ass2

This command creates a new directory (folder) named **git-ass2**.

cd git-ass2

This command moves you into the **git-ass2** directory, so you can start working inside it.



```
apurb@Apurba MINGW64 ~/git-ass2
$ cat > filee.txt

apurb@Apurba MINGW64 ~/git-ass2
$ echo hello apurba >> file.txt
```

cat > filee.txt

Creates a new file named **filee.txt** and lets you type content into it.

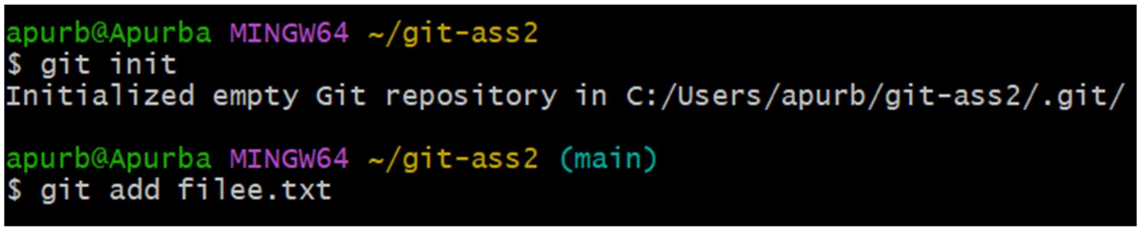
- Pressing Ctrl + D will save and exit.
- Note: This command is slightly unused here since no text is shown being added.

---

echo hello apurba >> file.txt

Adds the text **hello apurba** to a file named **file.txt**.

- >> means **append** (don't overwrite if file exists).
- If file.txt doesn't exist, it will be created.



```
apurb@Apurba MINGW64 ~/git-ass2
$ git init
Initialized empty Git repository in C:/Users/apurb/git-ass2/.git/

apurb@Apurba MINGW64 ~/git-ass2 (main)
$ git add filee.txt
```

git init

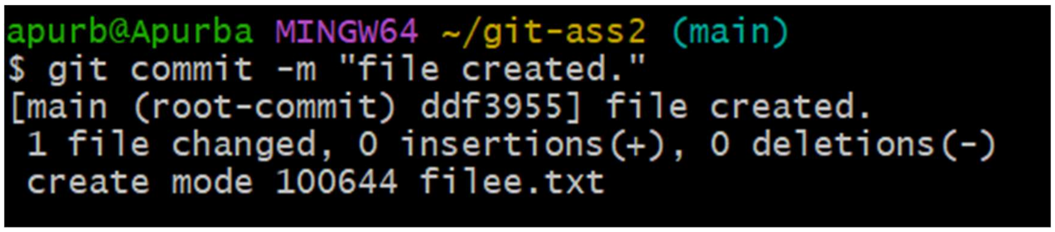
Initializes a new **Git repository** in the current folder (git-ass2).

- It creates a hidden .git folder that tracks version history.

git add filee.txt

Adds the file **filee.txt** to the **staging area**.

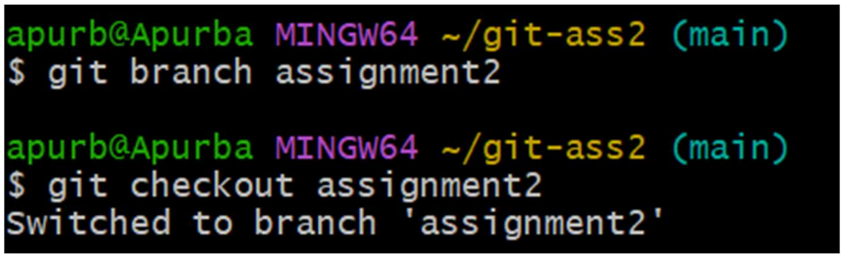
- This means Git is now tracking this file for the next commit.



```
apurb@Apurba MINGW64 ~/git-ass2 (main)
$ git commit -m "file created."
[main (root-commit) ddf3955] file created.
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 filee.txt
```

git commit -m "file created."

- **Purpose**: Commits the staged changes (in this case, a newly created file filee.txt) to the repository with the message "file created.".
- This is the **initial commit** in the repository (shown by root-commit).



```
apurb@Apurba MINGW64 ~/git-ass2 (main)
$ git branch assignment2

apurb@Apurba MINGW64 ~/git-ass2 (main)
$ git checkout assignment2
Switched to branch 'assignment2'
```

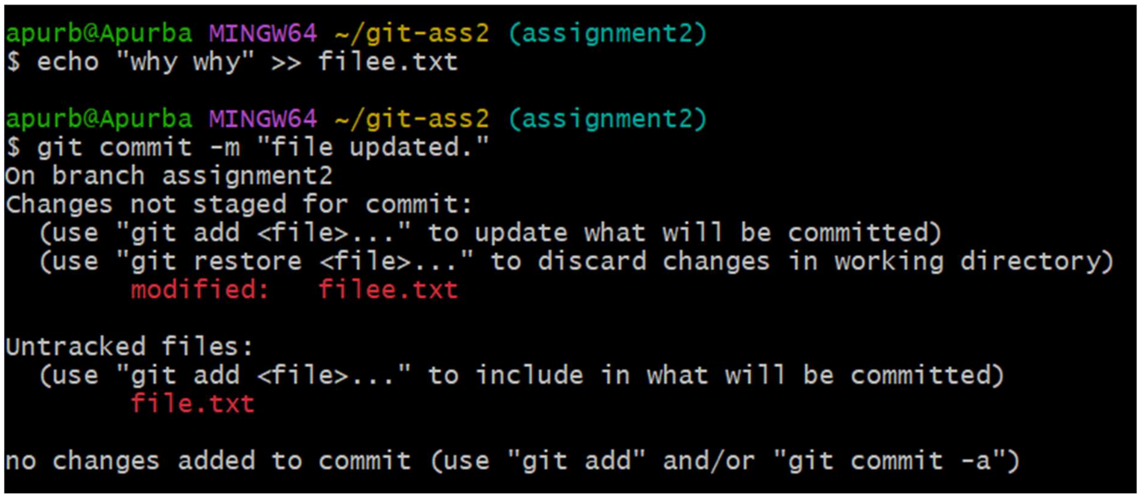
git branch assignment2

- **Purpose**: Creates a new branch named assignment2 from the current branch (main).

---

git checkout assignment2

- **Purpose**: Switches to the assignment2 branch.
- Now any new work will happen in this branch instead of main.



```
apurb@Apurba MINGW64 ~/git-ass2 (assignment2)
$ echo "why why" >> filee.txt

apurb@Apurba MINGW64 ~/git-ass2 (assignment2)
$ git commit -m "file updated."
On branch assignment2
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:   filee.txt

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

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

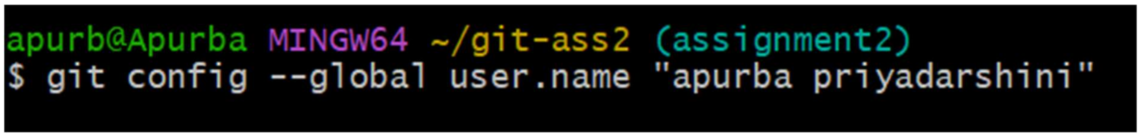
echo "why why" >> filee.txt

- **Purpose**: Appends the text "why why" to the file filee.txt.
- This modifies the file content.

---

git commit -m "file updated."

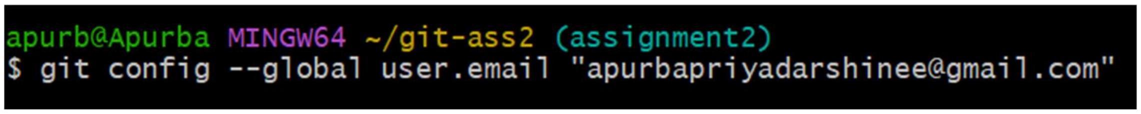
- **Purpose**: Tries to commit changes, but it **fails** because the changes were not staged.



```
apurb@Apurba MINGW64 ~/git-ass2 (assignment2)
$ git config --global user.name "apurba priyadarshini"
```

git config --global user.name "apurba priyadarshini"

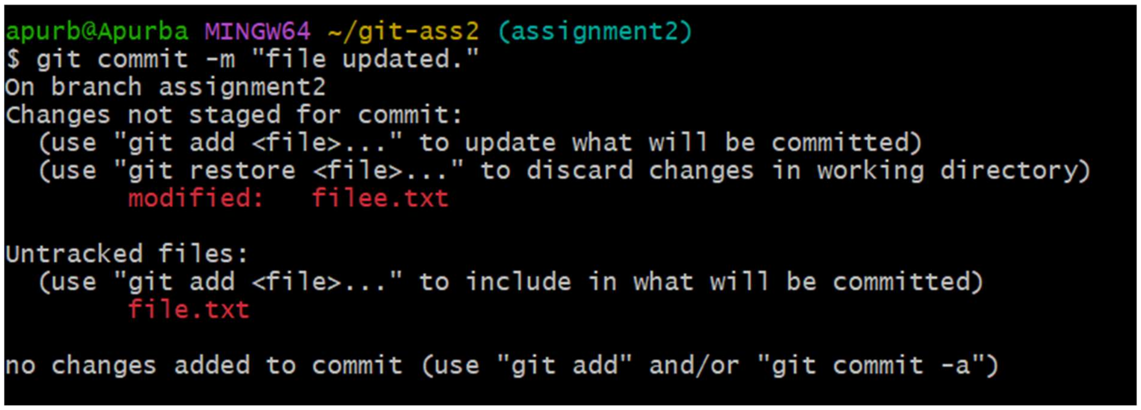
- **Purpose**: Sets the Git global configuration for the user's name.
- This name will be used in all commits made on the system.



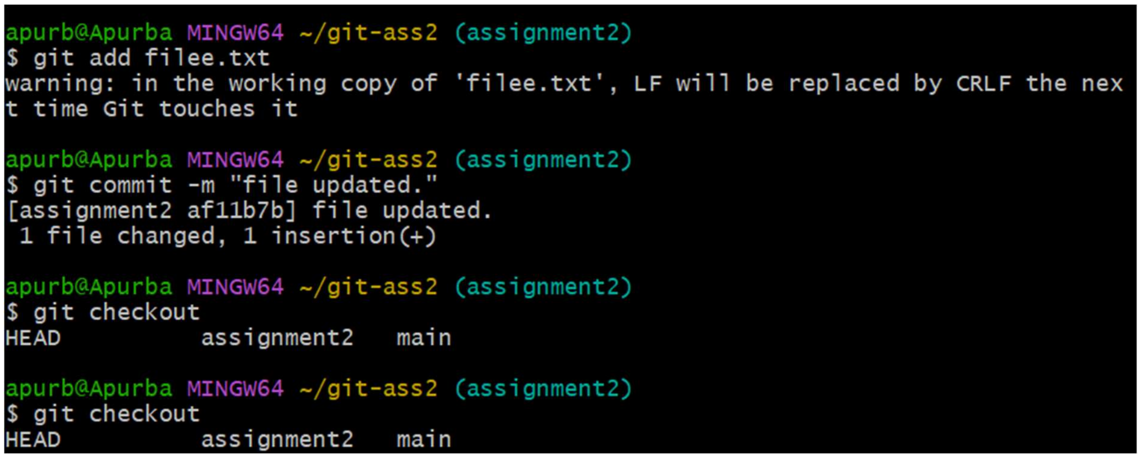
```
apurb@Apurba MINGW64 ~/git-ass2 (assignment2)
$ git config --global user.email "apurbapriyadarshinee@gmail.com"
```

git config --global user.name "apurba priyadarshini"

- **Purpose**: Sets the Git global configuration for the user's name.
- This name will be used in all commits made on the system.



```
apurb@Apurba MINGW64 ~/git-ass2 (assignment2)
$ git commit -m "file updated."
On branch assignment2
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:   filee.txt

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

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



```
apurb@Apurba MINGW64 ~/git-ass2 (assignment2)
$ git add filee.txt
warning: in the working copy of 'filee.txt', LF will be replaced by CRLF the nex
t time Git touches it

apurb@Apurba MINGW64 ~/git-ass2 (assignment2)
$ git commit -m "file updated."
[assignment2 af11b7b] file updated.
 1 file changed, 1 insertion(+)

apurb@Apurba MINGW64 ~/git-ass2 (assignment2)
$ git checkout
HEAD          assignment2   main

apurb@Apurba MINGW64 ~/git-ass2 (assignment2)
$ git checkout
HEAD          assignment2   main
```

git commit -m "file updated."

- Tries to commit changes but fails because filee.txt wasn't staged.

git add filee.txt

- Stages filee.txt so it's ready to be committed.

git commit -m "file updated."

- Successfully commits the staged changes.

git checkout

- Shows available branches (assignment2, main).

- The next command is incomplete (meant to switch branches).

Add files to staging

Add changes in a file to the staging area.

```
$ git add ‹file>
```

Committing Changes

Commit the staged changes to the local repository with a descriptive message.

```
$ git commit -m "message"
```

Display commit history

Display the commit history.

```
$ git log
```

Print working Directory

Pwd command stands for “Print working directory”,displays the full path of your current directory in the terminal code.

```
$ pwd
```

Create a file

This command is used to create file in git bash

```
$ touch
```

Connect to GitHub

Links the local repository to a remote one, usually on GitHub, for pushing and pulling code.

```
$ git remote add origin https://github.com/username/repo-name
```

Create a new branch

Create a new branch from the current HEAD, Useful for feature development without affecting the main code.

```
$ git branch <new-branch>
```

Checkout

Switches the working directory to the specified branch.

```
$ git checkout <new-branch>
```

Merge Branches

Combines the specified branch into the current branch, integrating changes made in the other branch.

```
$ Git merge new-branch
```

Delete Branches

To delete a branch locally

This will delete the branch only if it has already been fully merged .

```
$ git branch -d branch_name
```

Lists all the files

Lists all tracked files in the current repository

```
$ git ls-files
```