Setting up directory/folder

When we use git it is a common practice to create a separate directory to work with and then initialize git in that directory.

To create the working directory you can use the GUI tools (right click -> create folder) and then open that directory in the terminal (git bash for windows). To open the directory in the terminal you can (right click in the folder -> open in terminal OR git bash here).

Alternatively, you can use linux commands to create the desired file structure as follows -

Open the terminal and check the current working directory. You can do this by the **pwd** command.

```
➜  ~ pwd
/home/kanak
➜  ~
```

You can also check the current directory with the terminal display.

```
➜  ~ cd Desktop
➜  Desktop
```

The terminal displays the name of the current directory as shown.

Now I want to create a **demo** folder in the **Desktop/teaching** folder.
This can be done by -

```
➜  ~ cd Desktop/teaching
➜  teaching mkdir demo
➜  teaching cd demo
➜  demo
```

> **NOTE -**
> These commands will work if you have a folder named ‘teaching’ on your ‘Desktop’. You can change the name of the folders as you wish.

Once we are in the **demo** folder we can start working with git.

Git Commands

Initialize git

To use git commands in a directory and manage and track files we need to initialize git in that directory.
This is done by **git init**.

This command will create a .git folder in your directory. Since this is a hidden folder(starting with .) you cannot see it normally. You can see hidden files in GUI using ctrl + h shortcut or you can use **ls -a** command to see the hidden files in CLI.

```
→ demo ls -a
.  ..
→ demo git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/kanak/Desktop/teaching/demo/.git/
→ demo git:(master) ls -a
.  ..  .git
→ demo git:(master)
```

Adding files to staging area

1. Before adding files to the staging area we need to create them.

   Creating files -

```
→ demo git:(master) touch a.txt
→ demo git:(master) ✗ touch b.txt
→ demo git:(master) ✗ touch c.txt
→ demo git:(master) ✗ ls
a.txt  b.txt  c.txt
→ demo git:(master) ✗
```

2. Before adding the files we should check the current status of the staging area. This is done by **git status** command.

```
→ demo git:(master) ✗ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
→ demo git:(master) ✗
```

3. We see there are some untracked files in the staging area. We can add them using the **git add <filename>** command.

> **NOTE -** Use **git add .** to add all the untracked files to the staging area at once.

Add the files and run git status again to check the status of the staging area.

```
→ demo git:(master) ✗ git add a.txt
→ demo git:(master) ✗ git add .
→ demo git:(master) ✗ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   a.txt
        new file:   b.txt
        new file:   c.txt

→ demo git:(master) ✗
```

Removing files from staging area

1. We can remove unwanted files from the staging area using the **git rm --cached <filename>** command.

```
→ demo git:(master) ✗ git rm --cached a.txt
rm 'a.txt'
→ demo git:(master) ✗ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   b.txt
        new file:   c.txt

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

→ demo git:(master) ✗
```

Committing files

1. Committing files basically creates a snapshot of the staged files at that point of time. We can commit all the tracked files of the staging area with the help of **git commit -m "[commit message]"** command.

   In the [commit message] argument you pass meaningful messages that allow others to understand the changes you have done in the repository.

```
→ demo git:(master) ✗ git commit -m "initial commit"
[master (root-commit) 5761360] initial commit
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 b.txt
 create mode 100644 c.txt
→ demo git:(master) ✗
```

2. If you are committing files on your system for the FIRST TIME then you will need to configure your email address and name otherwise you will receive an **Author identity unknown** error.
   This can be resolved by -

   git config --global user.email "you@email.com"
   git config --global user.name "Your Name"

> **NOTE -**
> The email is the one linked with your github account and your name is the name you have in your github account (not to be confused with your username).

3. You can check your commit history using the **git log** command.

```
→ demo git:(master) ✗ git log
→ demo git:(master) ✗
```

```
commit 5761360540aeb97aa7bfdf3385f694cdadc59990 (HEAD -> master)
Author: Kanak Tanwar <kanaktanwarpro@gmail.com>
Date:   Wed Feb 21 10:16:07 2024 +0530

    initial commit
(END)
```