Installing Git

Git isn't usually set up by default on your computer, so you need to install and configure it before you can start using it to manage your code.

It's important to keep Git up to date, just like all the other software on your machine. Updates protect you from security vulnerabilities, fix bugs, and give you access to new features.

The recommended method of installing and maintaining Git is provided for three major platforms below:

Windows

Download and install Git for Windows. Once installed, you'll be able to use Git from the command prompt or PowerShell. We recommend that you stick with the defaults selected during the installation unless you have a good reason to change them.

Linux

On the terminal, just run `sudo apt install git-all`.

Mac

The best thing to do is to install Homebrew and then from the terminal run the command brew install git.

Start Using Git

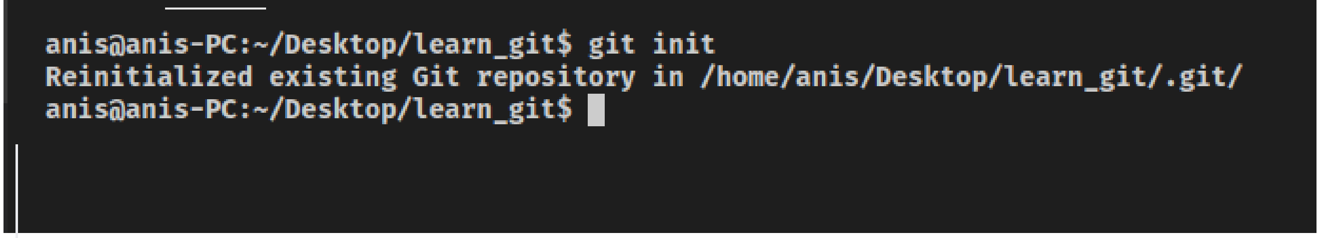
Once we have Git installed we need to "initialize" a repository before we can start using it.

It's very easy to do, just follow my lead:

1. Access the wanted folder using the terminal (prompt cmd).
2. To initialize a repository, we only have to run:

```
git init
```

Output



```
anis@anis-PC:~/Desktop/learn_git$ git init
Reinitialized existing Git repository in /home/anis/Desktop/learn_git/.git/
anis@anis-PC:~/Desktop/learn_git$ 
```

P.S : This command just created a hidden folder called .git, where all the magic happens.

Adding and Committing Files

Remember that folder we've created? It's time to use it. We'll be creating a file and placing it in

there

Feel free to add whatever you want!

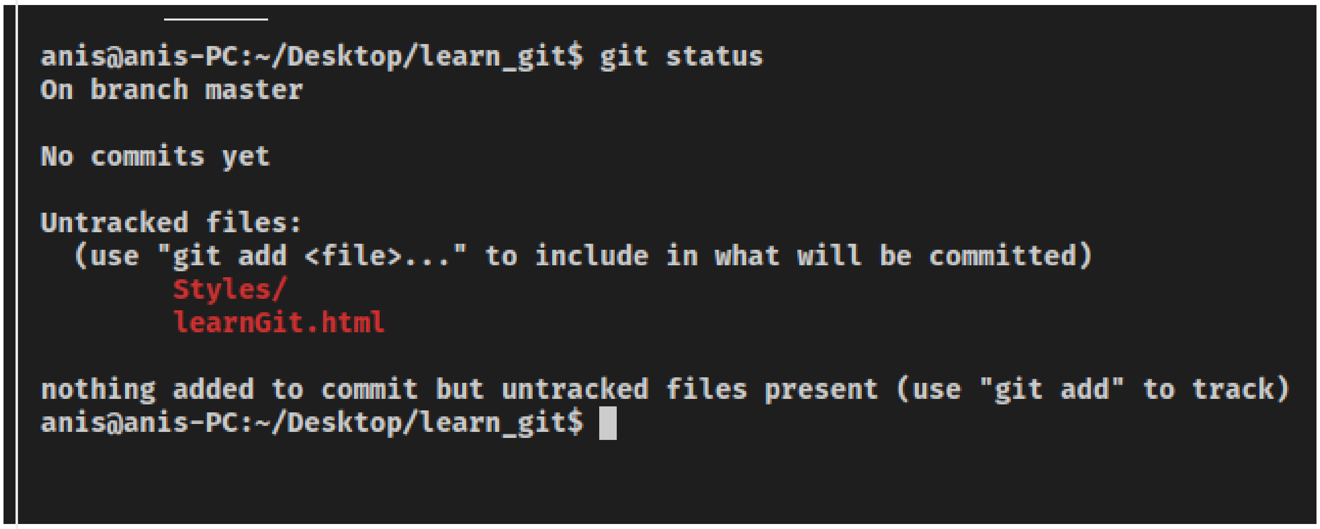
Check the Status of Your Repository

Now that we have a few files in our repository, let's see how Git processes them.

To check the current status of your repository we use the `git status` command.

```
git status
```

Output



```
anis@anis-PC:~/Desktop/learn_git$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
anis@anis-PC:~/Desktop/learn_git$ 
```

Adding Files for Git to Track

At this point, we do not have any files for Git to track.

We need to add files specifically to Git in order to tell Git to track them. We add files using the

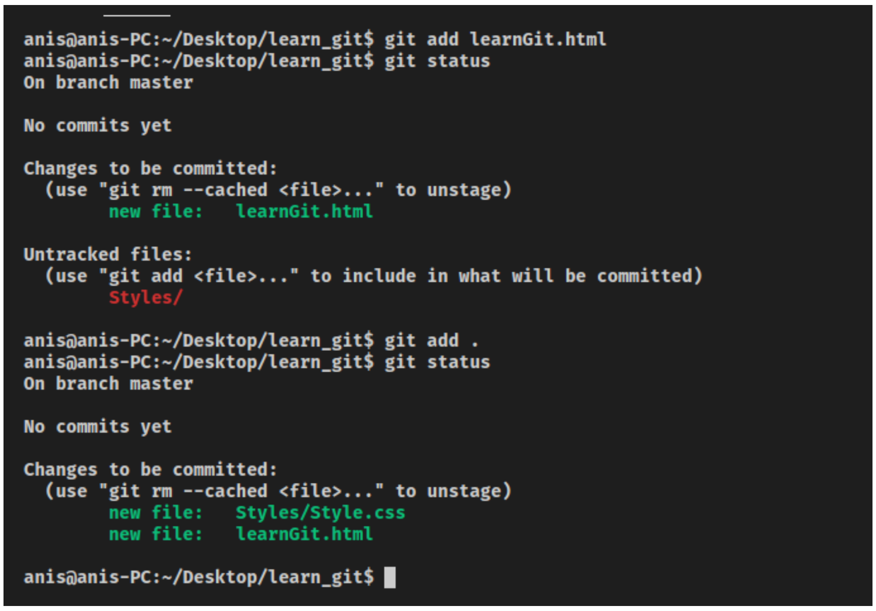
"add" command.

After running `git add .` Git will add all the repository files to an intermediate area called the

staging area. We can also add what we want simply by running `git add myFileName`

```
git add .
```

Output



```
anis@anis-PC:~/Desktop/learn_git$ git add learnGit.html
anis@anis-PC:~/Desktop/learn_git$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   learnGit.html

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

anis@anis-PC:~/Desktop/learn_git$ git add .
anis@anis-PC:~/Desktop/learn_git$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Styles/Style.css
        new file:   learnGit.html

anis@anis-PC:~/Desktop/learn_git$ 
```

Removing Files

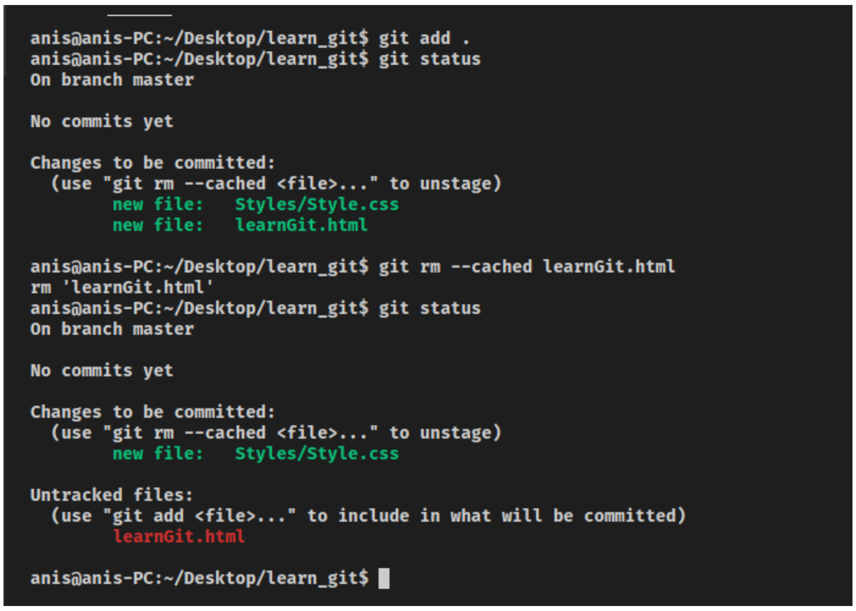
Let's say that you have added files to Git and you do not want it to track any of them.

In a situation like this you have to tell Git to stop tracking them.

However, running a simple `git rm` command will not only remove it from Git, but will also remove it from your local file system as well! To tell Git to stop tracking a file while still keep it in your local system, run the following command:

```
git rm --cached [file_name]
```

Output



```
anis@anis-PC:~/Desktop/learn_git$ git add .
anis@anis-PC:~/Desktop/learn_git$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Styles/Style.css
        new file:   learnGit.html

anis@anis-PC:~/Desktop/learn_git$ git rm --cached learnGit.html
rm 'learnGit.html'
anis@anis-PC:~/Desktop/learn_git$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Styles/Style.css

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

anis@anis-PC:~/Desktop/learn_git$ 
```

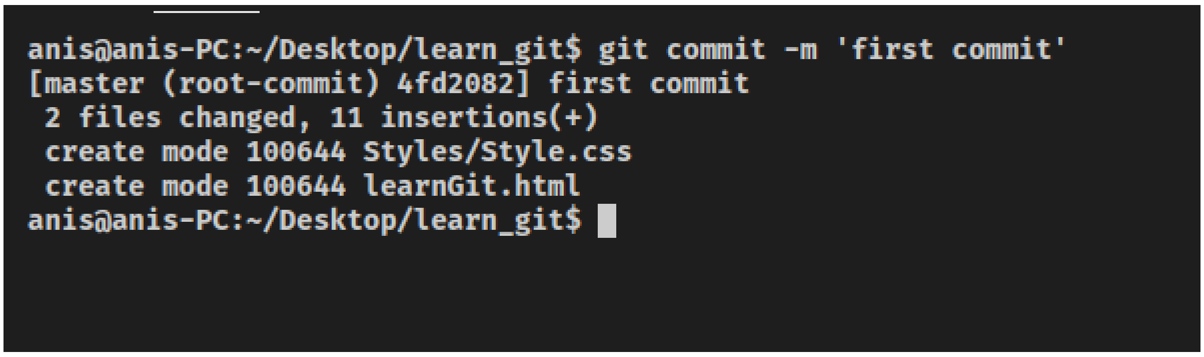
Committing Changes

Once you have staged your files, you can commit them into Git.

Imagine a commit command as a snapshot at a certain point and time where you can return back to access your repository at that stage. You assign a commit message to every commit, which you can pass with the -m prefix.

```
git commit -m 'first commit'
```

Output



```
anis@anis-PC:~/Desktop/learn_git$ git commit -m 'first commit'
[master (root-commit) 4fd2082] first commit
 2 files changed, 11 insertions(+)
 create mode 100644 Styles/Style.css
 create mode 100644 learnGit.html
anis@anis-PC:~/Desktop/learn_git$
```