# iLecture series: Creating your Local Repository (Git Bash)

#Let's create a Project on your desktop called "Urban data"

```
use@DESKTOP-S6MSCKF MINGW64 ~
$ pwd
/c/Users/use

use@DESKTOP-S6MSCKF MINGW64 ~
$ cd Desktop

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop
$ mkdir "Urban data"
```

#Without the quotation marks, " ", you will have two different folders/projects

#You can right click on the folder and choose Git Bash to access the folder directly or use a CLI instead

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop
$ cd "Urban data"
```

#Let's create 3 empty files of different formats in our project using the CL Touch

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data
$ touch python.html

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data
$ touch r.js

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data
$ touch scala.docx
```

#Let's create a new subdirectory named *.git* that will contain all of our necessary repository files.

#Remember again that a repository is a directory or storage space where all of your projects files can live, which can be either **local (Git Bash)** or **Remote (GitHub).**

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data
$ git init

Initialized empty Git repository in C:/Users/use/Desktop/Urban data/.git/
```

#This command just created a new *.git* repository

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$
```

#For now, the branch is always "master", which is the default.
#Will show you how to Branch later in the course.

#if you are using a new window, your hidden files may not have b
een **Enabled** and so you may not see the *.git* repository created.

#To *Enable*, Go to:
-File Explorer > View > options > Change folder and search optio
n > View > Show hidden file

#Also *uncheck* "Hide extensions for known file types" because you want the extensions to be shown all the time. Click Apply.

## Configure Username and Email

#Each commit to a Git Repository will be “tagged” with the username of the person who made the commit.

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git config --global user.name 'felixemekaanyiam'

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git config --global user.email 'felix.emeka.anyiam@gmail.com'
```

## Adding Files to Your Git repository

#Let’s Add the python html and the scala Word files to our *Git* repository

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git add python.html

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git add scala.docx
```

## Checking your Git Status to see what is in the Staging Area

#At this time, we are not told anything when we add files but if we want to check to see what is in the staging area, we use the CL `git status`:

#Remember that the *staging area* is an ‘Index’ or simply a *file* that stores the modified information you want to commit

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git status
On branch master

No commits yet

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

        new file:   scala.docx
        new file:   python.html

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

        r.js
```

## To *remove* a file from the Staging Area

#Let’s remove python file from the staging Area

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git rm --cached python.html
rm 'index.html'
```

#To confirm that the file is removed from the staging area, let's recheck git status

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git status
On branch master

No commits yet

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

        new file:   scala.docx

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

        r.js
        python.html
```

## To *add* a file to the Staging Area

#Let's add the python file back to the staging Area

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git add python.html

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git status
On branch master

No commits yet

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

        new file:   scala.docx
        new file:   python.html

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

        r.js
```

#To add all available html files to the staging Area, use the CLI `git add *.html`

#To add all available files irrespective of the file format to

the staging Area, use the CLI git add .

#Let's remove the python and scala files and add everything automatically, including the Java script file

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git rm --cached python.html scala.docx
rm 'scala.docx'
rm 'python.html'

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git add .

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git status
On branch master

No commits yet

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

        new file:   scala.docx
        new file:   r.js
        new file:   python.html
```

## To *commit* a file from the Staging Area to the Local Repository

#First lets edit the scala file and then check the current status

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/my data (master)
$ git status
On branch master

No commits yet

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

        new file:   scala.docx
        new file:   r.js
        new file:   python.html

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   scala.docx
```

#Taking the new modified file to the staging area:

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git add .

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git status
On branch master

No commits yet

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

        new file:   scala.docx
        new file:   r.js
        new file:   python.html
```

#Lets commit

```
$ git commit
```

```
Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   scala.docx
#       new file:   r.js
#       new file:   python.html
#
~
~
~
~
~
~
~
~
~
<sers/use/Desktop/my data/.git/COMMIT_EDITMSG [unix] (10:14 21/04/2019)1,0-1 All
"C:/Users/use/Desktop/my data/.git/COMMIT_EDITMSG" [unix] 13L, 278C
```

#We can see that the default commit message contains the latest output of the git status command committed out.

#The comments help us remember what we're committing

#We will not be able to type the comment and so we click "I" to go into the insert mode

#Type "Initial commit"

#Let's press the "esc" key to take us out of Insert mode and then we type :wq

#Those files have been committed and it tells us how many files have changed and also gives us the file names

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git commit
[master (root-commit) f825110] Initial commit
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Urban.docx
 create mode 100644 app.js
 create mode 100644 index.html
```

#When we do git status again, it says nothing to commit as we have committed all of our changes.

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git status
On branch master
nothing to commit, working tree clean
```

#Let's edit the scala file again

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   scala.docx

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

#Lets add it back

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/my data (master)
$ git add .

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/my data (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   scala.docx
```

#Let's do a commit in a different style that skips the whole insert stage by typing the commit message in line with the commit command with an -m flag:

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git commit -m 'Changed scala.docx'
[master 1828b6a] Changed Urban.docx
 1 file changed, 0 insertions(+), 0 deletions(-)
```

#Let's clear everything

```
$ clear
```

## Creating Branch from the Master

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git branch gis

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git status
On branch master
nothing to commit, working tree clean
```

# doing this does not take us to the branch as our git status is still showing "on branch master." To switch we say:

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git checkout gis
Switched to branch 'gis'

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (gis)
```

#Now we are in the gis branch

#Let's create 2 new files in the branch called Introduction.docx and login.html:

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (gis)
$ touch introduction.docx login.html
```

#Let's confirm that the files created are in the folder

#Let’s add the files to the staging area

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (login)
$ git add .
```

#Let’s confirm the status of our files in the staging area

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (gis)
$ git status
On branch gis
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   introduction.docx
        new file:   login.html
```

#Let’s commit

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (login)
$ git commit
```

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch gis
# Changes to be committed:
#       new file:   introduction.docx
#       new file:   login.html
#
~
~
~
~
~
~
~
~
~
~
~
~
~
</Users/use/Desktop/my data/.git/COMMIT_EDITMSG [unix] (12:05 21/04/2019)1,1 All
-- INSERT --I “Changes to be committed.”
```

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (gis)
$ git commit
[gis e7c761b] Changes to be committed
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 introduction.docx
 create mode 100644 login.html
```

## Switching Back from the Branch to the Master

#Let's switch back to our Master

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (gis)
$ git checkout master
Switched to branch 'master'

use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$
```

#If you look into the folder, the two new files created are gone. The reason for that is because that's in the login branch

#Now if we finish the functionality and we are ready to merge, then we can use the command while we are in the master

```
use@DESKTOP-S6MSCKF MINGW64 ~/Desktop/Urban data (master)
$ git merge gis
Updating 1828b6a..e7c761b
Fast-forward
 introduction.docx | 0
 login.html        | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 introduction.docx
 create mode 100644 login.html
```

#Even though we are in the master, we can see the login.html

### The End to creating a Local Repository