Getting Set Up and Your First Repository

Now that you have installed Git, let's make our first git repository. Open your terminal and navigate to the desktop.

```
cd ~/Desktop
```

Once there, create a new folder.

```
mkdir git_test
```

Let's change directories into the folder.

```
cd git_test
```

Now that we are in here, let's initialize a git repository.

```
git init
```

This creates a folder labelled `.git` in the current directory. **NEVER EDIT THIS FOLDER MANUALLY** as you will get yourself in a lot of trouble. Just be aware that it is there and that it stores all the information about this repository. If you delete the project folder, you delete the version information as well.

Now that we have created that folder, let's start developing a project.

Start by running

```
touch name_game.py
touch ignore_me.txt
```

This will create a Python file and a text file in the current directory. Now you will want to edit them. Open the Python file in the editor of your choice.

In the Python file, let's start writing a program.

```
name = input("Enter your name: ")
bname = "B" + name[1:]
print(name + ", " + name + ", bo " + bname)
```

Let's make sure that it works. We can do that by executing the Python file from the command line. Run

```
python3 name_game.py
```

Now that we have made those edits, let's return to the command line and commit those changes. First we will run

```
git status
```

and you will get a print out such as

```
On branch master

Initial commit

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

    name_game.py
    ignore_me.txt

nothing added to commit but untracked files present (use "git add" to track)
```

At this point we have our base git repository with no files being tracked. In order for git to track changes, we must add a file so it knows to track changes to this file. We do that with `git add FILENAME`. You can also add folders instead of specific files.

Let's add our `name_game.py` file.

```
git add name_game.py
```

Now run git status again, and you can see that things have changed.

```
git status

On branch master

Initial commit

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

    new file:   name_game.py

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

    ignore_me.txt
```

Now our `name_game.py` file is `staged` or `in staging` and ready to be committed. Git has not started tracking the file changes yet; it just knows that we are preparing to commit that file to our formal history.

Now let's do that. We can do that with

```
git commit
```

This will bring up a prompt that will ask us for our commit details, which is a message that we want to leave to others recording the changes that we made. In the prompt, write "My first commit".

Note: On many computers, the default text editor is one called vi. If the prompt you see is vi, you will need to press the `i` key before you can type.

```
Added name_game.py file.
```

If you are in vi, press Escape and then type `:wq`, which stands for `write quit`. That will write the changes, making them permanent.

We will get some feedback telling us something such as

```
[master (root-commit) 1e73b2f] test
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hello.py
```

Now we have permanently saved this version of name_game.py to our history. With git you can choose to commit whole files or just lines in files to the history; for the most part, individual files are granular enough, but it is a good feature to be aware of.

Let's try to improve our program and make a new version. Open your text editor again, and make the program print another line.

```
name = input("Enter your name: ")
bname = "B" + name[1:]
print(name + ", " + name + ", bo " + bname)
fname = "F" + name[1:]
print("Banana Fana Fo "+ fname)
```

Run git status again to see that the file is listed as modified. We are ready to add it to the staging area again.

```
git add name_game.py
```

Let's commit the changes.

```
git commit -m "Added second line of song"
```

You will notice that I used the `-m` flag; this means that we are going to just append the message as a text string right after. This greatly simplifies the way in which you can add messages. However, if you are writing a longer one, the `git commit` command allows you to do so easily.

Now all that remains is our ignore file. This is an example of a file that we do not want to track changes to, even if it changes. This may be a file with some notes in it that you do not need to track officially, even if they change. Git can be ordered to ignore certain files or folders; this is done with a plain text file called a `.gitignore` file.

Let's create that with the command line.

```
touch .gitignore
```

Typically files that start with . are hidden on Unix systems, so this may not appear in your folder view. However, we can still edit it. We are going to edit it with the `echo` command from the command line.

Run

```
echo "ignore_me.txt" >> .gitignore
```

This will append the quoted text "ignore_me.txt" to our `.gitignore` file. Now when we run

```
git status
```

we will see

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

    .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```

Our ignore file has disappeared. It disappears because git reads the .gitignore file and ignores any file names or folder names that it finds there. However, we now need to add the .gitignore file so that git can keep track of the files that we want to ignore.

Let's do that now.

```
git add .gitignore
```

Run git status to see how it has moved to staging. At this point we are going to commit that change. Once again, we will use the -m flag to speed things up.

```
git commit -m "Added gitignore"
```

Now you have learned the basics of git on your local machine. This has taught you a lot about running a local git repository. Of course, there is plenty more to learn, but that is a good start.

References:

- Git Website
- Git Cheatsheet
- Better Explained